Ensemble of Classifiers for Evaluating Game States in Heartstone: Heroes of Warcraft

Michal Myller, Krzysztof Hrynczenko, Szymon Piechaczek, and Jakub Nalepa

Silesian University of Technology, Gliwice, Poland
jakub.nalepa@polsl.pl

1 Introduction

In this report, we briefly present our approach for evaluating intra-game states in Heartstone: Heroes of Warcraft using an ensemble of classifiers. This technique was exploited to elaborate the class labels for the test set (Ψ) provided by the organizers of the *AAIA'17 Data Mining Challenge: Helping AI to Play Hearthstone* held at the FedCSIS 2017 meeting. The initial evaluation of the results obtained with our classification engine (performed at the competition server using approximately 5% of the test data) gave the performance of .7951 (area under the receiver operating characteristic curve). The predictions were submitted by Krzysztof Hrynczenko (jenio) on 2017-05-14, at 21:56:20.

2 Additional Features

The additional features included (all the features were normalized by removing the mean and scaling to unit variance):

- **Cost of each card**. The cost of each card was calculated using the method introduced by Elie Bursztein[1]. These costs allowed to quantify the cards and distinguish the most valuable ones (according to their mana cost and abilities). This approach could be potentially used for spells as well, however it would require understanding the possible interactions between the cards in more detail. Also, it would be tricky to handle if the cards could change.
- **Sum of the player and opponent's costs** of already played cards calculated using the Bursztein's method.
- **Player's hand strength**. The total cost of cards in the player's hand (calculated using the Bursztein's method).
- **Minimum, Average, Maximum and Standard Deviation** of the crystal costs of the player's cards.
- **Health points of both players**.
- **Matchup**. The probability of winning the game by the player based only on the class that the player and the opponent use. Some classes in Hearthstone counter other classes, which means that they have a higher probability of

[1] https://www.elie.net/blog/hearthstone/how-to-appraise-hearthstone-card-values

winning the game even before it starts. For instance, *Warrior* counters *Rogue* (different deck archetypes should be considered, but since the set contained only the basic cards, we decided to focus only on the general cases), and has about 60% win percentage against this class. The probabilities were chosen solely based on our own experience. (It is worth noting that we tried to extract those values from the training set as well).
- **Lethal possibility**. Knowing which minions are able to attack, we verified if the player is able to kill his opponent (and vice versa), also taking into account the minions with a taunt. This was presented as two binary features (`player_lethal` and `opponent_lethal`).

The feature importances were evaluated using random forests (the strength of the played cards, cards in player's hand and the health points of both players appeared to be the most important features).

3 Ensemble of Classifiers: How We Got There?

Our initial efforts were focused on training **artificial neural networks** (ANNs) of various architectures, along with the **logistic regression** and **random forest** classifiers. They were tuned using grid search on a randomly sampled subset of the entire training set $\boldsymbol{T}$, containing 10^5 examples (this subset was balanced). The best performance was exposed by the ANN and was .7800 for Ψ (on the other hand, applying the random forest gave .7005).

Then, these classifiers were grouped into an ensemble of voting classifiers (the label with the larger number of votes wins—in the case of predicting the probability, the average probability was retrieved). The classifiers (of each type) were trained using 10 equinumerous subsets (each containing $2 \cdot 10^5$ examples) of the training set (with no overlaps). Therefore, the initial ensemble encompassed 90 classifiers (9 classifiers—three best-performing models of each type—were trained for every subset). Finally, the ensemble was subsequently refined by removing the classifiers of each type separately. Although it was performed in the trial-and-error manner, we were able to boost the performance to .7951 on Ψ using the ensemble composed of the following classifiers (interestingly, the logistic regression classifiers were completely removed from the ensemble):

- **ANN**, first hidden layer: 20 neurons, second hidden layer: 10 neurons, initial learning rate: .001, $\alpha = .1$
- **ANN**, first hidden layer: 7 neurons, second hidden layer: 3 neurons, initial learning rate: .01, $\alpha = .001$
- **Random forest**, maximum number of features: 13, minimum samples at a leaf: 50, minimum number of samples for each split: 30
- **Random forest**, maximum number of features: 14, minimum samples at a leaf: 50, minimum number of samples for each split: 30

4 Experiments and Setup

Our approach was implemented in `Python` using the `scikit-learn` library along with the `numpy` and `pandas` modules. The experiments were executed on a Phenom II x4 965 BE personal computer (8GB RAM).

5 What Else Could be Done?

Our efforts (unfinished due to the lack of resources) were focused on: (i) including other types of classifiers (support vector machines alongside the k-nearest neighbor classifiers) into the ensemble, (ii) more efficient ensemble pruning and weighting strategies [1], (iii) selecting valuable training sets for the classifiers [2], and (iv) elaborating new features. These additional features (ideally reflecting other game characteristics) would most likely be pivotal to boost the performance of our ensembles.

References

1. Krawczyk, B.: One-class classifier ensemble pruning and weighting with firefly algorithm. Neurocomputing **150, Part B** (2015) 490 – 500
2. Nalepa, J., Kawulok, M.: Adaptive memetic algorithm enhanced with data geometry analysis to select training data for SVMs. Neurocomputing **185** (2016) 113 – 132