NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

Explorations of the Semantic Learning Machine Neuroevolution Algorithm

Dynamic Training Data Use and Ensemble Construction Methods

Marta Sofia Lopes Seca

Dissertation presented as the partial requirement for obtaining a Master's degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

Explorations of the Semantic Learning Machine Neuroevolution Algorithm

Dynamic Training Data Use and Ensemble Construction Methods

Marta Sofia Lopes Seca

Dissertation presented as the partial requirement for obtaining a Master's degree in Data Science and Advanced Analytics

Adviser: Ivo Gonçalves

Co-Adviser: Mauro Castelli

February 2020

Explorations of the Semantic Learning Machine Neuroevolution Algorithm

Copyright © Marta Sofia Lopes Seca, NOVA Information Management School, NOVA University of Lisbon.
The NOVA Information Management School and the NOVA University of Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.

This document was created using the (pdf)LaTeX processor, based in the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Abstract

As the world's technology evolves, the power to implement new and more efficient algorithms increases but so does the complexity of the problems at hand. Neuroevolution algorithms fit in this context in the sense that they are able to evolve Artificial Neural Networks (ANNs).

The recently proposed Neuroevolution algorithm called Semantic Learning Machine (SLM) has the advantage of searching over unimodal error landscapes in any Supervised Learning task where the error is measured as a distance to the known targets. The absence of local optima in the search space results in a more efficient learning when compared to other neuroevolution algorithms. This work studies how different approaches of dynamically using the training data affect the generalization of the SLM algorithm. Results show that these methods can be useful in offering different alternatives to achieve a superior generalization. These approaches are evaluated experimentally in fifteen real-world binary classification data sets. Across these fifteen data sets, results show that the SLM is able to outperform the Multilayer Perceptron (MLP) in 13 out of the 15 considered problems with statistical significance after parameter tuning was applied to both algorithms.

Furthermore, this work also considers how different ensemble construction methods such as a simple averaging approach, Bagging and Boosting affect the resulting generalization of the SLM and MLP algorithms. Results suggest that the stochastic nature of the SLM offers enough diversity to the base learner in a way that a simple averaging method can be competitive when compared to more complex techniques like Bagging and Boosting.

Keywords: Semantic Learning Machine; Neuroevolution; Evolutionary Machine Learning; Artificial Neural Networks; Deep Learning

Resumo

À medida que a tecnologia evolui, a possibilidade de implementar algoritmos novos e mais eficientes aumenta, no entanto, a complexidade dos problemas com que nos deparamos também se torna maior. Algoritmos de *Neuroevolution* encaixam-se neste contexto, na medida em que são capazes de evoluir Artificial Neural Networks (ANNs). O algoritmo de Neuroevolution recentemente proposto chamado Semantic Learning Machine (SLM) tem a vantagem de procurar sobre *landscapes* de erros unimodais em qualquer problema de Supervised Learning, onde o erro é medido como a distância aos alvos conhecidos. A não existência de *local optima* no espaço de procura resulta numa aprendizagem mais eficiente quando comparada com outros algoritmos de Neuroevolution. Este trabalho estuda como métodos diferentes de uso dinâmico de dados de treino afeta a generalização do algoritmo SLM. Os resultados mostram que estes métodos são úteis a oferecer uma alternativa que atinge uma generalização competitiva. Estes métodos são testados em quinze problemas reais de classificação binária. Nestes quinze problemas, o algoritmo SLM mostra superioridade ao Multilayer Perceptron (MLP) em treze deles com significância estatística depois de ser aplicado *parameter tuning* em ambos os algoritmos.

Para além disso, este trabalho também considera como diferentes métodos de construção de ensembles, tal como um simples método de *averaging*, Bagging e Boosting afetam os valores de generalização dos algoritmos SLM e MLP. Os resultados sugerem que a natureza estocástica da SLM oferece diversidade suficiente aos *base learners* de maneira a que o método mais simples de construção de ensembles se torne competitivo quando comparado com técnicas mais complexas como Bagging e Boosting.

Palavras-chave: Semantic Learning Machine; Neuroevolution; Evolutionary Machine Learning; Artificial Neural Networks; Deep Learning

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the world's technology evolves, the power to implement new and more efficient algorithms increases but so does the complexity of the problems at hand. Machine Learning algorithms have proved to be a great asset in several industries, from banking to health, essentially because they mimic human behavior but since they run on powerful machines and are capable of digesting much more information than a person, they end up surpassing our knowledge, *per se*, in many situations (Wang et al., 2016).

The **Artificial Neural Network (ANN)** is one of the most successfully used Machine Learning algorithms which is based on how our brain works, with neurons sending information through synapses between them. The algorithms of this nature evolved to something called Deep Learning simply because more computation power allowed for the usage of more layers of neurons. However, these methods require a lot of parameters to be selected and optimized – the number of neurons, number of layers, learning rate, among many others – so that the final result is reasonable and trustworthy. The Backpropagation algorithm, where the calculated error of a given Neural Network is propagated backwards with the goal of adjusting the existing weights, is still the most obvious option, however, it fails to provide the general topology of neurons and synapses. This means that this is still an open question in the realm of Machine Learning because the parameters and topology of an ANN rely heavily on the problem the Neural Network is trying to solve.

Neuroevolution originally tried to answer this question by making use of Evolutionary Algorithms (EAs) to find the best parameters for a given Neural Network. More recently, investigation showed that it was possible to have these algorithms creating and evolving Neural Networks as well as their parameters. The NeuroEvolution of Augmenting Topologies (NEAT) algorithm (K. O. Stanley & Miikkulainen, 2002) is

still broadly used, however, there are other competitive options. Recently, a Neuroevolution algorithm called Semantic Learning Machine (SLM) was proposed (Gonçalves, 2017; Gonçalves, Silva, & Fonseca, 2015b). The SLM has the advantage of searching over unimodal error landscapes in any Supervised Learning task where the error is measured as a distance to the known targets. The SLM showed superiority both in terms of training error and generalization ability when compared to NEAT and other well-established ML algorithms like Support Vector Machine (SVM) and Feedforward ANNs trained with the Backpropagation algorithm (Jagusch, Gonçalves, & Castelli, 2018).

The SLM neuroevolution algorithm has enabled different branches of investigation. Recently, the algorithm was used with Convolutional Neural Networks (CNNs) (Lapa, Gonçalves, Rundo, & Castelli, 2019a, 2019b) where the task of discriminating between benign and malignant prostate cancer lesions given multiparametric magnetic resonance imaging was under study. This image classification problem was addressed in the context of the PROSTATEx (Litjens, Debats, Barentsz, Karssemeijer, & Huisman, 2017) competition. The SLM was used as a background replacement for the training of the last fully connected layers of CNNs. In this case, the outputs of the convolutional layers are passed (without pre-training) to the SLM. The results are compared to the XmasNet state-of-the-art CNN (S. Liu, Zheng, Feng, & Li, 2017), specifically developed to address this challenge. Results suggest that the SLM is able to achieve a higher AUROC curve value than XmasNet with a statistically significant difference. It is important to mention that this performance is achieved without pre-training the underlying CNN or relying on backpropagation. Furthermore, it is of relevance to emphasize that the Semantic Learning Machine was only run on CPU (whereas XmasNet was trained using a GPU) and without any explicit parallelization. This adds value to the results obtained since each network evaluation could be suitable parallelized, thus achieving a higher speed-up.

The current work aims to continue the investigation on the SLM and how different methods of dynamically using the training data and ensemble construction approaches affect the resulting generalization.

This document is organized as follows: Chapter 2 intends to provide an in-depth overview of Neuroevolution. The Semantic Learning Machine (SLM) algorithm is explained in Chapter 3. Chapter 4 describes the Experimental Methodology procedures followed in the course of this project. Chapter 5 reports the experimental results and finally, Chapter 6 presents the final remarks and concludes the work.

Chapter 2

Neuroevolution Overview

Using an Artificial Neural Network is not an easy task. One of the main challenges when doing so may sit upon choosing an architecture or topology (here used as synonyms to reflect some of the hyperparameters inherent of any ANN, such as, the number of layers, number of neurons and how these are connected between each other). Figure 2.1 presents a simple example of a Neural Network with four input neurons, a hidden layer with two neurons and an output neuron.

By now, ANNs have proven to be useful in a vast amount of applications, however, specific rules to determine the best set of hyperparameters remain uncovered and something that is still problem dependent. Thus, it is still a task that requires a lot of effort and trial and error from the users. When trying to automate the aforementioned problem, one may recur to (1) use different search methods based on the gradient descent algorithm to optimize the weights and hyperparameters of the network, or (2) use evolutionary techniques to generate networks and to optimize the topology of the ANN. In the course of the last 30 years, neuroevolution techniques have been successfully applied in different areas (Floreano, Dürr, & Mattiussi, 2008), and many were the proposals to use Evolutionary Computing (EC) to optimize ANNs. In fact, it is possible to split neuroevolution's progress in three stages:

1. The use of EC to train an ANN;

2. The use of EC to optimize the architecture underneath an ANN;

3. The use of EC to optimize the topology of an ANN and train it;

Pioneers in the field appeared in the late 80's: Davis, 1989 proved it was possible to map Classifier Systems (CSs) – systems that incorporate Genetic Algorithms (GAs)

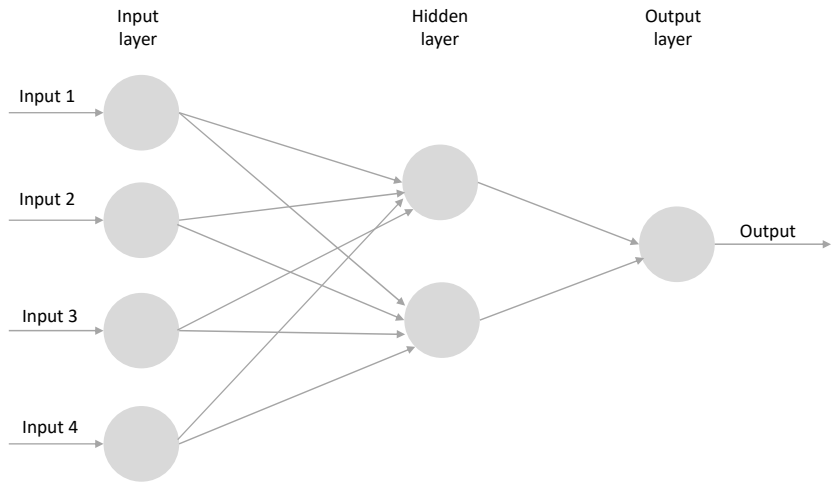

Figure 2.1: An example of an Artificial Neural Network

as the learning mechanism – into Neural Networks and vice versa. Whitley, 1989 attempted to train a Neural Network making use of Genetic Algorithms, however, at this point, researchers would still have to decide on the ANN's architecture, only allowing the evolution to generate new weights based on the network's performance. This meant that the two systems would work as separate components, thus, from the moment that it was generated, the neural network would not learn anything further during its existence. This technique was then established as fixed-topology neuroevolution in the literature (Montana & Davis, 1989). At this time, researchers thought this could be a valid alternative to the most traditional ANN training algorithm called backpropagation (Miller, Todd, & Hegde, 1989; Montana & Davis, 1989; Whitley & Hanson, 1989) which, due to how gradient descent works, can easily get trapped in a local minimum of the error function and it is not able to find a global minimum if the error function is multimodal and/or non-differentiable (Xin Yao, 1999).

Miller et al., 1989 were able to use a genetic algorithm for evolving neural network architectures for specific problems. Each topology of an ANN is represented as "a connection constraint matrix mapped directly into a bit-string genotype" (Miller et al., 1989). Modified standard genetic operators are applied upon populations of these genotypes to generate topologies with better fitness in consecutive generations. Topology fitness is evaluated through training the neural network and keeping track of the

final performance error. Miller et al., 1989's study is different from the previous ones in the sense that in this scenario the EC techniques are used to evolve the topology of the network. Evolving the architecture of a network can be established as an optimization problem where the objective is to find the global optimum in a search space where each point represents a network architecture. This search space (also known as fitness landscape or surface) shows some properties that make it suitable for EC techniques to search for the most appropriate architecture to represent an ANN (Miller et al., 1989). These properties were unveiled and discussed in Miller et al., 1989 as follows:

- The search space is infinitely large since the number of possible neurons and connections is unbounded;
- The fitness landscape is undifferentiable because changes in the number of units and connections must be discrete and can discontinuously affect the performance of the network. This aspect makes it impossible for gradient-based methods to be used;
- The mapping between the architecture of the network and its performance after the learning phase is indirect, extremely epistatic and dependent on the very initial conditions like the random original weights. Consequently, the surface is complex and noisy;
- Finally, networks with similar architectures can behave differently and vice versa (different architectures may show a similar behavior). Thus, the search space is deceptive and multimodal.

These aspects make EC techniques suitable to address the task of automatically finding an optimal network topology, proving to be a reliable asset and a competitive alternative to constructive and destructive algorithms. On the one hand, a constructive algorithm (Fahlman & Lebiere, 1990; Frean, 1990) is a hill climbing approach which starts with an ANN with minimum number of neurons, hidden layers and connections and adds new elements throughout the learning phase, when necessary and based on some criteria. On the other hand, destructive algorithms (Cun, Denker, & Solla, 1990; Mozer & Smolensky, 1989; Sietsma & Dow, 1991) start their search for the optimal topology with an ANN with the maximum number of units, hidden layers and connections, and while learning, unnecessary elements are removed (Xin Yao, 1999). These methods may seem simpler to implement when compared with EC-based methods, however, they are prone to get stuck in local optima architectures and are only able to explore a small part of the possible ANN topologies. Schaffer, Caruana, and Eshelman, 1990 illustrate how a genetic algorithm can be used to exploit the properties of backpropagation to solve difficult tasks. Their results show that networks evolved through a genetic algorithm perform better than a large network using the backpropagation learning method alone. Wilson, 1990's results demonstrated that

if genetic search is applied to a perceptron, it can learn more complex tasks than it would be initially thought. Schiffmann, Joost, and Werner, 1992 started by exploring evolution strategies which used only mutation to change the parents' topologies in a GA population. Such research was followed by the creation of a crossover operator for an automatic topology optimization genetic algorithm. Their results confirmed that allowing two parent networks with different number of neurons to mate and generate a child network which inherits their genes outperforms the fixed topologies and reaches classification performances close to optimal values. Along a similar line of thought, Alba, Aldana, and Troya, 1993 accomplished a full genetic ANN design by making use of a genetic algorithm to address the connectivity and structure definition problems.

On the one hand, EC techniques have been shown to be used in the optimization of neural networks' weights assuming that their topologies are fixed and defined from the beginning. On the other hand, EC-based methods have been applied in the evolution of ANNs' architectures considering that their activation functions are defined a priori and static throughout the whole process. Some studies revealed how choosing these activation functions is also important when measuring the performance of a neural network (Dasgupta & Schnitger, 1992; Mani, 1990). Schoenauer and Ronald, 1997 proposed a method to evolve both topologies and activation functions, showing how tuning the slopes of these functions for each processing unit in the NN improves its overall performance. To achieve the same effect, White and Ligomenides, 1993 presented a less complex method where 80% of the neurons in the initial population used the Sigmoid function whereas the remaining 20% used a Gaussian function. The learning phase, through the evolutionary process, was used to automatically define the most suitable combination for these percentages. Optimizing activation functions is currently under the investigation radar also due to the popularity of deep learning. The Rectified Linear Activation (ReLU) function (Jarrett, Kavukcuoglu, Ranzato, and LeCun, 2009) has been widely used to simplify the training of deep neural networks since it overcomes issues like weight initialization and vanishing gradient. Manessi and Rozza, 2018 summarized the different variations of ReLU proposed throughout the years:

- Leaky ReLU (LReLU) (Maas, 2013), which covers the dead neurons issue in ReLU networks;
- Threshold ReLU (Konda, Memisevic, and Krueger, 2014), which integrates a solution for problems like the large negative bias in autoencoders;
- Parametric ReLU (PReLU) (He, Zhang, Ren, and Sun, 2015), which uses the leakage parameter of LReLU as a per-filter learnable weight.

The authors introduced two methods that would automatically learn several combinations with different base activation functions, such as the identity function, ReLU

and the hyperbolic tangent. They thoroughly compared their two approaches to common architectures in standard data sets, showing relevant improvements in the general performance of the NN. While these studies introduced new and useful activation functions, other works had proposed advanced strategies to learn the best activation function for the particular topology at hand.

Agostinelli, Hoffman, Sadowski, and Baldi, 2014 came up with a new design for a piecewise linear activation function which is learned independently for each neuron using gradient descent. Using this adaptive activation function, the authors were able to improve deep neural network architectures composed of static ReLU units and achieve state-of-the-art performances on CIFAR-10, CIFAR-100 – data sets which contain several images in 10 and 100 classes, respectively –, as well as on a benchmark involving Higgs boson decay modes. These studies showed how evolving activation functions is, nowadays, seen as something so important as evolving the topologies of the neural networks (Manessi & Rozza, 2018).

Still during the 90's, Angeline, Saunders, and Pollack, 1994; Branke, 1995; Gruau, Whitley, and Pyeatt, 1996 and Xin Yao, 1999 showed that it was possible to go the next step and evolve network topologies as well as their weights in what was called Topology and Weight Evolving Artificial Neural Networks (TWEANNs). In particular, Angeline et al., 1994 argued that crossover was not well-suited for evolving the network topology. Instead, they proposed a method where offspring are solely created by mutation. Until this point, connection weights had to be learned in a subsequent step – this method reduces the complexity of evolving both the architecture and the weights, however, there are two main issues in doing so as shown by Yao and Liu, 1997:

1. Different and random initial weights can generate different training results, that is, the same genotype can lead to different fitness results due to the random weight initialization;

2. Different training algorithms may lead to different training values even when using the same set of initial weights. This is especially true for multimodal error functions. The idea behind the method that optimizes, at the same time, weights and architectures for the neural networks is that each individual in a population is a fully specified neural network with complete weight information. Consequently, there is a one-to-one mapping between the genotype and the phenotype, which allows the search process to overcome the already mentioned issues regarding fitness evaluation.

Srinivas and Patnaik, 1991 developed an approach to reduce the search space of a genetic algorithm to improve its performance in finding the optimal set of connection weights. The authors used the equivalent solutions in the search space, and from each set of equivalent solutions, they took one solution, called the base solution, to feature

in the reduce search space. They added an extra step to the algorithm where the solutions are mapped to their correspondent base solutions.
Bornholdt and Graudenz, 1992 presented a GA-based method used to evolve a network that represented a model for a brain with sensory and motor neurons. Oliker, Furst, and Maimon, 1993 showed that it was possible to have a distributed genetic algorithm to define and train neural networks. The approach establishes the neural network's topology and its weights for a specific task where the network is composed of binary linear threshold units.
White and Ligomenides, 1993 developed a novel algorithm which uses a genetic algorithm to define the topology and weights of a neural network. If the genetic algorithm fails to find a network as a solution, the best network to be developed until that moment is used to try to find a solution through backpropagation. This way, each algorithm is used to exploit its best advantage: the genetic algorithm, through its global search, defines the architecture and suboptimal weights to solve the task and backpropagation uses its local search to pursue the best neighbor of the architecture and weight structure found by the GA. There were other EC methods used to approach this optimization problem. Koza and Rice, 1991 showed that it was possible to use Genetic Programming to find both the weights and the topology of a neural network, including the number of layers, number of neurons per layer and how they are connected. Jian and Yugeng, 1997 proposed a novel approach to define the structure and weights of ANNs based on Evolutionary Programming. The Particle Swarm Optimization (PSO) algorithm was also considered to evolve both weights and topologies of neural networks (Chunkai Zhang, Huihe Shao, & Yu Li, 2000; Garro & Vázquez, 2015; Kiranyaz, Ince, Yildirim, & Gabbouj, 2009). In the same line of thought, Kiranyaz et al., 2009 proposed an extension of the PSO algorithm to a Multi-Dimensional Particle Swarm Optimization (MD-PSO) algorithm in a way that it was able to automatically design the ANNs while evolving the optimal network configuration (connections, weights and biases) within the architecture space. Garro and Vázquez, 2015 explored the simultaneous evolution of the three main components of a neural network: the set of synaptic weights, the connections, and the activation function of each neuron. The key value of this contribution was the assessment of eight different fitness functions used to verify the quality of each solution and find the best network design. Chunkai Zhang et al., 2000 introduced a new evolutionary system to build Feed-Forward ANNs which is restricted to PSO where both elements of the neural network – architecture and weights – are adaptively adjusted according to the quality of the network.
At the beginning of the twenty-first century, K. O. Stanley and Miikkulainen, 2002 converged their ideas to some of the limitations they saw in topology representations into a novel neuroevolution algorithm called NeuroEvolution of Augmenting Topologies (NEAT). In particular, one of the main challenges they found concerned something called *Competing Conventions* which essentially meant that multiple different genotypes decode into the same phenotype which could have a serious negative impact

on the algorithm. In that sense, NEAT's ability to evolve increasingly complex ANNs, whilst setting aside traditional TWEANNs' restrictions, allowed it to become the most popular and broadly used approach in the neuroevolution field. More recently, NEAT has been evolved to CoDeepNEAT (Miikkulainen et al., 2017) which is able to cover more complex fields such as vision, speech and language.
For further details about this investigation field, the reader is referred to (Ding, Li, Su, Yu, & Jin, 2013; K. Stanley, Clune, Lehman, & Miikkulainen, 2019; Xin Yao, 1999).

Chapter 3

Semantic Learning Machine

3.1 Algorithm

The Semantic Learning Machine (Gonçalves, 2017; Gonçalves et al., 2015b) relies on the Geometric Semantic Genetic Programming (GSGP)'s mutation operator defined by Moraglio, Krawiec, and Johnson, 2012 (Definition 3.1.1). In this work, Moraglio et al., 2012 proposed a new approach - GSGP - where traditional crossover and mutation are replaced by geometric semantic operators. These new operators run directly in the space of the underlying semantics (outputs) of the individuals which induces a unimodal error surface for any supervised learning problem (Gonçalves et al., 2015b). This means that the SLM shares the same semantic landscape properties as GSGP which contains no local optima.

Definition 3.1.1. Geometric Semantic Mutation (GSM)

$$T_M = T + ms \bullet (T_R1 - T_R2)$$

Where T is a parent function $T : \mathbb{R}^n \rightarrow \mathbb{R}$, T_R1 and T_R2 are random real functions in the codomain [0,1] and ms is a parameter called mutation step, responsible for the degree of change on the semantics.

The Semantic Learning Machine, by inheriting GSGP's unimodal error landscape, assures that the search is performed on a surface with no local optima. This means that it is possible for the SLM algorithm to be a Geometric Semantic Hill Climber (GSHC) for feedforward Neural Networks: the search process revolves around one individual (the best one) and, through the mutation operator, a sample of Neural Networks is produced at each generation and from those, the best one is selected replacing the current best if it is better than it. Figure 3.1 shows the application of said mutation operator. The SLM algorithm can be summarized in the following steps:

1. Generate N initial random NNs;
2. Choose the best NN (B) from the initial random NNs, according to the selected performance criterion;
3. Repeat the following steps until a given stopping criterion is met:
   a) Apply the geometric semantic mutation to the current best (B) N times to generate N new NNs (known as children or neighbors);
   b) Update B as being the NN with the best performance according to the selected criterion, considering the current B and the N newly generated NNs;
4. Return B as the best performing NN according to the selected performance criterion.

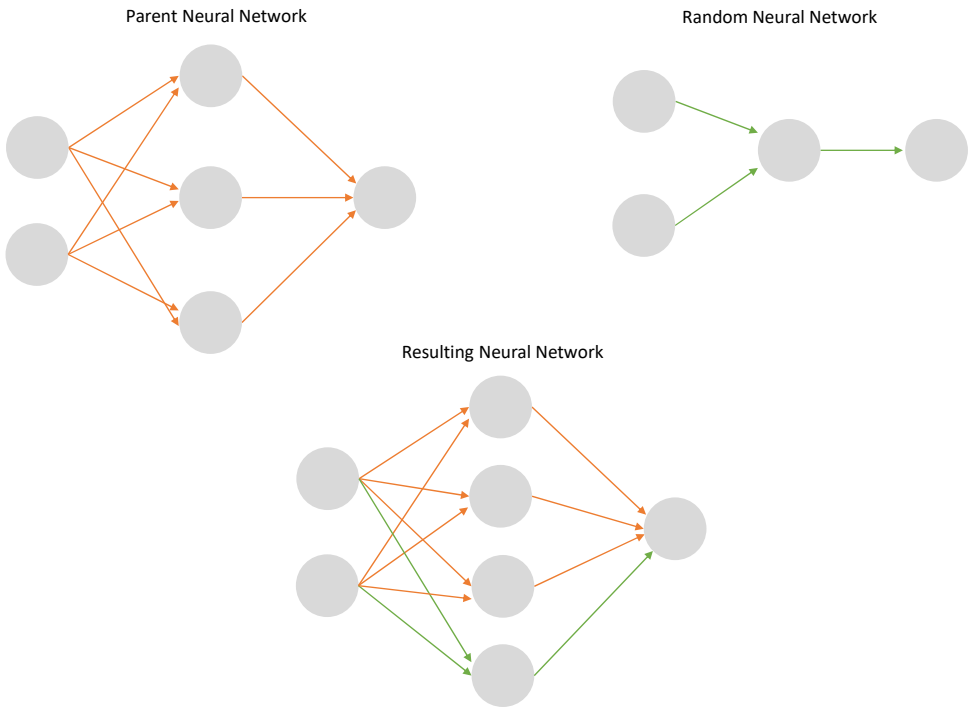

Figure 3.1: An example of an application of the GSM operator Gonçalves, 2017

The initial random neural networks can have any number of layers and neurons. Both the activation functions and weights in the connections between the neurons can be freely selected. Just like many other Neuroevolution algorithms, the SLM does not use backpropagation to adjust the weights of the neural networks. The key point of this algorithm is the definition of the geometric semantic mutation which adds new neurons to the already existent ones in the hidden layers. Each new neuron

can choose from which already available unit will receive a connection. This implies that the neural network does not need to be fully-connected and its sparseness can be controlled by establishing the number of incoming connections each neuron will receive considering the set of all possible incoming connections. Like in the initial step, the weights of each connection can be freely chosen. One of the main aspects about this mutation is the fact that new processing units do not feed their computations to existing neurons with the exception of the output neuron.

More recently, Gonçalves, Silva, Fonseca, and Castelli, 2017 developed two stopping criteria for SLM based on the information gathered on the semantic neighborhood (the set of new models generated after the mutation): Error Deviation Variation (EDV) and Training Improvement Effectiveness (TIE). The EDV stopping criterion measures the percentage of solutions that reduce the error deviation (sample standard deviation of the absolute errors of an individual over the training instances) in comparison with the error deviation of the current best model, within the individuals that are better than the current best. The TIE criterion measures the effectiveness of the mutation operator. Within the sample of generated individuals, it gives the percentage of solutions which are better than the current best. In both criteria, if the percentages are smaller than a given threshold (parameter), the search process stops, avoiding overfitting and contributing to a more computationally efficient algorithm. Furthermore, in Gonçalves et al., 2017, an Optimal Learning Step (OLS), calculated at each application of the mutation operator and based on the Moore-Penrose inverse, was tested rather than relying only on a Fixed Learning Step (FLS). This OLS computation was adapted from the optimal mutation step computation used in GSGP proposed by Gonçalves, Silva, and Fonseca, 2015a.

3.2 Previous Comparisons with Other Neuroevolution Methods

Combining the aforementioned factors with its original characteristics makes the SLM a promising and competitive algorithm when compared with the popular NEAT algorithm, fixed-topology neuroevolution approaches and other matured ML algorithms like Multilayer Perceptron (MLP) and Support Vector Machine (SVM) (Jagusch et al., 2018). This experiment was performed on nine real world free data sets (five classification and four regression data sets), taken from the UCI Machine Learning Repository (Lichman, 2013). Results showed that, in terms of learning, the SLM was superior, with statistically significant differences, in comparison to the other neuroevolution methods in all the data sets considered. In this work, the best SLM variant was, unsurprisingly, the one generated with the optimal learning step. Focusing specifically on the comparison with NEAT and in the generalization performance, in eight of the nine tasks, the Semantic Learning Machine was able to achieve statistically significant

differences. In the other data set, no statistically significant difference was found. Besides this, the SLM variant generated with the OLS and with the Semantic Stopping Criterion EDV achieved smaller neural networks and was able to reach speed-ups of various orders of magnitude over NEAT on several data sets.

Chapter 4

Experimental Methodology

4.1 Data sets

In this experimental study, fifteen real-world binary classification data sets were considered. These were pulled from the Penn Machine Learning Benchmark (PMLB) (Orzechowski, Cava, & Moore, 2018) repository, which contains a large collection of standardized data sets for classification and regression problems. In this work only classification tasks were considered. A more detailed description of the data sets as well as the objective for each one of them can be found in Table A.1. Table 4.1 and Figure 4.1 present the number of features (input variables), the number of instances (observations), and the % of class 1 instances in each of the thirteen problems under study.

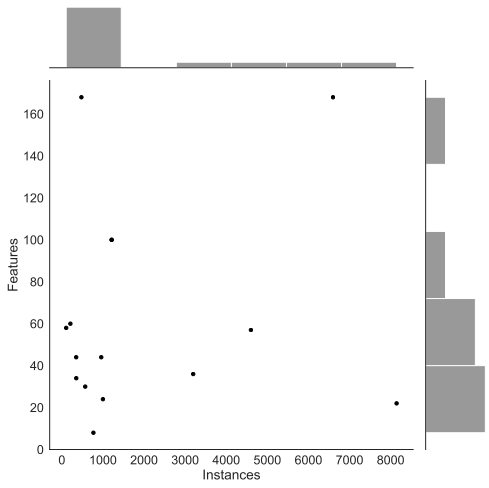

Figure 4.1: Characteristics of the binary classification data sets considered

It is of relevance to mention that some of the data sets, such as *clean1* and *clean2*

Data set	Features	Instances	% of class 1 instances
agaricus-lepiota	22	8145	≈ 48%
breast-cancer-wisconsin	30	569	≈ 37%
clean1	168	476	≈ 43%
clean2	168	6598	≈ 14%
credit-g	24	1000	≈ 64%
diabetes	8	768	≈ 35%
hill-valley-with-noise	100	1212	≈ 50%
hill-valley-without-noise	100	1212	≈ 50%
sonar	60	208	≈ 47%
ionosphere	34	351	≈ 64%
kr-vs-kp	36	3196	≈ 52%
molecular-biology-promoters	58	106	≈ 50%
spambase	57	4601	≈ 39%
spectf	44	349	≈ 73%
tokyo1	44	959	≈ 64%

Table 4.1: Binary classification data sets considered

and *hill-valley-with-noise* and *hill-valley-without-noise* contain the same kind of data, however, differ in terms of number of instances and noise, respectively, which pose different challenges to the algorithms under test.

4.2 Methods and Parameter Tuning

The base configuration for the SLM is the following:

- In the initial population, each NN is generated with a random number of hidden layers selected between 1 and 5;
- In the initial population, each NN contained in each hidden layer is generated with a random number of neurons selected between 1 and 5;
- Each hidden neuron randomly selects its activation function from the following options: Logistic, Relu, and Tanh;
- Each hidden neuron randomly selects the weight of each incoming connection from values within [-*mncw*, *mncw*], where *mncw* represents the maximum neuron connection weight parameter (subject to parameter tuning);
- Each hidden neuron randomly selects the weight of its bias from values in the range [-*mbw*, *mbw*], where *mbw* represents the maximum bias weight parameter (also subject to parameter tuning);
- Each time a new NN is created by the mutation operator, the number of new neurons to be added to each layer is randomly selected between 1 and 3.

The main differences between the SLM variants under study are the following:

- The strategy to choose the learning step where two options are considered:
  - Calculate the Optimal Learning Step (OLS), previously mentioned, for each application of the mutation operator;
  - Use a Bounded Learning Step (BLS) where there is an additional parameter which establishes the maximum learning step (*mls*) restricting the learning step. At each application of the mutation operator, the effective learning step is randomly assigned a value from the range [-*mls*, *mls*].
- The type of dynamic selection of training examples (if applicable) in which two methods are taken into account:
  - Random selection from a subset of data at each iteration and calculate the performance of each solution with this new subset, denominated Random Sampling Technique (RST) following (Gonçalves & Silva, 2013; Gonçalves, Silva, Melo, & Carreiras, 2012) and based on (Y. Liu & Khoshgoftaar, 2004);
  - Use the full data set but choose weights, between 0 and 1, for each record and change these weights at each iteration. In this work, this process is referred as Random Weighting Technique (RWT).
- The stopping criterion to decide the ending of the learning process where two approaches are considered:
  - Ending based on a fixed number of iterations;
  - Ending based on EDV or TIE, two Semantic Stopping Criteria introduced in (Gonçalves et al., 2017) and previously explained.

Taking the aforementioned aspects into account, the SLM variants are grouped and denominated in the following way:

1. BLS variants: SLM-BLS, SLM-BLS + RST, and SLM-BLS + RWT
2. OLS variants: SLM-OLS, SLM-OLS + RST, and SLM-OLS + RWT
3. BLS + TIE/EDV: SLM-BLS + TIE/EDV
4. OLS + EDV: SLM-OLS + EDV

Whenever the SLM-BLS or SLM-OLS methods are mentioned by themselves it means that the variants which do not use either of the sampling/weighting techniques are being referred. In this experiment it is followed a K-Fold Cross Validation (CV) methodology where a 30-fold outer CV is used to obtain 30 final generalization values (test set values) to perform statistical validation for the methods under comparison.

For each of the outer training fold, a 2-fold inner CV is used to apply parameter tuning in each method.

In turn, the Multilayer Perceptron (MLP), trained with backpropagation is also evaluated in two different variants: the most common Stochastic Gradient Descent (SGD) (Kiefer & Wolfowitz, 1952; Nikolopoulos & Fellrath, 1994), and the Adaptive Moment Estimation (Adam) SGD variant (Kingma & Ba, 2014). Both algorithms are allowed to test 72 random parameter combinations in the inner CV. The SLM tests 18 parameter combinations for each of the groups considered while the MLP tests 36 parameter configurations for each of the two variants taken into account.

All the SLM variants can tune the maximum neuron connection weight (*mncw*) and the maximum bias weight (*mbw*) in the interval [0.1, 0.5]. The BLS variants and the BLS + TIE/EDV can tune the maximum learning step (*mls*) in the range [0.1 and 2], and the number of iterations in the range [1, 100]. The BLS and OLS variants are allowed to select with equal probability the use of RST, RWT or none. BLS + TIE/EDV selects with equal probability the use of EDV or TIE as the Semantic Stopping Criterion. Whenever RST is used, the parameter that defines the ratio of the total training data to be considered – the subset ratio – is chosen from the interval [0.01, 0.99].

For SGD and Adam, the following parameters are tuned:

- The number of iterations in the range [1, 100]
- The batch size between 50 and the maximum number of training instances available
- The activation function to be used in the hidden layers: Logistic Relu, and Tanh
- The number of hidden layers in the range [1, 5]
- The number of hidden neurons per layer in the range [1, 200]
- The learning rate in the range [0.1, 2]
- The L2 penalty in the range [0.1, 10]

SGD can also select the momentum in the interval [0.0000001, 1] and decide to use or not the Nesterov's momentum. Adam can also select the beta 1 and beta 2 parameters in the range [0, 1[.

Chapter 5

Results and Analysis

This chapter analyses the results obtained in the experimental step. Section 5.1 presents the results reached by the different SLM variants, analyzing the performance achieved on the validation set and discussing some aspects related to the parameter selection. Afterwards, Section 5.2 shows the results obtained by the MLP variants in the same considered tasks and discusses the key differences between the two algorithms considered. Section 5.3 compares SLM and MLP after the best configuration was found and, finally, Section 5.4 digs deeper in the generalization performance of the SLM under different ensemble construction methods.

5.1 SLM Variants

This section presents the results obtained when considering the different variants in the SLM group. The discussion starts off by delving in the validation Area Under Receiver Operating Characteristic (AUROC) curve values generated by the SLM variants under study. The results can be found in Table 5.1. For each task and for each technique, the table reports the mean and standard deviation of the validation AUROC generated by the best model in each inner cross-validation process, which was performed to determine the best set of hyperparameters. Table 5.2 suggests that the OLS variants achieved the best results in every task considered. In fact, in 9 out of the 15 considered problems, the OLS variant was always the selected one. Only in one of the problems – *molecular-biology-promoters* – it is possible to find a slightly more varied distribution of the selected variants: three for BLS, five for BLS + TIE/EDV, six for OLS + EDV and sixteen for OLS, the lowest value registered for this variant. BLS + TIE/EDV achieved the weakest results in every data set amongst the considered variants. A reasonable explanation could be that the number of iterations is not

Data set	BLS + TIE/EDV	OLS variants	BLS variants	OLS + EDV
agaricus-lepiota	0.790 +- 0.048	0.966 +- 0.007	0.894 +- 0.022	0.869 +- 0.014
breast-cancer-wisconsin	0.839 +- 0.052	0.940 +- 0.006	0.910 +- 0.010	0.892 +- 0.018
clean1	0.632 +- 0.047	0.788 +- 0.016	0.704 +- 0.035	0.711 +- 0.032
clean2	0.848 +- 0.008	0.937 +- 0.005	0.867 +- 0.015	0.878 +- 0.011
credit-g	0.704 +- 0.003	0.727 +- 0.006	0.717 +- 0.006	0.715 +- 0.004
diabetes	0.720 +- 0.017	0.776 +- 0.006	0.768 +- 0.006	0.767 +- 0.007
hill-valley-without-noise	0.636 +- 0.106	0.934 +- 0.020	0.733 +- 0.054	0.757 +- 0.034
hill-valley-with-noise	0.609 +- 0.074	0.836 +- 0.024	0.677 +- 0.056	0.727 +- 0.017
ionosphere	0.766 +- 0.035	0.909 +- 0.010	0.871 +- 0.017	0.848 +- 0.012
kr-vs-kp	0.668 +- 0.049	0.946 +- 0.004	0.873 +- 0.032	0.873 +- 0.028
molecular-biology-promoters	0.850 +- 0.052	0.909 +- 0.022	0.870 +- 0.037	0.901 +- 0.026
sonar	0.634 +- 0.031	0.777 +- 0.026	0.734 +- 0.037	0.723 +- 0.031
spambase	0.770 +- 0.030	0.919 +- 0.004	0.851 +- 0.021	0.856 +- 0.019
spectf	0.728 +- 0.002	0.797 +- 0.018	0.751 +- 0.019	0.738 +- 0.010
tokyo1	0.772 +- 0.040	0.914 +- 0.004	0.881 +- 0.017	0.866 +- 0.012

Table 5.1: Validation AUROC for each SLM variant considered

enough for the Semantic Stopping Criterion to have an impact under the utilization of a Bounded Learning Step on the considered problems. In general, the OLS group (OLS variants and OLS + EDV) was able to achieve a higher AUROC in comparison with the BLS group (BLS variants and BLS + TIE/EDV). The results can be summed up in the following way:

1. Amongst the BLS group, BLS + TIE/EDV always achieved a lower value than the BLS variants;

2. Amongst the OLS group, the OLS variants were always the best performer regardless of the considered benchmark;

3. The OLS variants seem like the most suitable choice for the classification tasks considered.

In particular, for the *hill-valley* data sets, results suggest that all the variants are able to perform better on the problem without noise. As for the *clean* benchmarks, it looks like the SLM is negatively impacted by having less observations to train the model: the SLM achieves better results in *clean2* which contains more instances and is less imbalanced.

The subsequent analysis considers the mean value of iterations obtained for each SLM variant and is based on Table 5.3. In 9 out of the 15 tasks considered, the OLS variants which were previously considered the most suitable choice, present a higher number of iterations. In the six remaining benchmark problems, the BLS variants take superior values, which is expected since these do not use any Semantic Stopping Criterion or Optimal Learning Step. These two groups of variants present a number

Data set	BLS variants	OLS variants	BLS + TIE/EDV	OLS + EDV
agaricus-lepiota	0	30	0	0
breast-cancer-wisconsin	0	30	0	0
clean1	1	29	0	0
clean2	0	30	0	0
credit-g	4	26	0	0
diabetes	2	23	0	5
hill-valley-without-noise	0	30	0	0
hill-valley-with-noise	0	30	0	0
ionosphere	0	30	0	0
kr-vs-kp	0	30	0	0
molecular-biology-promoters	3	16	5	6
sonar	1	26	0	3
spambase	0	30	0	0
spectf	2	28	0	0
tokyo1	0	30	0	0

Table 5.2: Best SLM configuration by variant

Data set	BLS + TIE/EDV	OLS variants	BLS variants	OLS + EDV
agaricus-lepiota	50.500 +- 129.126	419.167 +- 59.887	369.667 +- 95.644	3.567 +- 3.126
breast-cancer-wisconsin	23.967 +- 79.003	330.200 +- 94.672	375.967 +- 92.992	1.167 +- 0.531
clean1	15.000 +- 34.339	350.033 +- 82.854	325.700 +- 110.046	8.967 +- 29.432
clean2	3.933 +- 4.828	424.767 +- 46.555	335.467 +- 129.398	5.233 +- 6.383
credit-g	3.767 +- 5.164	313.900 +- 123.399	301.333 +- 137.811	2.367 +- 0.850
diabetes	7.700 +- 26.761	285.133 +- 142.049	330.033 +- 111.033	1.467 +- 0.973
hill-valley-without-noise	13.800 +- 21.335	413.867 +- 65.554	351.400 +- 97.820	7.800 +- 5.845
hill-valley-with-noise	15.900 +- 19.361	405.533 +- 84.233	344.733 +- 99.606	26.933 +- 24.669
ionosphere	23.333 +- 91.975	333.833 +- 93.222	354.000 +- 108.958	2.533 +- 1.889
kr-vs-kp	20.533 +- 90.701	411.400 +- 71.944	376.800 +- 88.220	7.333 +- 5.054
molecular-biology-promoters	12.200 +- 18.817	278.567 +- 131.946	313.933 +- 132.128	2.533 +- 1.737
sonar	3.133 +- 2.569	339.400 +- 105.707	373.900 +- 106.740	2.800 +- 4.046
spambase	14.600 +- 55.260	412.533 +- 76.270	343.000 +- 109.710	2.767 +- 2.635
spectf	1.833 +- 1.392	355.567 +- 102.266	293.700 +- 134.954	2.433 +- 1.006
tokyo1	9.667 +- 34.948	353.933 +- 92.118	363.100 +- 114.439	3.333 +- 4.444

Table 5.3: Number of iterations for each SLM variant considered

of iterations significantly larger when compared with the other groups. Specifically, laying the OLS variants next to the OLS + EDV variant, it is possible to verify that the number of iterations of the latter is way lower. Using the Optimal Learning Step allows OLS + EDV to achieve a reasonably good performance in every benchmark even if it never surpasses the winner OLS variant. To sum up, the results suggest that using a Semantic Stopping Criterion has a positive impact when it comes to decreasing the computational effort in the training phase, however, it might not be enough to obtain the best overall performance.

Focusing specifically on the different Semantic Stopping Criteria, Table 5.4 shows

Dataset	EDV	TIE
agaricus-lepiota	22	8
breast-cancer-wisconsin	18	12
clean1	21	9
clean2	18	12
credit-g	17	13
diabetes	23	7
hill-valley-without-noise	22	8
hill-valley-with-noise	25	5
ionosphere	19	11
kr-vs-kp	23	7
molecular-biology-promoters	17	13
sonar	21	9
spambase	19	11
spectf	15	15
tokyo1	17	13

Table 5.4: EDV and TIE use in SLM-BLS

the distribution of using the EDV and TIE techniques in the SLM BLS variant. According to the presented values, we can clearly conclude that the EDV strategy is more efficient than the TIE one in the benchmarks considered. In 14 out of the 15 data sets, EDV was superior. In the remaining one – *spectf* – the results are even.
A further analysis takes into account the impact of using the random weighting/sampling techniques when combined with the SLM variants. Results of this analysis are shown in Table 5.5 where the distribution of using RST, RWT, and the complete data set (None) are compared in the context of the BLS and OLS variants. For the OLS variants, the table suggests that using the complete training set is actually more effective than relying on one of the weighting/sampling techniques. Only in two of the problems considered - *clean2* and *molecular-biology-promoters* - the RWT is the favorite selection. In two other benchmarks - *ionosphere* and *tokyo1* - the distribution between using the full set of observations and RWT is even. For the OLS variants, RST does not seem like an effective option as it is always selected the least number of times. In the BLS variants, the results are slightly more varied: using the complete data set is preferred in six of the considered tasks, RWT is selected in five of the problems and RST the favorite in two of the benchmarks.

5.2 MLP Variants

This section presents and discusses the results for the MLP variants considered: Adam and SGD. The first part of the analysis is based on Table 5.6 which shows the performance of the models achieved on the validation set. The results suggest that Adam is the better performer in 7 out of the 15 considered data sets. For the remaining two

Dataset	OLS variants			BLS variants		
	None	RST	RWT	None	RST	RWT
agaricus-lepiota	14	10	6	9	8	13
breast-cancer-wisconsin	16	2	12	6	9	15
clean1	14	3	13	18	2	10
clean2	10	7	13	11	14	5
credit-g	17	4	9	13	9	8
diabetes	16	7	7	14	8	8
hill-valley-without-noise	14	8	8	9	9	12
hill-valley-with-noise	21	4	5	7	11	12
ionosphere	12	6	12	9	12	9
kr-vs-kp	18	5	7	7	8	15
molecular-biology-promoters	12	4	14	15	3	12
sonar	15	6	9	15	4	11
spambase	18	6	6	7	11	12
spectf	16	2	12	17	6	7
tokyo1	14	2	14	13	10	7

Table 5.5: RST and RWT use in the BLS and the OLS variants

- *credit-g* and *spectf* – the value was the same for both variants. The choice between one variant or the other seems to be balanced and the values suggest that tending to one or the other might be problem dependent. In addition to these results, Table 5.7 presents the best MLP configuration per variant where it is possible to verify that out of the 15 problems considered, Adam outperforms SGD in 9 of them. In fact, for the *spectf* data set, the SGD variant is never selected. The opposite perspective can also be found in the *diabetes* data set where Adam was never selected. At this point, it is of extreme importance to compare the results present in Table 5.6 (obtained with MLP) with the ones available in Table 5.1 (obtained with SLM). Considering these results, the SLM variants are capable of outperforming the best MLP variant in 11 out of the 15 classification tasks under study - for *spectf*, the weakest AUROC value is the same in both algorithms. This comparison clearly shows how superior the Semantic Learning Machine is in creating models with a better validation AUROC when compared to MLP. In general, the SLM is a competitive option to take into account in these classification tasks since its performance is significantly better than the best MLP variant in more than 70% of the benchmarks considered.

When comparing these two methods it is also of major relevance to take into account the number of iterations necessary to obtain the final solution. Table 5.8 shows the number of iterations for each MLP variant considered. As seen previously, the SLM was able to generate the final classification model in a significantly low number of iterations when making use of a Semantic Stopping Criterion. These values are considerably lower than the ones achieved by Adam and SGD which are based on the backpropagation algorithm. This lower number of iterations does not seem to have a negative impact on the performance of the final solutions as these SLM variants are

Data set	Adam	SGD
agaricus-lepiota	0.669 +- 0.125	0.857 +- 0.112
breast-cancer-wisconsin	0.690 +- 0.081	0.631 +- 0.014
clean1	0.615 +- 0.053	0.581 +- 0.035
clean2	0.855 +- 0.020	0.852 +- 0.016
credit-g	0.700 +- 0.002	0.700 +- 0.001
diabetes	0.681 +- 0.034	0.762 +- 0.021
hill-valley-without-noise	0.513 +- 0.009	0.509 +- 0.007
hill-valley-with-noise	0.512 +- 0.011	0.503 +- 0.004
ionosphere	0.789 +- 0.060	0.827 +- 0.064
kr-vs-kp	0.678 +- 0.158	0.879 +- 0.133
molecular-biology-promoters	0.737 +- 0.088	0.632 +- 0.116
sonar	0.550 +- 0.018	0.579 +- 0.062
spambase	0.663 +- 0.055	0.621 +- 0.036
spectf	0.728 +- 0.001	0.728 +- 0.001
tokyo1	0.651 +- 0.018	0.690 +- 0.058

Table 5.6: Validation AUROC for each MLP variant considered

Data set	Adam	SGD
agaricus-lepiota	4	26
breast-cancer-wisconsin	25	5
clean1	27	3
clean2	19	11
credit-g	29	1
diabetes	0	30
hill-valley-without-noise	25	5
hill-valley-with-noise	23	7
ionosphere	10	20
kr-vs-kp	7	23
molecular-biology-promoters	27	3
sonar	16	14
spambase	24	6
spectf	30	0
tokyo1	14	16

Table 5.7: Best MLP configuration by variant

Data set	Adam	SGD
agaricus-lepiota	240.633 +- 157.372	292.667 +- 134.730
breast-cancer-wisconsin	270.600 +- 133.376	230.633 +- 175.833
clean1	311.033 +- 138.915	212.600 +- 145.270
clean2	271.833 +- 156.325	238.267 +- 149.608
credit-g	218.200 +- 137.295	273.267 +- 140.189
diabetes	213.033 +- 135.064	230.133 +- 118.258
hill-valley-without-noise	231.333 +- 153.482	255.733 +- 136.781
hill-valley-with-noise	269.900 +- 137.863	272.500 +- 140.118
ionosphere	203.100 +- 148.262	276.767 +- 116.114
kr-vs-kp	259.933 +- 157.215	281.767 +- 123.694
molecular-biology-promoters	235.733 +- 154.893	201.667 +- 103.557
sonar	254.500 +- 142.140	240.467 +- 134.315
spambase	243.733 +- 126.503	286.033 +- 128.667
spectf	301.300 +- 117.290	298.967 +- 140.357
tokyo1	226.600 +- 151.842	277.467 +- 130.240

Table 5.8: Number of iterations for each MLP variant considered

able to outperform MLP in 10 out of the 15 considered benchmarks: OLS + EDV is only outperformed by MLP in one data set (*kr-vs-kp*) and BLS + TIE/EDV is worse than MLP in five tasks (*agaricus-lepiota*, *clean2*, *diabetes*, *ionosphere* and *kr-vs-kp*). When comparing the best SLM performer with both MLP variants it is possible to conclude that the OLS variants need more iterations to achieve a better validation AUROC value. To better understand the main differences between Adam and SGD, Table 5.9 presents how the activation functions were used throughout the two variants. According to these values, the Adam variant has a clear preference for selecting the ReLu activation function, except in three of the problems - *credit-g*, *hill-valley-without-noise* and *spectf* - in which for *credit-g* and *hill-valley-without-noise* the Tanh activation function is the preferred one by five and four times, respectively. For *spectf* the Logistic activation function is the favorite one by two times. SGD shows much more distributed values across the three different types of activation functions: the Logistic function is the preferred one in six of the problems, ReLu is selected over the others in five of the tasks and Tanh is preferred in the remaining four considered benchmarks.

5.3 Generalization Analysis

This section provides an assessment of the generalization (i.e., the performance achieved on test set over the 30 outer folds) of SLM and MLP taking into account the best configuration after the parameter tuning phase. Figures 5.1 and 5.2 show the AUROC values for both algorithms in the form of boxplots. In each box, the middle mark represents the median, while the edges represent the first and third quartile (25th percentile and 75th percentile, respectively). The whiskers extend to the most extreme observations

Dataset	Adam			SGD		
	Logistic	Relu	Tanh	Logistic	Relu	Tanh
agaricus-lepiota	5	20	5	17	8	5
breast-cancer-wisconsin	10	16	4	5	19	6
clean1	4	22	4	6	15	9
clean2	10	14	6	14	7	9
credit-g	11	7	12	13	9	8
diabetes	1	17	12	0	13	17
hill-valley-without-noise	5	11	14	10	11	9
hill-valley-with-noise	6	15	9	13	12	5
ionosphere	2	26	2	3	16	11
kr-vs-kp	8	16	6	3	10	17
molecular-biology-promoters	2	24	4	12	7	11
sonar	11	13	6	2	15	13
spambase	10	19	1	13	14	3
spectf	21	4	5	21	2	7
tokyo1	7	19	4	8	4	18

Table 5.9: Activation functions use by MLP variant

which are not considered outliers. Any record out of those whiskers is considered an outlier.

These results show that the SLM achieved better results in 9 out of the 15 considered data sets: *breast-cancer-wisconsin*, *clean1*, *hill-valley-with-noise*, *hill-valley-without-noise*, *molecular-biology-promoters*, *sonar*, *spambase*, *spectf* and *tokyo1*. For *agaricus-lepiota*, MLP achieved a better median value, however, it shows a lot of variability. For *clean2*, *credit-g*, *diabetes* and *ionosphere*, the test AUROC values are very similar but in general, in these cases, SLM presents a higher value of outliers. There is only one benchmark – *kr-vs-kp* – where MLP clearly outperforms the Semantic Learning Machine algorithm.

To assess the statistical significance of these results, a set of statistical tests is applied. First, a Kolmogorov-Smirnov test is performed to evaluate whether these values come from a normal distribution. The outcome of this test (present in 5.10) suggests that the alternative hypothesis (i.e., the data do not come from a normal distribution) cannot be rejected considering a significance level (α) of 0.05 and consequently, a rank-based test like the Mann-Whitney U test is selected for the next step. In this case, the null hypothesis states that both samples have equal means and thus no statistically significant difference between the two models. Just like in the previous test, considering a significance level of 0.05, the results indicate that the SLM algorithm outperforms MLP in 13 out of the 15 classification problems. The p-values for these comparisons may be found in Table 5.11.

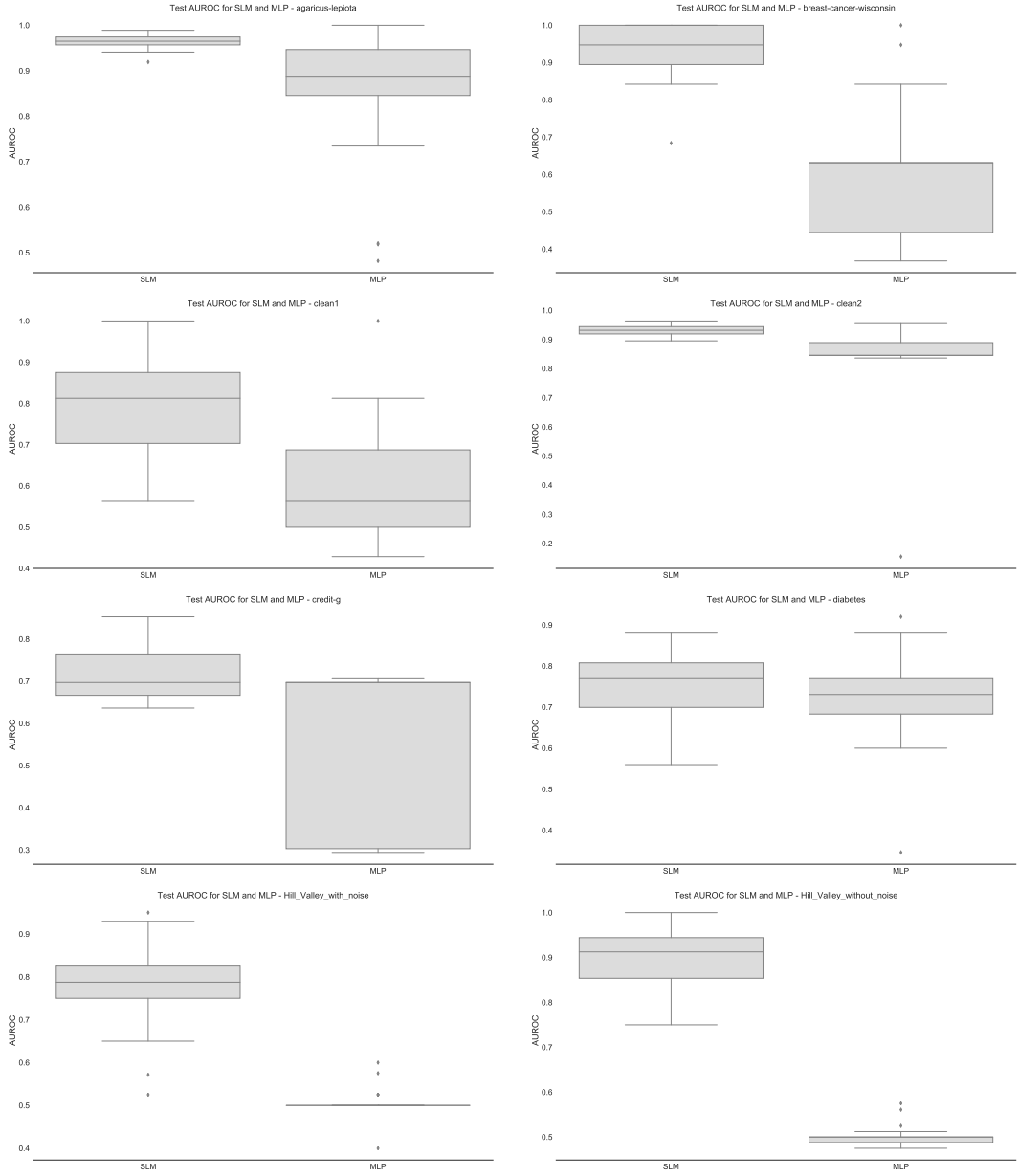

Figure 5.1: Boxplots for test set AUROC values of SLM and MLP: agaricus-lepiota, breast-cancer-wisconsin, clean1, clean2, credit-g, diabetes, hill-valley-with-noise, hill-valley-without-noise

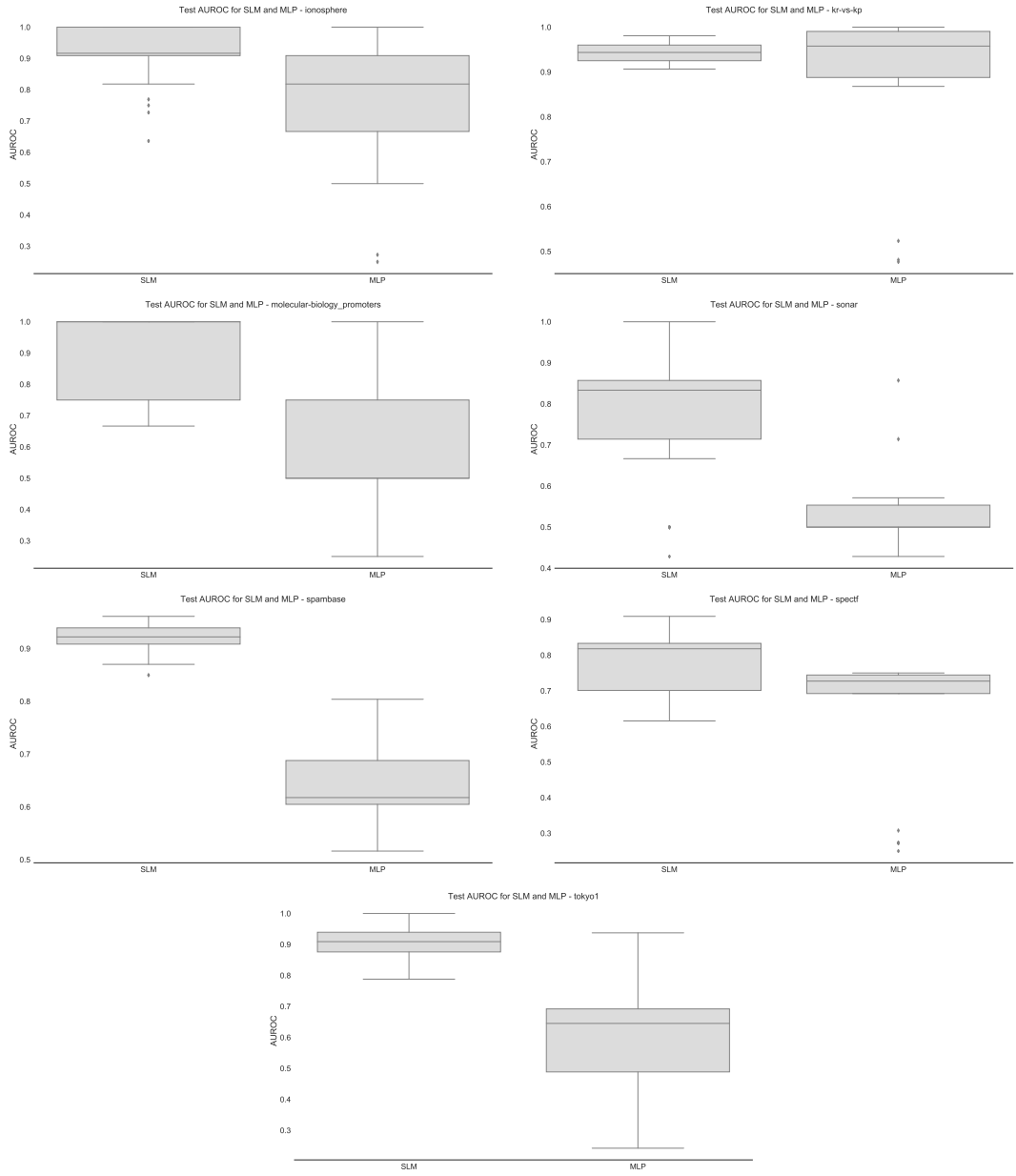

Figure 5.2: Boxplots for test set AUROC values of SLM and MLP: ionosphere, kr-vs-kp, molecular-biology-promoters, sonar, spambase, spectf, tokyo1

Data set	p-value (SLM)	p-value (MLP)
agaricus-lepiota	1.059×10^{-22}	1.621×10^{-14}
breast-cancer-wisconsin	5.569×10^{-19}	1.222×10^{-12}
clean1	6.240×10^{-16}	1.268×10^{-13}
clean2	3.146×10^{-22}	7.002×10^{-19}
credit-g	2.890×10^{-17}	1.792×10^{-11}
diabetes	6.912×10^{-16}	6.989×10^{-15}
hill-valley-with-noise	2.861×10^{-15}	2.844×10^{-13}
hill-valley-without-noise	2.159×10^{-19}	2.091×10^{-14}
sonar	1.502×10^{-7}	1.267×10^{-13}
ionosphere	2.890×10^{-17}	7.642×10^{-12}
kr-vs-kp	1.890×10^{-22}	1.250×10^{-15}
molecular-biology-promoters	7.983×10^{-18}	2.844×10^{-13}
spambase	2.533×10^{-21}	4.052×10^{-15}
spectf	6.981×10^{-17}	8.288×10^{-11}
tokyo1	4.050×10^{-20}	4.601×10^{-11}

Table 5.10: *p*-values of Kolmogorv-Smirnov tests over test set AUROC values of SLM and MLP

Data set	p-value
agaricus-lepiota	2.537×10^{-5}
breast-cancer-wisconsin	6.493×10^{-10}
clean1	8.189×10^{-7}
clean2	2.963×10^{-8}
credit-g	1.503×10^{-3}
diabetes	1.312×10^{-1}
hill-valley-with-noise	5.572×10^{-12}
hill-valley-without-noise	9.472×10^{-12}
sonar	1.502×10^{-7}
ionosphere	4.639×10^{-4}
kr-vs-kp	2.744×10^{-1}
molecular-biology-promoters	9.989×10^{-6}
spambase	1.448×10^{-11}
spectf	2.881×10^{-4}
tokyo1	8.366×10^{-10}

Table 5.11: *p*-values of Mann-Whitney U-tests over test set AUROC values of SLM and MLP

5.4 Ensemble Analysis

This section consists in the study of applying different ensemble construction methods when using the SLM and the MLP as a base learner. Bagging (Breiman, 1996) and Boosting (Drucker, 1997) are compared with a common simple averaging construction method which trains the base learner N times without altering the training set. These three methods of building ensembles are used to create ensembles of 30 NNs using the SLM as a base learner. In the case of Boosting, four variations of the AdaBoost.R2 were used and labeled in the following way:

- Boosting-1: weighted median prediction and fixed learning rate of 1
- Boosting-2: weighted median prediction and variable learning rate selected randomly in the interval $[0,1]$ for each new NN added to the ensemble
- Boosting-3: weighted mean prediction and fixed learning rate of 1
- Boosting-4: weighted mean prediction and variable learning rate selected randomly in the interval $[0,1]$ for each new NN added to the ensemble

5.4.1 SLM as a base learner

Figures 5.3 and 5.4 present the boxplots for the test set AUROC values of each ensemble construction method considered: Simple (averaging), Bagging and the four Boosting variants. Results suggest that in general, the six approaches considered behave not too differently. In terms of the median AUROC value, both the simple averaging method and the bagging approach achieve higher values in four of the data sets: *agaricus-lepiota*, *hill-valley-with-noise*, *hill-valley-without-noise*, *kr-vs-kp*, *sonar* and *spambase*. Both these construction methods outperform the remaining approaches in one of the classification tasks: the simple averaging ensemble method in *spectf* and bagging in *clean1*. It seems that the stochastic nature of the SLM confers enough diversity to the base learner in its search process in a way that even a simple averaging method is able to perform well without having to delve into more detailed intricacies inherent of the bagging and boosting approaches.
It is interesting to see that for the *hill-valley* benchmark problems, results are more concentrated on higher values for the version without noise, while for the task with noise, results show a higher variability. Nonetheless, as mentioned previously, the simple averaging method and bagging achieve a better performance on both of the challenges. The same analysis can be performed for the *clean* data sets. As seen previously, the SLM achieves better results in the version with more instances even if this means that the data set is more imbalanced.

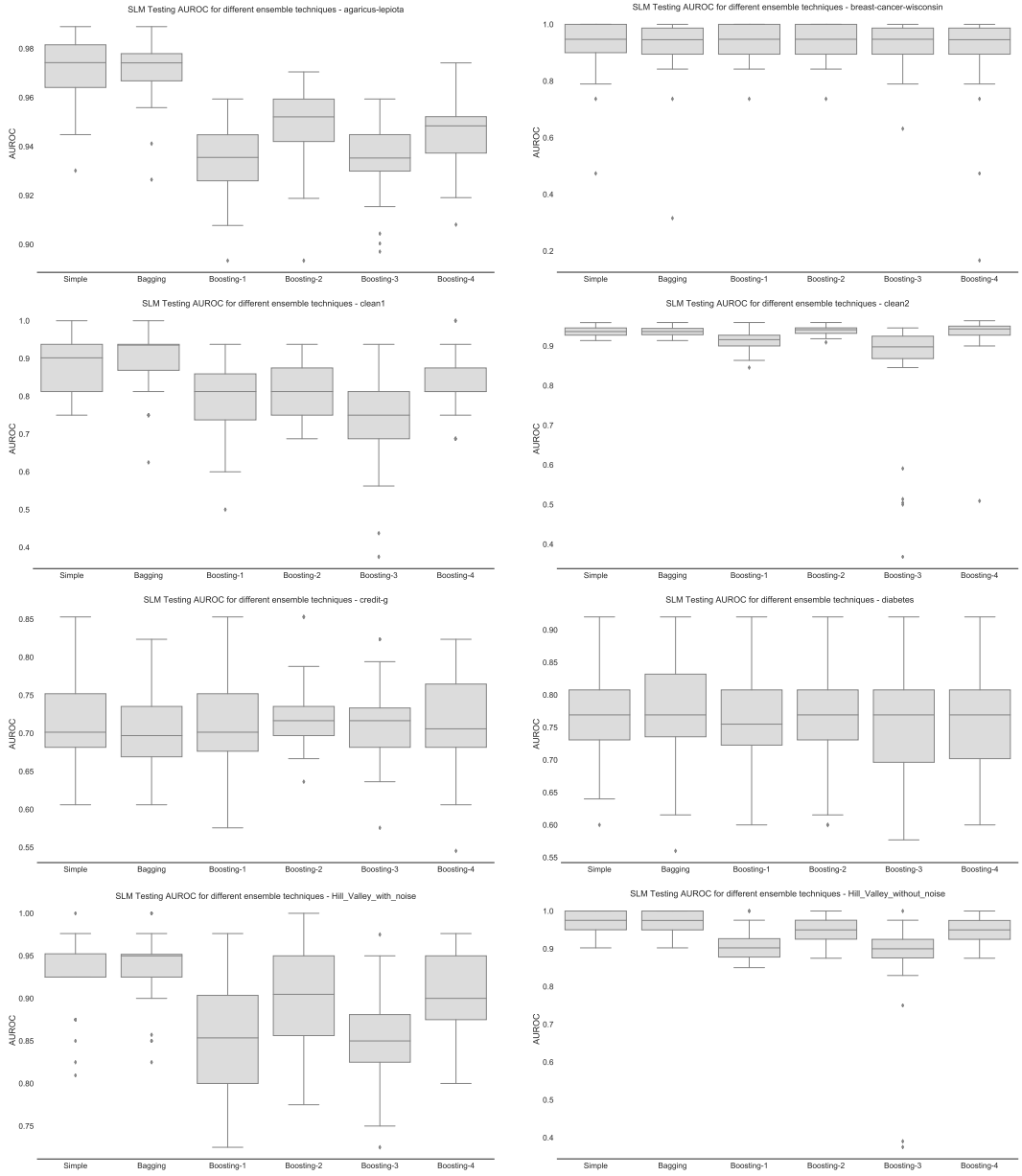

Figure 5.3: Boxplots for test set AUROC values of each ensemble construction method considered: agaricus-lepiota, breast-cancer-wisconsin, clean1, clean2, credit-g, diabetes, hill-valley-with-noise, hill-valley-without-noise

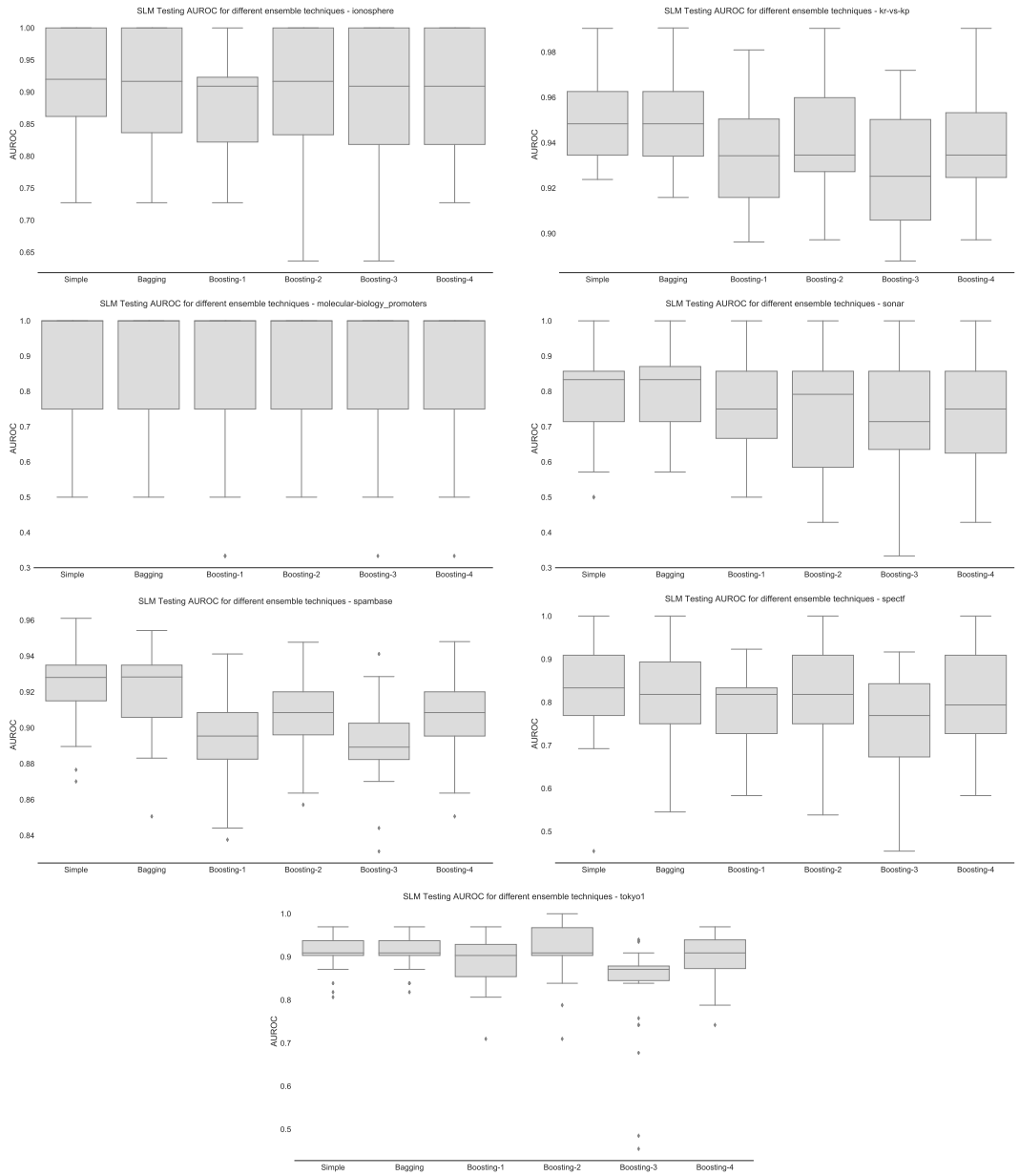

Figure 5.4: Boxplots for test set AUROC values for each ensemble construction method considered: ionosphere, kr-vs-kp, molecular-biology-promoters, sonar, spambase, spectf, tokyo1

5.4.2 MLP as a base learner

The same analysis can be performed when using the MLP as base learner. Figures 5.5 and 5.6 show the boxplots for the test set AUROC values of each of the six ensemble construction methods considered. In general, the six approaches perform similarly across the different benchmark problems. Considering the median AUROC values and comparing the ensemble results when using the MLP as a base learner rather than the SLM, it is possible to verify that the former reach higher values in two data sets: *credit-g* and *spectf*. For *agaricus-lepiota*, the Boosting variants perform better with MLP as a base learner than with SLM.

Once again it is interesting to compare the *hill-valley* data sets: using the MLP as a base learner results in the ensembles achieving better median AUROC values in the version with noise than in the version without noise, something that did not happen when using the SLM as the base learner. Nevertheless, ensembles built with the SLM outperform the ensembles built with the MLP in both benchmarks. Results for the *clean* tasks suggest that it is not very For the *clean* tasks, it is not very clear if MLP is able to perform efficiently with less instances to train the model. It looks like the minimum AUROC value in clean2 is higher than in clean1 but at the same time there are way more outliers in the former.

It is also interesting to see that, in general, the Boosting-1 and Boosting-2 variants tend to achieve either lower or more variable results than the remaining approaches.

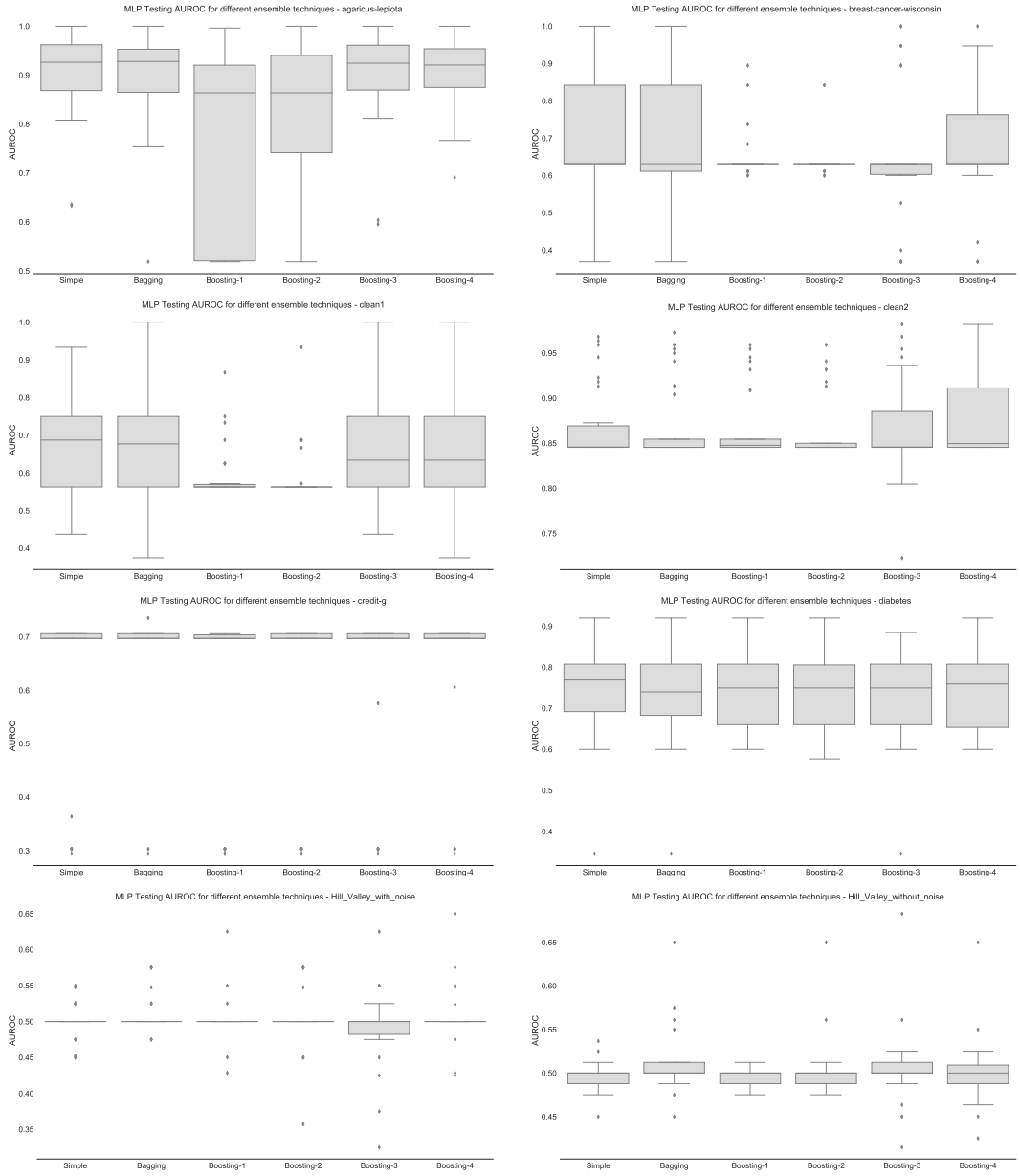

Figure 5.5: Boxplots for test set AUROC values of each ensemble construction method considered using the MLP as base learner: breast-cancer-wisconsin, clean1, credit-g, diabetes, hill-valley-with-noise, hill-valley-without-noise

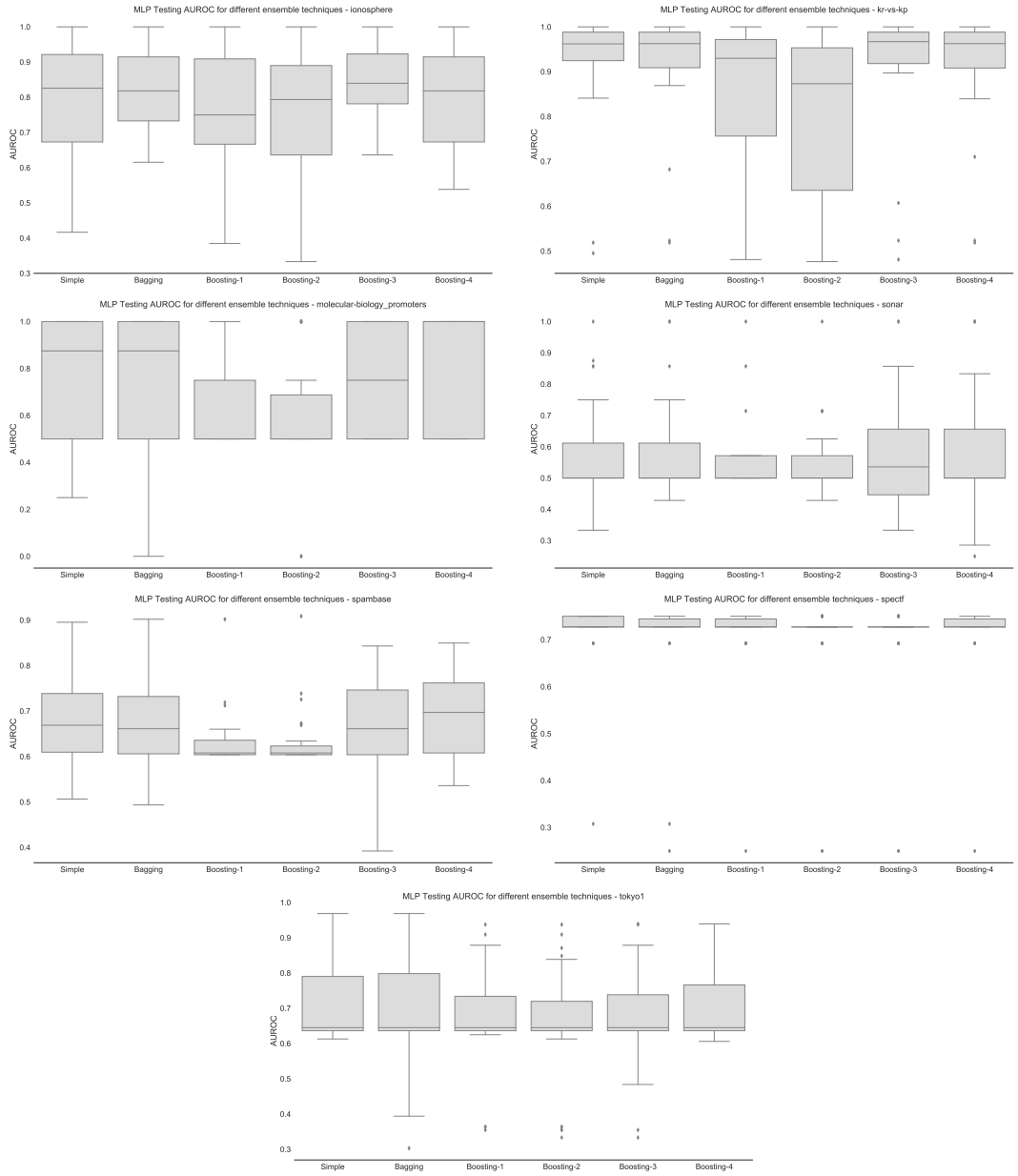

Figure 5.6: Boxplots for test set AUROC values for each ensemble construction method considered using the MLP as a base learner: ionosphere, kr-vs-kp, molecular-biology-promoters, sonar, spambase, spectf, tokyo1

Chapter 6

Conclusions

This work focused on two main objectives:

- Studying the dynamic use of training data under the SLM algorithm;
- Exploring different ensemble construction methods using the SLM as a base learner.

To explore the dynamic use of training data two techniques were considered: dynamic subset sampling of the training data, where at each generation a subset of the training examples is chosen, and dynamic weighting of the training instances, where random weights are assigned to these instances at each generation. Results showed that these approaches are useful to improve the resulting generalization.
To study the effect of using the SLM as a base learner for different ensemble construction methods, six ensemble variants were taken into account: simple averaging method, Bagging and four variants of Boosting. Results suggested that there is not really a method that stands out but it is important to mention that due to the stochastic nature of the SLM algorithm, it was possible to verify that a simple averaging ensemble method can be as competitive as Bagging and Boosting.
Finally, when assessing the generalization ability of the SLM algorithm, results proved that the SLM outperformed MLP in 13 of the considered tasks, with statistical significance, after parameter tuning was performed for both algorithms.

Bibliography

Agostinelli, F., Hoffman, M., Sadowski, P., & Baldi, P. (2014). Learning activation functions to improve deep neural networks. arXiv: 1412.6830 [cs.NE]

Alba, E., Aldana, J. F., & Troya, J. M. (1993). Full automatic ann design: A genetic approach. In J. Mira, J. Cabestany, & A. Prieto (Eds.), *New trends in neural computation* (pp. 399–404). Berlin, Heidelberg: Springer Berlin Heidelberg.

Angeline, P. J., Saunders, G. M., & Pollack, J. B. (1994). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, *5*(1), 54–65. doi:10.1109/72.265960

Bornholdt, S., & Graudenz, D. (1992). General asymmetric neural networks and structure design by genetic algorithms. *Neural Networks*, *5*(2), 327–334. doi:https://doi.org/10.1016/S0893-6080(05)80030-9

Branke, J. (1995). Evolutionary algorithms for neural network design and training. In *In proceedings of the first nordic workshop on genetic algorithms and its applications* (pp. 145–163).

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. doi:10.1023/A:1018054314350

Chunkai Zhang, Huihe Shao, & Yu Li. (2000). Particle swarm optimisation for evolving artificial neural network. In *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no.0* (Vol. 4, 2487–2490 vol.4). doi:10.1109/ICSMC.2000.884366

Cun, Y. L., Denker, J. S., & Solla, S. A. (1990). Optimal brain damage. In *Advances in neural information processing systems* (pp. 598–605). Morgan Kaufmann.

Dasgupta, B., & Schnitger, G. (1992). Efficient approximation with neural networks: A comparison of gate functions.

Davis, L. (1989). Mapping classifier systems into neural networks. In D. S. Touretzky (Ed.), *Advances in neural information processing systems 1* (pp. 49–56). Morgan-Kaufmann. Retrieved from http://papers.nips.cc/paper/162-mapping-classifier-systems-into-neural-networks.pdf

Ding, S., Li, H., Su, C., Yu, J., & Jin, F. (2013). Evolutionary artificial neural networks: A review. *Artif. Intell. Rev. 39*(3), 251–260. doi:10.1007/s10462-011-9270-6

Drucker, H. (1997). Improving regressors using boosting techniques. In *Proceedings of the fourteenth international conference on machine learning* (pp. 107–115). ICML '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Retrieved from http://dl.acm.org/citation.cfm?id=645526.657132

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In *Advances in neural information processing systems 2* (pp. 524–532). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Floreano, D., Dürr, P., & Mattiussi, C. (2008). Neuroevolution: From architectures to learning. *Evolutionary Intelligence*, *1*, 47–62.

Frean, M. (1990). The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation*, *2*(2), 198–209. doi:10.1162/neco.1990.2.2.198

Garro, B. A., & Vázquez, R. (2015). Designing artificial neural networks using particle swarm optimization algorithms. *Computational Intelligence and Neuroscience*, *2015*. doi:10.1155/2015/369298

Gonçalves, I. (2017). *An exploration of generalization and overfitting in genetic programming: Standard and geometric semantic approaches* (Doctoral dissertation, Department of Informatics Engineering, University of Coimbra, Portugal).

Gonçalves, I., & Silva, S. (2013). Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In *European conference on genetic programming* (pp. 73–84). Springer.

Gonçalves, I., Silva, S., & Fonseca, C. M. (2015a). On the generalization ability of geometric semantic genetic programming. In *Genetic programming* (pp. 41–52). Springer.

Gonçalves, I., Silva, S., & Fonseca, C. M. (2015b). Semantic learning machine: A feedforward neural network construction algorithm inspired by geometric semantic genetic programming. In *Progress in artificial intelligence* (Vol. 9273, pp. 280–285). Lecture Notes in Computer Science. Springer.

Gonçalves, I., Silva, S., Fonseca, C. M., & Castelli, M. (2017). Unsure when to stop? ask your semantic neighbors. In *Proceedings of the genetic and evolutionary computation conference* (pp. 929–936). GECCO '17. doi:10.1145/3071178.3071328

Gonçalves, I., Silva, S., Melo, J. B., & Carreiras, J. M. B. (2012). Random sampling technique for overfitting control in genetic programming. In *Genetic programming* (pp. 218–229). Springer.

Gruau, F., Whitley, D., & Pyeatt, L. (1996). A comparison between cellular encoding and direct encoding for genetic neural networks. In *Proceedings of the 1st annual conference on genetic programming* (pp. 81–89). Stanford, California: MIT Press.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. arXiv: 1502.01852 [cs.CV]

Jagusch, J.-B., Gonçalves, I., & Castelli, M. (2018). Neuroevolution under unimodal error landscapes: An exploration of the semantic learning machine algorithm. In *Proceedings of the genetic and evolutionary computation conference companion* (pp. 159–160). ACM.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., & LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In *2009 ieee 12th international conference on computer vision* (pp. 2146–2153). doi:10.1109/ICCV.2009.5459469

Jian, F., & Yugeng, X. (1997). Neural network design based on evolutionary programming. *Artificial Intelligence in Engineering*, *11*(2), 155–161. doi:https://doi.org/10.1016/S0954-1810(96)00025-8

Kiefer, J., & Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist. 23*(3), 462–466. doi:10.1214/aoms/1177729392

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv: 1412.6980 [cs.LG]

Kiranyaz, S., Ince, T., Yildirim, A., & Gabbouj, M. (2009). Evolutionary artificial neural networks by multi-dimensional particle swarm optimization. *Neural Networks*, *22*(10), 1448–1462. doi:https://doi.org/10.1016/j.neunet.2009.05.013

Konda, K., Memisevic, R., & Krueger, D. (2014). Zero-bias autoencoders and the benefits of co-adapting features. arXiv: 1402.3337 [stat.ML]

Koza, J. R., & Rice, J. P. (1991). Genetic generation of both the weights and architecture for a neural network. In *Ijcnn-91-seattle international joint conference on neural networks* (Vol. 2, 397–404 vol.2). doi:10.1109/IJCNN.1991.155366

Lapa, P., Gonçalves, I., Rundo, L., & Castelli, M. (2019a). Enhancing classification performance of convolutional neural networks for prostate cancer detection on magnetic resonance images: A study with the semantic learning machine. (pp. 381–382). doi:10.1145/3319619.3322035

Lapa, P., Gonçalves, I., Rundo, L., & Castelli, M. (2019b). Semantic learning machine improves the cnn-based detection of prostate cancer in non-contrast-enhanced mri. In M. López-Ibáñez (Ed.), *Gecco 2019 companion* (pp. 1837–1845). GECCO 2019 Companion : Proceedings of the 2019 Genetic and Evolutionary Computation Conference Companion (pp. 1837-1845). New York: Association for Computing Machinery, Inc. https://doi.org/10.1145/3319619.3326864. doi:10.1145/3319619.3326864

Lichman, M. (2013). UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences. Retrieved from http://archive.ics.uci.edu/ml

Litjens, G., Debats, O., Barentsz, J., Karssemeijer, N., & Huisman, H. (2017). Prostatex challenge data. *The Cancer Imaging Archive*. doi:https://doi.org/10.7937/K9TCIA.2017.MURS5CL

Liu, S., Zheng, H., Feng, Y., & Li, W. (2017). Prostate cancer diagnosis using deep learning with 3d multiparametric MRI. *CoRR*, *abs/1703.04078*. arXiv: 1703.04078. Retrieved from http://arxiv.org/abs/1703.04078

Liu, Y., & Khoshgoftaar, T. (2004). Reducing overfitting in genetic programming models for software quality classification. In *Proceedings of the eighth ieee international conference on high assurance systems engineering* (pp. 56–65). HASE '04. Tampa, Florida: IEEE Computer Society.

Maas, A. L. (2013). Rectifier nonlinearities improve neural network acoustic models.

Manessi, F., & Rozza, A. (2018). Learning combinations of activation functions. *2018 24th International Conference on Pattern Recognition (ICPR)*. doi:10.1109/icpr.2018.8545362

Mani, G. (1990). Learning by gradient descent in function space. In *1990 ieee international conference on systems, man, and cybernetics conference proceedings* (pp. 242–247). doi:10.1109/ICSMC.1990.142101

Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., ... Hodjat, B. (2017). Evolving deep neural networks. arXiv: 1703.00548 [cs.NE]

Miller, G., Todd, P., & Hegde, S. (1989). Designing neural networks using genetic algorithms. (pp. 379–384).

Montana, D. J., & Davis, L. (1989). Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th international joint conference on artificial intelligence - volume 1* (pp. 762–767). IJCAI'89. Detroit, Michigan: Morgan Kaufmann Publishers Inc.

Moraglio, A., Krawiec, K., & Johnson, C. G. (2012). Geometric semantic genetic programming. In C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, & M. Pavone (Eds.), *Parallel problem solving from nature - ppsn xii* (pp. 21–31). Berlin, Heidelberg: Springer Berlin Heidelberg.

Mozer, M. C., & Smolensky, P. (1989). Skeletonization: A technique for trimming the fat from a network via relevance assessment. In D. S. Touretzky (Ed.), *Advances in neural information processing systems 1* (pp. 107–115). Morgan-Kaufmann. Retrieved from http://papers.nips.cc/paper/119-skeletonization-a-technique-for-trimming-the-fat-from-a-network-via-relevance-assessment.pdf

Nikolopoulos, C., & Fellrath, P. (1994). A hybrid expert system for investment advising. *Expert Systems*, *11*(4), 245–250. doi:10.1111/j.1468-0394.1994.tb00332.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1468-0394.1994.tb00332.x

Oliker, S., Furst, M., & Maimon, O. (1993). Design architectures and training of neural networks with a distributed genetic algorithm. In *Ieee international conference on neural networks* (199–202 vol.1). doi:10.1109/ICNN.1993.298556

Orzechowski, P., Cava, W. G. L., & Moore, J. H. (2018). Where are we now? A large benchmark study of recent symbolic regression methods. *CoRR*, *abs/1804.09331*. arXiv: 1804.09331. Retrieved from http://arxiv.org/abs/1804.09331

Schaffer, J., Caruana, R. A., & Eshelman, L. J. (1990). Using genetic search to exploit the emergent behavior of neural networks. *Physica D: Nonlinear Phenomena*, *42*(1), 244–248. doi:https://doi.org/10.1016/0167-2789(90)90078-4

Schiffmann, W., Joost, M., & Werner, R. (1992). Synthesis and performance analysis of multilayer neural network architectures.

Schoenauer, M., & Ronald, E. (1997). Genetic extensions of neural net learning: Transfer functions and renormalisation coefficients.

Sietsma, J., & Dow, R. J. (1991). Creating artificial neural networks that generalize. *Neural Networks*, *4*(1), 67–79. doi:https://doi.org/10.1016/0893-6080(91)90033-2

Srinivas, M., & Patnaik, L. M. (1991). Learning neural network weights using genetic algorithms-improving performance by search-space reduction. In *[proceedings] 1991 ieee international joint conference on neural networks* (2331–2336 vol.3). doi:10.1109/IJCNN.1991.170736

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

Stanley, K., Clune, J., Lehman, J., & Miikkulainen, R. (2019). Designing neural networks through neuroevolution. *Nature Machine Intelligence*, *1*. doi:10.1038/s42256-018-0006-z

Wang, F.-Y., Zhang, J. J., Zheng, X., Wang, X., Yuan, Y., Dai, X., ... Yang, L. (2016). Where does alphago go: From church-turing thesis to alphago thesis and beyond. *IEEE/CAA Journal of Automatica Sinica*, *3*(2), 113–120.

White, D., & Ligomenides, P. (1993). Gannet: A genetic algorithm for optimizing topology and weights in neural network design. In J. Mira, J. Cabestany, & A. Prieto (Eds.), *New trends in neural computation* (pp. 322–327). Berlin, Heidelberg: Springer Berlin Heidelberg.

Whitley, D. (1989). Applying genetic algorithms to neural network problems. In *International neural network society* (p. 230). Department of Computer Science, Colorado State University, Fort Collins, CO 80523 USA.

Whitley, D., & Hanson, T. (1989). Optimizing neural networks using faster, more accurate genetic search. In *Proceedings of the third international conference on genetic algorithms* (pp. 391–396). George Mason University, USA: Morgan Kaufmann Publishers Inc.

Wilson, S. W. (1990). Perception redux: Emergence of structure. *Physica D: Nonlinear Phenomena*, *42*(1), 249–256. doi:https://doi.org/10.1016/0167-2789(90)90079-5

Xin Yao. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, *87*(9), 1423–1447. doi:10.1109/5.784219

Yao, X., & Liu, Y. (1997). A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, *8*(3), 694–713. doi:10.1109/72.572107

Appendix

Data sets description

Data set	Description
agaricus-lepiota	Includes descriptions of hypothetical samples corresponding to 23 species of mushrooms in the Agaricus and Lepiota Family. The objective is to classify the record as edible or poisonous.
breast-cancer-wisconsin	Contains continuous measurements from tumors. The algorithm must classify the tumor as benign or malignant.
clean1	Describes a set of 102 molecules where the goal is to learn to predict whether new molecules will be musks or non-musks.
clean2	Describes a set of 102 molecules where the goal is to learn to predict whether new molecules will be musks or non-musks.
credit-g	Contains credit card applications. The objective is to classify the individuals as either good or bad credit.
diabetes	Contains patient records where the goal is to predict whether the individuals have diabetes or not.
hill-valley-with-noise	Each record represents 100 points on a two dimensional graph, where the algorithm must classify the series as either a Hill (a "bump"in the terrain) or a Valley (a "dip"in the terrain) - contains noise.
hill-valley-without-noise	Each record represents 100 points on a two dimensional graph, where the algorithm must classify the series as either a Hill (a "bump"in the terrain) or a Valley (a "dip"in the terrain) - contains no noise.
sonar	Contains patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions. The algorithm must predict whether the objects are rocks or mines.
ionosphere	Contains continuous measurements from high-frequency antennas. The algorithm must classify whether the signals are good or bad.
kr-vs-kp	Contains chess data where the goal is to classify whether the white pieces can win or not.
molecular-biology-promoters	Contains molecular information and the objective is to predict whether the records are promoters or not.
spambase	Contains word frequencies in emails. The algorithm must classify whether the e-mails are spam or not.
spectf	Describes the diagnose of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the records is classified into two categories: normal and abnormal.
tokyo1	Contains Server Performance Data categorizing each record as either good or bad.

Table A.1: Description of the binary classification data sets considered