# To Play and Cooperate in Imperfect Information Game with Machine Learning

**Ng Argens**

The University of Hong Kong

**Author Note**

Argens Ng, MSc. (Comp Sc.), The University of Hong Kong (UID.: 3035072143)

This paper was prepared in partial fulfillment of the dissertation required by the curriculum of Master of Science (Computer Science)

# Abstract

Machine learning has been an emerging technology bringing success in multiple disciplines recently. It has been a particularly promising solution in pattern recognition problems such as speech recognition, optical character recognition and facial recognition. Its popularity rose as it solved the problem of playing chess and go elegantly, reaching a new height as DeepMind introduced AlphaZero, a general Reinforcement Learning algorithm that is capable of reaching superhuman level given no domain knowledge besides game rules. However, its influence in poker games is disproportionally little, due to the factor of chance and incompleteness of information involved. In this paper, the approach, difficulty and findings thus far of applying Machine Learning on a team-based poker game, bridge (in the aspect of bidding), will be covered.

# Table of Content

# Acknowledgement

I would like to take this opportunity to express my greatest gratitude towards Dr. Dirk Schnieders, my mentor, who has given me immeasurable amount of liberty and support throughout the project thus far. I would also like to thank the researchers at Deep Mind for their pushing of boundary of AI has lit up the path of me pursuing my interest and dream. I am also extremely grateful for the work previous bridge enthusiasts and researches made, in particular Bo Haglund and Soren Hein, authors of double dummy solver, which saved me time and effort that could be put on my main goal.

# Background

Bridge is chosen for three main reasons. Firstly, it is a game of incomplete information, where each player is only shown one fourth of the cards half the time, and half the cards at other times. The game, however, is still played with all 52 cards. Secondly, it is a team game, where players have to cooperate and compete in pairs. Thirdly, the game has a systematic and absolute “par” value, which allows us to infer “wins” and “losses” in a game otherwise dictated by chance.

We will now dedicate the next section to explain the game in greater detail.

# Rule of Bridge

The game, in essence, is a constant-sum trick-taking game in which each pair of players attempt to take the most tricks in combined effort, in order to receive scores correlated to the number of tricks. Each game, or deal, is consisted of two parts, namely ***bidding*** and ***card-play***. The winning pair of bidding is called declarer and in descending order of importance, they have the following objectives:

1) Reach the required number of tricks decided by the ***contract***
2) Reach a contract higher than or equal to game or slam level if possible
3) Get as many tricks as possible

The objective of the other pair is hence to avoid the declaring pair from achieving above goals.

## Contract

A contract consists of a level (an integer ranging from 1 to 7) and a trump suit (chosen from Clubs, Diamonds, Hearts, Spades and No-Trump). The integer indicates the number of additional tricks the declaring side promised to take beyond the minimum number, which is 6. Hence for example, a level of 3 means that the declaring pair promised to take 9 or more tricks, only then would they receive positive scores.

If the trump suit is Hearts or Spades and the level is higher than or equal to 4, the declaring pair would earn a game bonus if the contract is made (ie. the number of tricks required is reached). Similar game bonuses are granted to contracts made at level 3 or higher without a trump suit, or 5 or higher with Clubs or Diamonds being trump.

If the contract is at 6-level and made, the declaring pair earns a small slam bonus; And if the contract is at 7-level and made, they earn a grand slam bonus. All of these bonuses are higher if the pair is at vulnerable position, occurring twice every 4 deals.

Trump suit will be explained in greater detail in the section of card-play.

## Bidding

In bidding, each player bids in clockwise direction, with each bid being either a contract, a pass, a double or a redouble. A pass indicates a lack of interest in bidding and hence 4 consecutive passes without a contract or 3 consecutive passes otherwise signals the end of bidding.

If a player chooses to bid a contract, such contract must be either higher in level or same in level and higher in trump suit than the previous contract. The trump suits in ascending order are Clubs, Diamonds, Hearts, Spades and No-Trump, where No-Trump means that the declaring side is willing to play in the absence of a trump suit.

A player may also choose to double opponent's contract or redouble a doubled contract on his/her own side. This is similar to a raise and re-raise of bet in other poker games. 3 passes must also follow a double or a redouble for it to take effect.

## Card-play

The player in the declaring pair who first bed the suit of the final contract is the declarer, whereas the declarer's partner is called dummy. The dummy has no right to play cards and such right is given to the declarer, who shall play 2 deck of cards, or 2 hands.

The phase of card-play starts as the player to the left of declarer play a card freely on the table, followed by dummy revealing his/her hand. Card-play then commence by each hand playing a card in the clockwise direction, with the requirement that each hand must play a card from the same suit as the first card if possible. Otherwise any played card is deemed to lose, with the exception of a trump card. The hand that plays the highest card is the winning hand, and can freely play a card in the next turn.

The highest card in each suit is Ace, followed by King, Queen, Jack and so on. There is no ranking among the suits, with the exception that trump suit being higher than the rest.

## Par Value

Par value is the Nash-equilibrium of each deal in the perspective of N-S pair if players have complete information. Neither of the pairs can bid a better contract or play better to get a higher score than par value. This might be +600 for a 3-NoTrump by North making 9 tricks at vulnerable, or -400 for a 5-Clubs by East making 11 tricks at non-vulnerable. This par value is fundamental in constructing loss function during the training of neural networks.

# Literature Review

*Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm (Silver et al., 2017)*

In this paper, AlphaZero, the algorithm capable of achieving superhuman level of play in Chess, Go and Shogi is introduced. More astonishingly, it reached such level with minimal domain knowledge and self-play in merely 24 hours. Due to its wide range of application and highly successful outcome, the architecture of AlphaZero will be the backbone of our BridgeZ program.

In AlphaZero, the core search will be conducted by a MCTS (Monte Carlo Tree Search) algorithm, backed by neural network $f$. The network is fed with layers of raw input and game rules, stacked as $s$ and outputs an expected value v and a policy vector P, given parameter set $\theta$. The expected value v can be seen as the prediction of the final outcome given the current situation and should ideally be 1 (current player wins) or 0 (opponent wins). The policy vector P is the expected probability that each move is chosen if the best play is made.

$$(\mathrm{P},\ \mathrm{v}) = f_\theta(s)$$

At each iteration, the actual game outcome $z$ and the search probability of MCTS $\pi$ back-propagates from the final outcome, adjusting the parameter set $\theta$ of the network, using the following loss function:

$$l = \left(z - v\right)^2 - \pi^T log P + c||\theta||^2$$

Where c is the L2-regularizer.

Besides architecture, the literature also tells the advantages of playing chess, shogi and go using RL (reinforcement learning). On the same token, we can predict the challenges of devising an algorithm that plays bridge using RL:

1) The game is asymmetric
2) Wins and losses have values (ranging from -24 to +24 IMPS)
3) Cards have ranks while chess pieces do not

These challenges shall be paid extra attention in future research.

*Artificial Neural Network for Solving Double Dummy Bridge Problem*

**Summary**

Only one paper that is both related to bridge and neural network is found. It aims at predicting the maximum number of tricks N-S pair can take with North being a declarer, under the absence of a trump suit. This is achieved by inputting all four hands into the network and train using the difference with actual result.

Throughout the paper, multiple representations of inputs and outputs are tested. For input, the hands are coded as follow:

1) Suits and values of the cards are coded as real numbers in the range [0.1, 0.9]; Hence two real numbers are needed for each card, 26 for each hand. With a total of four hands, the shape of input is hence (104, ). (Remark: it is recorded as (52, 4) in original literature to differentiate from input representation (3)).

2) North is coded as 1.0, South 0.8, West -1.0 and East -0.8. Hence by assigning the position code to each card, only an array of 52 real numbers is needed to code a deal. The input is thus an array of (52, ).

3) The first 52 integers represent the pair holding the card (1.0 for N-S, -1.0 for E-W), while the next 52 integers represent the player in each pair (1.0 for N or W, -1.0 for S or E). Hence an array of (104, ) is able to represent the distribution of all the cards in a deal.

For output, the number of tricks are coded as follow:

1) The number of tricks are mapped to the range of real values [0.1, 0.9] and the problem is treated as a regression problem.

2) One output neuron is dedicated to each possible number of winning tricks (0 to 13) and the problem is treated as a classification problem.

Besides varying input and output representations, the author also sometimes adds hidden layers into the network. In terms of accuracy, based on test data and with a margin of error of 2 tricks, a network having the input representation of (2), output of (1), together with a hidden layer of 25 neurons performs the best with 95.81% accuracy.

The second best performing network uses an input of (3) and output of (1), together with 2 hidden layers of 30 and 4 neurons respectively, achieving an accuracy of 95.64%. The next best performing network has an accuracy of 94.77% using an input of (1), output of (1) and a hidden layer of 52 neurons.

**Interpretation**

In terms of output representation, it is clear that treating the problem as a regression problem is the more successful approach. On the other hand, the numbers are not so clear when it comes to input representation. However, if we take the objective of our own research into account, then it appears that both (1) and (2) have too much human interpretation. For example, (1) assumes that the suits are numerically ordered, and (2) assumes that the positions are numerically ordered. Ideally, we hope that the network could learn to play from self-play only, and these input representations have too much interpretation and are also arguably incorrect. (3) is a condensed representation of the raw hands and we expect the adapted input representation to be along the same line.

# Technology

To conduct our research, Python will be used along with Keras framework running on CUDA and TensorFlow which provides easy sandboxing and close-to-native speed with GPU acceleration.

# Project Plan

In this section, the actual plan of the project will be elaborated in detail.

The goal of the project is to design, create and train a network to bid in a game of bridge. Unlike other poker games, bridge has a par value, which can be an absolute indicator of the success of each bidding. Bridge also requires cooperation, which limits the usefulness of bluffing while encourages communication between agents. The element of communication is expected to be a highlight in our research.

To achieve our goal, we will need a large number of hands to prevent overfitting and remove the factor of chance. We will also need to devise a suitable architecture, input and output representation, as well as supporting framework for bidding, which is in essence a communication channel.

Once the above requirements are satisfied, networks randomly initialized will then be placed in repetitions in four positions, with only one hand visible to them at one time. They will then make a bid in turn using MCTS (Monte Carlo Tree Search), reach a contract in the end, compare the contract with the par value and receive rewards and losses respectively. This shall be repeated with hopefully better and better performing networks, sometimes swapping in previous generations to avoid converging in local maxima.

To train our network, the loss function will be:

$$l = (z - v) - \pi^T log P + c||\theta||^2$$

While the network can be seen as a function:

$$(\mathrm{P}, \mathrm{v}) = f_\theta(s)$$

*where* $\boldsymbol{\theta}$ *is the parameter set of the network;*
$\boldsymbol{z}$ *is the actual value of the game (1 meaning the current player ultimately wins, -1 otherwise);*
$\boldsymbol{\pi}$ *is the search probability of MCTS;*
$\boldsymbol{v}$ *is the expected value of the game, an approximate of* $z$*; and*
$\boldsymbol{P}$ *is the policy vector containing probability of each move being chosen, an approximate of* $\pi$

This is slightly different from the loss function in ***"Mastering Chess and Shogi"***, where we use $(z - v)$ instead of $(z - v)^2$ since we wish to preserve both the scale and sign of the difference.

# Project Progress

## Hand Generation

Generating random hands is a trivial task using pseudorandom number generators. However, we also need to find the maximum number of tricks for each side using different trump suits (including No-Trump). This is achieved by using the **double dummy solver** by **Bo Haglund** and **Soren Hein** by changing a small number of codes. Their contribution to bridge solving is deeply appreciated.

## Output Representation

For output representation, policy vector and expected value will be outputted, as stated in the above section. Illegal moves would then be masked with 0s in the policy vector and the remaining probabilities are scaled accordingly.

## Input Representation

A suitable input representation ought to be able to capture the information, compact and easy to train and transform, while containing little or none of expert knowledge. To find such representation, the problem is first simplified to finding the maximum number of tricks N-S pair can get with North being the declarer, in the absence of a trump suit, identical to that of ***"Artificial Neural Network for Solving Double Dummy Bridge Problem"***.

At first, each hand is one hot encoded and hence is an array of size 52 and has 13 1's. The encoding is conducted in the order of suits, followed by ranks. Hence for example, if a hand has all the Clubs of the deck, then this hand will be coded as 13 1's followed by 39 0's. With 4 hands in each deal, the input shape of the network is thus (208, ). The training (MSE) mean squared errors of the networks using such input representation are 8.30, 8.75, 10.53. The networks are also seriously over-fitted but this may be due to having a deep network and not enough training data.

The second representation tested is similar to (2) suggested in the paper "Artificial Neural Network", where each position is coded with a real number, and an array of 52 real numbers could then represent where all the cards lay in a deal. The networks result in MSE of 8.25, 6.90, 9.16, arguably better than the previous approach.

The next representation change also changes the network architecture. It is also a one hot encoding of all the cards and each hand is hence also a 52-array. However, these arrays are not linked head to tail but stacked together, forming a 2D input of 52 by 4. Since the hands at the edge then cannot interact with each other, they are hence repeated in the following manner: E, S, W, N, E, S, W, and the whole input becomes a 2D input of 52 by 7. The network also becomes a convolutional neural network. The resulting MSE ranges from 6.90 to 9.62, with the average being 7.72. While the MSE is significantly lower, there are still significant cases of over-fitting.

The last input representation that is come up with has the shape of (7, 13, 4). It is similar to the previous representation but the suits are separated into channels and thus cannot directly interact with each other. This is similar to the game of bridge where Ace of one suit holds no particular effect to the Queen of another suit. This can be argued as an expert knowledge of the game but in essence is only a more accurate representation of the cards. The resulting MSE ranges from 5.81 to 7.39 with 3 hidden layers, 6.07 to 6.73 with 1 hidden layer (64 neurons), and 4.60 to 7.08 with a 16-neuron hidden layer.

With the representation of the cards mostly decided, we still need to encode the meanings of biddings to enable networks to "communicate" using biddings. By tokenizing the meaning of biddings, only a handful of extra neurons is needed to handle the extra information.

(1) 16 pairs of neurons shall be used to capture the lowest number and highest number of cards of each suit in each hand represented in bidding.
(2) Another 16 triplets will be used as flags for the presence of Aces, Kings and Queens, again in each suit and each hand.
(3) Additionally, 4 pairs will be used for the lower and upper range of HCP (high card point, with presence of each Ace being 4, King being 3, Queen being 2 and Jack being 1) in each hand.
(4) Two extra flags will be used for forcing and relay;
(5) And 16 layers of 38 one hot encoding of bidding history will be used such that our agent would not become reliant on bidding interpretation — learning from raw bidding is ultimately the goal.

## Architecture Optimization / Genetic Algorithm

We take the opportunity to find the ideal architecture in this simplified problem to experiment genetic algorithm, which may also prove useful in later stages.

The standard approach of genetic algorithm is used, that is — initialize, selection, cross-breed, mutation and evaluate.

### Initialization

To standardize the hyper-parameters which can be positive integers and real numbers of different ranges (batch-size is usually in the range of 50 to 100, while learning rate is usually less than 0.1), we randomize a real number from (-50, 50) and as networks are built, each number is transformed into suitable format in the suitable range using sigmoid function. Two extra hyper-parameters in the form of Boolean is also defined to fully utilize the procedural approach of building network in Keras. The two Boolean values are essentially flags that determines building or not building an extra hidden layer, dropout and Gaussian noise layer bundle.

### Selection

The selection paradigm chosen is simply elitism, where the lowest 10% in terms of MSE of the population is chosen. Dictionaries of {network-name: parameters, results} of whole population and each generation is kept at all time in order.

### Cross-Breed

To cross-breed, each pair of network in the chosen networks are given two chances to breed. For each opportunity, the parameter of the offspring will at equal opportunity choose to inherit the parameter from either network.

### Mutation

The networks are allowed to mutate both at initialization and cross-breed, with different mutation chances. In our case, initialization had a mutation chance of 1% per flag, and cross-breed 3% per parameter. To actually mutate, during initialization, if a random number is lower than 0.01, then the flag that builds extra bundle of layer is switched on; As for cross-breeding, whenever a parameter is inherited from parents, and a random

number is lower than 0.03, then the parameter is either randomly added with [-50, 50], or flipped if the parameter is a flag. Mutation ensures the variability of the population.

**Evaluation**

For evaluation, each network is fitted with the same 700,000 deals and tested with 20,000 deals to receive a MSE value. Those with the lowest testing MSE values are chosen as explained in the selection section above.

**Result**

In the end, a third-generation network achieved a MSE of 3.79 while a total of 12 networks have a MSE lower than 4.5 — one of the networks is a first-generation network, three being second-generation and eight being third-generation.

## Bidding Framework

Following the tedious work in designing input representation, all that is left in a working bidding framework is the interpretation of bidding, or the tokenization of bidding into meanings. This shall be done by a dictionary matching longest matching string to all the meanings in different suits in different hands. This dictionary, in the current stage, would be inputted manually using a python snippet in accordance with the Hong Kong Youth Team bidding system 2019.

# Evaluation

To evaluate the success of our agent in the end, the par value of the deals can be compared with the actual results in agents' biddings, with each absolute error holding significance in calculating the final error.

We can also invite bridge players at a competitive level (including Hong Kong Youth Team and Hong Kong Team members) to play against our agent to observe qualitative feedbacks.

# References


1. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K. & Hassabis, D (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. arXiv:1712.01815v1 [cs.AI] 5 Dec 2017

2. Mossakowski, K. & Mańdziuk, J. (2004). Artificial Neural Networks for Solving Double Dummy Bridge Problems. Artificial Intelligence and Soft Computing, ICAISC 2004, doi: 10.1007/978-3-540-24844-6_142