Training Recurrent Neural Network strategies for Iterated Prisoner's Dilemma

Abstract

This manuscript introduces a new Iterated Prisoner's Dilemma strategy based on a recurrent neural network, the "LSTM Homie". Though artificial neural networks have been studied thoroughly in the literature not much attention has been given to recurrent neural networks. The advantage of using a recurrent neural network is that is can retain memory of the past interactions each time it makes a decision. The LSTM Homie is based on a recurrent neural network architecture called Long Short-Term Memory (LSTM), and it is trained on 5503 sequences of $\{C, D\}^{205}$ which correspond to best response sequences to 197 strategies. Many of these strategies are well studied in the literature.

1 Introduction

The Prisoner' Dilemma is a two player game. The two players must choose independently and simultaneously between cooperation, denoted as C, and defection, denoted as D. The payoffs of each strategy are given by:

$$\begin{pmatrix} R & S \\ T & P \end{pmatrix}$$

where $T > R > P > S$ and $2R > T + S$. The most common values used in the literature [2] are $R = 3, P = 1, T = 5, S = 0$. These values are also used in this work.

The Prisoner's Dilemma is commonly studied in its repeated form where the players interact continuously while their prior interactions matter. The Iterated version is called the Iterated Prisoner's Dilemma (IPD).

Artificial neural networks (ANN) were introduced in 1943 [5] and are based on mathematics and algorithms called threshold logic. ANN are becoming more and more popular in recent years mainly to their applications in solving complex problems in a reasonable amount of time [4].

Several sophisticated strategies were introduced using various training and representation methods. A number of these newly introduced strategies were presented using ANN, and their performance was tested in a large tournament of nearly 200 different opponents. These strategies illustrated aspects of strong dominance and stability.

ANN are feed forward neural networks. In feed forward neural networks information flows in a single direction from input to output at one layer a time. RNN can have information travelling in both directions by introducing loops in the network. They are powerful and can get extremely complicated. RNN receive a sequence of values as input and it can also provide a sequence of values as outputs. RNN have gained a lot of attention due to the work of [3] which provided several applications from speech recognition to driveless cars.

- More on literature on strategies being training using reinforcement learning.
- More on literature on neural networks successfully being trained to play the game.
- More on recurrent neural networks, LSTM and some early literature of their application in the PD
- The sequences being best responses to 197 strategies, more of which are strategies from the literature.

The generating of sequences and best response sequences used to train the LSTM Homie is covered in Section 2. The LSTM architecture and the specific models that have trained for this work are presented in Section 3. Finally, the results of the training process and the performance of the "LSTM Homie" are discussed in Section 4.

2 Sequences

The aim of this manuscript is to train an LSTM network as an IPD strategy. LSTMs are capable of learning and remembering over sequences of inputs. The sequences which will be used in this work are a series of IPD actions. More specifically, the inputs to the network are sequences of play of 197 strategies for 205 turns. The outputs of th training process for each input is the sequences which should be played for 1 vs 1 games of length 200 to gain the highest score per turn upon the games conclusion; the best response sequences. Best response sequences will be obtained using reinforcement techniques, specifically genetic algorithms, to continuously improve our sequences during training.

In the IPD a strategy is repeatedly choosing between C and D for a finite number of turns, defined as N. The series of actions/decisions made by a strategy, and subsequently the behaviour of a strategy against a given opponent q can be defined as a sequence:

$$S_q \in \{C, D\}^N \tag{1}$$

Consider a match between a Cooperator and a Defector of N turns. The match can be captured by the sequences:

	1	**2**	**3**	**4**	...	N	U_{S_q}
S_{Defector}	C	C	C	C	...	C	0.0
$S_{\text{Cooperator}}$	D	D	D	D	...	D	5.0

where U_{S_q} is the mean score achieved by a sequence.

Sequences can be used to demonstrate the play of more complicated strategies as well. Consider a player that is a Cycler which plays C, C, D until the match is over against Tit For Tat for $N = 8$:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	U_{S_q}
$S_{\text{Tit For Tat}}$	C	C	D	C	C	D	C	C	2.75
$S_{\text{Cycler } C\text{-}C\text{-}D}$	C	C	C	D	C	D	C	D	2.75

In order to simulate the play of two strategies and subsequently capture the sequences of player an open source library is used called Axelrod-Python [1].

This work uses 197 strategies which can been found in the Appendix alongside a citation of their paper of origin. From these 197 strategies 62 are stochastic and 135 are deterministic. The outcome of a match between two deterministic strategies is the same for every repetition. For example a match between Cooperator and Defector will always lead to a series of cooperations by Cooperator and a series of defections by Defector, and both strategies will always achieve the same mean score. However, in a match with a stochastic strategy this does not longer holds, as the stochasticity of even one strategy can lead to different outcomes. In order to capture and be able to reproduce the results of a match with a stochastic strategy this work makes use of computer seeding.

Consider a match between the stochastic strategy Random and Cooperator. Using different computer seeds allows for different behaviours of Random to be capture as shown by Table 1.

As long as a match is being seeded the behaviour of a stochastic strategy can be reproduced.

Seed 0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	U_{S_q}
$S_{\text{Cooperator}}$	D	D	C	C	D	C	D	C	4.00
S_{Random}	C	C	C	C	C	C	C	C	1.50
Seed 1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	U_{S_q}
$S_{\text{Cooperator}}$	C	D	D	C	C	C	D	D	4.00
S_{Random}	C	C	C	C	C	C	C	C	1.50
Seed 2	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	U_{S_q}
$S_{\text{Cooperator}}$	D	D	C	C	D	D	D	C	4.25
S_{Random}	C	C	C	C	C	C	C	C	1.125
Seed 3	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	U_{S_q}
$S_{\text{Cooperator}}$	C	D	C	D	D	C	C	D	4.00
S_{Random}	C	C	C	C	C	C	C	C	1.50
Seed 4	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	U_{S_q}
$S_{\text{Cooperator}}$	C	C	C	C	C	C	D	D	3.50
S_{Random}	C	C	C	C	C	C	C	C	2.25

Table 1: Seeded sequences of a Random player against a Cooperator.

Thus, the inputs of the training process include 135 sequences of length 205 turns for the deterministic strategies, and 62×10 for sequences for the stochastic strategies. Thus, a total of 755 sequences.

The outputs of the training method will the best response sequences to these 755 input sequences. A best response sequence to a q's sequence is the sequence for which the median score is maximised. More specifically,

$$\max_{S_q^*} : U_{S_q^*} \tag{2}$$

Identifying best responses to some opponents can be a trivial problem, for example the best response sequence for N turns against Cooperator is $\{D\}^N$, however, for most strategies identifying best responses is a complex problem. For this reason, best response sequences are not manually generated but instead a class of ML called genetic algorithms (GAs) is used. GAs fall under a branch of ML called evolutionary algorithms. More generally, genetic algorithms are put into a class of unsupervised reinforcement learning algorithms. These represent techniques of using genetic algorithms for generating solutions to problems that typically revolve around heuristically improving members of a population who represent these solutions. The concept of a genetic algorithm, and more generally an evolutionary algorithm, comes from nature; a survival of the fittest selection competition is created to evaluate a population and kill off the weakest members. After this cull an offspring is created from the most successful population or introduce new members from a predefined source. This process is then repeated for a maximum number of iterations. The specific genetic algorithm used in this work is given by Algorithm 1.

Algorithm 1: Get population of potential best response sequences

Input: $q, N, b, p, t_{\max}, s$
Output: The population at the last generation and the individuals' fitness
begin
  create initial population of individuals
  find fitness of each individual based on U_{S_q}
  sort population based on fitness
  $i \leftarrow 1$
  while $t_i < t_{max}$ **do**
    keep b top individuals
    while *current population size* $< s$ **do**
      select random parents
      use parents to create a kid through crossover
      for *gene in kid* **do**
        mutate gene with probability p
      end
      add kid to population
      find fitness of each individual based on U_{S_q}
      sort population based on fitness $i \leftarrow i + 1$
    end
  end
end

The following need to be specified for Algorithm **??**.

An individuals An individuals of a population is in the form of $\{C, D\}^{205}$. Thus, a gene of an individual is either $\{C\}$ or $\{D\}$.

A population. A population has s number of individuals.

An initial population. An initial population of size s is generated in this work as given by Algorithm 2.

Algorithm 2: Create initial population of individuals S^N

Input: s, N
Output: A population of size s.
begin
  set of cuts $\leftarrow$ s evenly spaced numbers over $[1, N]$
  for $c \in$ *set of cuts* **do**
    first new member $\leftarrow \{C\}^c \cup \{D\}^{N-c}$
    second new member $\leftarrow \{D\}^c \cup \{C\}^{N-c}$
    add both members to population
  end
end

Crossover. Every time two parents are selected a single child is reproduced. A random cut point is selected an the child takes the gene up to that point from the first parent and the rest from the second parent. A graphical representation of the crossover is given by Figure 1.

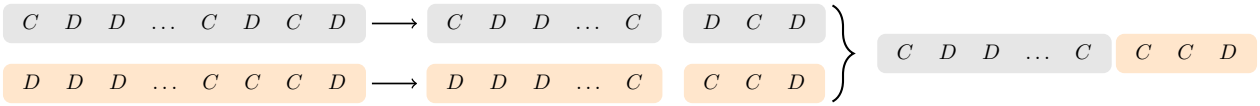

Figure 1: Crossover of two sequences.

Mutation. Every child goes through a mutation process. Every gene has a probability p of being mutated/flipped to the opposite gene.

The genetic algorithm for this work is performed for each strategy considered here. The values used to run the genetic algorithm are given by Table 2.

symbol	parameter	parameter values
N	number of turns	205
p	probability of gene mutating	$0.01, 0.05, 0.1$
b	bottle neck	$10, 20$
$t_{\max}$	maximum number of generations	2000
s	size of a population	$20, 30, 40$

Table 2: The parameters of the genetic algorithm.

For each q algorithm returns the final population. The best response is the sequence that maximised their median score (as given by). However, there can be more than one sequence that achieve the best score for a given q. Thus, the input of the LSTM network are not 760 sequences, but more specifically, there are 5503 rows in the main data set. This has been archived and it is available here.

The next section discuss the LSTM networks, and introduces the specific models trained on these sequences.

3 Recurrent neural networks

We use two LSTM layers, each of which consists of 512 hidden units. Dropout of 0.2 is added after every LSTM layers [16]. We use the Keras deep learning framework [2]. During the optimisation, categorical cross-entropy is used as a loss function and optimisation is performed by ADAM [9]. This optimiser shows an

equivalent final performance to Stochastic Gradient Descent with Nestrov momentum with faster convergence. The prediction is stochastic. In each prediction for time index n, the network outputs the probabilities of every states.

LSTM with unknown input length

LSTM with padded input

4 Results

There are several numerical experiments in this work. The basic tournament and the overall tournament.

4.1 Basic Tournament Results

5 The performance of RNN strategies

References

[1] The Axelrod project developers . Axelrod: 4.4.0, April 2016.

[2] Robert Axelrod and William D Hamilton. The evolution of cooperation. *science*, 211(4489):1390–1396, 1981.

[3] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[4] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[5] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.