Ph.D. Research Proposal

Computer Science and Information Engineering

<Deep Neural Game Theory>
<Game Theory Neural Network>

<Brown Wang>
<Brownwang0426@gmail.com>

<28/3/2018>

Department of Computer Science and Information Engineering

all come with two

Abstract

Can a well-trained neural network for algebra make deduction that A = 2 when given A + B = C, B = 8, and C = 10? Yes, it can.

Can a neural network be bestowed self-awareness and strife for its own benefit? Yes, it can.

This paper exploits the notion of Back Propagation in an untraditional way and forces the neural network to make the best guess for its best strategy. The most intrinsic is that this paper rather uses Back Propagation to find the best Input Neurons without changing the well-trained synapses. That is the Back Propagation toward Input Neurons, not toward already-well-trained synapses.

Can a neural network think in the way how human thinks when trying to manipulate his opponents? Yes, it can.

This paper further exploits the notion of Back Deduction in Game Theory and force the Recurrent Neural Network to guess for its best strategy in the same way human does in Sequential Game in Game Theory.

This technique has much potential future application such as reinforcing video game AI or helping human enacting policy. And it is the goal of this paper to try to explore its usage.

Keywords

Recurrent Neural Network, Back Propagation toward Input Neurons, Game Theory, Sequential Game, Back Deduction.

Table of Contents

Chapter 1
Introduction

How can a neural network make deduction for the easiest algebra, that is, how can a well-trained neural network for algebra make deduction that A = 2 when given A + B = C, B = 8, and C = 10?

This paper exploits the notion of <u>Back Propagation</u> in an untraditional way and forces the neural network to find the best Input Neurons <u>without changing the well-trained synapses</u>. That is the <u>Back Propagation toward Input Neurons</u>, not toward synapses.

In that sense, the machine can make deduction for its unknow Input Neurons. That is the machine can make deduction rather than just induction which we did in traditional Deep Feed Forward Neural Network (DFNN).

If the Input Neurons are the action or movement that a neural network should take in order to maximize its own profit (to make the Output Neurons [1, 0] where [1, 0] stands for the best profit for this present Input Neurons), we can say that this machine gains self-awareness for its own profit.

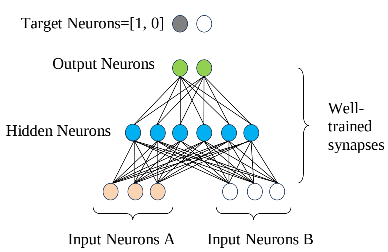
In Game Theory, in Sequential Game, given all player's possible strategies and consequential profits for all players, one player can find the best strategy to maximize its profit. The combination of these final strategies of all players is the famous Nash Equilibrium.

If we view the strategies of two players as two different sets of input neurons in one identical neural network. When the neural network is trained well (either by telling the machine all player's possible strategies and consequential profits for all players manually, or by letting the machine to learn by trial-and-error), we can duplicate the notion of A(Input Neurons) + B(Input Neurons) = C(Target Neurons) as stated above, except that this time the Input Neurons A was forced to output (1, 0) and the Input Neurons B was forced to output (0, 1). The exact way of exploiting the well-trained machine is that we let player A to move first (by changing Input Neurons A and forcing the Output Neurons to approximate Target Neurons (1, 0) through back propagation and gradient descent), and then we let player B to move afterward (by changing Input Neurons B and forcing the Output Neurons to approximate Target Neurons (0, 1) through back propagation and gradient descent). At least, we let machine repeat the movement above.

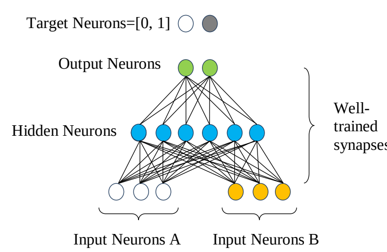
Miraculously, the final Input Neurons A and Input Neurons B will match Nash Equilibrium automatically. The whole process is illustrated as below:

		Player B strategy		
		[1, 0, 0]	[0, 1, 0]	[0, 0, 1]
Player A strategy	[1, 0, 0]	[0.3, 0.4]	[0.1, 0.3]	[0.6, 0.2]
	[0, 1, 0]	[0.2, 0.1]	[0.9, 0.4]	[0.3, 0.4]
	[0, 0, 1]	[0.2, 0.2]	[0.2, 0.3]	[0.4, 0.2]

【Figure 1】



【Figure 2】



【Figure 3】

First, we train the DFNN according to 【Figure 1】, Input Neurons A represents Player A strategy and Input Neurons B represents Player B strategy. The corresponding payoffs of each situation is the Target Neurons in this training process.

Second, set Input Neurons A and Input Neurons B to 0s. Then, we use Back Propagation and gradient descent to Back Propagate Input Neurons A only and hold Input Neurons B constant. The Target Neurons this time is set to [1, 0] as shown in 【Figure 2】.

Third, perverse the obtained Input Neurons A and B obtained in the previous step. Then, we use Back Propagation and gradient descent to Back Propagate Input Neurons B this time and hold Input Neurons A constant. The Target Neurons this time is set to [0, 1] as shown in 【Figure 3】.

At last, we iterate the process in 【Figure 2】 and 【Figure 3】 for a fair big epochs.

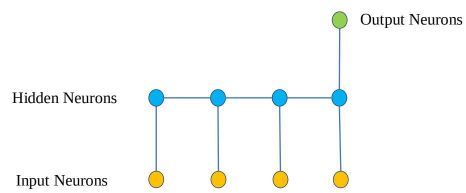
The final outcome of Input Neurons A and Input Neurons B are [1, 0, 0] and [1, 0, 0] (when argmaxed), which is exactly the Nash Equilibrium underlined in 【Figure 1】.

It is a miracle.

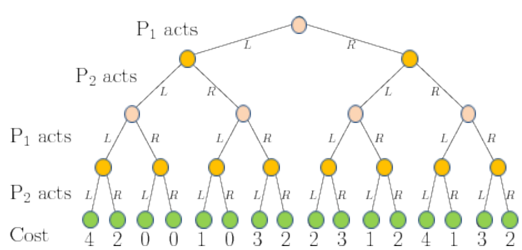
In fact, similar ideas of this kind of iterative training can be found in Generative Adversarial Neural Network (GAN), except that GAN emphasizes on training two different sets of synapses to meet different goals (to cheat discriminator or to discriminate cheater, vice versa) while Deep Neural Game Theory (DNGT) takes an inversed step and emphasizes on training two different sets of Input Neurons in one identical neural network to meet different goals while leaving already-well-trained synapses untouched.

Also, in Game Theory, there is Sequential Game. In this kind of game, each player tries to outsmart other players by a technique call “Back Deduction” which states that each player in an upper layer of a game tree will try to manipulate the players in the lower layer of the game tree by imitating the rationale of the lower players and try to choose the best strategy in the upper layer.

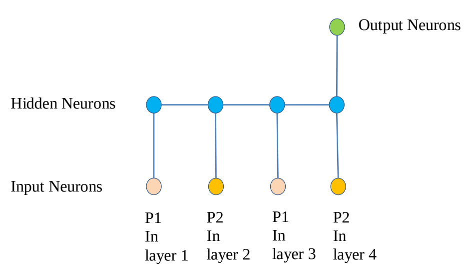
In Artificial Intelligence, aside from traditional Deep Feedforward Neural Network (DFNN), there is also Recurrent Neural Network (RNN, including LSTM, Neural Turing Machine, etc.). It has its famous simplified long body as below:



In the same sense as we elaborated in the combination of Simultaneous Game and DFNN which we let player A to move first (by changing Input Neurons A and forcing the Output Neurons to approximate Target Neurons (1, 0) through back propagation and gradient descent), and then we let player B to move afterward (by changing Input Neurons B and forcing the Output Neurons to approximate Target Neurons (0, 1) through back propagation and gradient descent). At least, we let machine repeat the movement above and we can find Nash Equilibrium automatically.
Conversely, in the combination of Sequential Game and RNN, we also view each set of Input Neurons of RNN as the strategy of each player in each layer in the game tree. Take a look at the simplified diagram illustrated below:



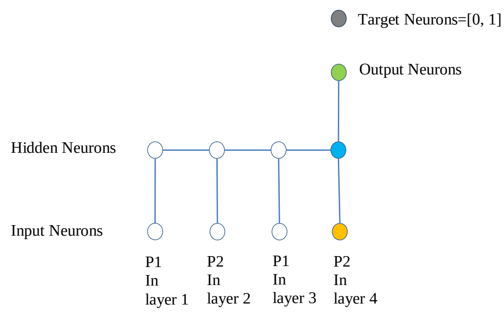
【Figure 4】



【Figure 5】

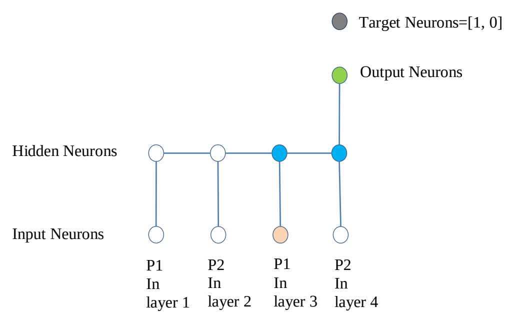
First, we train the RNN in 【Figure 5】 according to 【Figure 4】. Input Neurons P1 in layer 1 represents the strategy that player 1 adopts in layer 1 in the game tree as shown in 【Figure 4】 and Input Neurons P2 in layer 2 represents the strategy that player 2 adopts in layer 2 in the game tree as shown in 【Figure 4】. So on and so forth. The corresponding payoff of each situation is the Target Neurons in this training process.

Second, we set all sets of Input Neurons randomly. Then, by Back Propagation, we let the player P2 in the fourth layer to move first (let the Output Neurons of Input Neurons P2 in layer 4 to approximate Target Neurons (0, 1) through back propagation and gradient descent while holding P1 in layer 1, P2 in layer 2, and P1 in layer 3 constant) as illustrated as below:



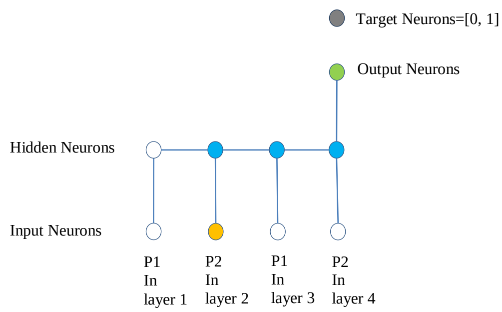
【Figure 6】

Third, we preserve the strategy (input neurons) of P2 in the fourth layer. And then we let the player P1 in the third layer move consecutively (let the Output Neurons of Input Neurons P1 in layer 3 to approximate Target Neurons (1, 0) through back propagation and gradient descent while holding P1 in layer 1, P2 in layer 2, and P2 in layer 4 constant) as illustrated as below:



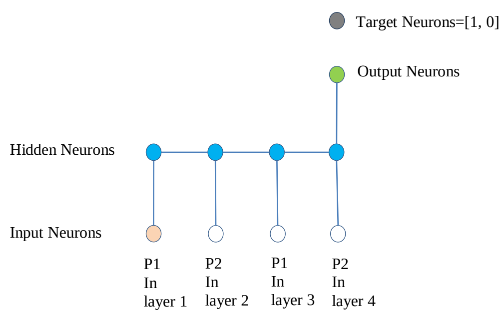
【Figure 7】

Fourth, we preserve the strategies (input neurons) of P2 in the fourth layer and P1 in the third layer. And then we let the player P2 in the second layer move consecutively (let the Output Neurons of Input Neurons P2 in layer 2 to approximate Target Neurons (0, 1) through back propagation and gradient descent while holding P1 in layer 1, P1 in layer 3, and P2 in layer 4 constant) as illustrated as below:



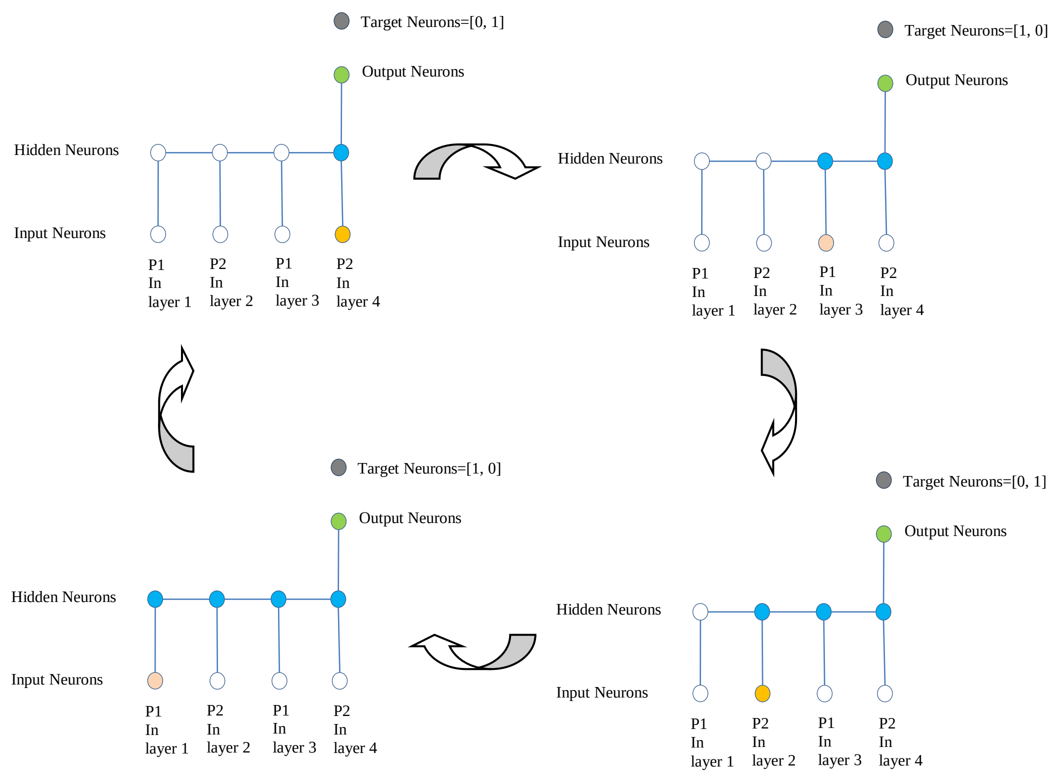
【Figure 8】

Fifth, we preserve all the strategies seen so far, and then we let the player P1 in the first layer move consecutively (let the Output Neurons of Input Neurons P1 in layer 1 to approximate Target Neurons (1, 0) through back propagation and gradient descent while holding P2 in layer 2, P1 in layer 3, and P2 in layer 4 constant) as illustrated as below:



【Figure 9】

At last, we iterate the process in 【Figure 6】 to 【Figure 9】 for a fair big epochs. As shown in 【Figure 10】:



【Figure 10】

In this iteration, we implicitly exploit the notion of Back Deduction in Game Theory.

In the end, the input neurons of P1 in layer 1 is the best strategy for player 1.

It is a miracle.

Chapter 2
Motivation

The motivation of this work is two-folded, one is to give the machine the ability to make deduction. The other is to combine Neural Network and Game Theory.

In order to overcome present technical barrier, this paper adopts an untraditional way to exploit Back Propagation. That is the <u>Back Propagation toward Input Neurons</u>, not toward synapses.

Upon this discovery, this paper takes a further step to merge Game Theory and Neural Network.

Chapter 3
State of the Art

For example, as tiny as a mosquito, whenever faced with a waving hand, it can make deduction about what movement best fits its interest – that is just flying away less that it gets smashed by human hands.

A traditional method is that we manually set the input neurons as “a waving hand” and the target neurons as “flying away”. In this notion, we can expect that this mechanic mosquito can fly away whenever it sees a waving hand.

However, in this state-of-the-art process, we implicitly install a human-bestowed knowledge that is “flying away saves your life”. The mechanic mosquito never needs to learn about whether flying way saves its life or not. When next time this mosquito meets a mosquito beat, it won’t fly away, and it dies. Even though this mosquito dies with this information stored in its soul (or in the google cloud), it still does not know how to use it. It dies next time as well. Human engineer must further manually set the input neurons as “a mosquito beat” and the target neurons as “flying away” this time and train the machine again.

If this mechanic mosquito were to human and human engineer were to God, God will be tired to death.

This paper saves this tiresome job by bestowing machines the ability to make deduction. All human engineer need to do is simply let the machine store every strategy and its corresponding outcomes. The machine will take the job itself afterwards

Upon this technique, this paper further combine Game Theory and Neural Network through the iterative training shown in 【Figure 1】to 【Figure 9】.

In fact, similar ideas of this kind of iterative training can be found in Generative Adversarial Neural Network (GAN).

However, GAN emphasizes on training two different sets of synapses to meet different goals (to cheat discriminator or to discriminate cheater, vice versa) while Deep Neural Game Theory (DNGT) takes an inversed step and emphasizes on training two different sets of Input Neurons in one identical neural network while leaving already-well-trained synapses untouched.

Chapter 3
Research Objectives and Approach

The paper aims to test the limit of this technique. This technique can be applied to video games or online games (including real-time strategy games, action games, board games, etc.) to reinforce the intelligence of the monsters or AI opponents in order to conquer human players.

The second usage of this technique is to make prediction about numbers concerning human tactical thinking in a time sequence or series such as stock market etc..

The third and the most influential usage of this technique to bestow the machine with self-awareness and assume the role as human-policy consultancy to assist human in policy making such as traffic control, criminal deterrence, etc..

However, the future obstacles of this technique are manyfold.

First, concerning the technical obstacle, it isn't in every situation that we know the payoffs of our opponents, so in practice, we might not know the payoffs of our opponents regarding their strategies which makes prediction according to game theory impractical. However, this obstacle can be easily overcome by simply observing the strategies adopted by our opponents and the likelihood of it. By observation, the more likely that a strategy is adopted, the more likely that the player gives it higher credit.

Second, concerning the philosophical obstacle, if this policy machine ever goes wrong, biased or defected and human trust the machine whole-heartedly nonetheless, this machine might force the society of human to fulfill its false-prophecy and makes it look like it never goes wrong. This is the famous "self-fulfilling prophecy problem".

However, the second problems is far beyond the reach of this paper. And it is expected to be amended by human legislators.

Chapter 4
Current Work and Preliminary Results

The current work of this paper is to apply this technique to Tic-Ta-Toe game. Actual source code can be seen in:
https://github.com/Brownwang0426/Deep_Neural_Game_Theory

The result shows that the machine can perfectly make the best move in response to every circumstance. In short, it thinks like human.

Chapter 5
Work Plan and Implications

In the future, this technique can be applied to video games or online games (including real-time strategy games, action games, board games, etc.) to reinforce the intelligence of the monsters or AI opponents in order to conquer human players.

The second usage of this technique is to make prediction about numbers concerning human tactical thinking in a time sequence or series such as stock market etc..

The third and the most influential usage of this technique to bestow the machine with self-awareness and assume the role as human-policy consultancy to assist human in policy making such as traffic control, criminal deterrence, etc..

Chapter 6
Conclusions

What is interesting enough is that there is Back Deduction in Game Theory while there is Back Propagation in Artificial Intelligence. There is Simultaneous Game in Game Theory while there is Deep Feedforward Neural Network in Artificial Intelligence. there is Sequential Game in Game Theory while there is Recurrent Neural Network in Artificial Intelligence.
We can skillfully merge these two giants into one leviathan, exerting the most devastating power.

References