

Warcraft II Map Classification

Andrew Trusty
College of Computing
Georgia Institute of Technology

CS4616 Pattern Recognition Final Project

1 Introduction

Warcraft II is a very complex and popular multi-player real-time strategy game. The game requires players to continuously monitor and respond to scores of different game variables in order to manage high-level and low-level economic and military aspects of their virtual kingdom. Because of this it is very difficult to create challenging computer controlled opponents which can match or surpass humans without resorting to cheating.

Researchers in the Cognitive Computing Lab at Georgia Tech under the direction of Ashwin Ram have developed the beginnings of a framework for creating the next generation of artificially intelligent game-playing agents [ICCBR07]. They are using a case-based reasoning system to support the quick development of intelligent agents based on cases extracted from human players. One of the key problems in this open research area is helping the agent to better understand what a human player is thinking while playing the game. In my work, I try to address this problem by developing a set of classifiers to learn a number of new high-level features that I believe are integral to the way humans play strategy games and which can be integrated into the case-based system being developed at Georgia Tech.

The set of features I attempt to learn are based on the game maps. My first goal is to generate a general characterization of the game maps in a way similar to how humans see the map. For example, the below map in Figure 1 is a simple 2-player map which I would characterize as an open-forest map.



Figure 1 - A very simple 2-player map of Warcraft II with two players playing a game.

Attribute	Description
obstacles	the number of obstacle tiles [trees, mountains, walls] on the map
water	the number of water tiles on the map
moveable	the number of moveable land tiles on the map
total	the total number of tiles on the map
water-ratio	the ratio of water tiles to the total
obstacle-ratio	the ratio of obstacle tiles to the total
continents	the number of land masses not completely surrounded by water
islands	the number of land masses completely surrounded by water
water-bodies	the number of disconnected bodies of water
fields	the number of disconnected fields of moveable tiles
forests	the number of disconnected groups of trees
mountains	the number of disconnected groups of mountains
tree-ratio	the ratio of tree tiles to the total
mountain-ratio	the ratio of mountain tiles to the total
obstacle-groups	the number of disconnected groups of obstacles [trees, mountains, walls]
max-water-border-count	the max number of borders water tiles connect
max-water-border-top	the number of water tiles connected on the top border
max-water-border-bottom	the number of water tiles connected on the bottom border
max-water-border-left	the number of water tiles connected on the left border
max-water-border-right	the number of water tiles connected on the right border
max-fields-border-count	the max number of borders moveable land tiles connect
max-fields-border-top	the number of moveable land tiles connected on the top border
max-fields-border-bottom	the number of moveable land tiles connected on the bottom border
max-fields-border-left	the number of moveable land tiles connected on the left border
max-fields-border-right	the number of moveable land tiles connected on the right border
<div style="display: flex; align-items: center; gap: 10px;"> <div style="display: flex; flex-direction: column; gap: 5px;"> <div style="width: 15px; height: 15px; background-color: white; border: 1px solid black;"></div> <div style="width: 15px; height: 15px; background-color: #90EE90; border: 1px solid black;"></div> <div style="width: 15px; height: 15px; background-color: #6495ED; border: 1px solid black;"></div> </div> <div> <p>Attribute used by both map and choke point classification</p> <p>Attribute used only by map classification</p> <p>Attribute used only by choke point classification</p> </div> </div>	

I used ratios for attributes to enable a model to easier generalize the labels for different size maps but I otherwise tried to use attributes that best expressed the content of the map and made sense empirically to me. In this way, I wanted to allow the classifier to access the same attributes that my brain extracted when working to label the maps.

Once I had all my data, I hand labeled the map categories and the choke points. I chose to use only three values for my labels [0.0, 0.5, 1.0] to represent negative, borderline, and positive labels respectively. In this way I could be more consistent with my labels and the labeling would take much less time. Also, I tried to classify choke points based on how centered they were in the map sample so that the larger windows won't misclassify large areas not relevant to the choke point. The distribution of my data labels is below. For example, the map in Figure 1 above would have a label of 1.0 in forest and open-forest but 0 in all other labels. The choke points in Figures 2 and 4 would all be 1.0 in land choke points and Figure 3 would be a 1.0 for water choke points.

Label	0.0 (negative)	0.5 (borderline)	1.0 (positive)
forest	7	27	65
open-forest	39	41	19
dense-forest	47	16	36
water	37	7	55
islands	70	6	23
lakes	79	12	8
continental	91	3	5
river	62	16	21
mountains	56	15	28
open-mountains	80	14	5
dense-mountains	73	5	21
land choke point	732	117	197
water choke point	1004	14	28

3 Approach

In terms of classifiers, my selection was restricted by the fact that some of my attributes and all my labels are real numbers. Not many of the algorithms could handle these conditions. I attempted to discretize my data but classifiers running on the discretized data did not interpolate their predicted labels between the negative, borderline, and positive values I had provided.

I proceeded to try many of the Weka [WEKA] algorithms that could handle my data using the default settings. Neural networks and support vector machines consistently had disappointing results. Nearest neighbors showed promise until I realized the Weka implementation was not interpolating its predicted labels. The best results were obtained using two of the decision tree algorithms, REP and M5P [M5P].

The REP tree learner builds regression trees using information gain/variance and prunes using reduced-error pruning with back fitting. The M5P tree learner uses the M5 pruning criteria and builds model

trees which can have values or multivariate linear models at its leaves which actually allows the tree to extrapolate outside of the given labels unlike regression trees. For both algorithms, I used the default Weka settings and trained them on my data using ten fold cross-validation to get the most out of my limited data sets.

To provide useful results and to take advantage of the fact that my choke point classifiers are able to handle different sample sizes, I adopted a sliding window classification approach. In order to determine where the choke points are on a map I slide my choke point classifier in increments of a third of the window size over all the tiles for all windows sizes from four up to fifteen. The predicted classification is then used to place a Gaussian-like mass in the center of the classification window which is modified by the strength of the prediction. These probability masses then get built up by successive windows so the end result is a combination of all the window predictions which provides a map choke point prediction overlay layer. Below is an example of a map before classification and afterwards with the prediction overlay layer. The combination of low resolution probability masses and windows sliding by a third of their width created the cross pattern seen in Figure 9 as strong responses are generated every window jump.

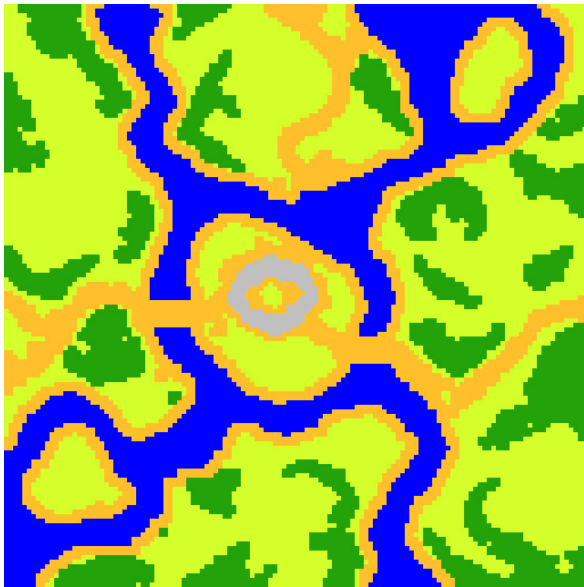


Figure 8 – An unclassified Warcraft II map. Blue is water, dark green are trees, light green is grassland, brown is mud, and grey is mountains.

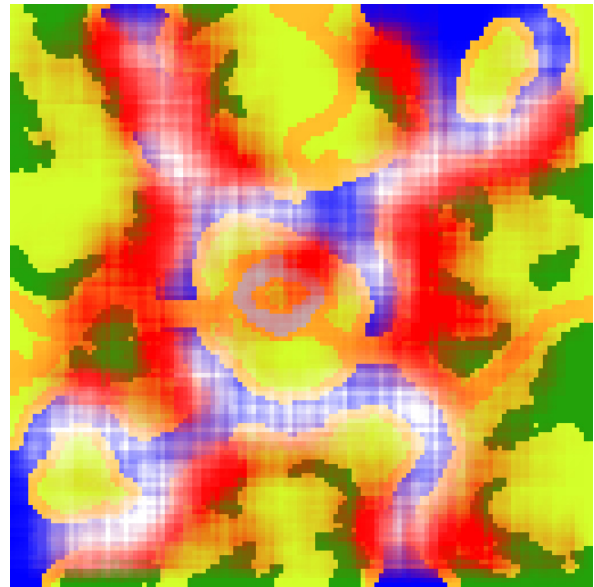


Figure 9 – A map with the prediction overlay layer. Darker red and white represents stronger choke point predictions. (Red = land choke points, White = water choke points)

4 Results

Label	Algorithm	Tree Size (/ M5P Models)	Mean Absolute Error
forest	REP	7	0.1736
open-forest	REP	7	0.2379
dense-forest	M5P	7 / 4	0.2716
water	REP	3	0.0745
islands	M5P	0 / 1	0.2026
lakes	M5P	2 / 2	0.2171
continental	M5P	15 / 8	0.1064
river	REP	11	0.2381
mountains	REP	9	0.1234
open-mountains	REP	9	0.1577
dense-mountains	REP	3	0.0930
land choke point	M5P	39 / 20	0.1609
water choke point	REP	13	0.0244

Once I had the data and labels, the training of the classifiers was very quick process. The M5P and REP algorithms were able to produce models in a matter of seconds or less. In terms of numbers, the table above describes the best performance of the REP and M5P trees I used. It was nice to see that the trees were able to easily pick up on the meaning of the simpler concepts. For example, with water maps the REP tree only looked at the water-ratio and with islands M5P did not even build a tree but rather developed a single model which linearly combined the water-ratio and the number of continents without even looking at the number of islands.

In general, the sub-categories did not perform as well as the general forest, water, and mountain categories. I expected this trend since I had so few map samples. In fact, the continental sub-category was the least represented category and it is obvious that the M5P developed an overly complex model to exactly match the handful of continental maps.

Ironically, the choke points were easier for me to label than the overall map characterizations but the models created by M5P and REP show that this task is much harder for the computer to emulate. It did manage to obtain quite low error rates especially with the water choke points. I found the visualizations of my classifiers performance, like Figure 8 and 9 above, made interpreting and evaluating the classifiers performance much easier. The shortcomings are obvious when the visualizer is run on a variety of maps.

In general there were extremely few false positives for either choke point classifier. The water choke point classifier actually seemed to quite conservative making positive predictions in comparison to the

land choke point classifier. This made many good water choke points not as strongly labeled as they could have been. Although it is somewhat unfair to compare the two choke point classifiers since the data set only had 28 water choke points to the 197 land choke points.

The land choke point classifier produced false negatives for choke points on the borders of the maps and over land bridges as can be seen in Figure 9. While hand labeling, I could not tell if a sample was on the border and so neither can the classifier. A quick fix for this would be to surround the whole map by mountains so that the border becomes an obstacle. For bridges, there were simply no training examples for the classifier to learn from since I sampled randomly from the maps and no bridges happened to be sampled since they are rare. Simply sampling some bridges and retraining the land choke point should solve this.

It is also apparent from watching the visualizer work that the smaller four to six tile classification windows were not able to produce many positive predictions. These window sizes don't cover much map space and almost all of the choke points are as wide as the small windows or wider so the training examples for these window sizes don't provide many positives. In addition, providing larger windows to handle false negatives from large choke points might be appropriate. At the same time you would have to carefully train these larger window sizes not to classify small choke points because they would then produce false positives around the small choke points just due to the window size.

Some, if not many, of the classifier issues are probably due to my error-prone hand labeling which was probably inconsistent in places. The classifiers also could have used more data to alleviate the effects of the curse of dimensionality. But given the results I think my approach stood up well to errors, noise, and insufficient data.

Overall, I was able to extract useful map features, especially in terms of choke points. These features can be usefully integrated into the intelligent agent case-based reasoning system with the help of some additional high-level reasoning to determine which chokepoints are interesting and how to take advantage of them.

5 Future Work

There is room for a lot of future work in this area. More substantial data sets with more positive examples of the map sub-categories and choke holds need to be generated. The existing map attributes need to be better evaluated and new ones developed that better fit each problem. Window sample sizes should also be evaluated for effectiveness.

Work could be done in characterizing the maps based on characterizations of sub-samples of the map through a voting scheme. Similarly, the choke points predictions may prove useful as an attribute in characterizing the whole map.

Computationally the prediction process could be sped up by determining the best way to slide the classifier windows over the map. In addition, on-demand computation of the attributes for the classifiers could be performed since the trees use different sets of attributes depending on which branch is followed and some of the attributes are costly to compute.

Finally, work needs to be done to integrate the features I have developed into the intelligent case-based reasoning agent. This includes post-processing the chokehold predictions to provide useful features for the case database and building a system to reason with choke points and then evaluating the effect of these new features.

6 Acknowledgements

I would like to acknowledge the help of Santiago Ontañón and Manish Mehta in helping me to come up with this project and in helping me brain storm many of the approaches I used. Also, Kane Bonnette provided a lot of useful information about the inner workings of the Wargus code and Warcraft II strategies.

7 References

[ICCBR07]

Santiago Ontañón, Kinshuk Mishra, Neha Sugandh and Ashwin Ram.
Case Based Planning and Execution for Real-Time Strategy Games
In Proceedings of ICCBR 2007

[WEKA]

Ian H. Witten, Eibe Frank, Len Trigg, Mark Hall, Geoffrey Holmes and Sally Jo Cunningham
Weka: Practical Machine Learning Tools and Techniques with Java Implementations
Department of Computer Science, University of Waikato, New Zealand

[M5P]

J.R. Quinlan, *Learning with Continuous Classes*
Basser Department of Computer Science, University of Sydney