An Evaluation of Fast Multi-Layer Perceptron Training Techniques for Games

David Robertson: 1301031

School of Arts, Media and Computer Games

University of Abertay Dundee

DUNDEE, DD1 1HG, UK

Word Count: 2816

# **ABSTRACT**

Using Error Back Propagation(EBP), the training time for a Multi-layered Perceptron (MLP) Neural Network is lengthy. This has been the Achilles heel of this artificial intelligence technique when it comes to game development. Coupled with the fact that creating an MLP network is a very time consuming process, for a games company with deadlines to hit, an MLP network for their game AI does not seem very feasible. The aim of this paper is to prove that there are methods that are faster than Error Back-Propagation at training an MLP network while maintaining a good level of performance by comparison. This will be carried out by comparing the performance of different MLP training methods on their ability to control a car in a racing game. This should show that alternative MLP training methods are just as effective and faster than EBP which could push for them to be more commonly used in games.

## **KEYWORDS**

Multi-layer perceptron(MLP), error back propagation(EBP), Random-minimum bit distance gram-schmidt(RMGS), Artificial Neural Network(ANN), Resilient propagation(RPROP)

# **INTRODUCTION**

The "Threshold Logic Unit" was the first proposed artificial "neuron" and was done so in 1943 by Warren McCulloch and Walter Pitts (Stanford University 2000) in their paper "A Logical Calculus of the ideas immanent with nervous activity"(McCulloch and Pitts 1943). In which "they modelled a simple neural network using electrical circuits" to describe how the brain could work, and thus artificial neural networks were born.

Simultaneously to the development of academic artificial intelligence, work was being done on creating artificial intelligence that could play games. It would only take a few years after the inception of the idea of artificial neural networks for these game-playing computers to surface. One of the earliest examples was Dietrich Prinz's "Mate-in-two problem"(Isenberg 2016) in which the Ferranti mark 1 computer could find the best move if checkmate was only two moves away. However, it was too slow to complete a full game.

Artificial intelligence has been at the absolute core of video games since the beginning, as Alex Champandard (2004) states "since the days of Pong and Pac-man artificial Intelligence has played an undeniable role in computer games.". It has been fundamental to keeping computer games engaging and enjoyable. Because of this, game AI has continued to develop and evolve over the years, slowly introducing more academic techniques into the field and adapting them to suit what will make the game the most fun. Although this trend has been going on for the last 40 years, games have never really found the place for artificial neural networks.

There have been a number of attempts to implement Artificial Neural Networks into video games, but nonetheless they have all done just what a far less complicated game AI technique could easily take care of. There are a number of drawbacks to using a neural network for controlling the game AI; including that if offline training is used, then once the network has been trained, it has been trained for good and can no longer learn on the fly. Online learning allows this kind of on the fly learning, but the majority of learning algorithms for neural networks are unsuitable for this and must be adapted for real-time dynamic processes (Charles and McGlinchey 2004).

The key problem with implementing an artificial neural network in a game is the learning time. It takes hundreds of iterations to train the network, so if any adaptations have to be made or the training data was incorrect the entire process will have to be stopped and restarted with the updated training data. Thus there have been attempts at different methods of training a neural network in particular, the multi-layer perceptron neural network, has had many different training methods proposed to speed up learning time. Methods such as Quick Propagation and Resilient Propagation reduce some of the issues with Error Back Propagation and are "batch" methods (Champandard 2004) which inevitably speed up the process. However, they do not reduce the time significantly.

The algorithm that this project will mainly compare to error back propagation is the "Random-Minimum Bit Distance Gram-Schmidt (RMGS)" method (Verma 1997). The training time for this particular method is negligible as it trains the neural network in one iteration instead of hundreds. It is noted that this method is not as accurate as other methods. However, in a game scenario, it is not necessary and in some cases important, to make the AI not 100% accurate, otherwise the player would never be able to beat them. Since this method only takes one iteration to train the network, there is potential for multi-layer perceptrons to be able to be used and trained on the fly during a game, and if it is feasible and accurate enough, it may finally start the rise in use of this mature technique in games. This project aims to prove that feasibility.

The relevance of this project is that it aims to discover if using a faster method of training a multi-layer perceptron neural network will still be as accurate as using the standard error back propagation method. If this were to be successful then it would show that not only can MLP networks be trained more quickly, but also in particular, using the Random minimum bit distance gram Schmidt method, MLP networks have the potential to be retrained in real time rather than taking incredible amounts of time to be retrained.

## **Research Question**

Can alternative training methods for Multi-Layer Perceptron Networks be as effective as Error Back Propagation in order to promote their use in games?

## **Aim**

This project aims to evaluate the effectiveness of different multi-layer perceptron training methods by comparing the training times, accuracy and viability of each.

## **Objectives**

 Research and review different training methods including:

o Error Back Propagation

o Resilient Propagation

o Random Bit Distance Gram Schmidt(RMGS)

 Construct a racing game in which to compare each of the techniques implemented

o Build the racing game

 Create a basic multi-layer perceptron neural network

o Create implementations of each of the training methods

o Generate training data to train the networks with

o Train each of the networks

 Evaluate the training methods

o Put the different techniques through a time trial race and record their times

o Have human opponents race against the different techniques and fill out a survey comparing each technique

o Find out which of the techniques was the fastest on its own and against human opponents

o Find out which of the techniques was the most fun to play against

o Evaluate the viability of each of the techniques taking into consideration the time taken to implement and train the network and the results of the tests

# **LITERATURE REVIEW**

It has been proven that multi-layer perceptron neural networks can control a car in a racing game, for example Colin McRae Dirt 2, however training multi-layer perceptrons takes a lot of time and thus they are rarely used in games. This project aims to achieve proof that alternative training methods for multi-layer perceptron networks can not only reduce the training time, but be as effective as the most commonly used "Error Back Propagation".

Racing games are especially good for testing multi-layer perceptron networks as there are many potential inputs to think about for driving a racing car around a track. For this kind of neural network, as the number of inputs increases, the harder the network has to work. This will test the Random Minimum Bit Distance Gram-Schmidt (RMGS) and resilient training methods thoroughly. These training methods will be discussed in the following sections.

## **Spiking vs Multi-Layer Perceptron Neural Networks**

In the paper "Spiking Neural Network vs multi-layer perceptron: who is the winner in the racing car computer game" the authors Urszulla Markowska-Kaczmar and Mateusz Koldowski created a top down racing game in which the competing networks would race against one another to aim for the best time around the track. Both of the networks are trained via genetic algorithms and given points based on their performance for further evolutions.

Spiking Neural Networks are the third generation of neural network (Markowska-Kaczmar and Koldowski 2015). They are modeled to most accurately represent a brain. Each neuron in a Spiking Neural network is given an activation voltage and the current voltage that the neuron has will be calculated by the input of the network. Once the activation voltage has been reached, a pulse will fire and the neuron will revert to the resting voltage. This means that the input of the network will not affect the size and shape of the "pulse" from the neuron; instead, it will determine when it fires. They are regarded as a "computationally powerful and biologically more plausible model of distributed computation"(Yee and Teo 2011).

This project is similar to this particular paper as it is comparing the performance of two types of neural networks in a racing game; the difference is that the comparison in this project is between two training methods for multi-layer perceptron networks instead. The paper states that the car uses ray casted sensors to follow a racing line around the track. In this project the car will also use sensors to find its position, however it will be searching for waypoints that will be placed on the track. A similar evaluation technique will also be used in this project; as the time it takes for the car to get around the track is the best measure of the performance of the training methods in a racing game scenario.

## **Colin McRae Dirt 2**

Colin McRae Dirt 2 is one of the only games that has been documented to use an Artificial Neural Network for controlling the AI in game. In an interview with the website "AI-Junkie" the programmer responsible for the neural network, Jeff Hannan, evaluated his work and how the process worked. The Neural network that he implemented was not extremely complex stating that "The outputs from the neural net were simply on/off flags for pressing buttons on the controller"(Buckland 2004), and that the neural net was only programmed to follow the racing line as "The AI in these other situations was simply rule based when it came to crashes and overtaking. The training method that was implemented in this neural network was resilient back propagation (RPROP).

This project could be compared to Hannan’s work and is somewhat a spiritual continuation on the AI that was implemented. For example, the output of the neural network in this project will be flags for the buttons on the "controller" and the neural network will be a multi-layer perceptron network. However, this project also has some differences including; using ray casts and waypoints around the track to steer correctly instead of following a racing line, and although resilient propagation will be implemented, the RMGS training method (which will be discussed in the next section) will be the focus of this project.

## **Training Methods**

### **Error Back Propagation**

The error back-propagation method is the most common training method for multi-layer perceptron neural networks. It works by firstly setting all of the neurons to a random weight and sending the training data through the network. Then it calculates the error of the network's output compared to the desired training data output. Following this, the error is then used to adjust the weights of the output layer's neurons; this sequence is then repeated for the hidden layers working backwards from the output layer until the entire network has been corrected. This entire process is then repeated multiple times until the calculated error reaches a minimum. (Bourg and Seemann 2004)

### **Resilient Propagation**

Resilient Propagation works somewhat similarly to Error Back Propagation in the sense that all the weights are updated depending on a calculated error. However, Resilient propagation does not update the weights until all of the training data has been seen. Since it is a batch algorithm a gradient must be calculated for each neuron, this is done by passing all of the training data through the network and calculating a gradient for the error on the neuron. Once this has been completed, the weights are then adjusted accordingly in relation to the gradient of error that has been calculated. (Champandard 2004)

### **The Random-Minimum Bit Distance Gram-Schmidt Method**

Many methods to speed up the training time of multi-layer perceptron neural networks have been proposed over the years. The training method that is going to be the main comparison to error back propagation is the random minimum-bit distance gram-schmidt method. The interesting thing about this method is that it uses different techniques to train different layers of the network. However, it only needs one iteration to be trained. Hypothesized by Brijesh Verma (1997) the method makes use of supervised and unsupervised learning for training the output layer and the hidden layers respectively. As stated by Verma(1997) "The proposed solutions are much faster and without local minima because they use direct solution methods". This makes the implementation of the method far more complicated but, once completed, the training time is negligible compared to error back propagation.

This project will implement each of these training methods to control different AI vehicles.

# **METHODOLOGY**

To complete the first object of this project, research will need to be carried out on the three MLP training methods discussed above, beginning with the most common Error Back Propagation followed by Resilient Propagation and finally, the least common, the RMGS method.

Research on how to recreate the RMGS training method will be the most focused part of the research as it has a lot of complexity, beginning with how to use the minimum bit distance equation detailed in the paper by Verma and using the modified Gram-Schmidt equation for solving the output of the network.

Once the research objective has been completed, the implementation objective will need to be undertaken. Beginning with building the game, it will be a 2D top down racing game, built in C++ using the Games Education Framework by Grant Clarke, and using Box2D as the physics library. Since it is a 2D game, there is no need for an overly complicated physics package. Next the basic multi-layer perceptron neural network will be created and the different training methods will follow shortly afterwards. Starting with the most common EBP training method, followed by RPROP and finally the RMGS training method. Once these have been created, the training data will be generated and used to train each of the methods to control the racing game.

The inputs of the network are still to be fully decided, however, taking inspiration from Hannan’s AI for Colin McRae Dirt 2, the outputs of the network will be simple on/off flags for the buttons to control the car. This will put the AIs on a level playing field as any player who may wish to compete with them

When all of the training methods have been implemented and fully trained, the application will have the ability to show that each of the training methods can race a car around a track with high accuracy and speed.

Finally, the application will need to be evaluated. This will be done by firstly comparing the training times of each training method. Followed by recording and comparing the individual lap times completed by the methods. Finally, a human player will race against all of the methods and fill out a questionnaire detailing which was the most realistic and which was the most fun to race against.

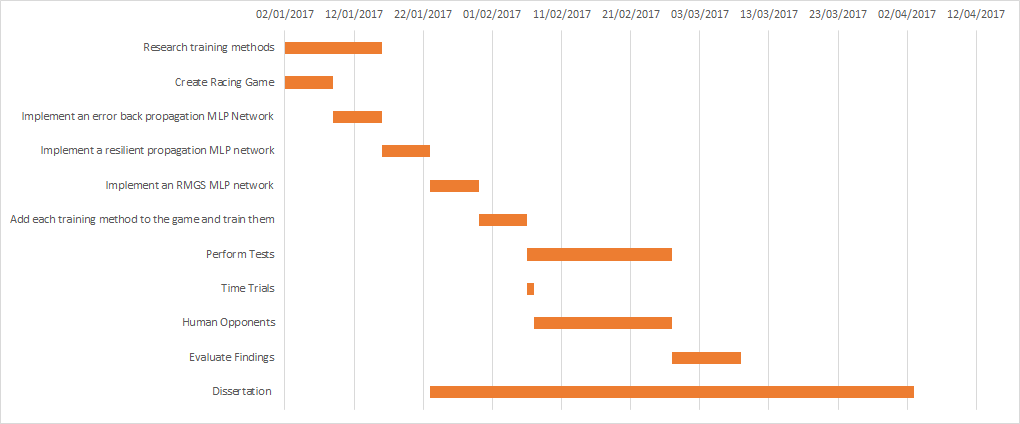
Once all of the evaluation data has been collected the training methods will be compared through all the methods to see if the alternatives to error back propagation can compete with its accuracy. This will give a sound understanding on whether or not alternative multi-layer perceptron training methods would open a door to such AI techniques being used in games in the future.

## **Possible risks**

Since the RMGS training method is very scarcely documented, it will be very challenging to implement, therefore the following fallback options are in place if it is deemed that completing an implementation of the RMGS method is unachievable.

Firstly, the RMGS method would be replaced by a different training method such as quick propagation, which is a batch training technique similar to resilient propagation or one of the other techniques proposed by Verma; error back propagation using direct solutions (EBUDS) or delta rule-symmetric gaussian elimination (DRSGE).

## **Project Schedule**



# **SUMMARY**

In conclusion, this project aims to show that alternative MLP training methods are faster than and just as accurate as error back-propagation. This will be done by building a top down racing game to compare the effectiveness of error back-propagation, resilient propagation and the RMGS training method. The comparison will be made from the time taken for each AI to drive around the track and the results of the human participant questionnaire, to see not only which of the methods is better at getting around the track, but also to see which is more fun to play against. Ideally, this project will prove that the RMGS training method is as accurate as error back-propagation when it comes to dealing with a complex problem such as a racing game. If this is proven true, then there is potential for the rise in use of multi-layer perceptron neural networks in games.

# **REFERENCES**

Bourg, D. and Seemann, G. 2004. AI for Game Developers. Sebastopol, United States of America: O’Reilly Media, Inc, pp.285-289.

Buckland, M. 2004. Interview with Jeff Hannan, Avaliable from : <http://www.ai-junkie.com/misc/hannan/hannan.html> [Accessed 23 September 2016]

Champandard, A. 2004. AI Game Development. Indianapolis, United States of America: New Riders Publishing, pp.3, 240-251.

Charles, DK and McGlinchey, S. 2004. The Past Present and Future of Artificial Neural Networks in Digital Games. *Proceedings of the 5th international conference on computer games: artificial intelligence, design and education.* pp. 163-169

Isenberg, G. 2016. *Mate-in-two.* Chess Programming Wiki Available from: <https://chessprogramming.wikispaces.com/Mate-in-two> [Accessed 22 October 2016]

Markowska-Kaczmar, U. & Koldowski, M. 2015. Spiking neural network vs multilayer perceptron: who is the winner in the racing car computer game*. Soft Computing*. 19(12), pp.3465-3478.

McCulloch, W. S. & Pitts, W. 1943. A Logical Calculus of the Ideas Immanent in Nervous Activity*. Bulletin of Mathematical Biophysics*. 5(1), pp 115-133.

Stanford University. 2000. *History: The 1940's to the 1970's* Available from : <https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html> [Accessed 10 October 2016]

Verma, B. 1997. Fast Training of Multilayer Perceptrons, IEEE Transactions on Neural Networks, 8(6), pp. 1314-1320.

Yee, E. and Teo, J. 2011. Evolutionary Spiking Neural Networks as Racing Car Controllers. *11th International Conference on Hybrid Intelligent Systems (HIS),* pp. 411-416