Molly Wirtz

Professor Neil Heffernan

CS 4341 Introduction to Artificial Intelligence

15 September, 2019

**Project 3: Decision Trees for Connect-4**

**Features:**

1. (Required) Which player has a piece at the bottom left corner of the board?

I calculated this feature by obtaining the first index of each example, as the board is built starting in the bottom left corner, and returning the player whose chip is residing there. If there is no chip, a zero is returned.

I do not believe this feature will have a significant impact, but will evaluate it as a required feature.

1. (Required) Which player has more pieces in the center columns?

I calculated this feature by iterating over the center three columns, totaling the pieces per player residing there, and comparing the totals. I return the player with the most chips, or if a tie, zero.

I think this feature will work because the center columns hold more opportunity for winning patterns, as they are less likely to be cut off by the edge of the game board. While occupying desirable space, the player who controls this area is also at an advantage, as they block their opponent from desirable spaces.

1. Which player has the most top pieces in the columns?

I calculated this feature by iterating up the columns, and making note of which player has a chip in the final (top) spot of the column. If the column is empty, I return a zero.

I think this feature will work because it shows if the player is taking advantage of the upwards mobility the game allows. Stacking pieces allows opportunities both vertically and diagonally.

1. Which player the most chips in horizontal connections?

I calculated this feature by iterating over each row, and totaling the number of chips each player has in a row (e.g. 2 or more chips). After examining the entire board, I return the player with the highest number of horizontal connections, and if a tie, zero.

I think this feature will work because it will tell who is creating opportunities for a four in a row connection horizontally, directly translating into potential winning moves.

1. Which player has a piece in the middle spot in the bottom column?

I calculated this feature by obtaining the chip residing the bottom middle spot on the board and returning the player whose chip it belonged to; if the space is empty, I returned a zero.

I think this feature will work because of research I did during the first project. Many sources said the bottom-row middle-column spot was key to winning the game, and I believe this may be true, for the same reasons mentioned above in feature one.

**Decision Tree Testing:**

I created my decision tree using sklearn’s Decision Tree Classifier, with a test size of 20% and no shuffle. I implemented cross-validation using sklearn’s Cross Validation Score function, with a 5-fold cross-validation. My decision tree classifier produced an average accuracy of 79.3% across the five folds, and the most important feature was calculated to be which player had the most pieces in the center columns.

**Feature Importance:**

To determine the importance of each feature in my decision tree, I decided to test each feature separately by creating a tree with only the feature being tested, and comparing the accuracies of each tree. The tree for my first feature, which player has a piece at the bottom left corner of the board, produced an average accuracy of only 16.8%. The tree for my second feature, which player has more pieces in the center columns, produced an accuracy of 37.7%. The tree for my third feature, which player has the most top pieces in the columns, produced an accuracy of 28.7%. The tree for my fourth feature, which player the most chips in horizontal connections, produced an accuracy of 6.5%. Finally, the tree for my fifth feature, which player has a piece in the middle spot in the bottom column, produced an accuracy of 9.3%. With this data, supported by a statistical analysis, I was able to conclude that second feature (which player has more pieces in the center columns) was the most important feature, followed by the third feature (which player has the most top pieces in the columns). The next most important feature was the first feature (which player has a piece at the bottom left corner of the board). The fourth and fifth features were almost equally insignificant.

I was surprised at these results because I felt sure that the fourth and fifth features (which player the most chips in horizontal connections, and which player has a piece in the middle spot in the bottom column, respectively) would have been significant indicators of predicting a winner, especially more so than which player has a piece in the bottom left corner. For that reason, I was slightly disappointed in the importance evaluation of features. My third feature, however, performed relatively well, which was also a bit of a surprise that it outperformed features four and five.

**Feature Importance on a Decision Tree:**

On the final page of this report, please find the original decision tree I learned, with a test set of 20%, maximum depth of 5, and a 5-fold cross-validation. It may be too small to see, but I will describe it in words as I speak to the analysis of the features. First, please note the colors of each node. Orange signifies Player 1 class, blue signifies Player 2 class, and the shade lightness indicates the strength of the class prediction (the darker the shade, the stronger the prediction). The first split is unsurprisingly the feature calculated to be the most important, which player has more pieces in the center columns. The next layer splits around the feature calculated to have the next highest importance, which player has the most top pieces in the columns. All nodes indicate Player 1 at this point. The next layer begins to become more muddled when it comes to splitting features: one side splits into the center columns feature and the bottom left piece feature, while the other side splits into two bottom left piece nodes, with each node predicting a different class outcome. The tree very quickly gets more complicated, but I will give a brief overview. Although the tree does predict some of the results to be Player 2 class, the majority of the predictions are in favor of Player 1. Additionally, the hierarchy of the tree corresponds with the calculated importance of the features (i.e. most important are higher up in the tree, as they are the clearer and more obvious splits to make in the data).

**Experiments:**

For my three experiments, I decided to implement and compare the depth of the decision tree, information gain versus Gini impurity, and the maximum leaf nodes on the accuracy of a decision tree.

I originally ran my program at a depth of 5 layers, equal to the number of features I was evaluating. I decided to also make a decision tree with a depth of 10 and 15 layers. For a decision tree of depth 10, the average accuracy across all 5 folds dropped to 78.8%, and a decision tree of depth 15 had an average accuracy of 78.9%. After calculating the reliability of these numbers using a T-test, I found that I was not able to reject the null hypotheses to conclude that there was a significant difference data between the original accuracy, the depth 10 accuracy, or the depth 15 accuracy. Therefore, my result of the difference in accuracy between 5 layers and 10 layers, as well as 10 layers and 15 layers, is not reliably different.

For my next experiment, I decided to use the information gain option of Decision Tree Classifier instead of the default Gini impurity. Recall that using the default criterion, my average accuracy was 79.3%. When running information gain, the average accuracy when running on five-fold cross-validation was 79.7%. After calculating the reliability of these numbers using a T-test, I found that there was not enough variation to reject the null hypothesis, and therefore was forced to conclude that changing the Gini impurity to information gain did not produced a significantly reliable difference in the accuracy of the decision tree.

For my final experiment, I adjusted the maximum leaf nodes of the decision tree. For my original decision tree, I used the default, which is an unlimited number of leaf nodes, and produced a result of 79.3%. For this experiment, I chose to set the maximum number of leaf nodes to 5, which came to an average accuracy of 72.2%. The T-test showed this to be a statically significant different in accuracy, and can therefore say that the accuracy reliably drops when comparing a decision tree of unlimited leaf nodes and a tree of maximum 5 leaf nodes. I also ran my decision tree classifier with 10 leaf nodes, producing a result of 78.5% accuracy. Although the T-test proved this is not significantly different in accuracy when compared to my original data, it is reliably better than when the maximum leaves were set to 5.

