PSEUDO NEURAL NETWORKS VIA ANALYTIC PROGRAMMING WITH DIRECT CODING OF CONSTANT ESTIMATION

Zuzana Kominkova Oplatkova, Adam Viktorin, Roman Senkerik

Tomas Bata University in Zlin, Faculty of Applied Informatics
Nam T.G. Masaryka 5555, 760 01 Zlin, Czech Republic
{oplatkova, aviktorin, senkerik}@.utb.cz

KEYWORDS
Pseudo neural networks, Analytic programming, Differential evolution.

ABSTRACT

This research deals with a novel approach to classification – pseudo neural networks (PNN). This technique was inspired in classical artificial neural networks (ANN), where a relation between inputs and outputs is based on the mathematical transfer functions and optimised numerical weights. Compared to ANN, the whole structure in PNN, i.e. the relation between inputs and output(s), is fully synthesised by evolutionary symbolic regression tool – analytic programming. Compared to previous synthesised models, the PNN in this paper were synthesised via a new approach to constant estimation inside the analytic programming – direct coding. Iris data was used for the experiments and PNN were used for the synthesis of a complex classifier for more classes. For experimentation, Differential Evolution (de/rand/1/bin) for optimisation in analytic programming (AP) was used.

INTRODUCTION

This paper follows and combines the previous research (Kominkova Oplatkova et al., 2014), (Kominkova Oplatkova et al. 2017a) and (Kominkova Oplatkova et al. 2017b). The new approaches for constant estimation (Kominkova Oplatkova et al. 2017b) inside the Analytic Programming (AP) (Zelinka et al., 2011) caused the interests in the behaviour of the synthesised pseudo neural networks (PNN). The aim is to observe the speed and the achieved quality of the classifier and compare it with the original meta-evolutionary approach.
The pseudo neural networks can be used in more or less the same tasks as artificial neural networks (ANN) like pattern recognition, prediction, control, signal filtering, approximation and others. All ANNs are based on some relation between inputs and output(s), which utilises mathematical transfer functions and optimised weights from training process. The setting-up of layers, number of neurons in layers, estimating of suitable values of weights is a demanding procedure. On account of this fact, pseudo neural networks, which represent the novelty approach using symbolic regression with evolutionary computation, namely Analytic Programming with the direct coding of constant estimation, is proposed in this paper.
Symbolic regression in the context of evolutionary computation means to build a complex formula from basic operators defined by users. The basic case represents a process in which the measured data is fitted and a suitable mathematical formula is obtained analytically. This process is widely known for mathematicians. They use this process when a need arises for a mathematical model of unknown data, i.e. the relation between input and output values. The symbolic regression can also be used for the design of electronic circuits or the optimal trajectory for robots and within other applications (Back et al., 1997), (Koza, 1998), (Koza, 1999), (O'Neill et al., 2003), (Zelinka et al., 2011), (Oplatkova, 2009), (Varacha et al., 2006). Everything depends on the user-defined set of operators. The proposed technique is similar to the synthesis of an analytical form of the mathematical model between input and output(s) in training set used in neural networks. Therefore the technique is called Pseudo Neural Networks.
Initially, John Koza proposed the idea of symbolic regression done by means of a computer in Genetic Programming (GP) (Back et al., 1997), (Koza, 1998), (Koza, 1999). The other approaches are e.g. Grammatical Evolution (GE) developed by Conor Ryan (O'Neill et al., 2003) and here described Analytic Programming (Zelinka et al., 2011), (Oplatkova, 2009), (Varacha et al., 2006), .
The above-described tools were recently commonly used for synthesis of artificial neural networks but in a different manner than is presented here. One possibility is the usage of evolutionary algorithms for optimization of weights to obtain the ANN training process with a small or no training error result. Some other approaches represent the special ways of encoding the structure of the ANN either into the individuals of evolutionary algorithms or into the tools like Genetic Programming. But all of these methods are still working with the classical terminology and separation of ANN to neurons and their transfer functions (Fekiac, 2011). In this paper, the proposed technique synthesizes the structure without a prior knowledge of transfer functions and inner potentials. It synthesizes the relation between inputs and output of training set items used in neural networks so

Proceedings 32nd European Conference on Modelling and Simulation ©ECMS Lars Nolle, Alexandra Burger, Christoph Tholen, Jens Werner, Jens Wellhausen (Editors)
ISBN: 978-0-9932440-6-3/ ISBN: 978-0-9932440-7-0 (CD)

that the items of each group are correctly classified according to the rules for the cost function value.
In general, there are two options:

a) a continuous version of classification when just one node is enough even for more classes, alternatively said a multi-input-single-output (MISO) version (Kominkova Oplatkova et al., 2013), (Kominkova Oplatkova et al., 2016). The number of required classes determines the number of intervals which the range of output values will be divided into.

b) more output nodes are used, which means that AP is launched so many times the number of output nodes is required since the combination of output values predicts the final class. Alternatively, it is possible to call this approach as a multi-input-multi-output (MIMO) version (Kominkova Oplatkova et al., 2014). In this presented case three output nodes are used. Each node represents just one class, i.e. the activated node (output value =1) stands for the appropriate class, the rest of nodes should be deactivated (the output value = 0). The AP is performed for each particular node separately and independently.

The dataset used for training is Iris data set (Machine Learning Repository, Fisher 1936). It is a very known benchmark dataset for classification problem, which was introduced by Fisher for the first time.

As mentioned above, the paper deals with the comparison in the quality of the trained classifier via the original meta evolutionary approach of AP and the new one with the direct coding of the constant estimation.

Firstly, Analytic Programming as a symbolic regression tool with its used strategies is described. Subsequently, Differential Evolution used for the optimisation procedure within Analytic Programming is mentioned. Afterwards, the proposed experiment with differences and obtained results follow and a conclusion finishes the paper.

ANALYTIC PROGRAMMING

Basic principles of the AP were developed in 2001 (Zelinka et al., 2005), (Zelinka et al., 2008), (Zelinka et al., 2011).

The core of AP is based on a special set of mathematical objects and operations. The collection of mathematical objects is the set of functions, operators and terminals, which are usually constants or independent variables. Various functions and terminals can be mixed in this set. This set is called general functional set (GFS) due to its variability of the content. The structure of GFS is created by subsets of functions according to the number of their arguments. For example, GFSall is a set of all functions, operators and terminals, GFS3arg is a subset containing functions with only three arguments, GFS0arg represents only terminals. The subset structure presence in GFS is of vital importance for AP. It is used to avoid synthesis of pathological programs, i.e. programs containing functions without arguments and similar. The content of GFS is dependent only on the user (Zelinka et al., 2005), (Zelinka et al., 2008), (Oplatkova, 2009).

The second part of the AP core is a sequence of mathematical operations, which are used for the program synthesis. These operations are used to transform an individual of a population into a suitable program. Mathematically stated, it is a mapping from an individual domain into a program domain. This mapping consists of two main parts. The first part is called discrete set handling (DSH) (See Figure 1) (Zelinka et al., 2005), (Lampinen and Zelinka, 1999) and the second one stands for security procedures which do not allow synthesising pathological programs. The method of DSH, when used, allows handling arbitrary objects including nonnumerical objects like linguistic terms {hot, cold, dark…}, logic terms (True, False) or other user-defined functions. In the AP DSH is used to map an individual into GFS and together with security procedures creates the mapping mentioned above which transforms the arbitrary individual into a program.

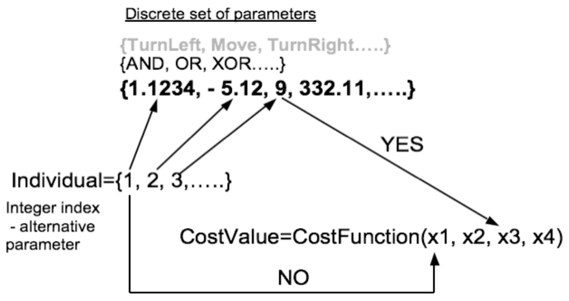
Discrete set of parameters
{TurnLeft, Move, TurnRight…..}
{AND, OR, XOR…..}
{1.1234, - 5.12, 9, 332.11,…..}
YES
Individual={1, 2, 3,…..}
Integer index - alternative parameter
CostValue=CostFunction(x1, x2, x3, x4)
NO

Figure 1: Discrete set handling

AP needs some evolutionary algorithm (Zelinka, 2004) that consists of a population of individuals for its run. Individuals in the population consist of integer parameters, i.e. an individual is an integer index pointing into GFS. The creation of the program can be schematically observed in Fig. 2.

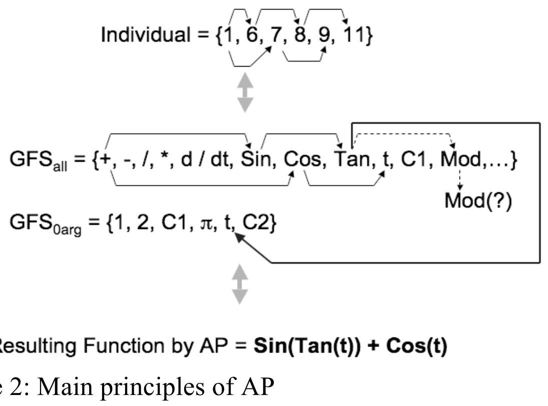
Individual = {1, 6, 7, 8, 9, 11}
GFS_all = {+, -, /, *, d / dt, Sin, Cos, Tan, t, C1, Mod,…}
Mod(?)
GFS_0arg = {1, 2, C1, π, t, C2}
Resulting Function by AP = Sin(Tan(t)) + Cos(t)

Figure 2: Main principles of AP

An example of the process of the final complex formula synthesis (according to the Fig. 2) follows.

The number 1 in the position of the first parameter means that the operator plus (+) from GFS_{all} is used (the end of the individual is far enough). Because the operator + must have at least two arguments, the next two index pointers 6 (sin from GFS) and 7 (cos from GFS) are dedicated to this operator as its arguments.

The two functions, sin and cos, are one-argument functions; therefore the next unused pointers 8 (tan from GFS) and 9 (*t* from GFS) are dedicated to the sin and cos functions. As an argument of cos, the variable *t* is used, and this part of the resulting function is closed (*t* has zero arguments) in its AP development. The one-argument function tan remains, and there is one unused pointer 11, which stands for Mod in GFS_{all}. The modulo operator needs two arguments but the individual in the example has no other indices (pointers, arguments). In this case, it is necessary to employ security procedures and jump to the subset of GFS_{0arg}. The function tan is mapped on *t* from GFS_{0arg} which is in the 11th position, cyclically from the beginning. The detailed description is represented in (Zelinka et al., 2005), (Zelinka et al., 2008), (Oplatkova et al., 2009).

ANALYTIC PROGRAMMING - VERSIONS

The above-described version is the basic one AP_{basic} (Zelinka et al., 2005) - without constant estimation. Except this, AP works with nonlinear fitting version AP_{nf}, meta-evolutionary strategy AP_{meta} and three novel direct approaches AP_{extend} (extended individuals) (Viktorin et al., 2016), $AP_{direct1}$ (Urbanek et al., 2016) and $AP_{direct2}$ (Kominkova Oplatkova et al. 2017a).

The following subsections are dedicated only to meta-evolutionary approach and one direct encoding $AP_{direct2}$ because the paper deals with their comparison in the case of PNN synthesis.

AP uses the general constant *K* (Zelinka et al., 2005) which is indexed during the evolution (1) - (3). The *K* is a terminal, i.e. GFS_{0arg}. So it is used as a standard terminal which is similar for instance to a variable *x* in the evolutionary process (1). When *K* is needed, a proper index is assigned – K_1, K_2, ... K_n (2). Numeric values of indexed *Ks* are estimated (3) via different techniques including AP_{meta},(meta-evolutionary approach with a second/slave evolutionary algorithm) (Zelinka et al., 2005, Oplatkova 2009) and $AP_{direct2}$.

$$\frac{x^2+K}{\pi^K} \tag{1}$$

$$\frac{x^2+K_1}{\pi^{K_2}} \tag{2}$$

$$\frac{x^2+3.156}{\pi^{90.78}} \tag{3}$$

AP_{meta} – meta-evolutionary approach

Generally, metaevolution means the evolution of evolution. Several directions, e.g. the usage of an evolutionary algorithm for tuning or controlling of another evolutionary technique or the evolutionary design of evolutionary algorithms, are discussed for instance in (Diosan, 2009), (Edmons, 2001), (Jones, 2002), (Oplatkova, 2009), (Kordik, 2010), Deugo, 2004), (Eiben, 2007).

In AP_{meta}, the metaevolution means that one evolutionary algorithm drives the main process of symbolic regression and the second is used for the constant estimation. This meta approach in AP is used when coefficients in the found model cannot be simply adjusted because of the character of the problem. It is not possible to interpolate the model to some "measured" values but the obtained solution is used further as a part of the complex technique with the aim to find the quality of the solution and cost value estimation.

AP_{meta} is a time-consuming process and the number of cost function evaluations, which is one of the comparable factors, is usually very high. This fact is given by two evolutionary procedures (Fig. 3).

$$EA_{master} \Rightarrow program \Rightarrow K_{indexing} \Rightarrow EA_{slave} \Rightarrow K_{estimation} \Rightarrow final \cdot solution$$

Figure 3: Schema of AP procedures

EA_{master} is the main evolutionary algorithm for AP, EA_{slave} is the second evolutionary algorithm inside AP. Thus, the number of cost function evaluation (CFE) is given by (4).

$$CFE = EA_{master} * EA_{slave} \tag{4}$$

The following $AP_{direct2}$ approach was developed to decrease the number of cost function evaluations to (5).

$$CFE = EA_{master} \tag{5}$$

$AP_{direct2}$ - direct encoding 2 of K in the individual

This version was developed after the analysis $AP_{direct1}$ behaviour. According to authors, the problematic issues were connected with the neighbourhood of arguments which are responsible for *K* estimation. Since *K* values are dependent only on the decimal part of the argument regardless the integer part of the value in $AP_{direct1}$ (Urbanek et al., 2016), two points placed on the opposite sides of the coordinate system can be neighbours from ind_K (6) point of view. It does not help the evolutionary optimisation process which expects that two points lie next to each other physically in the coordinate system for a successful performance. Surprisingly, (Kominkova Oplatkova et al. 2017b) has not proven such assumption and both variants work more or less in the same high-quality way. Despite this conclusion, authors will use the only $AP_{direct2}$ for further experiments.

$AP_{direct2}$ is based on the $AP_{direct1}$ (Urbanek et al., 2016). It works with a direct encoding in an individual and the difference is in the different computation of ind_K (9). No other slave evolutionary algorithm is necessary for *K* estimation as in AP_{meta}. The variable ind_K (6) works as

a proportional pointer which determines the value from the selected range of *K*.
It takes the value of the not rounded individual as the proportional part in respect of length of all components in GFS_{All}.

$$ind_K = \frac{ind}{Dim(GFS_{All})}, \qquad (6)$$

where $ind = \{x_1, x_2, x_3 \ldots x_n\}$. Dim($GFS_{All}$) means the number of all non-terminals and terminals used in AP. For instance, if GFS_{All}={+, -, /, *, x, K}, the Dim(GFS_{All}) = 6 and the valid range for arguments in the individual is in the interval <1,6>.
The corresponding *K* is then computed easily from (7).

$$K = ind_K * \left| rangeK_{\max} - rangeK_{\min} \right| + rangeK_{\min} \qquad (7)$$

The mapping is done in the standard procedure as in AP_{basic} and general approach of *K* indexation. When *K* is needed, the value in the corresponding position from (7) is directly used.

USED EVOLUTIONARY ALGORITHM - DIFFERENTIAL EVOLUTION

As mentioned above, the Analytic Programming needs an evolutionary algorithm for the optimisation - finding the best shape of the complex formula. This research used Differential Evolution (Price, 2005) in its canonical version DE/Rand/1/Bin. Future research expects to use some other strategies as DE/Best/1/Bin or SHADE which had a good performance in (Viktorin et al., 2016) and (Kominkova Oplatkova et al. 2017b).
DE is a population-based optimisation method that works on real-number-coded individuals (Price, 2005). For each individual $\vec{x}_{i,G}$ in the current generation G, DE generates a new trial individual $\vec{x}'_{i,G}$ by adding the weighted difference between two randomly selected individuals $\vec{x}_{r1,G}$ and $\vec{x}_{r2,G}$ to a randomly selected third individual $\vec{x}_{r3,G}$. The resulting individual $\vec{x}'_{i,G}$ is crossed-over with the original individual $\vec{x}_{i,G}$. The fitness of the resulting individual, referred to as a perturbed vector $\vec{u}_{i,G+1}$, is then compared with the fitness of $\vec{x}_{i,G}$. If the fitness of $\vec{u}_{i,G+1}$ is greater than the fitness of $\vec{x}_{i,G}$, then $\vec{x}_{i,G}$ is replaced with $\vec{u}_{i,G+1}$; otherwise, $\vec{x}_{i,G}$ remains in the population as $\vec{x}_{i,G+1}$. DE is quite robust, fast, and effective, with global optimisation ability. It does not require the objective function to be differentiable, and it works well even with noisy and time-dependent objective functions. Description of used DERand1Bin mutation strategy is presented in (8). Please refer to (Price and Storn 2001, Price 2005) for the description of all other strategies.

$$u_{i,G+1} = x_{r1,G} + F \bullet \left(x_{r2,G} - x_{r3,G} \right) \qquad (8)$$

PROBLEM DESIGN

For this classification problem, iris data set was used (Machine Learning Repository, Fisher 1936). This set contains 150 instances. The half amount was used as training data and the second half was used as testing data. The dataset contains three classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other two; the latter are NOT linearly separable from each other. Each instance has four attributes (sepal length, sepal width, petal length and petal width (Fig. 3a)) and type of class – iris virginica (Fig. 3b), iris versicolor Fig. 3c) and iris setosa (Fig. 3d). The attributes contain real values.

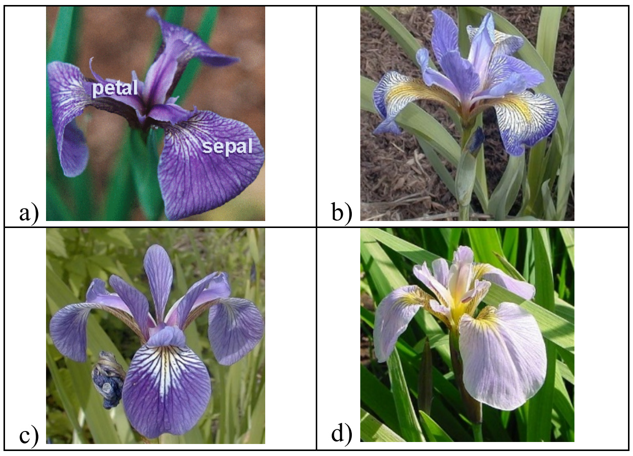

Figure 3: a) Iris - petal and sepal, b) Iris virginica, c) Iris versicolor, d) Iris setosa

Usually, the class is defined by three output nodes in classical artificial neural net and binary code. After the ANN training, an equation can be separated for each output node - relation between inputs, weights and the output node. The training is done in a parallel way - all weights are optimised and the final equations are produced in "one step". In this paper, MIMO approach was used. AP synthesised a pseudo neural net structure between inputs and three outputs independently. The procedure was carried out serially to reach the final solution. The output values were proposed similarly to ANN case, i.e. the combination of zeros and ones. The output vector for iris setosa was (1,0,0), for iris virginica (0,1,0) and for iris versicolor (0,0,1).
Since AP together with differential evolution are used for optimisation – finding of the best shape of the relationship between inputs and output – the cost function which measures the quality of the solution has to be designed. The cost function was selected based on the previous research and the natural character of the problem (9). Since it is necessary to synthesise three independent output equations, the cost function was given the same for all, i.e. if the cv is equal to zero, all training patterns are classified correctly.

$$cv = \sum_{i=1}^{n} \left| requiredOutput - currentOutput \right| \qquad (9)$$

The training patterns were set up according to the following way. If the final classifier model synthesised by AP (one independent expression) determines a class iris setosa, the appropriate item in this group will be assigned to the output value 1 in the training set and the other two kinds of plants will be equal to zero. The same way, the training data were also prepared for the other two independent expressions created by AP.

RESULTS AND DISCUSSION

The paper will compare AP_{meta} and $AP_{direct2}$ strategies with differential evolution DE/Rand/1/Bin. The setting was based on some previous research in this field (Tab. 1 and Tab. 2 for AP_{meta} and Tab. 3. for $AP_{direct2}$). The total Max CFE for AP_{meta} was 6 000 000 according to (4) and for $AP_{direct2}$ only 8 000 as stated in Tab. 3. which is 750 times less. The real used CFEs were even much smaller which means the significant reduction of computation time.

Table 1: DE settings for the main process in AP_{meta}

PopSize	20
F	0.8
CR	0.8
Generations	50
Max. CF Evaluations (CFE)	1000

Table 2: DE settings for meta-evolution in AP_{meta}

PopSize	40
F	0.8
CR	0.8
Generations	150
Max. CF Evaluations (CFE)	6000

Table 3: DE settings in $AP_{direct2}$

PopSize	40
F	0.5
CR	0.8
Generations	2000
Max. CF Evaluations (CFE)	8 000

The set of elementary functions for AP was inspired in the items used in classical artificial neural nets. The components of input vector x contain values of each attribute (x_1, x_2, x_3, x_4).
Basic set of elementary functions for AP:
GFS2arg= +, -, /, *, ^, exp
GFS0arg= x_1, x_2, x_3, x_4, K

All simulations were performed 30 times out. Firstly, statistical results are shown for each class separately. Tab 4. deals with the class 1 which is linearly separable from the rest two classes. It corresponds with iris setosa was (1,0,0). The cost value for training converged to the value zero and also the testing was carried out correctly with the testing error equal to zero.

Tab. 5. works with the class 2 and tab. 6. with class 3 in a similar manner.

Table 4: Statistical results for cost function evaluations when cost value is equal to zero – class 1

	AP_{meta}	$AP_{direct2}$
Min	54 000.0	47.00
Max	288 000.0	123.00
Avg	102 400.0	71.77
Median	75 000.0	66.50
St.Dev.	52 589.2	18.14

Table 5: Statistical results for cost function evaluations when cost value reached only value two (training error is equal to two misclassified items) – class 2

	AP_{meta}	$AP_{direct2}$
Min	$2.868*10^6$	378.00
Max	$5.964*10^6$	987.00
Avg	$4.862*10^6$	758.97
Median	$4.971*10^6$	788.00
St.Dev.	797743.	159.93

Table 6: Statistical results for cost function evaluations when cost value reached only value one (training error is equal to one misclassified items) – class 3

	AP_{meta}	$AP_{direct2}$
Min	$2.268*10^6$	443.00
Max	$5.922*10^6$	959.00
Avg	$4.585*10^6$	740.93
Median	$4.728*10^6$	766.00
St.Dev.	957741.	143.44

From the tables mentioned above, it is visible that the training error was equal to 3 items (it means 4% error, 96% accuracy from all 75 training patterns). The testing error was equal to 7 in those cases (10.67% error, 89.33% accuracy).
The training error for the meta-evolutionary approach was equal to 2 misclassified items (it means 2.66% error, 97.34% success from all 75 training patterns) and testing error equal to 5 misclassified items (8% error, 92% accuracy).
The direct approach is slightly worse than the meta-evolutionary approach in this preliminary results but much faster. Both results suffer a bit from small overfitting but they are comparable with standard ANN approach. The future experiments with the standard validation techniques might help to make a better score.

From carried simulations, several notations of input dependency were obtained for both approaches. The advantage is that equations for each output node can be mixed. Therefore, the best shapes for each output node can be selected and used together for correct behaviour. The following equations (10) - (12) are just examples of the found synthesised expressions of AP_{meta} and (13) - (15) for $AP_{direct2}$.

$$y1 = -692.512\,x3\exp\left(\frac{-x3\left(\left(x1^{438.154}-257.159\right)\left(x3-2.05754\right)-832.086\right)\left(\exp(\exp(x2))+\exp(x2)\right)^{-x4}}{\exp(x2)-\exp(-0.0010047\,x1)}\right) \tag{10}$$

$$y2 = -5.73269\times10^{-48} - \exp\left(-\frac{0.0377017\,x1\left(-387.792^{320.911x4}+x4\right)}{x4}\right) + \frac{-713.828+700.061^{\frac{5.45627\times10^{221}}{-953.763+\exp(x2)}}}{x1} \tag{11}$$

$$y3 = \exp\left(-\frac{950.429}{\exp(x1-1.\,x3\,x4)+\frac{\exp\left(\exp(-\exp(-238.261+x1))^{\exp(\exp(57.0214-x4))}\right)^{5.054576515780\times10^{-391}+x2}}{-16.4604+x3}}\right) \tag{12}$$

$$y1 = -3.24291 + x4^{-\exp(x4)} \tag{13}$$

$$y2 = \frac{1}{\left(\left(-2.35294-\exp(x1)+x1\right)^{x2}\right)^{0.02353}} \tag{14}$$

$$y3 = -\frac{8.25994\exp\left(-2094.48\left(3.52941-x1-x2\right)\right)}{-5.46965+\exp\left(\exp(x1)\right)-\exp(x3)+x4} \tag{15}$$

CONCLUSION

This paper deals with a novel approach to evolutionary synthesised Pseudo neural networks via Analytic programming and its two strategies – meta-evolutionary approach - AP_{meta} and direct coding of K estimation in the individual for speed up the process - $AP_{direct2}$. All simulations were performed with a DE/Rand/1/Bin strategy of differential evolution algorithm for the main optimisation process in AP and also the meta-evolutionary part (second slave algorithm).

The results showed that both approaches are comparable between themselves and also with classical artificial neural networks. However, $AP_{direct2}$ uses significantly less number of cost function evaluations than AP_{meta}, in our case even 750 times less. The speed of training is very important and each reduction of the training time is welcome.

Future plans include a comparison of other evolutionary techniques or their strategies, for instance SHADE since (Viktorin, 2016) presented better results with it in the case of data approximation or swarm algorithms – particle swarm optimisation, self.-organizing migrating algorithm and others.

ACKNOWLEDGEMENT

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme Project no. LO1303 (MSMT-7778/2014), further by the European Regional Development Fund under the Project CEBIA-Tech no. CZ.1.05/2.1.00/03.0089 and by Internal Grant Agency of Tomas Bata University under the Projects no. IGA/CebiaTech/2018/003. This work is also based upon support by COST (European Cooperation in Science & Technology) under Action CA15140, Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO), and Action IC1406, High-Performance Modelling and Simulation for Big Data Applications (cHiPSet). The work was further supported by resources of A.I.Lab at the Faculty of Applied Informatics, Tomas Bata University in Zlin (ailab.fai.utb.cz).

REFERENCES

Back T., Fogel D. B., Michalewicz Z., *Handbook of evolutionary algorithms*, Oxford University Press, 1997, ISBN 0750303921

Deugo D., Ferguson D.: Evolution to the xtreme: Evolving evolutionary strategies using a meta-level approach, Proceedings of the 2004 IEEE congress on evolutionary computation, IEEE Press, Portland, Oregon, pp. 31–38, 2004

Dioşan, L., Oltean, M.: Evolutionary design of evolutionary algorithms. Genetic Programming and Evolvable Machines, Vol. 10, Issue 3, p. 263-306, 2009

Edmonds, B.: Meta-genetic programming: Co-evolving the operators of variation, Elektrik, Vol. 9, Issue 1, pp. 13-29, 2001

Eiben A.E., Michalewicz Z., Schoenauer M., Smith J.E.: Parameter control in evolutionary algorithms, pp. 19–46, Springer, 2007

Jones D.F., Mirrazavi S.K., Tamiz M.: Multi-objective meta-heuristics: An overview of the current state-of-the-art, European Journal of Operational Research, Volume 137, Issue 1, 16 February 2002, Pages 1-9, ISSN 0377-2217.

Kominkova Oplatkova Z., Senkerik R. (2014): MIMO Pseudo Neural Networks for Iris Data Classification. In Advances in Intelligent Systems and Computing. 285. Heidelberg : Springer-Verlag Berlin, 2014, p. 165-172. ISSN 2194-5357. ISBN 978-3-319-06739-1.

Kominkova Oplatkova Z., Senkerik R. (2016): Control Law and Pseudo Neural Networks Synthesized by Evolutionary Symbolic Regression Technique, in Al-Begain K., Bargiela A.: Seminal Contributions to Modelling and Simulation - Part of the series Simulation Foundations, Methods and Applications, pp 91-113, doi: 10.1007/978-3-319-33786-9_9, ISBN: 978-3-319-33785-2.

Kominkova Oplatkova Z., Viktorin A., Senkerik R. (2017a): Comparison of Three Novelty Approaches to Constants (Ks) Handling in Analytic Programming Powered by SHADE, Mendel, Springer Series, in print

Kominkova Oplatkova Z., Viktorin A., Senkerik R., Urbanek T. (2017b): Different Approaches For Constant Estimation In Analytic Programming, ECMS 2017, p. 326 – 334, doi: 10.7148/2017-0326, ISSN 2522-2414, ISBN: 978-0-9932440-4-9

Kordík P., Koutník J., Drchal J., Kovářík O., Čepek M., Šnorek M.: Meta-learning approach to neural network optimization, Neural Networks, Vol. 23, Issue 4, p. 568-582, 2010, ISSN 0893-6080.

Koza J. R. et al., *Genetic Programming III; Darwinian Invention and problem Solving*, Morgan Kaufmann Publisher, 1999, ISBN 1-55860-543-6

Koza J. R., *Genetic Programming,* MIT Press, 1998, ISBN 0-262-11189-6

Lampinen J., Zelinka I., 1999, “New Ideas in Optimization – Mechanical Engineering Design Optimization by Differential Evolution”, Volume 1, London: McGraw-hill, 1999, 20 p., ISBN 007-709506-5.

O’Neill M., Ryan C., *Grammatical Evolution. Evolutionary Automatic Programming in an Arbitrary Language*, Kluwer Academic Publishers, 2003, ISBN 1402074441

Oplatkova Z.: Metaevolution: Synthesis of Optimization Algorithms by means of Symbolic Regression and Evolutionary Algorithms, Lambert Academic Publishing Saarbrücken, 2009, ISBN: 978-3-8383-1808-0

Price K., Storn R. M., Lampinen J. A., 2005, “Differential Evolution : A Practical Approach to Global Optimization”, (Natural Computing Series), Springer; 1 edition.

Price, K. and Storn, R. (2001), *Differential evolution homepage*, [Online]: http://www.icsi.berkeley.edu/~storn/code.html, [Accessed 29/02/2012].

Urbanek T., Prokopova Z., Silhavy R., Kuncar A.: New Approach of Constant Resolving of Analytical Programming. In *30th European Conference on Modelling and Simulation*, 2016, p. 231-236. ISBN 978-0-9932440-2-5.

Viktorin A., Pluhacek M., Kominkova Oplatkova Z., Senkerik R.: Analytical Programming with Extended Individuals. In *30th European Conference on Modelling and Simulation*, 2016, p. 237-244. ISBN 978-0-9932440-2-5.

Volna, E., Kotyrba, M., & Jarusek, R. (2013). Multi-classifier based on Elliott wave’s recognition. Computers & Mathematics with Applications, 66(2), 213-225.

Volna, E., Kotyrba, M., Kominkova Oplatkova Z., Senkerik R. (2018). Elliott waves classification by means of neural and pseudo neural networks. Soft computing, 2018, vol. 22, issue. 6., p. 1803-1813. ISSN 1432-7643 DOI: 10.1007/s00500-016-2097-y

Zelinka et al.: Analytical Programming - a Novel Approach for Evolutionary Synthesis of Symbolic Structures, in Kita E.: Evolutionary Algorithms, InTech 2011, ISBN: 978-953-307-171-8

Zelinka I., Varacha P., Oplatkova Z., *Evolutionary Synthesis of Neural Network*, Mendel 2006 – 12th International Conference on Softcomputing, Brno, Czech Republic, 31 May – 2 June 2006, pages 25 – 31, ISBN 80-214-3195-4

Zelinka I.,Oplatkova Z, Nolle L., 2005. *Boolean Symmetry Function Synthesis by Means of Arbitrary Evolutionary Algorithms-Comparative Study*, International Journal of Simulation Systems, Science and Technology, Volume 6, Number 9, August 2005, pages 44 - 56, ISSN: 1473-8031.

AUTHOR BIOGRAPHIES

ZUZANA KOMINKOVA OPLATKOVA

is an associate professor at Tomas Bata University in Zlin. Her research interests include artificial intelligence, soft computing, evolutionary techniques, symbolic regression, neural networks. She is an author of around 170 papers in journals, book chapters and conference proceedings. Her e-mail address is: oplatkova@utb.cz

ADAM VIKTORIN

was born in the Czech Republic, and went to the Faculty of Applied Informatics at Tomas Bata University in Zlín, where he studied Computer and Communication Systems and obtained his MSc degree in 2015. He is studying his Ph.D. at the same university and the field of his studies are: Artificial intelligence, data mining and evolutionary algorithms. His email address is: aviktorin@utb.cz

ROMAN SENKERIK

was born in the Czech Republic, and went to the Tomas Bata University in Zlin, where he studied Technical Cybernetics and obtained his MSc degree in 2004, Ph.D. degree in Technical Cybernetics in 2008 and Assoc. prof. in 2013 (Informatics). He is now an Assoc. prof. at the same university (research and courses in: Evolutionary Computation, Applied Informatics, Cryptology, Artificial Intelligence, Mathematical Informatics). He is an author of around 290 papers in journals, book chapters and conference proceedings. His email address is: senkerik@utb.cz