Metagame analysis of Natural Selection 2
CMSC 726 Fall 2012 Final Project

Rob Argue
rargue@cs.umd.edu

Abstract

I looked at the game Natural Selection 2 (NS2) from a machine learning perspective in order to analyize its metagame. The first goal of the project was to predict the outcome of individual matches based on a set of features of the round. I found a set of features with which I was able to predict matches with an 85.42% accuracy using an averaged perceptron. The second goal was to attempt to use machine learning to expose which features were most important in deciding a round, with an emphasis on determining what strategies are effective. I used a random forest to extract the most important feature over a subset of all features, and found player skill, resources, and alien upgrades to be the most important features in deciding a game.

Background

Natural Selection 2 (NS2) is an asymmetric first person shooter (FPS) / real time strategy (RTS) hybrid game. Two teams, aliens and marines, are tasked with elimitating the opposing teams base. Teams size can vary, but rounds are most commonly played with between 3 and 12 players per side. Each team has a commander who plays the game as a top-down RTS, while the rest of the team control individual units and play the game as an FPS. Teams can place resource towers (RTs) on resource nodes to coleect resources, which are in turn used to purchase upgrades. Resources go to two pools: team resources (T-res) and personal resources (P-res), which can be used by the commander and players respectively. T-res can be spent to place structures, and purchase upgrades for the entire team, either automatically applying the upgrade the players, or unlocking things for them to buy. P-res can be used to purchase upgrades for an individual player, some of which can also be purchased for a player by the commander using T-res. Some upgrades are limited to a team controlling a number of tech points, which can be captured by purchasing and building a command station for the marines or a hive for the aliens. All upgrades are available with either 2 command stations or 3 hives.

Winning a round comes down to a combination of individual player skill and overall strategy. One rough approximation of player skill is their kill to death ratio (KDR), which encapsulates their in-combat skills farily nicely. Strategy breaks down into two major areas: upgrades and map control. Upgrades are simply what upgrades are purchased by the commander, and at what time (from which upgrade order can be extracted). Map control is a bit harder to define, but includes which rooms are controlled by which team and what locations are being pushed and/or defended. Rooms can have strategic value from their position in the map, and resource nodes and tech points.

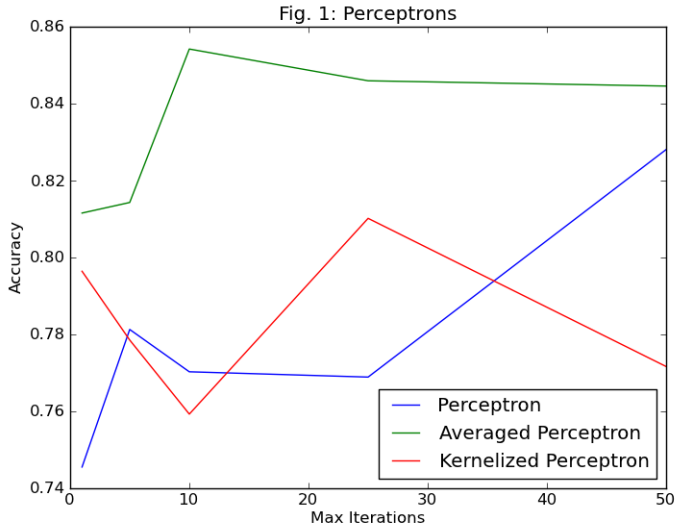
Data

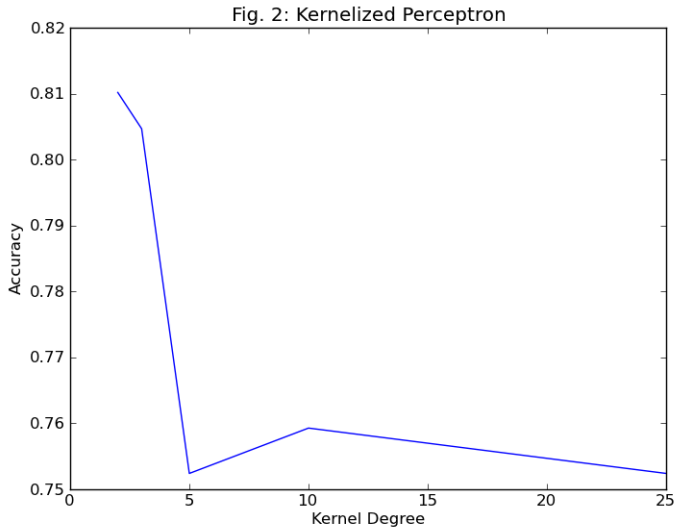
I have obtained a SQL database dump collected by NS2 Stats (`www.ns2stats.org`). For this project I used rounds from NS2 build 229, limited to rounds between 1 minute and 2 hours. From this I had a training set consisting of 2500 rounds and a test set consisting of 700 rounds. Using C# and the LinQ library I extracted features from the database and stored them into a CSV file for use in machine learning algorithms. The features I used for this project were round length, total KDR for each team, marine to alien T-res gathered ratio, marine to alien T-res used ratio, and the initial purchase time of structures and upgrades. I omitted all alien structures except the hive, as they are either directly tied to an upgrade and thus redundant, or fall into the category of structures used for map contol, which was beyond the scope of this project. Likewise sentries and senty batteries were omitted from the marine side. All buildings and upgrades not purchased during the game were set to have a time of 2 hours, which was the maximum time being considered.

It is important to note here what is not contained in this extraction of features. Most information regarding map control was beyond the scope of this project, and was thus omitted. The full information on the building and destroying of RTs is only included vaguely implicitly in the resource features, and could be expanded upon. The number of tech points is also only included implicitly by what upgrades limited to a certain number of tech points are puchased, however the first purchase of a second tech point is included as a feature. Note specifically that this completely omits any direct information about a third hive for the aliens. Player skill has also been wrapped up into a single team feature, which, while very idicative of the skill of a team, again does not tell the full story. Further information could be extracted regarding the spread of skill across a team, or about player skill by using information from other rounds. Part of the reason this last point in particular was omitted was for the robustness of the algorithms in potentially dealing with new players, as well as player skill improving over time.

Predicting round outcomes

Prediction of round outcomes is an important step in being able to talk about the NS2 metagame in terms of machine learning, as it confirms that games can be accurately modeled by a set of features. I decided to start with a perceptron to predict outcomes, and then extended it to an averaged perceptron and a kernelized perceptron. Fig.1 shows a plot of accuracies obtained by each of these three methods over different numbers of max iterations. For this the kernelized perceptron was held at a degree-3 polynomial kernel. The highest accuracy of 85.42% was achieved by the averaged perceptrion with 10 iterations. Fig.2 shows a plot of accuracies obtained by a kernelized perceptron with polynomial kernels of varying degrees, and the number of iterations fixed at 10.





From this it can be concluded that rounds can be fairly accurately modeled with the set of features chosen, using a linear classifier. I believe improvements could be made to this accuracy obtained here with a better set of features which contain more information about map control, but as that was outside of the scope of this project it has be left to future work.

Determining most important features

In order to determine the most important features in deciding a game, I started with the features which were found to be an approximate model of the

game by using a perceptron and ran a random forest algorithm on them. In order to do so, the continuous features were turned into categorical features with what I found to be effective splits. The forest used had 5000 descision stumps which were each created over a random subset of features of size 5. Then a tally was taken over the stumps of feature names. Adding together some of the similar features we arrive at the following top 10 most important features

Feature	**Count**
MarineToAlienResRatio	933
KDR	856
Leap	377
BileBomb	345
Blink	302
Regeneration	235
CragHive	231
Carapace	228
Spores	208
ShadeHive	171

As expected we see that the total resource flow is the most dominant factor in determining games, followed by player skill (as represented by KDR). Also present on the list are three features from one of the three branches in the alien tech tree (*Regeneration*, *CragHive*, *Carapace*), which again, were to be expected, as in NS2 build 229 that path was fairly overtly overpowered. Additionally we see alien lifeform abilities (*Leap*, *BileBomb*, *Blink*, *Spores*), and a feature from another alien tech branch (*ShadeHive*). Drawing from the first two points matching up to expectations I conclude that this is an accurate representation of the important features in deciding a game. From this list we also see that after resources and player skill that the alien upgrade choices are the most important factors in determining a game under this set of features. Also important to note is that the top marine-centric feature on this list was well below the top few features, and was only in 47 trees. This points to there not being any one marine upgrade option that defines a game, which speaks to the linearity of marine upgrades.

An important thing to note about these features is that they do not necessarily confirm that, *Leap* for example, is what wins games, but rather that something about when it is purchased, or that is was not purchased, is an indicator of the outsome of a round. Further work will refine this such that it gives a more direct means of mapping these features to actual gameplay choices.

Future work

I plan to extend this work to add some additional functionality. Right now, one of the major problems with this is that it is only retrospective, that is it is only working on games that are finished. I would like to modify it to also work on games that are in progress, that is, be able to make an accurate guess as to the outcome of a round within the first 5, 10, etc. minutes. Naturally this

won't be as accurate as the purely retrospective case, but could at least be used to make a statement on who is winning a round and by what sort of margin. I also plan to add in more features that have to do with map control, as they are essentially completely omitted at this point.