AI-FML Agent with Patch Learning Machanism for Robotic Game of Go Application

|  |  |
| --- | --- |
| Chang-Shing Lee, Yi-Lin Tsai, Mei-Hui Wang  Dept. of Computer Science and Information Engineering  National University of Tainan  Tainan, Tawain  [leecs@mail.nutn.edu.tw](mailto:leecs@mail.nutn.edu.tw) | Naoyuki Kubota  Dept. of Mechanical System Engineering  Tokyo Metropolitan University  Tokyo, Japan  kubota@tmu.ac.jp |

*Abstract*—In this paper, we propose an AI-FML agent with patch learning mechanism for robotic game of Go applications. The AI-FML agent contains three kinds of intelligence, including perception intelligence, cognition intelligence and computational intelligence for robotic application. The patch learning (PL) consists of three steps, first the PL trains an initial global model, then it trains a patch model for each identified patch, and finally it updates the global model using training data that do not fall into any patch. In addition, PL can be implemented using AI-FML agent for robotic applications. We adopt the Google DeepMind Master 60 games to be the training data and testing data set. The experimental results show the AI-FML agent with patch learning can improve the performance of regression for robotic game of Go applications.

Keywords—AI-FML, Agent, Patch Learning, Fuzzy Markup Language, Game of Go

# Introduction

Machine learning has been widely used in many real-world applications, but training a high performance machine learning model is usually an iterative process, relying on experience and trial-and-error, if its performance is not satisfactory, then some remedies are taken to enhance it [Jerry arXiv 2019 Patch Learning]. In [1], the authors define a patch as a connected polyhedron in the input domain. For example, a patch in a 1D input domain is an interval, and a patch in a 2D input domain can be a rectangle, an ellipse, and so on. The idea of Patch Learning (PL) introduced by Jerry Mendel is as follows: *Consider a sculptor who is sculpting a human figure, after his first pass at this, the sculptor examines the entire figure and notices that improvements need to be made to certain parts of the figure. He zooms into the certain parts that need more work, after which he blends in the refined portions of the figure with the rest of the figure. He continues such iterative refinements until he is satisfied with the entire figure. Each patch in PL is analogous to a part in the figure that needs more work* [1]. Based on the concept of PL, it consists of three steps: Train an initial global model using all training data first, then identify patches from the initial global model and train a patch model for each patch, finally update the global model using training data [1]. However, generally identifying the patch locations is a very challenging task, and not every problem can be easily visualized. Jerry Mendel [2] introduces how PL can be performed using rule-based fuzzy systems, because it is easy to initialize patch candidates in a fuzzy system.

****

1. Patch Learning架構流程 [TFS 2019]

With the success of AlphaGo [WCCI 2020 paper], there has been a lot of interest among students and professionals to apply machine learning to gaming and in particular to the game of Go. The Open Go Darkforest (OGD) prediction platform has been constructed since 2016 [3, 4, 5, 6]. The OGD cloud platform, including a Facebook AI Research (FAIR) dynamic Darkforest (DDF) AI bot and a FAIR ELF OpenGo AI bot prediction mechanism, has the ability to predict the top five selections for the next move. The AI bot provides each selection with real-time win rate, simulation numbers, and the top matching rate [6]. The IEEE Computational Intelligence Society (CIS) funded Fuzzy Markup Language (FML)-based machine learning competition for human and smart machine co-learning on game of Go in IEEE CEC 2019 and FUZZ-IEEE 2019. The goal of the competition is to understand the basic concepts of an FML-based fuzzy inference system, to use the FML intelligent decision tool to establish the knowledge base and rule base of the fuzzy inference system, and to optimize the FML knowledge base and rule base through the methodologies of evolutionary computation and machine learning [wcci2020].



1. *connecting multiple simple base learners in parallel* [TFS 2019]

There are some strategies to enhance the performance for machine learning mechanisms, such as: *using a single deeper model*, *using a single broader model*, *connecting multiple simple base learners in parallel*, or *connecting multiple simple weak learners in series*, and so on [Mendel TFS 2019].



1. *connecting multiple simple weak learners in serial* [TFS 2019]

Regression, ensemble methods, and deep learning are important machine learning methods for data scientists [10]. Extreme gradient boost (XGBoost) is a scalable tree boosting system that provides state-of-the-art results on many problems [11]. For example, Song et al. [12] proposed an optimization model combining XGBoost algorithm, Jiang et al. [13] proposed a genetic algorithm (GA)-XGBoost, and Zhang et al. [14] proposed XGBoost-based algorithm to recognize five indoor. Fuzzy Markup Language (FML) is based on XML and has been a standard of IEEE 1855 since 2016. It is a powerful tool for human to construct the knowledge base and rule base on various real-world applications [wcci 2020 20, 21]. We further propose the AI-FML agent to integrate AI tools, i.e., using AI- FML tools to construct the knowledge base and rule base of the fuzzy inference system, with Internet of Things (IoT) at IEEE WCCI 2020. The AI-FML agent can communicate with IoT devices or robots, and it has been introduced to the learning course of 5-grade computer studies at Rende elementary school in Taiwan from 2019 to 2020.

The remainder of this paper is as follows: Section II presents the structure of ANFIS-based patching learning mechanism. Section III describes the patch learning mechanism for game of Go data set. Section IV proposes the PL-based AI-FML agent for robotic application on predicting the win rate of Go game. Section V shows the experimental results. Finally, section VI draws the conclusion and discussion.

# Structure of ANFIS-Based Patch Learning Mechanism

This section presents the structure of ANFIS-based patch learning mechanism.

## Patch Learning Mechanism

The traditional design of ANFIS is global, in the sense that representative training data are used to optimize the input membership functions and consequent parameters over input domain, and the performance metrics are optimized using all training data [TFS 2019]. The ANFIS-based patch learning mechanism begins with a globally designed rule-based fuzzy system, but then locates the patches in *m* domains which have contributed the most to the performance metrics. The ANFIS-based patch learning is then using a subset of training data that are in that patch to improve the performance metrics [TFS 2019]. Finally, the global rule-based fuzzy system is updated, using the remaining training data that have not been used by any patch. Fig.1 shows the structure of patch learning for predicting the win rate of Go games.



1. *Structure of Patch Learning for predicting the gane of Go Data set.*

In Fig. 4, the structure of patch learning connects multiple models both simple base learners in parallel and simple weak learners in serial to improve the learning performance of ANFIS. The ANFIS-based patch learning will be embedded to the AI-FML agent and applied to predict the win rate of game of Go data set. Next subsection will briefly introduce the structure of ANFIS.

## Adaptive Network-based Fuzzy Inference System (ANFIS) Mechanism

The adaptive network-based fuzzy inference system (ANFIS) is one of artificial neural network models that is based on Takagi–Sugeno fuzzy inference system, and developed in the early 1990s [SMC 1993][WIKI]. The ANFIS integrates both machine learning model and fuzzy logic principles, it has potential to capture the benefits of both in real-world applications. Its inference system corresponds to a set of fuzzy rules that have learning capability to approximate nonlinear functions and is considered to be a universal estimator [SMC 1993]. 

1. *Structure of ANFIS with five layers[??].*

The structure of ANFIS contains two parts, including premise part and consequence part. In more details, the architecture is composed by five layers. Fig. 5 shows an example of five-layered ANFIS structure. The first layer is called fuzzification layer, and takes the input values and determines the membership functions belonging to them. The membership degrees of each function are computed by using the premise parameter set. The second layer is the rule layer and responsible of generating the firing strengths for the rules. The third layer is to normalize the computed firing strengths, by diving each value for the total firing strength. The fourth layer takes as input the normalized values and the consequence parameter set. The values returned by this layer are the defuzzificated layer and those values are passed to the fifth layer to return the final output. In this paper, we utilize the PyTorch framework to implement the ANFIS-based patch learning model.

# ANFIS-based Patch Learning for Go Game Data Set

## Introduction to Data Set from IEEE WCCI 2020

The authors would like to thank the financially support sponsored by the Ministry of Science and Technology of Taiwan under the grants MOST 108-2218-E-024-001.

In [SMC1993], Jang mentioned that: *“… fuzzy conditional statements* are expressions of the form ***IF A THEN B***, where *A* and *B* are labels of fuzzy sets characterized by appropriate membership functions…,” and, “… fuzzy *if-then* rules are often employed to capture the imprecise modes of reasoning that play an essential role in the human ability to make decisions in an environment of uncertainty and imprecision….”

In this paper, we adopt the data set from [wcci 2020] competition that is briefly described as follows: With the success of AlphaGo, there has been a lot of interest among students and professionals to apply machine learning to gaming and in particular to the game of Go. Several conferences have held competitions human players vs. computer programs or computer programs against each other. The goal of this competition includes: (1) Understand the basic concepts of an FML-based fuzzy inference system. (2) Use the FML intelligent decision tool to establish the knowledge base and rule base of the fuzzy inference system. (3) Use the data predicted by Facebook AI Research (FAIR) Open Source Darkforest AI Bot as the training data. (4) Use the data predicted by Facebook AI Research (FAIR) Open Source ELF OpenGo AI Bot as the desired output of the training data. (5) Optimize the FML knowledge base and rule base through the methodologies of evolutionary computation and machine learning in order to develop a regression model based on FML-based fuzzy inference system.

## Pre-processing Partition Function for Game Data Set

There are different kinds of partition functions for partitioning the input domains [Information Science 2019]. For example, *crisp partition function* or *Type-1 partition function*. A crisp partition of real variable is comprised of non-overlapping adjacent regions that are intervals of real numbers, where the membership degree belonging in each region is unity, and is zero outside of that region [Information Science 2019]. In addition, a *first-order uncertainty partition* of the real variable is comprised of overlapping intervals, where one is absolutely certain about where the overlap begins and ends, so that the degree of belonging in each of overlap is a real number that is an element of [0, 1]. A first-order uncertainty partition is characterized by non-rectangular *type-1 partition function* that overlap with their nearest neighbors, and it is not a mathematical partition. Fig. 6 shows an example of type-1 partition functions VL, L, M, H, and VH.



1. *Type-1 partition functions VL, L, M, H, and VH.[??].*

The data set for game of Go is produced as follows: (1) AlphaGo Master series: 60 online games Master in Dec. 2016 and in Jan. Over one week, AlphaGo played 60 online fast time-control games. AlphaGo won this series of 60 games. (Website: <https://deepmind.com/research/alphago/match-archive/master/>). (2) Each Game Data file includes the prediction by Darkforest AI Bot and EFL OpenGo AI Bot for one game. (3) MoveNo is the move number. MoveNo only lists "odd" numbers (i.e., 1, 3, 5, ...) because each row corresponds to a pair of one Black move and one White move. That is, the row with the MoveNo 1 corresponds to the Black first move (i.e., B1) and the White first move (i.e., W2). The row with the MoveNo 145 corresponds to the Black 145th move (i.e., B145) and the White 146th move (i.e., W146). If the final MoveNo is "odd", "White's information of the last row" will be vacant. (4) Each row includes eight values (i.e., DBSN, DWSN, DBWR, DWWR, DBTMR, and DWTMR were the outputs from Darkforest. EBWR and EWWR were the outputs from ELF OpenGo). (5) DBSN: The number of simulations for Black. DWSN: The number of simulations for White. DBWR: The win rate of Black. DWWR: The win rate of White. DBTMR: The top-move rate of Black. DWTMR: The top-move rate of White. (6) EBWR: The win rate of Black. EWWR: The win rate of White. (7) The number of rows represents the length of the game and is different among game data files (8) Choose any 40 Games from 60 Games as the training data and the remaining 20 Games as the testing data. The inputs are the number of simulations (DBSN, DWSN), the win rates (DBWR, DWWR), and the top-move rates (DBTMR, DWTMR) predicted by Darkforest. The desired outputs are the win rates, (EBWR, EWWR) predicted by ELF OpenGo AI Bot. Table I shows partial dataset for game of Go application. There are 3,758 data adopted from Master Game 1 to Game 40 for the training dataset. The Game 41 to Game 60 with 1880 dataset will be adopted to be the testing data in this papper.

1. Partial dataset for game of go prediction

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No | *DBSN* | *DWSN* | *DBWR* | *DWWR* | *DBTMR* | *DWTMR* | *EBWR* | *EWWR* |
| 1 | 3863 | 2274 | 0.52733 | 0.48533 | 0 | 1 | 0.49636 | 0.50884 |
| 2 | 9283 | 7866 | 0.51529 | 0.48532 | 0.5 | 1 | 0.45779 | 0.54844 |
| 3 | 11395 | 6798 | 0.51265 | 0.47717 | 0.6667 | 1 | 0.45671 | 0.57235 |
| 4 | 4499 | 10703 | 0.51885 | 0.46988 | 0.75 | 1 | 0.49086 | 0.5201 |
| 5 | 7388 | 20017 | 0.5288 | 0.46679 | 0.8 | 1 | 0.47218 | 0.53649 |
| 6 | 20098 | 9693 | 0.53309 | 0.46602 | 0.8333 | 1 | 0.47249 | 0.52224 |
|  | | | | | | | | |
| 3754 | 7211 | 12038 | 0.92754 | 0.28941 | 0.797 | 0.812 | 0.9566 | 0.08276 |
| 3755 | 20017 | 20134 | 0.81634 | 0.18352 | 0.7985 | 0.8134 | 0.87651 | 0.04687 |
| 3756 | 7691 | 12699 | 0.80645 | 0.15954 | 0.7926 | 0.8074 | 0.92438 | 0.18538 |
| 3757 | 17076 | 4603 | 0.85503 | 0.09789 | 0.7941 | 0.8088 | 0.9107 | 0.18915 |
| 3758 | 9015 | 20017 | 0.90843 | 0.10819 | 0.7956 | 0.8102 | 0.98549 | 0.00822 |

Fig. 7 shows the type-1 partition with 64 Gaussian function for the game of Go data pre-processing.



1. *Type-1 partition functions with 64 Gaussian functions for Go.*

There are two criteria utilized for data pre-processing in this paper. The first criteria is “*if the final prediction result of Darkforest is mistake, then the dataset of this game will not adopted for training data or testing data*.” For example, the dataset of Game 15, Game 25 and Game 29 will not adopted for the training data. Fig. 8 shows the prediction results of Darkforest for the three games.

|  |
| --- |
| (a) |
| (b) |
| (c) |

1. Prediction results of Darkforest: (a) Game15 (b) Game25 (c) Game29

In addition, the Darkforest also makes incorrect predictions for Game 48 and Game 60. Fig. 9 shows the prediction results for the two games. Therefore, we remove the five games from the dataset in this paper.

|  |
| --- |
| (a) |
| (b) |

1. Prediction results of Darkforest: (a) Game48 (b) Game60

The second criteria is “*if the predicted moves of White or Black are different between ELF OpenGo and Darkforest that are more than 40%, then the dataset for the game will not adopted in the training data or testing data*.” For example, if one game with 100 moves, but there are 40 moves predicted by ELF OpenGo and Darkforest are different, then this game will be removed from the dataset. Based on this criteria, the Game 8 and Game 36 are removed from training dataset. In addition, Games 44, 47, 56 and 57 are removed from the testing dataset in this paper. Moreover, we check the missing data from the dataset. If one game with the final move and the first move are the same as Black or White, then we move the final move from this game. After checking, we remove 19 data from training dataset and 6 data from testing dataset. Finally, we adopt 3143 data for the training dataset and 1239 data for the testing dataset in this paper.

# PL-based AI-FML agent for Robotic Game of Go Application

## ANFIS for AI-FML Agent on Game

The core fuzzy rule-based system of PL-based AI-FML agent is ANFIS and composed the following functional blocks: (1) a knowledge base with database, (2) a rule base, (3) the inference operators on the rules, (4) a fuzzification process, and (5) a defefuzzification process [SMC 1993]. The steps of fuzzy reasoning perform inference operators upon fuzzy *if-then* rules that performed by AI-FML agent is as follows: (1) retrieve the input value for fuzzy variables with membership functions on the premise part to computer the membership degree of each linguistic label, (2) combine the membership degrees on the premise part to get the weight of each rule, (3) gererate the qualified consequents of each depending on the weights, and (4) aggregate the qualified consequents to produce the final output..



ANFIS結合了兩種機器學習的優勢，分別為模糊邏輯與神經網絡。並透過神經網路學習方法來調整模糊推論系統的參數，可調控的參數包含每個高斯隸屬函數mean與variance。表二為ANFIS結合倒傳遞演算法虛擬碼。輸入為訓練集40盤棋局資料以及必要參數，n\_inputs(特徵數量)與n\_rules(規則數量)。輸出為預測目前黑棋勝率。演算法第一步神經網路初始化。第二步類神經元執行輸入與相對應的模糊集合的歸屬程度計算。第三步規則層類神經元Π執行T-norm運算，執行的是模糊規則前鑑部的啟動強度計算。第四步正規化層類神經元N，執行的是將啟動強度正規化之運算。第五步推論層類神經元執行的是函數式加乘，每個模糊規則後項之運算。第六步輸出層只有單一個類神經元，計算前一層中類神經元輸出值的總合，執行解模糊化以作為最後網路的輸出值。第七步預測並計算每一筆資料的Error並得出損失值，並藉由損失函數找出梯度慢慢的修正每個模糊規則的mean與variance。第八步判斷是否停止訓練，當到達訓練次數即終止訓練。

1. ANFIS 演算法架構

|  |
| --- |
| **Input:**  訓練資料: *X\_train* // Game1~Game40局圍棋資料  輸入數量: *n\_inputs* = 6 // DBSN、WBSN、DBWR、WBWR、DBTMR、DWTMR  規則數量*: n\_rules* = 64  **Output:**  預測目前黑棋勝率   1. 神經網路初始化    1. 建立mean與variance 陣列       1. *meanArray* = Array(*n\_inputs\* n\_rules*);       2. *varianceArray* = Array(*n\_inputs\* n\_rules*);    2. FOR(*i* ← 0 to *n\_inputs\* n\_rules*) // 隨機0~1初始化mean與variance 陣列       1. *meanArray*[*i*]= random(0,1);       2. *varianceArray* [*i*]= random(0,1); 2. 輸入層計算歸屬程度    1. // 使用高斯歸屬函數 3. 規則層T-norm運算    * 1. *.* 4. 正規化層 5. 推論層 6. 輸出層 7. 計算loss並計算梯度更新權重 8. 返回step2直到訓練結束 |

## Patch Learning on Game

Patch Learning是一個新的機器學習的一種想法，搭配模糊推論讓整個訓練出來的模型更具人性化。表三為Patch Learning其演算法架構，首先第一步我們需要使用所有的訓練資料訓練一個全域模型。訓練模型採用ANFIS架構並設定六十四條規則訓練一千代。接著透過梯度下降演算法來修正模型參數，並使用均方誤差(mean squared error, MSE)來評估模型好壞。透過計算MSE我們可以知道每次訓練過後模型所推論出來的勝率是否趨近ELF OpenGO所預測出來的勝率。因此MSE計算出來的值越小代表越接近ELF OpenGO的勝率。第二步從全域模型中找出找出某個區段表現最差的區域，也就是MSE最大的區段。並為每個此類的Patch訓練一個區域的模型，訓練模型我們一樣採用ANFIS架構並設定六十四條規則訓練一千代。第三步計算每一手的誤差並觀察是否還有學習不佳的區域，若有則返回第二步繼續訓練，反之停止訓練。整個模型訓練完畢後我們就可以使用測試資料來測試我們模型。在使用Patch Learning模型預測前我們要找出該段輸入相對應的Patch。假設我們有k個Patch若該筆資料落於第一個Patch我們就採用第一個Patch進行預測。若該筆資料皆無落在預訓練的Patch中則使用全域模型進行預測。



1. Patch Learning演算法架構

|  |
| --- |
| **輸入**  訓練資料: *trainData* // Game1~Game40局圍棋資料  測試資料: *testData* // Game41~Game60局圍棋資料  **輸出**  Patch Learning模型對圍棋資料的勝率預測  **方法**  // 訓練Patch Learning模型   1. 訓練全域模型    1. 使用ANFIS進行N代訓練trainData    2. 計算出MSE並評估模型好壞 2. Patch區間訓練    1. 從*trainData*中挑選出學習不佳的資料patchData    2. 使用ANFIS進行N代訓練patchData    3. 計算出MSE並評估模型好壞 3. 挑選學習不佳區間並返回Step2進行訓練直到所有區間訓練完畢   // 使用Patch Learning模型進行預測  for *t* = 1, ..., do  = True;  if 落入第k個Patch then  使用第*k*個Patch模糊模型進行預測;  = False;  Break;  end  if *useGlobal* == True then  使用更新的AI-FML全域模糊模型進行預測;  end  end |
|

圖19(a)與圖19(b)分別為第一型模糊集合**Patch切割**及第一型模糊集合的**First-Order規則切割**。圖20(a)及圖20(b)分別為第一型模糊集合的**First-Order激勵劃分區域**及第一型模糊集合的**Second-Order激勵劃分區域**。本計畫將融合 **AI-FML機器學習架構**、**Patch切割、First-Order規則切割、First-Order激勵劃分區域**及**Second-Order激勵劃分區域**等相關機制，建構AI-FML機器學習工具並導入全球中小學相關教育學習場域應用。

****

透過Patch Learning的概念我們將此學習模式運用在圍棋資料預測模型上。如圖五(a)(b)(c)是基於Patch Learning應用在圍棋資料，首先我們會先將所有訓練資料訓練出一個Global Model，接著我們透過找出學習不佳的區域來加強區段間的訓練。圖五(a)是1-Patch的學習模型，此模型僅有一個Patch專門來預測第一手到第四十手。其餘的手數交由Global Model進行預測。圖五(b)是2-Patch的學習模型，此模型經由兩個Patch進行訓練。在此模型中我們將前四十手切成兩等份，分別為第一手到第二十手，以及第二十一手到第四十手。其餘的手數交由Global Model進行預測。圖五(c)是3-Patch的學習模型，Patch1與2分別為第一手到第二十手，以及第二十一手到第四十手。Patch3為第四十一手到第九十手。其餘的手數交由Global Model進行預測。

|  |
| --- |
| (a) |
| (b) |
| (c) |

1. Patch Learning模型架構 (a) 1-Patch (b) 2-Patch (c) 3-patch

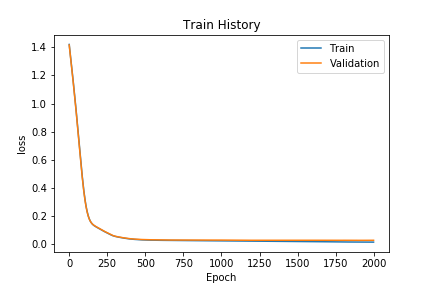
# Experimental results

## 實驗六(ANFIS)

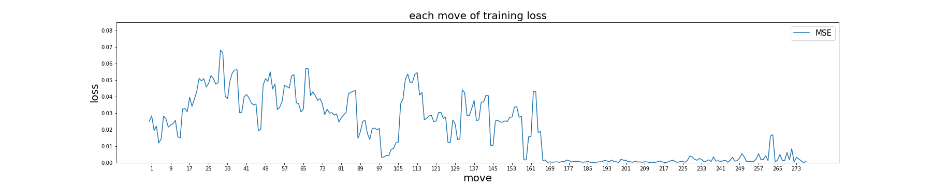
實驗六是使用ANFIS方法來進行圍棋勝率推論以及機器學習。總共有六個輸入及一個輸出，每個輸入變數有六十四個模糊集合。在此實驗中我們採用高斯作為隸屬函數，並隨機產生六十四條模糊規則。機器學習部分們使用Adam優化器並採用梯度下降法以0.001的學習速率來更新所有參數。表四為實驗六經過兩千代學習後的損失比較，我們可以發現實驗六跟實驗五DNN表現差不多。此方法不僅能夠減少學習的參數量，也能大幅降低學習時間。此外透過指定的規則數量，我們的模型可以自動去自適應所有規則所調控的參數達到學習的效果。圖10為實驗六機器學習兩千代的損失曲線，經過兩千代學習後訓練集的MSE為0.0158，驗證集的MSE為0.0286。此外我們將所有訓練資料依據手數個別計算每一手的MSE，如圖11所示橫軸為所有棋局的手數。縱軸為每一手的平均損失，我們以MSE作為模型好壞的評估標準。我們可以從圖11觀察出約前170手的MSE起伏比較大，且我們也能從中也觀察到某幾手區間預測結果較差。從每一盤棋局我們也能觀察出棋局一開始雙方勝負不定因此勝負起起伏伏，直到棋局尾聲勝負才會比較明朗。

1. 實驗六評估指標分析

|  |  |  |  |
| --- | --- | --- | --- |
|  | 訓練集 | 驗證集 | 測試集 |
| MAE | 0.0683 | 0.0876 | 0.1197 |
| MSE | 0.0158 | 0.0286 | 0.0438 |
| RMSE | 0.1258 | 0.1692 | 0.2093 |



1. 實驗六機器學習兩千代的損失曲線



1. 實驗六訓練集每一手損失(MSE)

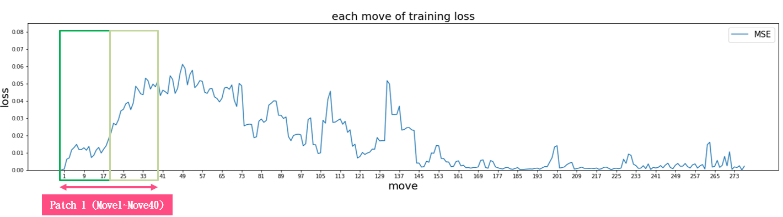
圖12為實驗六迴歸分析。由圖12(a)我們可以看出第39盤棋局經過ANFIS機器學習後預測曲線有大致擬合到ELF OpenGo的預測趨勢，但某些地方表現不佳。例如前5手以及第56手附近預測突然起伏較為明顯，且與實際的趨勢稍微有落差。圖12(b)是測試集第58盤棋局，我們可以發現前40手的預測過於保守與實際結果有點落差。不過整體的趨勢是跟真實結果相吻合。

|  |
| --- |
| (a) |
| (b) |

1. 實驗六迴歸分析(a) Game39 (b) Game58

## 實驗七(1-Patch)

實驗七使用ANFIS架構進行機器學習訓練模型，並加入Patch Learning學習機制。在此實驗中我們先使用全部訓練資料訓練一個全域模型，接著從每一手損失中觀察哪一個區段的表現較差。我們可以從中發現前40手的MSE較大，因此我們將前40手設為第一個Patch進行1000代的訓練，並設定64條模糊規則。圖13為訓練第1~40手後的損失圖，我們可以發現前二十手的MSE有明顯下降的趨勢，但從第21手開始到第40手MSE明顯的上升。圖14(a)(b)分別為全域模型以及Patch1分別訓練一千代的損失曲線，我們可以發現收斂效果不錯且在前兩百代就明顯的收斂到谷底。表五為實驗七的評估指標，我們可以發現在訓練集、驗證集以及測試集都有稍微地改善但效果沒有明顯的突破。



1. 實驗七訓練集每一手損失(MSE)

|  |  |
| --- | --- |
| (a) | (b) |

1. 實驗七機器學習一千代的損失曲線 (a) Global Model (b) Patch1
2. 實驗七評估指標分析

|  |  |  |  |
| --- | --- | --- | --- |
|  | 訓練集 | 驗證集 | 測試集 |
| MAE | 0.0932 | 0.0842 | 0.1318 |
| MSE | 0.0237 | 0.0202 | 0.0427 |
| RMSE | 0.1541 | 0.1421 | 0.2068 |

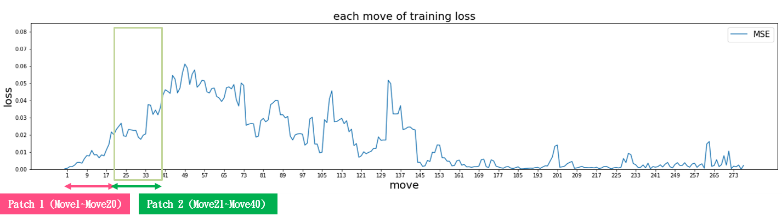
圖15(a)(b)為實驗七迴歸分析。由圖15(a)我們可以發現原本前5手表現不好的區域有稍微的改善但能有進步空間。而圖15(b)第58盤外部測試可以觀察出雖然前20手趨勢有稍微跟ELF OpenGo一樣，但Patch1訓練結果在第20手到40手預測依然不佳。此外我們可以從圍棋對局中發現，通常前20手是棋局剛開始，雙方都還在佈局並規劃策略，所以雙方的局勢通常是勝負不定，且剛開局前幾手勝率幾乎落於0.5左右。這裡我們再思考是否將前40手再分開訓練會比較好。因此實驗八我們將改善Patch1所表現不佳的區段再進一步的分析。

|  |
| --- |
| (a) |
| (b) |

1. 實驗七迴歸分析(a) Game39 (b) Game58

## 實驗八(2-Patch)

實驗八接續實驗七，目標是改善第21手到第40手學習不佳的區域。透過圍棋局勢的分析與觀察，我們將實驗七的1-Patch平均切成兩等份。因此實驗八將會有一個全域的模型以及兩個Patch，分別為第1手到第20手以及第21手到第40手。16為實驗八2-Patch的每一手損失曲線，我們可以發現原本在實驗七第21手到第40手表現不好的區域經過2-Patch的學習後有明顯的下降。圖17(a)(b)(c)分別為全域模型、Patch1與Patch2經過六十四條模糊規則分別訓練一千代的損失曲線。圖17(a)與(b)大約學習兩百代就逐漸收斂，而Patch2的模型，如圖17(c)學習曲線相對的比較不同，直到八百代才逐漸收斂的趨勢。表六為實驗八的評估指標，我們可以與實驗七做比較。實驗八在測試集表現又比實驗七略微改善，以MSE來觀察約改進了0.002。



1. 實驗八訓練集每一手損失(MSE)

|  |  |
| --- | --- |
| (a) | (b) |
| (c) | |

1. 實驗八機器學習一千代的損失曲線   
   (a) Global Model (b) Patch1 (c) Patch2
2. 實驗八評估指標分析

|  |  |  |  |
| --- | --- | --- | --- |
|  | 訓練集 | 驗證集 | 測試集 |
| MAE | 0.0859 | 0.0868 | 0.1267 |
| MSE | 0.0216 | 0.0215 | 0.0407 |
| RMSE | 0.1470 | 0.1469 | 0.2017 |

圖18(a)(b)為實驗八迴歸分析。首先圖18(a)在第20手到第40手確實有明顯的改善，原先表現不好的地方最後都有擬合到真實的局勢。此外在測試集的第58盤在前40手也有明顯的進步，圖18(b)在第20手到第40手的趨勢也逐漸地逼近於ELF OpenGo所預測的勝率。從實驗八我們可以得出一個結論，透過兩個Patch學習前40手的的圍棋資料我們可以有效的改善原先在全域模型中表現不佳的地方因而降低整體的MSE。最後我們再從圖16觀察其他手的損失曲線可以發現41手到第90手是目前全部當中MSE最高的區域，因此我們下一個實驗將會來學習第三個Patch查看是否將會改善預測結果。

|  |
| --- |
| (a) |
| (b) |

1. 實驗八迴歸分析(a) Game39 (b) Game58

## 實驗九(3-Patch)

實驗九接續實驗八的結果繼續觀察其他表現不佳的區域進而學習第三個Patch。從上一個實驗結果，我們找到第三個表現不佳的Patch落於第41手到第90手。因此我們使用ANFIS進行模型1000代的訓練，並設定64條模糊規則。圖19為經過三個Patch學習後所預測出來的每一手損失曲線，我們可以再次觀察到第三個Patch第41手到第90手經過學習後MSE也有下降趨勢但幅度並不大。其原因是此區間屬於大多數圍棋對局中的競爭較為激烈的地方，雙方戰況膠著因此勝率預測會起伏大，因此較難去擬合真實情況。圖20(a)(b)(c)(d)為三個Patch加上一個全域模型各訓練一千代的損失曲線圖。我們可以發現在此實驗第三個Patch如圖20(d)訓練兩百代後逐漸收斂。表7為實驗九的評估指標，跟實驗六相比我們可以發現測試集的MSE經過3個Patch訓練後降低了0.0109。除此之外與DNN和XGBoost相比加入Patch Learning的機制整體的表現更為穩定。



1. 實驗八訓練集每一手損失(MSE)

|  |  |
| --- | --- |
| (a) | (b) |
| (c) | (d) |

1. 實驗八機器學習一千代的損失曲線   
   (a) Global Model (b) Patch1 (c) Patch2 (d) Patch3
2. 實驗九評估指標分析

|  |  |  |  |
| --- | --- | --- | --- |
|  | 訓練集 | 驗證集 | 測試集 |
| MAE | 0.0766 | 0.0810 | 0.1110 |
| MSE | 0.0185 | 0.0189 | 0.0329 |
| RMSE | 0.1360 | 0.1377 | 0.1814 |

圖21(a)(b)為實驗九迴歸分析。如圖21(a)原先在第56手預測結果不佳的手數，經過第三個Patch學習後整盤的預測勝率與實際的區趨勢大致吻合。圖21(b)為外部測試棋局，我們可以發現尚未學習前在第40手到第90手模型預測結果表現不佳如圖18(b)所示。經過學習後第三個Patch加強這一區域的學習，外部測試第58盤所預測的結果最終也逼近於ELF openGo的趨勢。

|  |
| --- |
| (a) |
| (b) |

1. 實驗九迴歸分析(a) Game39 (b) Game58



我們最近有在研究Master對職業棋士的60盤對局。我們發現其中有些對局(1)黑森林最終的預測結果與真實對局勝負不相同。(2)我們各自統計黑森林對黑棋與白棋每一手預測勝負結果與ELF OpenGo對黑棋與白棋每一手預測勝負結果，有一些對局「兩者(黑森林與ELFOPENGO)預測出來的勝負相反的手數，超過該盤對局總手數的40%。

我們發現

Game 8、15、25、29、32、36

Game 44、47、48、56、57、60

有出現上面兩種情況。

我們想先請您幫忙看Game 15, 25, 29, 32, 36是否有特別不同的地方? 底下的圖，我們有註明的這些手數表示黑森林與ELF OpenGo預測勝率黑棋/白棋正好相反的手數，或是我們覺得有比較奇怪的地方，再請您幫忙就這幾手是否有特別的地方可能造成「我們觀察到的現象」呢?

GAME 15

黑線表示黑棋，紅線表示白棋，虛線表示黑森林的預測，實線表示ELFOPENGO的預測

Master(P)執白 vs XIUZHI(P)(樸廷桓)，白時間勝(黑貼6目半)

 Move16~80

 Move104~128

 Move138~146

 以上這幾手黑森林預測與ELF OpenGo相反

評論:

Game15 佈局階段黑棋的腳步過於保守緩慢，白10～24快速瓦解黑上邊的陣勢。之後黑29～33欲重起爐灶卻又被白34、36凌空壓制，向外發展受阻只得49～55委屈連回，雖也得到些許地盤但收穫有限。由於黑25的防守位置不佳（應再右移兩格才能守好角），被白70點入輕鬆破角後實空明顯不足，難以翻身，即便69、83、95在上邊圈起一塊看似肥美的腹地，但增加的目數無法彌補角上的損失，而且為了保護腹地不被入侵，也讓白在中間獲利不少，此後勝負已無懸念。

Game 15 白16～24處理完之後形勢已經不錯，黑25嚴重問題手，此後幾乎是沒有什麼機會，所以這盤後面好像已經沒有需要補充說明的地方了。

黑25手是嚴重問題手，「黑森林」好像判斷不出來，請問有什麼特別的局面狀況，可能造成黑森林看不出來勝率?

答: 25本意是要圍空，但下在錯誤的位置最後還是被對方破光地盤，接近快浪費了一手棋。不曉得黑森林是否沒有把右上角的手段判斷進去。

Game 25

黑線表示黑棋，紅線表示白棋，虛線表示黑森林的預測，實線表示ELFOPENGO的預測

Master(P)執白 vs XIUZHI(P)(樸廷桓)，白勝半目(黑貼6目半)

 Move234~258 AI Bot預測有動盪，此外最後黑森林是預測黑贏而ELF OpenGo預測白贏

評論:

Game25 白30肩衝是AI一貫的風格，通常跟著走棋不反擊多半被便宜了。白32深入敵營問黑動向，黑選擇強行吃掉，但先前已投入大量兵力，付出的代價不僅僅是47～57低位委屈的爬行渡過，還遭到白58、60一再的咄咄進逼，忍無可忍之下黑61、65斷開白棋求戰，卻正中白計。白66、68完美好調，以攻代守，處理好自身後，再76繼續瞄準黑棋弱點，黑79應下87回防，被白84跑出後，損兵折將，白局面大好。最後的勝負雖然只有半目之差，但其實勝負已定。人類即使在優勢的時候，也會力求每一步下到最好，AI的設定是只要贏半目就好，因此會簡化局面，安全運轉即可。

Game 25 234之後勝負已經確定。但是Move234~258 ELFOpenGo預測白棋勝率高低變化很厲害，這幾手的局面有什麼特別的嗎? 黑森林最後是預測黑贏(與事實相反)，ELFOpenGo仍是預測白贏(與事實相同)，不知是否有什麼比較奇怪的局面，造成黑森林(學生)看不出來，而ELFOpenGo(老師)則看出來了。

答: 234之後雙方地盤已經確定，不會有增加或減少目數的可能性，不太知道黑森林為何判斷錯誤。

GAME 29

黑線表示黑棋，紅線表示白棋，虛線表示黑森林的預測，實線表示ELFOPENGO的預測

Master(P)執白 vs 拼搏(P)(羋昱廷)，白勝半目(黑貼6目半)

 Move272 以後黑森林預測動盪極大勝負不定，但ELF OpenGo預測勝負已定。此外最終結果黑森林預測錯誤。

評論:

Game29 272之後勝負已經確定，可能是因為打劫的緣故，造成黑森林難以判斷。

再請凱馨幫忙補充其它手數是否有特別的地方。

答: 在alphago出來之前，打劫的部分是電腦的短版，打劫牽涉到交換與價值的判斷，要怎麼找劫等等，非常的複雜。這可能是黑森林的弱項，一進入打劫就頭昏眼花。Alphago則沒有這樣的問題。

補充: 黑23貪吃一子被白棋便宜了，白棋先手限制了黑棋右邊的發展，黑應該87反擊讓白棋脫不了身。左上是傳說中三大最困難的定石之一『大雪崩』。白44、46下出前所未見的新手，黑49應該下227，55應該先56打吃，61要下85，這個定石下完，局面也完了。

GAME 32

黑線表示黑棋，紅線表示白棋，虛線表示黑森林的預測，實線表示ELFOPENGO的預測

星宿老仙(古力)執黑 vs Master，白中盤勝(黑貼6目半)

 整盤黑森林所預測的勝率雙方膠著直到Move132才逐漸明朗，但在Move24左右ELF OpenGo勝負有開始逐漸拉開。

評論:

Game32 佈局行至51手止，形成白棋實利與黑棋外勢對抗的局面，不過跟電腦走成這樣的格局，人類要靠大模樣作戰取得優勢，應該是頗有難度，畢竟對中央的掌控能力，還是電腦略勝一籌。白52～58打入，很輕鬆的瓜分掉了黑棋的地盤，黑棋59反擊，但是61吃掉白58ㄧ子是錯誤的決策，應該62斷開白棋，繼續貫徹控制中腹的計畫，如此還可一戰。實戰至70止，右半邊黑棋的大本營宛如摩西分紅海，被隔開的兩塊地不但目數很少，外勢的潛力也消失殆盡。 此後已無勝機。

請凱馨幫忙看一下，為何M132與M24是否有特別的地方? 造成黑森林直到M132才看出勝負。

答: 24對人類來說是很正常的一手，看不出有什麼特別，但有可能ELF覺得23與24的交換是虧損的。132還沒下之前，黑棋有機會下到這手把白棋吃掉，但是白132是先手，黑棋必須跟著理一手，之後的進程白棋開始簡化局面，把自己的權利一步一步交換掉，黑森林可能才知道黑棋輸了。

GAME 36

黑線表示黑棋，紅線表示白棋，虛線表示黑森林的預測，實線表示ELFOPENGO的預測

Master執黑 vs 印城之霸(辜梓豪)，黑中盤勝(黑貼6目半)

 Move96~158黑森林預測與ELF OpenGo相反

 直到Move166左右黑森林預測勝負才逐漸明朗，而ELF OpenGo早在Move16手就逐漸定勝負。

評論:

Game36 黑17是令人猝不及防的一手，雖不會致人於死，但讓白棋相當難受，白棋不堪委屈連接，只好18位反擊，27止黑棋戰果豐碩。白34～38雖然厚實但是腳步緩慢，黑41碰的時候白棋理應利用外勢於43與黑一戰，結果42退讓還落後手被黑45佔得先機，右邊一排牆壁淪為背景裝飾。到97手止，雙方地盤大致已劃清界線，白棋因實地不足開始對黑中央進行攻擊，但過程中又被黑脫先搶到115的大官；126做最後努力的拼搏，黑一邊做活一邊搜刮白棋的地盤，白終究以失敗收場。

再請凱馨幫忙補充M16及M166是否有特別的地方，造成黑森林要到M166左右才能看出勝負，而ELF OpenGo則在M16就知道勝負了。

因這幾盤我們沒有放入訓練資料中，所以對於黑森林(學生)與ELF OpenGo(老師)預測勝率有差別的手數，想要知道是否局面有特別的地方，造成學生無法看出來? 但是老師早早就看出勝率?

再麻煩凱馨以職業棋士角度來看的意見與想法，幫忙我們補充這些手數附近是否局面比較奇怪? 例如: 有打劫或是死活等等「初學者(=黑森林)」比較難看出勝負，但是老師(=ELF OpenGo = 職業棋士)早早就知道勝負了。

答: M16對人類是正常的一手，此前沒有遇過實戰黑17的著法，因此白16應該是壞棋，才一開始沒幾手棋局瞬間走了樣。白166雖然把黑棋吃了，但為了吃被黑棋各種利用，到處被便宜，目數已經不夠。

# Conclusions and Discussions

The experimental results show that.

##### Acknowledgment

The

##### References

1. J.-S. R. Jang, " ANFIS: Adaptive-Network-based Fuzzy Inference Systems," IEEE Trans. on Systems, Man, and Cybernetics, vol. 23, pp. 665-685, May 1993.
2. P. Jagtap and G. N. Pillai, “Comparison of extreme-ANFIS and ANFIS networks for regression problems,” 2014 IEEE International Advance Computing Conference (IACC), Gurgaon, 2014, pp. 1190-1194, doi: 10.1109/IAdCC.2014.6779496.
3. T. G. Ling, M. F. Rahmat and A. R. Husain, “ANFIS modeling and Direct ANFIS Inverse control of an Electro-Hydraulic Actuator system,” 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA), Melbourne, VIC, 2013, pp. 370-375, doi: 10.1109/ICIEA.2013.6566397.