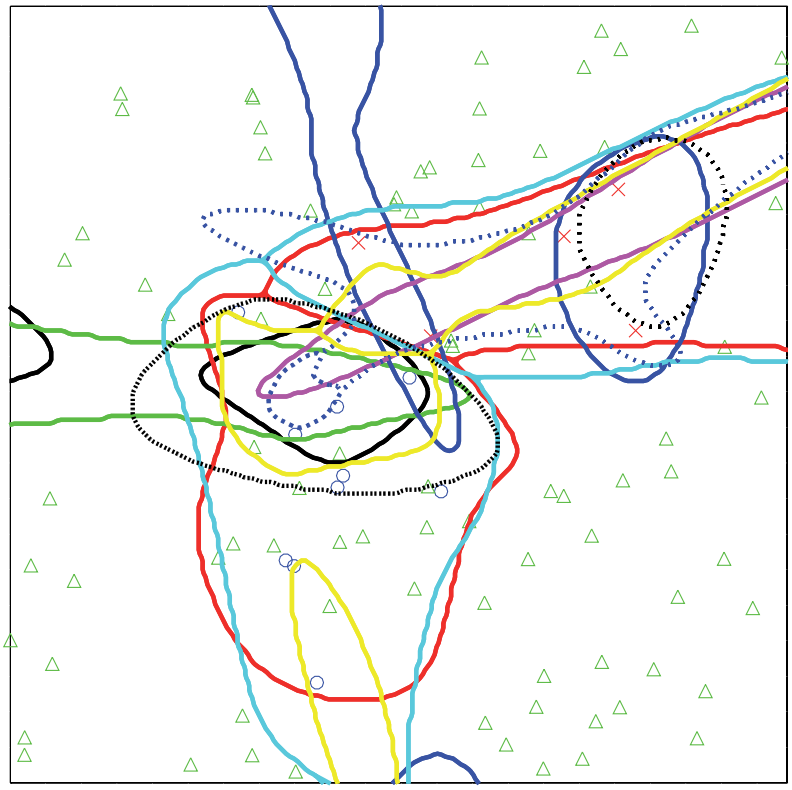
JYVÄSKYLÄ STUDIES IN COMPUTING
247

Paavo Nieminen

Multilayer Perceptron Training with Multiobjective Memetic Optimization



JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 247

Paavo Nieminen

Multilayer Perceptron Training with Multiobjective Memetic Optimization

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella julkisesti tarkastettavaksi yliopiston Mattilanniemen A-rakennuksen salissa MaA103 joulukuun 9. päivänä 2016 kello 12.

Academic dissertation to be publicly discussed, by permission of the Faculty of Information Technology of the University of Jyväskylä, in Mattilanniemi, auditorium MaA103, on December 9, 2016 at 12 o'clock noon.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2016

Multilayer Perceptron Training with Multiobjective Memetic Optimization

JYVÄSKYLÄ STUDIES IN COMPUTING 247

Paavo Nieminen

Multilayer Perceptron Training with Multiobjective Memetic Optimization

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2016

Editors
Timo Männikkö
Department of Mathematical Information Technology, University of Jyväskylä
Pekka Olsbo, Ville Korkiakangas
Publishing Unit, University Library of Jyväskylä

URN:ISBN:978-951-39-6824-3
ISBN 978-951-39-6824-3 (PDF)

ISBN 978-951-39-6823-6 (nid.)
ISSN 1456-5390

Jyväskylä University Printing House, Jyväskylä 2016

ABSTRACT

Nieminen, Paavo
Multilayer Perceptron Training with Multiobjective Memetic Optimization
Jyväskylä: University of Jyväskylä, 2016, 151 p.
(Jyväskylä Studies in Computing
ISSN 1456-5390; 247)
ISBN 978-951-39-6823-6 (nid.)
ISBN 978-951-39-6824-3 (PDF)
Finnish summary
Diss.

Machine learning tasks usually come with several mutually conflicting objectives. One example is the simplicity of the learning device contrasted with the accuracy of its performance after learning. Another common example is the trade-off that must often be made between the rate of false positive and false negative predictions in diagnostic applications. For computer programs that learn from data, these objectives are formulated as mathematical functions, each of which describes one facet of the desired learning outcome. Even functions that intend to optimize the same facet may behave in a subtly different and mutually conflicting way, depending on the task and the dataset being examined. Multiobjective optimization methods developed for simultaneous optimization of such multiple objectives found their way to machine learning a few decades ago.

This dissertation discusses the past and current uses of multiobjective optimization in supervised learning, especially in training a multilayer perceptron (MLP) artificial neural network for object classification. A literature overview of multiobjective MLP training is presented, supported by a semi-automatic survey using a software tool created partly by the author. Based on the literature, key goals and algorithmic elements are identified and applied to create a new framework for training MLPs consistent with an implementation used earlier for industrial projects using single-objective methods. Simulated datasets are used to illustrate the functionality of the created training algorithm, and how memetic Pareto-based multiobjective learning can be used for MLP classifier training. Emphasis is put on formulating useful representations and objective functions for the task.

Keywords: Machine Learning, Neural Networks, Memetic Algorithms, Multiobjective Optimization, Multilayer Perceptron, Classification Algorithms

Author

Paavo Nieminen
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Supervisor

Professor Tommi Kärkkäinen
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Reviewers

Professor João Paulo Pordeus Gomes
Departamento de Computação
Universidade Federal do Ceará, Fortaleza
Brazil

Senior Lecturer Jonathan Fieldsend
College of Engineering, Mathematics and Physical Sciences
University of Exeter
United Kingdom

Opponent

Associate Professor Carlos Cotta
Dept. de Lenguajes y Ciencias de la Computación
Universidad de Málaga
Spain

PREFACE

This work was under preparation for many years, while most of my time was spent in teaching, coordinating the studies of our master's degree students, and running around in various other departmental activities that were totally unrelated to research. At least for the standards of today, it took far too long for me to get to the end of my doctoral studies. I am grateful to the Department and the Faculty for having faith in me and for having a lot of patience during their long wait. Nevertheless, I am even more grateful and probably indebted for the rest of my life for having had the possibility of being involved in those wonderful other tasks. I sincerely feel that my ten years at the department have not been wasted, but used rather well in becoming educated not only as a researcher but also as a teacher, project worker, curriculum designer, community outreacher and what not. Good times, indeed.

The research itself eventually feels like a very logical step in the narrow and twisty avenue of research and development duties I have been involved in. From multiobjective optimization back to applying multiobjective optimization once again, and now in a way that I strongly feel would benefit future research in machine learning, which has made up most of the twists in my avenue. The scientific contribution between the covers of this dissertation may be modest, but I hope I have been able to write a clear exposition on a topic that I genuinely feel is current and worthwhile to the community.

ACKNOWLEDGEMENTS

I would like to express my utmost gratitude to Tommi Kärkkäinen for supervising my PhD studies from the start to the finish. The process did have its twists and it certainly took a longer time than anticipated. His patience as the supervisor must have been stretched to the limit at times. For the years of patience and support, I must praise Tommi.

I am most grateful to João Paulo Pordeus Gomes and Jonathan Fieldsend for reviewing and evaluating my manuscript. Their comments were well-founded and justified. For the final version, I tried my best, within the time allotted, to incorporate as many changes as possible from those suggested. I am very happy that my opponent in the public defence of this dissertation will be Carlos Cotta, one of the editors of the fundamental textbook I cite. I am indebted to Pekka Neittaanmäki as the dean of the Faculty of Information Technology and Tuomo Rossi as the head of the Department of Mathematical Information Technology for their support and their faith in the actualization of this dissertation.

For their friendship and support, I want to express my gratitude to all of my dear friends and colleagues over the years. There was a list of names here, but I deleted it. It was too long and too short at the same time, so I decided it must not exist.

Linkki Jyväskylä ry (the "subject union" of our students) has provided a lot of sauna and support over the years. I am somewhat humbled by the students' own initiative to make my public defence an official event of the union. If I am not mistaken, this is "a first". I hope it will become a tradition for the students to go and watch a doctoral defence together at least once a year.

I thank my parents, Seppo and Helka, for bringing me up in an environment that allowed me to choose my own path in life. I thank Eeva for being such a good companion to my dad after Helka's passing. Finally, I thank, with love, Piia for bearing with me, and feeding me while finalizing this work.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

This dissertation examines how supervised machine learning tasks, especially that of classifying objects based on measured features using feedforward neural networks, such as the multilayer perceptron, can be formulated as multiobjective optimization problems and solved using multiobjective memetic algorithms.

Most of machine learning has always been intrinsically at least bi-objective: There is usually a trade-off between the accuracy of the resulting model and its simplicity, related both to the interpretability of the model and its capability to generalize to data outside the set of examples used in the learning process. Also, in binary classification tasks, e.g., medical diagnosis or fault detection, there is a trade-off between the sensitivity (true positive rate) and the specificity (true negative rate) of the classifier. If these two bi-objective considerations are to be combined together, or if classification tasks have more than two classes, and we are interested in controlling classwise classification accuracies in addition to the overall performance, the resulting problem becomes truly multiobjective in nature, i.e., there are at least three mutually conflicting goals that cannot be optimally satisfied by any single instance of a learning model.

In multiobjective machine learning, the task of the person selecting a model, called the Decision Maker in the multiobjective optimization or decision making parlance, becomes that of finding a suitable compromise solution within a so-called non-dominated set (or Pareto optimal set) of candidate models. Exploration of the non-dominated set itself can yield insight into the nature of the training dataset – how complex it is and what can and what cannot be learned from it, eventually. The exploration becomes even more useful when population-based optimization methods are used that do not restrict, for example, the number of free parameters in the model before analysis. When applied to neural networks in particular, the set-valued optimization solution can contain models with varying numbers of neurons or connections instead of fixing the network architecture beforehand.

Simple methods to handle the bias-variance trade-off, such as regularization by means of a "weight decay" term, translate into the well-known method of scalarization (i.e., turning the problem into a scalar function) via aggregation

by weighted sums of objective functions in a multiobjective setting. For objective functions known to be convex, the weighting method could in fact be used to sample solutions belonging to the non-dominated set, but the method has serious limitations when the behavior of the functions is not simple. This work suggests that more versatile methods for multiobjective optimization should be used in nontrivial machine learning tasks, and that currently the most appropriate approach would be to use population-based memetic algorithms that aim to discover a representative sample of the Pareto optimal set of classifier candidates.

For example, the sensitivity vs. specificity issue is traditionally dealt with by using receiver operating characteristics graphs (ROC), in which a threshold parameter can be selected to find a suitable operating point when the actual parameters of the model have already been selected. Actual population-based learning methods can be used for the same purpose, but without the constraint of using just one free parameter in the ROC examination phase. Instead, the non-dominated population of varying kinds of models can be analyzed in the sense of the ROC curve.

In this thesis, multiobjective formulations of machine learning goals are spelled out in a way that is independent of the specific selection of methods. For the actual methodological contribution and computational experiments, two stabilized and well-known artefacts are selected to be combined: multilayer perceptron (MLP) as a prediction model and the NSGA-II optimization algorithm as the starting point of a memetic algorithm for learning. Benefits and applications of using a multiobjective memetic optimization approach in supervised learning are explored, and specific challenges arising from the selected combination of tools are identified. Solutions to conquer the challenges are proposed, implemented, and verified by experiments.

1.1 Earlier Work by the Author

Doctoral dissertations of the author's home department are most commonly compiled as collections of previously published articles with a short introduction that binds the earlier works in a common context. In this case, though, it was decided that a monograph is a more convenient format, since the work presented here is a complete "capstone study" bringing together some separate aspects of the various research activities of the candidate so far. In what follows, the history leading to the current work is presented with the most relevant citations. It may also be worthwhile to mention that all of the cited works except that of Heikkola et al. (2006) were prepared after the MSc degree of the author.

The author was first acquainted with multiobjective optimization while developing an interface between a physical modeling software (Numerrin) and an optimization platform (WWW-NIMBUS). A successful application to ultrasonic transducer design optimization (Heikkola et al. 2006) was published as a result, in addition to some departmental technical reports not worth citing here.

Later the author moved to machine learning and industrial data analysis using a specific multilayer perceptron implementation. Of special interest in that work were means of improving generalization of the learning model and its existing implementation (Nieminen and Kärkkäinen 2009) and exploring the use of ideas from robust statistics to reduce sensitivity to class noise in classification problems (Nieminen and Kärkkäinen 2010).

As yet another research activity, the author managed a publically funded research project related to industrial data mining. Results of the project (called RISC-PROS) are documented in technical reports by Nieminen et al. (2010), Averbuch et al. (2010), and Ivannikov et al. (2010). An application related to the design of experiments in a pilot paper machine was also published in Nieminen et al. (2011). More recently, the author temporarily visited the field of informetrics, taking part in developing a method for automatic literature clustering (Nieminen et al. 2013). Preliminary results related to this dissertation have already been published in a conference article by Nieminen and Kärkkäinen (2016).

1.2 Contributions of this Dissertation

The earlier works might seem unrelated at first, but after some thought, a sound logic emerges that binds everything together. In the applied works on machine learning, the learning problem was formulated, and dealt with, as an optimization problem where a cost function is minimized. The canonical ways to handle generalization manifest themselves as penalty terms that are added to the cost function to "regularize" the model and drive the optimization towards simpler or smoother models. The method used for class noise management published by Nieminen and Kärkkäinen (2010) was also based on modifying the cost function. Developments were surely achieved and applied in practical tasks in published and unpublished works, but minimizing a weighted sum of cost and penalty terms seemed inadequate when viewed from the background of multiobjective optimization. Another inadequacy was that single-shot optimization of the employed model, the multilayer perceptron, is sensitive to initial random starting values of the model parameters. Multistart optimization, i.e., local optimization from multiple randomized starting points, had to be used in applications, naturally always resulting in a whole population of trained models. No further advantage of the population or its diversity was systematically taken, though.

In the work at hand, the limitations of the earlier attempts are alleviated by taking a holistic approach that properly separates, rather than forcefully combines, the conflicting objectives naturally emerging in multilayer perceptron training. The training is also done with a fundamentally population-based algorithm, reducing the effect of the initial randomization. Incremental tool development is thus the driving purpose and contribution of the work documented in this dissertation.

The literature clustering tool of Nieminen et al. (2013) was put to one of

its first practical tests in preparing the literature overview of this dissertation. It happens that some new developments of the tool, again partly designed by the author, are presented here for the first time, adding to the overall contribution a surprising aspect of informetrics research.

In addition to mere results and algorithmic ideas, the output of computational sciences contains the implementations that ultimately describe the details of the actual computation. The datasets, computations, results, and illustrations of this dissertation can be exactly reproduced from publically available source code contributed by the author[1], in accordance with the Open Science Initiative[2] of the Finnish Ministry of Education and Culture.

The rest of this dissertation is organized as follows. Chapter 2 introduces the concepts, methods, and terminology used throughout. Chapter 3 reviews existing literature on multiobjective machine learning with an emphasis on multilayer perceptron training and closely related methods. Chapter 4 describes the automatic literature clustering tool of Nieminen et al. (2013) along with new developments, and details the process of charting the literature for this dissertation. Chapter 5 digests the literature review into a list of goals and features expected of a current multiobjective MLP training method. It also describes the specific goals, algorithms and implementation of the incremental method development performed for this dissertation. Chapter 6 contains illustrative examples of classification tasks and datasets in which the role and nature of multiobjective learning and the interplay of different objective functions can be experienced. Chapter 7 presents the summary and conclusions of the research.

1 http://users.jyu.fi/~nieminen/research/dissertation2016/

2 http://openscience.fi/

2 OVERVIEW OF CONCEPTS AND METHODS INVOLVED

This chapter outlines the key problems and methods discussed in this dissertation and defines the the terminology and notations used later on. Section 2.1 defines the notations and goals in classification tasks based on numerical data vectors. Section 2.2 describes the multilayer perceptron used here as the learning machine. Section 2.3 explains the multiobjective nature of supervised learning tasks and outlines methods suitable to solve them. Section 2.4 presents the algorithmic framework of population-based multiobjective memetic optimization suitable for multiobjective learning. The chapter is concluded by a few notes of the history of the related methodology in Section 2.5 and an example application in Section 2.6.

2.1 Machine Learning

Computational emulation of natural cognitive and learning processes dates back to the beginnings of the computer era. The first fundamental concepts such as those of McCulloch and Pitts (1943) were created even before the first computers. The current terminology of "learning machines" and "machine learning" started to appear in the titles of scientific articles, conference sessions, and journals in the late 1950's, as exemplified by the early works of Friedberg (1958), Friedberg et al. (1959), Campaigne (1959), and Samuel (1959). It is of course the computer program that learns instead of the machine that executes the program, so "learning machine" was known to be a "misnomer" from the very beginning (Ware 1955), but apparently it became the name that stuck.

2.1.1 What is "Learning" Anyway?

In an overview published roughly in the middle of the history of machine learning so far, Carbonell et al. (1983) described the subject matter of machine learning

broadly as "the study and computer modelling of learning processes in their multiple manifestations". As examples of these "multiple manifestations" of learning processes, they offered the following:

- "the acquisition of new declarative knowledge"
- "the development of motor and cognitive skills through instruction or practice"
- "the organization of new knowledge into general, effective representations"
- "the discovery of new facts and theories through observation and experimentation".

The historical overview of Carbonell et al. (1983) further goes on to categorize machine learning research up to the 1980's in various dimensions, namely the underlying *learning strategy* (such as construction of prior knowledge as in programming, memorization of facts from data, learning from a teacher, by analogy, from examples or from the external environment, or completely without a teacher), *type of knowledge acquired* (such as parameters in an algebraic model, decision trees, formal grammars, production rules, logical expressions, graphs, schemas, computer programs, taxonomies, or mixed representations including multiple types of knowledge), and *domain of application* (such as image recognition, robotics, natural language processing, and many others).

All the same general taxonomies of machine learning are of course still valid today. This dissertation deals specifically with so-called feedforward artificial neural networks as the machines that learn, so we will adopt a specific and useful taxonomy of learning strategies and learning tasks which is presented, for example, in the neural network textbook by Haykin (2009). In this taxonomy, the learning strategy, or learning process, is one of the following:

- *Learning with a teacher*, which is commonly called also *supervised learning*. In this strategy, there is a "teacher" involved in the learning process, actively changing the internals of the learning machine while the learning takes place.
- *Learning without a teacher*, which Haykin (2009) further divides in two subcategories:
  - *Reinforcement learning* where there is no teacher, but the learning machine will be changed according to feedback from a "critic" that analyses the current outcome (positive or negative) of the machine's interaction with the environment.
  - *Unsupervised learning* where there is no teacher nor a critic to alter the learning machine. Learning happens in a *self-organized* manner, after setting up a task-dependent ideal of what the machine is to learn from its environment.

Artificial neural networks are especially apt in some specific machine learning tasks, which are categorized by Haykin (2009) as follows:

- *Pattern association* where the brainlike activities of heteroassociation (memorizing patterns that relate to another pattern and later recalling the connection by association) and autoassociation (memorizing crisp patterns and later recalling a crisp pattern when presented with a possibly distorted or partial version of the same ideal pattern).
- *Pattern recognition*, i.e., "the process whereby a received pattern/signal is assigned to one of a prescribed number of classes".
- *Function approximation*, i.e., approximation of nonlinear functions.

In addition to the above three, Haykin (2009) mentions also the tasks of *Control* and *Beamforming*, for which neural networks are most useful, but which are further away from the focus of this dissertation.

Using the above taxonomies as a frame of reference, most of this dissertation deals specifically with *supervised* learning for *pattern recognition*. The word *classification* will be used for the domain here, because we want the machine to be able to tell the class of objects that an input pattern belongs to. Supervision comes into play when the learning is based on example data with labels indicating true classes of the input patterns. In this case, a "teacher" can be applied to guide the learning. In practice, the teacher is able to evaluate error functions and produce rules both to improve the internals of individual classifier models and also to guide a simulated evolutionary process of a population of models. In what follows, such learning with a teacher will be called *training*. As an analogy, one might think about the real-world scenario of a teacher or a coach with superior knowlegde training a novice. In the classification tasks considered here, the teacher knows the input patterns and the correct class labels that constitute perfect knowledge of a sample of a measured environment. The training process then aims to transfer this knowledge to the learning machine for practical use. The type of knowledge acquired is the structure and parameters of an algebraic model, namely the multilayer perceptron neural network.

2.1.2 Ways to Emulate Learning with Machines

The learning machines used in this work belong to the category of artificial neural networks, the history and fundamentals of which are thoroughly described, for example, in the classical textbooks of Bishop (1995) and Haykin (2009). A brief note must be made that there are many other software (and even hardware) devices that are able to emulate natural learning processes. A massive number of textbooks have been written about machine learning, describing different tools, techniques and both theoretical and practical points of view. A major web store[1] lists no less than 1415 book titles including the search word "machine learning". 31 of them have been published within a three month period preceding the query. It almost seems like there is a specific textbook written for every particular application domain and every programming platform or language imaginable.

[1] www.amazon.com - the query on "machine learning" in book title was made on Aug 6, 2015.

Most of the textbooks, the couple mentioned here being no exception, cover some fundamental aspects of probability theory and statistics, which form the theoretical foundations of most machine learning methods. This approach reflects the fact that statistical modeling, performed with or without computers, can be interpreted exactly as the kind of learning that we wish the machines to perform. In the actual computational world, though, there is always the algorithm and the program involved, and it may be possible to extend or modify statistically sound methods with further heuristics that work in practice, even if the roots of their performance cannot (yet) be explained by mathematical proofs. Some recent examples of this kind of a cycle between practice and theory are the findings of Mehta and Schwab (2014) and those of Choromanska et al. (2015) that attempt to explain theoretically what actually is happening when using so-called deep learning models (see, for example, Hinton et al. 2006) which have proven effective in practice even without a complete explanatory theory. Another example of the appearance of theoretical results after the algorithms have been succesfully used in practice are the universal approximation results of approximators, for example those for feedforward networks (e.g., Hornik et al. 1989) and for support vector machines (Hammer and Gersmann 2003). Also to be noted is that theoretical proofs, even if they are very satisfactory by themselves, have assumptions and limits that may not always be relevant in real-world practical use (Tikk et al. 2003).

Now, after citing some very theoretically oriented papers, a point must be made that the rest of this thesis is a continuation of practical development work, and the research involved is made from an explorative and empirical standpoint. In the case of machine learning, the point of view is supported by the long track record of algorithms that have found practical use before their complete theoretical underpinnings have been thoroughly investigated.

2.1.3 Learning from Numerical Data Sets

After the more philosophical considerations of what defines "learning" in nature or its emulation in a machine, we must come down to actual representations useful in computer programming, where everything has to be represented digitally. Especially in the context of multilayer perceptrons, we represent things in the observable environment of the learning machine as numerical vectors which can be observed and recorded. Then the learning tasks, especially that of pattern recognition, can be expressed as the task of approximating an unknown function that maps observation vectors to vectors that represent classes of objects to be recognized.

In what follows, we shall define that the word *data* means any "mass" of digitized information[2], be it recorded observations of real-world objects or any other subjects of study in computational tasks.

A *data object*, consisting of variable values measured from a real-world or

[2] Outside the confines of this dissertation, we are now living the era of "Big data", where the nature and structure of a single "datum" does not play a very significant role anymore.

simulated system during one observation, is written as a vector in n-dimensional Euclidean space, $\mathbf{x} \in \mathbb{R}^n$. Similar boldface notation is used for other vector-valued constructs in the equations hereafter. Such a vector can also be called a *point* in the measurement variable space. Yet another name for an individual data object especially in the classification context is *feature vector*, as the components are essentially some descriptive "features", some of which may be derived from original values.

A *dataset*, written $\{\mathbf{x}_i\}_{i=1}^N$ consists of multiple observations. It is common to write such a dataset as an $N \times n$ matrix, $\mathbf{X} \in \mathbb{R}^{N \times n}$,

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{pmatrix}.$$

Boldface capital letters are used also for other matrix-valued constructs. The vector subscript $\mathbf{x}_j$ means the j:th vector in a set of vectors $\{\mathbf{x}_i\}_{i=1}^N$ whereas a round bracket followed by a subscript $(\mathbf{x})_j$ means the j:th component of a vector $\mathbf{x}$.

When machine learning is used for function approximation tasks (or pattern recognition formulated as a special case), another dataset $\{\mathbf{t}_i\}_{i=1}^N$ is usually measured or created, containing the *target vectors*, i.e., the ideal or expected q-dimensional outputs $\mathbf{t}_i \in \mathbb{R}^q$ that are known to correspond to each input vector $\mathbf{x}_i$. The task for the machine is then to learn how to map each vector $\mathbf{x}_i$ in the input dataset to the corresponding vector $\mathbf{t}_i$ in the target dataset. In classification, the actual target output required from the model is an integer $c_i \in \{1, \ldots, K\}$ where K denotes the number of possible classes. Such an integral target c_i can be represented as a binary vector $\mathbf{t}_i \in \mathbb{R}^K$ where the c_i:th component is $(\mathbf{t}_i)_{c_i} = 1$ and all other components are zeros, $(\mathbf{t}_i)_j = 0$ for $j \neq c_i$.

Just memorizing the discrete mappings in the so-called *training set* of pairs $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^N$ is not enough, though. We want the machine to be able to *generalize* what it has learned by example, and apply the knowledge to new vectors $\mathbf{x}' \notin \{\mathbf{x}_i\}_{i=1}^N$ for which nobody knows the proper output before the machine makes its advisory guess, or *prediction*. Only through proper *generalization capability*, function approximation really becomes useful, and pattern recognition becomes robust against minor deviations or distortions from the "ideal" patterns available while training the learning machine. Ways of dealing with the generalization capability are one central focus of the work at hand.

2.2 Multilayer Perceptron as a Learning Machine

Haykin (2009) defines a neural network as a "massively parallel distributed processor made up of simple processing units that has a natural propensity for storing experiential knowledge and making it available for use". His definition further clarifies that (i) "Knowledge is acquired by the network from its environ-

ment through a learning process" and that (ii) "Interneuron connection strengths, known as *synaptic weights*, are used to store the acquired knowledge". In this section, a specific kind of such a device, namely the *multilayer perceptron* (MLP), is described through a brief historical overview. The multilayer perceptron has been explored for more than half a century, so this section contains no news for those already knowledgeable of neural networks. The reiteration here serves the purpose of detailing the notations and concepts used later in this dissertation.

2.2.1 Artificial Neural Networks

According to Haykin (2009), the published history of artificial neural networks dates back to the neuropsychological studies of the first half of the 20th century. McCulloch and Pitts (1943) pointed out that the function of biological neurons could be modeled by means of propositional logic, thereby setting up a link between biological cognitive systems and computational models. Rosenblatt (1958) proposed an artificial construct called the perceptron, which evidently had the capability of learning and generalizing from experience and had properties that could be quantified and analyzed theoretically.

These pioneers already made it clear that their mathematical models of cognition were rough simplifications of the actual natural processes in neuronal nets in physical organisms. Computational neuroscience has come a long way since the beginning, with models of increasing level of detail (see, for example, Koch 1998). Nevertheless, even the simple constructs derived from the early perceptron concept have found much use in applications (Haykin 2009). Like biologists use simple "model organisms" to study the general aspects of life, this dissertation explores the premises of multiobjective machine learning using a simple and well-known structure which shall be presented next.

2.2.2 Supervised Learning using the Multilayer Perceptron

As in Haykin (2009), we begin with a single computational unit, or "artificial neuron", required in models such as the early perceptron of Rosenblatt (1958). Figure 1 is a rough sketch of a biological neuron, found, for example, in the human nervous system and in the human brain. Biological data on this rough level of detail had been gathered at the time that the early works on artificial neural networks began. Already McCulloch and Pitts (1943) recall from yet earlier research how the nervous system was known to consist of cells, each of which has a nucleus, an axon, and synapses that connect the cell to others in a vast network of similar cells. It was known that signals travel between these cells (through the axons and synapses) at velocities between 1 and 150 meters per second, and that when the total sum of received input signals within a sub-millisecond timescale would exceed a certain threshold, an outgoing impulse would we triggered in the neuron and propagated through the axons of the cell within a half-millisecond timescale.

Skipping some history in between, a simple mathematical model for the

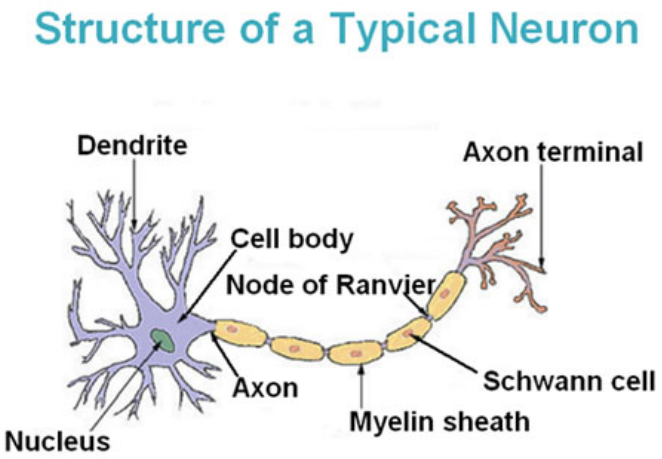


FIGURE 1 A rough sketch of a biological neuron. Public domain image (source: http://commons.wikimedia.org/wiki/Image:Neuron.jpg).

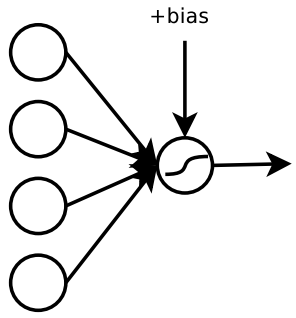


FIGURE 2 A rough model of a biological neuron.

activity of one biological neuron has become canonically formulated as follows:

$$o = \phi(b + \sum_{i=1}^{n_0} w_i(\mathbf{a})_i). \tag{1}$$

In the equation, $o \in \mathbb{R}$ stands for the output of the neuron, $(\mathbf{a})_i \in \mathbb{R}$ are the n_0 inputs connected to the neuron, $w_i \in \mathbb{R}$ are positive ("exhibitory") or negative ("inhibitory") weight coefficients roughly corresponding to the weight or "thickness" of synapses connected to the neuron, and $\phi : \mathbb{R} \to \mathbb{R}$ is an *activation function*, which should be a step function (or more preferrably a smooth sigmoidal function) that can "fire" the neuron when the input signals cross a threshold selected by the bias parameter $b \in \mathbb{R}$. Figure 2 is a schematic of the equation. The "s"-shaped icon in the central circle stands for a sigmoidal activation function ϕ. The signals from arbitrarily many input synapses (four in the case of the figure) arrive from inputs on the left, and a single output value is transmitted to the right.

From the beginning, it has been clear that in biological neural networks, such as the brain, the observed functionality emerges from the collaboration of a massive number of interconnected neurons. A simple way to model something like this parallelism mathematically is to consider a *layer* of parallel neurons in-

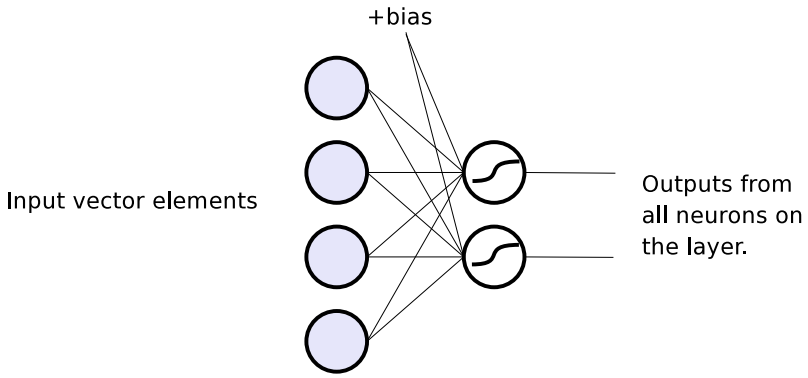


FIGURE 3 A parallel layer of artificial neurons.

stead of a single one. Figure 3 illustrates the idea with a layer of two parallel neurons. In general, there could be any number of parallel neurons on the layer. As in the previous consideration of a single neuron, inputs come from sources on the left side of the figure, through synapses with different weights. All the parallel neurons on the receiving layer are fundamentally similar to each other in their action, but the synaptic weights can be different for each neuron, and technically there could be variation in the activation functions (such as differences in steepness or shape). Each neuron will produce an output signal depending on its own weights of input synapses and its own bias, according to Equation (1). Now both the input and the output of this layer of neurons are vector valued:

$$(\mathbf{o}^l)_j = \phi_j(b_j^l + \sum_{i=1}^{n_{l-1}} w_{j,i}^l (\mathbf{o}^{l-1})_i). \tag{2}$$

In Equation (2), $\mathbf{o}^l \in \mathbb{R}^{n_l}$ stands for the vector-valued output of the layer with n_l neurons, given an upper index l here for purposes that will be clear in the few paragraphs that follow. $(\mathbf{o}^{l-1})_i$ are now the n_{l-1} input variables. $w_{j,i}^l$ is the weight coefficient of the i:th input in the j:th neuron of the layer. Again, ϕ_j is a sigmoidal activation function. b_j^l is the bias parameter of the j:th neuron. The indices $l-1$ for the input vector and l for the output vector anticipate the chaining of multiple layers, to yield the actual *multilayered perceptron* depicted in Figure 4.

We can let go of some of the tedious element indices, if the weights of the j:th neuron of the l:th layer are written as a vector $\mathbf{w}_j^l$. Then the function of the neuron can be written using matrix algebra as follows:

$$(\mathbf{o}^l)_j = \phi_j(b_j^l + (\mathbf{w}_j^l)^T \mathbf{o}^{l-1}).$$

A further trick can be used to shorten the notation. Let us define an extension operator, marked with a tilde, that adds a "zeroth component" of value one to a vector so that $\widetilde{\mathbf{o}}^{(l-1)}$ becomes

$$\widetilde{\mathbf{o}}^{l-1} = \begin{pmatrix} 1 \\ \mathbf{o}^{l-1} \end{pmatrix}. \tag{3}$$

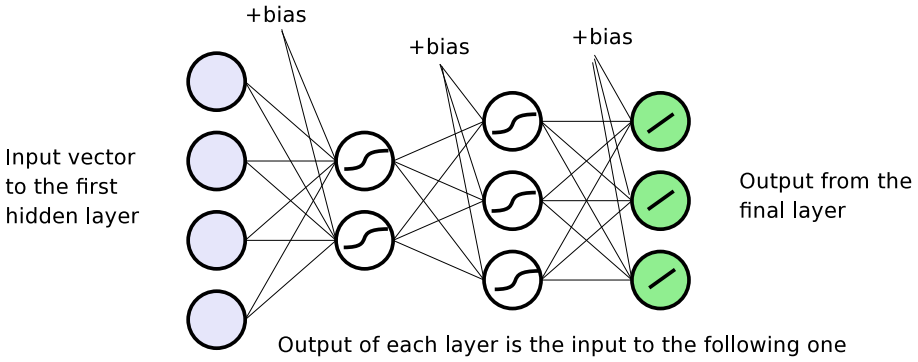


FIGURE 4 Putting together several layers of artificial neurons leads to the multilayered perceptron model.

Additionally, assemble a matrix of weights for the l:th layer so that the biases appear as the "zeroth" column vector:

$$\mathbf{W}^l = \begin{pmatrix} b_1^l & (\mathbf{w}_1^l)^T \\ b_2^l & (\mathbf{w}_2^l)^T \\ \vdots & \\ b_{n_l}^l & (\mathbf{w}_{n_l}^l)^T \end{pmatrix}. \tag{4}$$

Using the notations introduced so far, the whole action of a multilayer perceptron can be written in a simple matrix iteration as, for example, in the works of Hagan and Menhaj (1994) and Kärkkäinen (2002):

$$\mathbf{o}^0 = \mathbf{x}, \quad \mathbf{o}^l = \mathcal{F}^l(\mathbf{W}^l \tilde{\mathbf{o}}^{(l-1)}) \quad \text{for} \quad l = 1, \ldots, L. \tag{5}$$

In the equation, $\mathbf{x}$ is the input vector. We set it as the "output of the zeroth layer". Then, for each of the L layers in the MLP, the bias terms are included using the trick described in Equations (3) and (4). $\mathcal{F}^l$ plays the role of applying the activation functions $\phi_1^l, \ldots, \phi_{n_l}^l$ to the elements of the matrix-vector product. The final output of the MLP resides in the output vector $\mathbf{o}^L$. We use the notation $\mathcal{N}(\mathbf{x}) = \mathbf{o}^L$ to denote the output of the network for vector $\mathbf{x}$, purposely omitting notation for the obvious fact that the output depends on the weight matrices $\{\mathbf{W}^l\}_{l=1}^L$ and specific activation functions $\{\mathcal{F}^l\}_{l=1}^L$. As for terminology, we call layer 0 the *input layer*, layer L the *output layer*, and all other layers, i.e., $1, \ldots, L-1$, the *hidden layers*. The rows of weight matrices that compute values on the output layer are called *output neurons*. Those on the hidden layers are called *hidden neurons*. The elements of the input vector are called *input nodes* to differentiate between "neurons" that compute values and the inputs that merely receive values from the outside environment. The *topology* or *architecture* of an MLP is determined by the number and sizes of the layers, and the synaptic weights that are enabled in the computation.

Given the set $\{\mathbf{x}_i\}_{i=1}^N$ of training input vectors and the corresponding set $\{\mathbf{t}_i\}_{i=1}^N$ of training target vectors, an output vector $\mathcal{N}(\mathbf{x}_i)$ can be computed for each training input $\mathbf{x}_i$, and the *error vector* of the outputs evaluated for the i:th

training pair $(\mathbf{x}_i, \mathbf{t}_i)$ is obtained as

$$\mathbf{e}_i = \mathcal{N}(\mathbf{x}_i) - \mathbf{t}_i.$$

An example of a traditional way of training the neural network now works by minimizing the cost function

$$\mathcal{J}_\beta(\{\mathbf{W}^l\}) = \frac{1}{2N}\sum_{i=1}^{N}||\mathbf{e}_i||^2 + \frac{\beta}{2\#I}\sum_{(l,j,i)\in I}|w_{j,i}^l|^2, \tag{6}$$

where $\beta \geq 0$ is the *regularization* or "*weight decay*" parameter, common in MLP training. In our variant, the set I can be adjusted to contain only a subset of the indices of the weight matrix elements $w_{j,i}^l$ for which the original lowercase notation of Equation (2) is used. The number of elements in the index set I is denoted by $\#I$. For example, one could select all weights except those corresponding to the first column of $\mathbf{W}^L$, i.e., the biases of the linear output layer. Kärkkäinen (2002) provides further detail to the benefits of such a formulation, especially when the identity function is selected as the activation function of the final layer. For example, in such a formulation, the output biases are able to balance the nonlinear action of the hidden neurons so that that the average of the error vectors at any local cost function minimum is guaranteed to be the zero vector, i.e., $\frac{1}{N}\sum_{i=1}^{N}\mathbf{e}_i = \mathbf{0}$ at local minima.

Of great importance to the premises of this work is that Equation (6) has two terms: one that reflects the approximation error of the MLP with respect to the training data and another one that reflects the "complexity" of the model in the sense that larger weights result in a less linear and less smooth model. One reason to include such a complexity penalty term is to avoid *overfitting* the MLP to too much detail or noise in the training data, thus improving its desired capability to generalize what it has learned. Another usual way to avoid overfitting is to compute the approximation error separately for a *validation dataset* that consists of points that are not used for the actual training.

The algorithm of error backpropagation, introduced by Rumelhart et al. (1986a,b), offers the basis for a large number of learning mechanism for differentiable cost functions such as Equation (6). The most simplistic *backpropagation*, or "*backprop*" method computes partial derivatives (gradient) of the error with respect to each weight and then updates the weights in the direction opposite to the gradient, multiplied by a small step size. In such a method, the term weight decay has a natural interpretation, as each step tends to decrease the magnitude of each weight. Availability of the cost gradient allows the use of also other, more efficient, methods such as the conjugate gradient algorithm (Møller 1993).

Other popular learning machines similar to MLPs are radial basis function networks (RBFNN), support vector machines (SVM) (Haykin 2009), and the recently popularized extreme learning machine (ELM) (recently applied, for example, by Mesquita et al. 2016).

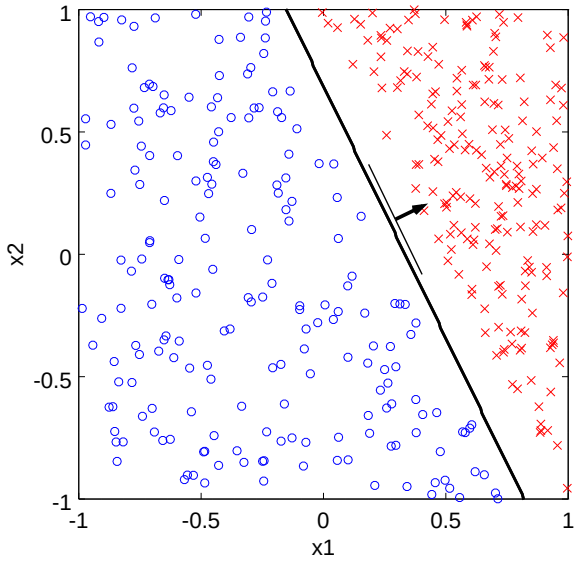


FIGURE 5 Linearly separable binary classification. One neuron (a perceptron) suffices.

2.2.3 Examples of the MLP Action for Binary Classification

Figure 5 illustrates the function of a single artificial neuron, i.e., "a perceptron", in perhaps the most simple case of classification. A complete simulated training dataset is shown. The task is to differentiate between crosses (class 1) and circles (class 2). Illustrated in two dimensions here, this kind of a task can be solved by a single neuron that creates a separating hyperplane between the classes. In the two-dimensional case, hyperplanes are straight lines. The illustration shows this *decision boundary* where the perceptron splits the space between its two possible predictions. The transition from one class to another is smooth because of the smooth sigmoidal activation, but the illustrations in this dissertation show only the hard boundary exactly on the hyperplane of transition. Also shown is the normal vector of the separating hyperplane. Thickness of the arrow reflects the magnitude of the weights and thus the steepness of the logistic sigmoid activation function operating in the neuron.

For the nonlinear case depicted in Figure 6, one separating hyperplane cannot differentiate between crosses and circles. Shown instead is the action of a two-layer perceptron where two hidden neurons are responsible for creating two separating boundaries which are then summed in the output to produce a nonlinear decision boundary (see Figure 4 and Equation (5)). The hidden action of both neurons is illustrated by showing the location and steepness of their activation similarly to Figure 5. The model shown in Figure 6 has relatively small weights so the decision boundary is a smooth curve. In this case, the approximation error could be further and further minimized by letting the weights grow unbounded, producing a sharper edge as the sigmoidal action would approach the step function. This is usually not desired, so to get the illustrated smooth action, either a penalty term like that in Equation (6) must be added or the learning must be stopped at an early point. Without any penalty (and given infinite precision in

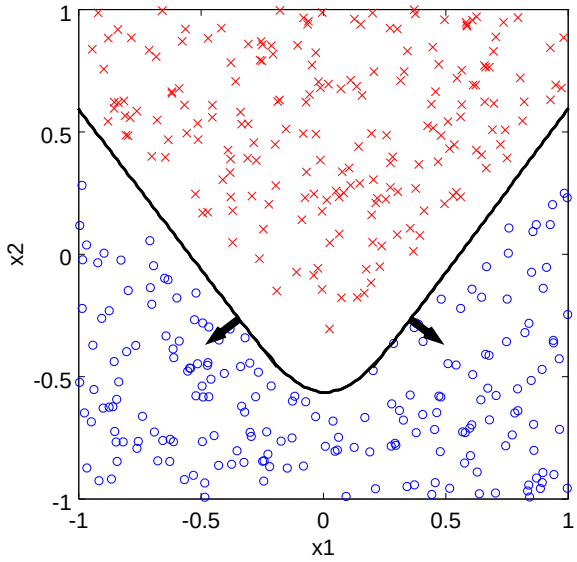


FIGURE 6 Nonlinear binary classification requiring at least two hidden neurons.

derivative computations), an iterative learning method would progress towards larger weights indefinitely.

2.3 Multiobjective Optimization for Supervised Learning

Multiobjective, or multicriteria, optimization (MOO) (see, for example, Ehrgott 2005; Miettinen 1999) augments the classical field of optimization by considering multiple *objective functions* (also called fitness functions, cost functions, criteria) simultaneously. The concept is treated already in the nonlinear optimization considerations of Kuhn and Tucker (1951) as "vector maximization". According to, for example, Miettinen (2008), the origins of MOO can be traced back even to the 19th century writings of Edgeworth (1881) and Pareto (1896). Over the years, MOO has matured and gained impetus via practical applications in many fields (Branke et al. 2008). This section first outlines the relevant definitions and concepts and then shows how multiobjective formulations appear naturally in MLP classifier training.

2.3.1 Multiobjective Optimization

Because the maximization of a function can be turned into minimization trivially by changing the sign of the function, we lose no generality by considering minimization only. For m objectives, we are to solve

$$\min_{X \in \Omega} \quad \mathbf{f}(X) = (f_1(X), f_2(X), \ldots, f_m(X)), \tag{7}$$

where $f_i(\cdot)$ denotes the ith real-valued objective function and Ω the set of admissible values for the unknown X. The general notation $X \in \Omega$ allows us to

consider a very free structure of the design space of the optimization problem. In the case of MLPs, the elements of Ω could be the possible values of the weight matrices. The set of admissible values can be limited by, for example, box constraints, i.e., maximal and minimal allowed values, for each weight. In general, Ω could in fact contain MLPs of different numbers of layers and different numbers and kinds of neurons on each layer. It is completely possible to allow also other learning models, and even combinations ("ensembles") of several models to exist in Ω.

The first example of multiobjective optimization in machine learning has already been lurking in the formulation of Equation (6), which can be considered as an *aggregated* or *scalarized* function consisting of two different objectives:

$$f_1(\{\mathbf{W}^l\}) = \frac{1}{2N}\sum_{i=1}^{N}||\mathbf{e}_i||^2 \tag{8}$$

$$f_2(\{\mathbf{W}^l\}) = \frac{1}{2\#I}\sum_{(l,j,i)\in I}|w_{j,i}^l|^2\,. \tag{9}$$

When formulated in terms of MOO, there is no longer a need for *a priori* setting of hyperparameters such as β in Equation (6). Of essence in MOO is the *conflict* and necessity of *trade-off* or *compromise* between the two or more functions being optimized. In the case of Equations (8) and (9) and the example of Figure 6, error improvements must be traded off with increased magnitudes of the synaptic weights and vice versa. As long as the position and direction of the separating hyperplanes has been found, any further trade-offs between the approximation error and the weight magnitude would have to be considered equally good, i.e., "equally optimal", unless some further decisions were made on smoothness required of the model. Experimentation or brute-force sampling using the regularization weight coefficient of Equation (6) suffices in a simple scenario to sample any number of such optimal points, but in general we are aiming at more flexible means to perform MOO than this well-known "weighted sum method" (as it is called, for example, in the textbook of Ehrgott 2005).

For defining what "optimal" means in MOO, we adopt the well-developed concept of *Pareto-optimality* based on the *dominance relation*. A solution image $\mathbf{f}(X')$ is said to dominate $\mathbf{f}(X)$, notated $\mathbf{f}(X') \preceq \mathbf{f}(X)$, if $f_i(X') \leq f_i(X)$ for all $i \in \{1,\ldots,m\}$ and $f_i(X') < f_i(X)$ for at least one $i \in \{1,\ldots,m\}$. Instead of a single optimum, of interest is the *Pareto set* (PS) of *non-dominated* (also known as non-inferior, efficient, or Pareto-optimal) solutions for which no dominating solutions exist, $P = \{X \in \Omega \mid \neg\exists Y \in \Omega : \mathbf{f}(Y) \preceq \mathbf{f}(X)\}$. The image of the PS in the objective space $\mathbf{f}(P)$ is called the *Pareto Front* (PF). Intuitively, a solution not in the PS is suboptimal because improvements are possible without compromises. In MOO methods based on sampling the (generally infinite) PS, an approximation of the PF is presented to the user, customarily called the Decision Maker (DM), who ultimately has to select a preferred compromise solution. The PF exploration itself can reveal insights into the MOO problem at hand.

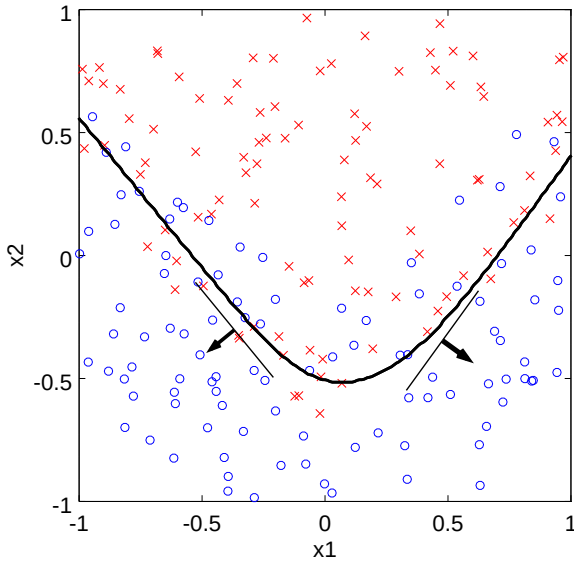


FIGURE 7 Binary classification task where a nasty overlap between classes makes it impossible to make a clear separation.

2.3.2 Inherently Conflicting Objectives in Classification

Figure 7 is where things start to get interesting for the research presented here. Observations from both of the classes in a binary classification task are found in an overlapping area, and a simple model with two hidden neurons just cannot differentiate between the two as it did in the separable (non-overlapping) case depicted in Figure 7. Indeed, this is the case also with many real-world scenarios. One might argue that the decision boundary shown in the image is in fact quite good for this case. With a seemingly complicated data like this, we perhaps should not do more than approximate a smooth boundary, and thereby simplify the phenomenon producing the data. We might actually gain some insight into its inner workings by analysing the simple model. The question remains whether it is good to have the decision boundary in the halfway of the class overlap. In a practical scenario, different costs could apply to false predictions to one class or the other, and tools such as Receiver Operating Characteristics (ROC) graphs are commonly used to decide which model is to be preferred (Fawcett 2006).

Another question is, whether the model should be simplified or not. The model in Figure 8 fits the data better with its 10 hidden neurons and sharper boundaries, and without the aid of the visualization, it would be hard to decide that a simple model is better than the complex one. On the other hand, oversimplification could sometimes make it impossible to meet the complexity of the data, as in Figure 9 where the classes are nicely separable, but the separation requires multiple neurons due to the more complex shape.

The first of the above questions relates to the well-known consideration of sensitivity versus specificity, and the latter one to that of bias versus variance. *A priori* answers to these difficult questions are required when only a single cost function is considered as the learning objective. The main argument advocated

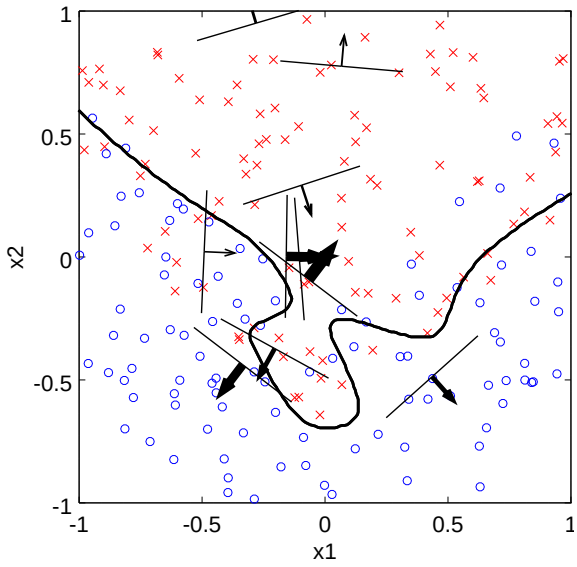


FIGURE 8 Overfitting or not? With inseparable data, further minimization of approximation error can lead to an otherwise overly complex model.

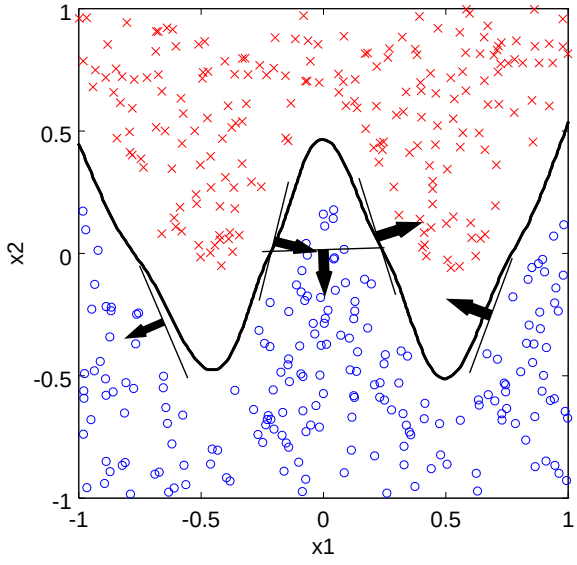


FIGURE 9 How much to simplify? Shape of the data distributions affects the complexity required of the model.

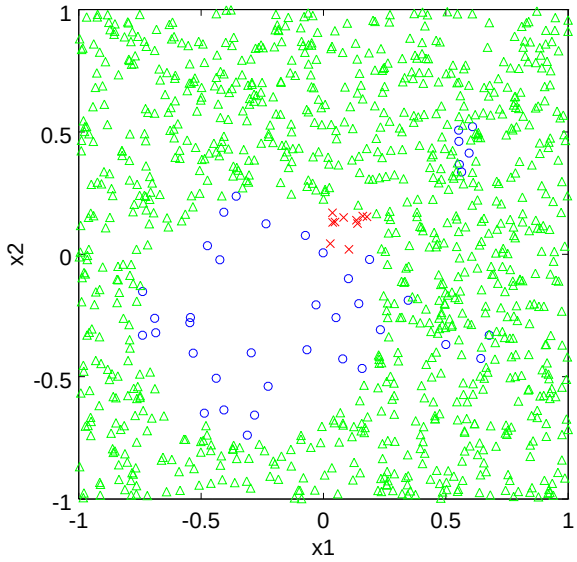


FIGURE 10 Three-class classification, mostly but not completely separable, with subconcepts. No noise. (cf. He and Garcia 2009)

here is that instead of *a priori*, empirical, or trial-and-error selections of model architecture or penalty coefficients, a complete multiobjective optimization scheme should be applied with separated, instead of aggregated, qualities of a solution represented as objective functions. The necessary decision of a final model selection sould be done *a posteriori* when knowledge of the trade-offs becomes available via the examination of the non-dominated solutions.

A multiobjective perspective also opens up an intuitive way of combining the considerations and augmenting them to more than two dimensions. Examples of classification datasets with further difficulties are shown in Figures 10 and 11. No classifier candidates are shown yet, as we will return to the examples later on. The figures serve to illustrate how the problem of classification gets more difficult when the separation is to be made between more than two classes, when the available labeled data might have widely different numbers of examples present in each class, when different costs may apply to wrong predictions in each class, when the distributions of classes overlap, and when the given labeling might be partly incorrect.

It is further argued here that population-based multiobjective optimization, introduced in the following section, is a key element in looking at the different objectives and achieving the necessary knowledge for *a posteriori* decisions regarding which model to select for the dataset being examined.

2.4 Memetic Algorithms for Multiobjective Learning

Machine learning tasks may thus be formulated as numerical optimization problems to be solved with suitable algorithms. In this particular work, the frame-

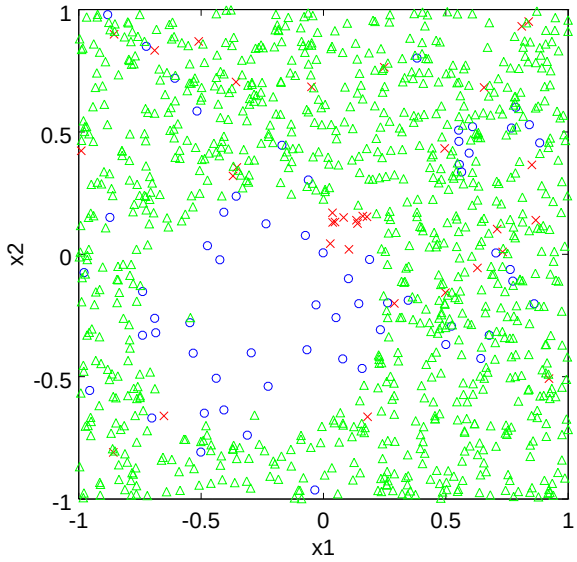


FIGURE 11 Three-class classification, mostly but not completely separable, with sub-concepts and class noise. (cf. He and Garcia 2009)

work of memetic algorithms (MAs) is used. A definition given in a primer by Neri and Cotta (2012) is that memetic algorithms are "population-based metaheuristics composed of an *evolutionary framework* and a *set of local search algorithms* which are activated within the generation cycle of the external framework". The concept is a structured merger of various families of optimization methods which have been used both alone and in hybrid forms long before it became mainstream to call these hybrids "memetic". Developments and applications of memetic algorithms are found also in earlier doctoral dissertations of the author's department, notably those of Neri (2007), Tirronen (2008), Mininno (2011), Iacca (2011), and Poikolainen (2014).

In this dissertation, the presentation of memetic algorithms is based on the textbook edited by Neri et al. (2012) that gives a broad and both current and historical introduction to the memetic framework and possible choices of algorithmic components that can be used in selected places. First the evolutionary part is described along with historical notes and then a general memetic multiobjective framework suitable also for machine learning is outlined.

2.4.1 Evolutionary Computation

The "evolutionary framework" of memetic algorithms refers to some optimization method selected from the palette of Evolutionary Computation (EC), a branch of optimization methods with roots in the computer simulations of the process of Darwinian natural evolution (Darwin 1859) performed since as early as the 1950s (Fogel 1998). In Evolutionary Computation, the best solution to an optimization task is searched among a *population* that undergoes iterated generations of simulated *crossover*, *mutation*, and peer *competition* under simulated environmental pressure. Bäck et al. (1997) overview the history of the first 40 years of evolu-

tionary computation and present a very general algorithmic framework which is recited here as Algorithm 1, using mnemonic variable names for populations and leaving out the clearly implied time index t for the iterations which are called *generations* for obvious reasons. The algorithm is the generic form of an *evolutionary algorithm* (EA).

Algorithm 1: A basic evolutionary algorithm, loosely following the outline of Bäck et al. (1997).

```
Function BasicEA()
    pop ← Initialize() ;
    Evaluate(pop) ;
    while not Terminate do
        newpop ← Variation(pop) ;
        Evaluate(newpop) ;
        pop ← Select(Union(newpop, Subset(pop))) ;
    return BestIndividual(pop);
```

The algorithm dictates only the progression from an initial population to better and better populations via the successive *variation* of the population individuals and the *selection* of individuals for the next iteration (i.e., "generation"). The basic algorithm gives much freedom of choice in the way by which the first population is initialized, how the variation and the selection are done, and even which individuals are considered for selection, based on the new and old variants. Technically, even the operations or their parameters might change from one iteration to the other in a dynamic or adaptive instantiation of the algorithm. Also open to customization in the general evolutionary computation framework is the way in which the population individuals are *represented* (or *encoded*) in what is called the *chromosome*, or *genome*, of the individual, and, of course, how their performance or *fitness* is evaluated. This freedom of customization is emphasized by Bäck et al. (1997) as a much-used strength of evolutionary computation and a driving force in its popularity. Historically, evolutionary computation started in a few independent and distinct approaches, or "schools", namely evolution strategies (ES), evolutionary programming (EP), and genetic algorithms (GA). Each one was inspired by emulating biological evolution, but using different representations and algorithmic operations that were based on different emphases of the philosophical underpinnings of how natural adaptation works (Fogel 1994).

In the terminology used in the latter parts of this dissertation, the simulated *genotype* of a solution individual is the representation used for the variation operations whereas the *phenotype* is a decoded actualization of the encoding that performs the task for which the fitness is evaluated. The phase of *decoding* the genotype into a phenotype before evaluation could be added in Algorithm 1 in order to make this duality between the *genotypic* and *phenotypic* representations explicit.

Even though a single best solution can be aimed for, the population con-

cept of evolutionary computation lends itself naturally to multiobjective optimization, where an approximated sample of the Pareto set is sought after. Deb (2008) overviews the progression from single-objective evolutionary computation to Evolutionary Multiobjective Optimization (EMO) where the population is used as a vehicle to approximate a diverse representation of the Pareto set. EMO using multiobjective evolutionary algorithms (MOEAs) has found much use in the field of data mining and machine learning, recently surveyed by Mukhopadhyay et al. (2014a,b).

This dissertation focuses on applying EMO to the previously introduced supervised pattern recognition tasks using MLP neural network models. Freedom of the algorithmic component selection in MOEAs will be followed as suggested, for example, by Bäck et al. (1997) and hybridized with data representations and local optimization approaches traditional in the study of neural networks. The opinion driving this dissertation is that a current and proper view of such hybrid MOEAs is the framework of memetic algorithms, which is a structured umbrella combining the concepts of EC with those of local optimization methods.

2.4.2 Lifetime Learning and Memetic Optimization

While the early theory of Lamarck (1809) was biologically incorrect in proposing that adaptations or traits acquired during the lifetime of a living individual would be inherited by its offspring through natural reproduction, the algorithmic simulation of such *Lamarckian evolution*, or *Lamarckism*, has proved to be an efficient tool in evolutionary computation. In practice, the concept takes the form of *lifetime learning* that improves an individual with respect to its simulated environment. The learned phenotypic adaptations are then encoded back into the individual's genotypic representation before the evolutionary variation operations of the EAs are applied.

Accordingly, in addition to the evolutionary framework, the other part of the memetic optimization approach are "local search algorithms", which are local optimization methods selected to suit the specific structure of the solutions sought in the optimization task (Neri et al. 2012). These play the part of Lamarckian evolution in memetic algorithms.

The general form of the memetic algorithm from Neri et al. (2012) is recited in Algorithm 2. The evolutionary framework is present here as the population of individuals being operated on to yield every new generation. Instead of the terms *variation* and *selection* of Algorithm 1, the "genetic" or "evolutionary" part now appears as general *cooperation* and *competition* of individuals. The arity of the cooperation operations can vary from unary "mutation"-like operations (emphasized in evolutionary strategies) to "crossover"-like binary (or n-ary) operations (emphasized in genetic algorithms). What is made specific in Algorithm 2 is the hybridization with local *improvement* operation for seeking performance improvement in the local neighborhood of each population individual. The basic version of the general memetic algorithm is single-objective, and the very best individual in the population is returned as the result after the iteration is termi-

nated.

Algorithm 2: A basic memetic algorithm, as presented in Neri et al. (2012).

```
Function BasicMA(P: Problem, par: Parameters): Solution
    pop ← Initialize(par, P) ;
    repeat
        newpop1 ← Cooperate(pop, par, P) ;
        newpop2 ← Improve(newpop1, par, P) ;
        pop ← Compete(pop, newpop2) ;
        if Converged(pop) then
            pop ← Restart(pop, par);
    until TerminationCriterion(par);
    return GetNthBest(pop,1);
```

The memetic framework fully preserves the generality and customizability of traditional evolutionary computation, and also the improvement operation is encouraged to be designed in a customized, problem-specific way. The roots of MAs in discrete optimization problems does not limit the applicability of the general algorithm to other application domains. It is to be noted that hybridization of evolutionary and local optimization methods is not new or specific to MAs, but as the formulation and name have matured and gained sufficient impetus in the optimization community, the decision is made here to base the current work on what can nowadays be safely and unambiguously called the memetic algorithm framework.

Very natural local improvement operations for MLP classifier training can be derived from the traditional gradient-based solution methods such as applying backprop training on Equation (6).

2.4.3 Multiobjective Memetic Algorithms

Even though the basic memetic algorithm is single-objective (with solely the 1st best individual selected in the end), the algorithm can be augmented into a multiobjective version, similarly to the basic EA, by using a suitable MOEA scheme. In this work, the non-dominated sorting (NS) approach of the well-known NSGA-II algorithm (Deb et al. 2002; Srinivas and Deb 1994), based on ideas attributed to Goldberg (1989), is selected. Several alternatives exist, such as SPEA2 (Zitzler et al. 2002) and PESA-II (Corne et al. 2001), but the choice of NSGA-II should be a safe one, since the method is popular and well-tried as of now. Also, the current focus is on the introduction of objective formulations and application-specific operations instead of fine-tuning the qualities of the result.

Algorithm 3 outlines the general Pareto-based multiobjective strategy followed in this dissertation. It combines the generic memetic operations of "Initialization", "Cooperation" and "Improvement" of a population of solution can-

didates with the non-dominated sorting and crowding-distance comparisons of NSGA-II. The algorithm still leaves open the specific definition of which objective functions to consider (problem definition), how to encode the individuals, and the way in which the operations are performed or parameterized. These are the interesting research foci of the rest of this dissertation dealing with variations around these issues.

2.5 Historical Remarks

As both artificial neural networks and evolutionary computation date back to the relatively early days of computing, so do their combinations. In his "Fossil Record", Fogel (1998, p. 482) dates early ideas of evolving neural networks back to Friedman (1956) and Bremermann (1968), and actual implementations to works as early as those of Mucciardi and Gose (1966) and Klopf and Gose (1969). Since the beginning, evolutionary steps have been hybridized with other learning methods as in those early works. Fogel (1998) proceeds to note the joint increase of interest in both evolutionary algorithms and their application in neural networks towards the end of the 1980's and a vast blooming of research later on, documented, for example, in the early reviews of Schaffer et al. (1992) and Yao (1993). From both of the two reviews, a surge of innovative ideas in the late 1980's and early 1990's can be felt.

Schaffer et al. (1992) review the initial decades of evolutionary neural network learning from a utilitarian engineering point of view. They categorize the method combinations to supportive ones and collaborative ones. In *supportive combinations* either the neural network or more often the evolutionary part has the role of supporting the other algorithm in performing some stage of its primary task. Supportive combinations contain early examples of auxiliary or "wrapper" methods that have since become commonplace, such as using an evolutionary approach for feature selection, feature generation, or data preprocessing. Other

Algorithm 3: General memetic MOO with non-dominated sorting.

```
input  : Definition of the problem, encodings, parameters
output: Approximated sample of the Pareto set
parents ← Initialize() ;
parents ← Improve(parents) ;
AssignRankAndCrowdingDistance(parents) ;
repeat
    children ← Cooperate(parents) ;
    children ← Improve(children) ;
    parents ← NondominatedSort(parents ∪ children) ;
until Terminate;
return parents;
```

common supportive tasks were found to be the determination of the network topology and the settings of various learning parameters.

Of much more interest related to the themes of this dissertation are the early examples which Schaffer et al. (1992) call *collaborative combinations* because both sides of the hybrid operate jointly on the common learning task. One class of such collaborative hybrids consists of attempts to use evolutionary computation directly by encoding the synaptic weight values in the chromosome. The review identifies Montana and Davis (1989) as the first authors describing an evolutionary implementation that was able to outperform backpropagation learning in a classification task of considerable complexity (approximately 500 connections). Another class of hybrids, evidently more popular at the time, and anticipated by the writers to be a more intuitive one, consisted of algorithms that use evolutionary computation to learn the neural network topology and use a local learning method to fine tune the performance. This kind of intuitive approach, readily identified in the early days, would fall immediately in the category of memetic algorithms as defined above. Many of the hybrids were reported to have been largely modified from the canonical genetic algorithm that represents individuals as binary strings and uses the exchange of substrings as the crossover operation mimicking biological inheritance. The review concludes that the results from the very first surge of research interest had mixed results related to attempts to improve weight optimization compared to gradient-based methods.

The review by Yao (1993) assesses many of the same considerations as that of Schaffer et al. (1992), although it is more narrowly focused on the task of learning a neural model instead of merely "supporting" parts of the process. It categorises research endeavours based on the scale of operations they concentrate on: the architectural (topology) level, the level of learning rules, and the level of network weights. A conclusion is made that evolutionary computation is likely to be most suitable on the slower and more coarse scales such as for determining the network topology and the learning rules, rather than on the faster and finer scale of weight determination, where local search would fare better. Yet, Yao (1993) suggests that the interactions between evolution on the different levels should be further explored.

Many different fitness functions were identified and used already in the first years, and it was mentioned that the learning can proceed using one of various criteria, but neither one of the two early reviews mentions actual multiobjective optimization (or vector optimization), even as the field had been developing quickly at the same time or even before (Deb 2008). It would seem that while evolutionary learning was hatching, the field was still on an island separated from that of multiobjective optimization, or perhaps the researchers were keeping their initial trials purposely simple by considering only single-objective cases at the time. Jin and Sendhoff (2008) overview multiobjective machine learning research up to the year 2007 with a focus on supervised learning using feedforward neural networks. They date the beginnings of multiobjective learning to the mid-1990's, and as an early work on the topic, they identify the paper by Liu and Kadirkamanathan (1995) in which a radial basis function network is optimized

with respect to three objective functions using a genetic algorithm. The gap between the proliferation of evolutionary machine learning and the appearance of multiobjective formulations seems to have lasted for approximately a decade.

More specific findings expressed in the reviews will be given in Chapter 3. This historical sidenote shall end with the conclusion that memetic hybrids combining evolutionary and local optimization are not new to the community, but multiobjective formulations have appeared relatively late compared to the age of the other branches of computation discussed earlier.

2.6 Example: Multiobjective MLP for Imbalanced Classification

Before continuing with an overview of contemporary research and a more formal introduction of the many possibilities in population-based supervised learning, let us start with a small proof-of-concept demonstration published earlier by Nieminen and Kärkkäinen (2016). The case study deals with the problem of *class imbalance* which was already hinted at in Figure 10 of Section 2.3.2. This "imbalanced learning" scenario is relevant in real-world tasks (He and Garcia 2009) and special techniques are constantly emerging to address and successfully manage it (see, for example, Zong et al. 2013).

As usual, we assume that a training dataset of N observations and corresponding integral target class labels $\mathbf{X} = \{(\mathbf{x}_i, t_i) \mid \mathbf{x}_i \in \mathbb{R}^n,\ t_i \in \{1, \ldots, K\}\}_{i=1}^{N}$ is available, and the task is to use these examples to learn how any $\mathbf{x} \in \mathbb{R}^n$ should be assigned to one of the K classes. By imbalance we mean that the numbers of labeled observations in each class $N_c = \#\mathbf{X}_c$, where $\mathbf{X}_c = \{(\mathbf{x}_i, t_i) \in \mathbf{X} \mid t_i = c\}$, or, equivalently, their frequency of occurrence in the whole dataset, $\varphi_c = N_c/N$, are markedly different from each other. Cost-sensitive learning (Elkan 2001) and ROC curves (Fawcett 2006) are traditional ways of dealing with the situation. In binary classification, different costs may apply to false positive and false negative predictions. A generic example could be that of medical diagnosis: A false positive prediction in an initial screening test would mean that a healthy person would unnecessarily be sent to further laboratory tests that have costs in money and human resources. A false negative prediction would mean that a sick person would be sent home with the conception of being healthy. In the latter scenario, money and resources are saved, but with the cost of possible worsening of the illness because proper treatment has not been started. With more classes, even more complex trade-offs must be addressed.

Clearly, imbalanced classification is an instance of multiobjective optimization whenever a single solution cannot achieve full accuracy for all the classes. Formally, given any K-class classification dataset $\mathbf{X} = \cup_{c=1}^{K} \mathbf{X}_c$, we wish to minimize the classwise numbers of misclassifications

$$f_c = \#\{(\mathbf{x}_i, t_i) \mid \text{prediction}(\mathbf{x}_c) \neq t_c\} \text{ for } c = 1, \ldots, K. \tag{10}$$

When $K > 2$, there is a quadratic increase in the number of costs that could be

considered (predictions to class i when j would be correct), but even the simple classwise errors chosen here suffice to exhibit the merits of breaking the imbalanced classification task down to a multiobjective formulation and applying a population-based memetic algorithm to sample the approximated Pareto set.

Then, as the *problem definition* for Algorithm 3, we consider objective functions defined as the classwise numbers of misclassifications in the three-class classification task. For simplicity, in this initial example, we restrict ourselves to the single-hidden-layer feedforward network (SLFN) as the classifier model and fix the neuronal architecture. In Equation (5), we select the number of layers to be $L = 2$, and we fix the layer sizes as $n_0 = n = 2$ (input), $n_1 = 20$ (hidden layer with logistic sigmoid activation), and $n_2 = K = 3$ (linear output layer; number of classes to separate).

Thus, our set of admissible solutions for the optimization (see Equation (7)) is now $\Omega = R^{n_2 \times (n_1+1)} \times R^{n_1 \times (n_0+1)}$. The *encoding* of such a solution is simply the storage of the synaptic weights as floating point numbers. For the experiments shown here, we use a population of 100 individual solution candidates encoded in this way. Other *parameters* controlling the method steps will be explained below.

We *initialize* the first population by assigning uniform random values from the distribution $U([-1,1])$ as the weights. Then, the first round of *improving* the population is performed by running at most 200 steps of a conjugate gradient method (Fletcher and Reeves 1964) on the single-objective (scalarized) cost function

$$\mathcal{J}(\{\mathbf{W}^l\}) = \sum_{c=1}^{K} \lambda_c \sum_{i=1}^{N_c} \|\mathcal{N}(\mathbf{x}_i) - \mathbf{t}_i\|^2 + \mu \sum_{(l,j,i)\in I} |w_{j,i}^l|^2,$$

where $\lambda_c \sim U([0.1,1])$ and $\mu \sim U([10^{-6}, 10^{-4}])$ are uniform pseudorandom values. The vector $\mathbf{t}_i \in \mathbb{R}^K$ is the usual K-dimensional "one-of-K" binary vector representation where the t_ith component is 1 and other components are 0. The index set I represents all the weights in the MLP. An important role of this traditional local search phase is to restrict the solutions to meaningful MLPs after disturbance by the evolutionary operations that facilitate global exploration. The network weight penalty also imposes a necessary soft constraint. Noteworthy is that the random scalarization frees the user from having to make strong assumptions of the problem before exploring the resulting Pareto set.

In the example here, the *cooperation* step uses only a unary mutation operator that perturbs each weight by noise drawn from the Gaussian distribution $\mathcal{G}(0, 0.4)$ with a probability of 10%, producing a sort of a "creep" in values. Extending the cooperation step with crossover operators designed specifically for MLP classifiers is one main focus in customizing the memetic framework for the application domain.

Figure 12 illustrates the class imbalance problem with a simulated dataset in which $n = 2, K = 3, N = 400, \varphi_1 = 5\%, \varphi_2 = 10\%$, and $\varphi_3 = 85\%$. The case is made difficult by letting all of the classes overlap in the middle, but none of them contain subconcepts or noise (cf. He and Garcia 2009) by design. The de-

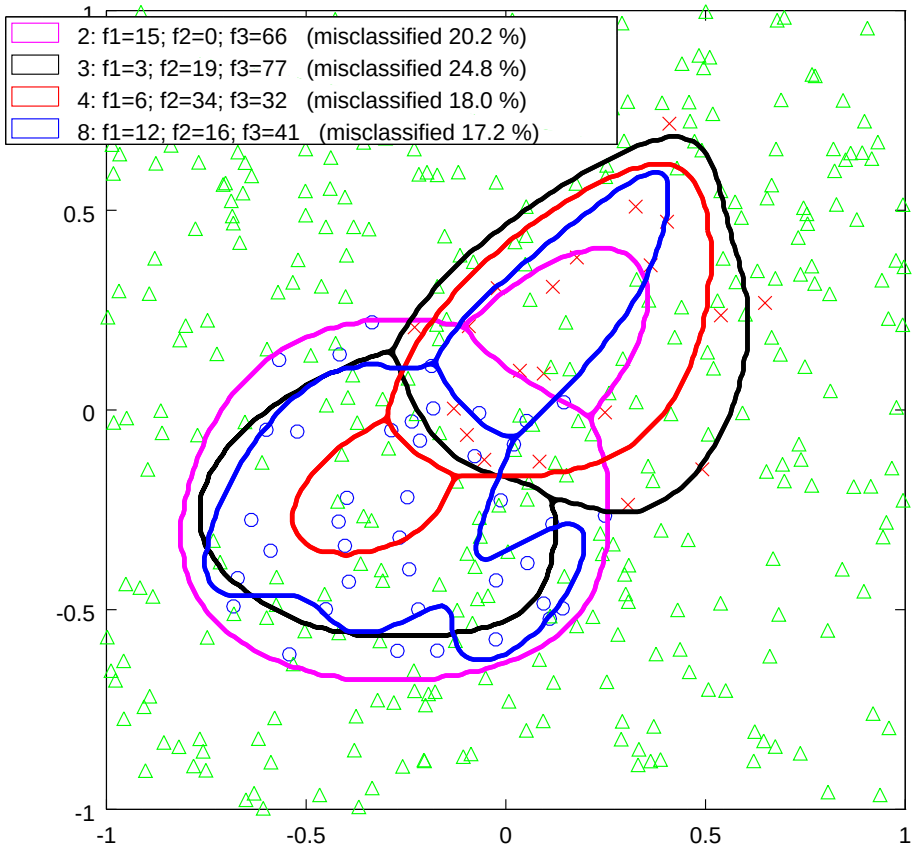


FIGURE 12 Post optimization exploration of Pareto-optimal classifiers.

cision boundaries of different classifier candidates from the approximated PS are also shown, indicated by colors and numerical identity tags in the legend. The legend contains also the three objective function values f_1, f_2, and f_3, as defined by Equation (10), at each solution and in parentheses the traditional performance metric of misclassification rate, i.e., ratio of false classifications, for comparison. Not one of the classifiers can be deemed better than the other without assigning some preferred costs to misclassifications in one class versus the other, so the pure multiobjective nature of the problem can be clearly seen. Another critically important observation is how the overall misclassification rate fails to yield useful information of the quality of the solutions. Figure 13 shows the same classifiers in a parallel coordinate plot where the trade-offs between objective functions can be compared visually. Each line in such a plot represents a solution candidate. The vertical axes correspond to objective functions, and the objective function value can be read from the intersection of the line and the axis. For each individual objective, the best solution crosses the axis at the lowest point. Due to their Pareto-optimality, the lines must cross each other at some points. If a line remained lower than all the others in a parallel coordinate plot, it would dominate the others. In that case, by definition, none of the other points could belong to the set of nondominated solutions.

The solutions were obtained by running the described simplistic memetic optimization algorithm for 100 iterations. Noteworthy with regard to the prospects

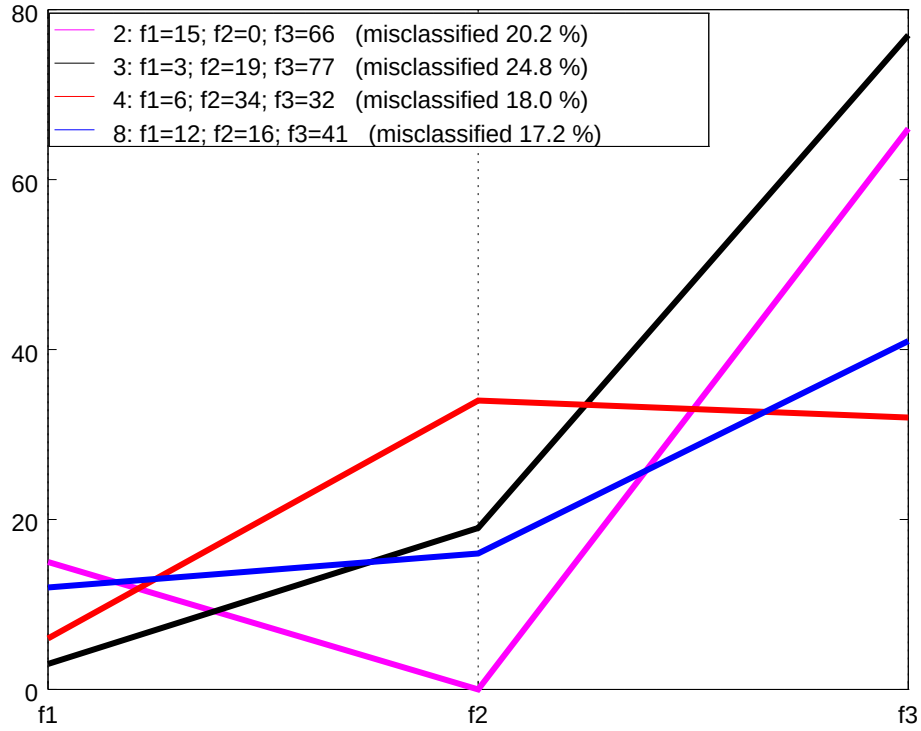


FIGURE 13 Parallel coordinates view of the solutions in Figure 12.

of our approach is the fourth solution (ID tag 8). It contains sharper edges and seems to be overfitting the example data in the lower right region. With less versatile learning frameworks one needs to compete against this kind of behavior by pre-assigning constraints or penalties to the network complexity. In a multiobjective setting, it is natural to add network complexity measure(s) as additional objectives, turning the danger of overfitting into the prospect of gaining valuable information of the complexity of the data itself. The memetic approach in general does not require limitations such as network structure to be set *a priori*, even if it was done in this early demonstration. To really set the framework and its user free, further development of the memetic improvement (lifetime learning) and cooperation (recombination) operators and solution encoding needs to be done that considers the specific properties of MLP networks. Hence the on-going work outlined in this dissertation.

3 OVERVIEW OF RELATED WORK

This chapter examines the existing body of knowledge related to evolutionary (and memetic) multiobjective optimization used for supervised machine learning with artificial neural networks. Evolutionary algorithms for neural network learning before the time of explicit multiobjective formulations are summarized in Section 3.1, the first decade of multiobjective evolutionary learning is covered in Section 3.2, and the last decade up to the time of writing this dissertation is covered in 3.3.

3.1 Evolutionary Learning with Neural Networks

As remarked in Section 2.5, truly multiobjective formulations and solutions of learning tasks started to appear in the middle of the 1990's, and thus they have some two decades of history at the moment. Population-based evolutionary algorithms, many of which could be termed memetic algorithms with a succession of population-wide operations and improvement operations for population individuals, had been used for single-objective learning tasks for about one decade longer. Considerations arising both from the evolutionary and from the multiobjective approach are relevant here. In what follows, some key issues related especially to feedforward neural network learning will be gathered from a chronological outline of secondary studies.

3.1.1 Early Developments and Genetic Representation Issues

In the early days, non-evolutionary algorithms were mostly used for learning the connection weights. Evolutionary changes of the network architecture based on fitness evaluation was done intermittently or before weight learning. The early reviews of both Schaffer et al. (1992) and of Yao (1993) identified the *method of representation* as a key factor in evolving neural networks. In the most *direct* representations, the full connectivity graph of the network is represented as a binary

string that includes the $N \times N$ possible connections of all the N "nodes" of the network (input nodes, computational neurons on possibly many hidden layers, and output nodes). Recurrent feed-back connections (going back from a deeper layer to one closer to the inputs) are possible in such a representation. Evolutionary algorithms provide a framework suitable for learning recurrent structures which have been found useful, for example, in time-series forecasting because of the dynamic "memory" and filtering capacity provided by time-lagged feedbacks. Still, more often recurrent connections (and also those that bypass an intermediate layer) were forbidden in order to simplify the structure and to use more efficient local learning methods to learn the weights.

The obvious drawback in the most direct representations is that they produce very long strings (order of N^2) for large networks, which makes the search space prohibitively high-dimensional for nontrivial tasks. Additionally, the real values of the connection weights could be encoded in the binary string, which would increase the representation size even more, depending on accuracy requirements. The review of Yao (1993), less inclined towards requiring crossover operations, considered also representations in which the connection weights are encoded and evolved as real numbers. Binary strings would nevertheless be necessary if the connectivity graph was to be optimized in addition to the weight values.

A major concern in the bit string representation is the sensitivity to the organization of the bits. The traditional substring exchange applied in the crossover operation is likely to preserve useful coadapted *building blocks* that are close to each other in the bit string. Useful collaboration produced jointly by further-away genes is more likely to be destroyed by being split between separate offspring individuals and thus losing the trait obtained via earlier coadaptation.

To reduce the explosion in search space dimension, *indirect* representations were proposed early on. In their simplest forms, restrictions would be applied so that only some connections, or groups of connections, would be encoded as bits in the representation. Various approaches were cited in the early reviews, such as applying evolutionary variation to only feed-forward connections, connections between one layer and the next, or only on one of the layers. Other examples included applying the evolutionary framework only for selecting one of a few alternative network architectures, the number of neurons on a layer, or the presence or absence of each intermediate layer in a multilayer architecture. Indirect representations were developed also to determine the locations and connectivity patterns of groups called receptive fields, i.e., connections responding to selected parts of the input pattern.

The level of indirection used in the representations varied up to that of using grammars and syntax trees resembling the branch of evolutionary computation called genetic programming (GP) where syntactic structures are evolved. The evolutionary variation operators used in such representations have to be specialized so that the grammatic and semantic structure is correctly preserved. Indirect representations are supported by biological intuition. For example, the number of synaptic connections in the developed human brain is known to be larger, by

many orders of magnitude, than the information contained in the whole biological human genome. Several indirect representation schemes were thus developed to evolve only the rules by which an actual network architecture would be generated upon decoding.

Yao (1993) elaborated on the concept of evolving learning rules (i.e., the weight update formulae in iterative learning methods) in addition to the network architecture or the connection weights. Representation issues were found to be even more difficult for learning rules than for the other aspects of learning. Evolving learning rules, while philosophically intriguing, is not within the range of issues considered in this dissertation, in which some technical implementation constraints from earlier and on-going development work are purposely retained, largely excluding the feasibility of applying different learning rules. Also, the chosen representation will be based on the layer-wise weight matrices of Equation (5).

A notable problem in direct structural representations of neural networks for evolution is referred to as "competing conventions" by Schaffer et al. (1992) and as the "hidden node problem" by Yao (1993). The problem is that the same phenotypic action can result from the combination of genes that reside in completely different parts of the genotypic representation. This is a direct result of the symmetries in the synaptic connection graph. In the case of the MLP formulation presented in Equation (5), the problem can be understood as the invariance of the matrix computations under certain permutations of rows and columns. Naïve crossover operations using direct representations cannot take into account these symmetries, and are likely to destroy coadaptations. The problem can be completely circumvented by emphasizing mutation operators as in evolution strategies (for an overview, see Fogel 1994), or using simple additions and deletions of one connection/weight at a time instead of attempting to simultaneously exchange multiple functional units between individuals.

Published a few years after his first review, Yao (1999) provided an even more thorough review on evolutionary artificial neural networks. During the time, he observed "a great interest" in combining learning and evolution, the "two fundamental forms of adaptation", with artificial neural networks. The main argument of Yao (1999) was that these combinations can lead to "significantly better intelligent systems" than relying on the separate components by themselves. From the earlier review, he maintained the three-fold categorization of evolution used in the different stages of learning: (i) connection weight optimization, (ii) network architecture/topology optimization, and (iii) finding (i.e., adapting or discovering) optimal learning rules. He highlighted the possibility of evolving activation functions in addition to the connectivity topology as part of the network architecture. The benefits of the evolutionary approach observed half a decade earlier remained in the focus. The problem of competing conventions present in direct representation schemes was discussed (as in Schaffer et al. 1992), and some new forms of indirect representations were recognized. The supremacy of hybrid methods combining gradient-based local search and evolutionary global search (which we refer to as memetic here) over either one

approach alone was recognized, too.

Yao (1999) also elaborated further on the prospect of evolving the ways how to learn, i.e., actual learning rules best fit for an ANN to learn the task at hand. While "learning how to learn" is interesting, these considerations are out of the scope of this dissertation for the reasons mentioned above. Multiobjective learning or optimization was not yet explicitly covered nor mentioned in the review of Yao (1999), while multiple objectives, such as the error between target and actual outputs and the complexity of the network were quite clearly mentioned, regularization term was explicitly involved in typical optimization schemes (and the merits of EA further elaborated with regard to nondifferentiable regularization terms), and it was stated that the most common mean square error measure is not the only one that can be used as the primary error measure. While discussing ensembles of ANNs, the ideas presented got very close to Pareto-based multiobjective optimization, and it was even stated that with evolutionary learning, it is "beneficial to make use of the whole population".

3.1.2 Concerns about the Computational Effort

The early works on evolutionary training of neural networks compared the computational performance of these new hybrid algorithms to backpropagation and other gradient-based methods both in terms of result quality and in computational times required. They were partly disillusioned because measurable improvements in results and computation times turned out to be difficult or impossible to achieve. It must certainly be true that for tasks in which gradient-based learning suffices, evolutionary augmentations are not necessary nor worthwhile to be used. But, as also recognized in the early studies, there are likely to be useful scenarios where gradients are unavailable or harder to make use of. Examples include recurrent networks where feedback connections are used in addition to feedforward ones, and the use of any non-differentiable or discrete objective functions or activation functions (like the step function). Evolutionary computation readily allows the use of any form of fitness functions, and resultingly also any form of activation functions and connectivity patterns within the neural network. A more mundane benefit of evolutionary computation is its natural capability of escaping local minima that plague gradient-based learning methods, especially in network architectures with fewer connections.

The consideration of computational cost in terms of time is, of course, always important in machine learning, and in all other computing, for that matter. But when dealing with tasks that are impossible or unreliable to perform with fast and direct algorithms, the computational cost becomes much less significant. Also, we have come to trust that the speed of computer systems increases exponentially, so the computational cost of algorithms with a polynomially bounded time complexity will be amortized by faster equipment in a matter of years for problems of constant size. It could be argued that with the current trends of massively parallel computation platforms being available "in the cloud" and as-a-service, such parameters as the size of a multistart batch or the population size in

evolutionary frameworks, or other complexities that can be trivially "parallelized away", become ever less significant.

The main theme in this dissertation is the multiobjective formulation of supervised learning, for which evolutionary approaches bring further and incommensurable benefits compared to the single-objective considerations of the first decade of evolutionary learning covered hitherto.

3.2 Multiobjective Learning: The First Ten Years

A broad and detailed account of multiobjective machine learning is found in the textbook edited by Jin (2006), comprised of chapters written by many authors, including those cited also in the survey part of a later article by Jin and Sendhoff (2008). The first ten years of multiobjective machine learning is well covered in these resources, so it feels rather justified to use them as the basis of the historical overview presented next. Significant developments starting from the earliest days of multiobjective supervised learning are presented next in the form of digests of original research papers suggested by Jin and Sendhoff (2008). Overall, the main resources cited (Jin 2006; Jin and Sendhoff 2008) do consider all the facets of machine learning outlined here in Section 2.1.1, observing the multiobjective nature of all these learning tasks. In the digests below, the emphasis is on supervised learning especially with MLP-type neural networks (defined in Section 2.2) using evolutionary multiobjective algorithms that approximate the Pareto front. All articles highlighted in the survey of Jin and Sendhoff (2008) for these properties are covered here. Necessity was felt to involve also some of the other works on related strutures and tasks, for completeness.

3.2.1 The Bigger Picture: Where MOO Matters and Why

Jin and Sendhoff (2008) categorized learning algorithms as *single-objective* ones, *scalarized* multiobjective ones, and *Pareto-based* multiobjective ones. They presented single-objective and scalarized multiobjective learning similarly to the approach taken in Sections 2.2 and 2.3, identifying approaches such as minimizing the objective function of Equation (6) as scalarized multiobjective optimization. Any number of objectives can be aggregated into a scalarized function using weight coefficients for each objective. The Pareto-based approaches are those that result in an approximation of the Pareto set for the decision maker (or further algorithms) to examine *a posteriori*. As a merit of Pareto-based machine learning, Jin and Sendhoff (2008) mentioned the possibility of *extracting knowledge* about the problem in order to make a better final decision among the solutions in the Pareto set.

The scalarization approach has several limitations. For example, Das and Dennis (1997) demonstrated how scalarization as a convex combination of objective functions always fails to match the global minimum of the aggregated

function with any solution that lies on a non-convex part of the Pareto front, should such forms arise from the shape of the objective functions or the feasible set of admissible solutions. Secondly, obtaining an even distribution of points on the Pareto front necessarily requires prior information about the curvature of the front, even when it is nicely convex. Jin et al. (2001) gave a further geometrical explanation of the Pareto front convexity issue, and showed how a slowly progressing optimization algorithm can indeed advance through the points in concave parts of the Pareto front, and how a dynamic, "slow", change in scalarization weights (called dynamic weighted aggregation, DWA, by the authors) can be used to sample both nonconvex and convex areas of the Pareto front. It is to be noted that DWA effectively results in a Pareto-based method, since an archive of non-dominated solutions is generated while the algorithm proceeds with dynamically varying scalarization weights. However, the method produces essentially one-dimensional trajectory curves through the search space, so its use is somewhat restricted to bi-objective optimization where the Pareto front is a curve instead of a multidimensional surface. Another way of finding solutions in non-convex parts of the Pareto front is the Tchebycheff metric (see, for example Miettinen 1999, p. 98) which is used in recent population-based MOO methods, such as the decomposition method proposed by Zhang and Li (2007).

Jin and Sendhoff (2008) note also the problem of local optima in which optimization methods may get stuck. They refer to the works of Abbass (2003b) and Teixeira et al. (2000) as early examples of using Pareto-based optimization to help escape local optima. The former of these cited works does directly refer to the problem of local solutions and the inherent ability of evolutionary optimization to avoid these. The latter improves MLP generalization over standard backprop learning via the use of multiple objectives, but the relation to local optima is not explicitly addressed.

Jin and Sendhoff (2008) argue in favor of Pareto-based approaches as a "natural idea" in machine learning. For the motivations of Pareto-based optimization, they identify the following three main categories: *generalization improvement*, *interpretability enhancement* (in rule extraction), and *diverse ensemble generation*. Additionally, they do mention Pareto-based *ROC analysis*, multiobjective *systems control*, and multiobjective *feature selection*.

3.2.2 The First Appearances of Multiobjective Learning

Liu and Kadirkamanathan (1995) acknowledged the connection between learning and function approximation, and observed that as of that time the common procedure was to aggregate the computationally simple mean square error and some model complexity measure to deal with approximation and generalization jointly. They formulated neural network learning (although not for MLPs but for Volterra polynomial basis function networks and Gaussian radial basis function networks) as a truly multiobjective problem in which three separate functions were considered: (i) the mean squared error, (ii) the maximum error, and (iii) the number of basis functions or polynomial coefficients in the network. They ob-

served that "these objectives can sometimes be conflicting and no solution may exist that optimises all the objectives". The conflict between an error norm and complexity is commonplace, and the two error norms are related to the distribution of the noise in the process and measurements being modeled (use of mean squared error presumes normal distribution of noise).

The optimization was based on the method of inequalities (Zakian and Al-Naib 1973) which produces one solution at a time, requiring *a priori* settings of preferences by the decision maker. The solution gained in such a way is only aimed to be sufficient with regard to the preferences (the upper bounds of acceptable values for each objective), and not necessarily Pareto optimal. Nevertheless, the paper remains notable as certainly one of the earliest works that explicitly considered learning as a multiobjective problem. Noteworthy is also the consideration of more than two objectives at once.

An evolutionary approach was used where the population individuals were represented as a combination of a binary vector indicating the existence of each model term and an array of real values representing the centers of the radial basis functions. Output layer weights for the radial basis functions were determined using the traditional least squares method during each population evaluation.

Kottathra and Attikiouzel (1996) considered the approximation of a real function using a feed-forward neural network with a single hidden layer of variable size, although the method could be extended for more involved structures. They formulated the approximation task as a bi-objective optimization problem where one objective was the mean square error (MSE) and the other one was the number of active neurons taking part in the computation, i.e., the hidden layer size. In essence, the objectives were the same as those used by Liu and Kadirkamanathan (1995) with the exception of not including the maximum error, and the neural model was based on a sigmoidal activation function (as in the MLP applied in this dissertation) instead of polynomial or radial basis functions. A branch-and-bound strategy (for method origins, see Land and Doig 1960) was applied as the optimization algorithm. Multiple objectives were handled by means of goal programming (Ignizio 1976) which, similarly to the method of inequalities applied by Liu and Kadirkamanathan (1995), provides a single solution that satisfies pre-determined preferences (objective-wise goals).

Matsuyama (1996) dealt with unsupervised learning. He modeled neurons as "competitive agents", showcasing solutions to the unsupervised learning tasks of vector quantization and vehicle routing. He recognized multiobjective optimization as the aggregation of the "main cost" (approximation error) and "subcosts" (constraints, including regularization), and explored the benefits of applying multiple costs. He presented a method for adjusting the scalarization of subcosts dynamically while the optimization is taking place. The goal was to achieve a single solution, but with fewer *a priori* preferences on the subcost weights, and with benefits to convergence characteristics. Already earlier, Matsuyama et al. (1994) had observed the relation of penalty terms and multiobjective optimization methodology in both supervised and unsupervised learning, although the Pareto set remained as a theoretical frame of reference, and a single solution op-

timizing only the primary objective was pursued.

Teixeira et al. (2000) proposed an adaptation of the ϵ-constraint scalarization method (Takahashi et al. 1997) to simultaneously minimize the MSE error of an MLP model and the squared sum of the connection weights. The objectives were converted into constraints for a single-objective optimization algorithm. The approximated Pareto set was sampled through varying the constraints gradually. In simple classification and regression examples, their optimization yielded MLP networks that showed performance comparable to SVMs and much better generalization than that obtained using single-objective backprop training of an MLP. The final solution was presumably selected based on minimal validation dataset error from among the sampled Pareto optimal solutions, which seems to be somewhat more clearly stated in a later publication (Costa et al. 2003, p.468) than in the original one.

3.2.3 Towards the Pareto-based Approach

Abbass (2001) claimed that, up to that time, "all of the research undertaken in the EANN literature ignores the fact that there is always a trade-off between the architecture and the generalization ability of the network". This is a strongly made claim, but it is likely to be mostly valid based on the apparent lack of explicitly multiobjective points of view in the early literature on evolutionary artificial neural networks (see Section 3.1). To amend the situation, he provided an approach called *Memetic Pareto Artificial Neural Networks* (MPANN), including the word "Memetic" coined by Moscato (1989) and the word "Pareto" already common in the MOO community. As far as the author of this dissertation can tell, this rather fitting name has been coined in the work by Abbass, although he also used it as the name of the specific algorithmic implementation instead of the class of algorithms it belongs to. It must be noted that originally Moscato (1989) just proposed to use the fitting word "memetic" for the already existing paradigm of hybridizing global and local search to solve optimization problems.

Abbass (2001, 2002, 2003b) assumed a Pareto-based approach, meaning that non-domination plays a role in the survival of individuals between generations in the evolutionary framework. The "memetic" flavor comes from the integration of a gradient-based local search (backprop) to improve the individuals while evolution is also taking place on the outer level of the algorithm. As the multiobjective evolutionary framework, Abbass used Pareto Differential Evolution (PDE) (Abbass et al. 2001) based on the single-objective Differential Evolution of Storn and Price (1997). He used a direct encoding of the connection weight matrix. He mentioned that feedback connections could be supported, but the documented tests were restricted to only feedforward variants being used. The participation of each hidden neuron in the performance evaluation was also represented as a binary vector of fixed length. The maximum number of hidden neurons was thus fixed. The weights and the architecture were both included in the chromosome representation and could be evolved simultaneously.

The two objective functions used by Abbass to highlight advantages of the

Pareto-based approach were the accuracy of the ANNs (via minimizing squared error) and the complexity (via minimizing the number of active neurons, i.e., sum of the non-zero elements of the binary vector representing the on/off state of each hidden neuron). Abbass (2003b) explained that these two were selected for the sake of simplicity from among many alternatives, as the main suggestion was to separate the measures of accuracy and complexity in a MOO sense, regardless of the choice of the specific measures as the two objectives.

Abbass (2003b) claimed that the Pareto memetic approach using PDE and backprop improvement is faster than traditional backprop, but this involves consideration of human effort in addition to computation. Fewer trials are needed, because the network architecture and even local search parameters can be evolved as part of the hybridized algorithm. Also, an evolutionary algorithm can naturally avoid local optima that may slow down gradient-based optimization or require re-starts from different randomized starting points. Case studies with three real-world benchmark datasets showed improvements over other then-current algorithms. Abbass also postulated, although without further investigation, that the multiobjective approach helps in maintaining population diversity better than single-objective evolutionary algorithms (EAs), which enables the population size to be safely set smaller than in a single-objective EA.

The computational experiments did show remarkable reduction in the number of MLP evaluations required in comparison to single-objective EA and also to backprop. In two out of three test cases, the best results were obtained with a self-adaptive version (called SPANN by the author) that evolved the parameters of the evolutionary operators along with the solutions. In the third test case self-adaptation yielded worse results, so its benefits remained inconclusive. The model with the best accuracy was selected in these studies, so multiobjective optimization was used only as an intermediate tool to improve speed and accuracy compared to single-objective approaches. Further exploration of the Pareto front, with possible applications to ensemble generation, were identified in the conclusions early on, though.

3.2.4 Further Explicit Assessments of Model Complexity

Jin et al. (2004b) addressed the regularization of neural networks as a multiobjective optimization problem by separating the objectives of minimizing the error and the model complexity. The error objective was chosen to be the MSE over the training dataset, although the existence of several other possibilities (such as Minkowski error and cross-entropy) was acknowledged. The complexity objective was suggested to be chosen among several possible regularization terms, common ones being the Gaussian regularizer, i.e., the mean of squared weight values, and the Laplacian regularizer, i.e., the sum of absolute weight values. Since evolutionary methods do not require differentiable nor even continuous objectives, a discrete objective function, namely the number of non-zero synaptic connection weights, was proposed and examined.

Direct representation of the weights (as real values) and the on/off state of

synaptic connections (as binary values, one for each connection) was employed. Multiple objectives were handled using DWA (Jin et al. 2001) and NSGA-II (Deb et al. 2002), the performances of which were compared experimentally. It turned out that DWA fared better for small population sizes, whereas NSGA-II was better for large population sizes. It was noted that despite its elitism between two generations, NSGA-II may forget some non-dominated solutions during the run, and the use of a complete solution archive was recommended to keep track of all non-dominated solutions.

One out of five mutation operators tuned for neural network training was randomly selected and applied to each individual in a generation: (i) addition of a neuron, (ii) deletion of a neuron, (iii) addition of a synaptic connection, (iv) deletion of a connection, and (v) a Gaussian creep operation. Crossover (known to be difficult for ANNs for reasons explained above) was not used. A "lifetime learning", i.e., a memetic improvement step in the terminology used here, was applied after evolutionary mutation by running the iRprop$^+$ algorithm (Igel and Hüsken 2000). The local improvement was single-objective, only minimizing the error objective.

Only two objectives were selected for each experiment, so a prior decision was made between the possible error functions and the regularization/complexity measures. The results obtained using the different bi-objective selections were then compared. The authors found that when evolution is applied to the binary connectivity pattern, there is not much difference between using the Laplacian and the Gaussian regularizer as the second objective, contrary to what had been found earlier in the case of gradient-based methods. Both regularization terms were found to grow hand-in-hand with the number of connections. In the single case study provided, the approximation of the three-dimensional Ackley function (Ackley 1987), tests on a validation data set showed different behavior of the different complexity objectives, leading the authors to conclude that "the relationship between model complexity and generalization is not as simple as we might imagine". While examining the results, a question might rise, whether the different measures are mutually conflicting, perhaps depending on the task, and whether they should be included as additional objectives instead of selecting one.

3.2.5 Ensembles

Abbass (2003a) applied his MPANN method to multiobjective ensemble generation. Based on earlier work on evolutionary ANN ensemble generation (Liu et al. 2000), he proposed two ways of formulating multiple objectives for the ANN learning to yield an ensemble of different ANNs that would perform better than an individual ANN. The first formulation was to separate the training dataset into two (stratified) subsets, and to minimize the prediction error on both subsets. The idea would be that an individual network overfitting one data subset would have a large error on the other subset. It was noted that if a network architecture is complex enough, it would be possible to overfit both subsets, thus nullifying the benefit of using multiple objectives. A modest network architec-

ture was therefore chosen *ad hoc*. No investigation was done on the possibility of combining network complexity as a third objective. The other formulation was based on minimizing the standard MSE error (over the whole training dataset) and a "blurred" version where Gaussian noise was added to the MSE.

Using either one of the two-objective formulations and the MPANN algorithm, a Pareto set would result with diverse ANNs usable as an ensemble. As a method of ensembling the Pareto set, Abbass tried three alternatives: (i) majority vote (i.e., most of the ANNs in the network agree on the result), (ii) winner-take-all (i.e., the ANN with the largest activation is allowed to make the decision), and (iii) simple averaging (i.e., the mean activation of the ANNs is aggregated as the final prediction). Results on two datasets showed results comparable to earlier studies based on penalty term aggregation, with the stratified subset errors being the better pair of objectives but no great differences between the actual ensembling methods. The point was highlighted that the Pareto-based selection scheme automatically determines the ensemble members whose diversity is presumably good because of the population-based evolutionary multiobjective approach.

Jin et al. (2004b) suggested that the Pareto set of ANNs obtained by multiobjective minimization of error and complexity can be used also for generating ensembles. Ensemble generation was presented as a potentially useful by-product of Pareto-based multiobjective regularization, because a number of networks with diverse structures are naturally provided. They constructed methods using all the non-dominated solutions and using only a (heuristically selected) "representative" subset. Further evolutionary optimization based on validation set error was used to find coefficients for individual networks in the ensemble. In another paper (Jin et al. 2004a), the same authors elaborated further on the ensembling of neural networks using multiobjective evolution, and observed that "most diversity based methods for generating ensembles can also be seen as a kind of regularization techniques". This can be interpreted as a two-way bridge between methods of regularization and ensembling; one can be obtained by using methods originally created for the other.

3.2.6 Risk and Return

Fieldsend and Singh (2002) extended on the methodology proposed earlier by Kupinski and Anastasio (1999) using a presumably better MOEA and applying the system to real-world financial time-series data, whereas Kupinski and Anastasio (1999) had used a simple synthetic test problem.

In the financial application domain, strengths of Pareto-based methods are readily available to address the main objectives of minimizing the risk and the return jointly, in some cases for the purpose of diversifying real-world investment actions using different parts of the Pareto set. The capital asset pricing model (CAPM) of Sharpe (1964) (textbook explanation found in Brealey et al. 2011, p.193) was used in the application of Fieldsend and Singh (2002). They considered three hypothetical decision makers ("risk averse", "profit maximiser", and "middle-way"), i.e., persons in the business of trading with different pref-

erences regarding the forecasting models that they wish to obtain from learning the time series recorded up to the current trading day. The risk was modeled as the root mean square error (RMSE) of predictions and the return was modeled by actualised monetary return when a pre-determined trading strategy was used and the simulated transactions performed using the actual data.

The authors were able to conclude that the Pareto-based approach for learning of financial forecasting ANNs produced models reflecting the needs of all the various decision makers, without *a priori* setting of objective weights (implying prior knowledge of the Pareto front shape). They also mentioned that further work would be required for the additional problem of generalization, which was not addressed by them or in the prior study by Kupinski and Anastasio (1999). They noted that generalization plays a large role especially in the noisy domain of financial forecasting. As another topic requiring further research, also related to generalization, they mentioned the optimization of the ANN architecture. For the record, they used a fixed network architecture with 5 sigmoidal neurons on a single hidden layer, 10 time-lagged input variables and 5 recurrent input units that received earlier time-lagged predictions.

The study was later extended by Fieldsend and Singh (2005) to daily time series of 37 international financial indexes lasting over multiple years. The earlier method was also developed further, and extensive analysis on its performance was provided. The two objectives of risk and return were still considered, but this time the risk was computed from the outcome of a trading strategy instead of using prediction error as a proxy, and transaction costs were factored into the objective function evaluation making the case more realistic than many others published, according to the authors.

Three general method variants called *Pareto evolutionary neural networks* were proposed and compared in the 2005 paper. Direct representation of the weights (as real values) and the existence of each input and hidden neuron (as binary values) in a strictly feed-forward MLP was used as the chromosome. Ten time-lagged values of a profit/loss time series computed based on the original time series and the trading strategy were used as ten inputs to the MLP. The binary representation facilitated feature selection, since each input node had a bit in the representation along with a bit for the existence of each hidden neuron. All connections between the 10 inputs (maximum), 10 hidden neurons (maximum) and one output neuron, including biases, were encoded resulting in $(10+1)\times 10+(10+1)=121$ real valued weights.

The variants included a "standard" learning approach and two variants that proposed ways of injecting model validation into the process of evolutionary learning. The first of the two "non-standard" variants was based on maintaining a separate archive of solutions that were non-dominated via evaluation against a validation set which took no part in the main evolution. The other variant used n bootstrap subsets of the training data generating nm objective values instead of the original m objectives, since the models are likely to yield different approximations on different partitions of the original training data. The worst of the n alternatives were selected as the final objective values, supporting generaliza-

tion by not "trusting" overly optimistic estimates possible on some subsets of the data.

Selection was based on partitioned quasirandom selection (PQRS) (introduced in Fieldsend et al. 2003). Evolutionary operators included bit mutation of the connectivity variables, i.e., addition / deletion of an input / hidden neuron, and perturbation of the real values by adding a perturbation drawn from some distribution (Gaussian, Laplacian, or other user-specified distribution) to randomly selected weights, and the possibility of deleting a weight by setting the real value to zero. Re-enabling of a formerly deleted weight was up to the perturbation operation. As such, there was no need for a separate binary string to represent the existence of connections. Crossover was not used. Local improvement via gradient-based methods was not used in the financial forecasting case study. In a preliminary function approximation example, backprop was used as part of the population initialization to seed the first generation, but for the real-world forecasting task, random initial population without auxiliary improvement was created.

The benefits of Pareto-based multiobjective learning reported in the earlier study (Fieldsend and Singh 2002) were still highlighted in the extended follow-up: knowledge gained from examining error interactions, and the opportunity of selecting either a single individual with a preferred trade-off or a group of different models. In addition, Fieldsend and Singh (2005) were able to give convincing empirical evidence that re-formulating an originally single-objective optimization problem as a multiobjective one can improve the search process, providing better results with less computation compared to the single-objective formulation.

Although related more to function approximation than to classification, it seems that the two objectives of risk versus return provide a very interesting application area for multiobjective learning where the Pareto-based approach is quite a natural one and where feed-forward MLP networks seem like suitable forecasting models.

3.2.7 ROC Curves

Already while focusing on financial time series forecasting, Fieldsend and Singh (2002) asserted that the paper by Kupinski and Anastasio (1999) was the first one in which a population of ANNs was trained with evolutionary optimization to produce an explicit Pareto surface that can serve the purpose of receiver operating characteristics (ROC) analysis, i.e., to find a compromise between sensitivity and specificity in a diagnostic case. They also claimed that no other research on the topic existed at the time, which may well be the case, similarly to the accuracy versus generalization considerations possibly starting with the works of Abbass (2001).

Everson and Fieldsend (2006) extended earlier work on the connection between the Pareto front in the two-objective space of sensitivity vs. specificity and ROC curves (for example, Kupinski and Anastasio 1999) by considering multiple

classes. Instead of only the true positive and false positive rates used to create the traditional ROC curve, they considered scenarios with K classes and the whole confusion matrix consisting of K^2 elements c_{kj} that indicate the number of input data points classified to class j when the correct class would have been k. The diagonal of the matrix contains correct classifications, which bear no cost. For the other $K \times (K-1)$ elements, a different cost λ_{kj} could be assigned, depending on the task. They proposed that evolutionary Pareto-based learning could be used to optimize all of these errors simultaneously, and that the resulting Pareto front could be interpreted as a multidimensional ROC surface from which the decision maker could select a preferred compromise model, and gain insight to the classification problem and the range of possible solutions obtainable.

The quadratic increase of costs is known to be difficult both for optimization methods and for the human decision maker, so Everson and Fieldsend (2006) also proposed reductions such as considering only the overall misclassifications for each class (as in the example of Section 2.6). With three classes, the Pareto surface embedded in three-dimensional space can still be plotted for human examination. They also proposed a visualization that uses the color of the plot markers to indicate the most common direction of misclassification from one class to another. The paper included such a visualization usefully reducing information of the 6 objectives resulting from a three-class scenario. For more than 4 classes, things would already get more difficult, and it was anticipated that additional *a priori* restrictions to the costs would be necessary.

As the means of comparing different classifier structures based on the ROC surfaces (Pareto fronts) obtained, the "area under curve" (AUC) method, traditional in ROC comparisons, was extended to multiple dimensions. An observation was made that any model better than random guessing should have its multiobjective solution vector inside the volume that lies between the origin and and the simplex resulting from random guessing (with the simplex vertices at unit distance from the origin, along each coordinate; i.e., points in which maximum misclassification takes place along one of the $K \times (K-1)$ possible directions of misclassification). Then, the performance of a Pareto front (ROC surface) A compared to that of front B would be defined by a non-symmetric measure $\delta(A, B)$ by computing the volume of the simplex dominated by solutions in A but not by those in B. Monte Carlo simulation was used to compute the volume. The surfaces may cross each other, similarly to how two-dimensional ROC curves can cross each other, so the measure is not symmetric, but it can be used to quantify how much a Pareto solution A is better than B in *some* areas of the objective space volume.

In computational experiments on simulated three-class data, an MLP neural network was found to be better (according to the comparison measure) than k-nearest-neighbor (k-NN), but it was also noted that the selected MLP architecture had 33 parameters whereas the k-NN had only two, so the winner was no surprise in this case. Technically, an evolution strategy using only mutations was employed, and the structure and number of parameters of both models was fixed. The MLP variant was initialized using local search before evolution but local im-

provements were not used later on. Again, PQRS (Fieldsend et al. 2003) was used for selecting individuals. It was noted also in this research that a complete archive of all Pareto-optimal solutions found during the run is beneficial, and, for the computing equipment available in 2006, cheap enough with respect to computation and storage. As an example, after running for 10000 iterations, the solution archive would contain approximately 7500 mutually non-dominating solutions.

Published during the same year, Gräning et al. (2006) addressed the question of how to manage the issue of generalization in classification tasks based on evolutionary multiobjective ROC curves. The method was based on perturbing the training patterns while only the two objectives of binary ROC, i.e., true positive rate and false positive rate, were explicitly optimized. Algorithmically, the direct encoding of MLP and NSGA-II -based multiobjective evolution strategy based on mutation and lifetime learning without crossover was used, as in earlier works of the authors. Again, multiobjective optimization was shown to be better than aggregation via objective weighting, and noise injection, i.e., the blurring of training patterns with Gaussian noise, embedded within the algorithm, was found to be a viable way of maintaining generalization while locating the Pareto-optimal solutions forming an ROC curve.

3.2.8 Cooperative Coevolution Improved using MOO

García-Pedrajas et al. (2002) introduced a multiobjective extension of cooperative coevolution (Potter and Jong 1994) for ANN learning, exemplified by solving classification tasks with MLPs. In cooperative coevolution, each individual of the main population is built by joining together subcomponents in a fixed arrangement. Each subcomponent is interchangeable with any subcomponent of the same "species", i.e., an individual of the same subpopulation. In simple optimization of a function of many variables, one subpopulation could play the role of a single variable to be optimized. García-Pedrajas et al. (2002) used parts of an ANN (a subset of neurons and connections) as the interchangeable subcomponents.

The synaptic connectivity patterns at subcomponent interfaces were defined so that a subcomponent could be meaningfully interchanged with any other subcomponent of the same subpopulation. Within the subcomponent, the number of neurons and connections was allowed to be freely changed, so the optimization of both the architecture and weights of an ANN was technically possible even with only one subcomponent. Benefits of using many subcomponents were demonstrated by running the same algorithm both with multiple subpopulations and with just a single one.

In cooperative coevolution, the task-specific objective functions can only be evaluated for individuals of the main population, so for separately evolving the subpopulations, credit for the total fitness must be assigned to the subcomponents. Credit assignment is not straightforward, and it is precisely this phase that was improved by a multiobjective approach. In total, seven objective functions were proposed: (i) the difference between the performance of networks with and

without the evaluated subcomponent (the authors did not specify how performances were combined, but an obvious choice of computation would be either the sum or average over all the networks currently using the subcomponent), (ii) the average performance gain/loss in k currently best networks if the evaluated subcomponent was used instead of the one currently in use, (iii) the average performance of all the networks in which the evaluated subcomponent is currently selected, (iv) the number of internal nodes, (v) the number of internal connections, (vi) the sum of internal weights, and (vii) the number of networks in the main population in which the evaluated subcomponent is currently participating. Of these, only the third objective was regarded as "traditionally used" in cooperative coevolution. The others were introduced by the authors to enforce competition within and between subpopulations, and to regularize the overall solution. The signs of objectives (iv–vi) were changed to maintain the maximization point of view in the paper. Pareto-based selection adapted from the idea of NSGA (Srinivas and Deb 1994) took place within subpopulations. In computational experiments on classification datasets, at least five of the seven objectives were used for all datasets.

A multiobjective approach was used also at the level of the main population, in that the classification performance was taken as one objective and the fitnesses of each of the subcomponents as additional objectives. This was done to "encourage the combination of the best individuals" of the subpopulations. The main population objectives were thus related to the quality of the coevolution process, while the objectives related to regularization were embedded in the subpopulation optimization as an intermediate tool.

The main population was represented as a string of integers that labels the selected participant subcomponent from each subpopulation. A classic genetic algorithm variant was used for optimizing this integer string. The evolutionary variation operations used on the level of subpopulations (ANN components) were the addition of a neuron, deletion of a neuron, addition of a connection, deletion of a connection, and a perturbation of weights by random values. Crossover was not used, due to the known problems with architectural crossover.

Convincing experimental results showed improved classification accuracy, especially with generalization to unseen validation data, compared to other popular classification methods, each with hand-tuned "best" parameters. It was also shown that using multiple subpopulations yielded better results than using the same algorithm for only one subpopulation, i.e., without coevolution.

3.2.9 Improvements for Real-time Systems

Wiegand et al. (2004) applied memetic Pareto-based multiobjective learning of ANNs to implement face recognition as part of a commercial video surveillance system. For the particular subcomponent responsible for the face recognition, the task was reduced via extensive preprocessing to a simple one: to tell whether a 20×20 pixel image contains an upright human face or not. This should be done as fast as possible, due to understandable real-time constraints. Critical to meet-

ing such requirements is that the ANN model has the fewest number of neurons possible while still being able to sufficiently classify between the positive case (face) and the negative case (non-face image). The number of synaptic connections was found to be insignificant, since the real-time computational expense was found to be proportional to the number of hidden neurons instead of the number of connections.

The application described by Wiegand et al. (2004) is an example of a situation where an application-specific objective needs to be considered from an engineering point of view rather than from an information theoretical one. The reduction of the number of hidden neurons was even mentioned to be the "primary goal" of the real-time system, which is different from the usual approach of prioritizing accuracy first and other objectives as secondary ones. Let us note here that this can be seen as an example of the different needs of different decision makers regarding multiple criteria in otherwise similarly structured problems.

Due to the real-time requirements and known properties of the application, special constraints on the network architecture were imposed. Receptive fields (i.e., small sets of input pixels localized together on the image plane) were favored instead of fully connecting each input to each hidden neuron. All results were compared to a reference architecture handcrafted by an expert and trained and validated with 100 runs of gradient-based learning. By means of receptive fields, the reference used only 2905 connections out of more than 20000 that would be used with full connectivity. The hidden layer size contained 52 hidden units feeding in from rectangular subsets of the 400 pixels. The engineering goal was then to obtain alternative models with fewer hidden units, allowing faster recognition, but still with acceptable accuracy.

Eight evolutionary operations quite similar to those in other studies were used: (i) addition of a connection, (ii) deletion of a connection, (iii) addition of a hidden neuron, (iv) deletion of a hidden neuron, (v) perturbing ("jogging") the weight of a connection, (vi) addition of a new receptive field (random rectangular area of the input image), (vii) partial deletion of a receptive field (randomly delete horizontal or vertical sets of input connections), and (viii) adding of a hidden neuron along with a randomly generated rectangular receptive field connected to the new neuron. The operations were selected at random, with probabilities that were adapted during the optimization process using a method that takes into account the improvements gained from each operation during previous phases of the learning.

While architecture minimization was handled by the evolution, at most 100 iterations of the gradient-based iRprop$^+$ algorithm (Igel and Hüsken 2000) was used as memetic improvement between generations. Early stopping based on a validation set was used in the improvement phase. The improvement was done by minimizing only the MSE error of the training dataset, but the trajectory of the iteration was tracked and the point with minimal sum of the MSEs of training and validation sets was picked as the final improvement to be stored in the genome. An observation was made that this scheme might end up overfitting the union of the train and validation sets, and a final model should be selected using yet

another unseen set after the optimization is finished.

The Pareto-based selection mechanism of NSGA-II (Deb et al. 2002) was found to perform better than tournament selection using a linear aggregation of objectives as a fitness function. In the NSGA-II version, two discrete-valued objectives were used: The number of hidden neurons and the misclassification percentage in the union of the training and validation datasets. The objectives for the evolutinary framework were thus different from the continuous objective used for local improvement. In addition to the evolving population (of size 25), an archive was kept of all non-dominated solutions found during the whole run.

A speed increase of 50% was reported compared to the reference architecture. Accuracy was given relative to that obtained by the reference architecture. No remarkable degradation with respect to test set accuracy happended while the hidden layer size was reduced from 52 to 25. A notable observation is that the solutions could contain fewer neurons but more connections than the reference model (for example, 3496 connections with hidden 25 neurons), which would not be possible by simply pruning the reference.

The final model selection was made by examining the multiobjective tradeoffs using the classification error (i.e., the percentage of wrong predictions) on an unseen test set. The archived solutions had not been evaluated with this set before, so solutions originally non-dominated with respect to the objective functions others could well be dominated by each other in this final examination phase using different metrics. Visualized solution sets were used to compare the methods and solutions. Quantitative comparison of the solution sets found with different algorithms was done using a suitable set comparison measure (Zitzler et al. 2003).

3.2.10 MOO for Alternative Learning Models

The digests presented above focused mostly on Pareto-based multiobjective supervised learning of feed-forward neural networks, specifically MLPs for classification purposes. For completeness, it must be noted that the same approaches have naturally been widely applied to learning structures other than MLPs. For example, Kim (2004) evolved decision trees using subtree swaps and different mutations of the tree structure. The two conflicting objectives were the classification error rate and the number of rules in the tree. A Pareto optimal set was created using non-dominated ranking (Fonseca and Fleming 1993) and the obtained Pareto fronts were examined to choose a final model among compromise solutions.

Bernadó i Mansilla and Garrell i Guiu (2001) introduced multiobjective optimization techniques to rule based classifier systems (CS) by separately formulating criteria for accuracy and generality of decision rules (and, for the whole rule set, a third criterion of covering the training set). Multiobjective optimization outperformed a single-objective variant, but it was found that special pressure towards the accuracy goal was better than using general Pareto ranking with no bias. For the final CS, a decision maker would have to select a set of rules from the whole Pareto front, and the unbiased Pareto ranking may stress the population-

based search with too many unacceptably overgeneral rules.

Radial basis function neural networks (RBFNN) have a feed-forward structure similar to the MLPs studied here, but the hidden neuron activation is based on locally centered proximity functions (the radial basis functions, RBFs) in the input space. Evolutionary multiobjective learning is naturally applicable also to RBFNNs. For example, Hatanaka et al. (2003) applied a Pareto-based approach to optimize both the accuracy (measured as MSE error) and the complexity (as the number of basis functions) of an RBFNN. Similar two-objective formulation for RBFNNs was used by González et al. (2003). In evolutionary operators, the locality of RBFs can be exploited in various ways as was broadly explored in the latter of the two works cited. Overall, the evolutionary operations are similar to those used in evolving MLPs: additions and deletions of neurons, and stochastically jogging parameters locally or globally. In the case of RBFNNs, crossover operation can be made useful due to the locality of the RBFs, and the weights connecting the hidden layer to the output can be optimized by solving a non-square linear system.

Specifically, González et al. (2003) used a crossover in which the RBFs are exchanged only with ones that are nearby in the input space. This can be done by measuring Euclidean distances between the centers of the RBFs. Locally adapted improvements can thus diffuse into the population via crossover while no unexplored gaps are left into the input space, which would likely happen if "blind" exchange of RBFs was used. The local improvement operators provided (i) random "blind" mutations of RBF centers and widths, (ii) mutations concentrating on currently less important RBFs, found by quantifying the relevance of each RBF using orthogonal least squares (OLS) and singular value decomposition (SVD), (iii) pruning mechanisms similarly concentrating on the less important RBFs, and (iv) addition of new RBFs to areas where the current number was seen insufficient based on a local error measure. Naturally, also the same multiobjective selection schemes based on Pareto optimality and non-dominance ranking are applicable in these algorithms.

Igel (2005) proposed a multiobjective learning approach to support vector machines (SVMs), which are yet another universal computational structure similar to MLPs and RBFNNs. In SVMs, a nonlinear kernel function (typically selected for each task manually) is used for transforming input data into a feature space, after which a linear transform is applied to yield the final result. In traditional approaches, the kernel functions and performance measures would have to be differentiable, and problems with getting stuck to local optima, similar to those with MLP learning, exist, as observed by Igel (2005). He proposed a Pareto-based evolutionary multiobjective optimization approach to overcome these issues. The complexity (as the number of support vectors) and the accuracy (as the number of misclassified training data instances) were taken as two objective functions. A self-adaptive evolutionary strategy scheme was used with the NSGA-II non-dominated ranking selection. A total archive of all non-dominated solutions found during the whole run was kept again.

The benefits highlighted by Igel were the possibility of using nondifferen-

tiable functions (due to the evolutionary approach) and the possibility of gaining insight from the Pareto fronts. Especially, a "knee point" in the two-objective Pareto front graphs was noticed where the decision maker would find "interesting" models for practical use. Let us recall here, also from the other studies of the period, that a single-shot optimization via weighted aggregation will not give such global information about the trade-offs possible. Real-time constraints were mentioned as a key motivator to find simple models, similarly to the work of Wiegand et al. (2004).

Jin et al. (2007) introduced the multiobjective evolutionary optimization approach to spiking neural networks (SNN), a biologically inspired augmentation of more simplistic models such as the feed-forward MLP with sigmoidal activation functions considered here. A major difference in SNNs is that the model involves simulating the neural response ("spike") as a function of time. The accuracy and complexity of SNN classifier models was optimized in a Pareto-based fashion using the NSGA-II selection scheme and task-specific mutation operators. Gaining insight into the learning properties of SNNs was highlighted as a merit of using a Pareto-based approach, which allows the examination of the obtained Pareto front.

As yet another example of a different kind of application within ANN learning, Jin and Sendhoff (2006) applied a multiobjective formulation to the catastrophic forgetting phenomenon, which means that earlier patterns may be forgotten while learning new patterns that are dissimilar to the earlier ones. This, of course, concerns "on-line type" applications where the ANNs are adapting to newly incoming data, without access to earlier training data. Pareto-based evolutionary multiobjective learning may be helpful via simultaneous optimization of both the error on new examples and on randomly generated "pseudo patterns" whose target values are generated by the current ANN. Again, direct encoding of connections and weights were used along with mutation operations (no crossover) and iRprop$^+$ (Igel and Hüsken 2000) for local improvement. The local learning step needs to be done for a single objective, so a random decision was made whether to improve on new or pseudo patterns. The NSGA-II scheme was used to pursue non-dominated solutions.

Let us finally observe that ANNs and multiobjective optimization have also been hybridized with a broader range of machine learning algorithms. For example, Markowska-Kaczmar and Wnuk-Lipinski (2004) used a feedforward ANN model as an embedded part of a rule extraction framework. Two conflicting objectives of *fidelity* (accuracy of classification) and *comprehesibility* (number of rules and premises used) were formulated and aggregated by multiplication to a single fitness function. A single-objective genetic algorithm with variable-length chromosomes and special operators for crossover and mutation were used without local improvement.

3.2.11 Summary of the First Decade

Let us summarize some of the specific discoveries that can be obtained from the works digested above. First of all, during the first ten years after machine learning techniques started to involve explicitly multiobjective optimization formulations (approximately mid-1990's), it was well proven through both theoretical considerations and practical experiments that the merger is useful. A multiobjective methodology not only opens totally new application possibilities but interestingly it can also improve the results and convergence characteristics of single-objective tasks when used as an intermediate step. A large number of different tools and purposes of machine learning have been investigated and found to benefit from MOO.

With regard to the main focus here, Pareto-based MLP network design for classification, we can observe the following:

- *Life-time learning*, i.e., the memetic *local improvement* is useful, and gradient-based methods can be used as the improvement even when the *local minimization objective(s) may be different* from the (possibly non-differentiable) overall objective functions.
- The local *improvement is usually single-objective*, requiring either scalarization or the selection of only one of the objective functions, typically the error criterion. Randomized choices of the direction of improvement have been successfully used.
- Local *improvement may result in overfitting*, which can be fought using variants of traditional methods like cross-validation, regularization via penalties, early stopping etc.
- Keeping a *full archive of non-dominated solutions*, possibly external from the main population used for evolution, is beneficial and commonly used in practice. For practical problem sizes and current computers, such an archive is computationally cheap enough.
- The usual way to evolve MLPs is based on small *structural mutations* such as jogging weights with a "creep" operation and adding or deleting one or few neurons or synaptic connections at a time. Decision between the operations is usually made by random choice, but the operation probabilities are likely better to be adaptive and evolving throughout the search.
- *Crossover is difficult* due to the fundamental structure of the MLP (involving the "competing conventions" problem), and thus in practice the methods have refrained from using crossover.
- It is generally acknowledged that *algorithms need to be tuned according to the particular task and implementation structures* being targeted.
- The early studies mostly concentrated on *only a few objectives*, and a usual selection was to optimize the main, or primary, objective of *accuracy*, and one additional secondary objective reflecting the *complexity* or the *generalization* capability of the model. Both objectives were selected from among many known alternatives.

3.3 Recent Developments

In the previous section, the initial years (ca. 1995–2005) of multiobjective machine learning were examined, based on well-founded resources (Jin 2006; Jin and Sendhoff 2008) published by the end of that "period of proliferation". Next, an overview should be made on what has happened since (i.e., during 2005–2015), up to the time of writing this dissertation. In Section 3.3.1, an outline of some of the few review articles available is presented. Sections 3.3.2 – 3.3.9 digest articles that were found to be (subjectively) most relevant to this dissertation among those identified in a literature search detailed later in Chapter 4. In Section 3.3.10, a summary of the most recent decade is laid out, and findings relevant especially to the current topic are highlighted.

3.3.1 Developments of The Bigger Picture

Let us start by looking at the trends of publication activity in the crossroads of multiobjective optimization and machine learning, especially artificial neural networks. Details of the search string used in the database query will be explained later in Section 4.2.1. The Scopus database and tools[1] were used to find the numbers presented here. Figure 14 shows a graph of yearly publication counts that indicates an increasing trend of research activity in the field. The numbers are absolute, though, and not normalized with the increase of overall scientific output. The query string matched 1333 publications, with 126 published last year (2015, as of writing). A peak of 169 publications occured in the year 2014. All document categories searchable with Scopus were included, and they were found with the following frequencies: journal articles (54.2%), conference papers (32.6%), conference reviews (9.0%), articles still in press as of January 2016 (1.9%), reviews (1.1%), book chapters, books, and errata, totaling $< 2.0\%$ in the last three categories. The results are likely to include methods and applications of hybrids in both directions: machine learning used in MOO (for example, surrogate models used to improve the computational performance of optimization methods), and MOO used in machine learning, which is the main focus here.

There is a lot of research going on, documenting various method improvements and applications, but it seems to be difficult to find surveys or reviews that would go into the trouble of summarizing and comparing the recent findings in a systematic way. If this is not just the failure of the author here to locate such studies, it would seem quite necessary to provide one. Performing a topically focused but otherwise comprehensive survey of MOO and MLPs is therefore a priority action for the near future. For this dissertation, time and space allows recent developments to be sampled less comprehensively. We shall start with the few related surveys that the author was able to find.

Mukhopadhyay et al. (2014a,b) provided a two-part survey on multiobjective evolutionary algorithms used for various tasks in data mining. Their cov-

[1] http://www.scopus.com/

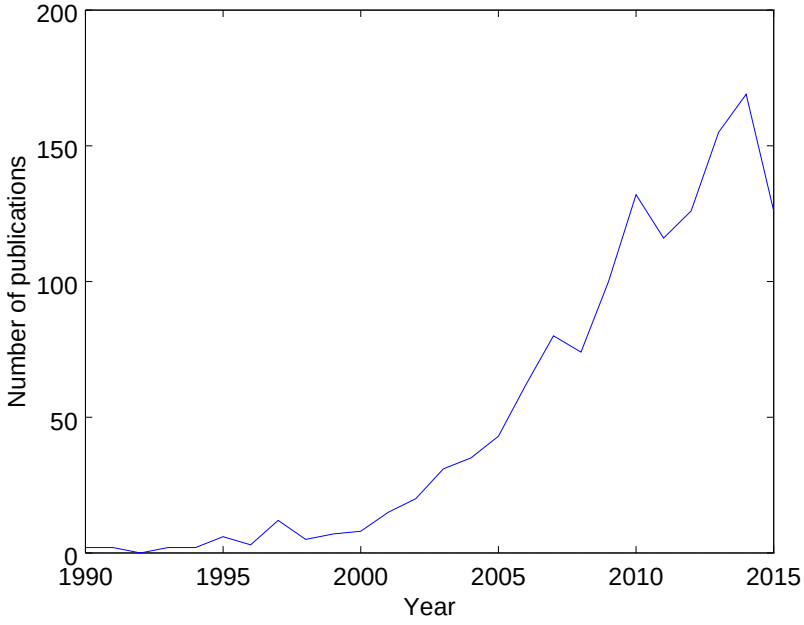


FIGURE 14 Publications at the crossroads of multiobjective optimization and machine learning per year (1990-2015). Data retrieved by searching Scopus with the query string explained in Section 4.2.1 and including all categories searchable via Scopus. Total number of publications is 1333. This is an underestimate, due to purposeful restrictions in the search scope, also to be explained in Section 4.2.1.

erage of data mining conveniently includes the machine learning tasks of supervised learning, ensemble generation, and feature selection. Due to the inclusion of other data mining tasks, and all families of methods available for solving the tasks, the scope of the survey is broad, and, resultingly, the coverage of sub-areas remains limited. Nevertheless, the relevant parts of their survey provide some useful pointers as to where multiobjective machine learning has been developing recently. Also, the categories of tasks overlap highly. For example, feature selection and ensembling can be applied in most of the other tasks, both unsupervised and unsupervised.

First of all, Mukhopadhyay et al. (2014a,b) recognized how the general applicability of the few fundamental methods of evolutionary multiobjective optimization currently pervades the whole field of data mining with differences only in details. They organized their survey of each data mining area according to these differences, which were observed to be (i) the *representation (encoding)* of the solution individual, (ii) the *objective functions* chosen to be optimized, (iii) the *evolutionary operators* applied, and (iv) the strategies used for *final solution selection* from the obtained Pareto front. In the light of this categorization, the fundamental structure of multiobjective learning problems seems to have remained stable since the earlier years.

The encodings and evolutionary operators still seem to be selected according to the tasks and implementations at hand, which is not surprising, of course. Expert knowledge and heuristics, such as special evolutionary operators and

memetic improvement steps, are applied in the algorithmic components to improve upon the basic forms of the underlying methods.

The number of objective functions simultaneously studied seems to have increased from the mostly bi-objective cases of the first decade into combinations of three or more objectives. The objective functions still seem to be selected from among the traditional ones. Specifically in MLP-based classification, it is usual to optimize various forms of accuracy, including ROC-type rates of various misclassifications, and model complexity. Feature selection methods naturally have the additional goal of minimizing the number of input features, and ensembling methods may have objectives measuring the ensemble quality. More detailed examples of ANN-based methods are found in the article digests presented in the following subsections.

Most of the recent studies have been using standard MOO algorithms such as NSGA-II, SPEA, SPEA2, PAES, PESA, and PESA-II, which are known to run into difficulties when the number of objectives is four or more. The situation is being mended by current research in multiobjective optimization, but Mukhopadhyay et al. (2014a,b) did not yet find them introduced to practical data mining tasks, including machine learning.

Mukhopadhyay et al. (2014a,b) also asserted that systematic comparisons of the different methods available are mostly lacking, possibly attributed to the unavailability of source codes or other details of implementation.

Some years earlier, a two-part review of neural networks in chemistry and chemical process modeling was published by Curteanu and Cartwright (2011) and Cartwright and Curteanu (2013). The first part (Curteanu and Cartwright 2011) concentrated on methods to select an optimal ANN architecture for a task at hand, and the second part (Cartwright and Curteanu 2013) was dedicated to evolutionary methods for ANN learning. The focus of this review series was in chemistry and process modeling applications, and its main goal was to educate the chemical research community about the proper use of ANNs as process models. Some of the definitions, claims, and references seem to be laid out a little hastily and, in part, somewhat vaguely, but the series does do a rather decent job in outlining the history and the overall state of the art in neural network learning, covering both single-objective and multiobjective training.

As to the current trends, differential evolution and swarm intelligence methods, such as particle swarm optimization (PSO), ant colony optimizer (ACO), group search optimizers (GSO) and bacterial swarming, were highlighted by Cartwright and Curteanu (2013). According to the review, at least in the chemistry domain, memetic combinations of gradient-based learning and evolutionary frameworks are common, and the NSGA-II is a widely used way to handle multiple objectives.

The reviews cited above, with all of their merits, do not give a very complete coverage of what has been done recently with multiobjective machine learning using neural networks. In the following subsections, some of the advances specific to ANNs within the last decade are covered, constituting a major part of the contribution of this dissertation and laying the foundations for a more compre-

hensive survey.

3.3.2 Structural Mutation and Optimal Hidden Layer Size

Goh et al. (2008) presented a Pareto-based memetic algorithm to jointly minimize the error and the structural complexity of single-layer feedforward networks (SLFN), i.e., MLPs with only one hidden layer. They used the typical choice of linear activation on the output layer and hyperbolic tangent as the hidden layer activation function. Restriction to SLFN was based on the universal approximation results by which a single hidden layer is sufficient for approximation (Hornik et al. 1989). Fundamental to their approach was the *geometric interpretation* that each hidden neuron creates a hidden feature by representing a *separating hyperplane* in the input space. They provided several SLFN-specific algorithmic elements based on the hyperplane interpretation and the singular value decomposition (SVD) (Klema and Laub 1980).

An idea fundamental to the proposals was presented earlier by Teoh et al. (2006). The SVD performed on the $N \times n_1$ matrix of hidden neuron outputs for all training samples was used to quantify the dataset-dependent number of relevant neurons. A gap, or a notable decay, in the magnitudes of the singular values given by SVD was observed to exist between the larger and the smaller ones. The number of the largest singular values would correspond to the number of actually useful hidden neurons. The rest would correspond to neurons that have adapted to noise, instead of actual information in the dataset. A user-selected threshold value would still be needed for exact quantification of what is large enough, but it was suggested that such a parameter is possible to select if some information about the source of the noise in the measured system would be known.

Goh et al. (2008) used this SVD-based quantification as part of a *crossover operation* suitable for the SLFN. This is noteworthy, because the known representation problems of MLPs are more often avoided (by not using crossover) than attacked. In particular, the SVD was used to estimate the number of redundant neurons in two parents. The angles between the neurons were then computed (without specifying the manner of computation, but the Euclidean inner product of the weight vectors, perhaps normalized, would seem like an obvious choice), to decide whether each neuron should be deleted ("pruned") or exchanged between parents. Neurons with the smallest intra-subspace angle (within a parent) were deemed redundant based on the geometric interpretation. They would be contributing via hyperplanes very close to each other, and only one of such redundant hyperplane copies should be retained in the child network. Similarly, neurons with the largest inter-subspace angle (between the two parents) were regarded as hyperplanes that have been learned by one parent but not the other, and such novel information should be mixed to produce children with the different learnings of both parents. A random choice with equal probability of $1/3$ was made for each parent to either (i) exchange the maximally different neurons with the other parent, (ii) only remove redundant neurons, or (iii) only add neurons from the other parent.

To facilitate the crossover, a *variable length chromosome* was used to represent the individual SLFNs in the population. The "smallest unit" of operation, i.e., the indivisible building block, was decided to be the neuron. The choice was based on the geometric interpretation and earlier results cited (Stanley and Miikkulainen 2002). Each neuron was represented as a fixed length vector containing the incoming synaptic weights, the bias applied in the neuron before the hyperbolic activation, and the output weights used in the output layer. The first two correspond to a row in $\mathbf{W}^1$ and the last correspond to a column in $\mathbf{W}^2$ in the matrix iteration formula of Equation (5) on page 25. Any number of such fixed-length neural representations was allowed, thus facilitating any size of the hidden layer. The actual number was to be determined through evolution using the SVD-based genetic operator.

Also the commonly used mutation operator based on Gaussian weight perturbation was used with an adaptive strength based on measuring the percentage of newly found non-dominated solutions among all non-dominated ones. The mutation strength would decrease when the global exploration phase has converged near the non-dominated front.

Two objectives were used to find a Pareto front: (i) the standard squared approximation error for one-of-K binary vectors representing classes, and (ii) the sum of squared synaptic weights. For the latter, also the discrete number of hidden neurons was considered as an alternative, but abandoned based on preliminary experiments.

The authors noted that the architectural operations would render the children in a noisy state, requiring a local improvement step before fitness evaluation. A real-valued micro genetic algorithm (μGA), i.e., a genetic algorithm with a very small population, facilitating fast local search, was used as the local improvement to learn locally better values for the weights using the structure that might have been changed by the architectural operations. The μGA was also hybridized with SVD so that only the input weights of the hidden layer were evolved, and optimal values in the least squares sense were determined for the output weights using the pseudoinverse matrix computed with SVD. Simulated binary crossover (SBX) and Gaussian mutation were used for input weight evolution. To balance the computational cost between global and local search, improvement using this μGA was used only for a subset of individuals. The size of this subset was adaptively increased as the convergence was found to move from global exploration to local exploitation.

The selected objective functions turned out to be differentiable, but evolutionary μGA was used instead of the traditional choice of backprop. The authors justified this decision based on the possibility of backprop getting stuck in a local minimum. A question could be raised, though, whether it should be in any way problematic to find a local minimum when the whole purpose of the step is to improve the solution locally.

Otherwise, the algorithm was based on standard components. Parents were selected using a standard binary tournament using Pareto ranking and niche sharing. A fixed-size external archive was used in which dominated or most

crowded non-dominated solutions were removed when new non-dominated solutions had been found. The archive was updated at the start of the evolutionary step, and parents were selected from a combination of the main evolving population and the archive. The algorithm returned the archive, whose size could be different from the one used in main evolution.

The presented geometrical interpretation (illustrated in Figures 5–9 of Chapter 2) provides an interesting way to handle the competing conventions problem, allowing a meaningful cross-over operator to be defined. Pruning operations can similarly be focused on building blocks that are likely to be redundant. However, the authors apparently measured only the angle between the hyperplanes and not their distance from the origin, which depends on the value of the bias and of the magnitudes of the other weights. Apparently, the authors also did not consider the remaining mirror symmetry, i.e., the fact that two completely oppositely directed hyperplanes can have an equal effect when their output weights have opposite signs. Extensions to more than a single hidden layer were not considered, which is valid from the universal approximation point of view, but is otherwise a restriction. It is known that deeper structures may cope with a smaller total number of neurons, which may be desired, for example, from the computational efficiency point of view in a final application. Some open questions therefore remain, but, since the recombination operator is known to be a crucial element in EAs, it is a remarkable asset to have one available for SLFNs.

3.3.3 Memetic MLPs for Classifier Sensitivity Improvement

Fernández-Caballero et al. (2011) applied MLP classifiers to predict the possibility of growth of micro-organisms under environmental conditions. They observed a "renewed interest" in ANNs among researchers in medicine and predictive microbiology, and contributed with a Pareto-based algorithm to improve the sensitivity of ANN classifiers for the domain. Datasets of the growth of different food-borne pathogens were examined, and comparisons were made between single-objective classifiers and the proposed memetic bi-objective MLP training method.

Two objective functions were maximised: (i) a continuous and differentiable cross-entropy measure, and (ii) the minimum classwise accuracy (called the minimum sensitivity by the authors) as a discrete ratio of correct predictions among all class members. The algorithm was an adaptation of NSGA-II (Deb et al. 2002), with local improvement of only the continuous entropy measure via iRprop$^+$ (Igel and Hüsken 2000). The authors called their method fittingly the Memetic Pareto Evolutionary NSGA2 (MPENSGA2). Specifically, the usual non-dominated sorting and binary tournament selection of NSGA-II were applied in population updates. To reduce computational cost, local improvement was performed only at selected generations (after $2/7$, $4/7$, and $6/7$ of the total number of generations computed).

Crossover operations were omitted on the usual grounds of being possibly detrimental in evolving MLPs. Structural mutations were the usual additions

and deletions of connections and neurons. Different ratio of connections were altered on the single hidden layer (30% of total) and on the output layer (5% of total). The parametric weight mutation operator was a Gaussian perturbation used in a fashion similar to simulated annealing, i.e., with a distribution width that is gradually decreased from early strong values (exploration phase) towards later softer values (exploitation phase) based on a user-specified geometric cooling strategy in which the user would select the initial temperature, a multiplier (less than 1), and the number of generations after which the current temperature would be multiplied.

A noteworthy technical feature of the algorithm of Fernández-Caballero et al. (2011) is that it uses an *object oriented representation*, in which the population individuals are fully specified ANN object instances, and all the evolutionary and local improvement operations are performed directly in the phenotype space instead of encoding and decoding a genotypic chromosome representation dictated by an underlying standard algorithm. Of course, the usual variable types still internally exist: the connections are represented as pairs of a binary variable representing the existence and a real value representing the weight of each connection.

Even with a complete Pareto front generated, only two final models were selected: the one with maximum entropy, and the one with best minimal sensitivity, corresponding to the two extremes of the Pareto front. The natural fact was observed that the Pareto fronts generated using the training data are not Pareto optimal with respect to the generalization test or validation datasets that were unseen during training. Also, the metrics used for method comparisons were different from the objectives used in the optimization process, and fronts plotted against these metrics are yet more different in shape from the ones plotted against the original objectives. Nevertheless, the MLP models obtained using the method compared favorably to 11 state-of-the-art classification methods.

As another related development, Cruz-Ramírez et al. (2012) applied an MLP model to a highly complex and unbalanced classification problem arising from matching patients waiting for a liver transplant with potential donors. The aim was to use supervised learning to predict the bi-class result of death or survival of the organ recipient after three months of the operation. As inputs, the medical records of the potential donor and the recipient at the time of transplant were used. Such medical applications are prime examples of situations where accuracy and sensitivity are conflicting, and the costs are incommensurable, involving the patient's death as a possible outcome. Data from 1001 actualized transplant operations was used, with more than 40 input factors. The actual time of survival after operation was recorded (followed up to a year, and thus including the 3 month threshold period selected for the case study). Approximately 11.3% of the cases resulted in non-survival, which is here the minority case, and of the greatest interest to the patient. As observed in the research, naïve optimization of accuracy using state-of-the-art single-objective classifiers yielded trivial classifiers with total accuracy of 88.7% with 0% accuracy (sensitivity) for the mortal cases.

To create non-trivial classification models that could actually be used in a medical application, Cruz-Ramírez et al. (2012) used the memetic Pareto differ-

ential evolution (PDE) neural network framework of Abbass (2001), explained earlier in Section 3.2.3. The two objectives of entropy and minimum sensitivity were considered similarly to the earlier study by Fernández-Caballero et al. (2011), while the algorithm was now based on PDE. This case study was also for two classes only, but all the formulations were written for handling any number of classes.

Before the couple of papers cited here, the use of the continuous entropy objective as part of the memetic PDE construct had been explored earlier by Fernández et al. (2009), and the memetic algorithm adaptation, called the Memetic Pareto Differential Evolutionary Neural Network (MPDENN) had been detailed by Cruz-Ramírez et al. (2010).

Again, local improvement using iRprop$^+$ was performed on selected generations (once in the beginning, once in the middle, and once near the end of the planned total number of generations). In this algorithmic variant, improvement was done only on a maximum number of selected individuals (only those within the first-ranked Pareto front, or, if the size of the front is greater than the specified maximum number of improvements, on a "representative" subselection based on k-means clustering). Also, again, the gradient-based local improvement step with iRprop$^+$ used only the differentiable entropy measure and neglected the discrete second objective which was used only in the Pareto dominance evaluations and final model evaluations. The structural and parametric crossover and mutation operators were adapted from the earlier works in multiobjective differential evolution for neural networks (Abbass 2003b; Abbass et al. 2001).

For model selection, Cruz-Ramírez et al. (2012) tried various methods based on the Pareto set of MLPs found with the memetic search: selection of objective-wise extreme points, representative points emphasising each half of the Pareto front (defined as solutions closest to k-means clustering centroids), ensembling with majority voting, simple averaging, and the winner-take-all method (the most certain value is taken). The different selection methods naturally yielded different trade-offs with generalization tests using data that was unseen in training (a four-fold crossvalidation scheme was used). Using the area under curve (AUC) metric for generalization, the best strategy was found to be the selection of the solution with the highest entropy measure, which was one of the two objective functions maximized. With different metrics, such as the classwise mean generalization sensitivities, there are expectedly different trade-offs between the different selection methods. Outlier removal from the majority class and neighborhood oversampling of the minority class were required for the best results, improving results for both the single-objective benchmarks and the proposed multiobjective method.

Sánchez-Monedero et al. (2011) proposed a scalarized single-objective MLP training method to strike one balanced result within the Pareto front of accuracy versus minimum class sensitivity. Such an approach serves its intended purpose of cutting down most of the computational effort required in memetic Pareto-based algorithms, but this happens at the cost of losing all information possibly conveyed by exploring the Pareto front.

Fernandez-Caballero et al. (2010) tested the MPENSGA2 algorithm and the two-objective proxy formulation of multiclassification sensitivity with 18 datasets, 10 of which involved at least 3 classes, again with favorable results. They conclude that the approach "reveals a new point of view for dealing with multiclass classification problems, and provides the opportunity to improve the sensitivity and accuracy of a multiclassifier for a wide range of data sets". As future research topics, they outlined the use of other memetic algorithms, other types of base classifiers, ensembling tools that make further use of the Pareto front that is obtained, and pre-processing tools to help with very imbalanced datasets.

Gutiérrez et al. (2012) elaborated further on the above methodologies. With the decision to use the minimum classwise accuracy (sensitivity) as a proxy for the sensitivity over all classes, any number of classes can be handled by the two objectives, but the trade-off information from all but the worst class cannot not be used. The authors made a note of this deficiency, but reminded that the choice "avoids computational and visualising disadvantages" of approaches (such as that of Everson and Fieldsend 2006) based on considering the $K \times (K-1)$ misclassification costs separately. A two-stage evolutionary algorithm for training MLPs to achieve a Pareto front in the accuracy vs. minimun sensitivity space was proposed. Again, the MLPs were represented as phenotypic objects only. Structural and parametric mutations were used, and crossover was omitted on the usual grounds.

This time, a total of five fitness functions were considered for the evolutionary algorithm: (i) total accuracy in the whole dataset, (ii) the minimum sensitivity, i.e., classwise accuracy, (iii) a differentiable cross-entropy function for MLP predictions, (iv) Cohen's Kappa statistic measuring the agreement between expected and obtained classifications, and (v) a measure of the two-dimensional area within the space of the first and second objectives residing between the evaluated classifier and the "ideal" (or utopian), generally unobtainable, classifier with full accuracy for each class.

In the first stage, exploration with strong structural mutations was to be used to push the population towards the ideal side of the accuracy-sensitivity space by using only the entropy function as fitness. Then, a small elite of the result was chosen to perform an exploitation phase with softer mutations, considering one of the other objective functions as the fitness. Experiments showed that the area function was the best choice. It is to be noted that the algorithm, while examining Pareto fronts as a theoretical foundation and for result exploration, actually uses two passes of single-objective optimization to arrive at a final result population. The paper is somewhat unclear about the final selection of an individual. It has to be assumed that it is the best individual with respect to the second stage, i.e., a solution closest to the ideal one, after starting from an initial population pre-conditioned by the first stage with a different objective.

Again, 18 multiclass datasets were used in the experiments. Combinations of the five objective functions were compared, and comparisons with other state-of-the-art classifiers and the fully multiobjective version (Fernandez-Caballero et al. 2010) were made. In comparisons, only the accuracy and the minimum sensi-

tivity on a validation dataset were used, while the stages of the algorithm were run using different combinations. It was found via the comparisons that the entropy criterion was good for the first stage, initially driving solutions towards the interesting end of the Pareto front, and the newly introduced Pareto-area criterion was best for the second stage to drive the elite of the first stage further towards the ideal edge of the feasible part of the Pareto solution space (which in this case is a linear graph, as formally proved in the paper). The earlier multiobjective algorithm outperformed the two-stage approach for some datasets, but for some datasets the final model decisions (from Pareto front extreme ends; see the digests above) involved an extremal trade-off between the two objectives. The authors concluded that the two-stage approach "provided an effective intermediate point" between either one selected by the multiobjective approach. This would naturally happen, since the first stage optimizes one of the final objectives quite directly, and the second stage further improves the resulting population towards the Pareto front, with an objective function that has its minimum approximately at the center of the sharp "tip", or knee point (Thorndike 1953), appearing in the Pareto front.

All in all, the two-objective proxy for multi-class sensitivity improvement suggested in the above papers is definitely an interesting approach to unbalanced multiclass classification, and it is appealing in its simplicity. While comparisons to other methods turned out to be favorable for all variants, the author of this dissertation must still find it appropriate to ask whether the Pareto front concept has yet been fully used in the approaches, observing that in each variant, a single solution is selected. It is fully explored in the papers that for nontrivial classification tasks the objectives are conflicting, and that the Pareto front is an actual curve, even if a sharp knee point appears near the ideal corner of the space. Still, the proposed methods will finally sample either one extreme or a single point somewhere in the middle. Nevertheless, the idea of a two-objective proxy for trading off global accuracy and sensitivity in multiclass classification tasks has been well examined with various approaches by the papers cited above.

In an even more recent work, Cruz-Ramírez et al. (2014) used the MPDENN of Cruz-Ramírez et al. (2012) to assess the problem of ordinal classification, or ordinal regression, in which the classes of a multiclass classification task have an internal ordering. An illustrative example given by the authors is the classification of people as belonging to the lower, working, middle, upper middle, or upper class. For example, it is a bigger error to predict "upper" when the true result is "lower" than it is when the true result is the closer alternative "upper middle". As a fitting continuation to their earlier case studies making use of a variety of classifier comparison metrics, the authors compare several metrics suggested for ordinal classification and also propose one of their own, which, somewhat analoguously to their proposed minimum sensitivity, considers the maximum per-class mean absolute deviation between the predicted and the observed ordinal classes. The purpose was to compare metrics that could be used as objective functions in training ordinal classifiers. It was also shown that the metrics can be mutually conflicting, especially in the later phases of evolution.

MPDENN was used for testing the suitability of metrics that were first found to be mutually conflicting in a correlation study using other classification methods. Also in this study, final models were selected at the extremes of the Pareto front obtained by the MPDENN, and compared using also other metrics that had not played a role during training. The pairing of the global mean absolute deviation and the maximum class-wise mean absolute deviation was found to be a competitive choice.

The research track digested above in this subsection serves as one living proof that MLP training using memetic multiobjective optimization is very much alive and actively studied in the machine learning community well into the 21st century.

3.3.4 Pareto Front Sampling with a Controller Approach

Costa et al. (2012) provided theoretical convergence analysis for a method proposed by Costa et al. (2003) as a continuation of the work initiated by Teixeira et al. (2000). As recognized also by the authors, their approach interestingly turns completely around the usual way of applying neural networks as part of an industrial control system. Instead, they use control theory to train MLPs. Specifically, the objective space of multiobjective learning is modeled as the state space, through which controlled trajectories can be driven by adjusting the gain parameters in a multiobjective modification of the sliding mode control (SMC) method (Utkin 1977).

Costa et al. (2003) presented gradient-based learning rules for the usual case of minimizing MSE error (accuracy) and the sum of squared synaptic weights (model complexity). Arbitrary trajectories through the objective space, ultimately leading to the Pareto front when the control target was set beyond the reachable area, were demonstrated. In the early work, the gains had to be manually adjusted experimentally. With the convergence analysis of Costa et al. (2012), the need of experiments for gain setting has been lifted. Values leading to theoretically guaranteed SMC convergence can be automatically selected, so it is sufficient to select a target objective point or multiple intermediate points. Once a point on the Pareto front is reached, the whole front curve can be traced by nudging the target point further in the unreachable part. The authors also mention the possibility of examining a third objective function along the way, but the control method itself is currently operating in two dimensions. After the tracing, the sampled Pareto front can be used as in any Pareto-based method.

The concept of *learning trajectories* is fundamental to the SMC learning formulation. For two-objective tracing of the full Pareto front, a trajectory method, especially with arbitrary trajectories, looks like a complete solution. After all, the front in two dimensions is a curve. In three or more dimensions, trajectories cannot sample a full Pareto front, but of course two-dimensional level sets could be traced similarly, regardless of the dimension of the full problem. Also, a noteworthy observation of Costa et al. (2012) is that *all* learning methods based on incremental weight update rules, such as backprop, end up with a learning

trajectory, be it controlled or not. The stability of the SMC-controlled trajectories plotted next to uncontrolled ones in the article is very impressive indeed.

As of yet, the SMC learning method for MLPs has been formulated only for MSE error and the squared norm of weights. The assumption of quadratic error is fundamental to the theory presented. It would be interesting to see more extensions of this approach to other kinds of objective functions. Also, it would be interesting to consider extensions of trajectory-based learning with more than two objective functions. Controlled or not, the trajectories traced through an objective space (of any dimension) seem like a good way to visualize the workings of an iterative multiobjective learning algorithm. Of course, such graphs are just many-dimensional variants of the usual convergence graphs of single-objective error vs. the number of training epochs. But still, not many of them have been seen yet, at least in the multiobjective learning literature that came up for this study.

The cited works on controlled learning trajectories do not consider the question of local minima. While the convergence to a locally Pareto optimal solution is guaranteed, i.e., the end point of the trajectory will be such that incremental gradient-based steps cannot take it further towards a point beyond the located Pareto front, it still seems possible that globally better solutions (dominating those on the front that has been found) might exist that have not been reached by traversing the current trajectory. As with any gradient-based methods, evolutionary operations should be able to help with such issues, but, of course, such perturbations would introduce discontinuities in the otherwise smooth trajectory.

3.3.5 Accuracy and Model Complexity for Limited Hardware

Capel-Cuevas et al. (2012) described an industrial application in which, similarly to the one of Wiegand et al. (2004), technical requirements necessitate simple models with few computations when operating in the field. In this case, a prototype of a portable (hand-held and battery operated) pH determination sensor unit was produced. The detection of the pH level, i.e., of the acidity or basicity of a liquid solution, was to be made with a photometric sensor array that registers the color change of chemicals attached to a supporting thin film. The array consisted of 11 sensor elements with different chemical compounds, called "indicators", that would change their color when dipped to a solution of a certain pH level. To reduce manufacturing costs of the arrays, minimization of the number of these sensor elements would be beneficial. Also, as little computation as possible was to be required in the final system, because it was to operate on battery power and with a cheap microchip with very limited computing and memory capacity. The limited memory capacity sets hard limits to the range of algorithms that can be implemented, and, with highly limited computational power, the response time from sensor array insertion to being able to read the result from the LCD display could also be a factor that affects user experience.

MLP was chosen as a nonlinear function approximation model to translate the color changes to a pH value. A training set was created by measuring the

ground truth value of the pH of 121 different solutions using a laboratory potentiometer. Then, the sensor element color changes were measured as they would be by the integrated camera of the handheld device. Only the H (hue) coordinate of the HSV color space was used, providing one real value of input for each of the 11 sensing elements. Multiple replicated measurements were made to avoid outliers.

Even with the manufacturing and computational requirements of model simplicity, the full range of pH levels was to be covered with good accuracy, so multiobjective machine learning was employed. The authors used NSGA-II hybridized with a local gradient-based search to minimize three objective functions: (i) the number of inputs (sensor elements), (ii) the number of neurons on the single hidden layer of the model, and (iii) the maximum error among the training data patterns.

The final selection was made from the extremal part of the Pareto front that emphasized prediction quality. All the sensing elements were used, after all, to meet the best prediction error possible. Also, the final model had the largest number of hidden neurons of all the Pareto optimal solutions. Still, the multiobjective approach was useful in that the number of hidden neurons did not have to be selected *a priori*. It turned out that less than the maximum number of neurons allowed in the representation (9 vs. 11) was sufficient to reach the best accuracy. The authors noted that by using the Pareto-based multiobjective scheme, the decision maker can make an assessment of developmental needs "such as energy saving, computing time response, array development costs and instrument minimisation, among others". An idea not mentioned by the authors is that, with some imagination, it could be possible to think of different implementations brought to the market, each corresponding to a different Pareto-optimal solution, so that the end user could decide between a cheaper and less accurate device and a more costly but more accurate one. In fact, since even the most accurate model fits inside the memory of the device, a selling-point could potentially be made of the possibility to choose between cheaper arrays (with less elements) and more accurate ones (with more elements). The same device could be used with either of these, and the Pareto front already contains MLP models that can trade-off between the number of inputs available and the measurement accuracy.

3.3.6 Multiobjective Swarm Intelligence for Prediction Intervals

We now turn to another application of multiobjective ANN learning presented by Taormina and Chau (2015a,b). Even though the application considers time series forecasting instead of classification, it is of interest here for a couple of reasons. Firstly, it shows some of the most recent developments in population-based algorithms that deal with ANNs. Even though MLP is not mentioned by the authors, both the figures in their work and the explanations given in the cited ones give reason to believe that feed-forward multilayer ANN constructs have been used even if the algorithms are usable for any structures. In fact, the algorithms could be used for any predictive base learning model. Particle swarm optimization has

been used, which seems to be an emerging overall trend among current method selections. Secondly, the application domain is related to hydrology and water resources management, which appears to have been one of the frontline applications of ANNs recently. Thirdly, this is an example that has very recently continued a traditional approach of multiobjective learning, with some new twists that may stir one's imagination about future algorithmic prospects.

That said, Taormina and Chau (2015a,b) used ANNs to forecast river flows. To predict the future streamflow measured in cubic meters per second in flood-prone rivers, they used the time series prediction methodology of Khosravi et al. (2011) and Quan et al. (2014) to create ANNs (likely at least very similar to MLPs) with two outputs that jointly give a prediction interval (PI) instead of a single prediction point. The outputs are defined as the lower and upper bounds between which the future point falls with a given probability. Such intervals are much more useful in actual applications than single point measures, because (un)certainty of the prediction is explicitly covered by the PI. The model itself is presented as a single-objective function that is a nonlinear aggregation of two quality indices: the percentage of target instances having been outside the predicted interval, and the overall width of the predicted interval (as RMSE of the difference between predicted lower and upper bounds).

Multiobjective optimization was directly applied in the particle swarm optimizer to handle generalization. Similarly to one of the approaches of Abbass (2003a), Taormina and Chau (2015a) divided the training dataset into two equally sized sets, and the two objective functions were taken as the main objective function value computed separately for both splits. An archive of Pareto-optimal solutions was used to inform the swarm of the best flocking directions found so far. In this way, also in this study, multiobjective optimization was embedded within the algorithm to improve generalization while still learning by optimizing a single main objective.

In addition to the multiobjective swarm algorithm, multiple criteria were used to select models from the Pareto front obtained. We shall say "criteria" instead of "objective" here, since these were measures that had not been used directly in the optimization phase, but were used to select a best individual from the resulting population after optimization. It turns out that the two criteria were based on the two aggregated terms of the single objective function. The Pareto set was searched for the "most precautionary" individual that was best in the percentage of mispredictions and the individual with "narrowest interval". It turned out that the final emphasis between these, obviously conflicting, criteria, depended on the dataset. For one of the two rivers investigated, it was better to select the most precautionary model. For the other one, the precautionary model had too broad intervals to be practically useful, while the narrowest interval model was deemed to be close enough to acceptable misprediction percentages. Here, we could imagine how such an optimization method would turn out if the two aggregated terms would be separated in a multiobjective fashion in the first place. In the current proposal, the outer tier of the algorithm is already multiobjective even though the inner tier works with a single aggregated objective.

For the purposes here, we skip some details given by Taormina and Chau (2015a) about feature selection, initial search for an optimal ANN architecture, and comparison with other methods. What matters here is that the authors were able to conclude that their multiobjective fully informed particle swarm optimizer found "substantially narrower" prediction intervals than a previous single-objective approach. Prediction quality improvements were shown especially in a time frame of special interest, when a heavy storm had resulted in peak values, which, of course, is a critical time to reliably predict the possibility of river flooding. A further observation was made that the multiobjective version presented a 40% to 50% improvement in computational time, compared to the single-objective swarm optimizer.

As seen from the digest here, and with more evidence in the full paper, several objectives (and "criteria") need to be assessed in the various stages of creating, selecting, and validating an actual predictor model. This feeds the imagination with regard to further possibilities of multiobjective ANN learning: Could more of the various criteria be introduced as explicit optimization objectives in a holistic algorithm that performs more of the necessary stages in a single run? For example, at least superficially, it would seem plausible to include feature selection and ANN architecture optimization, as was done in the previous application example of Capel-Cuevas et al. (2012), into the kind of algorithm used by Taormina and Chau (2015a) that evolves a population of ANNs also with respect to a main performance objective. The main objective itself could, in fact, possibly be split into the two separate objectives that are currently aggregated into one.

3.3.7 About Multiobjective SVM Model Selection

Rosales-Pérez et al. (2015) used a surrogate-assisted evolutionary algorithm to address the problem of classifier model selection. There are two major reasons to highlight this recent research here. Firstly, the proposed method exemplifies the range of possibilities in creating hybrids between already existing approaches. It is customary to use machine learning models as computationally cheaper surrogates that approximate objective functions that are very expensive to compute, for example in industrial optimization tasks based on physical simulations. In this case, though, it was the SVM classifier model selection that was identified as a computationally expensive optimization problem, and a multiobjective evolutionary algorithm was used with a surrogate that approximates the effect that various design choices (pre-processing method, feature selection, model structure, model hyperparameters) have on the two objectives of bias and variance. The expensiveness arises from the use of $n \times k$-fold crossvalidation used to evaluate the bias and variance for every choice of model design parameters.

A second reason to mention the research of Rosales-Pérez et al. (2015) here is that its goals are representative of problems that need to be tackled when using the SVM method even with its other theoretical and computational properties that have made it attractive to the community. A shift of interest towards methods with convex optimization formulations, starting with the appearance of

SVMs, was observed, for example, by Costa et al. (2012, p.21–22), who also noted that while "non-convexity is considered as an unavoidable part of the formulation" in their work with MLPs, the convexity of SVMs is not able to change the fundamental complexities arising from the data under consideration nor the need to build systems to properly handle those.

As for the method, Rosales-Pérez et al. (2015) represented the choices between alternative pre-processing methods and algorithm hyperparameters as binary and real variable vectors, respectively. Rather standard binary and real-valued evolutionary operators and binary tournament selection, alike that used in NSGA-II, was used in a surrogate-assisted multiobjective evolutionary algorithm described in an earlier work (Rosales-Perez et al. 2013). After evaluation of the initial population, the costly objective functions were left up to the surrogate model, which itself is actually an ensemble of SVMs for regression. During optimization, the surrogates were updated on each generation by evaluating the real objective functions only for those new solution candidates that appeared to be Pareto-optimal with regard to the objectives computed using the current surrogates. In this way, the surrogate objectives would become increasingly more accurate near the Pareto set, and all the costly crossvalidation computations would be made only within this area of highest importance.

The final model selection was made also in this research by evaluating generalization performance among alternatives that had been found Pareto-optimal (with respect to the bias and variance) with regard to the training set. Before evaluation against the validation set, the final models were re-trained once more with the complete training data (without the k-fold split used while optimizing). As future research steps, the authors anticipated generalization to machine learning methods other than SVMs, thus encompassing the complete model selection problem via selecting the base learner type in addition to the methods of preprocessing, feature selection, algorithmic parameters, and naturally the model parameters themselves. Other possibilities mentioned also by these authors are ensemble construction and examination of the Pareto front to analyse the behavior of models, given a certain dataset.

3.3.8 ROC Fronts

Another example of SVM model selection is that of Chatelain et al. (2010), who applied the standard (real-valued) NSGA-II algorithm to optimize the hyperparameters of an SVM model for binary classification. Three variables were selected: (i) the width of the RBF kernel, (ii) a constraint penalty value for the positive class error, and (iii) a similar parameter for the negative class. The two objective functions used were the true positive rate and the false positive rate.

The approach is in line with the earlier research on multiobjective interpretation of the ROC curve (for example, Everson and Fieldsend 2006; Fieldsend and Singh 2005; Kupinski and Anastasio 1999, covered earlier in this dissertation). The findings support the idea that a Pareto front optimizing the true and false positive error rates can be interpreted in the sense of ROC curves, but due to

the difference from the traditional approach of forming the ROC curve by varying a threshold parameter in an already trained classifier model, Chatelain et al. (2010) introduce the rather fitting name of *ROC front* for the MOO optimization result in which the individual classifiers are different from each other. They show by examples that each model on such an ROC front is locally better with respect to the ROC space than any classifier obtainable by modifying the threshold of some other model. Therefore, the fittingly named metric of *area under the front (AUF)* was proposed for measuring and comparing the ROC fronts obtained by different Pareto-based multiobjective learning algorithms.

The measured AUF is generally better than the AUC of any single model, but the authors hastened to clarify that the values of the two metrics should not be compared without hesitation because of the difference in what they measure (Pareto front of different models vs. a single individual model with a varying threshold). They also showed their algorithm in action in a real-world application that benefits from obtaining a Pareto front. In a system for hand-written number extraction, the SVM model was used as a fast preliminary check that should predict whether a separated image possibly contains a digit or something else (a letter, piece of text, other non-digit glyph). A more complex, slower, system was used for the actual digit determination, and the investigated component performed a filtering step in order to reduce unnecessary work later on. The function of the component was thus similar to the MLP for face recognition considered by Wiegand et al. (2004), covered earlier in Section 3.2.9. Real-time performance was not similarly critical in this case, but the total throughput would be degraded by any false positive decisions made by the filter component.

False positives require futile computations down the line that should be avoided. True positives should naturally be maximised to get most of the actual digits to be processed, but there is obviously a conflict in the goals. What makes the ROC front especially appealing in this case is that the engineering choice of the best operating point cannot be done before integration to the complete system. The costs of different kinds of misclassification are not known while optimizing. With the ROC front, it is possible, in the authors' words, to "postpone the choice of the final classifier as late as possible" and to "change the classifier without a computationally expensive new learning stage when target conditions change". Both of these merits are handy in the real-world system design presented.

The authors emphasized that their approach for modeling the objectives is not restricted to SVM or NSGA-II, but that any classification model and Pareto-based optimization algorithm could be selected, and more than two classes could be considered. They also suggested the possibility of including other engineering objectives, such as computation time needed for the decision when the model is operating in the field.

Let us briefly mention one more recent research regarding multi-objective SVMs. Aşkan and Sayın (2013) used the inherent likeness of SVM learning and multiobjective optimization to incorporate the error on positive and negative patterns in binary classification tasks, much like the formulation of Chatelain et al.

(2010). They minimized the three objectives of (i) separation margin maximization, (ii) the error made on positive patterns, and (iii) the error on negative patterns. Their approach is an exhaustive search scheme embedded in SVM learning rather than wrapping the traditional learning within an auxiliary evolutionary algorithm that tweaks hyper-parameters. Two-objective level sets of the Pareto front with one objective fixed in an iterated "grid search" manner is gained, containing trade-offs between positive and negative class errors and the norm of the vector describing the separating hyperplane. Hyper-parameters such as kernel type were not included in the approach.

The couple of examples of SVMs cited above demonstrate how uses of MOO are constantly developed for various kinds of learning machines. SVMs may have been more popular than MLPs lately, but it can be seen from the struggles faced in SVM learning that the fundamental problems of model selection in less trivial situations do not necessarily go away with the SVM formulation. Model selection is still inherently a multiobjective task, and the inherent non-convexity and other difficulties present in actual datasets manifest themselves as questions of how to select the hyperparameters and how to modify the internal optimization problem formulation.

3.3.9 Notes on Other Alternative Structures and Methodologies

Almeida and Ludermir (2010) described a memetic hybrid method combining elements of GA, ES, PSO, and gradient-based local optimization to train MLPs for classification tasks. They considered no less than five objectives: (i) a training set error, (ii) a validation test error, (iii) the number of hidden layers, (iv) the number of hidden neurons, and (v) the "complexity" of activation functions selected in the MLP network layers. Alas, they used empirically selected weight coefficients to aggregate these five measures into a single objective that was used in the underlying memetic optimization method. Therefore, no real Pareto front could be produced, and none of the acknowledged benefits of multiobjective optimization over single-objective optimization could be used in their proposal. Other than that, some interesting ideas were laid out. They reviewed both the strengths and the weaknesses of each individual method family available for ANN learning, and they suggested that a hybrid that picks the best part of each constituent method should perform better than any one method on its own. They also picked up an "object oriented" approach to individual encoding (although they did not call it by that name), which included, for example, the local improvement method selection (among a variety gradient-based methods) and values for all the parameters that the individual should use in an improvement step. These *optimization parameters were meant to co-evolve with the individual*, constituting what is usually called a self-adaptive algorithm. But, at least in this early proposal, it appears as if they did not yet go far enough to really exploit their formulation to its full capacity. The study is cited here because it is an example of how the continuous developments in optimization methodology should be, also continuously, integrated with machine learning.

Pasti et al. (2010) trained ensembles of MLPs for multiclass classification using a multiobjective artificial immune system (AIS) optimization algorithm (Castro and Timmis 2002), which is yet another nature inspired paradigm of optimization. AIS is based on modeling the immune system of a living organism that contains a large number of cells that learn to identify antigens (i.e., "germs" that produce illnesses). Optimization using AIS is a population-based metaheuristic, just like evolutionary algorithms, only with different mechanisms of population updates. One reason to digest the study here in the following few paragraphs is that it presents one more example of the variety of optimization algorithms being developed and used for machine learning. Another reason is that its overall topic was ensemble creation, which has been seen as one of the potential uses of multiobjective learning methods.

Pasti et al. (2010) continued on their previous studies with MLP ensemble training using AIS, and evaluated the effect of optimizing the diversity of ensemble members explicitly as a second objective alongside accuracy. They experimented with optimizing the two objectives of squared error on a training set and the negative correlation of individuals with respect to the population. The second objective was evaluated as the correlation of the errors made by one individual with regard to the errors made by an ensemble of all individuals in the population, computed over all the training data patterns. Minimization of this correlation (yielding highly negative error correlations) was hypothesized to explicitly increase the diversity of the resulting population of MLP classifiers, because the individuals would make errors on widely different subsets of the training data. The Pareto set thus created would contain MLPs with high accuracy, but with heterogeneous error profiles.

The results of Pasti et al. (2010) were negative in the sense that, at least in their experimental setting, the minimization of negative correlation yielded ensembles than performed worse than those produced with a single-objective version of the same algorithm, minimizing only the error objective. A gradient-based algorithm was also found to be superior in one of the five benchmark datasets used. This negative result may be useful in considering how to build ensembles. High accuracy of members with some, but limited, diversity, may be the best option in some cases.

Tsakonas (2014) used genetic programming (GP) to build a hierarchical ensemble structure for regression and function approximation tasks. Both MLPs and SVMs were used in the pool of weak base learners. The weak learners of the pool were first initialized and trained using their respective, traditional, local learning strategies, after which GP was used to create a stronger model by combining the weak learners into an executable tree with various operators including means and medians of the predictions made by sub-trees. As of the reported version, multiple objective functions were used in a scalarized fashion to incorporate pressure towards both accuracy and diversity (modeled essentially via a correlation measure) in the evolutionary selection, mutation, and crossover operation probabilities for each base learner. While not yet implemented, many possible extensions were suggested: proper Pareto ranking could be used, the pressure

probabilities could be re-adjusted during the run, and the base learners could be continuously re-trained as well. The papers of Tsakonas (2014) and Pasti et al. (2010) provide examples of the variety of approaches currently being used and developed to deal with persisting machine learning problems, such as diversity management in ensembles, where multiobjective considerations naturally enter the picture.

In addition to the very recent work of Taormina and Chau (2015a) on particle swarm optimization for MLPs outlined above, let us look at one more example of the emerging trend of PSO for machine learning. Qasem et al. (2013) used a memetic multiobjective PSO method to simultaneously optimize the accuracy and model complexity of RBFNN classifiers. Three objective functions were considered: (i) the classification accuracy, (ii) the total number of RBFs, and (iii) a smoothness metric of the RBF functions. A rather standard representation was used with a fixed-size binary vector representing the existence of each input and hidden neuron, and another vector of real values representing the weights of the network. Standard PSO was altered to accommodate a local search step, global guide selection based on dominance ranking, and an individual replacement strategy based on crowding distance. Comparisons with other state-of-the-art methods were favorable to the method.

As a final example of recent combinations of different optimization methodologies, Du et al. (2014) combined the multiobjective selection mechanism of NSGA-II with a recent version of differential evolution (Zhang and Sanderson 2009) designed to improve the convergence preformance of DE for single-objective optimization. In addition to introducing the NSGA-II selection scheme to make the proposed DE variation multiobjective, Du et al. (2014) modified the fitness function of NSGA-II to emphasize solutions close to the "knee-point" or "knee-region" of a bi-objective problem (Deb and Gupta 2011). The two objectives considered were the MSE error and the standard deviation of predictions in time series forecasting done by single-layer feedforward networks. It was suggested that when a single model must be selected from the Pareto optimal ones, the best one would be found near the knee point, in which the PF presents a notable turn between prioritizing one of the two objectives. Therefore, after an initial scouting of the whole Pareto front, it is beneficial to focus the rest of the search near the knee-point.

3.3.10 Summary of the Last Ten Years

For the purposes of this dissertation, it would have been wonderful to cover the whole history of multiobjective machine learning up to this day via textbooks or secondary studies. As it turns out, such resources either do not exist or are very hard to find using search words that the author was able to come up with. Hence, recent advances were sampled by looking at some works that appeared to be of the greatest interest regarding the current topic. More details of the sampling process itself are given in Chapter 4. The current end result was the digests of research articles given up to this point. Similarly to the key points distilled in

the summary of the first ten years of multiobjective machine learning made in Section 3.2.11, key points arising from the second ten years are highlighted here:

- A meaningful *crossover operator* can be designed for MLP classifiers through the geometric interpretation of the hidden neurons forming separating hyperplanes in their input space.
- *A single neuron is a suitable building block*, i.e., an indivisible unit, in the representation of MLPs for EAs. *Variable length chromosomes* and *object oriented phenotypic representations* have been found useful.
- In some approaches, the multiobjective formulation still plays the part of an *implicit helper in optimizing a single objective*, and possibilities of dividing aggregated main objectives into their separate terms may not have been thoroughly investigated.
- Methods that are internally Pareto-based are often *finally reduced into single objective ones* by selecting one solution from among the Pareto-optimal ones, using a pre-determined strategy. While MOO is shown to help the learning process in various ways, a question remains if the Pareto front could be utilized more effectively. Truely enough, only a single model can be selected to operate in a final application, but perhaps the human decision making capabilities could be exploited more by engaging the decision maker more in the final model selection process.
- *Combinations and interactions of many objective functions* have not yet been considered much. It is more usual to pick two or at most three objective functions, usually based on preliminary testing, than to explore Pareto fronts with more objectives. For example, the multiobjective nature and interplay of the alternative formulas for accuracy or model complexity has not been explored prominently.

To conclude, the traditional MLPs appear to be actively used as learning machines in various fields of applications, with possibly even a recent "renewal of interest" resulting from the observation that inherent non-linearities and non-convexities in the actual datasets considered result in problems that cannot be magically solved by otherwise simpler models such as RBFNNs or SVMs.

4 MAPPING THE LITERATURE ON MULTIOBJECTIVE LEARNING

In this chapter, we briefly depart from the realm of supervised machine learning, even if the goal is to gain insight into research on the very topic. What follows is an application of a method and a software tool detailed in the paper by Nieminen et al. (2013), which applies unsupervised learning, specifically clustering, to automatically find some structure in a body of scientific literature. The primary goal here was to get an overview, or a lightweight mapping, of the current body of knowledge on the general topic of "combinations of multiobjective optimization and machine learning, preferrably with emphasis on MLPs". The prioritized list of specific goals was as follows:

1. *Verify the level of coverage of this dissertation and its position with regard to previous studies*. In particular, the purpose was to check that no critically important research has been accidentally omitted in the overview of Section 3.2 that was mostly based on a secondary study by others.
2. *Identify recent research that is most closely related to the topic of this dissertation*. The particular target was to identify and prioritize the articles eventually digested in Section 3.3.
3. *Assess the usefulness of the automatic literature clustering method, and outline future ideas for its development*.
4. *Perform the first steps necessary for a follow-up secondary study*.

The first two goals were pursued by manually walking through a systematically selected database dump of titles and abstracts of articles published in peer-reviewed journals. The third goal was a by-product, since the manual walk-through was partly assisted by the automatic clustering method, and a subjective assessment of its use emerged naturally along with some ideas for further development. The fourth goal appeared along the way, because it seems, after a considerable effort of searching, that systematic review articles on the specific topic of multiobjective MLPs have been sparse, hard to find, somehow off-topic, or nonexistent, as of the most recent decade. Performing such a review is beyond

the schedule of getting this dissertation evaluated, but the first steps have been made as a by-product of the first three goals.

The structure of this chapter is as follows. First, in Section 4.1, the literature clustering method of Nieminen et al. (2013) itself is briefly outlined, and its necessary adaptation to the specific task is described. Some new algorithmic developments after the original publication are described for the first time. Section 4.2 documents the case study of mapping literature on multiobjective machine learning both automatically and manually with assistance from the tool. In Section 4.3, the results of the automatic tool are compared to manual literature categorization. Section 4.4 is a summary, discussion and reflection upon the findings of this exercise in informetrics.

4.1 The Literature Mapping Method and its Adaptation

The background story of the method in a nutshell is as follows. As doctoral students, we were tasked to do a literature review on data mining research. Knowing that the proper way to do such a feat is to follow some guideline of conducting a systematic literature review (SLR) (Jesson et al. 2011; Kitchenham 2007) or, as a first step, a mapping study (Budgen et al. 2008), we started with research questions, a review protocol, and the identification of the relevant subset of primary studies. Not so surprisingly, data mining turned out to be quite a broad matter to handle. Having already worked with methods suitable for handling large datasets, we instinctively started to "apply data mining to mine the data mining literature". The rabbit hole got deeper, and we ended up contributing a method that could be positioned in the field of scientometrics, "the quantitative study of science" or, on a more general level, informetrics. Meanwhile, of course, key questions regarding hot topics in data mining got answered, using the tool.

4.1.1 Original Version

Figure 15 recalls the outline of the Knowledge Discovery in Databases (KDD) process as presented by Fayyad et al. (1996a,b), and how automatic literature clustering can be considered a special case of KDD. In the current version, we examine metadata (such as titles, abstracts, year and venue of publication, and other bibliometric information) gathered by service providers that catalogue this information from scientific articles. The target data is selected by crafting a database query and requesting a metadata dump from suitable service providers. Preprocessing contains format translations and normalization from possibly differing dump formats. Transformation entails listing of unique words in the titles and abstracts, removal of stopwords, and stemmatization of the non-stopwords to produce a binary "bag-of-words"-style matrix where each article is represented as row vectors containing 1s for words that occur and 0s for words that are absent. The data mining step produces a hierarchical clustering of the articles, vi-

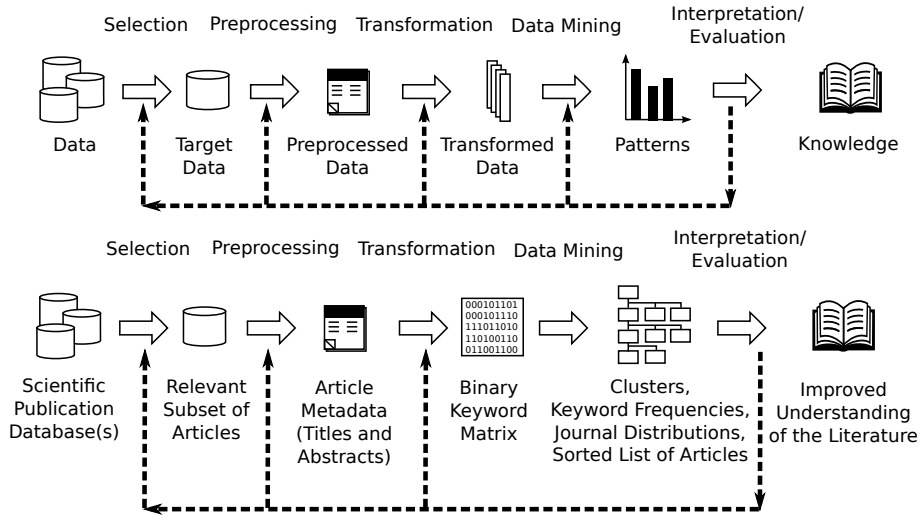


FIGURE 15 Automated literature mapping as an adaptation of the Knowledge Discovery in Databases (KDD) process, as presented in Nieminen et al. (2013), based on Fayyad et al. (1996a,b).

sualizations, and automatic summaries of the clusters. The figure shows an iteration loop, since findings at any stage may necessitate re-working the earlier ones. After human examination of the patterns, the goal is to help the user to gain improved understanding of the selected subset of research.

Similar systems have been actively developed all around the world (e.g. Agarwal et al. 2005; Aljaber et al. 2010; Bravo-Alcobendas and Sorzano 2009; Chen 2006; Cohen et al. 2006; Leydesdorff et al. 2013; Matwin et al. 2010; Szczuka et al. 2012; Tseng and Tsay 2013) and locally (Nurminen et al. 2005). We opted to work on word co-occurrences, which is a traditional approach in informetrics (Callon et al. 1983). The other traditional approach of citation co-occurrences (Small 1973) would require citation information, which we have not yet considered.

The clustering was done by first applying dimensionality reduction via diffusion maps (Coifman and Lafon 2006; Nadler et al. 2008). The method provides a low-dimensional embedding of original data points in which a "diffusion distance" is converted to Euclidean distance. Dimension reduction is commonly used in informetrics (e.g. Boyack et al. 2005; Waltman et al. 2010), but, to our knowledge, we were the first ones to use diffusion maps for a scientometrics application, even if the applicability to document clustering has been noted earlier (Lafon and Lee 2006). The diffusion distance is computed using a Markov process based on a selected distance metric in the original space. In our case, we used the Jaccard distance (Jaccard 1901) which makes vectors with co-occurring bits closer to each other, concentrating on non-zero bits. A kernel function was further used to make the distance differences more prominent. Rows of the distance matrix are then normalized to form a Markov matrix that can be interpreted as transition probabilities ("diffusion") between nearby points, and time steps of diffusion can be simulated efficiently. A lower-dimensional embedding of the original points

can then be formed using only a few first eigenvectors of the resulting matrix. Diffusion maps are able to locate nonlinear manifolds based on their density, and convert this information to a lower-dimensional Euclidean representation.

After the key step of diffusion map embedding, the simplified representation can be clustered using basic methods. We used agglomerative clustering as discussed by Everitt et al. (2011, ch. 4) and Hastie et al. (2011, p. 523), and a silhouette method for selecting the number of clusters as recommended by Rousseeuw (1987). We found that when our clustering method was used for the whole dataset, some small clusters were standing out in the embedding, but a large "residual" bulk was left in the center. Re-clustering of this residual bulk would again separate some new smaller clusters and leave a new residual. We iteratively re-clustered the largest remaining cluster until no further clusters of reasonable size appeared. Automated summaries of the separated clusters were then analysed, and manual inspection of the titles and abstracts in each cluster were used to verify the contents of the clusters and give human-selected names for each automatically found cluster.

4.1.2 New Features

The tool, originally pieced together and initially tested by Nieminen et al. (2013), has later raised some interest among the fellow researchers of its original authors, and the research has continued at the low pace possible alongside everyone's primary topics which do not concentrate around informetrics applications. This dissertation apparently wins the race of becoming the first publication to document some of the developments. Some of the very newest tricks are still left up our sleeves, though, for future publications. Technical tool development has been a balanced team effort for which the author of this dissertation shares credit equally with others. Since there are no prior publications yet to cite, it is necessary to mention that ideas to the methodological side and especially to the prospects of interfacing our tool with qualitative research workflow and human sciences are originally due to Hannakaisa Isomäki, who is currently with the Methodology Centre for Human Sciences, Faculty of Social Sciences, University of Jyväskylä, and with the Faculty of Information Technology, University of Jyväskylä. It turns out that a first, tentative, approach towards this goal is outlined and preliminarily tested in this dissertation.

The tool even in its original form worked surprisingly well, being a first attempt in informetrics by rookies in the field. Many limitations were noticed while performing the initial case study, and solutions for some of them have been figured out. Our data selection, retrieval, and preprocessing toolchain has been revamped completely. Improvements and changes have been made also to both the algorithm and the user interface of the data mining step. Thoughts have been given also to human interaction with the tool. In what follows, these changes are explored with their rationale.

In the earlier trials, we used a web crawler to collect information that was made totally free for the public on the journal publishers' websites. The approach

had its strong appeal in an ideological "free data" sense, but the workload of creating parsers for the various formats of web pages was diverging too far from the more interesting part of processing the harvested publication data. The amount of information made openly available varies between publishers, and can be partly incomplete in some cases. A natural next step for us was to start using the handy export tools of the Web of Science portal (WoK)[1] from which our home university is subscribing to journal databases with access to abstracts and some bibliometric information such as citation indices. Future development directions would involve examining larger databases with conference articles and full texts instead of only abstracts. Fusing data from several providers would be an important step too. As of yet, we have left those up to the bigger players in scientometrics, and decided to experiment with data that is more readily available in the database dumps provided in the standard subscriptions available at our home institution. To extend from using only WoK requires tinkering with data formats, and perhaps with usage agreements, which we have not yet considered.

Our original tool used only the titles and author-selected keywords as the article metadata. Now we use titles and abstracts, which gives us much more data to work on. As for preprocessing, we have adopted some basic tools from the Natural Language ToolKit (NLTK) (Bird et al. 2009), a natural language processing library written for the Python language. Specifically, we currently use its services for common stopword removal and stemmatization of words. More extensive use of NLTK might be beneficial and is under evaluation as of writing this. Stopwords are common words that bear no semantic significance (such as "a", "an", "the", "of", "and", etc. in English). An observation examined earlier, for example, by Zaman et al. (2011) in the context of text retrieval, is that "a tailored stop word list must be assembled for every unique large dataset". Human involvement is required to cleanse away stopwords that may bear important semantic meaning in other contexts but not necessarily in the one being examined. The necessity of this step will be explored further in the case study performed here.

The internal representation of the transformed data is still a binary word occurrence matrix. Also, on each level of clustering, we are still using the same dimension reduction and clustering algorithm as before. The top level of the algorithm has been changed though. Earlier we selected only the largest cluster as the "residual" bulk that was iteratively re-clustered. Even if useful, this approach was not very versatile. In the current version, we recursively re-cluster every cluster that is found. Recursion stops when no further cluster refinement is possible because the cluster size is less than the dimension selected for DM dimension reduction. This is a technical restriction of the DM algorithm used. Also, on each level of clustering, we disregard articles that have no keywords common with any other article in the subset. They remain in the higher level cluster but they do not enter any smaller child clusters. Let us note here that the result of the recursive clustering is different from one level of hierarchical clustering, because the diffusion map embeds the data points differently on each step. Selection of

[1] http://apps.webofknowledge.com/

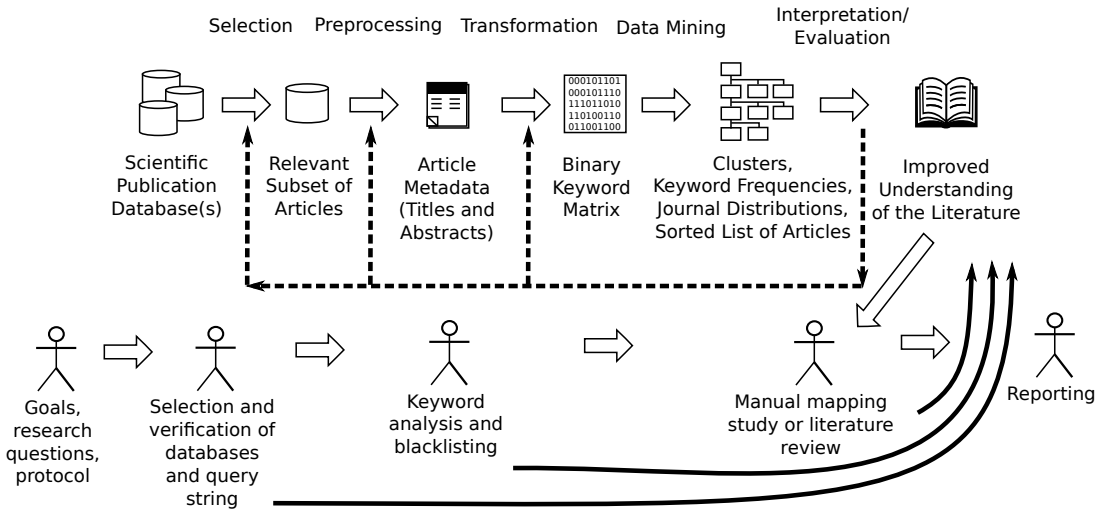


FIGURE 16 Automatic literature mapper as an aid to (human) learning.

clusters for further examination is done by the user *a posteriori*. A similar hierarchical clustering approach was used by Wartiainen and Kärkkäinen (2015) for prototype-based clustering of time series.

The representation of patterns to the user has been revised as well. The user interface tools shown here are all new compared to the original paper, as is the integration to a manual literature overview. In the context of assisting literature reviews, our tool seems to be approaching, for example, those of Malheiros et al. (2007) and Felizardo et al. (2010) in which visual text mining was found to increase efficiency of manual systematic literature reviews. A wide study of available approaches is found in the PhD Thesis of Felizardo (2012). While we have started on the road by implementing the algorithmic data mining step using methods familiar to us from other uses, we are behind the state-of-the-art in user interfaces and ease of use. The current Matlab command interface and text dumps are quite sufficient and convenient for a computational scientist, but such an interface would be next to impossible for someone with no programming experience. The method is a work-in-progress while this dissertation goes to print.

4.2 Application to Multiobjective Machine Learning Literature

Figure 16 shows the steps in which human interaction with the method is required. It turns out that some parts of the "improved understanding" ultimately sought are gained as a result of each necessary interaction step. The steps of (i) query string selection, (ii) keyword analysis and blacklisting, and (iii) actual mapping study will be detailed in the following subsections with the literature on multiobjective machine learning as a sample application.

4.2.1 Selection of Relevant Literature

Crucial to the coverage of relevant target articles is the query string that is used to retrieve a subset of the massive catalogue available. We do not want to miss too many relevant articles, but we want to mitigate noise from totally unrelated ones. The goal here is to select articles that cover *multiobjective machine learning*, with special emphasis on *artificial neural networks*. The simple query string `multiobjective AND "machine learning"` would be much too naïve. Let us see the approximate number (due to the possibility of duplicates between several databases being searched, according to the WoK documentation) of articles returned by some tentative queries using the WoK interface:

- `multiobjective` → 10,648 results
- `multi-objective` → 12,406 results
- `multi-objective OR multiobjective` → 21,653 results
- `multiobjective AND multi-objective` → 1,382 results

An observation can be made that people have generally written their abstracts very well in that they have selected one of two possible spellings and used it consistently (except for some 6% of the authors who have mixed two spellings by purpose or otherwise). To cover all of these articles in the search, we must use the `OR` operator, or a wildcard such as `"multi*objective"`. Here, the operator approach suffices for the purpose, and we end up with a more portable query string that doesn't use special syntax for the wildcard. Let us check if we should add the term "optimization" (or optimisation, for that matter):

- `(multiobjective OR multi-objective) NOT optimi?ation` → 4,576 results

There are some 20% of articles that mention multiple objectives but do not refer to optimization. It turns out (by manually going through some abstracts) that these articles are mostly about optimization, but they contain terms such as "multiobjective evolutionary algorithm" in their abstracts. No noise is expected to appear from leaving out "optimization" from the query, as long as it contains the two forms of "multiobjective". But, for multiobjective optimization, this is not the end of the story. Let us make some more tentative queries:

- `multicriteria` → 6,790 results
- `multi-criteria` → 6,826 results
- `multicriteria OR multi-criteria` → 12,834 results
- `multicriteria AND multi-criteria` → 764 results
- `multi-objective OR multiobjective OR multicriteria OR multi-criteria` → 33,756 results

The field of multicriteria decision making (MCDM) technically deals with different tasks and methods than multiobjective optimization. Also in MCDM, the

goal is to find a Pareto-optimal decision based on conflicting criteria, but the class of problems could be wider than with multiobjective optimization, for example, with qualitative criteria. Some noise from MCDM applications and methods that are not directly usable in machine learning may appear, but it is tempting to cover some 50% more articles than result from querying only with "multiobjective". It is expected that when finally the intersection with "machine learning" will be taken, it will cut off the MCDM articles that are actually irrelevant. There is one more search word that may be quite relevant:

- `Pareto` → 16,027 results
- `multi-objective OR multiobjective OR multicriteria OR multi-criteria OR Pareto` → 43,517 results

It turns out that words like "Pareto front", "Pareto set" and "Pareto-optimality" are sometimes used to imply multiobjective optimization. Use of only "Pareto" as the word requires the system to automatically split words at hyphens, to be able to match "Pareto-optimality". WoK works that way. Let us add just one more search word that probably at least doesn't hurt:

- `("vector optimisation" OR "vector optimization") NOT (multi-objective OR multiobjective OR multicriteria OR multi-criteria OR Pareto)` → 914 results

The original name of "vector optimization" seems to be steadily occurring in current publications that do not refer to the other keywords, so we will finalize the part of the query string that aims to cover "all things multiobjective":

- `multi-objective OR multiobjective OR multicriteria OR multi-criteria OR Pareto OR "vector optimisation" OR "vector optimization"` → 44,460 results

The coverage could be further augmented by listing the names of known methods (such as NSGA, SPEA, PAES, ...), but here we accept any losses incurred by not going so far. The query string is kept reasonably small, and it is expected that we already reach a quite representative set of articles by expecting one of the more general names for multiobjective optimization being present in the metadata.

Now, let us follow a similar logic for literature on *machine learning*. Some tentative queries at first:

- `learning` → 856,766 results
- `"machine learning"` → 27,556 results
- `"artificial neural network"` → 26,768
- `"artificial neural network" NOT learning` → 22,851 results

Searching with the general term "learning" is expected to locate articles related to all kinds of learning, and some kind of restriction to machine learning seems justified. Restriction to "machine learning" is likely to be too narrow, though. For

example, it is possible to find almost 27,000 articles that should quite certainly be about ANNs. In more than 22,000 of these, "learning" is not at all mentioned in the metadata. This is natural, of course, because we can expect everyone to know implicitly that we talk about "learning", when we explicitly talk about ANNs. When it comes to machine learning, it seems justified, unlike in the case of multiobjective optimization, to include general names of some computational structures of greatest interest to the current work:

- `"artificial neural network" OR ANN OR perceptron OR MLP OR RBFNN OR SVM` → 83,860 results

In this way, all general works that care to mention either "artificial neural network" or even the commonly used abbreviation "ANN" will be included. "Multilayer perceptron" and "MLP" will be included similarly. Some studies dealing with the ANN structures most similar to the MLP, i.e., radial basis function neural networks and support vector machines, will get included if they happen to appear in the abbreviated forms of "RBFNN" or "SVM". The selection is *purposely focused around MLPs*. More advanced neural models and other methods of classification and function approximation are purposely excluded. This restriction may need reconsideration if a broader scope is required in another study.

An almost final search query supposed to find literature in the intersection of ANN-biased machine learning techniques and multiobjective optimization thus becomes the following:

- `(multi-objective OR multiobjective OR multicriteria OR multi-criteria OR Pareto OR "vector optimisation" OR "vector optimization") AND ("artificial neural network" OR ANN OR perceptron OR MLP OR RBFNN OR SVM)` → 537 results

The above query string should be quite universally usable. Because the search engine of WoK allows it (as does Scopus, which was used earlier), we shall finally use a wildcard version that allows some more variations (such as "net", "nets", "network", "networks"; "ANN", "ANNs" but not "annihilation", "annealing", etc.):

- `(multi-objective OR multiobjective OR multicriteria OR multi-criteria OR Pareto OR "vector optimisation" OR "vector optimization") AND ("artificial neural net*" OR ANN OR ANNs OR perceptron* OR MLP* OR RBFNN* OR SVM*)` → 693 results

The search might be broadened further, of course. Chances could be taken with more general words like "learning", "supervised", "classification", which would risk a lot of noise from other uses of the words. Or loads of other potentially relevant artefacts and tasks could be listed, like classifier systems, spiking neural networks, ensembles, feature selection, or others. The latter option might not

bring so much noise from altogether unrelated fields, but it would make the selected dataset considerably larger and draw the intended bias away from MLPs and its closest relatives. With the benefits and drawbacks described above, the final query string is now one definition of what is relevant at the intersection of "all things multiobjective, all things MLP, and likely some other more or less related machine learning tools and tasks". Let us be reminded once more that only the WoK databases currently subscribed to by the author's workplace are considered (excluding, for example, the major arena of conference articles). The following semi-automatic study thus operates under the constraints of the defined query string, and the 693 articles whose metadata was retrieved in the standard WoK text format. All publications up to December 2015 were included. Some obviously duplicated or corrupted entries were manually removed, leaving 687 articles. Further manual cleaning of the dump would have been difficult and, as it turns out, unnecessary at this stage.

4.2.2 Keyword Analysis and Manual Stopword Addition

The algorithmic preprocessing step begins by splitting the titles and abstracts of each article into words around whitespace boundaries and converting the words to lowercase. Then, in the current implementation, common English language stopwords are removed by using the default English stopwords of the NLTK corpus[2]. Some pattern matching heuristics are applied to remove copyright statements like "Copyright (C) 2016 Company X Ltd." that some publishers insert into the abstracts. The remaining words are reduced into stems so that inflected forms turn into one. For example, "hobbit", "hobbits", and "hobbitses" will all turn out as "hobbit". Stemmatization is performed by the Snowball stemmer[3] via the NLTK toolkit[4]. While experimenting with the toolkit, we were deeply impressed especially of the example with hobbitses. NLTK turned out to work very reliably overall, so currently we are happy to leave our stemmatization to it. In our data of 687 articles, the metadata contained 56,420 non-stopword stems, 5,617 of which were unique. One of the first outputs to examine is the overall frequency of most common words present in the dataset.

As an example, the "top 50" frequent word stems in this case were the following (the total number of articles using each word is given in parentheses): *use (564), optim (489), network (486), neural (474), model (462), result (430), multi (419), algorithm (417), object (416), base (409), artifici (395), method (338), perform (337), paper (326), propos (314), approach (314), process (281), problem (274), studi (262), function (259), genet (255), data (254), design (250), obtain (249), present (247), paramet (241), develop (241), ann (234), two (224), set (220), predict (213), select (211), techniqu (210), compar (210), show (209), solut (207), system (205), differ (204), appli (201), effect (197), time (196), comput (189), improv (186), combin (183), train (173), pareto (173), applic (168), machin (167), effici (166), analysi (160).* Complete word

[2] `nltk.corpus.stopwords.words('english')`

[3] http://snowballstem.org/

[4] `nltk.stem.snowball.EnglishStemmer`

lists are available at the author's home page[5]. Presence or absence of each of the 5,617 unique words in the title or abstract of an article could be used as a binary feature vector $\mathbf{x} \in \{0,1\}^{5617}$ representing somehow the contents of each article. We would then like to perform the unsupervised learning task of clustering to break the bulk apart into sets of internally similar ones.

From the output, we can observe that further application-specific, subjective, knowledge must be used to exclude some words in addition to the standard stopwords. One reason for this necessary manual intervention is that we know what kind of literature we are dealing with, and we want to filter the document representation based on this perspective. We know that to "use" something is probably common to every research, regardless of whether it is explicitly mentioned in the abstract or not. We expect no real discriminative power from words such as "use", "employ", "apply". An automatic clustering method would happily separate clusters based on the specific choice of words in each abstract, whereas we would like the separation to be made on contents, not format. Thus, such words should be identified and removed from the representation.

Quite a few words common in scientific parlance should be excluded based on similar arguments. The stem "argu" would be on the list of removals (because arguing should always happen), as would be "result" (because every research should have some). On the same grounds, removed should be also "algorithm", "method", "approach", and other variants (because in this field, every research is likely to either present or use one or more approaches). Also "problem", "solut", "studi", "develop", "improv" present similar problems.

Not all of the common words are so easy to decide upon, though. We know that "control" and "respons" could be common in any algorithms that are controlled or responding, but in this case they might imply the class of *control problems* or *response surface models* and we might want to involve such different applications in the clustering process. Currently, the user of our method needs to make a decision about all of the words, and there might be a need to go back to the blacklisting process, if the found clusters seem to be too clearly separated on false grounds.

One more argument for manual word removal is that we know our original search string, but the clustering method does not, so we must involve this information in the preprocessing. Firstly, we know that every selected article must contain one of the OR'ed variants that should indicate multiobjective optimization. The specific wordings chosen by authors should play no role in clustering, so these are better to be removed. The case is not so simple with the AND'ed part about machine learning and neural networks with bias towards MLPs. We might want the different types of learning machines to play a role in clustering, so a decision is made to *keep* the words related to types of ANNs or other specific learning machines, but *remove* the words related to learning, because we know that ANNs are a subcategory of learning, and basically each article in the dataset should be about learning, whether it is mentioned in the abstracts or not.

Unfortunately, a subjective bias will be introduced along with possible ex-

[5] http://users.jyu.fi/~nieminen/research/dissertation2016/survey/

tra noise. For example, removal of stems like "appl", assumed to be synonymous with "use", is likely to remove also potentially interesting information about applications versus pure method development. The word "vector" must be removed as it is part of the original query string for "vector optimization", with the hope that the word "support" would still help in discriminating "support vector machines" from other kinds of learning models. Following this logic, "machine" may be removed, too, but we accept noise from other uses of "support". For example, some evidence could be said to "support" a conclusion, which is again usual in every research regardless of the words used. This part of the process is admittedly subjective, and it reflects the expectations of the person who decides to omit some words from the representation and leave others untouched. An alternative could be to consider some sets of words as synonyms, based on human decision or preferrably some stemmatized thesaurus corpus. More robust results could result from using combinations of consecutive words and from covering full texts instead of abstracts. It is up to future work to order larger dumps from the scientific databases to test our method with such schemes.

The following stems were removed based on knowledge about the query string: *optim (489), multi (419), object (416), function (259), pareto (173), machin (167), criteria (138), multiobject (134), learn (111), vector (110), minim (103), multipl (99), optimum (74), maximum (64), domin (54), minimum (51), conflict (43), multicriteria (33), criterion (31), optimis (27), nondomin (20), compromis (17), mcdm (15)*.

In addition, 342 words were removed based on the assumption that in the expected parlance they are most commonly used for purposes that are not related to the actual topic. Again, a complete list of these words is available at the author's website. Here are the most common ones removed: *use (564), result (430), algorithm (417), base (409), method (338), perform (337), paper (326), propos (314), approach (314), problem (274), studi (262), data (254), design (250), obtain (249), present (247), paramet (241), develop (241), set (220), select (211), techniqu (210), compar (210), show (209), solut (207), system (205), differ (204), appli (201), effect (197), time (196), comput (189), improv (186), combin (183), applic (168), effici (166), analysi (160), evalu (156), also (154), experiment (149), provid (148), valu (145), consid (145), decis (137), variabl (134), test (133), methodolog (132), determin (132), well (128), new (128), number (121)*.

Actually, there is a rather useful outcome from the necessary step of examining the words and their frequencies. During the process, one must unavoidably become aware of the words most commonly mentioned in the literature, and some insight can be gained as to what things have been considered more often than others. For example, even when the query string did not restrict the search to genetic or evolutionary algorithms, these are certainly in the top classes of algorithms in use, as can be seen from the stems *genet (255)* and *evolutionari (110)*. Support vector machines rank high, *svm (81)*, even when the MLPs were biased, as do fuzzy systems, *fuzzi (86)*. Of particular optimization methods, NSGA or NSGA-II are definitely most often mentioned, as we have the stem *nsga (52)* on top of any other stems that could imply a certain choice of an optimization algorithm. Swarm intelligence has been popular, too, because we have the stem

swarm (42). Regarding applications, it seems that "all things water" have been very much in focus, as the stem *water (48)* reveals. A closer look at the titles and abstracts mentioning "water" reveals that they consider such things as groundwater resources, fresh and waste water management, and saltwater intrusion. Such important application areas are quickly revealed by the keywords and their relatively high frequencies, and they may surprise a researcher whose own backyard has not yet been affected by these globally urgent problems.

If just a quick overview on what has been done most often is required, a simple keyword analysis turns out to be quite handy. Instead of going through 687 titles and abstracts, it suffices to start from the top of the word list and look at the most common ones. All of the stems mentioned above are found within the 300 most common ones in the original data. More insights appear as one proceeds down the list, and this happens only by looking at one word and its frequency at a time. It is a lot faster to carefully examine 300 words than to read through 56,420 words of text.

4.2.3 Structural View using Clustering

Now the target body of knowledge is singled out by defining a query string and by dumping article metadata from a database known to index a wide range of peer-reviewed scientific publication fora. The frequency of keyword occurrences has been examined, already revealing new insight on what most of the research has been about, and the keywords have been pruned by subjective expectations of what can really make a difference between different kinds of studies and what is likely just a difference in the choice of wordings. The real *magnum opus* would then be to automatically cluster the set of articles to categories that would provide yet more insight into what kind of subtopics are involved in the field, how active or recent these subtopics would be, and which would be the most relevant fora that publish results on them. We expect no "magic wand" that immediately enlightens us more than just patiently reading through all the research, making notes, and producing a proper synthesis by traditional means. But we do wish for another easy tool that could help in cases where time is limited and even a partial insight is better than no insight at all. Also, substantially larger sets of original research can be explored automatically than manually, which may help in the initial phase of a mapping study by delaying any too harsh decisions about how to restrict the final search.

Figure 17 shows the results of the hierarchical clustering as a tree. The root node, labeled as "cluster 1", represents the original bulk of articles examined. On the first level, it has been split into two clusters, labeled with running indices as "cluster 2" and "cluster 3". Proceeding recursively one depth level at a time, "cluster 2" was further split into clusters 4, 5, 6, 7, and 8. Similarly, "cluster 3" was split into clusters 9, 10, and 11, and so on. The area of each marker is proportional to the number of articles assigned to the respective cluster. The color bar further helps to quantify the size of each cluster in this illustration. On each level, or depth, each article may have been assigned to a new cluster, smaller than

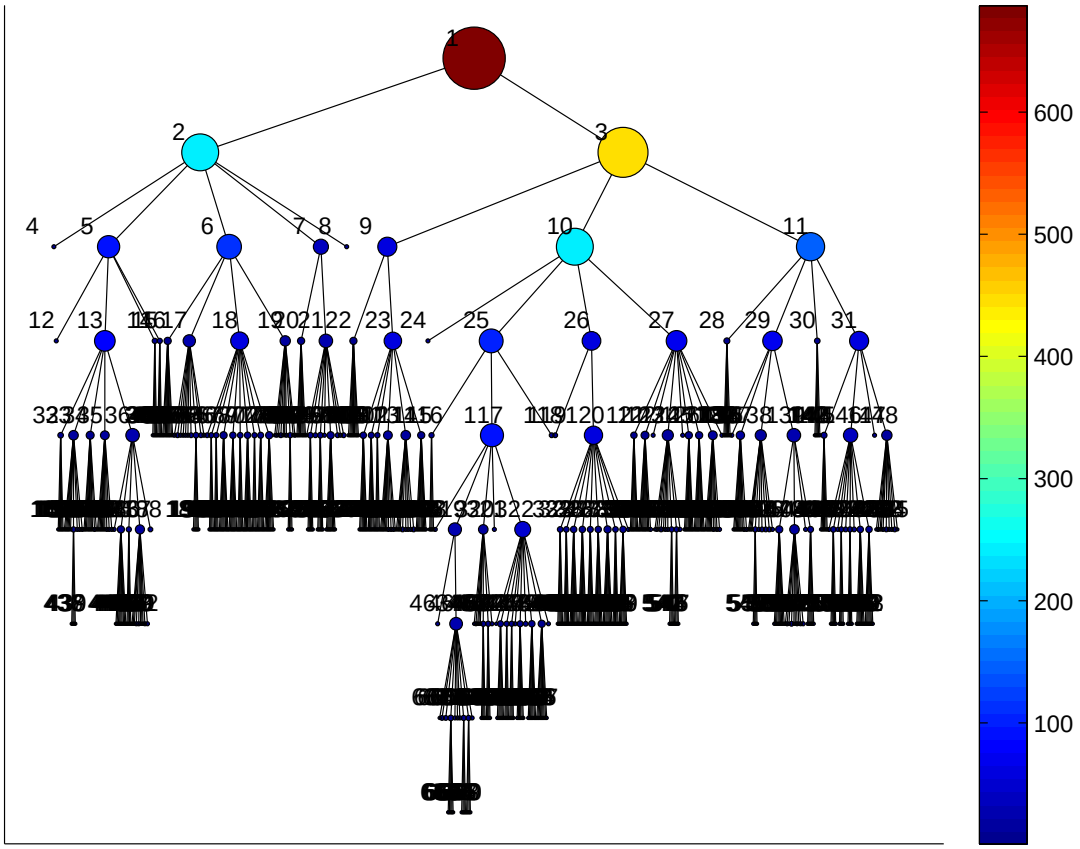


FIGURE 17 Results of the recursive literature clustering.

the "parent cluster" on the previous level. Thus, the granularity of the clusters increases with the depth. The leaf nodes depict tiny sub-clusters of one or very few articles that cannot be split into smaller ones anymore.

Figure 18 shows the main bulk ("cluster 1") of articles embedded by the diffusion map into the three dimensions used in this case study. Each marker in the plot represents one of the 687 articles examined. The marker types and colors represent the first level clustering result, where each article was assigned to either "cluster 2" or "cluster 3". Visually, it could be possible to find even as many as three clusters, but the silhouette selection method (Rousseeuw 1987) has decided that two is a good number of clusters on this level.

It is possible that better clustering results could have been obtained by using more dimensions, but it was decided that reduction to just three dimensions better serves the purpose of visualizing the workings of the method. In this case, the clustering output is in good agreement with visual inspection of the point cloud in the Euclidean space. On this level, it is of course impossible to say anything about the workings of the diffusion map embedding or the meaningfulness of the representation scheme and distance metric chosen.

Similarly, Figure 19 shows the breakdown of "cluster 2" into clusters 4, 5, 6, 7, and 8 by recursive re-clustering. Figure 20 shows the breakdown of "cluster 3" into clusters 9, 10, and 11. Again, the clustering seems to operate meaningfully on the articles embedded in the Euclidean space. On this level, some very

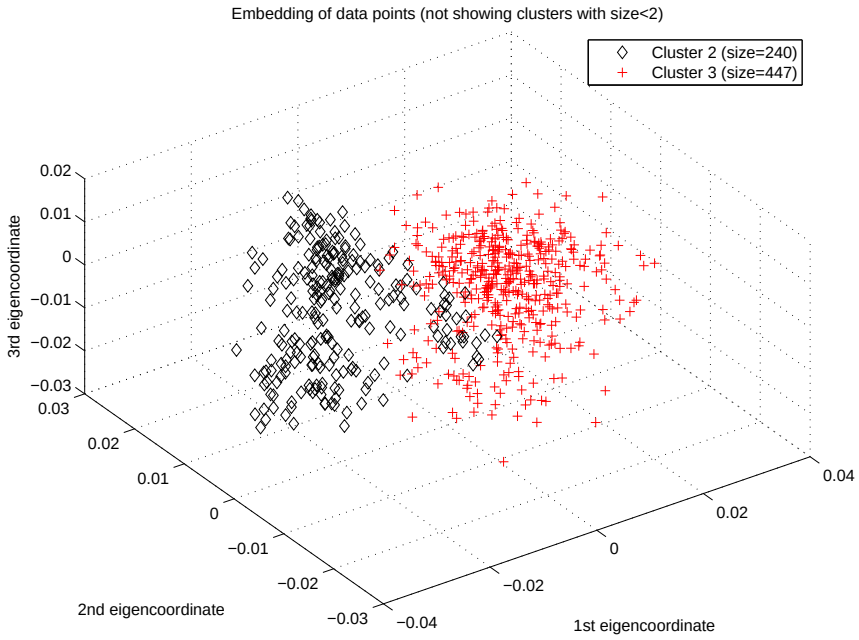


FIGURE 18 Diffusion map embedding of the main bulk of articles (level 1). Two new clusters were found, and running indices were given as their labels.

small clusters appeared, strongly separated from the others. They turned out to contain duplicate entries in the original database dump with minor differences or misspellings in author names, titles, or other metadata fields. Such duplicates were not invited but had been included in the data. It seems that our method is able to spot such anomalies by showing them as tiny clusters on the top levels. In a realistic scenario, such duplicate entries would be removed at the earliest time of discovery, and the algorithm would be run again, likely giving increasingly better results as the data is cleaned. To maintain the flow of demonstration, such iteration was not performed in this study. All the plots, labels, and results would necessarily be different after any cleaning. This study was made in such a backward way for a reason, though. A manual walkthrough of the 687 titles and abstracts was performed before looking at the machine generated clusters in depth, in order to be better able to assess the meaningfulness of the clustering without prior bias. In the intended final application, the clusters should be preliminarily analysed as early as possible.

This facet of applicability with data cleaning was actually first noticed during this case study. The dataset had already been manually cleaned, which takes some time and effort. Afterwards, when turning to actual clustering results, it turned out that the manual cleaning step had been incomplete, and it may not have been even necessary in the first place – duplicates and outliers could be much more easily spotted from the first levels of clustering. Further examination of small clusters that appeared on early levels of the hierarchical clustering revealed more of these duplicates, which is very helpful in fast cleaning of the possibly noisy database dump.

The system is also very cruel and effective in spotting situations in which

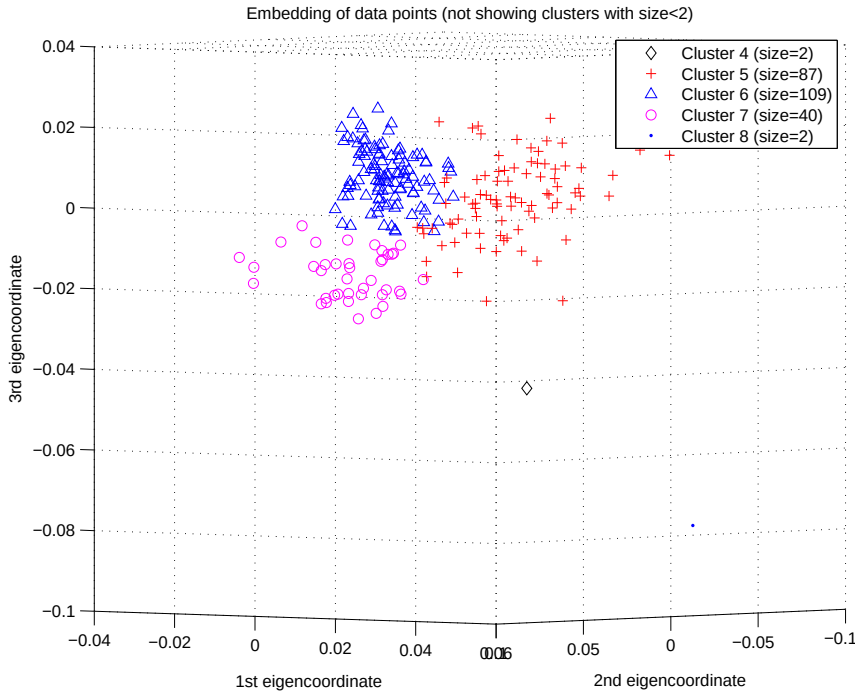


FIGURE 19 Diffusion map embedding of cluster 2. Five smaller clusters were found, the smallest of which revealed duplicate entries in the original data.

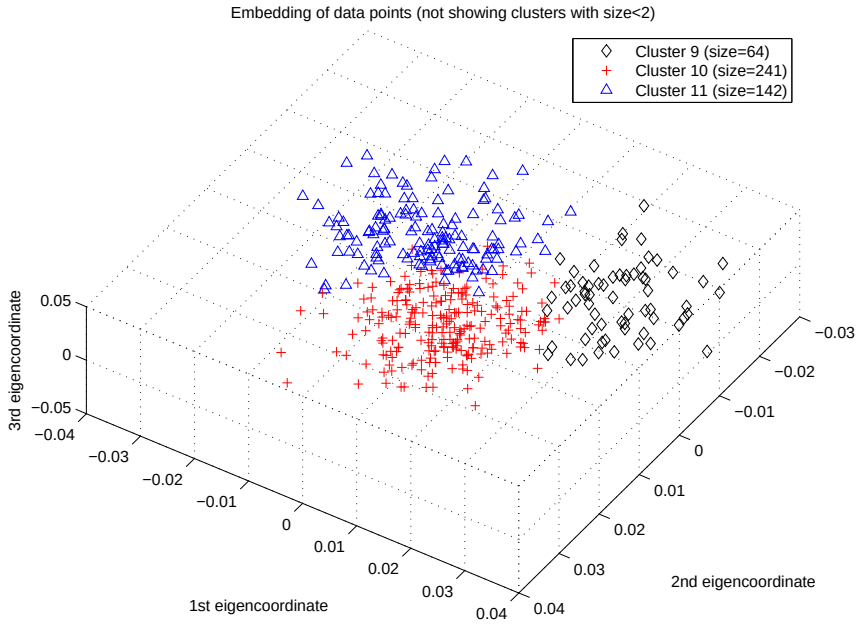


FIGURE 20 Diffusion map embedding of cluster 3. Each of the clusters was obtained by applying the same method recursively.

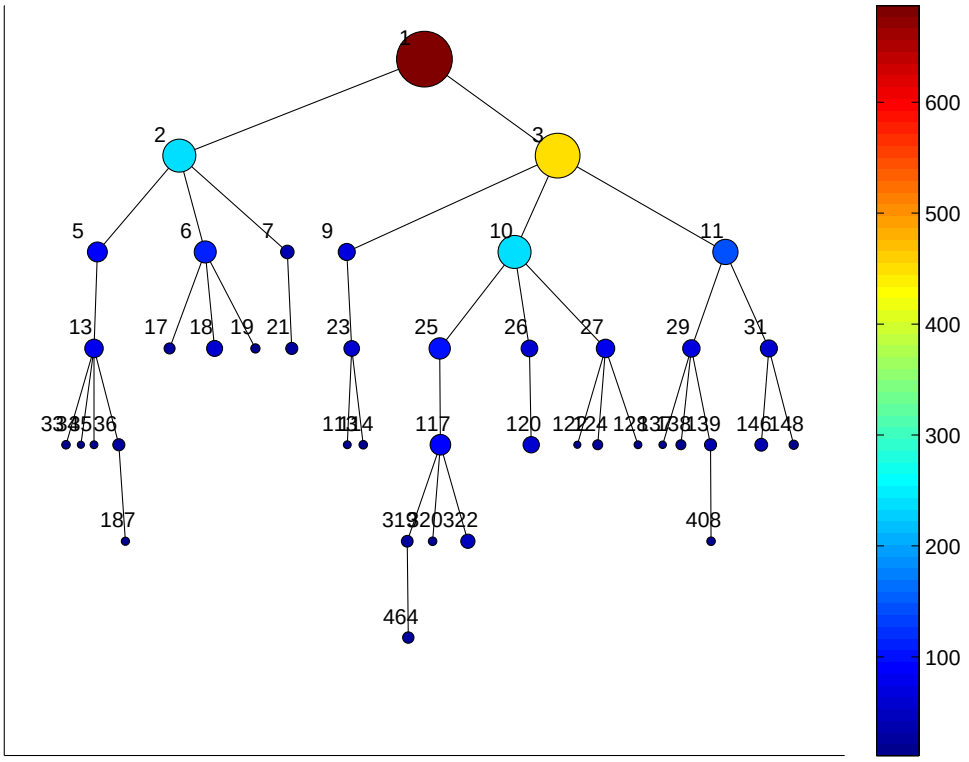


FIGURE 21 Results of the recursive literature clustering, omitting clusters that have less than 10 articles.

research output has apparently been maximized by means on the verge of self-plagiarism: some articles published with different titles in different journals described almost the same findings in slightly different terms in their abstracts. Our clustering method easily revealed those in the same way as other duplicates. Let us politely refrain from citing those works here, though.

For the goal of structuring the body of literature in clusters of useful size, a pruned view is more useful. Figure 21 shows only clusters with at least 10 articles. As of now, we feel that it is best to leave it up the user to select specific clusters that match the level of granularity wanted for the purpose at hand. For this case study, a subjective choice of reasonably sized clusters at shallow depths was made. Specifically, found in a left-to-right order in the figure, the nine clusters with indices 13, 6, 21, 23, 117, 120, 27, 29, and 31 were selected for further exploration. Any other choice could have been made, and also this step might involve an iteration loop. The selection here is done for purposes of explaining the method, as the actual literature examination had already been manually done. It is intended that the required human interaction increases the overall understanding of the literature also at this step, possibly without reading any or much of the material as of yet.

4.2.4 Automatic Summaries of Cluster Contents

Tables 1, 2, and 3 show automatic summaries of the contents of each hand-picked cluster. The title rows also contain tentative names subjectively given to each cluster by examining the summaries. Below each title, the cluster size is given. Then, in the second part, the ten most common keywords in the cluster are listed along with the percentange of articles within the cluster that contain each keyword.

The next part, labeled "Ancestor words", contains a list of word pairs that can be used to track the cluster tree from the main bulk down to the selected cluster. The two most common keywords are given for each ancestor cluster without repeating any of the same keywords down the recursion. Thus, the ancestor keywords give some coarse idea about how the recursion has progressed. The most common words in the initial bulk of data are, expectedly, "network" and "neural". Then, for example, Cluster 13 has decended from a parent cluster whose most common words (omitting those of its ancestors) have been "classif" and "classifi". That parent itself has been a child of a cluster with very common words "support" and "svm", and so on. This example, among some others, gives further evidence of the need of handling synonyms in future versions of the method.

The fourth part, labeled "Local words", lists the keywords and frequencies of keywords in the cluster that have not occurred in the ancestor keywords. This part is intended to give yet further clues about which words may have played a role in the formation of the cluster under examination.

Finally, three article titles are given that appear closest to the geometric center of the cluster within the embedded coordinates of its parent. More, or even all, titles could be listed in the examination phase. Three are selected here to conserve space in the tables. As a compromise solution to aesthetic and technical issues partly beyond our control (various title formats in the original database dump), the titles are printed in their normalized lowercase format.

These summaries offer only a very coarse way of examining the clusters, and further human examination of complete contents of the clusters would be necessary for more accurate assessment. Another thing quite evident from the summaries is that in a real application the manual stopword lists should probably be extended, because at this point common words such as "model", "one", or "two", which may have little semantic meaning, still appear in the summaries. The clustering should then be run again, and, as with other steps in the process, new and possibly improved results would be available for re-examination in a few seconds.

Even without such re-iteration, at least some of the cluster summaries appear to give quite reasonable information about the possible contents of the cluster. For example, Cluster 13 quite certainly deals mostly with SVMs, as the keywords "support" and "svm" are very frequent (over 70%) in the overall keyword list. The immediate parent cluster is about "classification". Therefore, the name "SVM classification" feels justified. For the next one, Cluster 6, no keyword stands out as prominently. Yet, the words "process" and "complex" may give some clues as to the cluster contents, somewhat supported by the example ti-

tles near the cluster center. The name "Process modeling using EAs" has been selected, because also terms related to GA appear in the list. The other clusters were tentatively named following a similar reasoning.

In order to reduce the number of clusters for the purposes of this dissertation, the selected clusters remain quite large. The smaller child clusters found in the lower levels of the recursion could be easier to examine and categorize. This step of closer cluster examination is completely a human task. Its goal is to shed more light into the bulk of literature and give coarse ideas about how it may be structured internally. Again, if a quick overview suffices, the process can be ended here.

4.2.5 Journal Distribution

Tables 4, 5, and 6 show the number of articles within each selected cluster that have been published by different journals. The journal titles are formatted in lowercase due to the same, unfortunate, technical issues as the article titles were. Only journals that have published at least two papers in each cluster are listed. This format of showing the distribution of papers in the publication venues is different from that in our original method. In this case, the original database dump contained articles from more than 350 journals, so it makes no sense to list them all. The journal names seem to be somewhat in line with the cluster names based on examination of the keywords. The idea of these journal distribution tables is to reveal publication venues that are inclined to publish literature on certain topics or topic groups, as defined by the clustering.

4.2.6 Manual Overview with Re-ordered Titles and Abstracts

Figure 22 depicts the whole literature bulk sorted according to the clustering. Each position on the horizontal axis corresponds to an article. The top row shows the first level as a box. On the second level, the articles have been sorted so that the first contiguous part contains articles of Cluster 2 and the second one contains articles of Cluster 3. On each level of the recursion, the articles are sorted again similarly within their parent cluster. The bottom row shows the final clustering (leaf nodes). By then, the articles have been sorted in a "dictionary order" so that each of them are close to others that have been clustered similarly on each level of the recursion.

As explained earlier, the titles and abstracts of all articles were also examined and manually clustered by the author in a two-pass recursive fashion prior to examining the automatic ones. During the first pass, each article was assigned to one of two categories: (i) one containing articles that are clearly about developing or comparing methods and (ii) another one that is more about some specific application of machine learning. The first pass was to be made quickly and thus mostly based on titles only. Abstracts were to be checked only when the title was unclear about the purpose of the article. Some of the first-level decisions had to be reverted while performing the second pass in which the two previously

TABLE 1 Automatic summaries of literature clusters 13, 6, and 21.

Cluster ID: 13	*(SVM classification)*
Size:	77
Keywords:	support(75%) svm(73%) classif(69%) model(51%) classifi(48%) two(45%) accuraci(35%) train(32%) featur(31%) process(31%)
Ancestor words:	(network neural), (model two), (support svm), (classif classifi)
Local words:	accuraci(35%) train(32%) featur(31%) process(31%) class(30%) sampl(30%) one(29%) predict(29%) genet(27%) singl(25%)
Example titles:	(1) assuring the authenticity of northwest spain white wine varieties using machine learning techniques (2) accurate and resource-aware classification based on measurement data (3) classification as clustering: a pareto cooperative-competitive gp approach
Cluster ID: 6	*(Process modeling using EAs)*
Size:	109
Keywords:	model(59%) process(36%) two(27%) complex(26%) oper(24%) high(23%) genet(22%) generat(21%) simul(21%) cost(19%)
Ancestor words:	(network neural), (model two), (process complex)
Local words:	oper(24%) high(23%) genet(22%) generat(21%) simul(21%) cost(19%) integr(19%) reduc(19%) one(18%) predict(18%)
Example titles:	(1) a study on uncertainty-complexity tradeoffs for dynamic nonlinear sensor compensation (2) de novo design: balancing novelty and confined chemical space (3) multiobjective linear programming model on injection oilfield recovery system
Cluster ID: 21	*(Prediction, estimation, and regression)*
Size:	32
Keywords:	model(97%) neural(91%) network(88%) predict(75%) artifici(59%) estim(53%) two(47%) high(44%) train(44%) accuraci(34%)
Ancestor words:	(network neural), (model two), (predict artifici), (estim high)
Local words:	train(44%) accuraci(34%) ann(34%) distribut(34%) addit(31%) one(31%) compon(28%) input(28%) process(28%) regress(28%)
Example titles:	(1) evaluation of an integrated modelling system containing a multi-layer perceptron model and the numerical weather prediction model hirlam for the forecasting of urban airborne pollutant concentrations (2) improvement in estimation of soil water retention using fractal parameters and multiobjective group method of data handling (3) learning based brain emotional intelligence as a new aspect for development of an alarm system

TABLE 2 Automatic summaries of literature clusters 23, 117, and 120.

Cluster ID: 23	*(RBF and MLP with EAs)*
Size:	55
Keywords:	network(98%) neural(91%) two(56%) model(47%) genet(45%) accuraci(44%) basi(44%) classif(44%) evolutionari(40%) perceptron(40%)
Ancestor words:	(network neural), (artifici model), (two train), (genet accuraci)
Local words:	basi(44%) classif(44%) evolutionari(40%) perceptron(40%) error(38%) radial(38%) layer(31%) mean(31%) predict(29%) classifi(27%)
Example titles:	(1) modelling commodity value at risk with psi sigma neural networks using open-high-low-close data (2) cooperative coevolution of artificial neural network ensembles for pattern classification (3) time series forecasting by neural networks: a knee point-based multiobjective evolutionary algorithm approach
Cluster ID: 117	*(ANN for industrial processes)*
Size:	94
Keywords:	network(98%) neural(98%) artifici(85%) process(67%) model(55%) ann(35%) complex(34%) genet(30%) evolutionari(29%) error(21%)
Ancestor words:	(network neural), (artifici model), (process genet), (ann complex), (evolutionari error)
Local words:	search(19%) train(19%) two(19%) engin(18%) oper(18%) predict(18%) industri(17%) inform(17%) one(16%) structur(16%)
Example titles:	(1) food processing optimization using evolutionary algorithms (2) thermochromic sensor design based on fe(ii) spin crossover/polymers hybrid materials and artificial neural networks as a tool in modelling (3) coupled data-driven evolutionary algorithm for toxic cyanobacteria (blue-green algae) forecasting in lake kinneret
Cluster ID: 120	*(Simulation models for industry and management)*
Size:	59
Keywords:	network(98%) neural(97%) artifici(92%) model(86%) simul(59%) manag(51%) ann(42%) generat(32%) control(29%) oper(29%)
Ancestor words:	(network neural), (artifici model), (process genet), (simul manag), (ann generat)
Local words:	control(29%) oper(29%) water(29%) one(25%) two(25%) adapt(24%) high(24%) increas(24%) intellig(24%) level(24%)
Example titles:	(1) high-level design space exploration of locally linear neurofuzzy models for embedded systems (2) artificial neural networks and multicriterion analysis for sustainable irrigation planning (3) robust design of mars entry micro-probe with evidence theory and multi-fidelity strategies

TABLE 3 Automatic summaries of literature clusters 27, 29, and 31.

Cluster ID: 27	*(Process modeling and optimization)*
Size:	76
Keywords:	network(97%) neural(97%) artifici(84%) model(62%) genet(61%) process(50%) simul(39%) reduc(37%) ann(34%) first(34%)
Ancestor words:	(network neural), (artifici model), (process genet), (simul reduc)
Local words:	ann(34%) first(34%) order(33%) generat(32%) approxim(29%) two(29%) train(28%) evolutionari(25%) dynam(24%) final(24%)
Example titles:	(1) identification of constitutive parameters using hybrid ann multi-objective optimization procedure (2) robust multi-fidelity design of a micro re-entry unmanned space vehicle (3) modelling and optimisation of laser shock peening using an integrated simulated annealing-based method
Cluster ID: 29	*(Modeling and prediction)*
Size:	68
Keywords:	network(100%) neural(99%) artifici(94%) ann(90%) model(88%) genet(69%) predict(62%) condit(47%) oper(47%) ga(43%)
Ancestor words:	(network neural), (artifici model), (ann predict), (genet condit)
Local words:	oper(47%) ga(43%) process(35%) temperatur(31%) two(29%) rang(28%) train(28%) high(26%) order(25%) ratio(25%)
Example titles:	(1) airflow and temperature distribution optimization in data centers using artificial neural networks (2) modeling and genetic algorithm-based multi-objective optimization of the med-tvc desalination system (3) separation of toluene/n-heptane mixtures experimental, modeling and optimization
Cluster ID: 31	*(Engineering and machining applications)*
Size:	63
Keywords:	network(100%) neural(98%) artifici(97%) model(86%) process(81%) ann(73%) predict(60%) surfac(52%) input(43%) train(43%)
Ancestor words:	(network neural), (artifici model), (ann predict), (process surfac)
Local words:	input(43%) train(43%) genet(38%) cut(37%) materi(35%) output(35%) import(30%) qualiti(30%) product(29%) rate(29%)
Example titles:	(1) influence of ultrasonic and microwave irradiation on cation exchange properties of clay material (2) development of a closed-loop diagnosis system for reflow soldering using neural networks and support vector regression (3) development of multi-objective optimization models for electrochemical machining process

TABLE 4 Top publishing venues within clusters 13, 6, and 21.

	Cluster ID: 13 *(SVM classification)*
6	neurocomputing
4	ieee transactions on geoscience and remote sensing
3	european journal of operational research
3	expert systems with applications
3	knowledge-based systems
2	annals of operations research
2	bmc bioinformatics
2	chemometrics and intelligent laboratory systems
2	ieee transactions on nanobioscience
2	ieee transactions on neural networks and learning systems
2	plos one
46	other (only one article in each)
	Cluster ID: 6 *(Process modeling using EAs)*
6	applied energy
5	european journal of operational research
4	expert systems with applications
4	materials and manufacturing processes
3	energy
2	applied mathematics and computation
2	chemical engineering journal
2	fuel
2	journal of the brazilian society of mechanical sciences and engineering
2	water resources management
77	other (only one article in each)
	Cluster ID: 21 *(Prediction, estimation, and regression)*
3	journal of hydrology
2	ecological modelling
2	environmental modelling & software
25	other (only one article in each)

TABLE 5 Top publishing venues within clusters 23, 117, and 120.

	Cluster ID: 23 *(RBF and MLP with EAs)*
7	neurocomputing
3	applied soft computing
3	ieee transactions on neural networks
2	information sciences
2	neural computing & applications
38	other (only one article in each)
	Cluster ID: 117 *(ANN for industrial processes)*
4	applied soft computing
3	computers & industrial engineering
3	journal of intelligent manufacturing
2	artificial intelligence in medicine
2	european journal of operational research
2	industrial & engineering chemistry research
2	international journal of advanced manufacturing technology
2	international journal of innovative computing information and control
2	international journal of production research
2	kybernetes
2	materials & design
2	materials and manufacturing processes
2	neural networks
2	sensors and actuators b-chemical
62	other (only one article in each)
	Cluster ID: 120 *(Simulation models for industry and management)*
6	water resources management
3	hydrological sciences journal-journal des sciences hydrologiques
2	energy and buildings
2	environmental modelling & software
2	international journal of geographical information science
2	journal of hydrology
2	proceedings of the institution of mechanical engineers part a-journal of power and energy
2	water science and technology
38	other (only one article in each)

TABLE 6 Top publishing venues within clusters 27, 29, and 31.

	Cluster ID: 27 *(Process modeling and optimization)*
4	applied thermal engineering
3	composite structures
2	building and environment
2	computational materials science
2	energy
2	engineering applications of artificial intelligence
2	ieee transactions on magnetics
2	international journal of advanced manufacturing technology
2	journal of mechanical science and technology
2	materials & design
53	other (only one article in each)
	Cluster ID: 29 *(Modeling and prediction)*
3	energy conversion and management
2	aiche journal
2	chemical engineering journal
2	desalination
2	desalination and water treatment
2	environmental monitoring and assessment
2	journal of the taiwan institute of chemical engineers
53	other (only one article in each)
	Cluster ID: 31 *(Engineering and machining applications)*
12	materials and manufacturing processes
5	international journal of advanced manufacturing technology
4	computational materials science
2	journal of intelligent manufacturing
2	journal of materials processing technology
2	materials & design
2	optics and lasers in engineering
2	proceedings of the institution of mechanical engineers part b-journal of engineering manufacture
32	other (only one article in each)

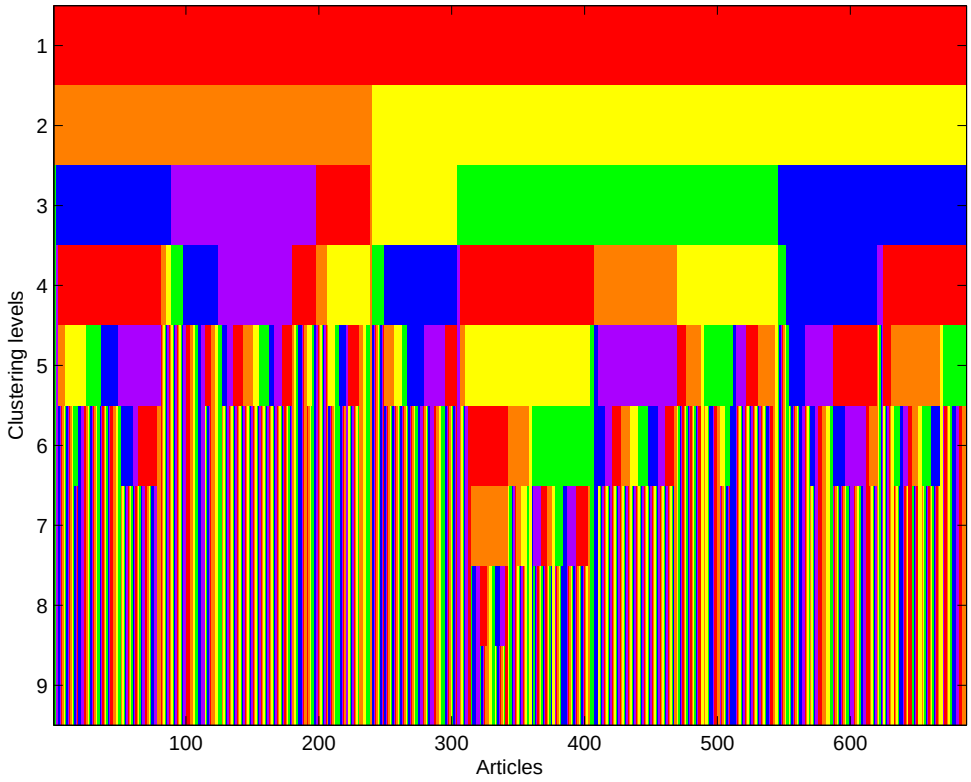


FIGURE 22 Sorting the literature based on clustering to benefit manual examination of the bulk.

found categories were sub-categorized into different genres of method development and applications, and the abstracts were always checked in addition to the titles.

Such crisp "clustering"-like categorization does not feel very natural for literature mapping, because usually the articles deal with, for example, both method development and at least one real-world application that has motivated or supported the development. In this study, the crispness was maintained to match the automatic clustering. Nevertheless, for future versions, we should consider allowing our literature clustering system to generate fuzzy, overlapping, clusters, which could match the real literature contents better than crisp, or "hard" clustering.

As for the applicability of the current methodology, the dictionary ordering made it much easier to read through the titles and abstracts. The observation is based on subjective, but very convincing, experience of the manual walkthrough. Despite some inevitable noise, each article in the re-ordered list is always somehow talking about matters similar to the previous one, which appears to lessen the cognitive load. There is no need to completely reset one's head after reading one title (and abstract, if need be), because there is already a suggested possibility of arriving in a similar conclusion as with the previous title examined. Care must still be taken to verify one decision or another, but the similarity of consecutive articles makes the task at least much more comfortable compared to random ordering.

After the two passes, the following crisp categories were observed by the author :

1 Applications (of combinations of learning and MOO)
   1.1 Medicine, biology, biomechanics, bioinformatics, and chemistry (incl. process technology)
   1.2 Engineering, manufacturing, logistics
   1.3 Environmental management, ecology, and remote sensing
   1.4 Business, ERP, and finance
   1.5 Miscellaneous applications
2 Method development (of combinations of learning and MOO)
   2.1 MLP, SLFN, RBFNN, or ANN learning
   2.2 Strictly SVM learning
   2.3 Optimization or Decision making methodologies (instead of learning *per se*)
   2.4 ROC point of view
   2.5 Miscellaneous hybrid methods
3 Query noise (i.e., unrelated to combinations of learning and MOO)

Meanwhile, on each pass, a separate shortlist was updated of articles that seemed to be most important to the focus of this thesis. About articles published prior to 2006, it was checked that the same topics had been covered by the articles cited in Section 3.2. Regarding articles published during 2006–2015, those whose abstracts seemed to promise some comparatively new developments or points of view were selected for full-text study and digesting in Section 3.3.

Certainly, a third pass would have resulted in more corrections to previous assignments and a more fine-grained categorization. Such detail was deemed unnecessary as of now, because the two passes already made the author confident enough that the goals set for this study (listed on page 85) had been met. Specifically:

1. No early studies were found that would differ greatly from the findings of Section 3.2.
2. The author is confident that the selection of references for Section 3.3 form a representative (although far from complete) set of developments in multiobjective supervised ANN learning published during the last decade, and that the summaries in Sections 3.2.11 and 3.3.10 can be used as the basis of a state-of-the-art implementation in 2016.
3. The automatic literature clustering method turns out to be useful for its original and intended purposes. Also, additional uses were discovered in this case study, and several future development ideas emerged, as discussed previously in this section.
4. A more complete secondary study focusing on multiobjective supervised learning research of the last decade seems to be lacking, and the preliminary steps documented here could be used to focus and plan such a study.

TABLE 7 Comparison of automatic and manual clustering: the first level of recursion against the first manual pass.

	1	2	3
c 1 (sz=687):	76.6	18.0	5.4
c 2 (sz=240):	62.9	21.7	15.4
c 3 (sz=447):	83.9	16.1	0.0

4.3 Comparison of Automatic and Manual Clustering

A remaining question of interest is how much the automatic clustering resembles the manual one. Tables 7–12 provide one way to assess this question. The rows of these tables correspond to the clusters that were found automatically. The automatic running indices shown in Figures 17 and 21 are used. The index and the size of each automatic cluster is given as the row title. The columns correspond to the manual categorization by the author (given on p. 111). Each number given in the tables shows how many percent of the automatic cluster consists of articles in each manually generated category.

From Table 7, it can be seen that 76.6% of all the articles in the original dataset were finally categorized manually under the title "1 Applications". Similarly, 18.0% were categorized as "2 Method development", and 5.4% turned out to be unrelated after all, and thus categorized as "3 Query noise". From the first level of automatic clustering into Cluster 2 and Cluster 3, it can be seen that Cluster 3 ended up with a higher concentration of strict applications (with roughly five times more applications than method papers) than Cluster 2 (with roughly three times more applications than method papers). Interestingly, all the unrelated query noise had been concentrated in Cluster 2, constituting 15.4% of its contents. Cluster 3 contained only articles that were deemed relevant by the manual inspection.

Table 8 gives a similar breakdown of the second level of recursion in the automatic clustering. Here, Cluster 4 and Cluster 8 are examples of duplicate entries, which were revealed early on. All the rest of the query noise ended up concentrated in Clusters 6 and 7. Some larger clusters, especially Clusters 7 and 11 already consisted quite purely of articles that were manually tagged as applications. Cluster 9, in turn, has more articles that were manually found to be more related to method development than specific applications. A feeling of this kind of structure emerged already during the first pass of the manual walkthrough, as the dictionary ordering of the titles had two distinguishable concentrations of method papers among the applied works that dominate the whole set. The table confirms this observation.

Table 9 shows which of the more fine-grained manually found categories constitute the whole dataset (Cluster 1) and the first level of automatic clustering (Clusters 2 and 3). The vast majority of the papers dealt with applications in "1.2 Engineering, manufacturing, logistics". The second largest category that emerged was labeled "1.3 Environmental management, ecology, and remote sens-

TABLE 8 Comparison of automatic and manual clustering: the second level of recursion against the first manual pass.

	1	2	3
c 4 (sz= 2):	50.0	0.0	50.0
c 5 (sz= 87):	60.9	39.1	0.0
c 6 (sz=109):	55.0	14.7	30.3
c 7 (sz= 40):	90.0	5.0	5.0
c 8 (sz= 2):	50.0	0.0	50.0
c 9 (sz= 64):	40.6	59.4	0.0
c 10 (sz=241):	86.7	13.3	0.0
c 11 (sz=142):	98.6	1.4	0.0

TABLE 9 Comparison of automatic and manual clustering: the first level of recursion against the second manual pass.

	1.1	1.2	1.3	1.4	1.5	2.1	2.2	2.3	2.4	2.5
c 1 (sz=687):	11.6	39.6	16.2	3.6	5.5	6.4	3.5	5.2	0.4	2.5
c 2 (sz=240):	9.2	18.8	21.7	5.0	8.3	0.8	8.3	7.9	0.8	3.8
c 3 (sz=447):	13.0	50.8	13.2	2.9	4.0	9.4	0.9	3.8	0.2	1.8

ing". On this level, some clear differences between the automatic Cluster 2 and Cluster 3 can be observed: Slightly more than half of the articles in Cluster 3 were manually categorized as applications in the field of engineering, manufacturing, or logistics. In Cluster 2, applications in environmental and ecological issues were more prominent. It can be seen also from this table that, overall, Cluster 2 contains more of the articles considering pure method development (i.e., manual categories labeled "2.x"). Yet, Cluster 3 has a lot of articles from the manually labeled category "2.1 MLP, SLFN, RBFNN, or ANN learning".

Again, Table 10 shows a similar breakdown of the second level of automatic recursive clustering. Clusters 4 and 8 are the very small clusters indicating duplicate entries. One of the duplicates was labeled "3 Query noise", so only 50% remain in the finer-grained categories shown here. Cluster 7 is most "purely" (70%) about applications in environmental management and ecology. Cluster 11 is also quite "purely" (62%) about applications in engineering, manufacturing, and logistics. Applications in medicine and biochemistry are also prominent in Cluster 11. A similar pattern can be seen in Cluster 10, but with more appearances of other manually tagged categories within the cluster. Exactly half of Cluster 9 consists of method development involving mostly RBF and MLP networks, whereas method development involving SVMs has been concentrated in Cluster 5. Further similar observations could be made from the table.

Perhaps of most interest is Table 11 that summarizes the constituent manual categories of the automatic clusters selected earlier for closer examination, based only on the visualized cluster hierarchy and cluster sizes. Here, Clusters 27 and 31 have a very great concentration (more than 80%) of articles manually catego-

TABLE 10 Comparison of automatic and manual clustering: the second level of recursion against the second manual pass.

	1.1	1.2	1.3	1.4	1.5	2.1	2.2	2.3	2.4	2.5
c 4 (sz= 2):	0.0	0.0	50.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
c 5 (sz= 87):	18.4	8.0	8.0	10.3	16.1	1.1	23.0	4.6	1.1	9.2
c 6 (sz=109):	2.8	33.0	14.7	2.8	1.8	0.9	0.0	12.8	0.9	0.0
c 7 (sz= 40):	7.5	5.0	70.0	0.0	7.5	0.0	0.0	2.5	0.0	2.5
c 8 (sz= 2):	0.0	0.0	0.0	0.0	50.0	0.0	0.0	0.0	0.0	0.0
c 9 (sz= 64):	17.2	9.4	6.2	4.7	3.1	50.0	1.6	4.7	0.0	3.1
c 10 (sz=241):	6.2	55.2	15.8	4.1	5.4	3.7	1.2	5.4	0.4	2.5
c 11 (sz=142):	22.5	62.0	12.0	0.0	2.1	0.7	0.0	0.7	0.0	0.0

TABLE 11 Comparison of automatic and manual clustering: hand-picked clusters against the second manual pass.

	1.1	1.2	1.3	1.4	1.5	2.1	2.2	2.3	2.4	2.5
c 13 (sz= 77):	20.8	9.1	9.1	9.1	13.0	1.3	23.4	5.2	1.3	7.8
c 6 (sz=109):	2.8	33.0	14.7	2.8	1.8	0.9	0.0	12.8	0.9	0.0
c 21 (sz= 32):	9.4	6.2	62.5	0.0	9.4	0.0	0.0	3.1	0.0	3.1
c 23 (sz= 55):	20.0	9.1	7.3	5.5	3.6	43.6	1.8	5.5	0.0	3.6
c117 (sz= 94):	9.6	46.8	9.6	4.3	5.3	7.4	3.2	7.4	0.0	6.4
c120 (sz= 59):	1.7	39.0	40.7	1.7	10.2	0.0	0.0	5.1	1.7	0.0
c 27 (sz= 76):	3.9	81.6	2.6	2.6	2.6	2.6	0.0	3.9	0.0	0.0
c 29 (sz= 68):	35.3	42.6	20.6	0.0	1.5	0.0	0.0	0.0	0.0	0.0
c 31 (sz= 63):	4.8	84.1	4.8	0.0	3.2	1.6	0.0	1.6	0.0	0.0

rized in "1.2 Engineering, manufacturing, logistics" applications. Cluster 21 has a strong concentration (62.5%) of articles on environmental and ecological applications. Of the other clusters, no such high concentrations of a single manual category can be observed. Yet, it is apparent that Cluster 29 contains only articles that have been manually tagged as applications (category label "1.x"), and also Cluster 120 is quite purely about applications. On the other hand, Cluster 13 seems to consist of very many manually tagged categories.

As an afterthought, the selection of automatic clusters to study could have been made from smaller clusters, further down the recursion tree. It is possible that the currently selected clusters could have been broken down to ones that represent more purely self-similar topics. It is an issue of further research to consider, for example, what the "optimal size" of an automatically generated clustering to examine would be. Already, the constituent percentages given above seem promising in that the further recursive clustering may have properly divided the clusters into semantically meaningful smaller clusters. Subjective evidence to this direction was experienced while reading the titles in the order given by the complete recursive clustering.

As a last comparison of the automatic clustering and the manual catego-

TABLE 12 Comparison of automatic and manual clustering: concentration of query noise.

	1	2	3
c 17 (sz= 26):	53.8	3.8	42.3
c 18 (sz= 56):	60.7	26.8	12.5
c 19 (sz= 18):	16.7	0.0	83.3
c 60 (sz= 3):	0.0	0.0	100.0
c 61 (sz= 5):	20.0	20.0	60.0
c 62 (sz= 3):	66.7	0.0	33.3
c 63 (sz= 2):	0.0	0.0	100.0

rization, let us take a closer look at the articles that were eventually identified as query noise. Table 12 lists just a few of the automatic clusters that contained a lot of articles that were considered noise. More clusters similar to these were found than appear here for brevity. Clusters 17, 18, and 19 are all derived from Cluster 6 that contained practically all of the query noise (see Table 8). Cluster 19 is mostly only noise (83.3%). Clusters 60, 61, 62, and 63 are recursively clustered from Cluster 19, containing much of the noise. Examination of these clusters shows that the abstracts had citations to Annals of Statistics, Annals of Regional Science, Annals of Mathematics, and similar, abbreviated in the text as "Ann. Stat.", "Ann. Reg. Sci.", "Math. Ann.", and similar. These articles naturally resulted from the inclusion of "ANN" in the original query string. Fortunately, these articles were clustered neatly in consecutively indexed clusters. In the dictionary order based on cluster indices, they all turned out one following the other, so after initially understanding the plot, it was very convenient and fast to tag these as query noise while performing the manual overview.

4.4 Discussion

In this chapter, a coarse mapping study of journal articles published in the crossroads of ANNs and multiobjective optimization was performed both using an automatic literature clustering tool co-developed by the author and manually, partly aided by the automatic tool. One objective of the study was to ascertain that the review of related work regarding this thesis, presented in Chapter 3, had no gaps. The part of the review dealing with the latest decade of related research was indeed based on articles found among the ones found using the original query which was designed to be as exhaustive as possible. Secondly, the automatic literature clustering tool was developed further and its usefulness as a tool to aid a manual literature review or a scoping study process was explored.

As for the outcome, the author first ascertained that the literature review made in this dissertation contained no significant gaps. Secondly, the tool itself seems to be a promising aid for manual mapping studies or literature reviews in the future. The strongest merits of the tool turned out to be the easy and early

identification of query noise, i.e., articles that were unpurposely obtained using the original query string, and the increased easiness and comfort of manual categorization of the rest of the literature simply via re-ordering of the titles and abstracts according to the automatically created clusters.

The agreement of manual and automatic clustering was explored to an extent: Some automatically found clusters showed quite high concentrations of articles categorized manually in a certain way, which was actually a small surprise to the author. It must be noticed that the automatic clustering, which is currently based only on word co-occurrences in the titles and abstacts of articles, is necessarily a very crude approximation of any real semantics of the scientific literature. Several ideas for future development of the tool arose during the case study performed here. It seems like a promising research avenue by itself to improve the literature clustering method and use it to improve the throughput and coverage of future mapping studies and literature reviews.

5 MULTIOBJECTIVE TRAINING OF MLP CLASSIFIERS

This chapter recapitulates the features necessary in a state-of-the-art implementation of a Pareto-based multiobjective MLP classifier training method and describes one step in an on-going incremental method development in the continuum that involves the earlier works of Nieminen and Kärkkäinen (2009, 2010, 2016) and Nieminen et al. (2011). The current step is far from final, but the author asserts that the groundwork presented in Chapter 3 of this dissertation thoroughly clarifies the road ahead for the next steps. More elaborate publications about the developments are planned for the future, along with more experiments to prove or disprove the hypothesized merits anticipated in this chapter.

Section 5.1 reviews the general goals of the implementation. Section 5.2 details the specific objective functions chosen to be explored and hypotheses to be explored with a package of experiments accompanying the implementation. Section 5.3 details the planned algorithmic framework and the current state of its implementation at the time of writing this dissertation, and it begins with a few passages about the repeatability of computational experiments and how this dissertation attempts to meet the basic requirements of such.

5.1 General Goals

The summaries made earlier about the first decade (in Section 3.2.11) and the latest decade (in Section 3.3.10) of research in multiobjective supervised learning lead to the following digested list of features required of a current implementation of a Pareto-based MLP classifier training method:

- A *memetic algorithm* consisting of a multiobjective evolutionary framework and *memetic operators* customized specifically for the MLP are to be used.
- An *object oriented phenotypic representation* of the MLP should be used.
- A *full archive of non-dominated solutions* should be kept throughout the itera-

tion.
- The possibilities of *overfitting or pre-mature convergence* to local optima during local improvement should be managed.
- The selection of operators and the settings of their parameters should be *adapted* or *evolved* throughout the iteration.
- A *crossover operator* of MLP classifiers based on the geometric interpretation of the hidden neurons forming separating hyperplanes in their input space should be available.
- Instead of using MOO *implicitly* as an intermediate helper in single-objective optimization, the Pareto-based approach calls for *explicit separation of objectives* and involvement of the user to make an informed human decision of the final model.
- *Ways to gain insight* into the properties of the dataset via Pareto-based MOO should be promoted.
- Especially, *the interactions of similar but subtly different objectives*, for example, various complexity measures, is to be jointly explored for datasets with different distributions of classes and other properties, such as noisy observations.
- Combinations of *more than two objective functions* should be aimed at, and useful applications of such combinations charted.

Furthermore, the additional main target of improving upon an existing MLP formulation, and building upon earlier studies using it, leads to the following goals specific to the work presented here:

- The MLP architecture is restricted to a fully connected feed-forward structure which can be represented as a set of non-sparse layer-wise weight matrices. Such a structure can be implemented in a concise and extremely efficient way in practically any software or hardware platform in which a finalized classifier needs to function. An efficient implementation in ANSI C has been made earlier, and usefully deployed in industrial field applications.
- Differentiable activation functions are used, and thus gradient-based training methods are applied where possible.
- A special emphasis is put on the investigation of the specific formulations that have been used before by the author and others in his research group in earlier studies.
- Thus, an incremental mode of development, building upon the existing research and code base, is to be used.

5.2 Multiple Objective Functions for MLP Classifiers

Next, the specific family of objective functions of interest here is defined. Most of the functions appear first as a general, parameterized, form. By varying the

parameters, a generally infinite number of different objective functions could be generated. Earlier experiments and the literature review made for this dissertation have lead to certain hypotheses of how each function should behave differently for datasets with some different properties of special interest. The following descriptions are organized into subsections based on these hypotheses of applicability.

5.2.1 Notations and General Properties of the Objectives

In the following descriptions, the MLP model of Equation (5) is used. To recapitulate, the symbol L stands for the number of layers, including both hidden layers and the output layer. The number of neurons on layer $l \in \{0, \ldots, L\}$ is written as n_l. The 0:th layer corresponds to the input layer. The symbol $\mathcal{N}(\mathbf{x}_i)$ stands for the output value after evaluating a network for an input vector $\mathbf{x}_i \in \mathbb{R}^{n_0}$. The weight matrices $\{\mathbf{W}_l\}_{l=1}^{L}$ used for computation are assumed to be known from the context in which $\mathcal{N}$ is used. Similarly, the symbol $\mathbf{e}_i$ is shorthand for the error vector $\mathbf{e}_i = \mathcal{N}(\mathbf{x}_i) - \mathbf{t}_i$ where $\mathbf{t}_i$ is the desired target vector corresponding to the training data input $\mathbf{x}_i$ in a completely labeled dataset. In all cases involving classification, $\mathbf{t}_i$ is taken to be a "1-of-K" binary encoding of the desired integral class label t_i, i.e., a vector $\mathbf{t}_i \in \mathbb{R}^K$ where the t_i:th component is 1 and others are either 0 (when the logistic sigmoid is used as the activation function) or -1 (when the hyperbolic tangent is used as the activation function). For classification, the predicted integral class label corresponding to the vector-valued output $\mathcal{N}(\mathbf{x}_i)$ is denoted $\mathcal{P}(\mathbf{x}_i) \in \{1, \ldots, K\}$ and is taken to be the "winner-take-all" interpretation where the index of the maximal element of $\mathcal{N}(\mathbf{x}_i) \in \mathbb{R}^K$ is selected. Any (unlikely) ties are resolved to the smallest index. Formally, $\mathcal{P}(\mathbf{x}_i) = \min\{j : (\mathcal{N}(\mathbf{x}_i))_j \geq (\mathcal{N}(\mathbf{x}_i))_k \text{ for } j, k = 1, \ldots, K\}$. The individual synaptic weights of the MLP model are written as $w_{i,j}^l$ where $l \in \{1, \ldots, L\}$ is the index of the layer, $i \in \{1, \ldots, n_l\}$ is the index of the neuron within the l:th layer, and $j \in \{0, \ldots, n_{l-1}\}$ is an index into the output vector $\mathbf{o}_{l-1}$ of the previous layer or to the value 1 for index $j = 0$ that refers to a bias value. The symbol I is used for index sets that select a subset of the MLP network structure, and the symbol S for index sets that refer to a subset of the labeled data. The notation $\#S$ denotes the number of elements in a set S.

Each objective function shall be given a shorthand symbol f_{mnemonic} including a subscripted mnemonic of its intended purpose. Each function is formulated as one to be *minimized*, which simplifies the algorithms and especially the examination of Pareto fronts. The functions are also always non-negative. Thus, in all plots, the (generally unobtainable) utopian solution minimizing all of the functions is always found at the origin. This is why, for example, the classification error rate (as the ratio of wrong predictions) is measured instead of the classification accuracy (as the ratio of correct predictions).

5.2.2 Continuous Error Measures with Class Noise Mitigation

General formulation: A parameterized family of continuous error measures (CEM) of an MLP classifier (or function approximator, for that matter) can be derived from the following general objective function, which is presented here after the notation used by Kärkkäinen and Heikkola (2004):

$$f_{\mathrm{CEM}}(\alpha, q, S)(\{\mathbf{W}_l\}_{l=1}^{L}) = \frac{1}{\alpha \# S} \sum_{i \in S} \|\mathbf{e}_i\|_q^{\alpha}. \tag{11}$$

Variations: The index subset S can be used to focus the objective function on a subset of the training data. The selection of S will be covered in a later subsection. For now, let us choose S so that it contains the indices of all the data points available in the training dataset. While an infinite number of different variations of this objective function can be obtained by varying its parameters, a few combinations of α and q provide cases of special practical interest. The parameter q selects a norm that is used in computing the contribution of the error made by the MLP approximation for each individual data point. Common choices would include the usual Euclidean error norm with $q = 2$, city-block error distance with $q = 1$, or maximum error with $q = \infty$, although any other selection of q could be made. The divisor/exponent parameter α provides further variations. Here, common choices could be $\alpha = q = 2$ for the most traditional mean of squared elementwise errors (MSE), $\alpha = q = 1$ for the mean absolute errors (MAE), and $\alpha = 1$ with $q = 2$ for the mean Euclidean error (MEE). It is worth an extra notice that for vector-valued outputs, like the outputs of MLPs for multiclass classification, the MAE and MEE are very different measures. As a divisor, α simplifies the computation of derivatives, but naturally it also affects the computation of the mean by a factor of $\frac{1}{\alpha}$.

Hypotheses on applicability: The formulation has been explored earlier, by Kärkkäinen and Heikkola (2004) who provided non-smooth subdifferentials for the above selections of q and α, and showed a connection between robust statistics and MLP training. When a linear output layer with biases is used, any MLP that locally optimizes $f_{\mathrm{CEM}}(2, 2, S)$ has an average error of zero over the selected dataset. More importantly, the zero vector is the *median* of the error vectors of any MLP that locally optimizes $f_{\mathrm{CEM}}(1, 1, S)$, and the *spatial median* of the errors of any MLP that locally optimizes $f_{\mathrm{CEM}}(1, 2, S)$. Kärkkäinen and Heikkola (2004) showed experimentally that the selections of the robust error functions $f_{\mathrm{CEM}}(1, 1, S)$ and $f_{\mathrm{CEM}}(1, 2, S)$ protect against outliers in function approximation tasks. As for classification tasks, preliminary supportive evidence was gained by Nieminen and Kärkkäinen (2010) that the selection of $f_{\mathrm{CEM}}(1, 2, S)$ would maintain a high classification performance in the presence of heavy class noise, i.e., a high percentage of the training data being mislabeled by accident. For clean data, there should not be a notable difference between MLPs minimizing either $f_{\mathrm{CEM}}(1, 2, S)$ or $f_{\mathrm{CEM}}(2, 2, S)$, the latter of which is the most commonly used mean

of squared errors (MSE) objective. It is hypothesized that the presence of class noise could indeed be discovered by observing a spread-out distribution of solutions to $f_{\text{CEM}}(1,2,S)$ and $f_{\text{CEM}}(2,2,S)$ in a Pareto front, which could inform the user to choose a model that minimizes $f_{\text{CEM}}(1,2,S)$ rather than a model that minimizes $f_{\text{CEM}}(2,2,S)$. In a clean situation without major class noise, the two measures should give similar results. Respectively, an observed accordance of the two functions in a Pareto front could serve as evidence that the training dataset is likely devoid of any corrupted labels.

5.2.3 Continuous Model Complexity Measures

General formulation: Using the absolute values of a subset of the synaptic weights, given as an index set I, a family of continuous model complexity (CMC) measures can be obtained from the following general form:

$$f_{\text{CMC}}(\gamma, I)(\{\mathbf{W}_l\}_{l=1}^{L}) = \frac{1}{\gamma \# I} \sum_{(i,j,l)\in I} |w_{i,j}^{l}|^{\gamma}. \tag{12}$$

Variations: While any selection of parameters are possible, a common one is the mean of squared weights, or Gaussian regularizer, $f_{\text{CMC}}(2, I^*)$, where the index set I^* covers all of the MLP weights. Similarly, the Laplacian regularizer is obtained as $f_{\text{CMC}}(1, I^*)$. The index set may be reduced in various ways. For example, only the weights on one layer of the MLP could be measured by a variant of I in this objective function. As another example, the output bias weights could be excluded as suggested by Kärkkäinen (2002).

Hypotheses on applicability: At least one of these regularization function variants is traditionally chosen to be aggregated into the single objective function used for training an MLP. For gradient descent methods, they contribute a "weight decay" effect, i.e., the tendency to reduce the absolute values of the weights during the iteration. The Gaussian and the Laplacian regularizer are known to affect the iteration process in different ways. Also, it is known that using different penalties for the weights of the different layers of MLP could be beneficial. From the single-objective optimization point of view, each aggregated regularization term could be viewed as a constraint function with an aggregation weight coefficient set *a priori* or via experimentation. In a Pareto-based "multiobjectivization" sense, such constraints could be considered as additional objective functions. The main hypothesis is that the values of the different function variants would show different behaviour on datasets with different properties, and increase the user's knowledge of the dataset before making the *a posteriori* decision of the optimal MLP classifier.

5.2.4 Discrete Error Measures

General formulation: In classification, the vector-valued output $\mathcal{N}(\mathbf{x}_i) \in \mathbb{R}^K$ is eventually interpreted as an integral class prediction $\mathcal{P}(\mathbf{x}_i) \in \{1,\ldots,K\}$ which

may or may not be the correct target class label t_i. The following function evaluates the (discrete) objective function of errors made over an indexed subset S of the labeled training data, which we call a discrete error measure (DEM):

$$f_{\text{DEM}}(S)(\{\mathbf{W}_l\}_{l=1}^{L}) = \frac{\#\{i \in S : \mathcal{P}(\mathbf{x}_i) \neq t_i\}}{\#S}. \tag{13}$$

Variations: Also this function can be focused on different subsets of the data by varying S. For now, let us consider the whole training dataset. Other selections of S will be considered shortly, while discussing cost sensitive and imbalanced learning and direct measuring of generalization.

Hypotheses on applicability: In the case of classification, the continuous, averaged, error measures are only a proxy of the actual capacity of an MLP to make correct predictions of each class. The ultimate requirement from the application point of view is to minimize the ratio of incorrectly made predictions. The main hypothesis is that even if the continuous error measures are used in gradient-based local improvement, the discrete measures are more interesting for the tasks of Pareto-front exploration and final solution selection.

5.2.5 Discrete Model Complexity Measures

General formulations: The following discrete functions measure the structural complexity of an MLP in various ways:

$$f_{\text{NL}}(\{\mathbf{W}_l\}_{l=1}^{L}) = L. \tag{14}$$

$$f_{\text{NN}}(I)(\{\mathbf{W}_l\}_{l=1}^{L}) = \sum_{l \in I} n_l. \tag{15}$$

$$f_{\text{NAC}}(I)(\{\mathbf{W}_l\}_{l=1}^{L}) = \#\{w_{i,j}^{l} : w_{i,j}^{l} \neq 0; (i,j,l) \in I\}. \tag{16}$$

The first one, f_{NL} stands for the number of layers. The second one, $f_{\text{NN}}(I)$ stands for the number of neurons in a part of the MLP selected using the set $I \subset \{0, \ldots, L-1\}$. The third one, $f_{\text{NAC}}(I)$, stands for the number of active connections, i.e., weights that have a non-zero value, in a part of the MLP selected using the index set I.

Variations: The number of layers is simply the structural depth of the MLP, which, for the current experiments, may not be varied after initial creation of an MLP. Also, the objective function is parameterless and cannot be varied. Variable-depth ANNs would be a topic of interest in follow-up studies, so the measure is included here for completeness. The number of neurons could be computed over the whole MLP, which may have many hidden layers, or focused on only one of the hidden layers by varying the index set I. The *maximum* number of neurons on the input layer, i.e., n_0, and the exact number of neurons on the output layer, i.e., n_L, are fixed by the dataset properties. Also the number of active connections

can be computed over the whole network, or only one or few layers. Of interest may also be the restriction to the bias weights of a layer instead of its inputs from the previous layer.

Hypotheses on applicability: Pareto-based multiobjective optimization allows several fine-grained complexity measures to be defined and simultaneously optimized. Also with regard to these measures, it is hypothesized that observed properties of the resulting Pareto front may shed light on otherwise reluctant properties of the dataset, and the *a posteriori* MLP model selection can be based on the measure that, after exploring the result set, turns out to be the most practical one.

5.2.6 Cost Sensitive Learning and Skewed Class Frequencies

General formulations: Consider the formulations of $f_{\mathrm{CEM}}(\alpha, q, S)$ and $f_{\mathrm{DEM}}(S)$ given above.

Variations: Consider splitting the indices of the dataset into subsets $S_1, \ldots, S_K$ so that each subset covers exactly the training data points labeled in one of the K classes. Then the resulting K objective functions $f_{\mathrm{CEM}}(\alpha, q, S_c)$ and $f_{\mathrm{DEM}}(S_c)$ for $c = 1, \ldots, K$ measure class-wise errors. The minimum class-wise error is obtained as $\min\{f_{\mathrm{DEM}}(S_c)\}_{c=1}^{K}$.

Hypotheses on applicability: The main hypothesis regarding these class-wise objective functions is that they give information about the necessary trade-offs in final MLP classifier selection that is more useful than that obtained by looking at the whole data at once. Experiencing a Pareto front that is spread out with regard to some of these class-wise objectives should reveal that the distributions of the classes overlap each other in the training data. The most important applications are expected to be those of cost sensitive learning, i.e., when different costs apply to different misclassifications, and of imbalanced learning, i.e., when the prior class frequencies differ from each other.

5.2.7 Feature Selection

General formulation: Consider the general formulation of $f_{\mathrm{NN}}(I)$ given above.

Variations: Consider selecting $I = \{0\}$, i.e., only the input layer.

Hypotheses on applicability: The main hypothesis is that feature extraction could be incorporated in the main optimization iteration, and that exploration of the Pareto front and resulting MLP structures would reveal information about the relative number and the identity of the most relevant input features.

5.2.8 Direct Measures of Generalization

General formulation: Consider, again, the general formulations $f_{\text{CEM}}(\alpha, q, S)$ and $f_{\text{DEM}}(S)$ given above.

Variations: Consider splitting the indices of the dataset via stratified sampling into either two subsets S^{tr} and S^{val}, where *tr* stands for training and *val* for validation, or even more subsets $\{S^i\}_{i=1}^{F}$ which could be called "folds".

Hypotheses on applicability: The classical splitting of the available labeled data into training and validation subsets, or a number of cross-validation folds, can be emulated to an extent as two or more objective functions that are evaluated during the main optimization iteration. Examples of the two-way split were cited in the literature review earlier in this dissertation. Augmentation of the idea to more than two sets looks like an interesting and novel approach, although there is no clear hypothesis about its potential outcomes before experimentation.

5.2.9 Combinations of Different Objective Functions

General formulations: All of the above.

Variations: Selected subsets of the above. There are no theoretical limits to the number of objective functions to combine or the way in which the combinations are taken. For example, it is completely possible to combine the above ideas to select class-wise stratified training and validation sets $\{S_i^{tr}\}_{i=1}^{K}$ and $\{S_i^{val}\}_{i=1}^{K}$ and produce robust, non-robust, and discrete error measures for each subset and further combine those with several continuous and discrete complexity measures. In practice, though, such many-objective optimization is known to be difficult both for the currently popular MOO algorithms and for the human user (see, for example Chand and Wagner 2015; Deb and Jain 2014). Practical use is thus likely to require some prior decisions about which objective functions could be the most interesting ones for the task at hand.

Hypotheses on applicability: The main hypothesis is that knowledge of the classification problem at hand, and its "inherent difficulty", can be gained via exploring the Pareto front in which a proper combination of the above functions has been jointly minimized. One of the most traditional examples is the trade-off between the complexity and the error of the MLP. Another one is the "ROC front" where the accuracies (or, symmetrically, errors) of two classes is jointly examined. It is anticipated also that the examination of more than one complexity measure, such as only one regularization term selected *a priori*, could reveal some intrinsics of the datasets that would be hard to spot using only one measure. All in all, Pareto-based MLP training should aim at results that simply cannot be obtained using single-objective training methods. For simple datasets, there is no need to use anything else but the standard backpropagation. For less trivial cases, the

foremost promise of Pareto-based training lies in extracting interesting information about the dataset in addition to obtaining, as the "necessary by-product", if such a viewpoint is taken, a set of Pareto-optimal MLPs among which a final one can eventually be chosen.

5.3 Algorithms and Implementation

This section describes the platform being produced in the development work continuously progressing as of writing this dissertation.

5.3.1 Reproducing Research from Source Code

Inspired long ago by Buckheit and Donoho (1995), the author of this dissertation advocates the principle that research results in computational science should always be accompanied by the source codes that can be turned into executables that reproduce any result obtained. Sometimes data confidentiality or intellectual property right issues make this hard to perform, as is unfortunately the case also for the literature mapping case study of Chapter 4 of this dissertation. For all the other cases and illustrations presented here, the source code to reproduce them is made available in the public source code repository "momulper" created for this thesis[1]. As of writing this dissertation, many of the planned components are still to be implemented.

5.3.2 Overall Memetic Framework

As of its current form, the compilable and executable scaffold includes an instantiation of the generic Algorithm 2 with the following components:

- An evolutionary optimization framework implemented in C++11 and using the reference C version of NSGA-II as the selection operation.
- A computational engine for MLP networks based on an earlier implementation used in the research group of the author, re-implemented in C++11. The engine supports efficient computation of the continuous error measures (CEM) defined above, with gradients (and subgradients for nondifferentiable CEMs).
- A Matlab counterpart of the MLP implementation that can be used for post-run analyses and visualizations of the MLPs and populations used in the algorithms.
- Basic structures for creating and inserting unary and n-ary memetic operators for population and individual updates within the algorithm. The algorithmic components are selected for every run using case study definition files in a specific, human-editable grammar.

[1] http://users.jyu.fi/~nieminen/research/dissertation2016/codes/

- An incrementally growing set of experiments for testing and evaluating new features and combinations of objective functions, as they are added in the implementation.
- Generation of simulated datasets in N dimensions, implemented in Matlab code that is compatible with and runnable using the open source computational software Octave.
- Storage formats for MLP populations that is easy to parse with Matlab.
- A "trace history" archival mechanism for the optional storing of each new population individual with not only the objective functions used for optimization but also any additional measures defined in the case study and the identities of its parents. The purpose of this tracing functionality is to be able to trace the progression of any individual in the memetic framework to its (grand)parents in previous populations all the way to the initialization phase. For faster computation in an actual application case, the trace would be disabled, but for method development, it stores crucial data for assessing the performance of the memetic operators.
- Initial visualizations for examining the progression and result of the runs of the main algorithm, implemented in Matlab.

5.3.3 Genetic Representation

Currently, the chromosome representation is internally an array of floating point values that contain the packed row vectors of the weight matrices of an MLP. This allows for the traditional fast computations and inner-loop optimizations for the forward and backward signal propagations.

The chromosomes are variable-size so that any fully-connected feed-forward MLP topology can be used as an individual. The MLP individuals are wrapped inside an object-oriented class interface in C++ so that the internal representation needs not to be known by the overall algorithm that works through objective evaluations and the commencing of alteration operations to be performed. The operators are offered MLP-specific access functionality such as gradient computation and weight updates based on a labeled dataset, and access that appears as the layer-wise matrix representation as in Equation (4).

5.3.4 Evolutionary and Improvement Operators

As of writing this dissertation, the platform is ready for inclusion and trials of different MLP-specific unary and n-ary operators that modify the individuals. Implementations and tests with these will be included in the source code repository in an incremental manner according to the following plan:

- At first, include simple operators well-tried already in the early days of multiobjective MLP training (see 3.2.11).
- Proceed to include further operators up to the currently known ones (see 3.3.10).

- Try out new experimental versions of n-ary recombination operators that first imitate and then, perhaps, improve on the earlier ones based on geometric interpretation of hidden neuron hyperplanes (Goh et al. 2008; Teoh et al. 2006).
- Attempt to extend the recombination to MLPs with more than a single hidden layer.

The work is underway as of writing this dissertation, and can be monitored at the source code site.

6 PRELIMINARY COMPUTATIONAL EXPERIMENTS

The task undertaken here by the author was to incrementally develop an existing MLP formulation and implementation to meet, and, with luck, exceed, the expectations put on a current Pareto-based multiobjective classifier training method. The fundamental groundwork was laid out in Chapter 3 and the development plan in Chapter 5. As of writing this last chapter of the dissertation body, the work is still on-going, and the final outcome is yet to be witnessed from results generated by the source code being developed. This chapter provides the experimental plan and a few first examples that can be reproduced by the public experimental codes underlying the endeavor. An additional grandiose goal was to implement also a complete experimental testbed that would automatically generate an array of convincing computational results with visualizations. While the actual convincing results are still pending, the experimental testbed is already functioning. It produces the datasets and MLP illustrations seen so far in this thesis and is already capable of much more. More results are already available by running the experiments using the provided source codes than those that are represented here as examples, and yet more are anticipated to appear for future publications. Adapting the words of Buckheit and Donoho (1995), this chapter is merely an "advertisement of the scholarship" that this dissertation is the most recent part of.

Section 6.1 samples a couple of simple test cases that were used in the initial bootstrapping phase for verifying that a very early mock-up version of the algorithm and its connection with the result visualization was working. They continue to function as sanity checks for the MLP computations and any new local improvement operators to be added. The sampled experiments also automatically produce some of the first illustrations of the dissertation. The section also introduces the format used in the automatic experiment reports and visualizations currently produced. Section 6.2 illustrates just a couple of further test cases for the functioning of the platform and some recently added features. They were used for producing the first examples of nontrivial classification tasks in this dissertation. The implementation is not yet finalized as of writing this dissertation, so Section 6.3 must end this chapter and the whole dissertation by anticipations

of how the development is to continue in later works.

6.1 Sanity Checks and Introduction to Result Notation

The first part of the experiments is performed using simulated two-dimensional datasets where the data complexity can be fully controlled, and the action of the resulting MLPs completely visualized. An overview of an experiment and some of its parameters can be output from the platform as text that can be included in a LaTeX document in the following format that can be cross-referenced no differently from equations, figures, or sections of text:

Experiment 1:

Brief	Linearly separable data; first illustration in the thesis Introduction.
Description	Emulated standard single-objective backpropagation.
Task	classification
Dataset	2dlinsep.data
Objectives	mse
Population size	1
Evol. framework	none
MLP init arch.	2-1-2
Improvement opers	backprop

Experiment 1 reduces the platform into the most traditional single-objective MLP training method by selecting only one objective, one member as the population size, and only the MSE error to be minimized. The only operation selected is the backprop step, and evolution is not used (the selected evolutionary framework is "none"). In fact, in the first bootstrapping phase, this was everything that the platform was capable of doing, which was quite sufficient for making sure that the freshly re-implemented MLP computation code gives sensible error and gradient values for the MSE error. The case definition for this very first case study is likely to remain in the bank of experiments for the rest of the lifetime of the research code as a test case that will forever make sure that the MSE computation is not accidentally broken by changes in the MLP codes. Not all parameters are shown in the purposely simplified outline, but this is no problem because every detail is preserved in the original problem description file.

With the definitions of Experiment 1, the algorithm runs as an emulated single-objective backprop training method that produces an output population of one MLP. The postprocessing part of the platform implemented in Matlab can be used to read the final population created by the C++11 part and produce output

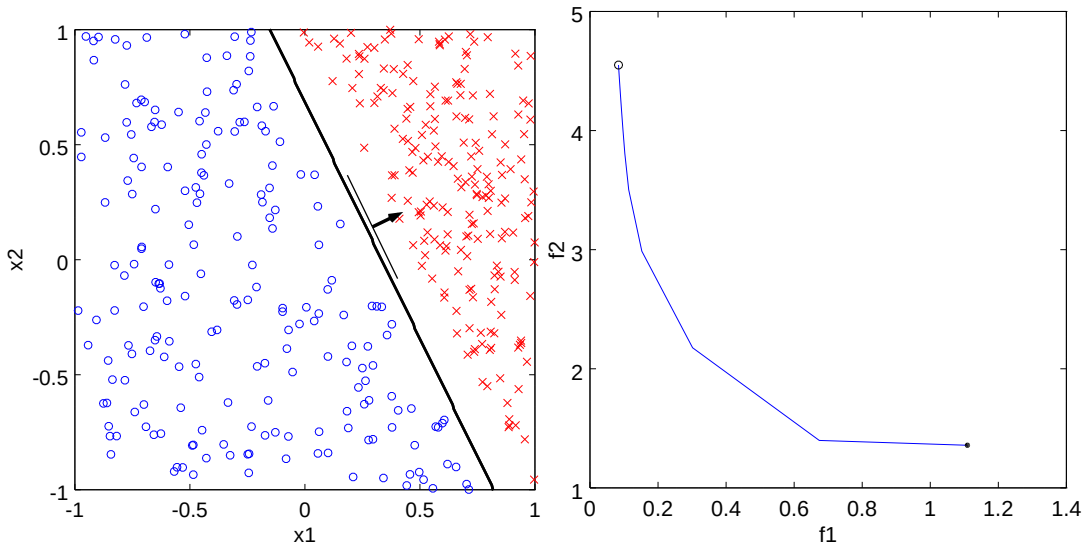


FIGURE 23 Linearly separable binary classification, once again, now with a traced complexity measure.

visualizations. The left pane of Figure 23 is an exact replica of Figure 5, which was in fact produced by running and visualizing Experiment 1. The right pane illustrates one of the already implemented features of the experiment platform, which is the tracing of objective functions or measures that did not take part in the optimization process. In this case, the trajectory of the solutions from the initial random MLP to the final trained one is plotted using the MSE error and the mean of squared weights of the MLP as the coordinates. With such illustrations, the interplay of two functions during single-objective learning can be explored. As already explained in Chapter 2, the left pane shows the action of the hidden layer of a single-hidden-layer MLP using the arrow width to represent the weight magnitudes. The right pane can be used to trace the history of the magnitudes throughout the learning process.

To prove the point that the testbed makes result exploration convenient, let us go through another simple experiment:

Experiment 2:

Brief	Non-linear but separable data; 2nd thesis illustration.
Description	Emulated single-objective multistart backpropagation. This shape requires more than two neurons. Initial guesses are crucial, but redundant neurons can help. Backprop can do this quite easily with 6 hidden neurons, although less would be optimal. Can do with 5, too, even without regularization. 4 is difficult.
Task	classification

Dataset	2dw.data
Objectives	mse
Population size	12
Evol. framework	none
MLP init arch.	2-5-2
Improvement opers	backprop

The result of Experiment 2 is illustrated in Figure 24, the left pane of which matches the earlier Figure 9. In this case, the platform is emulating a multistart backprop with 12 random initial MLPs. For the illustration, the best individual (with smallest MSE) is selected. Again, the right pane shows the usual steep increase in weight values when the error is finally decreasing in only very small steps. From the left pane, it can be readily observed that the hyperplanes created by the single hidden neuron may be pointing in either of the two possible directions (see the one on the right compared to the others). This mirror symmetry is compensated by the action of the next layer. The visualization of the hidden layer action is designed with the purpose of examining the function of MLP crossover operators which are on the research roadmap formed in this dissertation but not implemented as of writing the manuscript.

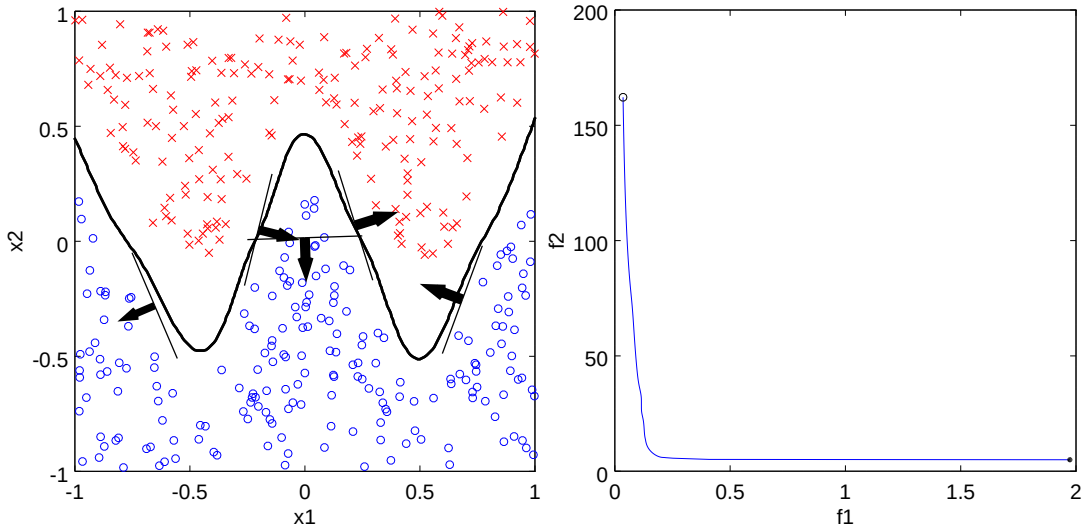


FIGURE 24 Nonlinear binary classification with a traced complexity measure.

6.2 Simple Datasets Exhibiting Non-Trivial Features

To prove the point that the platform is already capable also of actual multiobjective optimization, let us show yet one more simple example case:

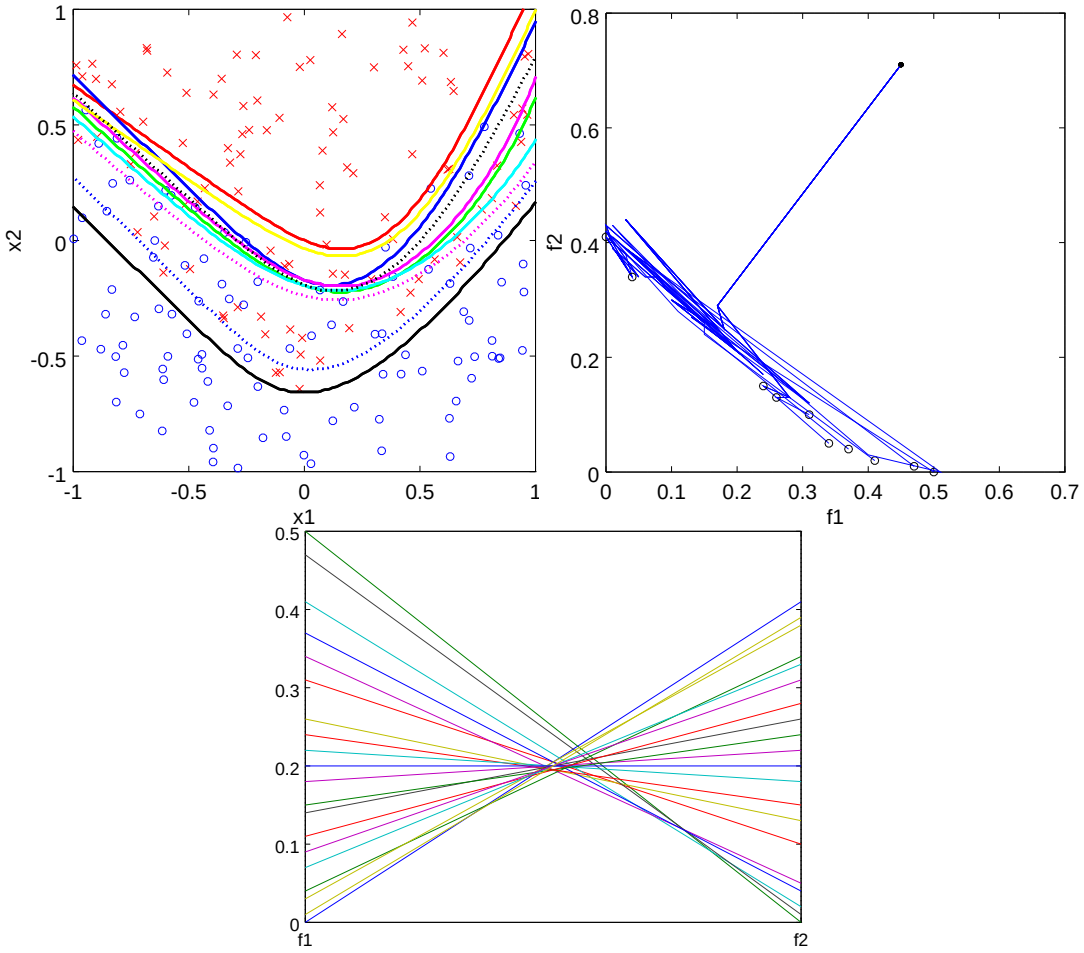


FIGURE 25 Multiobjective binary classification with a trace of the classwise errors and a parallel coordinate plot of the final population.

Experiment 3:

Brief	Non-linear data with overlapping distributions.
Description	Real two-objective multiobjective optimization for two classwise errors. Very simple example to illustrate the concept.
Task	classification
Dataset	2dvOverlap.data
Objectives	cwerr
Population size	20
Evol. framework	ns
MLP init arch.	2-2-2
Improvement opers	backprop

The results of Experiment 3 are shown in Figure 25. This was the first trial created to test (for the first time of development, and for the forthcoming future) that the nondominated sorting part of the implementation works in a sensible manner.

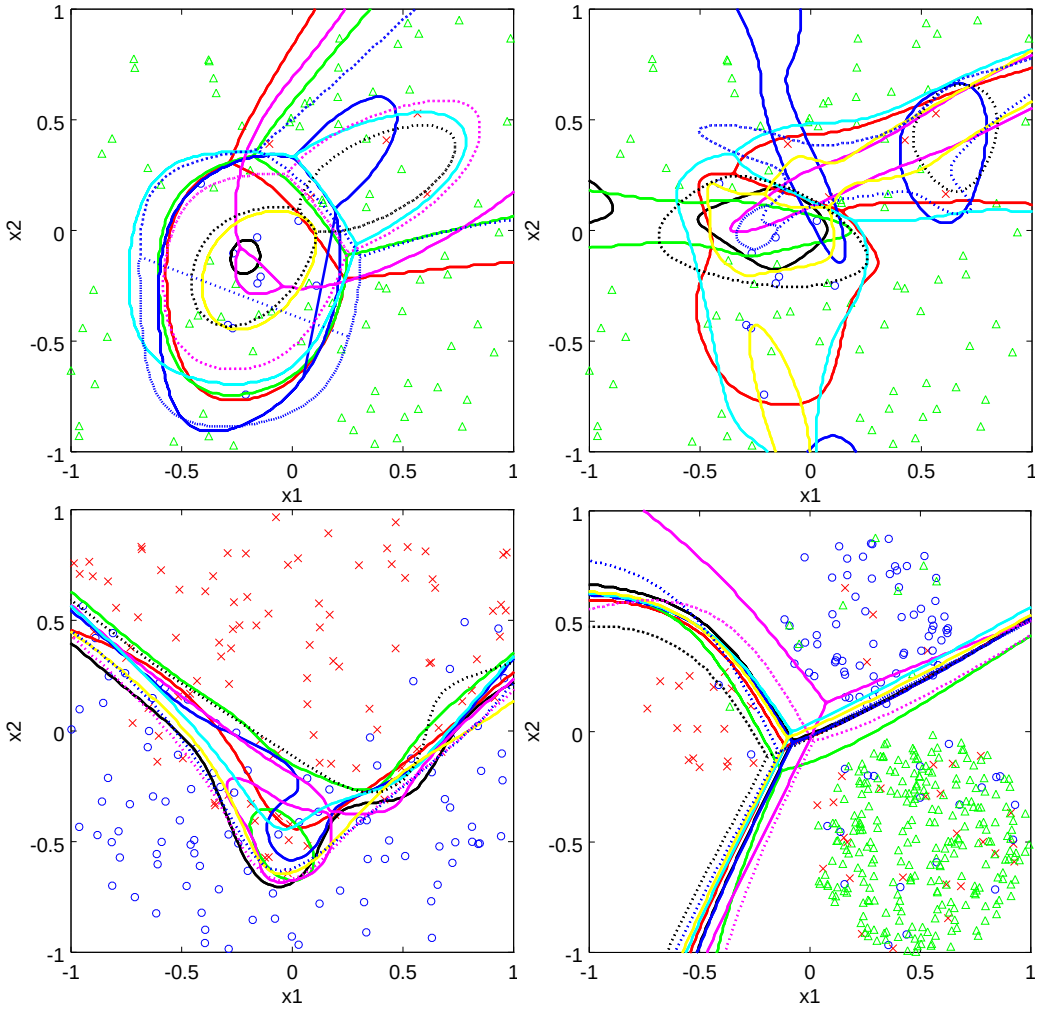


FIGURE 26 Some further examples of result illustrations from the current implementation.

In this case, a random selection of 10 individuals from the final population are selected for plotting in the top left pane. Their traces throughout the iteration using the actual classwise error rates are shown in the top right pane, and the whole population is shown in the bottom pane as a parallel coordinates.

Further examples of initial experiments using the current codes are shown in Figure 26. The two top panes show different results for a three-class imbalanced case similar to the one explained in Section 2.6. On the bottom left is an example of the different classification boundaries of a Pareto-optimal population of MLPs trained by minimizing both classifier error and complexity. On the bottom right is an initial example of minimizing both MSE and MAE errors under class noise.

6.3 Future Work

The implementation is currently in a state of development, the results of which must be output in later publications after this dissertation that provided the mission statement and the goals. The intention is to proceed with the roadmap given in Chapter 5 to first reach the state-of-the-art and then proceed to research of possible extensions of MLP-specific crossover operators. At each stage, the newly added features are to be explored with experiments that then remain as permanent test cases in the experimental platform which was shown in this chapter to work. Interesting results can then be selected as illustrations for reports of the development.

As for experiments and datasets, incrementally more dimensions should be added after the method components are found to work by examining the two-dimensional, easily visualizable cases. Naturally, when the method is proven to work on simulated datasets with known properties, the ultimate goal is to try the MOO formulations on common benchmark datasets in order to compare it with others and attempt to assess if the trade-offs found in the Pareto fronts of the benchmark datasets would exhibit traits that are known for the simulated cases to exhibit interesting properties such as class overlap or class noise.

7 SUMMARY

This dissertation documented a step in an on-going research track that the author has been involved in, on and off, for the past many years. Chapter 1 recalled his earlier research on some aspects of multilayer perceptron (MLP) neural network training and outlined limitations that could possibly be lifted by a multiobjective approach in training. The purpose was to find out if and how the concepts of multiobjective optimization could be used for training MLPs. Chapter 2 explained the basic concepts involved with brief remarks of the history of each.

Chapter 3 reviewed existing literature on the topic of multiobjective machine learning with an emphasis on MLPs and closely related learning machines. It turned out that a comprehensive survey of the most recent ten years of developments seems to be lacking, and the latter part of the chapter became an attempt to initiate one, thus forming a major part of the research documented here. Several successful applications of multiobjective methods were found, and their inner workings digested as a reference of what has been done in the field.

The literature survey process was supported by an automatic tool partly developed by the author. Chapter 4 described the process and also introduced some new methodological advances of the support tool for the first time. In a way, the chapter contributed a small study of its own, in the field of informetrics. The tool was found to be useful in increasing the comfort and possibly also throughput of charting a body of literature.

Chapter 5 synthesized the earlier research into a shortlist of features required of a current multiobjective MLP training method and proposed a roadmap for method development. The method is still under development, but it is briefly demonstrated in Chapter 6 to be already functional. The implementation aims to follow the ideal of perfectly reproducible research all the way from the scratch.

The dissertation, with its contributions mentioned above, documented a track that has a history and a future. Follow-up studies would include not only further development of the initiated MLP training method but also finalizing the survey of other contemporary alternatives. The literature survey method was also found to be on the right track, and it will surely be developed and applied in the future.

YHTEENVETO (FINNISH SUMMARY)

Monikerroksisten hermoverkkojen muodostaminen memeettisen monitavoiteoptimoinnin avulla

Koneoppiminen tarkoittaa luonnollisten oppimisprosessien jäljittelemistä keinotekoisten laitteiden tai ohjelmistojen avulla – eli esimerkiksi sellaisten tietokoneohjelmien tekemistä, jotka oppivat niin sanotusti luokittelemaan ennaltanäkemätöntä aineistoa. Perusesimerkki tällaisesta luokittelutehtävästä voisi olla vaikkapa henkilöiden tunnistaminen digitaalisten kasvokuvien perusteella. Toinen helposti ymmärrettävä sovelluskohde voisi olla jonkin sairauden tunnistaminen potilaasta mitattujen veriarvojen perusteella. Ohjatussa koneoppimisessa opetusaineisto mitataan parhaimmillaan joukosta, jonka todellinen luokittuminen tunnetaan. Tavoitteena on saada muodostettua väline, joka pystyy salamannopeasti tekemään vastaavan luokituksen, esimerkiksi lääketieteellisen esidiagnoosin, uuden havaintoaineiston perusteella. Monikerroksiset hermoverkot ovat yksi perinteinen ja paljon käytetty matemaattinen malli, joka on varsin yksinkertainen toteuttaa tietokoneella. Malli on kuitenkin aina vain sapluuna, joka täytyy muodostaa ja täsmentää keinotekoisen oppimisprosessin kautta.

Koneoppimistehtävät sisältävät yleensä useita keskenään ristiriitaisia tavoitteita. Yksi esimerkki tästä on, että yksinkertainen malli ei yleensä pysty seurailemaan opetusaineiston piirteitä tarkasti, vaikka sekä yksinkertaisuutta että tarkkuutta tavoitellaan. Toinen esimerkki on diagnostiikkasovelluksissa usein vastaan tuleva kompromissi väärien positiivisten ja väärien negatiivisten diagnoosien välillä. Tietokoneohjelmien opettamiseksi määritellään matemaattisia funktioita, joista kunkin optimointi vastaa yhtä näkökulmaa suhteessa toivottuun lopputulemaan. Yhtä ja samaakin näkökulmaa voidaan kuvata erilaisilla funktioilla, jotka voivat olla keskenään ristiriitaisia – riippuen opetuksessa käytetystä aineistosta. Koneoppimisessa on käytetty tällaisten keskenään ristiriitaisten funktioiden samanaikaiseen optimointiin kehitettyjä monitavoiteoptimointimenetelmiä järjestelmällisesti vasta muutaman vuosikymmenen ajan.

Tämä väitöskirja käsittelee aiempia ja nykyisiä tapoja käyttää monitavoiteoptimoinnin menetelmiä ohjatussa koneoppimisessa, erityisesti monikerroksisten hermoverkkojen muodostamisessa luokittelutehtäviä varten. Väitöskirjassa luodaan aihepiiriin kattava kirjallisuuskatsaus, jonka tekemistä on tuettu osaksi kirjoittajan kehittämällä työkaluohjelmistolla. Katsauksen perusteella listataan otsikon mukaisen aihepiirin tärkeimmiksi havaitut tavoitteet ja toteutustavat, joita myös sovelletaan uuden toteutuksen pohjana. Uusi monitavoitetoteutus on suunniteltu tuottamaan rakenteeltaan samanlaisia malleja kuin aiempi, teollisissa projekteissa käytetty, yksitavoitteinen hermoverkkototeutus. Toteutusta ja Pareto-rintamaan perustuvaa hermoverkon muodostamista esitellään simuloitujen opetusaineistojen avulla. Painopiste on tehtävään soveltuvien tiedonesitystapojen ja tavoitefunktioiden muotoilemisessa.

BIBLIOGRAPHY

Abbass, H. A. 2001. A memetic pareto evolutionary approach to artificial neural networks. In AI 2001: Advances in Artificial Intelligence. Ed. by Stumptner, M., Corbett, D., and Brooks, M. Vol. 2256. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1–12.

Abbass, H. A. 2002. An evolutionary artificial neural networks approach for breast cancer diagnosis. Artificial Intelligence in Medicine 25 (3), 265–281.

Abbass, H. A. 2003a. Pareto neuro-evolution: constructing ensemble of neural networks using multi-objective optimization. In The 2003 Congress on Evolutionary Computation, CEC '03. Vol. 3, 2074–2080.

Abbass, H. A. 2003b. Speeding up backpropagation using multiobjective evolutionary algorithms. Neural Computation 15, 2705–2726.

Abbass, H. A., Sarker, R., and Newton, C. 2001. PDE: A Pareto-frontier differential evolution approach for multi-objective optimization problems. In Proceedings of the 2001 Congress on Evolutionary Computation. Vol. 2, 971–978.

Ackley, D. H. 1987. A Connectionist Machine for Genetic Hillclimbing. Norwell, MA, USA: Kluwer Academic Publishers.

Agarwal, N., Haque, E., Liu, H., and Parsons, L. 2005. Research paper recommender systems: a subspace clustering approach. In Advances in Web-Age Information Management. Ed. by Fan, W., Wu, Z., and Yang, J. Vol. 3739. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 475–491.

Aljaber, B., Stokes, N., Bailey, J., and Pei, J. 2010. Document clustering of scientific texts using citation contexts. Information Retrieval 13 (2), 101–131.

Almeida, L. M. and Ludermir, T. B. 2010. A multi-objective memetic and hybrid methodology for optimizing the parameters and performance of artificial neural networks. Neurocomputing 73 (7–9), 1438–1450.

Aşkan, A. and Sayın, S. 2013. SVM classification for imbalanced data sets using a multiobjective optimization framework. Annals of Operations Research 216 (1), 191–203.

Averbuch, A., Kärkkäinen, T., Neittaanmäki, P., Nieminen, P., Rabin, N., and Zheludev, V. 2010. Applications of Dimension Reduction, Classification, and Neural Prediction for Industrial Process Data. Technical report. Series C. Software and Computational Engineering. Department of Mathematical Information Technology, University of Jyväskylä.

Bäck, T., Hammel, U., and Schwefel, H.-P. 1997. Evolutionary computation: comments on the history and current state. IEEE Transactions on Evolutionary Computation 1 (1), 3–17.

Bernadó i Mansilla, E. and Garrell i Guiu, J. M. 2001. MOLeCS: using multiobjective evolutionary algorithms for learning. In Evolutionary Multi-Criterion Optimization. Ed. by Zitzler, E., Thiele, L., Deb, K., Coello Coello, C., and Corne, D. Vol. 1993. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 696–710.

Bird, S., Klein, E., and Loper, E. 2009. Natural Language Processing with Python. O'Reilly Media Inc.

Bishop, C. M. 1995. Neural Networks for Pattern Recognition. Oxford University Press.

Boyack, K. W., Klavans, R., and Börner, K. 2005. Mapping the backbone of science. Scientometrics 64 (3), 351–374.

Branke, J., Deb, K., Miettinen, K., and Słowiński, R., eds. Multiobjective Optimization, Interactive and Evolutionary Approaches 2008. Vol. 5252. Lecture Notes in Computer Science. Springer.

Bravo-Alcobendas, D. and Sorzano, C. O. S. 2009. Clustering of biomedical scientific papers. In IEEE International Symposium on Intelligent Signal Processing, 2009. WISP 2009. 205–209.

Brealey, R. A., Myers, S. C., and Allen, F. 2011. Principles of Corporate Finance. 10th ed. McGraw-Hill/Irwin.

Bremermann, H. J. 1968. Numerical optimization procedures derived from biological evolution processes. In Cybernetic Problems in Bionics. Ed. by Oestreicher, H. L. and Moore, D. R. New York: Gordon and Breach Science Publishers Inc., 597–616.

Buckheit, J. B. and Donoho, D. L. 1995. WaveLab and reproducible research. In Wavelets and Statistics. Ed. by Antoniadis, A. and Oppenheim, G. New York, NY: Springer New York, 55–81.

Budgen, D., Turner, M., Brereton, P., and Kitchenham, B. 2008. Using mapping studies in software engineering. In Proceedings of PPIG, 195–204.

Callon, M., Courtial, J.-P., Turner, W. A., and Bauin, S. 1983. From translations to problematic networks: an introduction to co-word analysis. Social Science Information 22 (2), 191–235.

Campaigne, H. 1959. Some experiments in machine learning. Proceedings of the Western Joint Computer Conference, 173–175.

Capel-Cuevas, S., López-Ruiz, N., Martinez-Olmos, A., Cuéllar, M. P., Carmen Pegalajar, M. del, Palma, A. J., Orbe-Payá, I. de, and Capitán-Vallvey, L. F. 2012. A compact optical instrument with artificial neural network for pH determination. Sensors 12 (5), 6746–6763.

Carbonell, J. G., Michalski, R. S., and Mitchell, T. M. 1983. Machine learning: a historical and methodological analysis. AI Magazine 4 (3), 69–79.

Cartwright, H. and Curteanu, S. 2013. Neural networks applied in chemistry. II. Neuro-evolutionary techniques in process modeling and optimization. Industrial & Engineering Chemistry Research 52 (36), 12673–12688.

Castro, L. N. de and Timmis, J. 2002. Artificial Immune Systems: A New Computational Approach. London. UK.: Springer-Verlag, 182–196.

Chand, S. and Wagner, M. 2015. Evolutionary many-objective optimization: a quick-start guide. Surveys in Operations Research and Management Science 20 (2), 35–42.

Chatelain, C., Adam, S., Lecourtier, Y., Heutte, L., and Paquet, T. 2010. A multi-model selection framework for unknown and/or evolutive misclassification cost problems. Pattern Recognition 43 (3), 815–823.

Chen, C. 2006. CiteSpace II: detecting and visualizing emerging trends and transient patterns in scientific literature. Journal of the American Society for Information Science and Technology 57 (3), 359–377.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. 2015. The loss surfaces of multilayer networks. In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015. Ed. by Lebanon, G. and Vishwanathan, S. V. N. Vol. 38. JMLR Proceedings. JMLR.

Cohen, A. M., Hersh, W. R., Peterson, K., and Yen, P.-Y. Y. 2006. Reducing workload in systematic review preparation using automated citation classification. Journal of the American Medical Informatics Association: JAMIA 13 (2), 206–219.

Coifman, R. R. and Lafon, S. 2006. Diffusion maps. Applied and Computational Harmonic Analysis 21 (1), 5–30.

Corne, D. W., Jerram, N. R., Knowles, J. D., and Oates, M. J. 2001. PESA-II: region-based selection in evolutionary multiobjective optimization. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '2001). Morgan Kaufmann Publishers, 283–290.

Costa, M. A., Braga, A. P., Menezes, B. R., Teixeira, R. A., and Parma, G. G. 2003. Training neural networks with a multi-objective sliding mode control algorithm. Neurocomputing 51, 467–473.

Costa, M. A., Braga, A. P., and Menezes, B. R. de 2012. Convergence analysis of sliding mode trajectories in multi-objective neural networks learning. Neural Networks 33, 21–31.

Cruz-Ramírez, M., Hervás-Martínez, C., Gutiérrez, P. A., Pérez-Ortiz, M., Briceño, J., and Mata, M. de la 2012. Memetic Pareto differential evolutionary neural network used to solve an unbalanced liver transplantation problem. Soft Computing 17 (2), 275–284.

Cruz-Ramírez, M., Hervás-Martínez, C., Sánchez-Monedero, J., and Gutiérrez, P. 2014. Metrics to guide a multi-objective evolutionary algorithm for ordinal classification. Neurocomputing 135, 21–31.

Cruz-Ramírez, M., Sánchez-Monedero, J., Fernández-Navarro, F., Fernández, J. C., and Hervás-Martínez, C. 2010. Memetic pareto differential evolutionary artificial neural networks to determine growth multi-classes in predictive microbiology. Evolutionary Intelligence 3 (3), 187–199.

Curteanu, S. and Cartwright, H. 2011. Neural networks applied in chemistry. I. Determination of the optimal topology of multilayer perceptron neural networks. Journal of Chemometrics 25 (10), 527–549.

Darwin, C. 1859. On the origin of species by means of natural selection, or, The preservation of favoured races in the struggle for life. London: John Murray.

Das, I. and Dennis, J. E. 1997. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. Structural optimization 14 (1), 63–69.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6 (2), 182–197.

Deb, K. and Jain, H. 2014. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE Transactions on Evolutionary Computation 18 (4), 577–601.

Deb, K. 2008. Introduction to evolutionary multiobjective optimization. In Multiobjective Optimization, Interactive and Evolutionary Approaches. Ed. by Branke, J., Deb, K., Miettinen, K., and Słowiński, R. Vol. 5252. Lecture Notes in Computer Science. Springer, 59–96.

Deb, K. and Gupta, S. 2011. Understanding knee points in bicriteria problems and their implications as preferred solution principles. Engineering Optimization 43 (11), 1175–1204.

Du, W., Leung, S. Y. S., and Kwong, C. K. 2014. Time series forecasting by neural networks: a knee point-based multiobjective evolutionary algorithm approach. Expert Systems with Applications 41 (18), 8049–8061.

Edgeworth, F. Y. 1881. Mathematical Psychics: An Essay on the Application of Mathematics to the Moral Sciences. C. Kegan Paul & Co.

Ehrgott, M. 2005. Multicriteria Optimization. Springer-Verlag New York, Inc.

Elkan, C. 2001. The foundations of cost-sensitive learning. In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01), 973–978.

Everitt, B. S., Landau, S., Leese, M., and Stahl, D. 2011. Cluster Analysis. 5th ed. John Wiley & Sons, Ltd.

Everson, R. M. and Fieldsend, J. E. 2006. Multi-class ROC analysis from a multiobjective optimisation perspective. Pattern Recognition Letters 27 (8), 918–927.

Fawcett, T. 2006. An introduction to ROC analysis. Pattern Recognition Letters 27 (8), 861–874.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. 1996a. From data mining to knowledge discovery in databases. AI Magazine 17 (3), 37–54.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. 1996b. The KDD process for extracting useful knowledge from volumes of data. Communications of the ACM 39 (11), 27–34.

Felizardo, K. R. 2012. Evidence-based software engineering: systematic literature review process based on visual text mining. PhD thesis. Universidade de São Paulo.

Felizardo, K. R., Nakagawa, E. Y., Feitosa, D., Minghim, R., and Maldonado, J. C. 2010. An approach based on visual text mining to support categorization and classification in the systematic mapping. In Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering. EASE'10. UK: British Computer Society, 34–43.

Fernández, J. C., Hervás, C., Martínez, F. J., Gutiérrez, P. A., and Cruz, M. 2009. Memetic Pareto differential evolution for designing artificial neural networks in multiclassification problems using cross-entropy versus sensitivity. In Hybrid Artificial Intelligence Systems: 4th International Conference, HAIS 2009, Salamanca, Spain, June 10-12, 2009. Proceedings. Ed. by Corchado, E., Wu, X., Oja, E., Herrero, Á., and Baruque, B. Springer Berlin Heidelberg, 433–441.

Fernández-Caballero, J. C., Hervás-Martinez, C., Martínez-Estudillo, F. J., and Gutiérrez, P. A. 2011. Memetic Pareto evolutionary artificial neural networks to determine growth/no-growth in predictive microbiology. Applied Soft Computing 11 (1), 534–550.

Fernandez-Caballero, J. C., Martinez, F. J., Hervas, C., and Gutierrez, P. A. 2010. Sensitivity versus accuracy in multiclass problems using memetic Pareto evolutionary neural networks. IEEE Transactions on Neural Networks 21 (5), 750–770.

Fieldsend, J. E., Everson, R. M., and Singh, S. 2003. Using unconstrained elite archives for multiobjective optimization. IEEE Transactions on Evolutionary Computation 7 (3), 305–323.

Fieldsend, J. E. and Singh, S. 2002. Pareto multiobjective nonlinear regression modelling to aid CAPM analogous forecasting. In Proceedings of the 2002 International Joint Conference on Neural Networks, 2002. IJCNN '02. Vol. 1, 388–393.

Fieldsend, J. E. and Singh, S. 2005. Pareto evolutionary neural networks. IEEE Transactions on Neural Networks 16 (2), 338–354.

Fletcher, R. and Reeves, C. M. 1964. Function minimization by conjugate gradients. The Computer Journal 7 (2), 149–154.

Fogel, D. B. 1994. An introduction to simulated evolutionary optimization. IEEE Transactions on Neural Networks 5 (1), 3–14.

Fogel, D. B. 1998. Evolutionary Computation: The Fossil Record. 1st ed. Wiley-IEEE Press.

Fonseca, C. M. and Fleming, P. J. 1993. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In Proceedings of the 5th International Conference on Genetic Algorithms. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 416–423.

Friedberg, R. M. 1958. A learning machine: part I. IBM Journal of Research and Development 2 (1), 2–13.

Friedberg, R. M., Dunham, B., and North, J. H. 1959. A learning machine: part II. IBM Journal of Research and Development 3 (3), 282–287.

Friedman, G. J. 1956. Selective feedback computers for engineering synthesis and nervous system analogy. Master's thesis, University of California, Los Angeles.

García-Pedrajas, N., Hervás-Martínez, C., and Muñoz-Pérez, J. 2002. Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks). Neural Networks 15 (10), 1259–1278.

Goh, C.-K., Teoh, E. J., and Tan, K. C. 2008. Hybrid multiobjective evolutionary design for artificial neural networks. IEEE Transactions on Neural Networks 19 (9), 1531–1548.

Goldberg, D. E. 1989. Genetic Algorithms in Search, Optimization and Machine Learning. 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

González, J., Rojas, I., Ortega, J., Pomares, H., Fernández, F. J., and Díaz, A. F. 2003. Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation. IEEE Transactions on Neural Networks 14 (6), 1478–1495.

Gräning, L., Jin, Y., and Sendhoff, B. 2006. Generalization improvement in multi-objective learning. In International Joint Conference on Neural Networks, 2006. IJCNN '06. 4839–4846.

Gutiérrez, P. A., Hervás-Martínez, C., Martínez-Estudillo, F. J., and Carbonero, M. 2012. A two-stage evolutionary algorithm based on sensitivity and accuracy for multi-class problems. Information Sciences 197, 20–37.

Hagan, M. T. and Menhaj, M. B. 1994. Training feedforward networks with the Marquardt algorithm. IEEE Transactions on Neural Networks 5 (6), 989–993.

Hammer, B. and Gersmann, K. 2003. A note on the universal approximation capability of support vector machines. Neural Processing Letters 17 (1), 43–53.

Hastie, T., Tibshirani, R., and Friedman, J. H. 2011. The Elements of Statistical Learning. New York: Springer.

Hatanaka, T., Kondo, N., and Uosaki, K. 2003. Multi-objective structure selection for radial basis function networks based on genetic algorithm. In The 2003 Congress on Evolutionary Computation, CEC '03. Vol. 2, 1095–1100.

Haykin, S. 2009. Neural Networks and Machine Learning. 3rd ed. Upper Saddle River: Pearson Education, Inc.

He, H. and Garcia, E. A. 2009. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering 21 (9), 1263–1284.

Heikkola, E., Miettinen, K., and Nieminen, P. 2006. Multiobjective optimization of an ultrasonic transducer using NIMBUS. Ultrasonics 44 (4), 368–380.

Hinton, G. E., Osindero, S., and Teh, Y.-W. 2006. A fast learning algorithm for deep belief nets. Neural Computation 18 (7), 1527–1554.

Hornik, K., Stinchcombe, M., and White, H. 1989. Multilayer feedforward networks are universal approximators. Neural Networks 2 (5), 359–366.

Iacca, G. 2011. Memory-saving optimization algorithms for systems with limited hardware. PhD thesis. University of Jyväskylä.

Igel, C. 2005. Multi-objective model selection for support vector machines. In Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005). Vol. 3410. Lecture Notes in Computer Science. Springer Verlag, 534–546.

Igel, C. and Hüsken, M. 2000. Improving the Rprop learning algorithm. In Proceedings of the Second International Symposium on Neural Computation, NC'2000. ICSC Academic Press, 115–121.

Ignizio, J. P. 1976. Goal programming and extensions. Lexington Books.

Ivannikov, A., Nieminen, P., and Kärkkäinen, T. 2010. A Beamforming Framework for Detection of Underwater Sound-Emitting Objects Using Acoustic Sensor Arrays. Technical report. Series C. Software and Computational Engineering. Department of Mathematical Information Technology, University of Jyväskylä.

Jaccard, P. 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bulletin de la Société Vaudoise des Sciences Naturelles 37, 547–579.

Jesson, J. K., Lacey, F. M., and Matheson, L. 2011. Doing your literature review: traditional and systematic techniques. SAGE Publications Ltd.

Jin, Y., ed. Multi-Objective Machine Learning 2006. Studies in Computational Intelligence. Springer.

Jin, Y., Okabe, T., and Sendhoff, B. 2004a. Evolutionary multiobjective approach to constructing neural network ensembles for regression. In Applications of Multi-objective Evolutionary Algorithms. Ed. by Coello, C. A. C. and Lamont, G. B. Singapore: World Scientific, 653–672.

Jin, Y., Okabe, T., and Sendhoff, B. 2004b. Neural network regularization and ensembling using multi-objective evolutionary algorithms. In Proceedings of the 2004 Congress on Evolutionary Computation (CEC '04). IEEE, 1–8.

Jin, Y., Olhofer, M., and Sendhoff, B. 2001. Dynamic weighted aggregation for evolutionary multi-objective optimization: why does it work and how? In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '2001). Morgan Kaufmann Publishers, 1042–1049.

Jin, Y. and Sendhoff, B. 2006. Alleviating catastrophic forgetting via multi-objective learning. In International Joint Conference on Neural Networks, 2006. IJCNN '06. 3335–3342.

Jin, Y. and Sendhoff, B. 2008. Pareto-based multiobjective machine learning: an overview and case studies. IEEE Transactions on Systems, Man, and Cybernetics, Part C 38 (3), 397–415.

Jin, Y., Wen, R., and Sendhoff, B. 2007. Evolutionary multi-objective optimization of spiking neural networks. In Artificial Neural Networks – ICANN 2007. Ed. by Sá, J. M. de, Alexandre, L. A., Duch, W., and Mandic, D. Vol. 4668. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 370–379.

Kärkkäinen, T. 2002. MLP in layer-wise form with applications to weight decay. Neural Computation 14 (6), 1451–1480.

Kärkkäinen, T. and Heikkola, E. 2004. Robust formulations for training multilayer perceptrons. Neural Computation 16 (4), 837–862.

Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. 2011. Lower upper bound estimation method for construction of neural network-based prediction intervals. IEEE Transactions on Neural Networks 22 (3), 337–346.

Kim, D. 2004. Structural risk minimization on decision trees using an evolutionary multiobjective optimization. In Genetic Programming. Ed. by Keijzer, M., O'Reilly, U.-M., Lucas, S., Costa, E., and Soule, T. Vol. 3003. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 338–348.

Kitchenham, B. 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering Version 2.3. EBSE Technical Report. Keele University and University of Durham.

Klema, V. and Laub, A. 1980. The singular value decomposition: its computation and some applications. IEEE Transactions on Automatic Control 25 (2), 164–176.

Klopf, A. H. and Gose, E. 1969. An evolutionary pattern recognition network. IEEE Transactions on Systems Science and Cybernetics 5 (3), 247–250.

Koch, C. 1998. Biophysics of Computation: Information Processing in Single Neurons. 1st ed. Oxford University Press.

Kottathra, K. and Attikiouzel, Y. 1996. A novel multicriteria optimization algorithm for the structure determination of multilayer feedforward neural networks. Journal of Network and Computer Applications 19 (2), 135–147.

Kuhn, H. W. and Tucker, A. W. 1951. Nonlinear programming. In Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability. Berkeley, California: University of California Press, 481–492.

Kupinski, M. A. and Anastasio, M. A. 1999. Multiobjective genetic optimization of diagnostic classifiers with implications for generating receiver operating characteristic curves. IEEE Transactions on Medical Imaging 18 (8), 675–685.

Lafon, S. and Lee, A. B. 2006. Diffusion maps and coarse-graining: a unified framework for dimensionality reduction, graph partitioning, and data set parameterization. IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (9), 1393–1403.

Lamarck, J. B. 1809. Philosophie zoologique, ou, Exposition des considérations relative à l'histoire naturelle des animaux.

Land, A. H. and Doig, A. G. 1960. An automatic method of solving discrete programming problems. Econometrica 28 (3), 497–520.

Leydesdorff, L., Carley, S., and Rafols, I. 2013. Global maps of science based on the new Web-of-Science categories. Scientometrics 94 (2), 589–593.

Liu, G. P. and Kadirkamanathan, V. 1995. Learning with multi-objective criteria. In Fourth International Conference on Artificial Neural Networks, 1995. 53–58.

Liu, Y., Yao, X., and Higuchi, T. 2000. Evolutionary ensembles with negative correlation learning. IEEE Transactions on Evolutionary Computation 4 (4), 380–387.

Malheiros, V., Hohn, E., Pinho, R., and Mendonca, M. 2007. A visual text mining approach for systematic reviews. In First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007). IEEE, 245–254.

Markowska-Kaczmar, U. and Wnuk-Lipinski, P. 2004. Rule extraction from neural network by genetic algorithm with Pareto optimization. In Artificial Intelligence and Soft Computing - ICAISC 2004. Ed. by Rutkowski, L., Siekmann, J. H., Tadeusiewicz, R., and Zadeh, L. A. Vol. 3070. Lecture Notes in Computer Science. Springer, 450–455.

Matsuyama, Y. 1996. Harmonic competition: a self-organizing multiple criteria optimization. IEEE Transactions on Neural Networks 7 (3), 652–668.

Matsuyama, Y., Nakayama, H., Sasai, T., and Chen, Y. P. 1994. Penalized learning as multiple object optimization. In Proceedings of the 1994 IEEE International Conference on Neural Networks. Vol. 1, 187–192.

Matwin, S., Kouznetsov, A., Inkpen, D., Frunza, O., and O'Blenis, P. 2010. A new algorithm for reducing the workload of experts in performing systematic reviews. Journal of the American Medical Informatics Association: JAMIA 17 (4), 446–453.

McCulloch, W. S. and Pitts, W. 1943. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics 5 (4), 115–133.

Mehta, P. and Schwab, D. J. 2014. An exact mapping between the variational renormalization group and deep learning. arXiv:1410.3831 [stat.ML].

Mesquita, D. P. P., Neto, A. N. A., Neto, J. F. Q., Gomes, J. P. P., and Rodrigues, L. R. 2016. Using robust extreme learning machines to predict cotton yarn strength and hairiness. In ESANN 2016: Proceedings of the 24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 65–70.

Miettinen, K. 1999. Nonlinear Multiobjective Optimization. Vol. 12. International Series in Operations Research and Management Science. Kluwer Academic Publishers.

Miettinen, K. 2008. Introduction to multiobjective optimization: noninteractive approaches. In Multiobjective Optimization, Interactive and Evolutionary Approaches. Ed. by Branke, J., Deb, K., Miettinen, K., and Słowiński, R. Vol. 5252. Lecture Notes in Computer Science. Springer, 1–26.

Mininno, E. 2011. Advanced optimization algorithms for applications in control engineering. PhD thesis. University of Jyväskylä.

Møller, M. F. 1993. A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks 6 (4), 525–533.

Montana, D. J. and Davis, L. 1989. Training feedforward neural networks using genetic algorithms. In Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1. IJCAI'89. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 762–767.

Moscato, P. 1989. On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms. C3P Report 778. Pasadena, CA.

Mucciardi, A. N. and Gose, E. E. 1966. Evolutionary pattern recognition in incomplete nonlinear multithreshold networks. IEEE Transactions on Electronic Computers EC-15 (2), 257–261.

Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., and Coello Coello, C. A. 2014a. A survey of multiobjective evolutionary algorithms for data mining: part I. IEEE Transactions on Evolutionary Computation 18 (1), 4–19.

Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., and Coello Coello, C. A. 2014b. A survey of multiobjective evolutionary algorithms for data mining: part II. IEEE Transactions on Evolutionary Computation 18 (1), 20–35.

Nadler, B., Lafon, S., Coifman, R., and Kevrekidis, I. G. 2008. Diffusion maps – a probabilistic interpretation for spectral embedding and clustering algorithms. In Principal Manifolds for Data Visualization and Dimension Reduction. Ed. by Barth, T. J., Griebel, M., Keyes, D. E., Nieminen, R. M., Roose, D., Schlick, T., Gorban, A. N., Kégl, B., Wunsch, D. C., and Zinovyev, A. Y. Vol. 58. Lecture Notes in Computational Science and Engineering. Berlin Heidelberg: Springer, 238–260.

Neri, F. 2007. Fitness diversity adaptation in memetic algorithms. PhD thesis. University of Jyväskylä.

Neri, F. and Cotta, C. 2012. A primer on memetic algorithms. In Handbook of Memetic Algorithms. Ed. by Neri, F., Cotta, C., and Moscato, P. Springer, 43–52.

Neri, F., Cotta, C., and Moscato, P., eds. Handbook of Memetic Algorithms 2012. Springer.

Nieminen, P. and Kärkkäinen, T. 2009. Ideas about a regularized MLP classifier by means of weight decay stepping. In Adaptive and Natural Computing Algorithms: 9th International Conference, ICANNGA 2009. Ed. by Kolehmainen, V., Toivanen, P., and Beliczynski, B. Vol. 5495. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 32–41.

Nieminen, P. and Kärkkäinen, T. 2010. Comparison of MLP cost functions to dodge mislabeled training data. In Proceedings of The 2010 International Joint Conference on Neural Networks (IJCNN '10). IEEE.

Nieminen, P. and Kärkkäinen, T. 2016. Multicriteria optimized MLP for imbalanced learning. In ESANN 2016: Proceedings of the 24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 459–464.

Nieminen, P., Kärkkäinen, T., Luostarinen, K., and Muhonen, J. 2011. Neural prediction of product quality based on pilot paper machine process measurements. In Adaptive and Natural Computing Algorithms: 10th International Conference, ICANNGA 2011. Ed. by Dobnikar, A., Lotrič, U., and Šter, B. Vol. 6593. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 240–249.

Nieminen, P., Pölönen, I., and Sipola, T. 2013. Research literature clustering using diffusion maps. Journal of Informetrics 7 (4), 874–886.

Nieminen, P., Rabin, N., Kärkkäinen, T., Averbuch, A., and Äyramö, S. 2010. Robust Clustering and Neural Network Training with Dimension Reduction for Industrial Use. Technical report. Series C. Software and Computational Engineering. Department of Mathematical Information Technology, University of Jyväskylä.

Nurminen, M., Honkaranta, A., and Kärkkäinen, T. 2005. ExtMiner: combining multiple ranking and clustering algorithms for structured document retrieval. In 16th International Workshop on Database and Expert Systems Applications (DEXA'05), 1036–1040.

Pareto, V. 1896. Cours d'Economie Politique. Genève: Librairie Droz.

Pasti, R., Castro, L. N., Coelho, G. P., and Zuben, F. J. 2010. Neural network ensembles: immune-inspired approaches to the diversity of components. Natural Computing 9 (3), 625–653.

Poikolainen, I. 2014. Simple memetic computing structures for global optimization. PhD thesis. University of Jyväskylä.

Potter, M. A. and Jong, K. A. D. 1994. A cooperative coevolutionary approach to function optimization. In Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature. Ed. by Davidor, Y., Schwefel, H.-P., and Männer, R. PPSN III. Springer Berlin Heidelberg, 249–257.

Qasem, S. N., Shamsuddin, S. M., Hashim, S. Z. M., Darus, M., and Al-Shammari, E. 2013. Memetic multiobjective particle swarm optimization-based radial basis function network for classification problems. Information Sciences 239, 165–190.

Quan, H., Srinivasan, D., and Khosravi, A. 2014. Particle swarm optimization for construction of neural network-based prediction intervals. Neurocomputing 127, 172–180.

Rosales-Perez, A., Coello, C. A. C., Gonzalez, J. A., Reyes-Garcia, C. A., and Escalante, H. J. 2013. A hybrid surrogate-based approach for evolutionary multi-objective optimization. In IEEE Congress on Evolutionary Computation (CEC), 2013, 2548–2555.

Rosales-Pérez, A., Gonzalez, J. A., Coello, C. A. C., Escalante, H. J., and Reyes-Garcia, C. A. 2015. Surrogate-assisted multi-objective model selection for support vector machines. Neurocomputing 150, Part A, 163–172.

Rosenblatt, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review 65 (6), 386–408.

Rousseeuw, P. J. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20 (1), 53–65.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1986a. Learning internal representations by error propagation. In Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations. Ed. by Rumelhart, D. E., McClelland, J. L., and PDP Research Group. Cambridge, MA, USA: MIT Press, 318–362.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1986b. Learning representations by back-propagating errors. Nature 323, 533–536.

Samuel, A. L. 1959. Some studies in machine learning using the game of checkers. IBM Journal of Research and Development 3 (3), 210–229.

Sánchez-Monedero, J., Gutiérrez, P. A., Fernández-Navarro, F., and Hervás-Martínez, C. 2011. Weighting efficient accuracy and minimum sensitivity for evolving multi-class classifiers. Neural Processing Letters 34 (2), 101–116.

Schaffer, J. D., Whitley, D., and Eshelman, L. J. 1992. Combinations of genetic algorithms and neural networks: a survey of the state of the art. In International Workshop on Combinations of Genetic Algorithms and Neural Networks, 1992, COGANN-92. 1–37.

Sharpe, W. F. 1964. Capital asset prices: a theory of market equilibrium under conditions of risk. The Journal of Finance 19 (3), 425–442.

Small, H. 1973. Co-citation in the scientific literature: a new measure of the relationship between two documents. Journal of the American Society for Information Science 24 (4), 265–269.

Srinivas, N. and Deb, K. 1994. Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation 2 (3), 221–248.

Stanley, K. O. and Miikkulainen, R. 2002. Evolving neural networks through augmenting topologies. Evolutionary Computation 10 (2), 99–127.

Storn, R. and Price, K. 1997. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11 (4), 341–359.

Szczuka, M., Janusz, A., and Herba, K. 2012. Semantic clustering of scientific articles with use of DBpedia knowledge base. In Intelligent Tools for Building a Scientific Information Platform. Ed. by Bembenik, R., Skonieczny, L., Rybiński, H., and Niezgodka, M. Vol. 390. Studies in Computational Intelligence. Springer, 61–76.

Takahashi, R. H. C., Peres, P. L. D., and Ferreira, P. A. V. 1997. Multiobjective H_2/H_∞ guaranteed cost PID design. IEEE Control Systems 17 (5), 37–47.

Taormina, R. and Chau, K.-W. 2015a. ANN-based interval forecasting of streamflow discharges using the LUBE method and MOFIPS. Engineering Applications of Artificial Intelligence 45, 429–440.

Taormina, R. and Chau, K.-W. 2015b. Neural network river forecasting with multi-objective fully informed particle swarm optimization. Journal of Hydroinformatics 17 (1), 99–113.

Teixeira, R. A., Braga, A. P., Takahashi, R. H., and Saldanha, R. R. 2000. Improving generalization of MLPs with multi-objective optimization. Neurocomputing 35 (1–4), 189–194.

Teoh, E. J., Tan, K. C., and Xiang, C. 2006. Estimating the number of hidden neurons in a feedforward network using the singular value decomposition. IEEE Transactions on Neural Networks 17 (6), 1623–1629.

Thorndike, R. L. 1953. Who belongs in the family. Psychometrika, 267–276.

Tikk, D., Kóczy, L. T., and Gedeon, T. D. 2003. A survey on universal approximation and its limits in soft computing techniques. International Journal of Approximate Reasoning 33 (2), 185–202.

Tirronen, V. 2008. Global optimization using memetic differential evolution with applications to low level machine vision. PhD thesis. University of Jyväskylä.

Tsakonas, A. 2014. An analysis of accuracy-diversity trade-off for hybrid combined system with multiobjective predictor selection. Applied Intelligence 40 (4), 710–723.

Tseng, Y.-H. and Tsay, M.-Y. 2013. Journal clustering of library and information science for subfield delineation using the bibliometric analysis toolkit: CATAR. Scientometrics 95 (2), 503–528.

Utkin, V. 1977. Variable structure systems with sliding modes. IEEE Transactions on Automatic Control 22 (2), 212–222.

Waltman, L., Eck, N. J. van, and Noyons, E. C. 2010. A unified approach to mapping and clustering of bibliometric networks. Journal of Informetrics 4 (4), 629–635.

Ware, W. H. 1955. Introduction to session on learning machines. In Proceedings of the March 1–3, 1955, Western Joint Computer Conference. AFIPS '55 (Western). ACM, 85–85.

Wartiainen, P. and Kärkkäinen, T. 2015. Hierarchical, prototype-based clustering of multiple time series with missing values. In ESANN 2015: Proceedings of the 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 95–100.

Wiegand, S., Igel, C., and Handmann, U. 2004. Evolutionary multi-objective optimization of neural networks for face detection. International Journal of Computational Intelligence and Applications 4 (3), 237–253.

Yao, X. 1993. A review of evolutionary artificial neural networks. International Journal of Intelligent Systems 4, 539–567.

Yao, X. 1999. Evolving artificial neural networks. Proceedings of the IEEE 87 (9), 1423–1447.

Zakian, V. and Al-Naib, U. 1973. Design of dynamical and control systems by the method of inequalities. Proceedings of the Institution of Electrical Engineers 120 (11), 1421–1427.

Zaman, A. N. K., Matsakis, P., and Brown, C. 2011. Evaluation of stop word lists in text retrieval using latent semantic indexing. In Sixth International Conference on Digital Information Management (ICDIM), 2011, 133–136.

Zhang, J. and Sanderson, A. C. 2009. JADE: adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 13 (5), 945–958.

Zhang, Q. and Li, H. 2007. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation 11 (6), 712–731.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Fonseca, V. G. da 2003. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation 7 (2), 117–132.

Zitzler, E., Laumanns, M., and Thiele, L. 2002. SPEA2: improving the strength pareto evolutionary algorithm for multiobjective optimization. In Evolutionary Methods for Design, Optimisation, and Control with Applications to Industrial Problems: Proceedings of the EUROGEN2001 Conference. CIMNE, Barcelona, Spain, 95–100.

Zong, W., Huang, G.-B., and Chen, Y. 2013. Weighted extreme learning machine for imbalance learning. Neurocomputing 101, 229–242.