CS534: Milestone Report
Prediction of Chess Endgame

Anderson, Mike; Callahan, Adam; Gutshall, Gregory

May 23, 2012

Abstract

Traditionally, computer chess programs evaluate positions heuristically, by considering for each player factors such as the number of uncaptured pieces, king safety, central control, and number of possible moves available. Endgame positions are often evaluated by starting from positions that are known to be won or drawn, and then working backward through a search tree. We would like to propose a third approach: formulate the problem of deciding the result of a chess position as a classification problem. Toward this end we will implement and test a variety of classification algorithms for our data set, and attempt to justify the performance of our algorithms by comparing the mathematical theory behind each of them to the peculiarities of our problem. We would also like to see how these algorithms perform on reparameterizations of our data (for example, it may be that the exact location of each piece is more information than we need, and that the Manhattan distances between the pieces can tell us just as much about the true class label), and we would like to try simpler class label spaces (e.g. use 2 class labels instead of 18: either a position is won for white or it's drawn).

1 Methods

We are going to attempt three different methods of classification for this problem. We will then supply a heuristic analysis of the results and also provide a hybrid method to achieve the highest prediction accuracy. We will be using cross-validation on the training set to ensure that over-fitting is minimized.

1.1 Decision Tree

Mike is responsible for implementing the Decision Tree Method.

Completed:

- Implemented entropy and mutual information.
- Implemented a set of 3 meta-features, one of which solves a 2-class version of our problem (draw or no draw).
- Ran a number of experiments to see how much each original feature and each meta-feature reduces the entropy of the class label.
  - Meta-features perform fairly well, but only reduce a fraction of the entropy in the class.
- Started implementing decision trees with information gain criterion.

Remaining work:

- Finish implementing decision trees with information gain criterion.
- Implement a meta-feature that solves another 2-class version of our problem (checkmate or no checkmate).
- Implement more advanced decision tree modifications.
  - Bagging (and possibly boosting).
  - Over-fitting avoidance:
    - Early stop.
    - Post pruning.
  - Try different splitting criteria other than information gain (e.g. bias avoidance).
- Run many different experiments with decision trees using different combinations of the original features and some meta-features, and the more advanced decision tree modifications.
- Report and compare results, try to theoretically justify the differences in performance between different flavors of the decision tree algorithm.

1.2 Neural Network

Adam is responsible for implementing the Neural Network Method.

Completed:

- Reparameterize data

Remaining work:

- Construct a neural network algorithm with parameters ($|\mathbf{X}|$, number hidden nodes, class labels) with a single hidden layer and an output layer that has the number of possible outcomes, $\mathbf{y} \in [1, 2, 3, \ldots, 17]$ to the number of output nodes.
- Try different numbers of hidden layer nodes and compare performance.
- Use online and online with momentum and back propagation to train network.
- Try multiple runs with different initial inputs in order to avoid possible local minima. Also first try using just X_0, X_1 and X_2 - then include X_3 and X_4.
- When this is completed, try filtering the data first to remove draws, etc as is being done for the other two methods of learning.
- Compare the results - convergence time and accuracy - amongst the methods here and the other two approaches for the project.

1.3 Support Vector Machine

Greg is responsible for implementing the SVM Method.

Completed:

- Reparameterize and scaled data.
- Implemented Linear and Polynomial kernel to $\mathbf{X}$
- Tested Pegasos: Primal Estimated sub-Gradient solver for SVM
  - Broke the original multi-class problem into smaller two class problems
  - Used cross-validation with $k = 0.1 * m$, where m is the row length of $\mathbf{X}$

  - Tunned λ and C to achieve best class separation accuracy of 96% ($y_i = 3$ vs. $y_i = 16$) and worst seperation accuracy of 68% ($y_i = 13$ vs. $y_i = 14$).
- Created some data visualization scripts for viewing the decision boundary in $\Re^3$

Remaining work:

- Implement Radial Basis Function(RBF) kernel
- Test and implement multiclass methods
  - One-versus-the-rest
  - Voted one-versus-one
  - Rank-Loss
  - Error Correcting Output Code
- Try to justify results and possibly create a performance bound.

Appendices

Purposely left blank