MOBILE APPLICATION AS DISTRIBUTED COMPUTING SYSTEM FOR ARTIFICIAL NEURAL NETWORKS TRAINING USED IN PERFECT INFORMATION GAMES

Delyan Keremedchiev, Maria Barova, Petar Tomov
Institute of Information and Communication Technologies - Bulgarian Academy of Sciences
"akad. Georgi Bonchev" Str., block. 2, 1113 Sofia, Bulgaria
+35929793237, d_keremedchiev@bas.bg

Abstract

Mobile devices are getting faster and faster nowadays and it is reasonable to use them for artificial neural networks training. The strongest advantage of mobile devices is their availability. Separation of a calculating task on different machines, connected by network, is known as distributed computing. There are different distributed computing platforms, like the most popular BOINC, created in Berkeley. The biggest difficulty in distributed computing platforms is heterogeneity of the computational nodes. The best way for solving computational nodes heterogeneity is touse well established technology like Android and Java by the client for computing. In this study an Android based distributed computing platform is presented. The platform is used for Artificial Neural Networks (ANN) training. Already trained ANN is used as computer opponent in a perfect information game (game in the class of combinatorial games).

Keywords: artificial neural networks, mobile devices, optimization, training

INTRODUCTION

Artificial neural networks (ANN) began their development back in 1943 with the development of Warren McCulloch and Walter Pitts [1]. In recent developments the most common model of ANN is a three-layer network with back propagation of error (BP). This type of ANN is a targeted weight graph. Each node has its own activity. The strength of the connections between nodes determines how you interact with individual nerve elements. Conventionally the network is divided into three layers. The first layer serves as the input from the external environment and is referred to as an input. The third layer provides information outside of the network to the external environment and is called output. Between input and output layer stands a hidden layer, which has an essential role in the operation of the network, but also hides uncertainties (for example the hidden layer size). In the classical three-layered model of ANN connections go only from input to output. Feedback is characteristic of recurrent ANN. The most frequent use of ANN is in tasks for classification or prediction[2][3]. The main task of the classic three-layer ANN is to correlate function between the input and the output. This process of comparison is called training. Training is a problem associated to finding such values of weights in the network that the network will perform the task for which it was designed. Once trained, ANN are extremely effective for use in practice. The learning process is slow and not very effective[4][5][6]. For the determination of optimal values of weights many learning algorithms are developed (exact or approximated). Methods are based on gradient methods, evolutionary algorithms (EA) and heuristic approaches of global optimization [7][8][9][10][11]. When EA or population algorithms are used for training the training can be done in a parallel implementation or as calculations in a distributed environment[12][13]. Big tasks have application as calculations in a distributed environment. Not all calculations can be done in parallel but those that can are preferred for calculations in a distributed environment. In the modern development of mobile devices and more reliable and efficient network connectivity, ANN training in a mobile distributed environment becomes attractive. There are different platforms in a distributed

computing environment. The most popular is BOINC and the project SETI@home[14][15]. A major drawback of the most popular platforms for calculations in a distributed environment is the challenge to work in a heterogeneous environment where hardware and operating systems on individual computers in the system varies greatly. This shortcoming is brightly illustrated in BOINC platform. A creator of a project for distributed computing is responsible to write client programs for almost any configuration (hardware-operating system) that would like to be supported. Even the presence of 32 bit, 64 bit, Windows, Linux and Mac OS X systems leads to developing at least 6 different client applications. The variety of hardware and operating systems is a major problem for scalability of a platform for distributed computing environments. At the same time, high expandability of the system can be achieved if the calculations are carried out with technology that is widespread on different hardware platforms and operating systems such as Android operating system and the programming language Java. The second major disadvantage is the need for users to have a desktop or laptop computer. This disadvantage can be overcome by using the computing power of modern mobile devices.

PROBLEM DESCRIPTION

There are different approaches for implementation of computer opponents in perfect information games. In perfect information games opponents know everything in the game (in the general case there are two opponents). They have the whole necessary information to form an optimal strategy. The most popular representative of perfect information games is chess in which both opponents see the playing board and all opponent's positions. Perfect information games are deterministic and can be completed in a finite number of moves. On completion of the full tree of states it is possible to determine optimal strategies for each player. The construction of the tree is a combinatorial problem and often leads to an impossible task due to the excessive state space. This approach makes complete depletion applicable and leads to demand for various heuristics to provide adequate strategy game without a thorough investigation of the states space. A step in this direction is the procedure of alpha-beta punching that allows not building the whole states tree. Some of the possible scenarios are to drop out at an early stage of the study. Another common approach is the use of empirical rules laid down in an expert system. The information set out in the expert system is processed by machine logical conclusion and thus determines the strategy of the game. The disadvantage of the approach with expert systems is that expert should introduce rules in the system and in fact the system does not have the capacity to accumulate knowledge alone. One way to overcome this deficiency is the use of artificial neural networks instead. The state of the playing board is submitted as an input for the ANN. The output of the ANN is the move to be played. In this study ANN is trained to play the game Complica[19]. The game consists of vertical columns in which each player places a piece of its color. The aim of the game is to form a line of four pieces with the same color (orthogonally or diagonally). The difference between Complica game and Connect Four game is that in the Complica game when a vertical column is full at the next placement of a piece it falls at the bottom of the board. This modification allows one player to beat even after the opponent's move. It is also possible two players to form a line of four simultaneously. Although the rules of the game are quite simple, creating a sufficiently skilled computer opponent is sufficiently complicated, especially when it comes to systems that need to learn and to adapt according increasing knowledge and skills of the opponents.

MODEL DESCRIPTION

This study proposes a model for ANN training based on back propagation of error algorithm. BP is in the group of the exact gradient numerical methods. ANN training is done in a mobile distributed environment. The network is represented as a Java object from the library Encog[16]. The communication with the central calculating node (the server) is carried on using the HTTP protocol. The goal of the training is improvement of the computer

opponent in a game with perfect information[18].

For the purposes of move selection, a mathematical model is built. The base of the model is ANN, trained by BP in distributed computing. Classical three-layered ANN is used. The size of the input layer is determined by the size of the playing field (in this case 35 cells) and 4 inputs are added to determine one of the four computer opponents the network presents[18]. For the output layer 5 elements are used. The game requires to choose one column of five available. The size of the hidden layer in the ANN is a matter of experimental study. The model presented in the paper uses half the size of the input layer. The activation function is a linear activation function:

(1) $u[i] = \sum (w[i][j] * x[j])$

The activation function defines the way the input signals in combination with the weighting factors influences the activity of the respective neuron. Although models with other types of activation functions are available at this stage preferences are in favor of the simplest model based on linearity.

The result of the activation function is necessarily normalized by a threshold function (a sigmoid function with values ranging from 0.0 to 1.0). Because of the different number of connections between different neurons output would perform differently if it is not normalized.

(2) $x[i] = 1 / (1 + \exp(-u[i]))$

Sigmoid function is preferred for threshold function because it is differentiable and asymptotic to infinity. It is possible to use a binary function or a linear function (to improve performance) but their properties deviate the results that can be achieved by ANN. If the network is working with values from -1.0 to +1.0 a hyperbolic tangent can be used as a threshold function.

ANN training is carried out by unsupervised learning. During the game the mobile device accumulates a list of moves that make a particular player win . Such a database is used as a source of training examples. Database (SQLite embedded database) is filled in with information for both the victories achieved by man and victories realized by computer opponents. ANN training can be organized in the form of competitions between different ANN, but since there is sufficient information from the external environment, it is not necessary.

The playing board consists of 5x7 cells. Each cell may be empty or may contain a piece with the color of one of the four players. These 5 different values are coded at the input of the 35 elements. A scaling function that scales integers from interval [0, 4] to real numbers in the range [0.0; 1.0] is used. At the input of the network four signals are further submitted. That determines one of the four players presented by the ANN. Three of these inputs are zeros (the opponents). The fourth one can be determined according to the player's number presented by the ANN.

ANN output makes assessment for prospects of each of the five possible moves. As according to the rules of the game there are no available moves the column to be played is estimated with 1.0 and the other with 0.0 in the learning mode. When the ANN is used in operational mode, the move is selected by a column with value assessment approximately 1.0.

There are many advantages of the proposed model. First, by using BP for training ANN faster convergence of the process of training is achieved. The second advantage is the possibility to carry out training of three-layer ANN with different size of the hidden layer. The third advantage is the possibility to train ANN with a growing number of training examples by accumulating examples in the local database. The fourth advantage is the ability to train various copies of the ANN and to do so in parallel. Parallel training can result in improvement of performance and better coverage of the area to search.

The proposed model has some disadvantages. Training using ANN, is a very slow and difficult process. Although very effective, after being trained, ANN requires large amounts of computing resources. Using BP significantly limits the opportunities for recurrent

topologies compared with algorithms such as differential evolution or genetic algorithms. An interesting model is one that combines BP with heuristic algorithms from the group of population algorithms for global optimization. The development of distributed computing systems is significantly more complicated than developing linear programs and even more complicated than developing parallel programs. Nevertheless a purely technological trend exists.

EXPERIMENTS AND RESULTS

The computer opponent, presented by the ANN, is opposed to three other opponents - a man, computer opponent based on random search (Random Search Optimization) and a computer opponent based on the system with five simple rules (Simple Rule Engine) [18].

For the purposes of the experiment, ANN is trained within a week. Four devices with Android operating system (two tablets and two telephones) were used. Two different people participated in 30 games on each device.

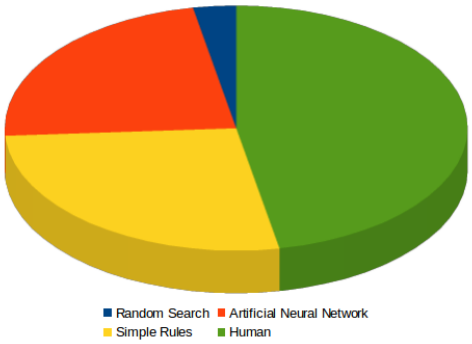

Fig. 1. *Success of ANN compared to man, random search system and simple rules. The chart reflects the achieved victories in 100 games.*

Player	Success
Random Search	3%
Artificial Neural Network	23%
Simple Rules	27%
Human	47%
Total	100%

Tab. 1. Success of ANN compared to man, random search system and simple rules. Number of wins in the 100 games.

ANN achieves better results than algorithms for random search, but weaker results than the five simple rules system based on 100 games played (Tab. 1 and Fig. 1). The results also indicate that the man was able to win almost a half of the games ANN training in a longer period of time would be an interesting task.

CONCLUSION

The realization of calculations in a distributed environment, as an Android system, leads to a very high degree of expandability in the system. Practically, distributed computing can run on any device supporting the Android operating system. Since the calculation is performed in the hardware with limited opportunities, the results cannot match the capabilities of desktop, mobile or super computers. Further experiments can be done with ANN topology as described in [17]. For distributed environment incident node participation can be used as described in [20] and [21]. For ANN training Differential Evolution can be used very successfully as described in [22].

ACKNOWLEDGMENTS

This work was supported by private funding of Velbazhd Software LLC.

REFERENCE

[1] McCulloch, Warren; Walter Pitts (1943), A Logical Calculus of Ideas Immanent in Nervous Activity, Bulletin of Mathematical Biophysics 5 (4): 115–133. doi:10.1007/BF02478259.

[2] Zissis, Dimitrios (October 2015), A cloud based architecture capable of perceiving and predicting multiple vessel behaviour, Applied Soft Computing 35.

[3] Forrest MD (April 2015), Simulation of alcohol action upon a detailed Purkinje neuron model and a simpler surrogate model that runs >400 times faster, BMC Neuroscience 16 (27). doi:10.1186/s12868-015-0162-6.

[4] Werbos, P.J. (1975), Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.

[5] Schmidhuber, Jurgen (2015), Deep learning in neural networks: An overview, Neural Networks 61: 85–117. arXiv:1404.7828. doi:10.1016/j.neunet.2014.09.003.

[6] Edwards, Chris (25 June 2015), Growing pains for deep learning", Communications of the ACM 58 (7): 14–16. doi:10.1145/2771283.

[7] M. Forouzanfar, H. R. Dajani, V. Z. Groza, M. Bolic, and S. Rajan, (July 2010), Comparison of Feed-Forward Neural Network Training Algorithms for Oscillometric Blood Pressure Estimation, 4th Int. Workshop Soft Computing Applications. Arad, Romania: IEEE.

[8] de Rigo, D., Castelletti, A., Rizzoli, A.E., Soncini-Sessa, R., Weber, E. (January 2005), A selective improvement technique for fastening Neuro-Dynamic Programming in Water Resources Network Management, In Pavel Zítek. Proceedings of the 16th IFAC World Congress – IFAC-PapersOnLine. 16th IFAC World Congress. Prague, Czech Republic: IFAC. doi:10.3182/20050703-6-CZ-1902.02172. ISBN 978-3-902661-75-3. Retrieved 30 December 2011.

[9] Ferreira, C. (2006), Designing Neural Networks Using Gene Expression Programming, In A. Abraham, B. de Baets, M. Köppen, and B. Nickolay, eds., Applied Soft Computing Technologies: The Challenge of Complexity, pages 517–536, Springer-Verlag.

[10] Da, Y., Xiurun, G. (July 2005), T. Villmann, ed. An improved PSO-based ANN with simulated annealing technique. New Aspects in Neurocomputing, 11th European Symposium on Artificial Neural Networks. Elsevier. doi:10.1016/j.neucom.2004.07.002.

[11] Wu, J., Chen, E. (May 2009). Wang, H., Shen, Y., Huang, T., Zeng, Z., ed. A Novel Nonparametric Regression Ensemble for Rainfall Forecasting Using Particle Swarm Optimization Technique Coupled with Artificial Neural Network, 6th International Symposium on Neural Networks, ISNN 2009. Springer. doi:10.1007/978-3-642-01513-7-6. ISBN 978-3-642-01215-0.

[12] Rumelhart, D.E; James McClelland (1986), Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Cambridge, MIT Press.

[13] Russell, Ingrid, Neural Networks Module, Retrieved 2012.

[14] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, SETI@home: An experiment in public-resource computing, Communications of the ACM, Nov. 2002, Vol. 45 No. 11, pp. 56-61.

[15] D. Anderson. BOINC, A System for Public-Resource Computing and Storage, In proceedings of the 5th IEEE/ACM International GRID Workshop, Pittsburgh, USA, 2004.

[16] Encog Machine Learning Framework, http://www.heatonresearch.com/encog/

[17] Zankinski I. Stoilov T. The effect of permutations of neurons in training artificial neural networks, genetic algorithms in a distributed environment, Proceedings of the XXIV International Symposium Management warm energy facilities and systems, energy management, industrial and environmental systems, ISSN 1313-2237, pp. 53-55.

[18] Balabanov, T., Complica4 - Android modification of the original board game Complica, https://raw.githubusercontent.com/VelbazhdSoftwareLLC/Complica4/master/src/eu/veldsoft/complica4/model/ia/NeuralNetworkArtificialIntelligence.java

[19] Pro Ligno Spielewerkstatt, Complica, https://boardgamegeek.com/boardgame/7476/complica

[20] Balabanov, T., Zankinski, I., Barova, M., Strategy for Individuals Distribution by Incident Nodes Participation in Star Topology of Distributed Evolutionary Algorithms, Cybernetics and Information Technologies, Vol. 16 No 1, ISSN: 1314-4081, 2016.

[21] Balabanov, T., Zankinski, I., Barova, M., Distributed Evolutionary Computing Migration Strategy by Incident Node Participation, International Conference on Large-Scale Scientific Computing, 10th International Conference, LSSC 2015, Sozopol, Bulgaria, June 8-12, ISBN 978-3-319-26519-3, pp. 203-209, 2015.

[22] Balabanov, T., Zankinski, I., Dobrinkova, N., Time Series Prediction by Artificial Neural Networks and Differential Evolution in Distributed Environment, International Conference on Large-Scale Scientific Computing, 8th International Conference, LSSC 2011, Sozopol, Bulgaria, June 6-10, ISBN 978-3-642-29842-4, pp. 198-205, 2011.