Macromanagement and Strategy Classification in Real-Time Strategy Games

1st Fuheng Dai
College of Intelligence Science and Technology
National University of Defense Technology
Changsha, China
dfh90@qq.com

2nd Jianxing Gong
College of Intelligence Science and Technology
National University of Defense Technology
Changsha, China
fj_gjx@qq.com

3rd Jian Huang
College of Intelligence Science and Technology
National University of Defense Technology
Changsha, China
nudtjhuang@hotmail.com

4th Jianguo Hao
College of Intelligence Science and Technology
National University of Defense Technology
Changsha, China
504343990@qq.com

***Abstract*—We have done some research on the macromanagement modeling and strategy classification in the RTS(real-time strategy) game StarCraft. Similar to the real military battlefield, simulation in the virtual environment could do favor to modeling, realize and predict the opponent strategy which could lead us to the counter strategy. Firstly we collect several typical replays played by professional player, then we do some preprocess and extract the key features to readable data, finally, we use HMM-KNN to do the strategy classification.**

***Index Terms*—RTS, strategy, HMM-KNN, sequence alignment**

I. Introduction

Nowadays, intelligence research in video game is attracting more and more attention of domain experts and the game environment has become an appropriate testbed for the artificial intelligence.

StarCraft is a very popular real-time strategy game developed by Blizzard. In the game, players can choose three races from Protos, Terran and Zerg. What a player should do is to collect resources, build arms, choose strategies and defeat opponents. RTS game tasks can be categorized into strategy, tactics and reactive control, in which strategy means the plan to achieve the overall goal such as "rush" and "passive defend" , tactics usually focus on combat like troop dispatch and maneuver for short-term goals, while reactive control deals with specific units and buildings [1]. These tasks can in turn be grouped into layers [2], which is shown in Fig.1.

This paper aims at the classification of player strategies which corresponding to the high-level macromanagement. The paper is structured as follows. Firstly, we do some overview of the related research in the strategy modeling and prediction. Then we present the classification model and the experiments were carried out. Finally, we draw a conclusion and the further work is discussed.

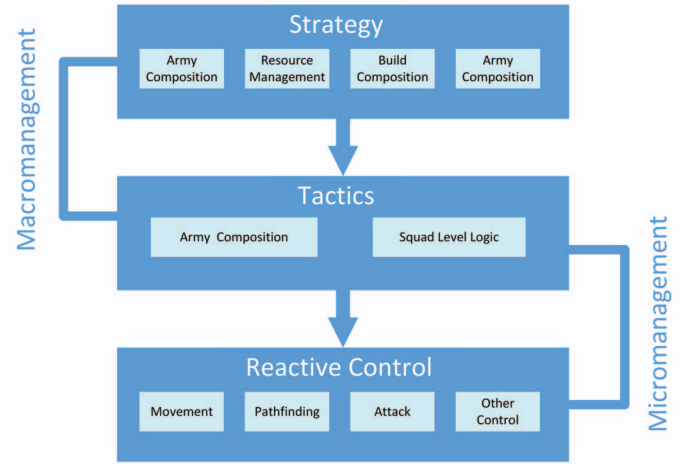

Fig. 1. RTS game layers and tasks.

II. Related Works

Several works have been done in the domain of RTS AI research. D. Churchill presents a abstractions and heuristics to speed up the search of build order choice [3]. H.-C. Cho uses decision trees to predict the strategy and build order by analyzing the replays [4]. Based on HMM(Hidden Markov Model), E. Dereszynski generates a model to automatically learn the opponent behavior, and the detect of the unusual game is also involved [5]. G. Synnaeve presents a Bayesian model both in opening prediction and plan recognition in StarCraft [6], [7]. Owing to the existence of "Fog of War" , opponent information turns incomplete, and to verify the influence, H. Cho chooses RF(Random Forest) and MLP(Muti-Layer Perception) algorithm to fulfill the validation [8]. Furthermore G. Synnaeve develops a Starcraft-Defogger using encoder-decoder neural networks [9]. N. Justesen propose a deep learning method to learn macromanagement in the game [10].

978-1-7281-4091-9/19/$31.00 ©2019 IEEE

P.-A. Andersen develops Deep RTS as an environment for deep reforcement learning [11].

III. Approach

In order to classify the game strategy, we firstly collected several labeled, typical, professional and representational replays. Then we did some preprocess and extract the key features for further use. Lastly we proposed the feasible method of classification and tested it's performance.

A. Data Collection

Replays in StarCraft are the game logs which contain all movements, attacks, chats and other informations from players. Contrast to the real time environment, formulating the simulation in the offline state is much more convenient and has less resource consumption.

On the one hand, we collect and structure our dataset from professional website such as GosuGamers [1] and TeamLiquid [2] by data mining, on the other hand, we choose appropriate replays from the available datasets [12]–[14]. We have more interests in the games played by the Protoss race whose strategies are easy to distinguish. Finally, we collect 240 Protoss replays from professional and high-ranked amateur matches. The categories and amounts are as follows.

TABLE I
Protoss Replays Category.

Strategy	Amount	Strategy	Amount
Zealot Rush	43	Two Gate Observer	21
Gateway Expand	27	Two Gate DT	46
Nexus	21	Gate Core Expand	19
Arbiter	18	Two Base Carrier	25
Forge Expand	20	Total	240

B. Feature Extraction

Replays from Starcraft are stored in a binary format, the original and unprocessed data are unreadable to people. We use the BWAPI[3] to convert the replays to readable game logs. Some preprocess has been done to increase the feature dimensions [15].

The logs contain various features such as APM, resource states, units types, buildings types, upgrades, building orders, commands and hotkeys. As for our research in the macromanagement and high-level strategies, the key features are consist of buildings, units and tech tree.

Learning the expert domain knowledge and analyzing the typical replays, we extracted 31 key features [16] which are shown as follows.

[1]https://www.gosugamers.net/
[2]https://tl.net/
[3]http://bwapi.github.io/

TABLE II
Protoss Replays Key Features.

Id	Feature	Id	Feature
A	Probe	Q	Observatory
B	Pylon	R	Stargate
C	Nexus	S	Scout
D	Gateway	T	Arbiter Tribumal
E	Zealot	U	Arbiter
F	Cybernetics Core	V	Shield Battery
G	Dragoon	W	Dark Templar
H	Assimilator	X	Shuttle
I	Singularity Charge	Y	Reaver
J	Forge	Z	Observer
K	Photon Cannon	a	Corsair
L	High Templar	b	Fleet Beacon
M	Citadel of Adun	c	Carrier
N	Templar Archives	d	Leg Enhancements
O	Robotics Facility	e	Psionic Storm
P	Robotics Support Bay		

C. HMM-based Classification

HMM is a statistical model, which is used to describe a Markov process with hidden unknown parameters, and the basic structure is shown in Fig. 2. We firstly tried HMM to do the strategies modeling and classification, the results shows that it is feasible in most circumstances, but insensitive to confusing situation.

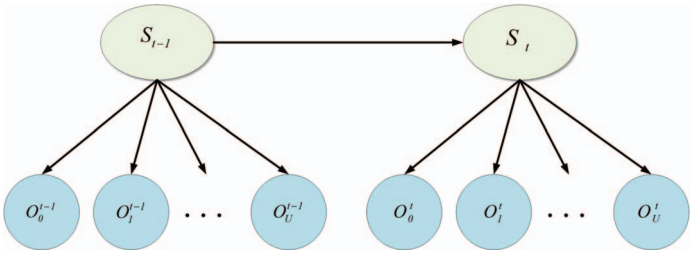

Fig. 2. HMM model.

To formulate the HMM model, we should definite the model parameters $\lambda = (M, N, A, B, \pi)$:

- M represent hidden state number.
- N represent observation number .
- $A = P(S_t|S_{t-1})$ represent state transition probability.
- $B = P(O^t|S_t)$ represent observation probability.
- $\pi = P(S_0)$ represent initial state probability.

We divide the replays into several 30 sec intervals, in each interval we have generated the discrete observation matrix $O^t = (O_o^t, O_1^t, O_2^t, ...O_U^t)$, where U equals the total extracted features. O_u^t is 1 if the feature u is observed during interval t and 0 vice versa.

The hidden states take values in $\{1, ..., M\}$,we define $\partial_{m'}^m$ as the transition probability from state m' to state m , then we have the state transition probability:

$$P(S_t|S_{t-1} = m) \sim Multinomial(\partial_1^m, \partial_2^m, ..., \partial_M^m) \quad (1)$$

Due to the conditionally independence of each features, the joint observation probability equals the product of individual observation probability:

$$P(O^t|S_t) = \prod_u P(O_u^t|S_t) \tag{2}$$

We define β_m as the probability if the initial state is in state m, and initial state can be treated as irregular M-sided dice. Thus we have the initial state probability:

$$P(S_0) \sim Multinomial(\beta_1, \beta_2, ..., \beta_M) \tag{3}$$

As for the choice of hidden state number M, we have done five-fold cross validation. Considering the accuracy and time complexity, we take 9 as the hidden state number. We use Baum-Welch algorithm to learn the HMM model parameters, corresponding to the predefined strategies, we have trained 9 HMM models.

Given normalized observation matrix, we use Forward algorithm to calculate the conditional probability $P(O|\lambda)$, thus we have the strategy classification:

$$Strategy_i = \arg\max(LogP(O|\lambda_i)) \tag{4}$$

The classification act like Algorithm.1.

Algorithm 1: HMM Classification

Input: observation matrix $O^t(i)$, model parameters λ_j
Output: $Strategy_i$

```
for i = 1; i ≤ m do
    m_i = 1;
    for j = 1; j ≤ n do
        if LogP(O^t(i)|λ_j) > LogP(O^t(i)|λ_{m_i}) then
            m_i = j;
    Strategy_i = m_i;
return Strategy_i;
```

Note that, as for the distinguishable strategy like "Zealot Rush" show in Fig.3 , the model performed well, while it can not discriminate the confusing strategies like "Two Gate Observer" and" Two Gate DT" shown in Fig.4.

D. Modified HMM-KNN Classification

In order to identify the confusing strategy, we combine the HMM model with KNN model.

Look back to the definition of strategy, though game players own various strategies, the key component called build order is necessary. The build order made up of buildings, units and upgrades constituted an individual sequence which inspired us to do the sequence comparison.

For instance, involving the features defined in Table , the build order of replay 1 has a partial sequence formed like "BAADDCABDDA" , while sequence from replay 2 is "BBADCBA" . To calculate the sequence similarity, we prefer the Needleman-Wunsch algorithm which is widely

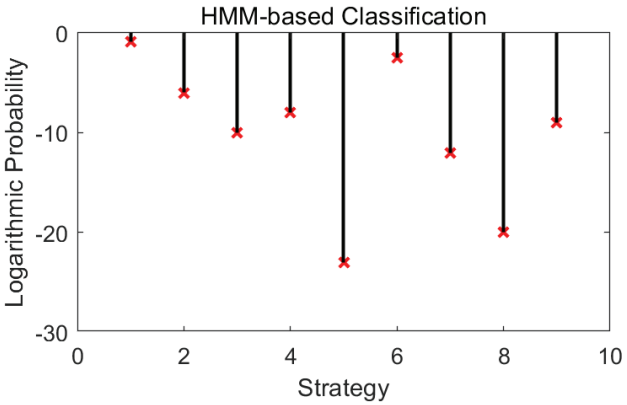

Fig. 3. Distinguishable HMM classification.

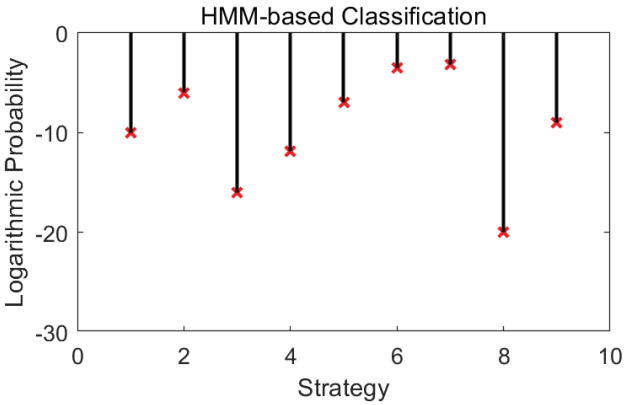

Fig. 4. Confusing HMM classification.

used in DNA sequence matching [17]. LCS(Longest Common Sequence) is defined as follows:

$$L[i,j] = \begin{cases} 0 & if\ i = 0\ or\ j = 0 \\ L[i-1, j-1] + 1 & if\ i,j > 0\ and\ x_i = y_j \\ Max(L[i, j-1], L[i-1, j]) & if\ i,j > 0\ and\ x_i \neq y_j \end{cases} \tag{5}$$

The sequence comparison is show in Fig.5 while the alignment score shown in Fig.6 which illustrated the LCS is 6 and matching score is 2.33. The higher the number, the higher the match degree.

As for KNN algorithm, given build order sequence X and Y, we define the distance:

$$dist(X,Y) = \frac{20}{Score * LCS(X,Y)} \tag{6}$$

We have done five-fold cross validation to verify the parameter K and choose 7 as the final result. We define the testing build order sequence as X_i, and the model central sequence Y_j , sort the top K $dist(X_i, Y_j)$ and form the set $D = \bigcup_K (dist(X_i, Y_j))$, therefore we have the strategy

		B	A	A	D	D	C	A	B	D	D	A
	0	0	0	0	0	0	0	0	0	0	0	0
B	0	1	1	1	1	1	1	1	1	1	1	1
B	0	1	1	1	1	1	1	1	2	2	2	2
A	0	1	2	2	2	2	2	2	2	2	2	2
D	0	1	2	2	3	3	3	3	3	3	3	3
C	0	1	2	2	3	3	4	4	4	4	4	4
B	0	1	2	2	3	3	3	4	5	5	5	5
A	0	1	2	3	3	3	3	4	5	5	5	6

Fig. 5. Build order sequence comparision.

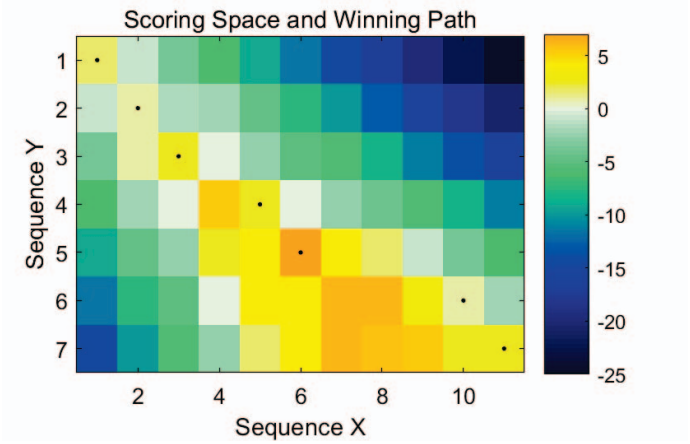

Fig. 6. Build order sequence alignment score.

classification:

$$Strategy_i = \arg\max \sum_K ((X_i, Y_j) \in D) \tag{7}$$

Using PCA(Principal Component Analysis) for dimension reduction, the classifier is shown in Fig.7, comparing to the HMM model, KNN model is more sensitive to confusing strategy classification.

Above all, in order to identify the confusing strategy, we combine the HMM model with KNN model. We firstly choose HMM model , if the difference is not obvious, we prefer the KNN classifier. The integrated classification is shown in Algorithm. 2 .

E. Experiment

We make a comparison among HMM, KNN and their combination in the classification accuracy. As illustrated in Fig.8 and Table.III , in most cases, HMM model own relatively high accuracy, while for the confusing strategies 6 and 7, KNN obtain better performance.

For each strategy, we could obtain an evaluation, thus we have an combined assessment matrix which could generate 9

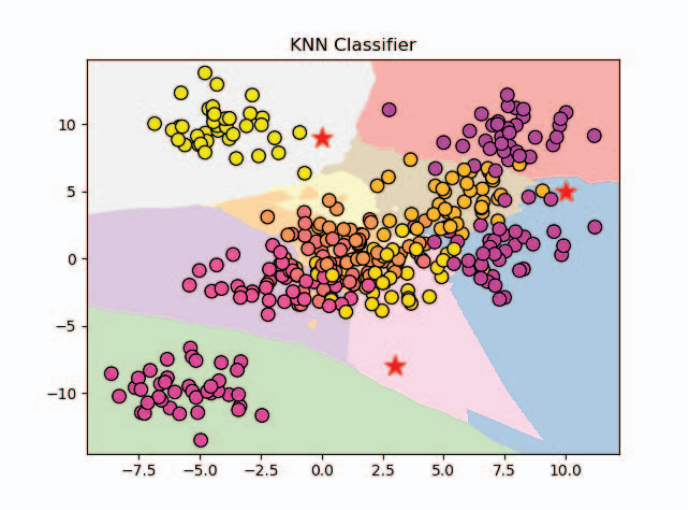

Fig. 7. KNN classification.

Algorithm 2: Modified HMM-KNN Classification

Input: observation matrix $O^t(i)$, build order sequence X_i, model parameters λ_j, model central sequence Y_j and threshold θ_r

Output: $Strategy_i$

```
for i = 1; i ≤ m do
    m_i = 1;
    for j = 1; j ≤ n do
        if LogP(O^t(i)|λ_j) > LogP(O^t(i)|λ_{m_i}) then
            m_i = j;
            if ||LogP(O^t(i)|λ_j) − LogP(O^t(i)|λ_{m_i})|| < θ_r and m_i ≠ j then
                compute and sort dist(X_i, Y_j) as set D = ⋃_K (dist(X_i, Y_j));
                m_i = arg max Σ_K ((X_i, Y_j) ∈ D);
    Strategy_i = m_i;
return Strategy_i;
```

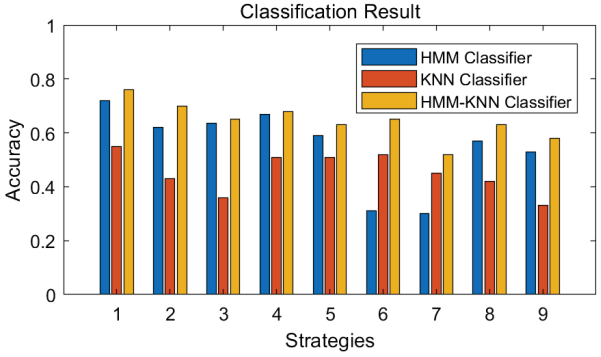

Fig. 8. Classifier accuracy.

TABLE III
CLASSIFIER ACCURACY AMONG STRATEGIES

Algorithm	Zealot Rush	Gateway Expand	Nexus	Arbiter	Forge Expand	Two Observer	Two DT	Core Expand	Two Base Carrier
HMM	0.72	0.62	0.635	0.67	0.59	0.31	0.30	0.57	0.53
KNN	0.55	0.43	0.36	0.51	0.51	0.52	0.45	0.42	0.33
HMM-KNN	0.76	0.70	0.65	0.68	0.63	0.65	0.52	0.63	0.58

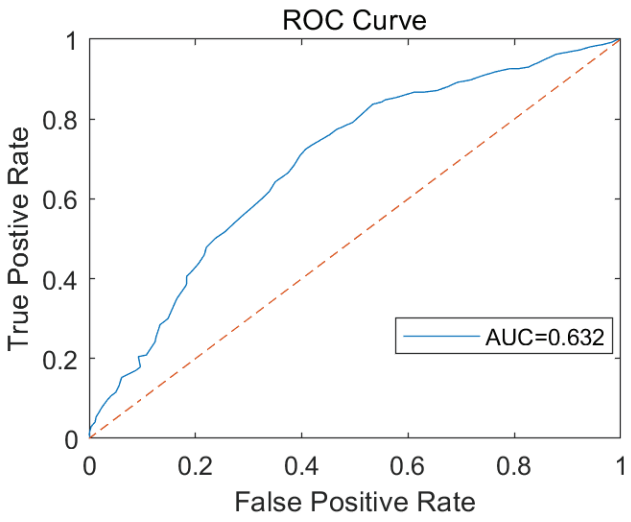

Fig. 9. Classifier ROC curve.

ROC curves. Take average value of the curves and we can get the general ROC curve of the multi-classifier shown in Fig. 9.

IV. CONCLUSION AND DISCUSSION

In order to do the classification of strategies in Starcraft, we combine the HMM and KNN model, the result shows that our approach is effective.

At the same time, we should pay attention to the problems:

- The observation information might be incomplete, though players always send scouter to explore the "fog of war".
- The model is learned from professional player and is available to the classification of regular strategy from players familiar with the game. However it is of no avail to the new bee and unusual strategies.
- The work just focus on the strategy classification while insufficient to the opponent strategy prediction and counter strategy choice.

Some further works should be done on the existing problems and transfering the achievements to the realistic battlefield environment.

REFERENCES

[1] D. Novak, A. Čep, and D. Verber, "Classification of modern real-time strategy game worlds." *GSTF Journal on Computing*, vol. 6, no. 1, 2018.

[2] B. G. Weber, M. Mateas, and A. Jhala, "Building human-level ai for real-time strategy games," in *2011 AAAI Fall Symposium Series*, 2011.

[3] D. Churchill and M. Buro, "Build order optimization in starcraft," in *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.

[4] H.-C. Cho, K.-J. Kim, and S.-B. Cho, "Replay-based strategy prediction and build order adaptation for starcraft ai bots," in *2013 IEEE Conference on Computational Inteligence in Games (CIG)*. IEEE, 2013, pp. 1–7.

[5] E. Dereszynski, J. Hostetler, A. Fern, T. Dietterich, T.-T. Hoang, and M. Udarbe, "Learning probabilistic behavior models in real-time strategy games," in *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.

[6] G. Synnaeve and P. Bessiere, "A bayesian model for plan recognition in rts games applied to starcraft," in *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.

[7] G. Synnaeve and P. Bessiere, "A bayesian model for opening prediction in rts games with application to starcraft," in *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*. IEEE, 2011, pp. 281–288.

[8] H. Cho, H. Park, C.-Y. Kim, and K.-J. Kim, "Investigation of the effect of fog of war in the prediction of starcraft strategy using machine learning," *Computers in Entertainment (CIE)*, vol. 14, no. 1, p. 2, 2016.

[9] G. Synnaeve, Z. Lin, J. Gehring, D. Gant, V. Mella, V. Khalidov, N. Carion, and N. Usunier, "Forward modeling for partial observation strategy games-a starcraft defogger," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 738–10 748.

[10] N. Justesen and S. Risi, "Learning macromanagement in starcraft from replays using deep learning," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2017, pp. 162–169.

[11] P.-A. Andersen, M. Goodwin, and O.-C. Granmo, "Deep rts: a game environment for deep reinforcement learning in real-time strategy games," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018, pp. 1–8.

[12] G. Synnaeve and P. Bessiere, "A dataset for starcraft ai and an example of armies clustering," in *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.

[13] G. Robertson and I. Watson, "An improved dataset and extraction process for starcraft ai," in *The Twenty-Seventh International Flairs Conference*, 2014.

[14] Z. Lin, J. Gehring, V. Khalidov, and G. Synnaeve, "Stardata: A starcraft ai research dataset," in *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2017.

[15] M. Mourad, M. Aref, M. Abd-Elaziz *et al.*, "Opponent models preprocessing in real-time strategy games," *International Journal of Intelligent Computing and Information Sciences*, 2016.

[16] H. Alburg, F. Brynfors, F. Minges, B. P. Mattsson, and J. Svensson, "Making and acting on predictions in starcraft: Brood war," *Bachelor of Science Thesis. University of Gothenburg*, 2014.

[17] G. K. Erickson, "State evaluation and opponent modelling in real-time strategy games," 2014.