Static Classifier Team
Brian Fu
Kendall Choy

Goal:

To use a static classifier to more efficiently store position values. Static classifiers are functions that output the primitive given position. We expect the space complexity of a function to be lower than that of storing the raw positional data. Any positions not evaluated correctly can be separately stored.

What we did:

We coded the game, solver, and symmetries.
Transformed the positions into one-hot-encoded arrays.
Trained 3 different classifiers.

Approach 1:

We used a Logistic Regression model and tried to fit the data we had. Logistic regression attempts to fit a single line to minimize the least-squares mean error. This resulted in relatively low accuracy due to random complexity in the order of the data. The line would need to alternate from positive to negative on a short and random interval resulting in a zig-zag shape that isn't easily modeled at low complexity even with a sin interpolation.

Approach 2:

We used a Decision tree model and tried to fit the data we had. Decision trees mock the branching nature of each player's game tree to output a grouped result for the entire branch. Since each player (x and o) are trained separately, the model given a winning position would remember which children result in winning branches and which result in losing branches. This reduces the storage complexity by a logarithmic factor due to the grouping storage of the branches.

Approach 3:

Use TensorFlow, a machine-learning platform. Neural networks are trained on complex pattern matching using identifying neurons. Each neuron can be trained to recognize a specific pattern such as a 2-in-a-row or a 3-in-a-row and make predictions about the probable winner given this data. However, we were unable to design a custom model following this spec in time. This approach could be expanded upon in another semester.