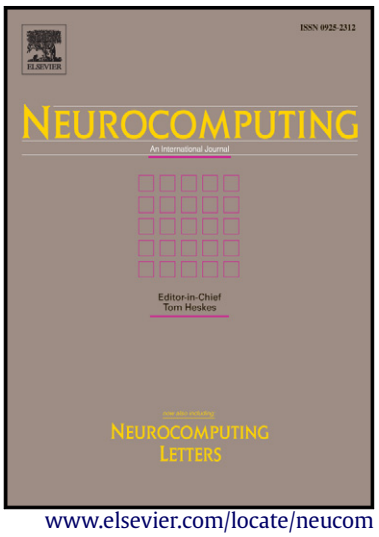


www.elsevier.com/locate/neucom

Comparison of a genetic algorithm and simulated annealing for automatic neural network ensemble development

Symone Soares, Carlos Henggeler Antunes, Rui Araújo

PII: S0925-2312(13)00579-1
DOI: http://dx.doi.org/10.1016/j.neucom.2013.05.024
Reference: NEUCOM13428

To appear in: *Neurocomputing*

Received date: 29 November 2012
Revised date: 21 March 2013
Accepted date: 15 May 2013

Cite this article as: Symone Soares, Carlos Henggeler Antunes, Rui Araújo, Comparison of a genetic algorithm and simulated annealing for automatic neural network ensemble development, *Neurocomputing*, http://dx.doi.org/10.1016/j.neucom.2013.05.024

Comparison of a Genetic Algorithm and Simulated Annealing for Automatic Neural Network Ensemble Development

Symone Soares[a,*], Carlos Henggeler Antunes[b], Rui Araújo[a]

[a] *Institute for Systems and Robotics, Department of Electrical and Computer Engineering, University of Coimbra, Pólo II, PT-3030-290 - Coimbra, Portugal.*
[b] *Department of Electrical and Computer Engineering, and INESC Coimbra, University of Coimbra, Pólo II, PT-3030-290 - Coimbra, Portugal.*

Abstract

In the last decades ensemble learning has established itself as a valuable strategy within the computational intelligence modeling and machine learning community. Ensemble learning is a paradigm where multiple models combine in some way their decisions, or their learning algorithms, or different data to improve the prediction performance. Ensemble learning aims at improving the generalization ability and the reliability of the system. Key factors of ensemble systems are diversity, training and combining ensemble members to improve the ensemble system performance. Since there is no unified procedure to address all these issues, this work proposes and compares Genetic Algorithm and Simulated Annealing based approaches for the automatic development of Neural Network Ensembles for regression problems. The main contribution of this work is the development of optimization techniques that select the best subset of models to be aggregated taking into account all the key factors of ensemble systems (e.g., diversity, training ensemble members and combination strategy). Experiments on two well-known data sets are reported to evaluate the effectiveness of the proposed methodologies. Results show that these outperform other approaches including Simple Bagging, Negative Correlation Learning (NCL), AdaBoost and GASEN in terms of generalization ability.

Keywords: Ensemble Learning, Neural Network, Genetic Algorithm, Simulated Annealing.

1. Introduction

In the last decades ensemble learning has established itself as a valuable strategy within the computational intelligence modeling and machine learning community. Ensemble learning has proven to be effective in a broad set of machine learning problems, including feature selection [1], small data sets [2], local learning [3], concept drift theory [4], among others. Ensemble learning is a paradigm in which multiple models combine in some way their decisions, or their learning algorithms, or different data to improve the prediction performance. This technique is also known as Ensemble Method (EM) or Ensemble system. Studies have shown that an ensemble system is generally more accurate than any individual model, and its effectiveness has been recognized in different benchmark data sets [5, 6, 7, 8, 9].

In this setting, Neural Network Ensembles (NNEs) have been widely investigated for both classification and regression problems [10]. The main motivation is that the generalization ability of the system can be significantly improved.

Key factors of the ensemble system are diversity, training and combining ensemble members [11]. The success of the ensemble system is mainly attributed to the diversity degree within the ensemble. A good ensemble is one in which the models make different errors on the same data point [6]. Research has encouraged this diversity by manipulating the training data set [9], or by designing ensembles with different architectures or heterogeneous learning algorithms [12, 8]. The ensemble member training is the strategy employed to train individual ensemble members. Several algorithms have been developed for this task, including Bagging [13] and Boosting [14]. The last key factor of an ensemble system is the approach used to combine the individual models in the ensemble. This strategy depends on the type of classifiers. For example, some classifiers provide only discrete-valued label outputs and others only continuous valued class-specific outputs.

Despite of the remarkable performance of ensemble systems, a major drawback is that it is usually necessary to combine a large number of models to ensure the ensemble accuracy. A good way to alleviate this problem is the adequate selection of the subset of models from the original set of models [15, 16]. This approach is also known as ensemble pruning [17]. The aim is to find a good subset of ensemble members in order to improve generalization ability, and which additionally reduces the system complexity. However, the ensemble pruning is a difficult problem whose solution is commonly computationally expensive. Pruning an ensemble with n models requires searching in the space of the $2^n - 1$ non-empty solutions to minimize a cost function correlated with the generalization error [18].

To address this problem, a number of different meta-heuristics has been developed for model selection [19]. For example, Kim and Oh [20] adopt a Hybrid Genetic Algorithm inspired by the feature selection problem to select the most appropriated models for the ensemble. Other example is the Genetic Algorithm based Selective Ensemble (GASEN), which trains a

*Corresponding author
Email addresses: symonesoares@isr.uc.pt (Symone Soares), ch@deec.uc.pt (Carlos Henggeler Antunes), rui@isr.uc.pt (Rui Araújo)

set of Neural Networks (NNs) using bootstrap to increase the diversity among the models. GASEN uses a Genetic Algorithm (GA) to select an optimal subset of NN models to include in the ensemble. In this strategy, a weight derived from the marginal improvement in the *fitness* (measuring the solution quality) associated with including a model in the ensemble is assigned to each model. Then the models whose weights are higher than a fixed threshold are selected for inclusion in the ensemble [21]. The main drawback of GASEN is that the NNs have fixed architectures and the combination techniques are only simple average and weighted average for regression and classification, respectively.

Liu et al. [7] present an automatic strategy for designing ensemble systems using Evolutionary Learning and Negative Correlation Learning (EENCL). Negative Correlation Learning (NCL) generates negatively correlated NN models using a correlation penalty term in the error function to encourage specialization and cooperation among the models. EENCL does not explore the linear combination among the models and the models' architectures are also predefined. On the other hand, Bayesian Artificial Immune System (BAIS) is an immune-inspired methodology for designing NNEs with better generalization ability when compared to the EENCL. BAIS introduces diversity in the models' architecture. However, only one combination type is used for designing the NN models [22].

This work proposes and compares Genetic Algorithm and Simulated Annealing (SA) based approaches for the automatic development of NNEs for regression problems. The main contribution of this work is the development of optimization techniques to select the best subset of models to be aggregated take into account all the key factors of ensemble systems (i.e. diversity, training ensemble members and combination strategy). Firstly, a set of models with a high degree of diversity is generated. For each model, the proposed approach creates a different training data set by applying bootstrap. Then the method selects the best model's architecture by varying the number of hidden neurons, the activation functions and the weight initializations. Secondly, the optimization strategy is employed to select both the best subset of models and the optimal combination strategy for aggregating the subset of models. Experiments on two well-known data sets are reported to evaluate the effectiveness of the proposed methodologies. Results show that these outperform other approaches including Simple Bagging, NCL, AdaBoost and GASEN in terms of generalization ability.

The paper is organized as follows. Section 2 reports the key factors in ensemble system development. In Section, 3 key issues for designing ensemble systems are outlined. Section 4 details the proposed methodologies. Experimental results are detailed and analyzed in Section 5. Finally, Section 6 contains some concluding remarks.

2. Key Factors in Ensemble Systems

Generalization is an important issue in machine learning. Generalization refers to the predictor ability to perform well when applied to unseen data. An ensemble of models has shown to generalize better than any single model in the ensemble. Theoretical [6] and empirical [2] studies have proven why ensembles perform better than single learners. Despite the remarkable performance of ensembles, building ensemble systems is not an easy task. The important key of ensemble systems is to design an ensemble which performs better than random individual predictors and models which make different errors on the same sample [23]. That is, *diversity* is necessary in the ensemble members' decisions. If the models provide the same output, there is nothing to be gained from their aggregation. Different models usually make different errors, which means that by combining diverse models it is possible to make more accurate decisions.

During the ensemble development, diversity should be taken into account. The system may or may not explicitly try to optimize some metric of diversity during the design of the ensemble system. These strategies are divided into *explicit* and *implicit* diversity methods, respectively. While implicit methods rely on randomness to generate diversity, explicit methods deterministically generate diversity. For example, Bagging (short for **B**ootstrap **Agg**regation Learn**ing**) employs an *implicit* strategy to achieve diversity [8, 24]. Bagging randomly samples the training data set by applying bootstrap to create a different training data set for each individual predictor [13]; at no point is a measurement taken to promote diversity. On the other hand, Boosting is an *explicit* strategy. Boosting directly manipulates the training data set distributions by the specific weight changes to ensure some form of diversity in the set of models [14, 25]. The main drawback is that there is no guarantee to be the *right* way to promote diversity.

Brown et al. [6] state that the majority of ensemble diversity approaches can be subdivided into three main categories:

(*i*) starting the learning with different conditions;

(*ii*) altering the set of predictors;

(*iii*) altering the trajectory used by the components in the search space.

The first category (*i*) creates each predictor with different initial components. For an ensemble of NN models, training each NN with a different weight initialization technique may increase the probability of continuing on a different trajectory with respect to the other NN models. Approaches in this category generally give poor results, because the predictors are not diverse enough [19]. Methods in this category (*ii*) aim at modifying each ensemble member. Common strategies attempt to manipulate the training data set that each member receives (e.g., k-*fold cross-validation* [26], Bagging, Boosting, or noise injection [27]), or to alter the model's architecture (e.g., NN models with different architectures or different activation functions), or to design members with heterogeneous learning algorithms (e.g., Bayesian, RBF NN models and Support Vector Machines) [8]. Approaches in the third category (*iii*) aim at modifying the way the search space is traversed, leading different component models to converge to different hypotheses. This category can be subdivided into evolutionary methods and penalty methods.

Penalty methods introduce a correlation penalty term into the cost function of the ensemble system so that each model minimizes its error together with the error correlation within the ensemble. On the other hand, evolutionary algorithms can also evolve a population of models using techniques to promote diversity. Penalty methods and evolutionary algorithms can be hybridized. A penalty term can be employed to promote interaction and diversity among the ensemble members and evolutionary algorithms can be used to select the ensemble members [7].

Member selection is also a key strategy for ensemble development. This strategy can lead to better generalization performance. One motivation is that during this process a subset of models with uncorrelated models (or diverse models) can be selected, promoting the diversity in the ensemble. Several strategies have been employed to select the members for the ensemble, including Genetic Algorithms [21], Particle Swarm Optimization [28], Bayesian Artificial Immune System [22], and pruning strategies [17].

During the ensemble development some issues are at stake [29]: how to generate the ensemble members (diversity should be promoted here), how to evaluate the ensemble members, and what member selection should be employed. Other important issue is what combination strategy should be applied to aggregate the models' outputs. The combination strategy is crucial for enhancing the ensemble performance [30] and balancing the diversity among the ensemble members. The main drawback of most ensemble systems is that usually they consider only one combination strategy during ensemble development.

In order to address all the above mentioned key issues related to ensemble system development, the methodologies proposed in this work apply the above mentioned diversity approaches (*i*) and (*ii*) to promote diversity and achieve good generalization ability. In this work the diversity approach (*i*) is applied to start the NN learning with different conditions by using three different weight initializations (details in Subsection 3.2). In this work the diversity approach (*ii*) is employed to promote diversity by modifying each NN. In this case, the NN models are generated using different training data sets obtained by bootstrap and then the best NN architecture is chosen by altering the number of neurons in the hidden layer and the activation functions. Genetic Algorithm and Simulated Annealing are then compared as methods to select the best subset of models and the optimal combination strategy for aggregating this subset.

3. Creating a Neural Network Ensemble

NN is a learning paradigm inspired by biological neurons, and consists of processing elements and connections between them [31]. In NN modeling, there are several network structures and training parameters that need to be carefully chosen. They include the number of layers, number of neurons in each layer, activation functions, weight initialization method, learning rates etc. Several techniques have been proposed for the parameter selection [32, 33]. However, even if the resulting NN is correctly designed, the generalization ability can be a problem [34].

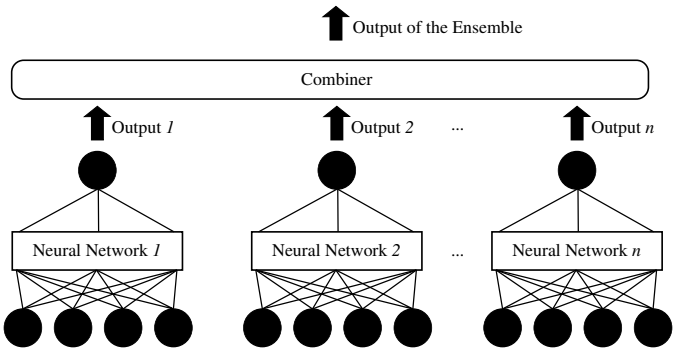


Figure 1: Architecture of a Neural Network ensemble.

Ensemble learning has been established as a very promising approach for improving the generalization of NN systems [5]. Figure 1 illustrates a NNE architecture [35], where the *combiner* is able to aggregate the NN models' outputs. In NNE modeling, optimal combination can enhance the robustness and the accuracy [36]. In this paper, robustness is related to the approximation performance of the NNE on unseen data points (generalization ability). The next subsections are subdivided according to the ensemble development steps.

3.1. Training, Validation and Testing Data Sets

Consider an initial data set (original data set) $\mathbf{D}_{init} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{k}$ of size k, where $\mathbf{x}_i \in \mathbb{R}^{v\times 1}$ (v is the number of input variables) and y_i is the output variable. The initial data set is divided into a training data set $\mathbf{D}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{k_1}$, a validation data set $\mathbf{D}_{valid} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{k_2}$, and a testing data set $\mathbf{D}_{test} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{k_3}$, of size k_1, k_2 and k_3, respectively, and $k = k_1 + k_2 + k_3$.

Bootstrap [37] can be applied in the training data set for promoting diversity in the ensemble. In machine learning, bootstrap is employed to expand upon a single realization of a distribution or generate different data sets that can provide a better understanding of the mean and variability of the original unknown distribution [24]. Bootstrap is performed by randomly sampling with replacement from the original training data set $\mathbf{D}_{train}$. To sample with replacement, one sample $\{(\mathbf{x}_i, y_i)\}$ from $\mathbf{D}_{train}$ is randomly selected and then placed into a new training data set $\mathbf{D}^b_{train}$. $\mathbf{D}^b_{train}$ must contain the same number of samples of $\mathbf{D}_{train}$, i.e. k_1 samples. Random sample selections from $\mathbf{D}_{train}$ continue until $\mathbf{D}^b_{train}$ has been filled with k_1 samples. $\mathbf{D}^b_{train}$ data set may include multiple copies of the same sample and no copies of other samples from $\mathbf{D}_{train}$.

Let us assume an ensemble with n NN models. Then n different training data sets $\mathbf{D}^b_{train}$ are obtained by bootstrap. For each training data set $\mathbf{D}^b_{train}$ the best NN's architecture is selected according to the performance in the validation data set $\mathbf{D}_{valid}$. $\mathbf{D}_{valid}$ is also used to control the overfitting by early stopping. The testing data set $\mathbf{D}_{test}$ is employed to evaluate the ensemble's performance.

3.2. Generation of Neural Networks

After creating n training data sets $\mathbf{D}^b_{train}$, the next step is to produce n NN models. For this purpose, for each training data

set the best NN's architecture is selected.

NN models are implemented using one hidden layer using Multilayer Perceptron Neural Network (MLP NN) trained by the Levenberg-Marquardt (LM) algorithm [38]. LM is a well-known learning algorithm for non-linear problems, widely used in a broad range of applications. The LM algorithm is a hybridization of steepest descent and Gauss-Newton method.

For each training data set, a NN's topology is chosen from a collection of NN models based on its performance. This evaluation is done using the Mean Squared Error (MSE) between the estimated output of NN and the actual output y in the validation data set. The collection of models is obtained by varying the number of neurons in the hidden layer (from 1 to 10); varying among two activation functions (linear and fast hyperbolic tangent) for both the hidden layer and output layer; and three different weight initialization methods. The weight initialization methods are:

- Randomly initialize the weights within the interval $[-1/ni, 1/ni]$, where ni is the number of input neurons [31];
- Nguyen-Widrow approach: set initial weights using the Nguyen-Widrow initialization method [39];
- Randomly initialize the weights within the interval $[-0.5, 0.5]$, using a uniform distribution [40];

3.3. Combiner

This step aims at combining the n NN models. In this process, the NNs' outputs are combined using predictions of the testing data set. This paper uses the main combination strategies reported in the literature: *mean*, *trimmed mean*, *median* [41], and *weighted mean* [36]. Assuming n models, $f_j()$ as output of model j, and $F()$ as the ensemble's output, combination strategies are given by:

1. Mean: the ensemble's output is calculated by averaging the n models' predictions:
$$F(\mathbf{x}_i) = \frac{1}{n}\sum_{j=1}^{n} f_j(\mathbf{x}_i); \quad (1)$$
2. Trimmed mean: the ensemble's output is obtained as the trimmed mean of the n predictors' outputs. Trimmed mean excludes the lowest predictors' outputs and the highest models' outputs before obtaining the *mean*, avoiding extreme outputs. For example, considering a $P\%$ trimmed mean, the *mean* is calculated by removing $P\%/2$ from the highest NN models' outputs and $P\%/2$ from the lowest NN models' outputs. (This work sets $P\%$ as 10%).
3. Median: the ensemble's output is the median among all models' outputs:
$$F(\mathbf{x}_i) = \underset{j=1,\ldots,n}{\text{median}}\{f_j(\mathbf{x}_i)\}; \quad (2)$$
4. Weighted mean: the ensemble's output is calculated through a weighted sum of the models' outputs:
$$F(\mathbf{x}_i) = \sum_{j=1}^{n} w_j \cdot f_j(\mathbf{x}_i); \quad (3)$$
where each weight w_j is related to the accuracy of model j. In this paper, the accuracy of a model j is determined using the MSE in the validation data set, and is calculated using [22]:
$$w_j = \frac{\text{adjusted MSE}_j}{\sum_{k=1}^{n} \text{adjusted MSE}_k}, \quad (4)$$
where the "adjusted MSE_j" is determined by:
$$\text{adjusted MSE}_j = 1 - \text{average MSE}_j, \quad (5)$$
and the average MSE_j is given by:
$$\text{average MSE}_j = \frac{\text{MSE}_j}{\sum_{k=1}^{n} \text{MSE}_k}. \quad (6)$$

To evaluate the ensemble performance, it is used the MSE between the estimated output $F()$ and the actual output y in the testing data set MSE^{test}.

4. Proposed Methodology

This work proposes two different methods for automatic ensemble development: Genetic Algorithm for Designing Neural Network Ensembles (GA-NNE) and Simulated Annealing for Designing Neural Network Ensembles (SA-NNE). The ensemble construction using GA-NNE and SA-NNE is performed by two main steps:

1. Generation of candidate NN models;
2. Selection of a subset of NN models and the best combination strategy for aggregating this subset.

The aim is to produce an ensemble which has good performance when compared to an individual NN performance. This objective is achieved by producing and selecting diverse NN models and then selecting the most suitable combination strategy.

The sub-steps for the generation of candidate NN models are the same as in Subsections 3.1 and 3.2. For generating n candidate NN models, first n different training data sets are generated by applying bootstrap. Then, the most suitable NN's architecture is chosen for each training data set. At the end of this process n candidate NN models are generated.

For defining the methodology proposed for the selection of a subset of NN models and a combination strategy using GA-NNE and SA-NNE algorithms, firstly it is necessary to introduce how the possible solutions are encoded and the *fitness* function.

- Solution encoding: a candidate solution to the problem is encoded as a binary string sequence. The solution contains information about the ensemble of NN models to be designed. The solution structure consists in two parts, as illustrated in Figure 2. The first part is the *model* section, which contains information about the subset of NN models for composing the ensemble. The second part is the *combination type*, which represents the combination strategy to be employed for aggregating the subset of NN models.

  As an example, consider a set of n candidate NN models $\{net_1, net_2, \ldots, net_n\}$, where each *locus* of the *model* part is

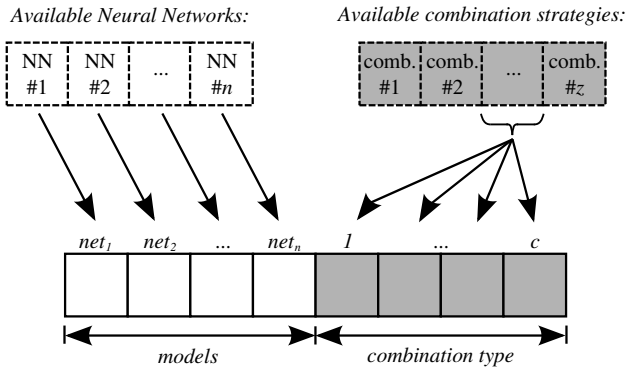
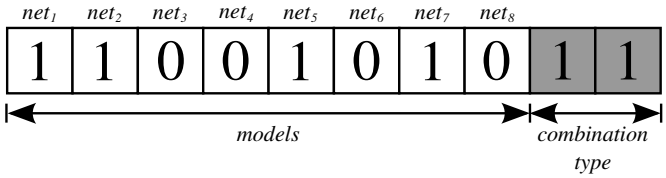
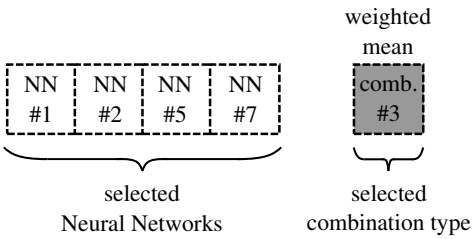


Figure 2: Binary solution representation.



(a) Binary solution representation.



(b) Decoding the solution.

Figure 3: Example of a solution.

related to the absence "0" or presence "1" of a model net_j in the ensemble system. Considering z as the number of combination strategies, and c as the number of bits to represent them, then $c = \lceil \log_2(z) \rceil$, where $\lceil x \rceil$ is the smallest integer not lower than x. Our proposed methodology uses the four combination strategies mentioned in Subsection 3.3: *mean*, *trimmed mean*, *median* and *weighted mean*, with their binary representations being "00", "01", "10" and "11", respectively. Figure 3 illustrates an example of a solution representation using a set of eight NN models $\{net_1, net_2, net_3, \ldots, net_8\}$. Figure 3a shows the binary encoding of the solution and Figure 3b shows the decoding of the same solution. The final subset of NN models to compose the ensemble is $\{net_1, net_2, net_5, net_7\}$ and the selected combination strategy is *weighted mean* (i.e., "11").

- *Fitness* function: for a candidate solution, the *fitness* function is calculated based on the ensemble performance. This is obtained through the aggregation of the subset of NN models using the selected combination strategy. The ensemble evaluation is performed using the predictions on the testing data set, where the *fitness* function is defined by $1/\text{MSE}^{test}$ (see Section 3.3).

Algorithm 1 Genetic Algorithm for Designing Neural Network Ensembles (GA-NNE)

Inputs n; c; *maxgenerations*; $p_m^{\%}$; $p_s^{\%}$; e; m;

1. Produce n candidate models according to Subsection 3.2;
2. Generate randomly an *initial population* P_1 with m individuals;
3. Evaluate the *fitness* of each individual of P_1 with all possible combination strategies;
4. Assign the best combination strategy (on the last c bits) to each individual of P_1 according to the *fitness*;
5. Set generation number as $t \leftarrow 1$;
6. **Repeat**:
   (a) *Select* a percentage $p_s^{\%}$ of the individuals of P_t;
   (b) Perform *crossover* on the selected individuals to generate a new population of offspring O_t;
   (c) *Mutate* randomly with probability $p_m^{\%}$ on the first n bits (the *model* part) of the individuals in O_t;
   (d) Evaluate the *fitness* of each individual in O_t with all possible combinations strategies;
   (e) Assign to each individual in O_t the best combination strategy according to the *fitness*;
   (f) *Select* individuals for the new population P_{t+1}:
      i. Set P'_t as a temporary population $P'_t \leftarrow (O_t \cup P_t)$;
      ii. Apply *elitism* by assigning to P_{t+1} the e individuals of P'_t with the best *fitness*;
      iii. *Select* $(m - e)$ individuals of P'_t for P_{t+1};
   (g) Set $t \leftarrow t + 1$;

   until $t = maxgenerations$.

4.1. Genetic Algorithm for Designing Neural Network Ensembles (GA-NNE)

Genetic Algorithms were proposed by Holland [42] as a global optimization approach inspired by natural evolution and survival of the fittest. GAs use a solution population (*chromosomes*) which evolve by means of *selection*, *crossover* and *mutation* operators [43].

GA-NNE is herein proposed for evolving a population of candidate ensembles [44], as shown in Algorithm 1. GA-NNE starts by setting the parameters n, c, *maxgenerations*, $p_m^{\%}$, $p_s^{\%}$, e and m. In GA-NNE, each chromosome represents an ensemble to be designed. A chromosome is implemented using a binary solution representation shown in Figure 2, and $1/\text{MSE}^{test}$ is the *fitness* function.

In Step 1, a set of n candidate NN models is produced according to Subsection 3.2. In Step 2 an initial population P_1 with m individuals is generated, where each individual has length of $(n + c)$ bits.

Step 3 evaluates each individual of P_1 using the *fitness* function. This is done by evaluating the performance of the subset of models (information contained in the *model* part) using all possible combination strategies. Step 4 assigns the combination strategy with best performance to the chromosome (last

c bits, i.e., *combination type* part). This strategy ensures that the ensemble system will always be designed using the optimal combination type.

In Step 6, the algorithm loops over t generations. Sub-step 6a selects $p_s^{\%}$ of the individuals of population P_t by using Roulette Wheel *Selection* [43]. In this operation, an individual of P_t is picked to be a parent with a probability proportional to its *fitness*.

Sub-step 6b combines the selected parents to compose a new population of offspring O_t. This operation is done using uniform crossover with a random mask, where two parents, $parent_1$ and $parent_2$, generate two offspring, $offspring_1$ and $offspring_2$. The method applies crossover only on the *model* part of each individual. First a random binary mask of length n is produced. If there is 0 in the bit of the mask, the corresponding bit from $parent_1$ is copied to $offspring_1$ and the corresponding bit from $parent_2$ is copied to $offspring_2$. If there is 1 in the bit of the mask, the corresponding bit from $parent_1$ is copied to $offspring_2$ and the corresponding bit from $parent_2$ is copied to $offspring_1$ [45].

Sub-step 6c selects $p_m^{\%}$ individuals from population O_t to be mutated on the *model* part, where $p_m^{\%}$ is the mutation probability. In this operation, for each selected individual one or more bits can be randomly changed to foster diversity in the offspring population. Sub-step 6d and sub-step 6e evaluate each individual of O_t with all possible combination strategies and then the best combination type is assigned to the last c bits.

Individuals for composing the new population P_{t+1} are picked in sub-step 6f. In this operation, *elitism* is applied, by means of which e (with $e < m$) individuals with best *fitness* from $O_t \cup P_t$ are assigned to P_{t+1} (of size m). Then $(m-e)$ individuals of P_{t+1} are selected from $O_t \cup P_t$ using Roulette Wheel *Selection*.

After *maxgenerations*, the individual with best *fitness* of the last population is selected as the final solution to the problem.

4.2. Simulated Annealing for Designing Neural Network Ensembles (SA-NNE)

Simulated Annealing is a meta-heuristic that has proven to be effective in solving many difficult, especially combinatorial, problems [46]. Annealing is a process to change materials' properties. It is performed by heating a material and then freezing it slowly until it crystallizes. As the heating allows the atoms to move randomly, the cooling process should be slow enough to allow atoms to move themselves to lower energy positions. Considering this procedure as an optimization problem, if the atoms' arrangement is achieved with lowest energy level, the arrangement is an optimal solution to the energy minimization problem. SA applies this analogy in order to search for the optimal solution to an optimization problem [47]. The main SA's advantage is the ability to avoid becoming trapped at local optima.

This paper develops a SA based approach for designing ensembles of NN models, as detailed in Algorithm 2. SA-NNE is able to select the best subset of models from a set of candidate models, and then select the best combination type for aggregating this subset. A solution is encoded according to Figure 2 and $1/\text{MSE}^{test}$ is used as the evaluation function, given by *eval*().

Algorithm 2 Simulated Annealing for Designing Neural Network Ensembles (SA-NNE). (*Maximization problem*)

Inputs n; c; T_i; T_f; tr; α; h;

1. Produce n candidate models according to Subsection 3.2;
2. Generate randomly a current solution s_c;
3. Set $T \leftarrow T_i$;
4. **Repeat**:
   (a) Set $q \leftarrow 0$;
   (b) **Repeat**:
      i. Select randomly a new solution s_n in the neighborhood of s_c using a Hamming distance h;
      ii. **if** $eval(s_c) < eval(s_n)$
         A. **then** $s_c \leftarrow s_n$;
         B. **else if** $random[0,1) < \exp\left(\frac{eval(s_n)-eval(s_c)}{T}\right)$
            a. **then** $s_c \leftarrow s_n$;
      iii. Set $q \leftarrow q + 1$;
   until $q = tr$.
   (c) Set $T \leftarrow T \times \alpha$;
   until $T = T_f$.

SA-NNE is started by setting the parameters n, c, T_i, T_f tr, α and h. Step 1 produces n candidate NN models according to the description in Section 3.2. In Step 2, a current solution s_c of size $(n + c)$ bits is randomly generated.

Step 3 assigns the initial temperature T_i to the temperature parameter T. In Step 4, SA-NNE loops over until temperature T is equal to final temperature T_f. Temperature T is gradually decreased according to the cooling ratio $T \leftarrow T \times \alpha$, where α is the cooling factor. In Step 4(b)i, a new solution s_n is randomly selected in the neighborhood of s_c. This neighborhood is defined using a Hamming distance h, that is, only h bits at most change from s_c to s_n. The algorithm always accepts s_n as the new s_c, if s_n is better than s_c. If s_n is worse than s_c, there is a probability of acceptance of s_n that depends on the current value of T and a random value, as described in Step 4(b)ii. Parameter tr is the maximum number of *tries* allowed for a given value of the temperature parameter.

5. Experimental Results

In this Section, experiments to evaluate the proposed GA-NNE and SA-NNE approaches are described. The main objectives of the experiments are: (*i*) to evaluate the performance of a single Neural Network; (*ii*) to analyze the characteristics of the candidate NN models; (*iii*) to evaluate the GA-NNE and SA-NNE performances by varying important parameters; and (*iv*) to compare GA-NNE and SA-NNE to other ensemble techniques.

5.1. Data Set Description

Experiments are performed using two data sets available at Luís Torgo's website [48]: Friedman and Boston Housing. These data sets are adopted because they are widely used in the literature. Therefore they are useful to compare our algorithms with other methodologies.

- Friedman data set: Friedman function is a well-known function for data generation [49]. It uses both non-linear and linear relations between output and inputs. The original Friedman function contains five independent variables:

$$y = 10\sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5 + \epsilon, \quad (7)$$

  where $\epsilon \sim N(0,1)$ is a standard normal deviation. In the data set, the input space is increased by adding other five independent variables $x_6, \ldots, x_{10}$ that do not have influence on y. The total set of variables $\{x_1, x_2, \ldots, x_{10}\}$ is uniformly distributed over $[0, 1]$. The data set includes 40768 samples.

- Boston Housing data set: this data set has been applied extensively in literature to benchmark methods. It contains information collected by the U.S. Census Service about housing in the area of Boston, Massachusetts [50]. The data set consists of 13 independent variables (mainly socio-economic) and 1 output variable (median housing price). The data set is small in size with 506 samples.

Experiments are organized in runs. Each run is evaluated using 10-*fold cross-validation*, where the data set is split into 10 subsets. The result is the average of the results of the 10 subsets, where each subset is in turn used as a testing data set while the samples of the other 9 subsets are randomly divided into a training data set (90%) and a validation data set (10%). At the end of this process, there are 10 *artificial* data sets each of which consists of training, validation and testing data sets. Below, the results of MSE^{test} are given by averaging the MSE of all 10 testing subsets.

5.2. Individual Neural Networks

Firstly, the performance of individual NN models is investigated by altering the number of neurons in the hidden layer. The experiment was done by setting the weight initialization approach and activation function according to popularity, where Nguyen-Widrow was selected as weight initialization technique, and fast hyperbolic tangent and linear as activation functions for the hidden layer and output layer, respectively.

Early stopping criteria is applied as a strategy to control overfitting [51]. Early stopping has been recognized as a good strategy for avoiding overfitting and optimizing the generalization performance of NN models in practice [52]. The main idea is to inspect the test error of a NN model on a independent set using a validation data set, so that when the validation data set error starts to increase the NN training is stopped to avoid overfitting. In this paper, early stopping is employed by the following procedures: set the maximum number of epochs as 500; train each NN calculating the validation data set error every 50 epochs and keep the NN weights at this current point; if the validation data set error has decreased in comparison to the last point, continue the NN training and assign the current weights as final NN weights; if the validation data set error at the current point has risen in comparison to the last point, terminate the NN training and assign the NN weights at the last point as final NN weights.

Figure 4 shows the performance of the individual NN models, where MSE is obtained by 10-*fold cross-validation*. The experiment reveals that NN models with lower number of neurons in the hidden layer are less prone to overfitting and consequently these NN models have better generalization capability. For this reason, in this paper, the maximum number of neurons in the hidden layer is limited to 10.

Moreover, Figure 4a and Figure 4b illustrate the minimum MSE value achieved for a NN in the testing data set over all NN models. For Friedman data set, the minimum value is $7.0\text{x}10^{-3}$ and the NN has 5 neurons in the hidden layer; and, for Boston Housing data set, the minimum value is $8.8\text{x}10^{-3}$ and the NN has 4 neurons in the hidden layer. In the next experiments, a reduction of the minimum value is observed.

Table 1: Results of MSE^{test} using EBP. All the MSE values have been multiplied by 10^3 in the table.

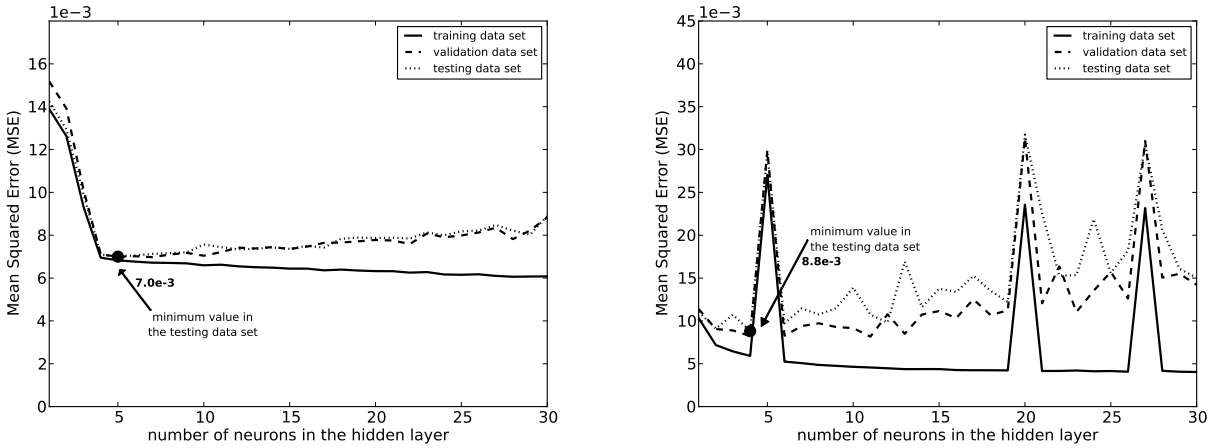
Data Set	Combination Type			
	mean	tr. mean	median	wt. mean
Friedman	2.100	2.100	2.100	2.100
Boston Housing	4.500	4.400	4.400	4.500

5.3. Generation of the Candidate Neural Networks

This Sub-section details the characteristics of the set of candidate Neural Networks to be used in the next experiments. A set of 20 candidate models was generated according to Subsection 3.2 by 10-*fold cross-validation*. The set is used by GA-NNE and SA-NNE approaches. Before doing the experiments with GA-NNE and SA-NNE, all the 20 candidate NN models were aggregated to constitute an ensemble. Therefore, no optimization techniques were employed to select the best subset of models. We use the term Ensemble Before Pruning (EBP) to refer to this ensemble.

EBP was implemented using *mean*, *trimmed mean*, *median*, and *weighted mean* as combination types. Table 1 shows the results of EBP based on the MSE in the testing data set. For Friedman data set, the combination types have the same value of MSE^{test}. On the other hand, *median* and *trimmed median* outperforms *mean* and *weighted mean* for Boston Housing data set. From the results, it is possible to notice that EBP has good generalization ability when compared to the individual models generated in Sub-section 5.2.

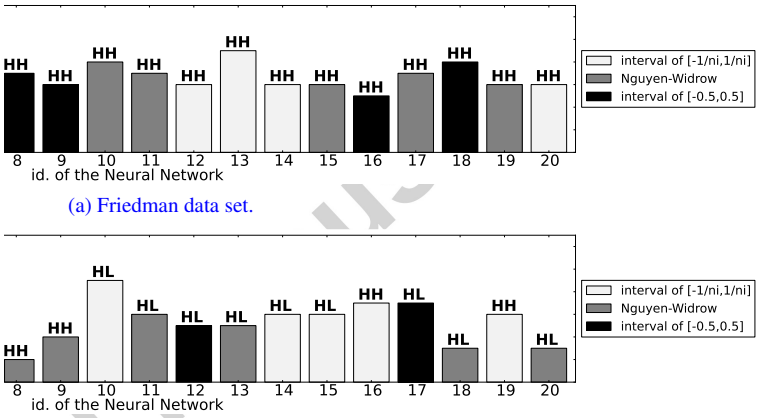
For the Friedman and Boston Housing data set, an *artificial* data set from 10-*fold cross-validation* was randomly chosen to show the characteristics of the candidate Neural Networks. Figure 5 shows the NN's properties, such as the number of neurons



(a) Friedman data set.

(b) Boston Housing data set.

Figure 4: Performance of the individual NN models.



(a) Friedman data set.

(b) Boston Housing data set.

Figure 5: Neural Network's properties of a subset from 10-*fold cross-validation*. See Table 2 for identifying the activation functions in the layers.

Table 2: Abbreviations for the activation functions in the layers.

Abbreviation	Hidden layer	Output layer
HH	fast hyperbolic tangent	fast hyperbolic tangent
LL	linear	linear
HL	fast hyperbolic tangent	linear
LH	linear	fast hyperbolic tangent

in the hidden layer, the weight initialization type and the activation function types for the hidden layer and output layer. For the activation functions in the layers the abbreviations displayed in Table 2 are used. It is observed that the models from Friedman data set have a higher number of neurons in the hidden layer when compared to the models from Boston Housing data set. Moreover, for the Friedman data set all the NN models have the fast hyperbolic tangent as activation function for both the hidden layer and the output layer.

5.4. *Genetic Algorithm for Designing Neural Network Ensembles (GA-NNE)*

After generating the 20 candidate models, Algorithm 1 proceeds with Step 2. Several experiments were performed by varying the GA-NNE's inputs in the following values:

- Mutation probability: $p_m^{\%} \in \{5\%, 10\%, 15\%\}$;
- Selection probability: $p_s^{\%} \in \{60\%, 100\%\}$;
- Population size: $m \in \{20, 40\}$.

The crossover probability is set to 1, the number of mutated bits for the mutation operations is set to 2, the maximum number of generations is set to 500 (*maxgenerations*), and 1 individual is selected by elitism for the next generation (e).

Table 3 and Table 4 show the mean, standard deviation (S.D.), minimum, and maximum of the MSE obtained with GA-NNE on 12 experiments, with 20 runs on each experiment, using the Friedman and Boston Housing data sets, where 10-*fold cross-validation* is performed for each run.

Table 3: Experimental results: mean, standard deviation (S.D.), minimum, and maximum of the MSE obtained with GA-NNE using the Friedman data set on 12 experiments.

No. of the experiment	1	2	3	4	5	6	7	8	9	10	11	12
$p_m^{\%}$	5%	5%	5%	5%	10%	10%	10%	10%	15%	15%	15%	15%
$p_s^{\%}$	60%	100%	60%	100%	60%	100%	60%	100%	60%	100%	60%	100%
m	20	40	20	40	20	40	20	40	20	40	20	40
Mean	1.794	1.789	1.794	1.787	1.791	1.788	1.788	1.787	1.793	1.787	1.789	**1.786**
S.D.	0.005	0.002	0.005	0.002	0.003	0.003	0.002	0.001	0.003	0.001	0.003	**0.001**
Min	1.787	1.785	1.785	1.785	1.786	1.785	1.785	1.785	1.787	1.785	1.785	**1.785**
Max	1.808	1.795	1.803	1.795	1.799	1.797	1.794	1.790	1.798	1.791	1.797	**1.788**

Each experiment is composed of 20 runs, and each run consists of a 10-*fold cross-validation*. All the MSE values have been multiplied by 10^3 in the table.

Table 4: Experimental results: mean, standard deviation (S.D.), minimum, and maximum of the MSE obtained with GA-NNE using the Boston Housing data set on 12 experiments.

No. of the experiment	1	2	3	4	5	6	7	8	9	10	11	12
$p_m^{\%}$	5%	5%	5%	5%	10%	10%	10%	10%	15%	15%	15%	15%
$p_s^{\%}$	60%	100%	60%	100%	60%	100%	60%	100%	60%	100%	60%	100%
m	20	40	20	40	20	40	20	40	20	40	20	40
Mean	2.454	2.445	2.458	2.449	2.447	2.445	2.445	2.445	2.447	2.443	2.445	**2.441**
S.D.	0.008	0.004	0.010	0.008	0.007	0.003	0.004	0.003	0.007	0.003	0.004	**0.002**
Min	2.442	2.439	2.441	2.440	2.439	2.440	2.438	2.439	2.439	2.439	2.439	**2.438**
Max	2.472	2.453	2.478	2.471	2.471	2.452	2.457	2.452	2.461	2.449	2.457	**2.450**

Each experiment is composed of 20 runs, and each run consists of a 10-*fold cross-validation*. All the MSE values have been multiplied by 10^3 in the table.

Considering mean as the metric to evaluate the performance in the experiments, some characteristics are noticed for both data sets. For example, in most experiments, GA-NNE's results improve when the mutation probability ($p_m^{\%}$) increases. Moreover, selection probability $p_s^{\%} = 100\%$ has better performance when compared to $p_s^{\%} = 60\%$. The experiments indicate that improvements are obtained when the population size is $m = 40$.

The experiments with best performance in Table 3 and Table 4 are shown in bold. Considering the best individuals at the end of the 20 runs of experiment 12, the percentage of selection of each combination type are shown in Table 5. For both data sets *median* is the most frequently selected strategy.

Considering again experiment 12, the runs with best MSE performance (i.e., *min* value) are $1.785\text{x}10^{-3}$ and $2.438\text{x}10^{-3}$ for Friedman and Boston Housing data sets, respectively. Figure 6 shows the properties of these best runs according to the average of the 10 test subsets of the 10-*fold cross-validation*. *Mean* is the average of the MSE (i.e., 1/*fitness*) of all the individuals in the population in a generation, and *Best* is the best value of MSE for all individuals of the population in a generation. As can be seen, no important improvements are shown after 300 generations for Friedman data set and 100 generations for Boston Housing data set.

5.5. *Simulated Annealing for Designing Neural Network Ensembles (SA-NNE)*

Using the 20 candidate models, several experiments were carried out by varying the SA-NNE's parameters as follows:

- Cooling factor $\alpha \in \{0.85, 0.90, 0.95\}$;
- Hamming distance $h \in \{1, 2\}$;
- Number of *tries* $tr \in \{1, 2\}$.

The initial temperature T_i is set to 1000 and the algorithm stops when the final temperature T_f is 10^{-20} or less.

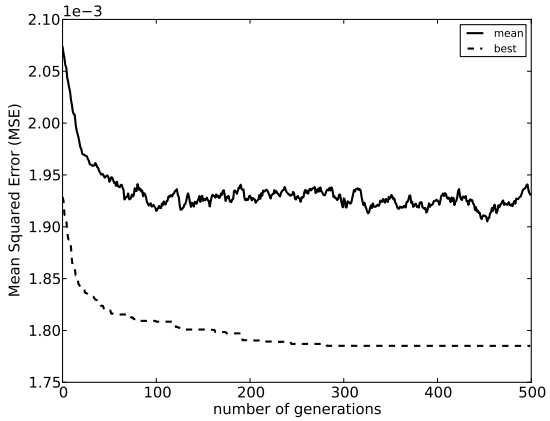
Table 6 and Table 7 show the mean, standard deviation (S.D.), minimum, and maximum of the MSE obtained with the SA-NNE on 12 experiments, with 20 runs on each experiment, using the Friedman and Boston Housing data sets. Considering the mean to evaluate the performance of all experiments, some patterns are observed. For example, in general experiments with a high cooling factor (α) have best results. In general, a Hamming distance $h = 2$ outperforms $h = 1$ and the number of *tries* $tr = 2$ has the best performance.

The experiment with best performance in Table 6 and Table 7 are shown in bold. In this case, the best experiment for both the Friedman data set and Boston Housing data set is experiment 12. Considering the best individuals after 20 runs of these best experiments, the percentage of selection of each combination type are shown in Table 8. Again, *median* is the most frequently selected combination strategy.

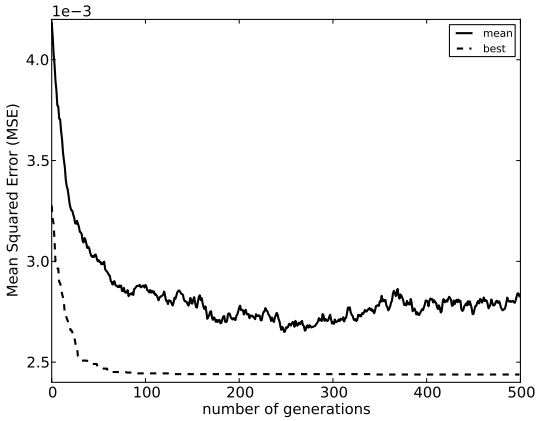
For experiment 12, the run with best MSE performance has a MSE value of $1.785\text{x}10^{-3}$ for Friedman data set and $2.441\text{x}10^{-3}$ for Boston Housing data set. The behavior of these runs are shown in Figure 7. As can be seen, improvements of the MSE follow the decaying of temperature. In the initial *tries*, it is observed that MSE increases its value. This happens because

Table 5: GA-NNE - Percentage of combination type selection on the 20 runs of the best experiment.

Data set	Combination type			
	mean	tr. mean	median	wt. mean
Friedman	0%	0%	**90%**	10%
Boston Housing	0%	0%	**90%**	10%



(a) Friedman data set.



(b) Boston Housing data set.

Figure 6: GA-NNE - mean of the *fitness* of the individuals in the population and *fitness* of the best individual in the population *versus* the number of generations of the best run of the best experiments of Table 3 and Table 4. The average of the 10 test subsets of the 10-*fold cross-validation* is presented.

Table 6: Experimental results: mean, standard deviation (S.D.), minimum, and maximum of the MSE obtained with SA-NNE using the Friedman data set on 12 experiments.

No. of the experiment	1	2	3	4	5	6	7	8	9	10	11	12
Cooling factor (α)	0.85	0.85	0.85	0.85	0.90	0.90	0.90	0.90	0.95	0.95	0.95	0.95
Hamming distance (h)	1	1	2	2	1	1	2	2	1	1	2	2
Number of *tries* (tr)	1	2	1	2	1	2	1	2	1	2	1	2
Mean	1.835	1.838	1.811	1.800	1.836	1.836	1.797	1.791	1.839	1.832	1.796	**1.790**
S.D.	0.010	0.016	0.006	0.007	0.013	0.011	0.006	0.003	0.012	0.009	0.006	**0.003**
Min	1.820	1.818	1.797	1.788	1.811	1.809	1.788	1.785	1.808	1.814	1.785	**1.785**
Max	1.857	1.879	1.822	1.814	1.859	1.864	1.813	1.798	1.855	1.847	1.815	**1.798**

Each experiment is composed of 20 runs, and each run consists of a 10-*fold cross-validation*. All the MSE values have been multiplied by 10^3 in the table.

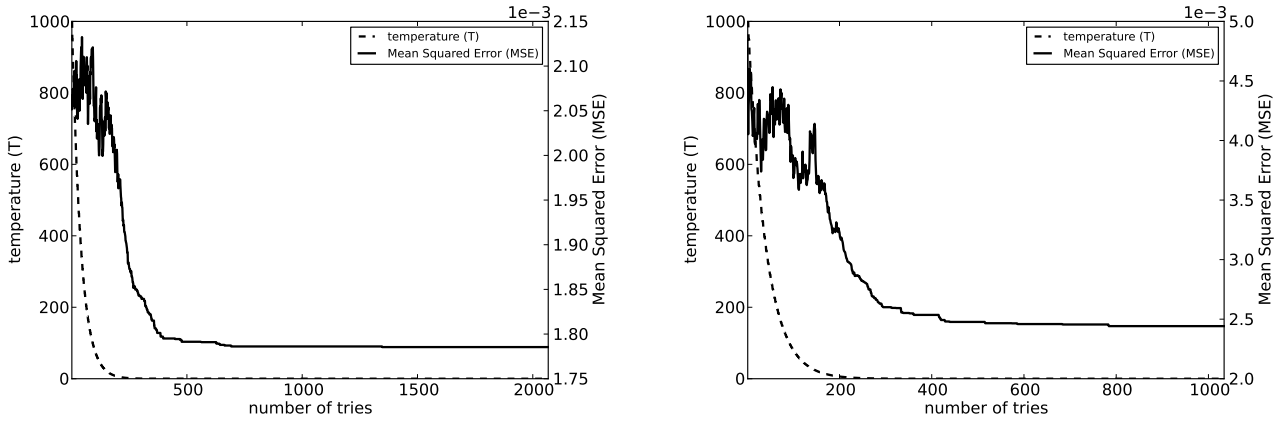
Table 7: Experimental results: mean, standard deviation (S.D.), minimum, and maximum of the MSE obtained with SA-NNE using the Boston Housing data set on 12 experiments.

No. of the experiment	1	2	3	4	5	6	7	8	9	10	11	12
Cooling factor (α)	0.85	0.85	0.85	0.85	0.90	0.90	0.90	0.90	0.95	0.95	0.95	0.95
Hamming distance (h)	1	1	2	2	1	1	2	2	1	1	2	2
Number of *tries* (tr)	1	2	1	2	1	2	1	2	1	2	1	2
Mean	2.656	2.633	2.507	2.480	2.649	2.639	2.484	2.464	2.612	2.602	2.466	**2.454**
S.D.	0.053	0.055	0.024	0.023	0.059	0.075	0.024	0.013	0.048	0.051	0.019	**0.011**
Min	2.558	2.551	2.463	2.448	2.545	2.533	2.456	2.438	2.535	2.534	2.440	**2.441**
Max	2.792	2.743	2.547	2.529	2.746	2.805	2.542	2.487	2.712	2.743	2.509	**2.484**

Each experiment is composed of 20 runs, and each run consists of a 10-*fold cross-validation*. All the MSE values have been multiplied by 10^3 in the table.

Table 8: SA-NNE - Percentage of combination type selection on the 20 runs of the best experiment.

Data set	Combination type			
	mean	tr. mean	median	wt. mean
Friedman	0%	0%	**90%**	10%
Boston Housing	0%	0%	**90%**	10%



(a) Friedman data set.

(b) Boston Housing data set.

Figure 7: SA-NNE - Decay of temperature and MSE *versus* number of *tries*. The average of the 10 test subsets of the 10-*fold cross-validation* is presented.

SA-NNE can more easily accept worse solutions when the temperature is high.

5.6. The Selected Models by GA-NNE and SA-NNE

In this Subsection, characteristics of the selected models by GA-NNE and SA-NNE are detailed. The same *artificial* data set from 10-*fold cross-validation* of Sub-section 5.3 is considered. Here, the results discussed are based on the best experiments from Sub-section 5.4 and Sub-section 5.5.

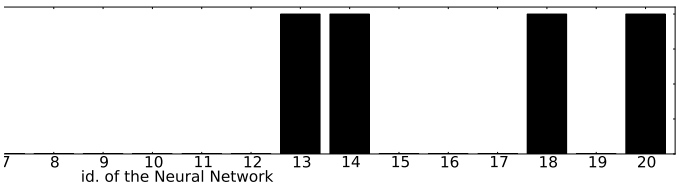
Figures 8a, 9a, 10a and 11a show the statistics of the NN models that participate in the ensembles on the 20 runs, considering the presence of a NN in the final solution of each run. Figures 8b, 9b, 10b and 11b display the accuracy of each NN based on the training, validation and testing data sets. The dashed line represents the MSE^{test} value of the ensemble on the run with minimum MSE value. The underlined NN numbers represent the NN models selected for designing the ensemble of this run. As can be seen, all the cases design ensembles with MSE^{test} values lower than the NN with the lowest MSE^{test} value. This proves that the ensemble is more accurate than any single model in the ensemble.

In Figures 8, 9, 10 and 11, it is observed that the most common models selected by all runs are also selected by the best run, i.e., the one with the minimum MSE value. Common characteristics are noticed for the GA-NNE and SA-NNE approaches. Specifically, GA-NNE and SA-NNE select the same models for designing the ensemble and the same combination type. For the Friedman data set, as depicted in Figures 8b (GA-NNE) and 10b (SA-NNE), the selected NN models for designing the ensemble are $\{2, 13, 14, 18, 20\}$, $MSE^{test} = 1.481\text{x}10^{-3}$ and *median* is the combination type. For the Boston Housing data set, as displayed in Figures 9b (GA-NNE) and 11b (SA-NNE), the selected NN models for aggregating the ensemble are $\{3, 4, 13, 17, 19\}$, $MSE^{test} = 1.231\text{x}10^{-3}$ and *median* is the combination type.

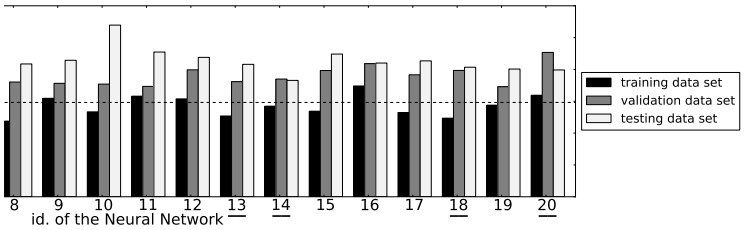
5.7. Comparisons of the Ensemble Systems

In this Sub-section, the proposed GA-NNE and SA-NNE methodologies are compared to other ensemble systems. The ensemble systems include Simple Bagging, GASEN, NCL and AdaBoost.

As mentioned before, Bagging creates an ensemble where each model is trained by a different training data set using bootstrap resampling. Bagging is a common technique applied to GASEN, GA-NNE and SA-NNE. However, as these approaches employ pruning techniques for selecting the best subset of models, in this paper the Bagging strategy is applied for designing an ensemble without pruning technique, i.e. all the candidate NN models are aggregated. Additionally, the NN models have fixed architecture and parameters since most Bagging applications apply this strategy. To distinguish from other approaches (GASEN, GA-NNE and SA-NNE), this ensemble of NN models is called "Simple Bagging". In our experiment, Simple Bagging is composed by 20 NN models using *mean*

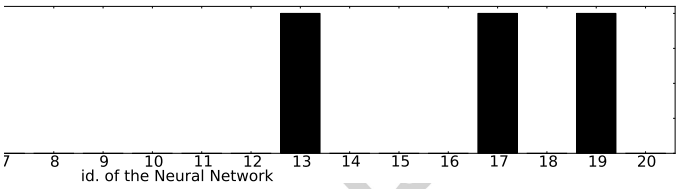


(a) Statistics of NN models that participate in the ensembles using GA-NNE on 20 runs.

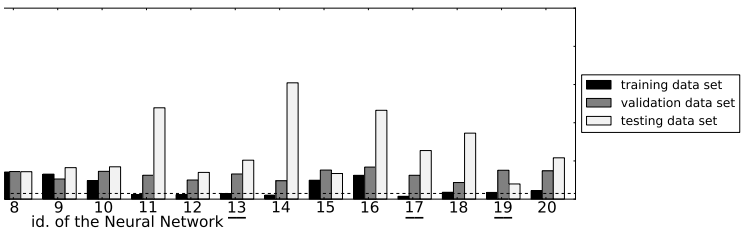


(b) Accuracy of each NN in the data set.

Figure 8: Results of the GA-NNE on the best experiment on the Friedman data set. The dashed line represents the MSE^{test} of the ensemble of best run; underlined numbers represent the selected NN models to design such ensemble.



(a) Statistics of NN models that participate in the ensembles using GA-NNE on 20 runs.



(b) Accuracy of each NN in the data set.

Figure 9: Results of the GA-NNE on the best experiment on the Boston Housing data set. The dashed line represents the MSE^{test} of the ensemble of best run; underlined numbers represent the selected NN models to design such ensemble.

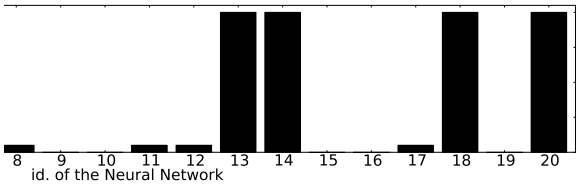
as combination strategy. The Nguyen-Widrow method is employed for weight initialization. Fast hyperbolic tangent and linear activation functions are used for the hidden layer and output layer, respectively.

The number of neurons in the hidden layer was chosen using the experiment in Subsection 5.2. Specifically, this number was selected according to the performance in the validation data set *(see Figure 4)*. Early stopping (as detailed in the Subsection 5.2) was chosen for controlling overfitting.

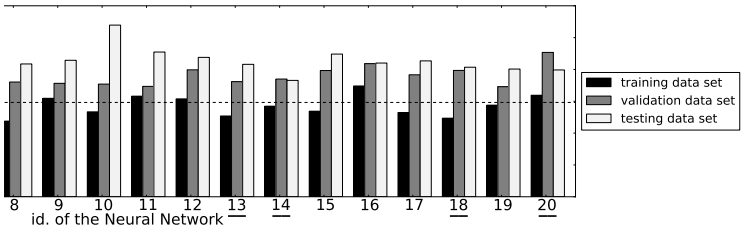
GASEN employs a GA to select the appropriate subset of NN models to constitute the ensemble [21]. GASEN assigns a weight to each model and then models with weights higher than a specified threshold λ_{GASEN} are selected to compose the ensemble. In the GASEN procedure, weights evolve using a GA and the *fitness* function is characterized by the generalization error of the ensemble. Experiments were done using the code available at the website `http://lamda.nju.edu.cn/files/Gasen.zip`.

The experiment and parameter setting were performed according to [21], where the genetic operators are set to the default values of GAOT toolbox [53]. The pre-defined threshold λ_{GASEN} is set to 0.05. The initial set of candidate models is composed by 20 NN models. Each NN has just one hidden layer with five hidden neurons, where the NN is trained using *back-propagation* algorithm. Other NN's parameters are set to the default values, such as hyperbolic tangent sigmoid as activation function for the hidden layer, linear activation function for the output layer, and the training stops when the number of iterations reaches 100. GASEN uses simple average for combining the models' outputs.

NCL produces an ensemble of NN models using negative correlation [54]. The aim is to train the NN models in parallel and use a correlation penalty term λ_{NCL} in their function for assuring specialization and cooperation among the

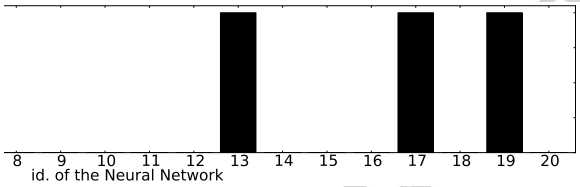


(a) Statistics of NN models that participate in the ensembles using SA-NNE on 20 runs.

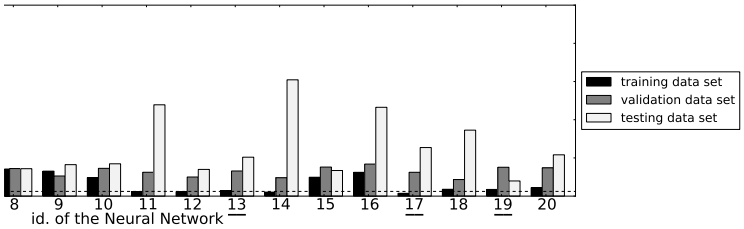


(b) Accuracy of each NN in the data set.

Figure 10: Results of the SA-NNE on the best experiment on the Friedman data set. The dashed line represents the MSE^{test} of the ensemble of best run; underlined numbers represent the selected NN models to design such ensemble.



(a) Statistics of NN models that participate in the ensembles using SA-NNE on 20 runs.



(b) Accuracy of each NN in the data set.

Figure 11: Results of the SA-NNE on the best experiment on the Boston Housing data set. The dashed line represents the MSE^{test} of the ensemble of best run; underlined numbers represent the selected NN models to design such ensemble

Table 9: Comparison of ensemble systems: experimental MSE^{test} results using the Friedman data set.

	AdaBoost	Simple Bagging	NCL	EBP				Pruned Bagging		
				mean	tr. mean	median	wt. mean	GASEN	SA-NNE	GA-NNE
Mean	24.465	7.081	2.438	2.100	2.100	2.100	2.100	1.914	1.790	**1.786**
S.D.	0.000	0.027	0.029	-	-	-	-	0.034	0.003	**0.001**
Min	25.465	7.036	2.396	-	-	-	-	1.862	1.785	**1.785**
Max	25.465	7.138	2.510	-	-	-	-	1.964	1.798	**1.788**

The results are for 20 runs, except for EBP (one run). All the MSE values have been multiplied by 10^3 in the table.

individual NN models. The term λ_{NCL} should assume values in the interval $[0, 1]$. In this paper, the value of λ_{NCL} is determined using $\lambda_{NCL} = \frac{M}{M-1}$, where M is the number of NN models [55]. NCL is tested using the code available at Gavin Brown's website `http://www.cs.man.ac.uk/~gbrown/projects/nc/NCL.zip`. A set of 20 NN models is produced by *back-propagation* algorithm. The models' architecture and parameters are the same of GASEN. NCL also uses simple average for combining the models' outputs.

In this paper, the AdaBoost algorithm uses the GentleBoost logistic regression method as detailed in [25]. A weak learner is selected at each round and the residual displacements from the real output and predicted output are adjusted. In this model, the training samples have equal weight. After R rounds a strong re-

Table 10: Comparison of ensemble systems: experimental MSE^{test} results using the Boston Housing data set.

	AdaBoost	Simple Bagging	NCL	EBP				Pruned Bagging		
				mean	tr. mean	median	wt. mean	GASEN	SA-NNE	GA-NNE
Mean	19.162	9.129	7.817	4.500	4.400	4.400	4.500	6.254	2.454	**2.441**
S.D.	0.000	0.509	0.092	-	-	-	-	0.564	0.011	**0.002**
Min	19.162	8.104	7.629	-	-	-	-	5.457	2.441	**2.438**
Max	19.162	10.176	8.016	-	-	-	-	7.440	2.484	**2.450**

The results are for 20 runs, except for EBP (one run). All the MSE values have been multiplied by 10^3 in the table.

gressor function $F(\mathbf{x})$ is the final output, where the weak learners have the same weights. GentleBoost is implemented using the AdaBoost Toolbox [56], where the weak learners have the same weights. The best performance of the boosting ensemble was achieved in 50 rounds.

Table 9 and Table 10 show the experimental results of MSE^{test} using different ensemble systems on 20 runs, except for EBP where the results are for one run. As pointed out in Sub-section 5.3, EBP is an ensemble with all the candidate NN models used by GA-NNE and SA-NNE. Therefore, EBP is an intermediate ensemble obtained before performing these pruning techniques. For GA-NNE and SA-NNE, the presented experimental results were taken from the best experiments of Sub-section 5.4 and Sub-section 5.5, respectively.

It can be seen that ensembles of NN models (e.g. NCL, Simple Bagging, EBP, GASEN, SA-NNE, and GA-NNE) outperform AdaBoost in the Friedman data set and Boston Housing data set. NCL also obtains more accurate predictions than Simple Bagging and AdaBoost. Negative correlation can produce NN ensembles with good generalization ability (compared to Simple Bagging). However, NCL ensembles have predefined models' architectures making the ensemble with low degree of diversity.

Simple Bagging presents the worst generalization ability when compared to the other ensembles of NN models. The main issue is that Simple Bagging achieves diversity just by manipulating the training data set while the models have the same architecture and parameters. Moreover, Simple Bagging does not employ any strategy for selecting the best subset of models and combination type.

On the other hand, EBP considerably outperforms Simple Bagging in terms of generalization ability. The success is attributed to the several diversity levels employed by EBP, for example, using a different weight initialization and a different architecture for each model in the ensemble.

It is observed that pruned Bagging systems (e.g., GASEN, SA-NNE, and GA-NNE) have better results when compared to ensembling all techniques (e.g., EBP and Simple Bagging), except the lower performance obtained by GASEN for the Boston Housing data set.

As pointed out before, here EBP aggregates all the candidate models used by GA-NNE and SA-NNE. Therefore, it is noticed that the proposed GA-NNE and SA-NNE approaches achieve good results when compared to EBP. The results prove the efficiency of subset selection of models and combination type selection during the ensemble development to obtain good generalization ability.

Table 11: Average number of the selected NN models using different ensemble systems.

Data Set	GASEN	SA-NNE	GA-NNE
Friedman	**4.17**	5.31	5.10
Boston Housing	**4.87**	5.69	5.81

Table 11 shows the average number of selected models by GASEN, GA-NNE and SA-NNE. The results were obtained by averaging the number of selected models of the best individuals after the 20 runs on experiments described in the Table 9 and Table 10. Table 11 shows that SA-NNE and GA-NNE select a higher number of models when compared to GASEN.

6. Conclusions

NNE has established itself as a valuable tool for computational intelligence modeling. The main motivation is that the generalization ability of the system can be significantly improved. Most studies consider key factors during the ensemble development: diversity among the models, training a set of candidate models, subset selection of models and optimal combination strategy. Since there is no automatic procedure to implement these steps, this work proposes and compares two approaches for automatic development of NNE: GA-NNE and SA-NNE.

The main contributions of the proposed methodologies are the selection of the subset of models and combination type providing a high degree of diversity among the models. Firstly, models are generated by starting the learning with different conditions (weight initialization methods), using different training data sets (applying bootstrap), and using models with different learning parameters and architectures. Secondly, two optimization techniques, GA and SA, are used to select the best subset of models and the optimal combination strategy.

GA-NNE and SA-NNE obtained a superior performance when compared to well-known ensemble systems, including Simple Bagging, NCL and AdaBoost and GASEN. This success results from the diversity among the NN models, and the optimal selection of the subset of models and combination type. These are crucial to ensure the ensemble robustness in terms

of generalization ability. Moreover, experiments have shown that GA-NNE and SA-NNE have good performance when compared to a single model and the aggregation of all candidate models (EBP). The results also revealed that GA-NNE and SA-NNE obtained a similar performance. However, SA-NNE selects a slightly larger number of models to compose the ensemble when compared to GA-NNE.

Acknowledgments

Symone Soares is supported by the Fundação para a Ciência a Tecnologia (FCT) under the grant SFRH/BD/68515/2010.

Carlos Henggeler Antunes acknowledges the support of FCT project PEst-C/EEI/UI0308/2011 and QREN Mais Centro Program iCIS project (CENTRO-07-ST24-FEDER-002003).

This work was supported by Project SCIAD/2011/21531 co-financed by QREN, in the framework of the "Mais Centro - Regional Operational Program of the Centro", and by the European Union through the European Regional Development Fund (ERDF).

[1] H. Wang, T. M. Khoshgoftaar, A. Napolitano, Software measurement data reduction using ensemble techniques, Neurocomputing 92 (0) (2012) 124–132.

[2] S. Soares, R. Araújo, P. Sousa, F. Souza, Design and application of soft sensor using ensemble methods, in: Proc. 2011 IEEE 16th Conference on Emerging Technologies Factory Automation (ETFA 2011), 2011, pp. 1–8.

[3] P. Kadlec, B. Gabrys, Local learning-based adaptive soft sensor for catalyst activation prediction, AIChE Journal 57 (5) (2011) 1288–1301.

[4] J. Z. Kolter, M. A. Maloof, Dynamic weighted majority: A new ensemble method for tracking concept drift, in: Proc. Third IEEE International Conference on Data Mining (ICDM 2003), 2003, pp. 123–130.

[5] T. G. Dietterich, Ensemble methods in machine learning, in: J. Kittler, F. Roli (Eds.), Proceedings of the First International Workshop on (MCS 2000) Multiple Classifier Systems, Vol. 1857 of Lecture Notes in Computer Science, Springer-Verlag, Berlin/Heidelberg, 2000, pp. 1–15.

[6] G. Brown, J. Wyatt, R. Harris, X. Yao, Diversity creation methods: A survey and categorisation, Information Fusion 6 (1) (2005) 5–20.

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

[8] A. L. V. Coelho, D. S. C. Nascimento, On the evolutionary design of heterogeneous bagging models, Neurocomputing 73 (16-18) (2010) 3319–3322.

[9] Z. Xie, Y. Xu, Q. Hu, P. Zhu, Margin distribution based bagging pruning, Neurocomputing 85 (15 May) (2012) 11–19.

[10] H. Lee, S. Hong, E. Kim, Neural network ensemble with probabilistic fusion and its application to gait recognition, Neurocomputing 72 (7-9) (2009) 1557–1564.

[11] R. Polikar, Ensemble learning, in: C. Zhang, Y. Ma (Eds.), Ensemble Machine Learning, Springer, 2012, pp. 1–34.

[12] M. D. Redel-Macías, F. Fernández-Navarro, P. A. Gutiérrez, A. J. Cubero-Atienza, C. Hervás-Martínez, Ensembles of evolutionary product unit or rbf neural networks for the identification of sound for pass-by noise test in vehicles, Neurocomputing 0 (0), in press.

[13] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.

[14] Y. Freund, R. Schapire, A short introduction to boosting, Japanese Society for Artificial Intelligence 14 (5) (1999) 771–780.

[15] D. Wang, M. Alhamdoosh, Evolutionary extreme learning machine ensembles with size control, Neurocomputing 102 (0) (2013) 98–110.

[16] R. Caruana, A. Niculescu-Mizil, G. Crew, A. Ksikes, Ensemble selection from libraries of models, in: Proc. Twenty-First International Conference on Machine Learning (ICML 2004), ACM, Banff, Alberta, Canada, 2004, pp. 18–.

[17] G. Martínez-Muñoz, D. Hernández-Lobato, A. Suárez, An analysis of ensemble pruning techniques based on ordered aggregation, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2) (2009) 245–259.

[18] G. Martínez-Muñoz, A. Suárez, Pruning in ordered bagging ensembles, in: Proc. 23rd international Conference on Machine Learning (ICML 2006), ACM, Pittsburgh, Pennsylvania, 2006, pp. 609–616.

[19] P. A. D. Castro, F. J. V. Zuben, Learning ensembles of neural networks by means of a bayesian artificial immune system, IEEE Transactions on Neural Networks 22 (2) (2011) 304–316.

[20] Y.-W. Kim, I.-S. Oh, Classifier ensemble selection using hybrid genetic algorithms, Pattern Recognition Letters 29 (6) (2008) 796–802.

[21] Z.-H. Zhou, J. Wu, W. Tang, Ensembling neural networks: Many could be better than all, Artificial Intelligence 137 (1-2) (2002) 239–263, code available at `http://lamda.nju.edu.cn/files/Gasen.zip`.

[22] S. Dondeti, K. Kannan, R. Manavalan, Genetic algorithm optimized neural networks ensemble for estimation of mefenamic acid and paracetamol in tablets, Acta Chimica Slovenica 52 (4) (2005) 440–449.

[23] A. Chandra, H. Chen, X. Yao, Trade-off between diversity and accuracy in ensemble generation, in: Y. Jin (Ed.), Multi-Objective Machine Learning, Vol. 16 of Studies in Computational Intelligence, Springer Berlin / Heidelberg, 2006, pp. 429–464.

[24] Y. Jia, T. B. Culver, Bootstrapped artificial neural networks for synthetic flow generation with a small data sample, Journal of Hydrology 331 (3-4) (2006) 580–590.

[25] D. Cristinacce, T. F. Cootes, Boosted regression active shape models, in: Proc. British Machine Vision Conference, BMVA Press, 2007, pp. 79.1–79.10.

[26] M. Ries, O. Nemethova, M. Rupp, Performance evaluation of mobile video quality estimators, in: Proc. 15th European Signal Processing Conference, Poznan, Poland, 2007, pp. 159–163.

[27] L. Fortuna, S. Graziani, M. G. Xibilia, Comparison of soft-sensor design methods for industrial plants using small data sets, IEEE Transactions on Instrumentation and Measurement 58 (8) (2009) 2444–2451.

[28] T. Yu-Bo, X. Zhi-Bin, Particle-swarm-optimization-based selective neural network ensemble and its application to modeling resonant frequency of microstrip antenna, in: N. Nasimuddin (Ed.), Microstrip Antennas, In-Tech, 2011, pp. 69–82.

[29] M. Re, G. Valentini, Ensemble Methods: A Review, Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, CRC press, Boca Raton, 2012, Ch. 26, pp. 563–582.

[30] J. Torres-Sospedra, M. Fernández-Redondo, C. Hernández-Espinosa, A research on combination methods for ensembles of multilayer feedforward, in: Proc. IEEE International Joint Conference on Neural Networks (IJCNN 2005), Vol. 2, 2005, pp. 1125–1130.

[31] N. K. Kasabov, Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering, 1st Edition, MIT Press, Cambridge, MA, USA, 1996.

[32] C. H. Aladag, A new architecture selection method based on tabu search for artificial neural networks, Expert Systems with Applications 38 (4) (2011) 3287–3293.

[33] I. Gómez, L. Franco, J. M. Jerez, Neural network architecture selection: Can function complexity help?, Neural Processing Letters 30 (2) (2009) 71–87.

[34] P. L. Rosin, F. Fierens, Improving neural network generalisation, in: Proc. International Geoscience and Remote Sensing Symposium (IGARSS '95), Vol. 2, Firenze, Italy, 1995, pp. 1255–1257.

[35] L. K. Hansen, P. Salamon, Neural network ensembles, IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (10) (1990) 993 –1001.

[36] S. Hashem, Optimal linear combinations of neural networks, Neural Networks 10 (4) (1994) 599–614.

[37] B. Efron, R. J. Tibshirani, An Introduction to the Bootstrap, Chapman & Hall, New York, 1993.

[38] M. T. Hagan, H. B. Demuth, M. Beale, Neural Network Design, PWS Publishing, Boston, MA, USA, 1996.

[39] D. Nguyen, B. Widrow, Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights, in: Proc.

International Joint Conference on Neural Networks (IJCNN 1990), Vol. 3, 1990, pp. 21–26.

[40] J. Škutová, Weights initialization methods for mlp neural networks, Transactions of the VŠB - Technical University of Ostrova, Mechanical Series LIV (2) (2008) 147–152.

[41] R. Polikar, Ensemble based systems in decision making, IEEE Circuits and Systems Magazine 6 (3) (2006) 21–45.

[42] J. H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, MA, USA, 1992.

[43] S. N. Sivanandam, S. N. Deepa, Introduction to Genetic Algorithms, Springer, 2007.

[44] S. Soares, C. Antunes, R. Araújo, A genetic algorithm for designing neural network ensembles, in: Proc. Fourteenth International Conference on Genetic and Evolutionary Computation Conference (GECCO 2012), ACM, Philadelphia, Pennsylvania, USA, 2012, pp. 681–688.

[45] R. L. H. S. E. Haupt, Practical Genetic Algorithms, 2nd Edition, Wiley-Interscience, 2004.

[46] R. Chibante (Ed.), Simulated Annealing, Theory with Applications, Global Optimization, Sciyo, 2010.

[47] Z. Michalewicz, D. B. Fogel, How to Solve it: Modern Heuristics, Springer-Verlag, Berlin, Germany, 2000.

[48] L. Torgo, Regression Datasets, Laboratory of Artificial Intelligence and Decision Support (LIAAD), University of Porto, `http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html` (2011).

[49] J. H. Friedman, Multivariate adaptive regression splines, The Annals of Statistics 19 (1) (1991) 1–67.

[50] D. A. Belsley, E. Kuh, R. E. Welsch, Regression Diagnostics: Identifying Influential Data and Sources of Collinearity, Wiley, 1980.

[51] R. Caruana, S. Lawrence, L. Giles, Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping, in: Proc. Neural Information Processing Systems Conference (NIPS 2000), 2000, pp. 402–408.

[52] D.-I. Jeong, Y.-O. Kim, Rainfall-runoff models using artificial neural networks for ensemble streamflow prediction, Hydrological Processes 19 (19) (2005) 3819–3835.

[53] C. R. Houck, J. A. Joines, M. G. Kay, A genetic algorithm for function optimization: A matlab implementation, Tech. rep., North Carolina State University, Raleigh, NC, USA, `http://read.pudn.com/downloads152/ebook/662702/GAOTV5.PDF`, `http://www.daimi.au.dk/~pmn/Matlab/dochome/toolbox/GAOT/` (1996).

[54] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Networks 12 (10) (1999) 1399–1404.

[55] G. Brown, J. L. Wyatt, P. Tiňo, Managing diversity in regression ensembles, Journal of Machine Learning Research 6 (2005) 1621–1650.

[56] A. Cordiner, Adaboost toolbox - a matlab toolbox for adaptive boosting, Tech. rep., School of Computer Science and Software Engineering, University of Wollongong, Wollongong, Australia, `http://thedeadbeef.files.wordpress.com/2010/07/techreport_boosting.pdf`, `http://dl.dropbox.com/u/6830023/blog/adaboost_toolbox/AdaBoost.zip` (2009).