|  |  |
| --- | --- |
| **SOURCE CODE**  **EXPLANATION** | Representative source code: Deep Neural Game Theory  Under traditional deep learning, we exert the notion of Back Propagation to adjust the synapses of neural network to make the output neurons to match its expected target neurons.  This is at least what Jeoffrey Hinton originally intended around 1980s when he first invented the notion of back propagation and deep learning.  In this notion, we let the machine to infer the target neurons by already-known input neurons.  However, when some information is lost in the input neurons, how can the machine make deduction about the supposed but lost input neurons?  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, did you notice that, in this process, we implicitly install a human-bestowed knowledge that is “flying away save 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 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 state the input neurons as “a mosquito beat” and the target neurons as “flying away” this time.  If this mechanic mosquito were to human and human engineer were to God, God will be tired to death.  In this case, can we really proudly say that we invented a machine that can make deduction and infer the best strategy for itself? Probably not.  To solve this problem, this paper further exploits and explore the notion of Back Propagation to bestow the machine with the ability to make deduction. For a well-trained neural network, this paper purposely vacuumizes some of its input neurons. Through the notion of Back Propagation, we force the neural network to make deduction about the best input neurons (which was vacuumized) to fit its expected pre-set target neurons. However, the well-trained synapse is intact and left unchanged in this process.  For example, we can force the machine to learn some basic algebra such as 1+1=2, 1+2=3, A(Input Neurons)+B(Input Neurons)=C(Target Neurons). When the machine is trained well, we tell that B=2 and C=10 and force the machine to make deduction that A=8 through back propagation.  Maybe it does not seem much. But if the input neurons are  “flying away”(Input Neurons) + “a waving hand”(Input Neurons) = “survive”(Target Neurons)  “stay”(Input Neurons) + “a waving hand”(Input Neurons) = “death”(Target Neurons)  We can force the machine to make inference or deduction about what strategy best fits its interest (to survive) when faced with a waving hand.  In a sense, the purpose of the Hinton machine was to force the output neurons to approximate the target neurons by adjusting the synapse. However, this machine takes a reversed direction – that is this machine forces the output neurons to approximate the target neurons by adjusting part of the input neurons while left the whole synapse unchanged and intact. In this process, the machine gains “self-awareness” to maximize its profit in different circumstances either (1) through trial-and-error learning or (2) through pre-set information by human engineer.  But what does it have anything to do with game theory?  Under game theory, every player adjust their own strategy on the basis of the action of other players to maximize its profit. It is upon the same rationale that was described in the mosquito case. Since the machine can imitate the thinking of a mosquito to maximize its profit, why not let it imitate the thinkings of two players?  For example, in the most traditional Simultaneous Game in game theory as illustrated as below:  “simultaneous game”的图片搜索结果Figure 1  If we view the strategies of the two players (column and row) as two different input neurons and the payoffs in Figure 1 as target neurons. When the machine is trained well (either by telling the machine the strategies along with the consequences 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 designed to output (1, 0) and the Input Neurons B was designed to output (0, 1). The exact way of exploiting the well-trained machine is that we let player column to move first (let the Output Neurons of Input Neurons column to approximate Target Neurons (1, 0) through back propagation and gradient descent), and then we let player row to move afterward (let the Output Neurons of Input Neurons row to approximate Target Neurons (0, 1) through back propagation and gradient descent). At least, we let machine repeat the movement above.  In the end, this machine can automatically find Nash Equilibrium in this simultaneous game—that is (T, L) and (M, C).  Wanna see something more interesting? Sure.  The technique as illustrated above is just one combination of Deep Feedforward Neural Network in Artificial Intelligence and Simultaneous Game in Game Theory. However, this technique is not limited to this and can also be exerted in a broader sense such as combining Recurrent Neural Network and Sequential Game in Game Theory. But how does it really work? We will elaborate as below.  In Game Theory, aside from Simultaneous Game, there is one more game called Sequential Game as below:  “sequential game”的图片搜索结果 Figure 2  In this kind of game, each player tries to outsmart other players by a technique call “Back Deduction” which states that each player in a 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 chose the best strategy in the upper layer.  But how can Recurrent Neural Network be combined with Sequential Game?  In Artificial Intelligence, aside from traditional Deep Feedforward Neural Network, there is also Recurrent Neural Network (including LSTM, Neural Turing Machine, etc.). It has its famous long body as below:  Input Neurons  Hidden Neurons  Output Neurons  In the same sense as we elaborated in the combination of Simultaneous Game and DFNN (see Figure 1) which we let player column to move first (let the Output Neurons of Input Neurons column to approximate Target Neurons (1, 0) through back propagation and gradient descent), and then we let player row to move afterward (let the Output Neurons of Input Neurons row 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 input neurons of RNN as the strategies of each player in each layer in the game tree. Take a view as below:  :  “sequential game”的图片搜索结果  Input Neurons  Hidden Neurons  Output Neurons  P1 acts  P2 acts  P1 acts  P2 acts  In layer 4  In layer 2  First, we randomize the strategies of each player (randomize the input neurons. Don’t worry, they will converge to steady points separately in the end). 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) as illustrated as below:  Target Neurons = (0, 1)  Input Neurons  Hidden Neurons  Output Neurons  P1 acts  P2 acts  P1 acts  P2 acts  In layer 4  In layer 2  Then 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) as illustrated as below:  Target Neurons = (1, 0)  Input Neurons  Hidden Neurons  Output Neurons  P1 acts  P2 acts  P1 acts  P2 acts  In layer 4  In layer 2  The 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) as illustrated as below:  Target Neurons = (0, 1)  Input Neurons  Hidden Neurons  Output Neurons  P1 acts  P2 acts  P1 acts  P2 acts  In layer 4  In layer 2  In the end, 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) as illustrated as below:  Target Neurons = (1, 0)  Input Neurons  Hidden Neurons  Output Neurons  P1 acts  P2 acts  P1 acts  P2 acts  In layer 4  In layer 2  At least, we let the machine repeat all the moves aboe.  In the end, the input neurons of P1 in layer 1 is the best strategy for player 1.  It is a miracle!  In actual application, we can take Tic Tac Toe for example. In the present movement, we can first train the machine with future possible strategies and outcomes for each player. When the machine is trained well, we can further exploit the technique as illustrated above and force the machine to find the optimal strategy for the present player.  For more detail information, please see:  <https://github.com/Brownwang0426/Deep_Neural_Game_Theory>  Interesting enough, 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.  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 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. You can also say it is the prototype of Skynet. |