My First LaTeX Document

Mohammad Ibrahim Khan

December, 2024

Contents

List of Figures

List of Tables

1 Background and Context

1.1 Chess

Modern chess is a game which had its origins in India, dating back to the 6th century [1] as a way of devising strategy and tactics in war. Today it is a game seen as a benchmark for skill and intelligence, played by people in their millions. Some see chess as a way to relax while others see it as a sport, a competition [2].

1.2 Chess Engines

Since 1997 when IBM's Deep Blue beat Kasparov [3], a world champion, chess engines have been a popular topic of research. Chess engines analyse millions of positions per second in order to defeat the best human players. However, even with technological advancements since 1997, chess is still unsolved. According to Shannon, there are 10^{120} possible positions in chess [4], making it infeasible to generate all possible positions and evaluate them. Engines have implemented a variety of ways in order to reduce this search space which will be discussed in subsequent sections.

1.3 Search Algorithms

Chess engines most commonly implement a minimax search algorithm, where every node is a in the game and the legal moves create the next layer of nodes. Ideally, the engine would search until the end of the tree, thus always finding the best move. However, due to the very large search space, this is not feasible. Therefore, engines implement a number of different

1.4 Machine Learning in Chess

Many researchers have used machine learning techniques to improve the performance of chess engines. These techniques include classification techniques like Neural Networks and Naive Bayes as well as clustering techniques like K-means as well as reinforcement learning techniques. Some have even used Natural

Language Processing techniques [5]. These techniques are usually used in conjunction with traditional search algorithms.

1.5 Motivation

Chess is still an unsolved game, in the sense that we don't definitively know the optimal strategy for every position. This is the main motivation for many researchers implementing different techniques to try to get close to solving chess. Machine Learning algorithms have been shown to be effective. Naive Bayes isn't a common technique used in chess engines, however, it has been shown to be effective in other domains. This research aims to explore how Naive Bayes can be implemented in a chess engine, especially due to its computational efficiency, making it ideal for fast real time predictions where there is limited computational power, like mobile devices.

2 Literature Review

2.1 Minimax and Alpha-Beta Pruning

The concept of Minimax was first proposed by Shannon in 1950 [6]. A zero-sum game is where "the loss of one player is the gain of the other"[7]. Minimax works on the principle of zero-sum games and assumes that the opponent will always make the best move. The algorithm recursively alternates between the maximising player and the minimising player, until a terminal node is reached. The terminal node is then evaluated using a heuristic function. Figure 2 is taken from Plaat's paper [7] and shows the possible plays for a specific game of Tic Tac Toe. Figure 2 shows the minimax tree for a game of Tic Tac Toe. The algorithm

Alpha-Beta pruning is a an approach used to decrease the number of nodes evaluated by the minimax algorithm. It does this by not evaluating nodes that would not affect the final outcome of the minimax algorithm. It introduces two new values α and β where α represents the maximum value that can be attained and β represents the minimum value that can be attained. If the value of a node is less than α or greater than β, the tree does not need to be traversed further.

2.2 Naive Bayes

Many Machine Learning algorithms have been used in chess engines. The most popular being Neural Networks like used in AlphaZero [8]. Another popular technique is using reinforcement learning where the engine plays against itself and learns from the results [9]. Naive Bayes hasn't been as popular in chess engines. Naive Bayes, sometimes also known as Idiot Bayes or Simple Bayes, is a simple classification algorithm that utilises Bayes' theorem (Equation 1).[10].

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{1}$$

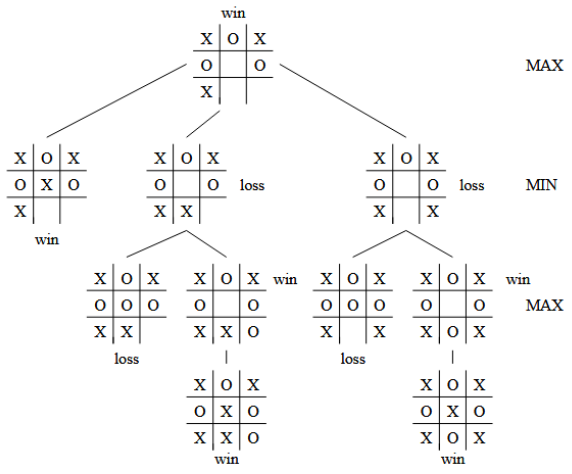


Figure 1: Possible plays for Tic Tac Toe

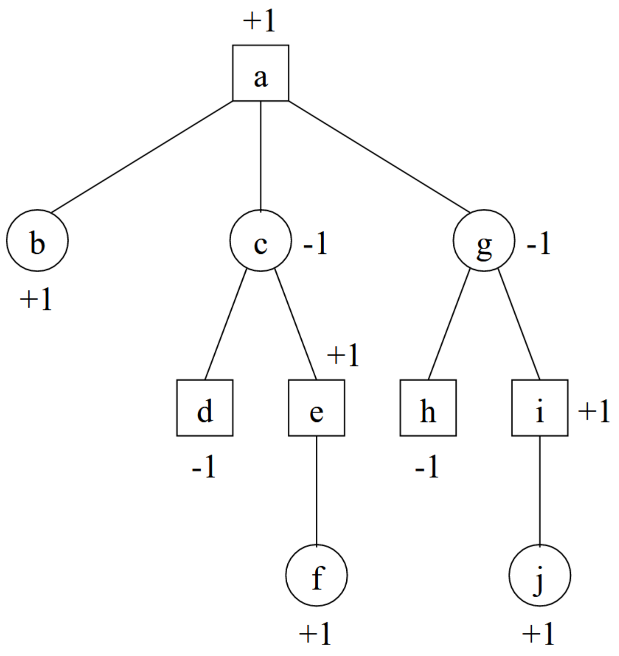


Figure 2: Minimax Tree for Tic Tac Toe

Algorithm 1 Minimax Algorithm

```
function MINIMAX(Node, Depth, MaximizingPlayer)
    if Depth = 0 or Node = Leaf then
        return EVAL(Node)
    end if
    if MaximizingPlayer then
        $Value \leftarrow -\infty$
        for each in Node do
            $Value \leftarrow \max(value, \text{MINIMAX}(child, depth - 1, \textbf{false}))$
        end for
        return $Value$
    else
        $Value \leftarrow \infty$
        for each Child in Node do
            $Value \leftarrow \min(value, \text{MINIMAX}(child, depth - 1, \textbf{false}))$
        end for
        return $Value$
    end if
end function
```

It is referred to as naive as it assumes that each input variable $X_1, X_2, ..., X_n$ is conditionally independent given the classDespite this assumption not being true in most cases, Naive Bayes has still been shown to work well. This assumption allows probability distributions to be efficiently represented as the product of the individual probabilities of the input variables (Equation 2) [10].

$$P(X_1, X_2, ..., X_n, C) = P(C) \cdot \prod_{i=1}^{n} P(X_i|C) \tag{2}$$

Naive Bayes has been shown to be effective in many domains, for example in spam detection [11]. Some academics from Jaypee Institute of Information Technology compared between two different techniques, Naive Bayes Classification and Multivariate Linear Regression. They found that Naive Bayes was better suited for categorical predictions while Multivariate Linear Regression struggled with predictions [12]

3 Requirements

3.1 Functional Requirements

- The system must evaluate chess positions using a Naive Bayes Classifier.
- The classifier should be able to output a classification for a position. (Winning, Losing, Drawing)

- The evaluation should be integrated with Minimax and Alpha-Beta pruning.
- The engine should be able to play a full game of chess.
- The engine should be able to evaluate the board position.
- The engine should be able to make a move based on the evaluation.

3.2 Non-Functional Requirements

- The classifier should be computationally efficient for real time evaluation.
- The classifier should be scalable to handle large datasets.
- The system should be able to evaluate a position in less than 4 second.
- The system should be able to run on a standard laptop.
- The system should be testable against Stockfish.

4 Methodology

4.1 Problem Definition

Naive Bayes has been proven to work in certain domains, this research aims to see if it can be effective in a chess engine. The system will be implemented in Python, utilising the python-chess package for board representation and move generation. The Naive Bayes classifier will be trained to categorise chess positions into 3 categories, Winning, Losing, and Drawing. The classifier will be used within the minimax algorithm to then help decide which nodes to evaluate further and which to terminate at.

4.2 Data Collection and Representation

The classifier will be trained on a dataset of chess positions. The dataset that will be used is from the lichess database. This database contains over six billion games of chess. Due to constraints on computational power as well as time, only a subset of this database will be used. Due to the nature of Naive Bayes, the dataset will need to be preprocessed to include two main parts, the features and the labels/classes. The labels is what the classifier will predict. We will use three labels for the classifier:

- Winning
- Losing
- Drawing

The features are the input variables that will be used for the classifier since Naive Bayes is a supervised learning algorithm which require numerical or categorical features to calculate the conditional probabilities of the labels. These features are the "evidence". For the purpose of this research, the following features will be used:

4.2.1 Material Balance

The material balance is arguably the most important feature in chess. Material balance is the difference between the number of white pieces and the number of black pieces refer to Equation 3. This feature is a good indicator of the current standing of the game. The material balance is positive if White has more pieces than Black and negative if Black has more pieces than White, and 0 otherwise.

$$\text{Material Balance} = \text{Number of White Pieces} - \text{Number of Black Pieces} \tag{3}$$

However, considering each piece of equal value is not representative of the true value of pieces during the game, for example arguably 1 queen is worth more than 2 pawns. Keeping this in mind, this research will use the values in Table 1 to calculate the material balance, which are the commonly accepted values of pieces in chess[13]. Many other values have been proposed, however, this research will use these values as they are the most commonly accepted.

Piece	**Value**
Pawn	1
Knight	3
Bishop	3
Rook	5
Queen	9
King	0

Table 1: Values of Chess Pieces

4.2.2 King Safety

Material Balance is a very good indicator of the current standing of the game, however, it does not consider anything related to the king, which could single handedly change the outcome of the game. King Safety is a feature that will be used to evaluate the safety of the king. Again, there are many proposed methods to evaluate the safety of the king [14], however, this research will calculate King Safety solely based on the number of pieces attacking the king.

4.2.3 Piece mobility

Another drawback of relying solely on material balance, is that it doesn't consider the state of the game, if you have the same pieces in two games, the material balance would be the same, however, the state of the game could be very different. One way of tackling this issue is introducing the idea of piece mobility. Piece mobility is the number of legal moves a piece can make. The more mobility one side has, generally indicates they have more control over the board. This feature will be the difference of the number of legal moves White has and the number of legal moves Black has as shown in Equation 4.

$$\text{Mobility} = \sum_{i=1}^{n} \text{LegalMoves}_i^{\text{White}} - \sum_{j=1}^{m} \text{LegalMoves}_j^{\text{Black}} \tag{4}$$

4.2.4 Pawn Structure

Despite being the weakest piece in chess, pawns can be instrumental in the progression of the game due to their ability to control the centre of the board and being able to promote to another piece. There are 3 main pawn structures that we will consider. Doubled pawns, isolated pawns, and passed pawns [15].

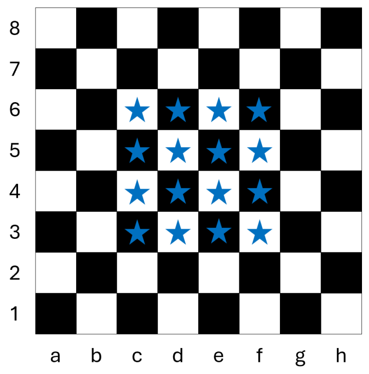
- **Doubled Pawns**: Doubled pawns are two pawns of the same colour that are on the same file. They are usually seen as weak since they can't defend each other and can be easily attacked.
- **Isolated Pawns**: Isolated pawns, also known as isoloni pawns, are pawns that do not have any friendly pawns on adjacent files. They are also considered weak as they can't be protected by other pawns
- **Passed Pawns**: Passed pawns are pawns that have advanced past all opposing pawns [16]. In other words they have no enemy pawns in front of them or on adjacent files that can stop them from being promoted [16]. Unlike doubled and isolated pawns, passed pawns are considered strong as they can be easily promoted to a queen, forcing the opponent to use their pieces to stop that pawn.

4.2.5 Control of the Centre

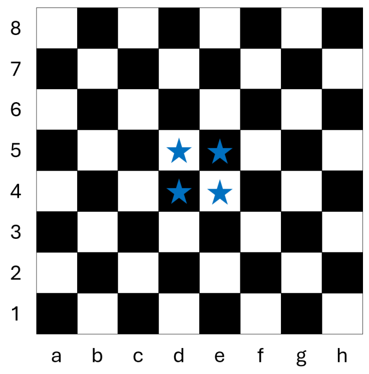
Another important feature is to consider the control of the centre. Having control of the centre is a strong advantage as it allows for more mobility and more options for the player [17]. This will be evaluated by calculating the number of pieces in the centre of the board and taking the difference between the number of white pieces and the number of black pieces. The centre of the board is defined by the 16 squares in the middle of the board (Figure 3a), giving more weight to the 4 squares in the centre (Figure 3b).

4.3 Training the Classifier

Once the data has been collected and preprocessed, the classifier can be trained. There is the input data and the labels/classes that the input data will be classified into. The data will be split into training data and testing data, with a 80%/20% split respectively.



(a) Big centre of the Board



(b) Small centre of the Board

Figure 3: Comparison of the small and big centres of the board.

4.3.1 Prior Probabilities

The first step is to calculate the prior probabilities of the classes, which is based on the training data. Equation 5 shows how it is calculated, where C is the class, N_c is the number of instances of the class and N is the total number of instances in the training data.

$$P(C) = \frac{N_c}{N} \tag{5}$$

4.3.2 Likelihood Probabilities

The likelihood probability, $P(X|C)$, involves calculating the probability distribution of each feature given a class. There are 2 main types of Naive Bayes classifiers, Gaussian Naive Bayes and Multinomial Naive Bayes. Gaussian Naive Bayes is used for continuous features so calculates the likelihood probability using a Gaussian Normal Distribution (Equation 6).

$$P(X|C) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(X-\mu)^2}{2\sigma^2}} \tag{6}$$

- X: Feature
- μ: Mean of the feature
- σ: Standard Deviation of the feature

Multinomial Naive Bayes is used for discrete features so calculates the likelihood based on the count of feature in the class (Equation 7).

$$P(X|C) = \frac{N_{X,C} + 1}{N_C + V} \tag{7}$$

- $N_{X,C}$: Number of instances of the feature X in class C
- 1: Laplace smoothing constant
- N_C: Number of instances in class C
- V: Number of possible values for X

For this research we use features that are continuous and discrete, so we will use both Gaussian and Multinomial Naive Bayes and then evaluate which one fits better for the data.

4.3.3 Posterior Probabilities

Now that we have the prior and likelihood probabilities we can use Bayes' Theorem to calculate the posterior probabilities.

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)} \tag{8}$$

P(X) is considered the "evidence" and is the same for all the classes so can be ignored when comparing the posterior probabilities. To calculate the predicted class for a given instance, we choose the class with the highest posterior probability as shown in Equation 9.

$$PredictedClass = \arg\max_{C} P(C|X) \tag{9}$$

4.3.4 Evaluation

The classifier will be then evaluated using the testing data. We will use accuracy and f_1 score as the evaluation metrics The accuracy is the ratio of correct predictions to total predictions and the f_1 score is the harmonic mean of precision and recall.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{10}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{11}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{12}$$

$$f_1 = \frac{1}{\text{Precision}^{-1} + \text{Recall}^{-1}} \tag{13}$$

4.4 Integration with Minimax

We now want to integrate the classifier into the minimax algorithm. The classifier's output will be used during the evaluation phase of each node. The classifier will output the probability of winning, losing, or Drawing for a given game state. Using this a single score will be calculated.

$$\text{ClassifierScore(X)} = P(\text{Winning}|X) - P(\text{Losing}|X) \tag{14}$$

Then this score will be used in conjunction with a traditional evaluation function to evaluate a overall score, which will inform which nodes to evaluate further and which to terminate at.

References

[1] H. A. Davidson, *A Short History of Chess.* Crown.

[2] Y. Averbakh, *A History of Chess: From Chaturanga to the Present Day.* SCB Distributors.

[3] F.-H. Hsu, "IBM's Deep Blue Chess grandmaster chips," vol. 19, no. 2, pp. 70–81.

[4] C. E. Shannon, "XXII. Programming a computer for playing chess," vol. 41, no. 314, pp. 256–275.

[5] I. Kamlish, I. B. Chocron, and N. McCarthy, "SentiMATE: Learning to play Chess through Natural Language Processing."

[6] Y. Xie, W. Gao, Z. Dai, and Y. Li, "Research and Improvement of Alpha-Beta Search Algorithm in Gobang,"

[7] A. Plaat, "Research Re: Search & Re-search."

[8] D. Klein, "Neural Networks for Chess."

[9] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," vol. 362, no. 6419, pp. 1140–1144.

[10] D. Lowd and P. Domingos, "Naive Bayes models for probability estimation,"

[11] J. J. Eberhardt, "Bayesian Spam Detection," vol. 2, no. 1.

[12] Department of Computer Science and Engineering, Jaypee Institute of Information Technology, Noida, 201304, India, P. Lehana, S. Kulshrestha, N. Thakur, and P. Asthana, "Statistical Analysis on Result Prediction in Chess," vol. 10, no. 4, pp. 25–32.

[13] A. Gupta, C. Grattoni, and A. Gupta, "Determining Chess Piece Values Using Machine Learning," vol. 12, no. 1.

[14] "King Safety - Chessprogramming wiki."

[15] "Pawn Structure - Chessprogramming wiki."

[16] "Passed Pawn - Chess.com."

[17] A. Mikhalchishin and G. Mohr, *The Centre. A Modern Strategy Guide*.