Annals of DAAAM for 2011 & Proceedings of the 22nd International DAAAM Symposium, Volume 22, No. 1, ISSN 1726-9679
ISBN 978-3-901509-83-4, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2011
Make Harmony between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals & Proceedings of DAAAM International 2011

PROBLEM-DEPENDENT, GENETICALLY EVOLVING DATA MINING SOLUTIONS

NEUKART, F[lorian]; MORARU, S[orin] - A[urel] & SZAKACS SIMON, P[eter]

Abstract: *The already introduced generic Data Mining system SHOCID (System applying High Order Computational Intelligence in Data Mining) (Neukart et al., 2011) applies Computational Intelligence (CI) paradigms for solving any numeric Data Mining problem statement. Within this paper, we introduce the evolutionary approach by which the system is able to decide on its own, which of the possible evolutionary approaches suits best for solving a presented problem statement. Moreover, the system is, by the application of genetic algorithms, able to adapt the architecture and learning method of the Data Mining solution until coming to or at least close to the optimal solution.*
Key words: *computational intelligence, data mining, genetic algorithms, metaheuristic algorithms*

1. INTRODUCTION

The application of CI-paradigms within Data Mining applications still lacks complexity, and furthermore requires the user to understand the underlying algorithms. SHOCID does not require the user to understand how the result of a mining process is achieved. Depending on the problem, the system is able to combine techniques and is, when allowed to, able to decide on its own which strategy suits best. CI substitutes intensive computation for insight into how the system works.

2. CURRENT RESEARCH

Genetic algorithms (GAs) belong to the class of metaheuristic algorithms and are used for solving optimization problems or for training Artificial Neural Networks (ANNs). GAs are adaptive, robust algorithms, and particularly useful for applications conducting search and optimization. GAs are population-based algorithms for which any communication and interaction are carried out within the population and therefore, they possess a high degree of implicit parallelism, especially useful for solving NP-hard problems. Above all, there is no unified explanation of what genetic algorithms exactly are. However, there are certain, in the field generally accepted elements, all descriptions of GAs have in common (Fulcher et al., 2011):

- A population of chromosomes encoding candidate solutions,
- a mechanism for reproduction,
- selection according to fitness, and
- genetic operators.

In ANNs making use of GAs, evolution can be introduced at various levels, starting from weight evolution, going to architecture adaption and finally leading to the evolution of the learning mechanism (Jain et al., 2008; Yao, 1999). Genetic algorithms represent possible solutions to a problem as chromosomes, and the sum of the chromosomes a population. Some chromosomes might represent fairly good solutions, some others not. If a solution is good or bad has to be determined by a so-called fitness function. The fitness function constitutes the measure of fitness of a given individual in the population, which allows to select the individuals that are the best fit (that is, having the highest fitness function), in accordance with the evolutionary principle of the survival of the strongest ones (Rutkowski, 2008). These chromosomes are granted the privilege of recombination. For ANNs, the fitness function represents the error function. Both recombination and mutation are carried out with a probability of a predefined percentage. The probability of recombination is calculated for each chromosome, and the mutation probability for each single gene. At recombination, the selected chromosomes are summarized as pairs, and for each pair of chromosomes a random number between a range is calculated.

3. ALGORITHM FOR EVOLVING DM-SOLUTIONS

The evolutionary search for the solution of a presented DM problem statement is conducted by an evolutionary adaption of each ANN's weights within an ANN committee, combined with the evolutionary adaption of each ANN's architecture, as long as no optimal solution or one close to the optimal one has been found.

3.1 Genetic determination of ANN weights

Several chromosomes, or possible solutions (ANNs) differing in their weights weights, are created and evaluated against the optimal solution during the training phase, which is conducted by the application of the Root Mean Square Error (RMSE):

$$x_{rmse} = \sqrt[2]{\frac{1}{n}\sum_{i=1}^{n}(actual_i - desired_i)^2} \quad (1)$$

3.2 Genetic determination of solution architecture

The evolutionary architecture adaption is being achieved by a constructive algorithm, meaning that the algorithm adds complexity to an ANN started with a simple structure (Frean, 1990; Mascioli et al., 1995; Omlin et al., 1993; Stepniewski et al., 1997).

3.3 Evolving committee machines

By our approach, we let SHOCID combine ANNs to committee machines. However, by applying a committee machine the detection of a correct solution is highly probable, but unfortunately the detection of the optimal one is not. Within the overall amount of training data, only a percentage of the overall data is used for training, and the remaining data sets are used for verifying the solution. Due to the verification, it can therefore be determined how accurate the created DM-model solves a presented problem. This is done by applying the RMSE as well. The system requires any presented problem to run through the following algorithm:

1. ***Start***
2. Creation of initial committee with n_c members and for each member a population of n_p ANNs with m_{min} hidden layers and n_{min} hidden neurons for each hidden layer.
3. ***Repeat***
   - ***a) If $n < n_{max}$:***
     - i. Increase n by c_n
   - ***b) If $n == n_{max}$:***
     - i. Increase m by c_m
   - ***c) Repeat***
     - i. Creation of population of x ANNs with m_{cm} hidden layers and n_{cn} hidden neurons for each hidden layer.
     - ii. Randomization of weights and threshold values of each chromosome.
     - ***iii. Repeat***
       1. Calculate the network output for the value d ∈ D
       2. Evaluate the fitness of each chromosome:
          1. Calculate the error $\delta_{d\in D}$ for each output neuron o_n
             $\delta_{d\in D=}(desired_{d\in D} - actual_{d\in D})actual_{d\in D}(1 - actual_{d\in D})$
          2. Calculate the error δ_{hid} for each hidden neuron hid
             $\delta_{hid} = actual_{hid}(1 - actual_{hid}) \sum_{d\in D} (w_{hid d\in D}\delta_{d\in D})$
       3. Selection of chromosomes to recombine
       4. ***Repeat***
          1. Crossover of chromosomes
          2. Mutation of offspring
       5. ***Until all selected chromosomes are recombined***
     - iv. ***Until criteria is reached***
   - ***d) Until number of committee members is reached***
   - e) Verification
     - i. Present eacht verification data set
     - ii. Increase committee error array for RMSE > r_{max} for each verification data set by 1
4. ***Until m_{max} is reached***
5. Comparison of stored ANN solutions by RMSE
6. Present best solution
7. ***End***

Fig. 1. Problem-dependant genetic evolution

Breakdown:

n_c: Number of committee members
n_p: Number of chromosomes per committee
m_{min} /m_{max}: Minimum/maximum number of hidden layers
n_{min} /n_{max}: Minimum/maximum number of neurons per m_n
c_n /c_m: Hidden neuron/layer counters
$d \in D$: Actual input
$\delta_{d\in D}$: Error for input $d \in D$ at output neuron o_n in the current generation
δ_{hid}: Error for hidden neuron
r_{max}: Allowed RMSE

Before applying the algorithm, SHOCID determines the number of input and output neurons according to the presented training data. n_{max} can be determined automatically, based on the number of input neurons. m_{max} will never exceed 2, as with 2 hidden layers ANNs are capable of representing an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy (Heaton Research, 2008). The algorithm shows that the system creates a number of committees n_c, each of these having n+1 neurons in the actual hidden layer, compared to the last committee. When the system has reached n_{max} for the current hidden layer m_{cur}, the second hidden layer, which simultaneously represents m_{max},is created. For m_{max}, the same constructive procedure for creating the neurons is adducted, with the exception that for calculating n_{max} the hidden layer m_{max} - 1 is drawn on. During the verification phase, an array is created for each committee, which is increased by 1 when r_{max} has been exceeded for the acutal data set. The best solution is the one with the lowest array value.

4. RESULTS

By the application of problem-dependant, genetic evolution of a DM-solution, the system is able to detect the most accurate solution without user interaction. Due to the conducted tests, the system is able to find solutions of at least the accuracy a solution of manually defined ANNs on a specific problem produces. At least, because humanly defined ANNs are usually built on the adduction of rules like the one for calculating hidden neurons only. This performs well with most of the presented problems, but also the contrary might happen. As the evolving architecture does not only rely on rules, but also verifies the performance of committess consisting of ANNs with more or less hidden neurons and layers, SHOCID also detects such solutions taks them into consideration. Current verification results on thithereto presented training data showed an accuracy between 80 and 90 %. However, the introduced algorithm is restricted to the use of classificative, predictive and regresssive Feed Forward ANNs.

5. FUTURE RESEARCH

The committee machines of SHOCID, as well as the different implementations of evolutionary approaches working together in the mining strategy processes, have been developed according to agent theory. Each of the ANNs is a single program, fulfilling a task in its own environment by the use of its own special skills. The summarization of the single agent's skills empowers them to solve complex Data Mining problems. Future research will target BDI-agents for possibly increasing the further increase of the system's performance (Neukart et al., 2011).

6. CONCLUSION

SHOCID takes up decisions for the user, which we denominate as Decision Intelligence. Recapitulatory, SHOCID is the first DM-system capable of solving most of the presented, numeric DM problems by CI-approaches, regardless to their nature. This is made possible by the system's ability to come to decisions on its own.

8. REFERENCES

Abraham, A. et al. (2008). Engineering Evolutionary Intelligent Systems Berlin Heidelberg: Springer-Verlag, p. 5

Frean M. (1990). The upstart algorithm: a method for constructing and training feed forward neural networks. Neural Comput 2:198–209

Fulcher, J. et al. (2008). Computational Intelligence – A Compendium; Springer; Berlin-Heidelberg

Heaton Research (2008). The number of Hidden Layers [2011-28-09]; URL: http://www.heatonresearch.com/node/707

Jain, L. C. et al. (2008). Computational Intelligence Paradigms Innovative Applications. Berlin Heidelberg: Springer Verlag, p. 4

Karplus W. (1998). cited in: Kaynak O, Zadeh LA, T¨urksen B, Rudas IJ (eds.) Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications; Springer-Verlag; Berlin.

Mascioli F. et al. (1995). A constructive algorithm for binary neural networks: the oil spot algorithm. IEEE Trans Neural Netw 6(3):794–797

Neukart F. et al. (2011). High Order Computational Intelligence in Data Mining A generic approach to systemic intelligent Data Mining. IEEE Press: Proceedings of Speech Technology and Human-Computer Dialogue (SpeD), 2011 6th Conference, p. 1 - 9

Omlin C. W. et al. (1993). Pruning recurrent neural networks for improved generalization performance. Techincal report No 93-6, CS Department, Rensselaer Institute, Troy, NY

Rutkowski, L. (2008). Computational Intelligence Methods and Techniques. Berlin Heidelberg: Springer-Verlag, p. 268

Stepniewski S. W. et al. (1997). Pruning back-propagation neural networks using modern stochastic optimization techniques. Neural Comput Appl 5:76–98

Yao, X. (1999). Evolving neural networks. Proceedings of the IEEE 87(9), 1423–1447