

# Neural Network Synthesis Dealing with Classification Problem

PAVEL VAŘACHA

Tomas Bata University in Zlin

Faculty of Applied Informatics

nam. T.G. Masaryka 5555, 760 01 Zlin

CZECH REPUBLIC

varacha@fai.utb.cz http://www.fai.utb.cz

*Abstract:* This article deals with Analytic Programming (AP) which was proven to be highly effective tool of Artificial Neural Network (ANN) synthesis and optimization. AP is used here to obtain optimal ANN which satisfactory solve given problem of classification. The algorithm is theoretically explained and successfully used to perform classification upon real life data of breast cancer diagnosis. Very simple but effective ANN is acquired as a result.

*Key-Words:* Neural Network, classification, Analytic Programming, SOMA, optimization

## 1 Introduction

Classification, such as breast cancer diagnosis (see chapter 5), is very important domain of Artificial Neural Network (ANN) usage. Nevertheless obtaining of optimal ANN which successful deals with provided classification task represents complex problem.

This article describes process of ANN synthesis via symbolic regression and compares it with concurrent solutions. There are two well-known methods: Genetic Programming and Grammatical Evolution, which can both symbolically regress using evolutionary algorithm. However, in this article, more recent and flexible procedure called Analytic Programming (AP) is used here.

AP performed well in many separate cases (for example [1, 2]) together with different evolutionary algorithms (EA) as its “engine”. **Asynchronous** implementation of **SOMA** – **Self-Organizing Migration Algorithm** [3] is applied here to boost AP with the ambition to show unusual electivity of such arrangement.

SOMA is based on a self-organizing behavior of groups of individuals in a “social environment”. It can also be classified as an evolutionary algorithm [4], despite the fact that no new generations of individuals are created during the search (due to philosophy of this algorithm). Only positions of individuals in the searched space are changed during one generation, called a “migration loop”. The algorithm was published in journals, book and presented at international conferences, symposiums as well as in various invitational presentations, for example [5, 6, 7].

## 2 Analytic Programming

Main principle (core) of AP is based on discrete set handling (DSH) (Fig. 1). DSH shows itself as universal interface between EA and a symbolically solved problem. This is why AP can be used almost by any EA.

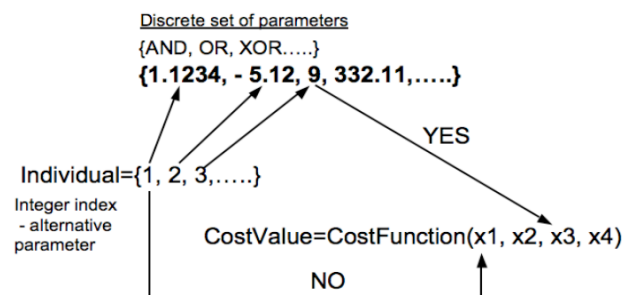


Fig. 1, DSH principle

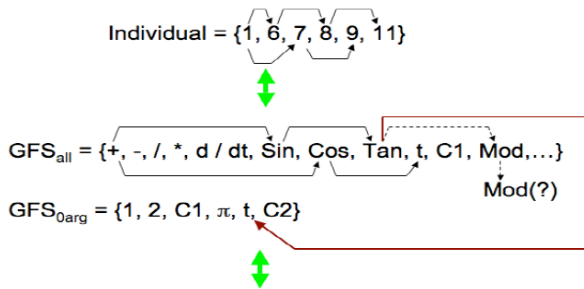
Briefly put, in AP, individuals consist of non-numerical expressions (operators, functions, ...), which are within evolutionary process represented by their integer indexes. Such index then serves like a pointer into the set of expressions and AP uses it to synthesize resulting function-program for Cost Function evaluation.

All simple functions and operators are in so called General Function Set (GFS) divided into groups according to the number of arguments which can be inserted during the evolutionary process to create subsets  $GFS_{3arg}$ ,  $GFS_{2arg}$ ... $GFS_{0arg}$ .

$$\begin{aligned}
 GFS_{all} &= \{+, -, /, *, \text{Sin}, \text{Cos}, \text{Tan}, t, C1, 1, \pi, \\
 &\quad \text{BetaRegularized}, \text{Log}, \text{Mod}, \dots\} \\
 GFS_{3arg} &= \{\text{BetaRegularized}, \dots\} \\
 GFS_{2arg} &= \{+, -, /, *, \text{Log}, \text{Mod}, \dots\} \\
 GFS_{1arg} &= \{\text{Sin}, \text{Cos}, \text{Tan}, \dots\} \\
 GFS_{0arg} &= \{1, C1, \pi, t, \dots\}
 \end{aligned}$$

Fig. 2, Example of GFS and its subsets

Functionality of discrete set handling can be seen on the concrete example in Fig. 3:



Resulting Function by AP =  $\text{Sin}(\text{Tan}(t)) + \text{Cos}(t)$

Fig 3, Main principles of AP

The individual consists of 6 arguments (indices, pointers to GFS). The first index is 1 meaning that + is taken from the set of functions  $GFS_{all}$ . Function *plus* has two arguments, therefore indexes 6 and 7 are arguments of *plus*.

$$6 + 7 \tag{1}$$

Index 6 is then replaced by *Sin* and index 7 by *Cos*.

$$\text{Sin} + \text{Cos} \tag{2}$$

*Sin* and *Cos* are one-argument functions. Then, index 7 follows index 8, which is replaced by *Tan*.

$$\text{Sin}(\text{Tan}) + \text{Cos} \tag{3}$$

*Tan* is also one-argument function. Then, after index 8 the individual takes index 9, which is replaced by *t* and this *t* becomes the argument of *Cos*.

$$\text{Sin}(\text{Tan}) + \text{Cos}(t) \tag{4}$$

But in our case there is a function *Mod*. *Mod* needs an argument to work properly. AP will not allow this, as there is not any other free pointer to be used with the argument. Instead of *Mod*, the AP will jump into the subspace, in this case directly to

$GFS_{0arg}$ . In the  $GFS_{0arg}$  it finds 11th element which is *t*. And by doing so, we get (5) [1].

$$\text{Sin}(\text{Tan}(t)) + \text{Cos}(t) \tag{5}$$

Number of actually used pointers from an individual before synthesized expression is closed is called *depth*.

### 2.1 Constant Processing

Synthesized ANN, programs or formulas may also contain constants “K” which can be defined in  $GFS_0$  or be a part of other functions included in  $GFS_{all}$ . When the program is synthesized, all Ks are indexed, so  $K_1, K_2, \dots, K_n$ , are obtained and then all  $K_n$  are estimated by second EA. In this case the EA is, again, asynchronous implementation of SOMA.

This is especially convenient for an ANN synthesis.  $K_n$  can be interpreted as various weights and thresholds and their optimization by SOMA as ANN learning.

## 3 SOMA All-to-One

Several different versions of SOMA exist, nevertheless, this article is focused on most common **All-to-One** version, which is most suitable for asynchronous parallel implementation. All basic All-to-One SOMA principles important for correct understanding of executed experiment are described below.

### 3.1 Parameter definition

Before starting the algorithm, SOMA’s parameters: Step, PathLength, PopSize, PRT and Cost Function need to be defined. The Cost Function is simply the function which returns a scalar that can directly serve as a measure of fitness. In this case, Cost Function is provided by AP.

### 3.2 Creation of Population

Population of individuals is randomly generated. Each parameter for each individual has to be chosen randomly from the given range <Low, High>.

### 3.3 Migration loop

Each individual from population (PopSize) is evaluated by the Cost Function and the Leader (individual with the highest fitness) is chosen for the current migration loop. Then, all other individuals begin to jump, (according to the Step definition) towards the Leader. Each individual is evaluated after each jump by using the Cost Function. Jumping continues until a new position defined by

the PathLength is reached. The new position  $x_{i,j}$  after each jump is calculated by (6) as is shown graphically in Fig. 1. Later on, the individual returns to the position on its path, where it found the best fitness.

$$x_{i,j}^{MLnew} = x_{i,j,start}^{ML} + (x_{L,j}^{ML} - x_{i,j,start}^{ML})t PRTVector_j \quad (6)$$

where  $t \in <0, \text{by Step to, PathLegth}>$   
and  $ML$  is actual migration loop

Before an individual begins jumping towards the Leader, a random number  $rnd$  is generated (for each individual's component), and then compared with PRT. If the generated random number is larger than PRT, then the associated component of the individual is set to 0 using PRTVector.

$$\text{if } rnd_j < PRT \text{ then } PRTVector_j = 0 \text{ else } 1 \quad (7)$$

where  $rnd \in <0, 1>$   
and  $j = 1, \dots, n_{param}$

j	rnd <sub>j</sub>	PRTVector
1	0,234	1
2	0,545	0
3	0,865	0
4	0,012	1

Table 1, An example of PRTVector for 4 parameters individual with PRT = 0.3

Hence, the individual moves in the N-k dimensional subspace, which is perpendicular to the original space. This fact establishes a higher robustness of the algorithm. Earlier experiments demonstrated that without the use of PRT, SOMA tends to determine a local optimum rather than global one. [8]

### 3.4 Test for stopping condition

If a divergence between current Leader and Leader from the last migration loop is less than defined number, stop and recall the best solution(s) found during the search.

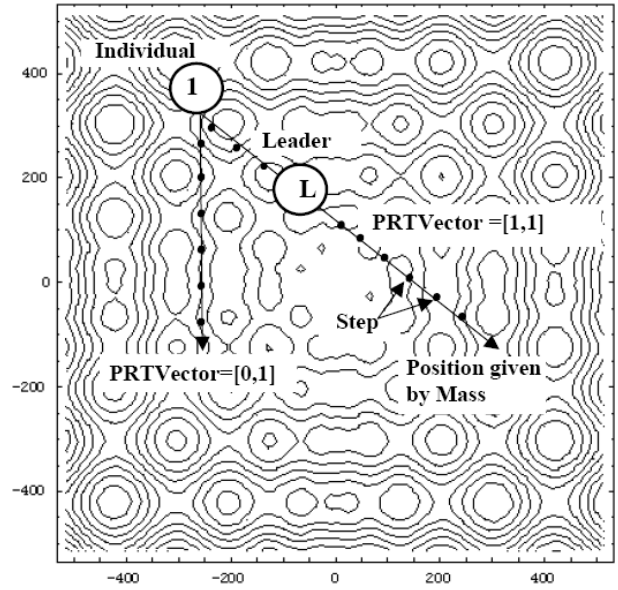


Fig. 4, PRTVector and its action on individual movement [8]

## 4 Neural Network Synthesis

There is a very easy way of using AP for ANN synthesis. [9] The most important part is to define items from which ANN will be composed. In this case GFS contains only three items.

$$GFS_{all} = \{+, AN, K*x\} \quad (8)$$

Most important item of (8) is Artificial Neuron (AN) (9) with hyperbolic tangent as transfer function (10). Weight of output, steepness and thresholds are computed as K in AP (11).

$$GFS1 = \{AN\} \quad (9)$$

$$AN = w * (e^{(2 * \lambda * (input + \phi))} - 1) / ((e^{(2 * \lambda * (input + \phi))} + 1)); \quad (10)$$

$$AN = K_1 * (e^{(2 * K_2 * (input + K_3))} - 1) / ((e^{(2 * K_2 * (input + K_3))} + 1));$$

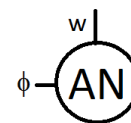


Fig. 5, Graphical example of AN

To allow more inputs into one ANN simple plus operator (12) is used.

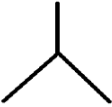
$$GFS_2 = \{+\} \tag{12}$$


Fig. 6, Graphical example of plus operator

Finally, (13) represents weighted input data.

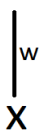
$$GFS_0 = K*x \tag{13}$$


Fig. 7, Graphical example of weighted input

Under such circumstances, translation of an individual to ANN can be easily grasped from Fig. 8.

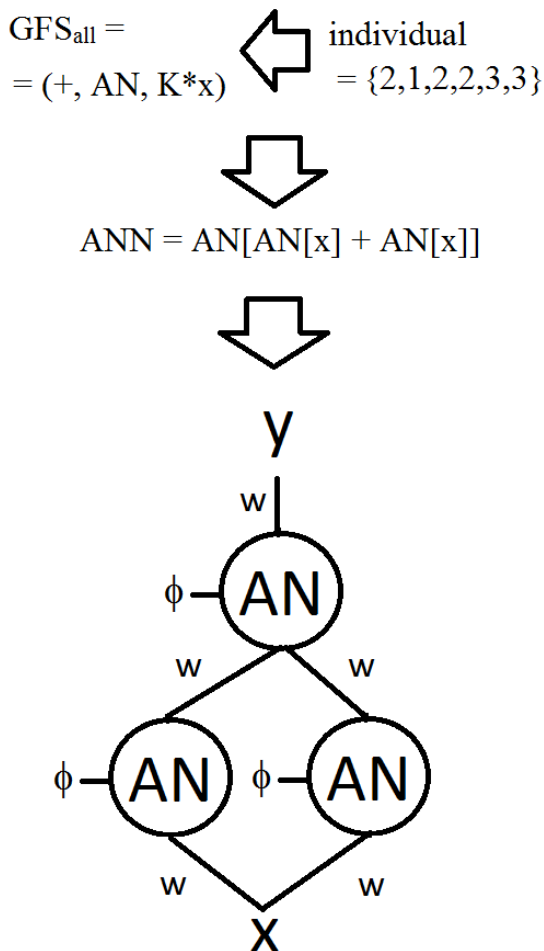


Fig. 8, Translation of an individual to ANN

Whole process is cyclical. Individuals provided by EA are translated to ANNs. ANNs are evaluated in accordance with training data set and their global errors are used to set fitness to these individuals. Consequently, a new generation is chosen and the whole process is repeated in next migration loop.

### 3.1 Reinforced evolution

If ANN of adequate quality cannot be obtained during AP run, AP puts the best ANN it found as a sub ANN into  $GFS_0$  and starts over. This arrangement considerably improves AP ability to find ANN with desirable parameters.

In this experiment (15) (see chapter 6) will be expanded to (14).

$$\tag{14}$$

$$GFS_0 = \{+, AN, K*x0, ANN0, +, AN, K*x1, ANN0, +, AN, K*x2, ANN0, +, AN, K*x3, ANN0, +, AN, K*x4, ANN0, +, AN, K*x5, ANN0, +, AN, K*x6, ANN0, +, AN, K*x7, ANN0, +, AN, K*x8, ANN0\}$$

## 5 Diagnosis of Breast Cancer

Diagnosis of breast cancer is a classification problem introduced in [10]. ANN try to classify a tumor as either benign or malignant based on cell descriptions gathered by microscopic examination.

Input attributes are for instance the clump thickness, the uniformity of cell size and cell shape, the amount of marginal adhesion, and the frequency of bare nuclei.

The dataset includes 699 examples with 9 inputs and 2 outputs. All inputs are continuous; 65.5% of the examples are benign. This makes for entropy of 0.93 bits per example.

This dataset was created based on the "breast cancer Wisconsin" problem from the UCI repository of machine learning databases originally obtained from the University of Wisconsin Hospitals, Madison, from Dr. William H. Wolberg [11].

For the purpose of executed experiment *cancer1* set was chosen in the concrete. Based on [12], fourth ANN optimization methods provide dissimilar mean testing classification error dealing with *cancer1*:

de Falco et al. [13]	2.46%
Prechelt [10]	1.38%
Brameier and Banzhaf [14]	2.18%
The CMAC NN classifier [12]	3.94%

Table 3, ANN mean testing classification error

### 6 Experiment set up

The experiment was prepared in conformity with rules proposed in [10].

To synthesize optimal ANN AP used GFS with equal rate of neurons, connection and inputs:

$$GFS = \{+, AN, K*x0, +, AN, K*x1, +, AN, K*x2, +, AN, K*x3, +, AN, K*x4, +, AN, K*x5, +, AN, K*x6, +, AN, K*x7, +, AN, K*x8\} \tag{15}$$

while Cost function was formulated as: (16)

$$CF = \text{number of wrongly classified examples} + \text{depth}/100;$$

Such approach ensured finding of best possible ANN as well as ANN with minimal structure.

Setting of Asynchronous SOMA used as EA for AP (ANN structure synthesis) can be seen in table 4.

<b>Number of Individuals</b>	48
<b>Individual Parameters</b>	100
<b>Low</b>	0
<b>High</b>	100
<b>PathLength</b>	3
<b>Step</b>	0,11
<b>PRT</b>	1/ depth
<b>Divergence</b>	0.001
<b>Period</b>	3

Table 4, Setting of SOMA used as EA for AP

Table 5 describes setting of SOMA used to optimize  $K_n$  (ANN learning).

<b>Number of Individuals</b>	number of $K_n * 0.5$ (at least 10)
<b>Individual Parameters</b>	100
<b>Low</b>	-10
<b>High</b>	10
<b>PathLength</b>	3
<b>Step</b>	0,11
<b>PRT</b>	1 / number of $K_n$
<b>Divergence</b>	0.001
<b>Period</b>	6

Table 5, Setting of SOMA used to optimize  $K_n$

### 7 Results

During 100 runs of the algorithm ANN structurally described as (17), (18) was found to be the best solution of the given classification problem with

a test classification error 1.14%. Two wrongly classified examples within test set had positions 81 and 87.

$$ANN0 = AN[x5] + x0 + x2 + x3 + AN[x7] \tag{17}$$

$$ANN1 = AN[ANN0 + x3] + x8 \tag{18}$$

Functions (19) and (20) described learned ANN which can be easily tested on *cancer1* publicly provided by [10]:

$$ANN0 = -2,97309632219583 * AN[-1,46365223054944 * (-5,03444335192183 * x5 + 1,76603626076413)] + -7,40609983802126 * x0 + -5,46830267210878 * x2 + -6,94991567402608 * x3 + -5,99052909574962 * AN[1,59467356605207 * (3,68066486608268 * x7 + -3,61373674292757)] \tag{19}$$

$$ANN1 = -2,83643286341635 * AN[-0,179040669733212 * (ANN0 + 0,796079062345568 * x3 + 0,670777686792787)] + -2,95757076519615 * x8 \tag{20}$$

Structural evolution of resulted ANN can be seen on fig. 9

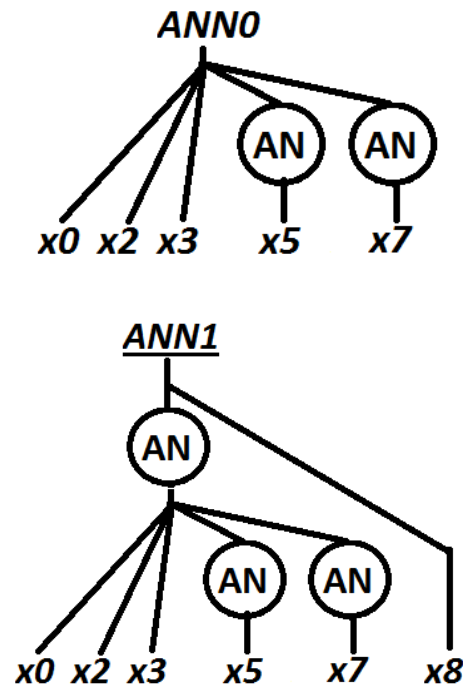


Fig. 9, Resulted ANN structural evolution

## 8 Conclusion

AP proves its ability to synthesize and, in the same time, optimize ANN which effectively classify given task while its structure is minimized.

Best obtained ANN had even for 0.28% better test classification error than mean test classification error of best concurrent method [10].

AP successfully found ANN containing only three AN and totally omitting inputs  $x1$  and  $x4$ . Such performance ratifies AP as efficient tool of ANN optimization.

## 9 Acknowledgment

The work was performed with financial support of research project NPVII-2C06007, by the Ministry of Education of the Czech Republic. [15]

### References:

- [1] Z. Oplatková, I. Zelinka, Creating evolutionary algorithms by means of analytic programming - design of new cost function. In *ECMS 2007, European Council for Modelling and Simulation*, 2007, p. 271-276. ISBN/ISSN: 978-0-9553018-2-7
- [2] Z. Oplatková, I. Zelinka, Investigation on Artificial Ant using Analytic Programming. In *Genetic and Evolutionary Computation Conference*, 2006, p. 949-950. ISBN/ISSN: 1-59593-186-4.
- [3] P. Vařacha, I. Zelinka, Distributed Self-Organizing Migrating Algorithm Application and Evolutionary Scanning. In *Proceedings of the 22nd European Conference on Modelling and Simulation ECMS 2008* Nicosia, 2008. p. 201-206. ISBN/ISSN: 0-9553018-5-8.
- [4] E. Král, V. Dolinay, L. Vašek, P. Vařacha, Usage of PSO Algorithm for Parameters Identification of District Heating Network Simulation Model. In *14th WSEAS International Conference on Systems. Latest Trends on Systems*. Volume II, Rhodes, WSEAS Press (GR) , 2010. p. 657-659. ISBN/ISSN: 978-960-474-214-1.
- [5] M. Červenka, I. Zelinka, Application of Evolutionary Algorithm on Aerodynamic Wing Optimisation. In *Proceedings of the 2nd European Computing Conference*, Venice, WSEAS Press (IT), 2008, ISBN/ISSN: 978-960-474-002-4.
- [6] Z. Oplatková, I. Zelinka, Investigation on Shannon - Kotelnik Theorem Impact on SOMA Algorithm Performance. In *European Simulation Multiconference*, 2005, Riga, ESM , 2005. p. 66-71. ISBN/ISSN: 1-84233-112-4.
- [7] R. Šenkeřík, I. Zelinka, Optimization and Evolutionary Control of Chemical Reactor. In *10th International Research/Expert Conference Trends in the Development of Machinery and Associated Technology*, TMT, Zenica, Bosna and Hercegovina, 2006, p. 1171-1174. ISBN/ISSN: 9958-617-30-7.
- [8] I. Zelinka, *Studies in Fuzziness and Soft Computing*, New York : Springer-Verlag, 2004.
- [9] P. Vařacha, Impact of Weather Inputs on Heating Plant - Agglomeration Modeling. In *Proceedings of the 10th WSEAS Ing. Conf. on Neural Networks*, Athens, WSEAS World Science and Engineering Academy and Science , 2009. p. 159-162. ISBN/ISSN: 978-960-474-065-9.
- [10] L. Prechelt, Proben1—A Set of Neural Network Benchmark Problems and Benchmarking Rules, Universität Karlsruhe, Germany, 1994
- [11] O. L. Mangasarian, W. H. Wolberg: "Cancer diagnosis via linear programming", *SIAM News*, Volume 23, Number 5, September 1990, pp 1 & 18.
- [12] I. de Falco, A.D. Cioppa, E. Tarantino, Discovering interesting classification rules with genetic programming, *Applied Soft Computing* 1 (2002) 257–269.
- [13] M. Brameier, W. Banzhaf, A comparison of linear genetic programming and neural networks in medical data mining, *IEEE Transactions on Evolutionary*
- [14] Jui-Yu Wu, MIMO CMAC neural network classifier for solving classification problems, *Applied Soft Computing*, Volume 11, Issue 2, The Impact of Soft Computing for the Progress of Artificial Intelligence, March 2011, Pages 2326-2333, ISSN 1568-4946,
- [15] B. Chramcov, Forecast of heat demand according the Box-Jenkins methodology for specific locality. In *Latest Trends on Systems*, Rhodes, WSEAS Press (GR) , 2010, p. 252-256, ISBN/ISSN: 978-960-474-199-1.