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 (PL) mechanism for robotic game of Go applications. The proposed AI-FML agent contains three kinds of intelligence, including a perception intelligence, a cognition intelligence, and a computational intelligence, for the robotic application. Additionally, we embed the PL mechanism into the AI-FML agent. The method for running PL involves three steps. It first trains an initial global model, then trains a patch model for each identified patch, and finally updates the global model using the training data that do not fall into any patch. This paper adopts the Google DeepMind Master 60 games to be the training data and testing data set. The experimental results show the AI-FML agent with the patch learning mechanism can improve the performance of regression for robotic game of Go applications.

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

# Introduction

Nowadays, machine learning has been used in multiple real-world fields and industries, such as medical diagnosis, image processing, regression etc. However, training a high performance machine learning model is usually an iterative process which relies on experience and trial-and-error. Sometimes, we need to take some remedies to enhance its performance when it is not satisfactory [1]. Some strategies are implemented to enhance the performance on 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* [1, 2]. In this paper, we propose an AI-FML agent with the patch learning (PL) mechanism to improve the performance of the machine learning [1, 2]. The idea of PL introduced by 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]. According to [1], PL consists of three steps described as follows: (1) Train an initial global model using all training data first. (2) Identify patches from the initial global model and train a patch model for each patch. (3) Update the global model using training data [1]. Wu and Jerry [1] defined 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. However, generally identifying the patch locations is a very challenging task, and not every problem can be easily visualized. The PL connects multiple simple models both in parallel and in series to improve the learning performance. Mendel [2] introduced how PL can be performed using rule-based fuzzy systems, because it is easy to initialize patch candidates in a fuzzy system.

Regression, ensemble methods, and deep learning are important machine learning methods for data scientists [9]. An adaptive network-based fuzzy inference system (ANFIS) was proposed by Jang [12] in 1993. The ANFIS integrates both machine learning model and fuzzy logic principles and it has a 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 [12]. In our previous paper [3], we used AlphaGo Master sixty games as the experimental dataset to make the win rates predicted by the DDF AI bot closer to those predicted by the ELF Open Go AI bot based on FML-based genetic learning (GFML), XGBoost learning, and a seven-layered deep fuzzy neural network (DFNN) learning. In this paper, we further combine FML, PL, and adaptive network-based fuzzy inference system (ANFIS) with the deep learning to train a model to predict the win rates of the AlphaGo Master sixty games. The experimental results show that introducing PL mechanism has a better performance than the methods presented in our previous paper [3].

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

## Patch Learning Mechanism

During the design stage of the traditional ANFIS, the 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 [1, 12]. Based on the concept of PL in [1], this paper proposes an AI-FML agent with patch learning mechanism and it begins with a globally designed rule-based fuzzy system, but then locates the patches which have made the most contribution to the performance metrics. Fig.1 shows the structure of patch learning for predicting the win rate of Master sixty Go games which described as follows: (1) We use all of the training data to train the global model . (2) We identify that the input regions from move 1 (M1) to move P1 (MP1) give rise to large learning errors so that we use the training data which fall into this region to train the patch model 1 (PM1) to reduce the overall learning error. Then, we use the input regions from “move P1+1 (MP1+1) to move P2 (MP2)”, …, and “move N (MN) to move PN (MPN)” to train the local patch model 1 (PM1), patch model 2 (PM2), …, and patch model L (PML), respectively. (3) Finally, the global rule-based fuzzy system is updated, using the remaining training data that have not been used by patches 1, 2, .., and L. Fig. 1 takes Game 2 as an example to show that L = 3, MP1 = 20, MP2 = 40, and MP2 = 90.



1. Structure of patch learning for predicting the gane of Go Data set.

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

The ANFIS-based patch learning mechanism is embedded into the AI-FML agent and we apply it to predict the win rate of game of Go data set. Additionally, ANFIS algorithm is adopted to train the patch models to improve the performance metrics [1, 12]. The ANFIS is one of artificial neural network models that is based on Takagi–Sugeno fuzzy inference system, and developed in the early 1990s [12, 13]. In this paper, we utilize PyTorch to implement the framework of the ANFIS-based patch learning model. The structure of ANFIS contains two parts, including premise part and consequence part and its architecture is composed of five layers [12] described as follows: (1) First Layer/Fuzzification Layer: It takes the input values to determine the membership functions belonging to them. The membership degrees of each function are computed by using the premise parameter set. (2) Second Layer/Rule Layer: It is responsible of generating the firing strengths for the rules by multiplying the incoming signals and sends the product out. (3) Third Layer/Normalizes Layer: It normalizes the computed firing strengths by diving each value for the total firing strength, that is, it sends the normalized firing strengths. (4) Fourth Layer/Inference Layer: It multiples the normalized values from the third layer with the consequent parameters and sends the results to the fifth layer. (5) Fifth Layer/Defuzzified Layer: It computes the overall output as summation of all incoming signals and to generate the final output.

# ANFIS-based Patch Learning for Go Game Data Set

## Introduction to Data Set from IEEE WCCI 2020

In this paper, we adopt 60 online games Master in Dec. 2016 and in Jan. to be the experimental data and they were designated to be the competition data held in IEEE CIS flagship conferences [3]. The participates can choose any 40 Games from 60 Games as the training data and the remaining 20 Games as the testing data. The participates construct the knowledge base and the rule base of the fuzzy inference system. Table 1 shows the information of move 1 to move 20 of Game 1. Each game includes the information predicted by Darkforest AI Bot and by EFL OpenGo AI Bot. MoveNo is the move number but 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. 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 paper.

1. Move 1 to Move 20 of Game 1.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Move  No | DBSN | DWSN | DBWR | DWWR | DBTMR | DWTMR | EBWR | EWWR |
| 1 | 3863 | 2274 | 0.52 | 0.48 | 0 | 1 | 0.49 | 0.50 |
| 3 | 9283 | 7866 | 0.51 | 0.48 | 0.5 | 1 | 0.45 | 0.54 |
| 5 | 11395 | 6798 | 0.51 | 0.47 | 0.66 | 1 | 0.45 | 0.57 |
| 7 | 4499 | 10703 | 0.51 | 0.46 | 0.75 | 1 | 0.49 | 0.52 |
| 9 | 7388 | 20017 | 0.52 | 0.46 | 0.8 | 1 | 0.47 | 0.53 |
| 11 | 20098 | 9693 | 0.53 | 0.46 | 0.83 | 1 | 0.47 | 0.52 |
| 13 | 20017 | 14595 | 0.53 | 0.47 | 0.85 | 1 | 0.48 | 0.49 |
| 15 | 6786 | 4892 | 0.52 | 0.45 | 0.87 | 0.875 | 0.48 | 0.53 |
| 17 | 20017 | 6432 | 0.53 | 0.46 | 0.88 | 0.7778 | 0.48 | 0.53 |
| 19 | 5267 | 13267 | 0.51 | 0.48 | 0.9 | 0.8 | 0.48 | 0.50 |
| Note | | | | | | | | |
| Each row includes eight values, where *DBSN*, *DWSN*, *DBWR*, *DWWR*, *DBTMR*, and *DWTMR* were the outputs from Darkforest, and *EBWR* and *EWWR* were the outputs from ELF OpenGo.  (1) *DBSN* and *DWSN* denote the number of simulations for Black and White, respectively.  (2) *DBWR* and *DWWR* are the win rate of Black and White, respectively.  (3) *DBTMR* and *DWTMR* are the top-move rate of Black and White, respectively.  (4) EBWR and EWWR are the win rate of Black and White predicted by ELF OpenGo. | | | | | | | | |

## Pre-processing Partition Function for Game Data Set

Jang [12] 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 with uncertainty and imprecision …” in 1993.

There are different kinds of partition functions for partitioning the input domains, for example, a *crisp partition function* or a *Type-1 partition function* [2]. 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 [2]. 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 overlaps with their nearest neighbors, and it is not a mathematical partition. Fig. 2 shows the adopted type-1 partition with 64 Gaussian function for the game of Go data pre-processing in this paper.



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

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

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

In addition, 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 and ELF openGo for the three games.

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 composed of 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 [12]. Fig. 4 shows the structure of ANFIS for AI-FML agent on robotic application which is constructed by following the steps as follows: (1) retrieve the input values for fuzzy variables with membership functions on the premise part to compute the membership degree of each linguistic label, (2) combine the membership degrees on the premise part to get the weight of each rule, (3) generate the qualified consequents of each depending on the weights, and (4) aggregate the qualified consequents to produce the final output [12]. Human input the data and AI-FML agent output the defuzzied value to the robots, such as Kebbi Air, Palro or Zenbo, to achieve the goal of human and robot colearing in the classroom [3]. Table III shows the learning ANFIS model using a backpropagation algorithm.



1. Structure of ANFIS for AI-FML agent

In this paper, we use 64 type-1 Gaussian partition functions to partition the six input fuzzy variables, including *DBSN*, *DWSN*, *DBWR*, *DWWR*, *DBTMR*, and *DWTMR*. For example, if *x* is the input value of *DBSN* for the PL-based AI-FML agent, then there are 64 rules as follows:

|  |
| --- |
| Rule 1: If *x* is *Partition 1*, Then *y*=*y*1(*x*) |
| Rule 2: If *x* is *Partition 2*, Then *y*=*y*2(*x*) |
|  |
| Rule 64: If *x* is *Partition 64*, Then *y*=*y*64(*x*) |

where *y*1(*x*), *y*2(*x*), …, and *y*64(*x*) are different functions of *x*. In Partition P(1|*x*), if only Rule 1 is fired, and hence the fuzzy system output is *y*=*y*1(*x*). In addition, in Partition P(2|*x*), both Rule 1 and Rule 2 are fired, and hence the fuzzy system output is the weighted average of *y*1(*x*) and *y*2(*x*) with the membership degree of *Partition 1* and *Partition 2*, respectively. Fig. 5 shows the type-1 partition with 64 Gaussian function for the game of Go in the initial state.

1. ANFIS model using backpropagation algorithm.

|  |
| --- |
| (a) |
| **Input:**   1. *TDS*: Training data set with *NTDS* examples. 2. **x** = [*x*1, *x*2, …, *x*n] where *x*1, *x*2, …, and *x*n denote *BSN*, *WSN*, *BWR*, *WWR*, *BTMR*, and *WTMR*, respectively.   **Output:**  = [*BWR*1, *BWR*2, …, *BWRNTDS*]: Inferred win rate of Black  **Method:**  **Step 1:** Initialize ANFIS neural network  **Step 1.1:** Randomize a constant formean () and variance () of Gaussian membership functions of each input feature in **x** usinganormal distribution.  **Step 1.2:** Randomize a constant for the output *fi* of the *i*th fuzzy rule using normal distribution.  where *i* = 1, …, and *NFR*, *n* = 1, …, and *NINF*, *NINF* is number of input features, and *NFR* is number of fuzzy rules.  **Step 2:** Input the first example of the training data set  **Step 2.1:** For each input feature *x*1, *x*2, …, *x*n in **x**  **Step 2.1.1:** Calculate the membership degree of each fuzzy rule by  where *i*=1, 2, …, *NFR* and represents the *i*th Gaussian fuzzy set of the *n*th input feature *xn* with mean and variance .  **Step 2.1.2:** Perform multivariable T-norm operator to calculate the firing strength of each fuzzy rule by  =  **Step 2.1.3:** Normalize the firing strengths for each fuzzy rule  **Step 2.1.4:** Calculate the outputfor each fuzzy rule by  *fi*  **Step 2.1.5:** Compute the overall output *BWR*1 by  **Step 3:** Repeat **Step** 2 for the remaining examples 2, 3, .., and *NTDS* in *TDS* to acquire *BWR*2, *BWR*3, …, and *BWRNTDS*, respectively.  **Step 4:** End |
| (b) |
| **Input:**   1. **BWRDO =** [*BWR*1, *BWR*2, …, *BWRNTDS*]: Label value of the win rate of Black /\**WWRDO* = 1-*BWRDO*\*/ 2. = [*BWR*1, *BWR*2, …, *BWRNTDS*]: Inferred win rate of Black   **Output:**  *Learned Model*  **Method:**  **Step 1:** For *k1* to epochs  **Step 1.1:** Calculate loss value of the *k*th epoch by  *Lossk* =  **Step 1.2:** Implement the backpropagation algorithm with an Adam optimization to update the parameters of our learned model, including , , and *fi* listed in Table III (a).  **Step 2:** Store the learned model  **Step 3:** End |



1. Initial Gaussian functions of the PL-based AI-FML agent for the game

## PL-based AI-FML Agent for Game of Go Dataset

The PL-based AI-FML agent can use 1-Patch, 2-Patch and 3-Patch for game of Go Dataset. Fig. 13 shows the structure of PL-based AI-FML agent with 1-Patch, 2-Patch and 3-Patch for Go Dataset and future student learning applications. In Fig. 13, we first construct a global PL-based AI-FML agent by training all the dataset. Then construct the 1-Patch Learning model by predicting the move 1 to move 40, and the other moves are trained by the global model, shown as Fig. 13(a). Fig. 13 (b) shows the 2-Patches learning model by the first Patch based on move 1 to move 20, the second Patch based on move 21 to move 40, and Global model based on the other moves. Fig. 13 (c) shows the 3-Patches learning based on (1) Patch 1: move 1 to move 20, (2) Patch 2: move 21 to move 40, Patch 3: move 41 to move 90, and (4) Global: move 91 to end move. Finally, Fig. 13 (d) shows the PL-based AI-FML agent for future student learning applications.

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

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

# Experimental results

In this paper, we use 64 type-1 Gaussian partition functions to partiting the six input fuzzy variables, including DBSN, DWSN, DBWR, DWWR, DBTMR, and DWTMR. For example, if *x* is the input value of DBSN for the PL-based AI-FML agent, then there are 64 rules as follows:

Rule 1: If *x* is *Partition 1*, Then *y*=*y*1(*x*)

Rule 2: If *x* is *Partition 2*, Then *y*=*y*2(*x*)

…

Rule 64: If *x* is *Partition 64*, Then *y*=*y*64(*x*)

where *y*1(*x*), *y*2(*x*), … and *y*64(*x*) are different functions of *x*. In Partition P(1|*x*), only Rule 1 is fired, and hence the fuzzy system output is *y*=*y*1(*x*). In addition, in Partition P(2|*x*), both Rule 1 and Rule 2 are fired, and hence the fuzzy system output is the weighted average of *y*1(*x*) and *y*2(*x*) with the membership degree of *Partition 1* and *Partition 2*, respectively. Fig. 6 shows the type-1 partition with 64 Gaussian function for the game of Go data pre-processing.



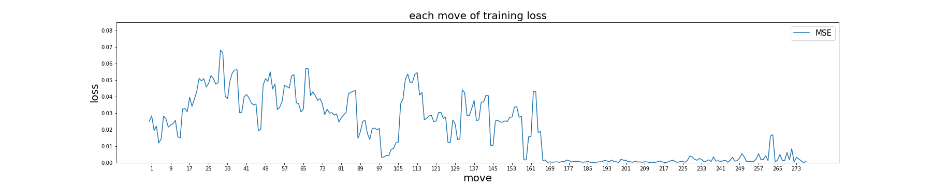
1. *Type-1 partition functions with 64 Gaussian functions for Go.*

## Experiment 1: ANFIS Global Learning for AI-FML Agent

In Experiment 1, we adopt the ANFIS with Adam optimizer and Gradient Decent learning algorithm with learning rate 0.001 to update all the parameters. Table IV shows the loss with MAE, MSE, and RMSE criteria after 2,000 epoches. This methods can reduce both learning parameters and learning time. Fig. 7 shows the average loss based on MSE for each move.

1. Loss Evaluation based on MAE, MSE, and RMSE

|  |  |  |  |
| --- | --- | --- | --- |
|  | Training set | Validate set | Testing set |
| MAE | 0.0683 | 0.0876 | 0.1197 |
| MSE | 0.0158 | 0.0286 | 0.0438 |
| RMSE | 0.1258 | 0.1692 | 0.2093 |



1. average loss based on MSE for each move (EXP. 1)

Fig. 8 shows the regression analysis for Game 39 and Game 58 in Experiment 1.

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

1. Regression analysis for EXP. 1: (a) Game39 and (b) Game 58

## Experiment 2: PL-based AI-FML Agent with 3-Patches

In Experiment 2, we adopt the PL-based AI-FML agent with 3-Patches mechanim. Fig. 9 shows the loss curve for each move between moves 1 to 20, moves 21 to 40, and moves 41 to 90, with 3-Patches learning mechanis after 1,000 epoches. We observe that the loss curve based MSE gets better results between move 41 to move 90. Fig. 10 (a), (b), (c) and (d) shows the loss curves for Global, 1-Patch, 2-Patches, and 3-Patches, respectively, after 1,000 epoches.



1. MSE curve for each move with 3-Patches after 1,000 epoches

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

1. loss curves for Global, 1-Patch, 2-Patches, and 3-Patches, respectively, after 1,000 epoches

# Conclusions and Discussions

The AI-FML agent with patch learning mechanism for robotic game of Go applications is presented in this paper. Various patch learning (PL) models including, PL with global model, 1-patch, 2-paches, or 3-patches model for each identified patch, are performed in the experiments. 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.

##### Acknowledgment

The

##### References

1. D. Wu and J. M. Mendel, “Patch learning,” *IEEE Transactions on Fuzzy Systems*, Jul. 2019. (DOI: 10.1109/TFUZZ.2019.2930022)
2. J. M. Mendel, “Adaptive variable-structure basis function expansions: Candidates for machine learning,” *Information Science*, vol. 496, pp. 124-149, 2019.
3. C. S. Lee, Y. L. Tsai, M. H. Wang, W. K. Kuan, Z. H. Ciou, and N. Kubota, “AI-FML agent for robotic game of Go and AIoT real-world co-learning applications,” *2020 World Congress on Computational Intelligence* (*IEEE WCCI 2020*), Glasgow, Scotland, UK, Jul. 19-24, 2020. (Accepted)
4. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, pp. 484–489, Jan. 2016.
5. C. S. Lee, M. H. Wang, L. W. Ko, Y. Hsiu Lee, H. Ohashi, N. Kubota, Y. Nojima, and S. F. Su, “Human intelligence meets smart machine: a special event at the IEEE International Conference on Systems, Man, and Cybernetics 2018,” *IEEE Systems, Man, and Cybernetics Magazine*, vol. 6, no. 1, pp. 23-31, Jan. 2020.
6. C. S. Lee, M. H. Wang, Y. L. Tsai, L. W. Ko, B. Y. Tsai, P. H. Hung, L. A. Lin, and N. Kubota, “Intelligent agent for real-world applications on robotic edutainment and humanized co-learning,” *Journal of Ambient Intelligence and Humanized Computing*, 2019. (DOI: 10.1007/s12652-019-01454-4).
7. C. S. Lee, M. H. Wang, L. W. Ko, N. Kubota, L. A. Lin, S. Kitaoka, Y. T Wang, and S. F. Su, “Human and smart machine co-learning: brain-computer interaction at the 2017 IEEE International Conference on Systems, Man, and Cybernetics,” *IEEE Systems, Man, and Cybernetics Magazine*, vol. 4, no. 2, pp. 6-13, Apr. 2018.
8. C. S. Lee, M. H. Wang, S. C. Yang, P. H. Hung, S. W. Lin, N. Shuo, N. Kubota, C. H. Chou, P. C. Chou, and C. H. Kao, “FML-based dynamic assessment agent for human-machine cooperative system on game of Go,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 25, no. 5, pp. 677-705, 2017.
9. J. Castanon, “10 machine learning methods that every data scientist should know,” May 2019, [Online] Available: <https://towardsdatascience.com/10-machine-learning-methods-that-every-data-scientist-should-know-3cc96e0eeee9>.
10. J. S. R. Jang, “ANFIS: Adaptive-Network-based Fuzzy Inference Systems,” *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 23, pp. 665-685, May 1993.
11. Wikipedia, “Adaptive neuro fuzzy inference system,” 2020 [Online] Available: <https://en.wikipedia.org/wiki/Adaptive_neuro_fuzzy_inference_system>.
12. 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.
13. 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.