EVOLUTIONARY ALGORITHMS FOR HYPERPARAMETER TUNING ON NEURAL NETWORKS MODELS

David Orive[(a)], Gorka Sorrosal[(b)], Cruz E. Borges[(c)], Cristina Martin[(d)], Ainhoa Alonso-Vicario[(e)]

[(a-e)] Deusto Institute of Technology - DeustoTech Energy, University of Deusto, Avda. Universidades 24, Bilbao, Spain

[(a)] david.orive@opendeusto.es, [(b)] gsorrosal@deusto.es, [(c)] cruz.borges@deusto.es [(d)] cristina.andonegui@deusto.es
[(e)] ainhoa.alonso@deusto.es

ABSTRACT
In this work we present a comparison of several Artificial Neural Networks weights initialization methods based on Evolutionary Algorithms. We have tested these methods on three datasets: KEEL regression problems, random synthetic dataset and a dataset of concentration of different chemical species from the Bioethanol To Olefins process. Results demonstrated that the tuning of neural networks initial weights improves significantly their performance compared to a random initialization. In addition, several crossover algorithms were tested to identify the best one for the present objective. In the post-hoc analysis there were found significant differences between the implemented crossover algorithms when the network has four or more inputs.

Keywords: Evolutionary Algorithm, Artificial Neural Networks, hyperparameter tuning

1. INTRODUCTION

The hyperparameter tuning of soft computing modelling techniques, such as Artificial Neural Networks (ANN) or Support Vector Machines (SVM), is an important step in the model development. This task used to be performed manually based on expert experience and using a reduced number of trials because of the low computational power available (Bardenet et al. 2013). Presently, computer clusters and GPU processor are able to run more trials, allowing the use of some methods and algorithms to automate the procedure such as Estimation of Distribution Algorithms (EDA) (Nannen and Eiben 2007), local search based algorithms (Hutter and Leyton-brown 2009) or sequential hyperparameter optimization algorithms (Bergstra et al. 2011).

In other areas of knowledge such as image processing, it was proved that state of the art of image classification could be improved not only by building new algorithms but also by tuning the hyperparameters of existing techniques (Pinto et al. 2009; Bergstra et al. 2011). In the case of ANN, the tuning of some hyperparameters such as layers, number of neurons, activation functions or recurrencies has been widely researched (Murata, Yoshizawa, and Amari 1994; Hagiwara, Toda, and Usui 1993; Panchal et al. 2010; Leung et al. 2003). Nonetheless, ANN initial weights are quite often not tuned with the above mentioned hyperparameters but only initialized by training functions. Frequently used training functions like Levenberg-Marquart backpropagation (Marquardt 1963; Hagan and Menhaj 1994) or Bayesian Regularization backpropagation (MacKay 1992), use random weight initializations. However, as final results are dependent of the neural network initial weights, optimization of these parameters should be considered as a technique for the improvement of training functions performance. Moreover, optimizing the initial weights reduces both the possibility of stalling at local optima and the number of iterations required to achieve the objective training error criterion (Yam and Chow 2000; Lee, Oh, and Kim 1991).

The simplest techniques are random samples or grid search strategies over the hyperparameters operating ranges. The first case uses pure random sampling while the second option defines a grid of points evenly distributed over the whole operation range. However, as the number of hyperparameters becomes large these techniques are inviable because their computational cost (Do, Foo, and Ng 2008). Evolutionary Algorithms (EA) are, in general, more efficient as intelligent tuning strategies. Based on biological evolution, EA allow a selective exploration of the operation range using fitness functions that determine which is going to be the next point to be sampled (Holland 1975). In hyperparameter tuning problems, EA have been proved to perform better than grid search techniques based on accuracy-speed ratio (Martino et al. 2011; Friedrichs and Igel 2005; Huang and Wang 2006).

In the case of ANN, many weight initialization methods have been proposed (Yam and Chow 2000). Some of these methods are quite simple, like the methodology proposed by Waghmare, consisting on initializing all weight to the same value (usually 0 or 1) instead of random initialization (Waghmare, Bidwai, and Bhogle 2007). Other researchers use statistical techniques such as the Independent Component Analysis (ICA) to extract useful information from training datasets in order to determine the network weights (Y.-F. Yam et al. 2002), or Partial Least-Squares (PLS) algorithms to determine the initial weights and the optimum node number of hidden layers (Ryan Hsiao, Lin, and Chiang 2003).

EA have been used to improve or to develop new training algorithms for ANN. Some researchers maintain a random initialization and use EAs for the training itself (Montana and Davis 1989), while others use these techniques both for the initialization and training of neural network weights (Al-Shareef and Abbod 2010). Initial weight optimization followed by backpropagation training algorithms has also been used with very good results (Venkatesan, Kannan, and Saravanan 2009; Chang et al. 2012). These works have been implemented only using two-point and four-point crossover operations respectively. Although, the use of EA in the ANN models weight initialization improves the performance compared to a random initialization, the results obtained by the previous algorithms could be further improved by selecting the optimal genetic operators.

This paper demonstrates not only that the tuning of initial weights by EA improves the performance of ANN, but also the existence of significant differences between some genetic operators. The current study allows determining which of the implemented crossover operators for weight initialization provide better results.

This paper is arranged as follows. Section 2 explains the implemented evolutionary algorithm. The datasets used for the experimentation are explained in Section 3. Experimental results are in Section 4. Conclusions and future works are in Section 5.

2. EVOLUTIONARY ALGORITHM

EA are optimization algorithms that operate by modifying a set of candidate solutions (population) according to certain rules called *operators*. In this study we will consider EA as the algorithms that follow the pseudocode of Figure 1. One of the main advantages of the EA is their generality, i.e., they can be used in a broad range of conditions due their simplicity and independence of the underlying problem. In this sense, only the codification of the population and the FitnessOperator depends on the specific problem, the rest of the operators are (almost) independent of it.

In this paper, the problem to be resolved is the optimization of the initial weights and biases of an ANN. In this sense, we have codified a solution as two weights matrices and two biases vectors (Figure 2). The first matrix represents the relationship between the input layer and hidden layer and the second one represents the relationship between the hidden layer and output layer. In the case of the biases, each vector sets the biases for the neurons of each layer. Please note that crossovers operators only cross matrices and vectors that have same code position within the individual.

As our main objective is to assess the performance of different crossovers operators, in order to do so we have defined a broad collection of them using strategies of other EA.

- **Arithmetic crossover (A):** Being P_1 and P_2 the individuals to be crossed and λ a random variable uniformly distributed between [0,1] the two offsprings *C1* and *C2* are defined as:

```
Using the GenesisOperator, a population of
  randomly generated individuals is initialized;
Using the FitnessOperator, evaluate each individual
  of the population;
while a criteria is satisfied do
    Create an empty new population;
    Select between crossover and mutation
      GeneticOperators;
    repeat
        Use SelectionOperator to select the parents
          from the old population;
        Apply a GeneticOperator to generate
          offsprings;
        Calculate the offsprings fitness values;
        With ReproductionOperator select between
          the parents and the offsprings and add them
          to the new population;
    until A new population is filled;
    Apply the EliteOperator to maintain the best
      individual from previous generation;
    Substitute the old population with new
      population;
end
```

Figure 1: Pseudocode of the Evolutionary Algorithm

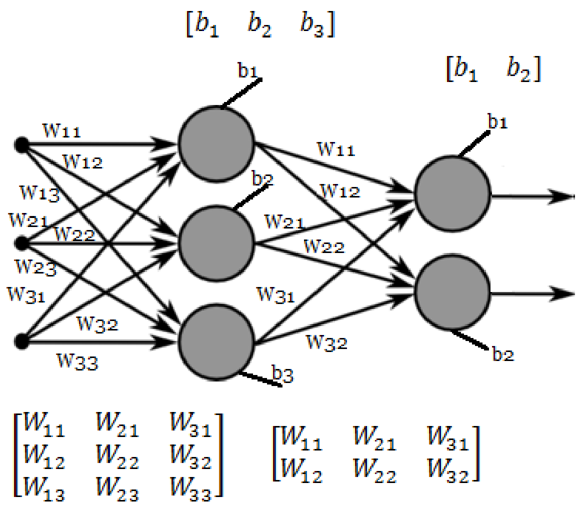

Figure 2: Structure of an individual

$$C_1 = \lambda \cdot P_1 + (1-\lambda) \cdot P_2 \quad (1)$$
$$C_2 = \lambda \cdot P_2 + (1-\lambda) \cdot P_1 \quad (2)$$

- **Differential crossover (D):** Being P_1, P_2 and $P3$ the three individuals to be crossed and λ a random variable uniformly distributed between [0,1] the six offsprings are:

$$C_x = P_y + \lambda(P_z - P_w), \quad (3)$$

  where $\{y, z, w\}$ are all possible permutations of the set $\{1,2,3\}$.

- **Uniform crossover (U):** Given two individuals P_1 and P_2 with L attributes, L binomial random variables λ_i with probability 0.5 are flip. C_1 is made with the features from P_1 such that λ_i are 0 and the features from P_2 such that λ_i are 1. C_2 is made switching P_1 with P_2.

- **Swarm crossover (S):** For each position *i* in the population we denote by P_i^e as the best qualification obtained by the elements at that position. Moreover, P^e will represent the best individual in the population. If the element *P* to be crossed is at position *i* in the population, then the crossover could be defined as:

$$C = \lambda_1 P_i + \lambda_2 (P_i^e - P_i) + \lambda_3 (P^e - P_i), \quad (4)$$

where λ_1, λ_2 and λ_3 are three random real numbers uniformly distributed over the range [0, 1].

- **2 Points crossover (2P):** Given two individuals P_1 and P_2 as a matrix of size *mxn* as attribute and two randomly chosen cross points $t_1, t_2 \in (1, m)$ we define the two point crossover as:

$$C_1 = [P_{1(1,t1)x(1,n)}, P_{2(t1+1,t2)x(1,n)}, P_{1(t2+1,m)x(1,n)}], \quad (5)$$
$$C_2 = [P_{2(1,t1)x(1,n)}, P_{1(t1+1,t2)x(1,n)}, P_{2(t2+1,m)x(1,n)}], \quad (6)$$

where C_1 and C_2 are the offsprings generated by the operator.

The rest of operators are as follow:

- **Mutation Operator (***Non Uniform***):** If *P* is the individual to be mutated and min_i, max_i the inferior and superior limits of its *i* attribute, P_i, the two offsprings are defined as:

$$C_1 = Pi + \Delta(t, max_i - P_i), \quad (7)$$
$$C_2 = P_i - \Delta(t; P_i - min_i), \quad (8)$$

where $\Delta(t, y) = yr(1-t/T)$, being *r* a random number uniformly distributed between [0,1] and *T* the maximum number of generations.

- **Genesis Operator:** The values of the three matrices and vectors are initialized using uniform random variables between [0,1].
- **Fitness Operator**: In order to avoid overfitting (Hawkins 2004) we have used as fitness function the validation error of the ANN over the dataset. The datasets have been divided in two parts: the training set and the validation set. In order to determine the performance of the developed models, k-cross validation has been used. This technique uses an iterative procedure in which the validation data section is changed at each simulation. Training each time with the training dataset section and validating the models with the portion of data that have been excluded from the training. In our case the validation error is based in the Mean Absolute Percentage Error (MAPE) defined by Equation 9:

$$MAPE = 100 \cdot \frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_i - \bar{x_i}}{x_i}\right|, \quad (9)$$

where n is the number of samples in the dataset, x_i is the real value and $\bar{x}_i$ the output of the trained neural network. The validation strategy will be different depending of the dataset so we will discuss this topic further in the next section.

- **Selection Operator** (*Round Robin*): Being *P* the population and *S* the size of the population then the selected individual for $k + 1$ iteration would be $P_{\phi(k+1)}$ where $\phi(k+1) = mod(\phi(k), S) + 1$. In the first iteration $\phi(1)$ is assigned a random value between 1 and *S*.
- **Reproduction Operator** (*N-Different*): The best *N* different individuals are selected where *N* is the number of individuals that will survive.
- **Elite Operator** (*best selection*): Selects the individual with the lowest prediction error. Please note that the last individual selected by this operator will be the solution.

3. EXPERIMENTAL SETUP

The proposed method has been validated on three different datasets: KEEL dataset, a synthetic dataset built by random sampling the space of neural networks and a dataset containing the concentrations over the time of several chemical species in a catalytic reaction.

The KEEL dataset repository was created with the aim of providing the researchers with a set of benchmarks to analyze the behavior of their learning algorithms. There are benchmarks for classification, regression and unsupervised learning (Alcalá-Fdez et al. 2011). The FitnessOperator used in this experiment is the mean of the MAPE errors obtained applying 10-fold cross-validation by initializing the ANN with the weights of an individual.

The synthetic datasets were created by randomly creating feedforward Neural Networks (rNN). A rNN is defined as a neural network with the number of inputs, outputs, layers and neurons selected randomly within some range values. Several random inputs are given to the rNN and outputs are recorded as a dataset. Using this technique infinite datasets can be obtained representing each rNN a different process to model. In order to create realistic conditions, normally distributed noise is added to the outputs of the training set. It was also taken into account the curse of dimensionality phenomenon (Bellman 1961) by increasing the amount of data according to the number of inputs, that is, the more number of inputs the bigger the training dataset has to be. For this reason, the size of the synthetic dataset has been set to depend on the number of inputs following the relation $m = 100n$ where m is the number of instances in the training data and *n* the number of inputs. Rigorously, the relationship should be $m = 100^n$, however this would yield an exponential grow in the training set that would be impossible even to store. In this dataset, the FitnessOperator used is the same as the previous one.

The last used dataset contains the concentration of different chemical species of a Bioethanol To Olefins (BTO) process. This dataset comes from experimental results in a laboratory scale reactor (Gayubo et al. 2010). The BTO process consists on the catalytic transformation of bioethanol, as a substitute of oil, into olefins, which are commodities for the petrochemical industry. The modelling objectives are the olefins production rate curves at different operational conditions. The activity of the catalyst is one of the most important variables to take into account because of its influence in the final production of olefins. The FitnessOperator used in this problem is the results of applying Leave-One-Out Cross Validation to the 23 experiments available. The fitness value is the maximum absolute error obtained from the 23 iterations.

Regarding to evolutionary algorithms, 75 generations with 50 individuals in the population for each experiment are done. In all the experiments the evolutionary algorithm is set with non-uniform mutation in company with a crossover. ANN with only one hidden layer are used with the number of neurons determined by the relation $h = 10n$ where h is the number of neurons in the hidden layer and n represents the number of inputs. Sigmoid function is used as activation function and Levenberg-Marquardt backpropagation is used as a training algorithm with initial weight given by our algorithm.

We have used Nguyen-Widrow (NW) random weight initialization as contrast method. This method generate weights that follow a normal distribution (Nguyen and Widrow 1990). In order to be fair with the rest of the proposed methods, $75 \cdot 50 = 3750$ repetitions are made. Please note that in all the proposed methods we have selected the neural network with best validation error as the solution of the problem.

4. EXPERIMENTAL RESULTS

In order to assess if the differences between proposed methods are significant, we have executed a Friedman test with Hochberg's and Bergmann's post hoc analyses (Demšar 2006; Derrac et al. 2011; Bergmann and Hommel 1988). First, a Friedman test was carried over the results of the different crossovers. The null hypothesis of this test states that the outcomes of each algorithm have the same probability distribution. I.e. this means that all algorithms show the same performance. In the case of positive results, i.e. the p-value is less than 0.05, this test only states that there are differences, but not between which pair of methods. In this case a post hoc analysis is needed. We have used Hochberg's post hoc to compare all methods against NW (as the statistical power of this method is low but it is computationally simple) and Bergmann's method to compare the crossover operators between them (as this method gives us a better statistical power but has exponential complexity).

Table 1 shows the results of the different crossover operators over every dataset of the KEEL regression database. These results show that ANN without hyperparameter tuning (NW) tends to be the poorest (around 7 points worse than the others) while there are little differences between the rest of used techniques. The p-value of the Friedman test carried on over the KEEL regression is $4.5 \cdot 10^{-9}$ way below the significance level so we can assure without any doubt that there are significant differences between at least one of the methods. Table 2 shows the results of Hochberg's post hoc analysis over the KEEL database. Please remember that this method compares all crossovers against NW. The adjusted p-values are way below the significance level given by the Hochberg's post hoc analysis (the second column is way below the third one) so we can conclude that the NW is worse than the rest of proposed methods. Finally, Bergmann's procedure has been carried out in order to test if there are significant differences between any pair of crossovers. In this case the procedure did not found any significant differences between the crossovers.

Table 3 shows the results of the different crossover operators over the 100 synthetic datasets. We only show the mean result for space purposes. Again, the NW seems to be the poorest but in this case the differences are smaller. However, the p-value of the Friedman test carried on over this dataset is $5.1 \cdot 10^{-11}$, again way below the significance level so we can assure without any doubt that there are significant differences between at least one of the methods. Table 2 shows the results of Hochberg's post hoc analysis over this dataset. Once again, the adjusted p-values below the significance level so we can conclude that the NW is worse than the rest of proposed methods. Next, we use the Bergmann's post hoc to study the synthetic datasets. In this case we have divided the samples in two groups: datasets with 3 or less inputs (group A) and the rest (group B). In group A Bergmann's procedure only found significant differences between NW and the rest of crossovers. However, in the case of group B we have found three groups. The first one consists on NW. The second group is composed by U, S, A, and 2P crossovers; between them the Bergmann's procedure has not found any significant difference. The final group consist only on the D crossover that has a result significantly better. Figure 3 shows this information graphically. Please note that algorithms that are linked are not statistically different according to Bergmann's procedure.

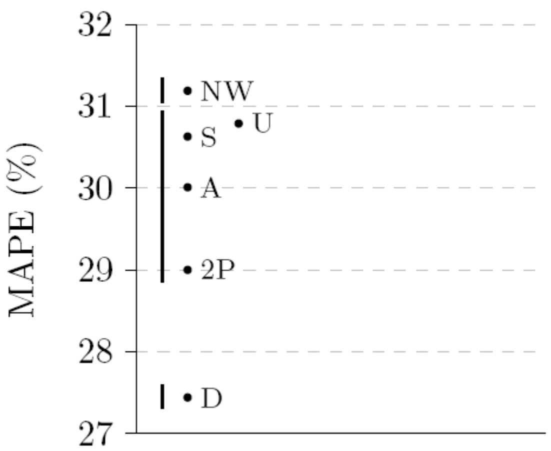

Figure 3: Groups of crossover operators that are statistically different for examples of the synthetic dataset with more than 3 inputs (group B)

Table 1: KEEL Dataset Results with the Different Crossover Algorithms. MAPE (%)

Dataset	A	D	U	2P	S	NW Random
ANACALT	1,99	1,93	2,04	2,01	1,94	2,09
abalone	10,66	10,75	10,46	10,60	10,76	13,27
ailerons	8,32	8,45	8,71	8,51	8,33	9,93
autoMPG6	24,71	24,53	23,91	24,68	24,73	29,37
autoMPG8	10,43	9,32	10,41	10,35	10,18	10,66
california	134,72	152,16	133,92	148,02	152,72	264,34
compactiv	14,27	15,17	14,33	14,56	14,57	15,90
concrete	50,77	50,47	50,55	49,81	51,07	51,84
dee	13,29	13,23	13,37	13,48	13,3	15,6
diabetes	36,33	37,54	33,34	36,28	33,64	53,29
ele-1	15,11	15,37	14,92	15,24	14,92	15,55
ele-2	1,13	1,14	1,12	1,14	1,15	1,16
elevators	92,64	81,65	92,43	91,57	93,42	94,36
forestFires	7,90	8,09	8,04	8,12	8,08	8,20
friedman	24,74	24,77	24,74	24,68	25,05	25,03
house	2,40	2,38	2,41	2,41	2,40	2,41
laser	2,44	2,47	2,50	2,46	2,64	2,84
machineCPU	85,43	79,72	87,42	87,33	81,27	82,81
mortgage	2,07	2,01	2,26	2,26	1,82	2,26
mv	2,48	2,37	2,38	2,44	2,29	2,47
plastic	1,57	1,53	1,59	1,64	1,51	1,67
quake	8,53	8,79	8,39	8,40	8,48	10,36
stock	8,81	8,67	8,42	8,30	8,53	11,22
tic	50,26	54,16	52,68	52,01	51,67	76,19
treasury	25,76	25,91	26,08	25,96	25,34	26,06
wankara	2,62	2,60	2,65	2,62	2,60	2,66
wizmir	20,52	21,04	20,01	20,74	20,09	22,44
MEAN	24,44	24,67	24,41	25,02	24,91	31,63

Table 2: Post Hoc Analysis done for the Different Crossover Algorithms of KEEL datasets and Synthetic datasets

KEEL datasets			Synthetic datasets		
Null hypothesis	p-value	Significance level	Null hypothesis	p-value	Significance level
A = NW?	$2.47 \cdot 10^{-9}$	0.006	D = NW?	$2.15 \cdot 10^{-22}$	0.006
S = NW?	$6.10 \cdot 10^{-9}$	0.007	S = NW?	$5.14 \cdot 10^{-15}$	0.007
U = NW?	$1.10 \cdot 10^{-8}$	0.008	2P = NW?	$4.52 \cdot 10^{-12}$	0.008
D = NW?	$6.08 \cdot 10^{-8}$	0.010	A = NW?	$1.33 \cdot 10^{-11}$	0.010
2P = NW?	$1.43 \cdot 10^{-6}$	0.012	U = NW?	$1.81 \cdot 10^{-8}$	0.050

Table 3: Synthetic Dataset Results with the Different Crossover Algorithms. MAPE (%)

Group	A	D	U	2P	S	NW Random
A	2,91	2,54	2,88	2,79	2,68	3,41
B	30,01	27,44	30,79	29,00	30,63	31,19
Total	16,46	15,00	16,84	15,90	16,66	17,30

Finally, the results of the BTO process modelling using swarm crossover are shown in Table 4. Taking into account that this experiment is very time consuming (more than one week for of computation for each crossover), we can only afford executing this simulation with one of the crossovers. We have selected the swarm crossover as it seems the most promising given the previous results. Fortunately, using this method we have been able to reduce the error of the model in 4 points when using EA for ANN hyperparameter optimization. This represents a significant improvement in this particular dataset.

Table 4: BTO Process Modelling Dataset Results

Error	S	NW
MAPE (%)	18.09	22.04

5. CONCLUSIONS AND FUTURE WORKS

In this paper it is statistically demonstrated that the tuning of artificial neural network initial weights improves significantly the performance compared to Nguyen-Widrow random initialization. Moreover, it is also demonstrated that when the number of inputs is four or more, there are significant differences between the results shows by the different crossover operators used internally in the evolutionary algorithm. For synthetic data the differential crossover present significantly better results than other crossover operator types.

As future work it is necessary to improve the weight initialization techniques by customizing the existing crossover algorithms in order to make an intelligent selection of parents attributes taking into account the neural nodes specialization.

ACKNOWLEDGMENTS
This work has been carried out with the financial support of the Basque Government via the PI2013-40 project and a researchers training grant (COL-2013-1-135).

REFERENCES

Alcalá-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., Garcia, S., Sánchez, L., Herrera, F., 2011. KEEL Data-Mining Software Tool : Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *Journal of Multiple-Valued Logic and Soft Computing,* 17 (2-3), 255–287.

Al-Shareef, A.J., Abbod, M.F., 2010. Neural Networks Initial Weights Optimisation. *IEEE 12th International Conference on Computer Modelling and Simulation*, pp. 57–61. March 24-26, Cambridge (UK).

Bardenet, R., Brendel, M., Kégl, B., Sebag, M., 2013. Collaborative Hyperparameter Tuning. *30th International Conference on Machine Learning*. Vol. 28. June 16-21, Atlanta (USA).

Bellman, R., 1961. *Adaptive Control Processes - A Guided Tour*. Princeton: Princeton University Press.

Bergmann, B., Hommel, G., 1988. Improvements of General Multiple Test Procedures for Reduntant Systems of Hypotheses. *Medizinische Informatik Und Statistik,* 70, 100–115.

Bergstra, J, Bardenet, R., Bengio, Y., Kégl, B., 2011. Algorithms for Hyper-Parameter Optimization. *Advances in Neural Information Processing Systems (NIPS),* 24, 1–9.

Chang, Y., Lin, J., Shieh J., Abbod, M.F., 2012. Optimization the Initial Weights of Artificial Neural Networks via Genetic Algorithm Applied to Hip Bone Fracture Prediction. *Advances in Fuzzy Systems, 2012*, 1–9.

Demšar, J., 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research,* 7, 1–30.

Derrac, J., García, S., Molina, D., Herrera, F., 2011. A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms. *Swarm and Evolutionary Computation,* 1 (1), 3–18.

Do, C.B., Foo, C.S., Ng, A.Y., 2008. Efficient Multiple Hyperparameter Learning for Log-Linear Models. In: J.C. Platt and D. Koller and Y. Singer and S.T. Roweis ed. *Advances in Neural Information Processing Systems, 20,* 377–384.

Friedrichs, F., Igel C.. 2005. Evolutionary Tuning of Multiple SVM Parameters. *Neurocomputing,* 64, 107–117.

Gayubo, A.G., Alonso, A., Valle, B., Aguayo, A.T., Bilbao, J., 2010. Selective Production of Olefins from Bioethanol on HZSM-5 Zeolite Catalysts Treated with NaOH. *Applied Catalysis B: Environmental,* 97 (1-2), 299–306.

Hagan, M.T., Menhaj, M.B., 1994. Training Feedforward Networks with the Marquardt Algorithm. *IEEE Transactions on Neural Networks,* 5 (6), 989–993.

Hagiwara, K., Toda, N., Usui, S., 1993. On the Problem of Applying AIC to Determine the Structure of a Layered Feed-Forward Neural Network. *International Joint Conference on Neural Networks*, pp. 2263–2266. October 25-29, Nagoya (Japan).

Hawkins, D.M., 2004. The Problem of Overfitting. *Journal of Chemical Information and Computer Sciences,* 44 (1), 1–12.

Holland, J.H., 1975. *Adaptation in Natural and Artificial Systems*. Michigan: University of Michigan Press.

Huang, C.L., Wang, C.J., 2006. A GA-Based Feature Selection and Parameters Optimization for Support Vector Machines. *Expert Systems with Applications*, 31 (2), 231–240.

Hutter, F., Leyton-brown, K., 2009. ParamILS : An Automatic Algorithm Configuration Framework. *Journal of Artificial Intelligence Research,* 36, 267–306.

Lee, Y., Oh, S.H., Kim, M.W., 1991. The Effect of Initial Weights on Premature Saturation in Back-Propagation Learning. *International Joint Conference on Neural Networks*, 1 pp. 765–770. July 8-12, Seatle (USA).

Leung, F.H.K., Lam, H.K., Ling, S.H., Tam, P.S., 2003. Tuning of the Structure and Parameters of a Neural Network Using an Improved Genetic Algorithm. *IEEE Transactions on Neural Networks,* 14 (1), 79–88.

MacKay, D.J.C., 1992. Bayesian Interpolation. *Neural Computation*, 4 (3), 415–447.

Marquardt, D.W., 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11 (2), 431–441.

Martino, S.D., Ferrucci, F., Gravino, C., Sarro, F., 2011. A Genetic Algorithm to Configure Support Vector Machines for Predicting Fault-Prone Components. *Lecture Notes in Computer Science*, 247–261.

Montana, D.J., and Davis, L., 1989. Training Feedforward Neural Networks Using Genetic Algorithms. *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pp. 762–767. August 20-25, Detroit (Michigan, USA).

Murata, N., Yoshizawa, S., Amari, S., 1994. Network Information Criterion-Determining the Number of Hidden Units for an Artificial Neural Network Model. *IEEE Transactions on Neural Networks*, 5 (6), 865–872.

Nannen, V., Eiben, A.E., 2007. Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters. *International Joint Conference on Artificial Intelligence*, pp. 975–980. January 6-12, Hyderabad (India).

Nguyen, D., Widrow, B., 1990. Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights. *International Joint Conference of Neural Networks*, pp. 21–26. June 17-21, San Diego (California, USA).

Panchal, G., Ganatra, A., Kosta, Y.P., Panchal, D., 2010. Searching Most Efficient Neural Network Architecture Using Akaike 's Information Criterion (AIC). *International Journal of Computer Applications*, 1 (5), 41–44.

Pinto, N., Doukhan, D., DiCarlo, J.J., Cox, D.D., 2009. A High-Throughput Screening Approach to Discovering Good Forms of Biologically Inspired Visual Representation. *PLoS Computational Biology*, 5 (11), 1–12.

Ryan Hsiao, T.C., Lin, C.W., Chiang, H.K.., 2003. Partial Least-Squares Algorithm for Weights Initialization of Backpropagation Network. *Neurocomputing*, 50, 237–247.

Venkatesan, D., Kannan, K., Saravanan, R. 2009. A Genetic Algorithm-Based Artificial Neural Network Model for the Optimization of Machining Processes. *Neural Computing and Applications*, 18 (2) (January 15), 135–140.

Waghmare, L.M., Nikhil, N.B., Bhogle, P.P. 2007. Neural Network Weight Initialization. *2007 International Conference on Mechatronics and Automation*, pp. 679–681. August 5-8, Harbin (China).

Yam, J.Y.F., Chow, T.W.S., 2000. A Weight Initialization Method for Improving Training Speed in Feedforward Neural Network. *Neurocomputing* 30 (1-4) (January), 219–232.

Yam, Y.F., Leung, C.T., Tam, P.K.S., Siu, W.C., 2002. An Independent Component Analysis Based Weight Initialization Method for Multilayer Perceptrons. *Neurocomputing*, 48 (1-4), 807–818.

AUTHORS BIOGRAPHY

DAVID ORIVE finished his studies in computer engineering at University of Deusto in 2014. Currently he is a research assistant at DeustoTech Institute of Technology working on evolutionary algorithms applied to optimization problems.

GORKA SORROSAL, graduated in Automatic and Industrial Electronic engineering at University of Deusto in 2011, and Master in Control, Automatic and Robotic Engineering at University of the Basque Country in 2013. Ever since 2009, he has been collaborating with DeustoTech Energy in projects related to computer vision and processes optimization. Currently, he is developing his Ph.D. Thesis in optimization and control of the BTO process.

CRUZ E. BORGES In 2005 he received his diploma degree in Mathematics from University of La Laguna (Spain). Then he joined Ph.D program "Mathematics and it applications" in University of Cantabria under the advisor of Luis M. Pardo and J. Luis Montaña. His thesis was related to Root-finding and Symbolic Regression Problems. He has also worked in Genetic Programming and Numerical and Evolutionary Methods for Decision Problems. Now he is working in prediction models of power demand and energy consumptions as well as the introduction of Automatic Modelling in the catalyst design processes in the Energy Unit of DeustoTech.

CRISTINA MARTIN, PhD in Industrial Engineering at the University of Navarra. Her research work has been related to the development of mathematical models, systems analysis and numerical techniques for the uncertainty analysis. Cristina worked for eight years in the CEIT (San Sebastián) where she participated in projects of water quality mathematical modelling, development of systems engineering techniques, Bayesian optimization algorithms, etc. Her career continued with a postdoc at Laval University (Canada), where she participated in various research projects for the design and optimisation of wastewater treatment plants. She is currently researcher at the Department of Environment and Energy of DeustoTech (University of Deusto).

AINHOA ALONSO-VICARIO, PhD in Chemical Engineering from the University of the Basque Country and Head of the Energy Unit of DeustoTech. After his PhD, he joined the R&D Division of Sustainable Energy Production and Leia CDT Foundation where he worked for two years. In November 2009 he joined as a researcher at the Energy Unit of DeustoTech, and since September 2011 he is Head of the Unit. His research interests focus on: the integration of distributed energy sources and CO2 capture processes, and improving the energy efficiency and environmental processes using artificial intelligence.