Hybrid Inductive Models: Topology of Model can Reveal much about Problem

Pavel Náplava, Pavel Kordík

Dept. of Computer Science and Engineering, Karlovo nam. 13, 121 35 Praha 2, Czech Rep.

naplava@email.cz, kordikp@fel.cvut.cz

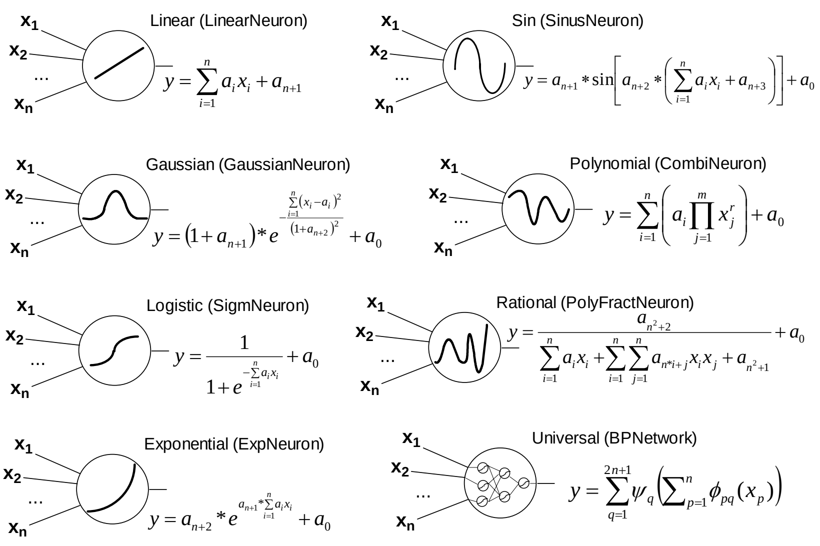
Abstract. *In this paper we study how much information about a problem we can get from topology of inductive models build to model this problem. Our experiments show that for each problem, combination of various units is the most efficient. Also the performance of optimization methods varies on different problems. When we study the topology of models, we can extract much information about problems modeled.*

Keywords

Inductive modeling, hybrid models.

1 Introduction

Hybrid inductive models such as the one produced by the Group of Adaptive Models Evolution (GAME) [2] method can contain more types of units (neurons) within single model (network). In case of GAME method, the list of selected units' types is in the Figure 1.1.



$$y=\sum_{i=1}^{n}a_i x_i + a_{n+1}$$

$$y=a_{n+1}*\sin\left[a_{n+2}*\left(\sum_{i=1}^{n}a_i x_i + a_{n+3}\right)\right]+a_0$$

$$y=(1+a_{n+1})*e^{-\frac{\sum_{i=1}^{n}(x_i-a_i)^2}{(1+a_{n+2})^2}}+a_0$$

$$y=\sum_{i=1}^{n}\left(a_i\prod_{j=1}^{m}x_j^r\right)+a_0$$

$$y=\frac{1}{1+e^{-\sum_{i=1}^{n}a_i x_i}}+a_0$$

$$y=\frac{a_{n^2+2}}{\sum_{i=1}^{n}a_i x_i+\sum_{i=1}^{n}\sum_{j=1}^{n}a_{n*i+j}x_i x_j + a_{n^2+1}}+a_0$$

$$y=a_{n+2}*e^{a_{n+1}*\sum_{i=1}^{n}a_i x_i}+a_0$$

$$y=\sum_{q=1}^{2n+1}\psi_q\left(\sum_{p=1}^{n}\phi_{pq}(x_p)\right)$$

Fig. 1.1. Units of several types competing to survive in the GAME models.

A niching genetic algorithm [3] optimizes structure of the GAME model. Which units survive is based of their fitness derived from their performance on the validation data.

Every data set has different nature (complexity, size, dimensionality, etc). We assumed that for each data set, there will be some type of unit superior to other thus replacing other types in the population of the genetic algorithm. In GAME there are also several optimization methods implemented. The evolution decides which methods are the most appropriate for given problem.

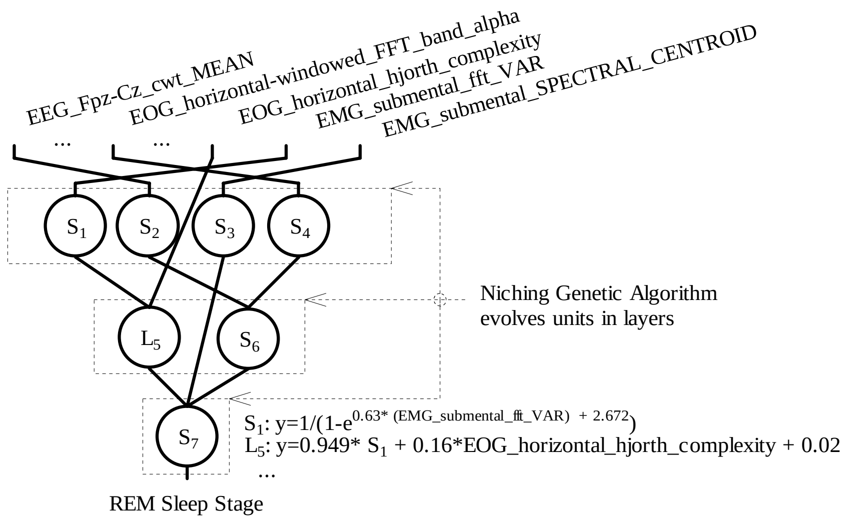


Fig. 1.2. For classification of sleep stages majority of sigmoid transfer function units survived – they are suitable for classifiers. There is also one unit with linear function.

The Figure 1.2 shows the example of a GAME model evolved to classify the sleep stages. No surprise that sigmoid transfer function units dominated. The only unit with linear transfer function would not survive alone, but in combination with sigmoid transfer function is more suitable for this problem.

2 Which units will survive for following problems?

We generated GAME models for four different real world problems. Some of the problems have more target variables. We were interested how many units of certain type will survive in GAME models for particular target variables. Tables on the Figure 2.1 and 2.2 summarize the results.

Type of the unit	Motol-questionary data		Ecoli data							
			Cp		Im		imS			
	Un	Uns	Un	Uns	Un	Uns	Un	Uns	Unc	Uncv
sinus 2	0.6	3.1915	0.5	5.8824	0.4	5.7971	0.2	3.7736	0.425	4.3038
sinus 3	1	5.3191	0.1	1.1765	0.1	1.4493	0.1	1.8868	0.325	3.2911
gauss	0.2	1.0638	0	0.0000	0	0.0000	1.6	30.1887	0.450	4.5570
sinus 1	0.9	4.7872	0.2	2.3529	0.2	2.8986	0.2	3.7736	0.375	3.7975
sinus4	0.8	4.2553	0.1	1.1765	0.1	1.4493	0	0.0000	0.250	2.5316
sinus 5	1.8	9.5745	0.2	2.3529	0.1	1.4493	0	0.0000	0.525	5.3165
sinus 6	1.2	6.3830	0.2	2.3529	0.8	11.5942	0	0.0000	0.550	5.5696
sinus 9	0.7	3.7234	0.3	3.5294	1	14.4928	0.1	1.8868	0.525	5.3165
sinus 10	1.2	6.3830	0.2	2.3529	0.5	7.2464	0	0.0000	0.475	4.8101
sinus 11	1.6	8.5106	0.5	5.8824	0.1	1.4493	0.1	1.8868	0.575	5.8228
sinus 12	2.2	11.7021	1	11.7647	1.2	17.3913	0.1	1.8868	1.125	11.3924
linear	1.4	7.4468	0.3	3.5294	0.3	4.3478	0.2	3.7736	0.550	5.5696
polynomial	0.3	1.5957	0.3	3.5294	0.5	7.2464	1	18.8679	0.525	5.3165
perceptron	0	0	0	0	0	0	0	0	0	0
rational	3.5	18.6170	0.5	5.8824	0.3	4.3478	0.9	16.9811	1.300	13.1646
exponencial	0.4	2.1277	0.5	5.8824	0.8	11.5942	0.4	7.5472	0.525	5.3165
sigmoid	1	5.3191	3.6	42.3529	0.5	7.2464	0.4	7.5472	1.375	13.9241
SUMA	18.8	100	8.5	100	6.9	100	5.3	100	9.875	100

Fig. 2.1. For Motol-questionare problem, units with rational transfer function are the most efficient. In case of the Ecoli problem, there are more target classes. The **Cp** class can be separate from others with simple decision boundary (sigmoid units). Ims class has just 2 occurrences – gauss units are useful.

The Motol-questionare data has continuous target variable. It is so called regression problem. This is why the most successful (see Fig. 2.1). On the other hand, the Ecoli data have binary outputs (classification problem) and therefore other transfer functions are more appropriate. Results on the Figure 2.1 shows that units with sigmoid (logistic) transfer function are the most successful in separating members of the class **Cp** from other data vectors. For Ims class with two occurrences, the Gaussian units are most successful.

Type of the unit	Mandarin data		Iris data							
			Setosa		Versicolor		Virginica			
	Un	Uns	Un	Uns	Un	Uns	Un	Uns	Unc	Uncv
sinus 2	0	0	0.2	6.4516	0	0	0.1	1.4925	0.075	0.977199
sinus 3	0	0	0.2	6.4516	0.2	3.7736	0.1	1.4925	0.125	1.628664
gauss	0	0	0.1	3.2258	1.3	24.5283	0.9	13.4328	0.575	7.491857
sinus 1	0.3	1.9231	0.3	9.6774	0.1	1.8868	0.1	1.4925	0.200	2.605863
sinus4	1.2	7.6923	0	0	0	0	0	0	0.300	3.908795
sinus 5	0.9	5.7692	0	0	0	0	0	0	0.225	2.931596
sinus 6	1.6	10.2564	0.1	3.2258	0	0	0	0	0.425	5.537459
sinus 9	0.8	5.1282	0	0	0	0	0	0	0.200	2.605863
sinus 10	0.5	3.2051	0	0	0.1	1.8868	0	0	0.150	1.954397
sinus 11	1.1	7.0513	0.1	3.2258	0	0	0.1	1.4925	0.325	4.234528
sinus 12	0.9	5.7692	0.2	6.4516	0.5	9.4340	0.1	1.4925	0.425	5.537459
linear	0.7	4.4872	0	0	0	0	0	0	0.175	2.28013
polynomial	1.2	7.6923	0.6	19.3548	0	0	0	0	0.450	5.863192
perceptron	3.4	21.7949	0	0	1.9	35.8491	2.1	31.3433	1.850	24.10423
rational	2.8	17.9487	0	0	0.2	3.7736	0.8	11.9403	0.950	12.37785
exponencial	0.2	1.2821	0	0	0.1	1.8868	0.1	1.4925	0.100	1.302932
sigmoid	0	0	1.3	41.9355	0.9	16.9811	2.3	34.3284	1.125	14.65798
SUMA	15.6	100	3.1	100	5.3	100	6.7	100	7.675	100

Fig. 2.2. For Iris data, sigmoid and perceptron units are beneficial. Interesting is that no linear units survived.

For mandarin data set, perceptron and rational units dominated. They have complex transfer functions, meaning that resulting models will be also complex. The Iris data set is well known. Class Setosa can be easily separated from other two classes (by sigmoid units). When you look at the units surviving in models trying to separate class Vesicolor and class Virginica from other data vectors (Figure 2.2), you can say, it is not an easy task.

From several different forms of sin transfer function, the following one (sinus 12) was best survivor:

$$y_j = a_{n+1} * \sin \underbrace{\left[a_{n+2} \left(\sum_{i=1}^{n} a_i * x_{ij} + a_{n+3} \right) \right]}_{\rho_j} + a_0$$

It has superior performance mainly for the Motol questionare and the Ecoli data sets (see Figure 2.1).

2.1 Experiment with homogeneous models

Let us consider just single type of units within a model. In case we allowed just units with a linear transfer function, resulting models will be homogeneous and can be called “linear” models.

Our next experiment is aimed to answer the question if any homogeneous model can be more accurate than the heterogeneous model for given real world data set. The Figure 2.3 shows errors of several homogeneous models (*Fract*, *Gaussian*, *Linear*, etc.) and some heterogeneous models (*all*, *all-fast*, *all-simple*) on the Boston data set.

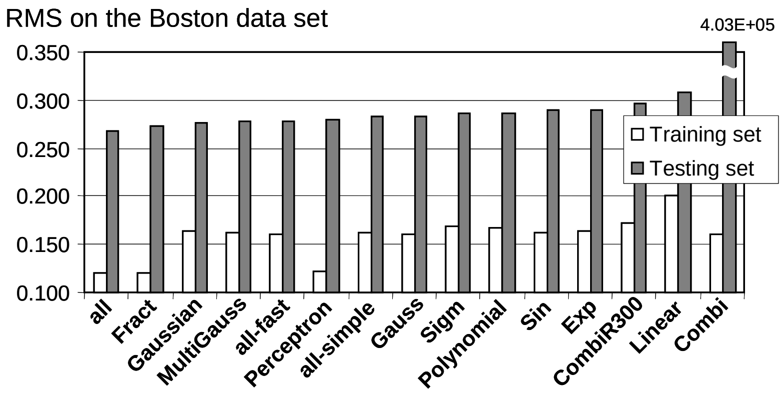


Fig. 2.3. For Boston data we generated several models where just single type of units was enabled. You can see that the most accurate model was model "*all*" where all types of units were enabled. The hybridization of the model brings new quality and increases accuracy.

It is apparent that the most successful model is the heterogeneous one (*all* units are enabled). Comparable accuracy is achieved by complex homogeneous models (*Fract, Gaussian, Perceptron*). *Linear* model is too simple and Evolving non-regularized polynomial model (*Combi*) is overfitted.

3 How is it with optimization methods?

The GAME method allows also the evolution of proper optimization method for given problem. The list of the optimization methods available in the GAME engine is given in the Table 3.1.

Tab. 3.1. List of the optimization methods available in the GAME engine

Abbrv.	Search	Optimization method
QN	Gradient	Quasi-Newton method
CG	Gradient	Conjugate Gradient method
PalDE	Genetic	Differtial Evolution ver. 1
DE	Genetic	Differtial Evolution ver. 2
SADE	Genetic	SADE genetic method
PSO	Swarm	Particle Swarm Optimization
CACO	Swarm	Cont. Ant Colony Opt.
ACO*	Swarm	Ext. Ant Colony Opt.
DACO	Swarm	Direct ACO
AACA	Swarm	Adaptive Ant Colony Opt.
API	Swarm	ACO with API heur.
HGAPSO	Hybrid	Hybrid of GA and PSO
SOS	Other	Stoch. Orthogonal Search
OS	Other	Orthogonal Search

These methods optimize coefficients of transfer functions of units in the GAME models. Again, it is not easy to decide which method will be the most efficient one for given data set. Even within single model, the complexity of the optimization problem grows with the number of layer in the model (error surface is more complicated for units deeper in the network).

We monitored which optimization methods used the successful units surviving in models of Building energy consumption (WBE) and the cold water consumption (WBCW).

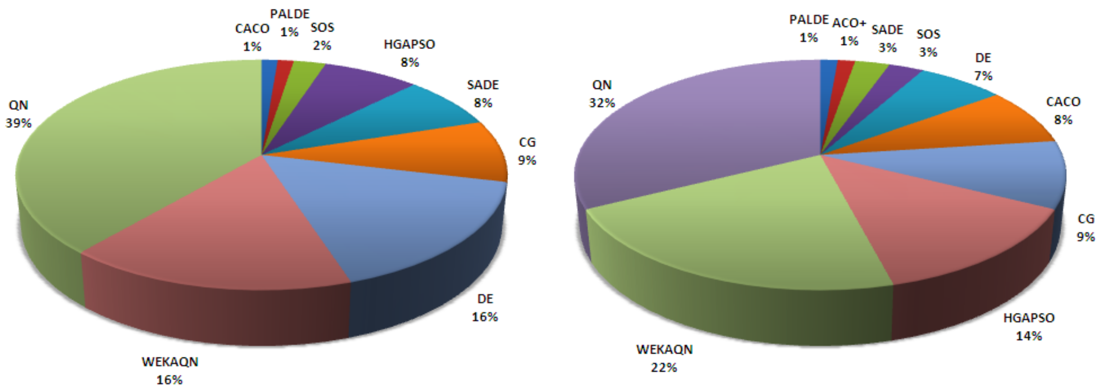


Fig. 3.2. Building WBE and WBCW – the proportional number of optimization methods surviving in final models.

When you look at the Figure 3.2, you can observe that methods of Quasi Newton type are the most efficient on this problem (more than half of successful units optimized their coefficients by Quasi Newton methods. Differential evolution (DE) is also quite successful given that it does not utilize the analytic gradient of the error [2].

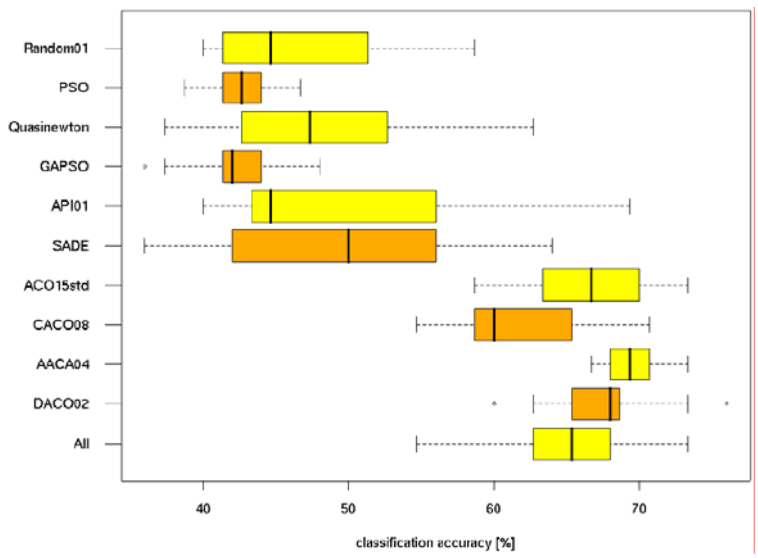


Fig. 3.3. The performance ant colony based methods (ACO15std, CACO08, AACA04, DACO02) on well known “Two intertwined spirals” problem is surprisingly much better than the performance of other methods.

The next experiment was designed to find out, if there is any superior optimization method for all data sets or if it is better to let the evolutionary algorithm selecting the most successful optimization method based on the performance of optimized units. On the Figure 3.3, the PSO means that only Particle Swarm Optimization was used to optimize units in GAME models trying to tell two intertwined spirals apart. The “All” label means that proper optimization methods were selected by the evolutionary process.

In the Figure 3.4 you can find the overall score of the optimization methods – how good models they generated, in case that only single optimization method was enabled for each model. In case of the All configuration, all methods were enabled and the evolutionary algorithm selected the best for given problem. We evaluated the performance on several problems. The Quasi Newton method performed very well on all problems. PSO algorithm showed good performance on bankruptcy data only, ACO based methods were good on spiral data set. The evolution of optimization methods generates the most accurate models, but the speed of training is significantly slower than when just Quasi Newton Method is used.

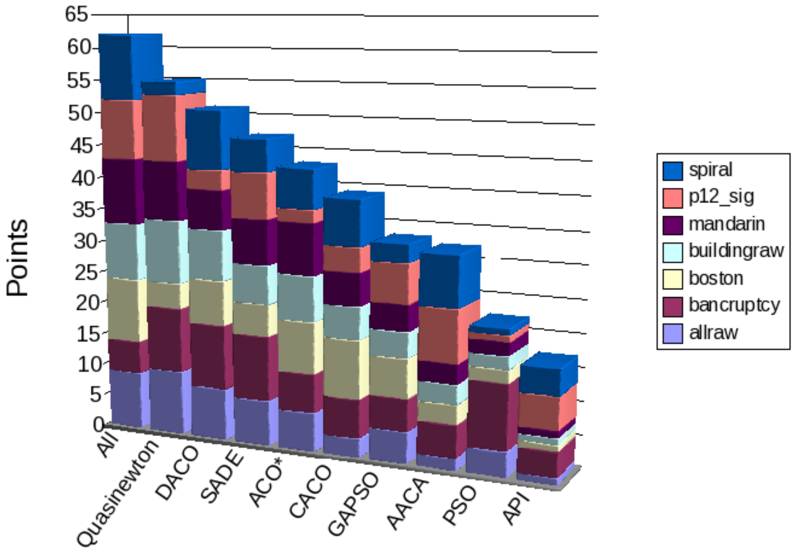


Fig. 3.4. The PSO algorithm shows good performance on bankruptcy data only, ACO based methods are good on spiral data set. The best is to enable all optimization methods and let the process of evolution to select the most appropriate for a given data set.

Also optimization methods that were selected can reveal something about problem. E.g. ACO based methods survive when the problem is very hard (spiral data). However, to be able to extract useful information about the data set from the type of surviving units and optimization methods used in the model, we need to construct an information database based on statistical results of several experiments with different real world data sets, first. Such database should contain data from Figures 2.1 and 2.2.

4 Conclusion

In this paper we showed that the evolution of both transfer functions of units and optimization methods for adjusting their coefficients is very beneficial. More accurate models are generated than in case of homogeneous models. From the type of transfer function that survived in the model, we can say much about problem modeled – lots of sigmoid transfer functions can indicate classification problem, the complexity of model is proportional to complexity of problem. Optimization methods that were selected can also tell us much about the problem.

5 Acknowledgements

Thanks to Ondrej Filipek, Ales Pilny and Oleg Kovarik for performing experiments. This research is partially supported by the grant Automated Knowledge Extraction (KJB201210701) of the Grant Agency of the Academy of Science of the Czech Republic and the research program "Transdisciplinary Research in the Area of Biomedical Engineering II" (MSM6840770012) sponsored by the Ministry of Education, Youth and Sports of the Czech Republic.

References

[1] Ivakhnenko A. G.: Polynomial theory of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-1(1):364378, 1971.

[2] P. Kordik. Fully Automated Knowledge Extraction using Group of Adaptive Models Evolution. PhD thesis, Dep. of Comp. Sci. and Computers, FEE, CTU Prague, Czech Republic, September 2006.

[3] Mahfoud, S.W. (1995),'Niching Methods for Genetic Algorithms'(95001), Technical report, Illinois Genetic Algorithms Laboratory (IlliGaL), University of Ilinios at Urbana-Champaign.