Clustering Similar Chess Positions

Jacob Leppek
MSCAPP
leppekj@uchicago.edu

Ashu Tayal
MPP
ashutayal@uchicago.edu

Abstract

Improvement in chess relies on pattern recognition; as such, the ability to search games by position is key to understanding different responses and strategies. Yet the number of legal chess positions is estimated to be possibly 10^{40}. As such, very few of these positions have actually been played. To accommodate for this, we explore a number of approaches to approximating positions that are highly similar to the initial query. We encode positions as vectors; use k-means to group similar positions together to limit the number of positions searched; and finally return positions with the highest cosine similarity scores from the searched cluster. The GitHub repository containing the code for this project is accessible online.[1]

1 Introduction

It is often beneficial for a chess player during a game to know whether previously archived chess games lead to positions approximately similar to the current position. It allows players to improve through repetition and understand how positional advantages emerge with seemingly inconsequential differences in piece position. It can, thus, be really helpful in deciding the next set of moves. Many chess databases exist online that let users retrieve games based on position. This allows players to input a position and see what games that position emerged from as well as what moves were played before and after. But no free solution exists for letting users search for games with similar positions.

Our goal is to understand how linear algebra could provide a possible solution to returning similar positions from a given position, allowing users to generate new games with subtle differences that they can use to improve their positional understanding. Given the large possible set of chess positions, this also allows users to find the most similar board positions when no exact match is possible. We are curious about how linear algebra can be used to develop similarity scores for board positions and recommend new positions to players for study.

A 2014 paper from Dublin City University provides a discussion of approximate searching methods from an information retrieval perspective.[2] The authors briefly discuss encoding the chess board and pieces as feature vectors before dismissing it as inconvenient relative to textual representations for search purposes. They discuss what features should be used when developing a similarity score, and construct functions to calculate and combine these features for database retrieval purposes. We have not come across other academic papers that explicitly discuss this topic.

2 Background and Methodology

This section provides an introduction to some chess terminologies; one way to measure similarity of chess positions using the cosine similarity score; k-means classification, a popular unsupervised learning technique in machine learning; and the most-widely used formats for representation of chess positions and moves - the FEN and PGN notations, respectively.

[1] https://github.com/leppekja/chess-mml

[2] http://doras.dcu.ie/20378/1/ganguly-sigir2014.pdf

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Chess notation and terminologies Chess is a two player game played on a 8x8 board with 32 pieces. Each piece has a well-defined set of positions it can take on the board, and the objective of the game is to capture the opponent's king. Each square on the chess board can be represented by a two character notation - the coordinate pair. The Standard Algebraic Notation (SAN) uses a letter and a number. The columns from white's left are represented by the letters from a through h, while the rows are denoted by the numbers 1 to 8 starting at the bottom. Thus, the bottom left corner – the origin of this coordinate system – is denoted by a1, while the farthest square – the top right corner – is denoted by h8.

PGN Notation The Portable Game Notation is a format to record moves in a chess game. It is human readable, and is recognized by most chess software. The encoding of a move consists of the piece being moved along with its destination square. If this piece takes an opponent's piece in that move, an additional 'X' is added before the destination coordinates.

FEN notation The Forsyth-Edwards Notation encodes a particular position of a game, as opposed to encoding a move. The FEN string can be decoded into a position matrix for the 8x8 chess board. Our code takes a FEN string as the input position.

K-means K-means is a popular unsupervised machine learning algorithm. The objective is simple - to find and group similar data points together based on underlying patterns. K-means requires user input to determine the number of clusters 'k', and tries to classify points into one of these k clusters by minimizing their distance from a center point that is updated with every iteration. Put differently, K-means tries to minimize the intra-cluster sum of squares.

Cosine similarity scores Cosine similarity is a measure of similarity between non-zero vectors that measures the cosine of the angle between the vectors. This value can range from 0 to 1; a higher score indicates more similarity. It is important to note that this is merely a measure of similarity of orientation, not magnitude, as the vectors are normalized. It is often used with high dimensional data to measure the cohesion between clusters. We use this metric to measure intra-cluster position similarity.

Interpreting Distances The Euclidean distance between two chess positions can be interpreted as the average number of total moves required for the first position to overlap with the second.[3] Since a lower distance indicates more similar positions, in general, we expect the distance to go down as we increase the number of clusters.

3 Discussion

We use a data set of roughly 60,000 positions. There are 64 features; each feature corresponds to a specific square on the board. What piece occupies that square is denoted by a negative or positive value, which reflects whether the piece is black or white. Different pieces are given different values, e.g., a pawn is 1 and a king is 10.

Given the massive variation in positions, this means that our feature matrix is high-rank, no feature in our matrix is redundant and positions are very rarely repeated. This creates two problems: clustering is more difficult in higher dimensions and dimension reduction means we are losing key information that informs how to cluster positions.

The elbow curve is a heuristic used to suggest the appropriate number of clusters k to use when performing k-means clustering. Above, we plot the average distance from each point to the centroid of its assigned cluster for $k = 1, 3, 5, 7, 9, 11, 15, 18, 21$. While we see a general decrease as the number of clusters increases, this curve offers no clear sharp breaks to determine an appropriate number of clusters. Instead, it shows another issue with the clustering approach - the distances are still high, even as the number of clusters increases to 21, indicating that a large amount of variability still exists within each cluster.

We can confirm this by averaging the cosine similarity of each position to every other position within a sample of positions taken from each cluster. Above, we use 9 different options of k clusters. For

[3] https://www.chessprogramming.org/Distance

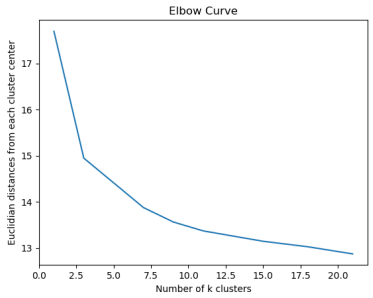

Figure 1: Elbow Curve for 8 different cluster sizes.

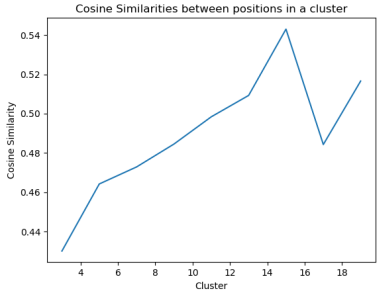

Figure 2: Average Cosine Similarity across clusters in a kmeans model.

each cluster within the 9 trained models, we sample 100 positions and calculate the cosine similarity scores for each position relative to the 99 other positions, which results in $\binom{10}{2}$ calculations, since order doesn't matter for the cosine similarity equation. We see that while the average of the average cosine similarity within each cluster does increase when we increase the number of clusters within each model, it still does not reach a high enough level to appropriately return similar positions in most cases. Note that the sample size from each cluster is relatively small, and that variability is still high for each test.

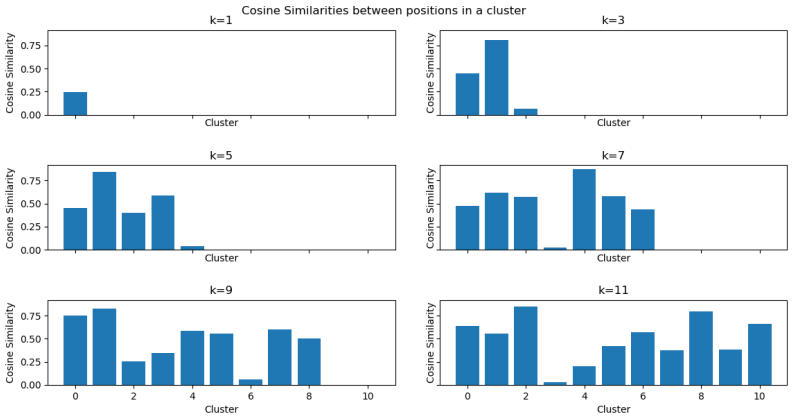

Figure 3: Average Cosine Similarity between positions within a cluster.

Further examining this, we see that the average similarity scores are skewed, usually by a very low outlier cluster. Even with this outlier considered, it is clear that the clusters do not do very good in clustering similar positions together.

Principal Component Analysis (PCA) assumes that the data is drawn from a single low-dimensional linear subspace. Likewise, k-means is unable to provide high clustering accuracy because of high dimensionality. Broadly, this problem is often called the 'curse of dimensionality' - which can refer to the presence of irrelevant attributes conceals useful information or the inefficiency that arises from performing operations on high dimensional data. Intuitively, clustering becomes increasingly harder with higher dimension data because the relative contrast between 'near' and 'far' neighbours diminishes as dimensions increase. A 2012 paper on clustering in high dimensions illustrates why k-means and other commonly used clustering techniques are inelegant when used with high dimensional data.[4] The authors show that the range of distances shrinks rapidly as the dimensions are increased; the standard deviation of distance between neighbour virtually disappears at 10 dimensions. This phenomenon is referred to as 'distance concentration'.

In the following discussion, we use a k value of 10 for sake of visualization. We look at how these points are clustered together. We can use PCA to decrease the dimensionality of the dataset.

Figure 4: Positions colored by cluster assignment in 2D

A few clear clusters emerge when performing PCA. Yet the data is not split well in two dimensions, resulting in a large clump.

4 Conclusion

Overall, K-Means does not provide an effective way to group clusters together due to the high-dimensionality and shape of the data. Any natural clusters are very different sizes; the start position is guaranteed to be in each game, for instance, while the chance of every other position is less than 1. Furthermore, from PCA, it is clear that positions are clumped together, making a clustering algorithm not the more suitable solution. Applying PCA on a chess position means losing data, and while this does speed up the algorithm, it does not provide better results. However, using the cosine similarity does provide an excellent comparison between two positions, and encoding the positions as vectors does make sense in this context.

[4]https://doi.org/10.1002/sam.11161

References

[1] Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U. (1999). When is "nearest neighbor" meaningful?. In *International conference on database theory* (pp. 217-235). Springer, Berlin, Heidelberg.

[2] Distance - Chessprogramming Wiki." *Accessed December 12, 2020.* https://www.chessprogramming.org/Distance.

[3] Ganguly, D., Leveling, J., Jones, G. J. (2014). Retrieval of similar chess positions. In *Proceedings of the 37th international ACM SIGIR conference on Research development in information retrieval* (pp. 687-696).

[4] Zimek, A., Schubert, E., & Kriegel, H. P. (2012). A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal,* 5(5), 363-387.