Reversed Neural Network (Abstract)

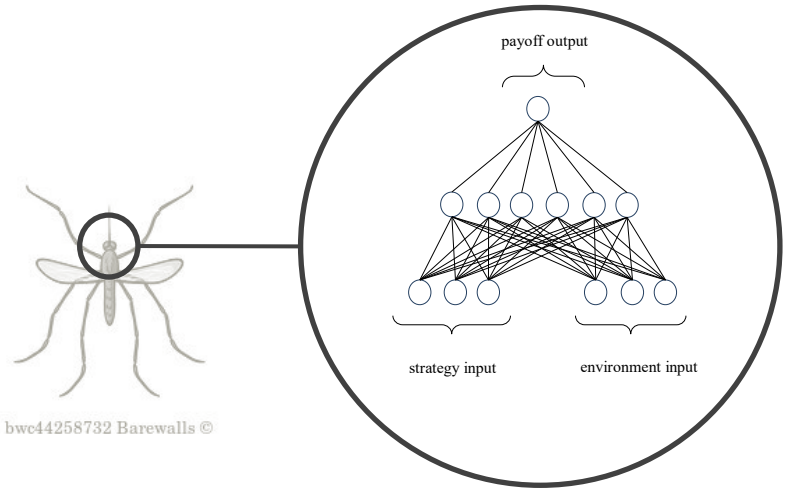
Automatically Finding the Nash Equilibrium

Brown Wang, National Taiwan University, Doctoral Student, Brownwang0426@gmail.com,

D07944007@ntu.edu.tw

ABSTRACT

Contrary to most reinforcement learning research, which emphasizes on training a deep neural network to have its output layer to approximate a certain strategy, this paper proposes a revolutionary and a reversed method of reinforcement learning. We call this "Reversed Neural Network". In short, after we train a deep neural network according to a strategy-and-environment-to-payoff table well enough, then we use back-propagation algorithm and propagates the error between the actual output and the desired output back to the "input layer" of a deep neural network gradually to perform a task similar to "human deduction". And we view the final "input layer" as the fittest strategy for a neural network in a natural environment. To grab this ideology, we will take a mosquito for example.



Imagine a brain of a mosquito only consists of a deep neural network with only one input layer, one hidden layer, and one output layer. The part of the neurons of the input layer records information concerning the strategies that the mosquito has adopted in response to certain environment information, which is also recorded by the part of the neurons of the input layer as well. Meanwhile the output layer records the payoff to the mosquito when it adopts a certain strategy under a certain environment.

The brain of the mosquito is trained to recognize the consequence (payoff) of every combination of a certain strategy under a certain environment. For example, the consequence of seeing human's hand and not flying away is recorded. Of course the consequence is death, in which case the payoff to the mosquito is [0]. The software (or the soul, arguably) of the mosquito records the consequence (payoff) of every combination and trains the brain (the neural network) of the mosquito according to the table as below:

	strategy	environment		payoff	
fly away	[1, 0, 0]	[1, 0, 0]	human hand exists	[1]	alive
stay	[0, 0, 1]	[1, 0, 0]	human hand exists	[0]	death
fly away	[1, 0, 0]	[0, 0, 1]	human hand not exist	[1]	alive
stay	[0, 0, 1]	[0, 0, 1]	human hand not exist	[1]	alive

When the brain (the neural network) of the mosquito is trained well enough according to the table above, upon seeing a human's hand waving through, the brain (the neural network) of the mosquito must figure out what strategy it shall adopt in order to stay alive. In another word, **how could the neural network figure out a strategy that, when put into the neural network, can generate the payoff [1]?**

The answer is simple – Just put an initially-randomized strategy into the input layer of the well-trained neural network and let the neurons that represent the strategy to adjust itself gradually to make the neural network to generate payoff [1] by propagating the error between the actual output and the desired output [1] back to the strategy neurons itself (only) every time when the generated actual output (of the adjusted strategy neurons) does not adhere to [1].

If we take the mosquito and the hand of human as the two players A and B in a Simultaneous Discrete Game. After the neural network being trained according to the strategy-payoff table for player A and B, we can tune the neurons which represent the strategy of player A in the input layer to generate the desired output for player A, and vice versa.

After we interchangeably and iteratively tune the two sets of neurons which represent the strategy of player A and B, the two sets of neurons will converge to the Nash Equilibrium!!!

Source Code can be seen at: https://github.com/Brownwang0426/Deep_Neural_Game_Theory

REFERENCES

[1] Avinash K. Dixit, Susan Skeath: Games of Strategy, Second Edition, 2004.

[2] Robert Gibbons: A Primer in Game Theory, 1992.

[3] Colin Adams, Robert Franzosa: Introduction to Topology_ Pure and Applied. 2007.

[4] Anurag Agrawal and Deepak Jaiswal, "When Machine Learning Meets AI and Game Theory", cs229.stanford.edu, 2012.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller, "Playing Atari with Deep Reinforcement Learning", cs.toronto.edu.