# CLASSIFYING THE WINNING PLAYER IN CONNECT 4

JAI ASLAM, JOHN DARGES AND DANA DROZ

## 1. Introduction

The trademarked version of Connect 4 popular today was originally created by the Milton Bradley company in 1974. It is a two player game played on a $6 \times 7$ grid where each player has tokens of a color specific to them. The two players take turns choosing a column to drop a token into and the token then moves to the lowest available entry of the grid in that column. The objective of the game for a player is to place 4 tokens of their specific color in a row either vertically, horizontally or diagonally. It is possible for neither player to accomplish this goal before the board is full, resulting in a draw.

From a theoretical point of view Connect 4 is interesting to analyze because it is a simple turn-based game that also has a combinatorially large number of possible board positions. According to the OEIS [Inc20], there are 4531985219092 possible states of the game in Connect 4. In 1984, Connect 4 was solved by Victor Allis [All88]. The first player can force a win by beginning in the center column and continuing play optimally. If the first player places their first piece in either of the columns adjacent to the center column then the second player can force a draw. If the first player begins in any other column then the second player can force a win.

The issue with such optimal strategies in games with a large state space is that they are computationally expensive to compute. In this paper we see if there is a way to learn about the optimal strategy without computing it outright. To do this, we consider board positions from part way through the game and apply different classification methods to determine if the game will result in a win or loss for the first player assuming optimal play. We consider neural networks, SVM and decision tree models.

We analyze the neural network model in more depth and see if it learns anything about the game beyond the specific data used to build it. For example, we see if the model knows anything about the symmetry of game outcomes when the board state is reflected about the center column. We also consider whether the model knows anything about boards where the next move is forced. Lastly, we vary the data we train on to see how this effects the percentage of correctly predicted wins, losses and draws.

## 2. Data

The data set is the Connect 4 data set on the UCI machine learning repository [Tro95]. The data set has 67557 instances which consists of all board states where each player has placed 4 pieces and the next move is not forced. See Figure 1 for an example of a forced board state. Since the board is $6 \times 7$ each data point consists of a vector of length 42 where each component corresponds to a position on the board and is either 'x', 'o' or 'b' where 'x' corresponds to player 1, 'o' corresponds to player 2 and 'b' is blank. Each data point comes with a label of win, loss or draw. The labels are the outcome of the game if both players continued play optimally from the current board state. From the 67557 data points 44473 are wins, 16635 are losses and 6449 are draws making up approximately 65%, 24% and 9% of the data respectively.

*Date*: December 9, 2020.

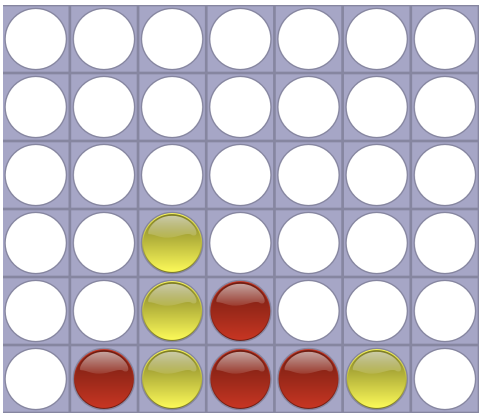


FIGURE 1. Red is forced to play their next token in column 3

## 3. METHODOLOGY

In order to evaluate the effectiveness of our models we first need to establish a baseline accuracy for comparison. Intuitively, the player with more options to place a token to get three in a row will be at an advantage. To make a baseline prediction we count the number of cells a player could place a token which would then give them 3 in a row. We did not check to see if a player could legally place the token in that position yet, only if it was open or not. Whichever player had more such cells was predicted to win. If the number was the same for both players, then a draw was predicted. This provides a better accuracy than assigning the win, loss or draw uniformly at random.

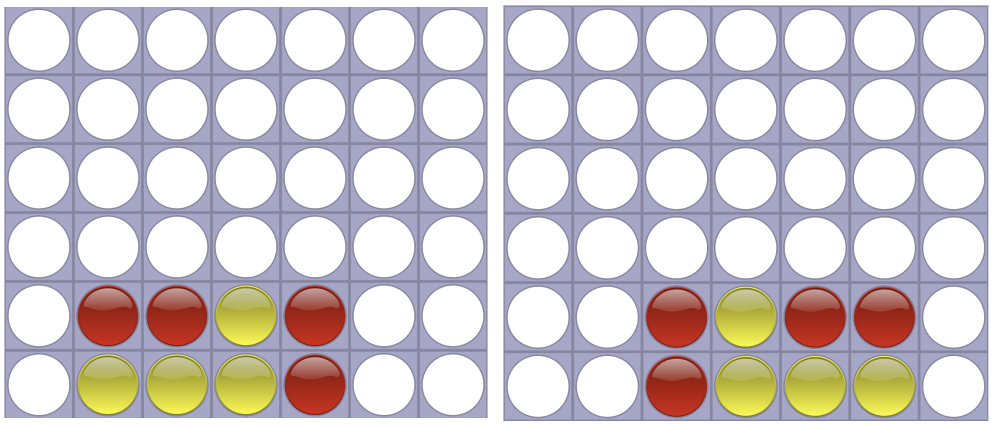
After establishing a baseline, we will compare the accuracy of a few different machine learning methods. Specifically we consider ANNs, SVM models and decision trees. We choose the number of hidden layers in our ANN through experimentation and analysis of the average accuracy and standard deviations of these models. For the decision tree model we find the optimal max depth and min sample split using 5-fold cross validation. We also average these parameters over multiple splits of our data set in order to account for variation in the models due to the split. We then compare the accuracy of the models. Since the data comes in the form of a grid, one natural idea would be to apply a CNN to the data points, interpreting them as images. We did not pursue this idea since CNNs recognize structures regardless of their location in an image. In Connect 4 the middle is far more important to control than the sides of the board, so any accurate model would have to be capable of distinguishing the same structure at a different position in the image.

Since Connect 4 is solved, one interesting question to ask is how much data is needed to train a model to a high degree of accuracy. It's likely that the specific data that is used in training will have a large influence on the outcome. In order to deal with this problem we train 100 neural networks at each of $10\%, 20\%, \ldots, 90\%$ of the data for a total of 900 models. At each percentage level we compute the average accuracy of the 100 models as well as the standard deviation.

Our data is skewed toward wins and draws represent only 9% of the data. This led us to question if our accuracy measurements depend on the label of the data points. To test this, we artificially adjust the data set and even the split between the three labels (6449 of each wins losses and draws) and investigate if that has any effect on the accuracy measurements for each label.

Our data set has an innate symmetry across the center row. This means that mirrored data points should be classified the same. We will investigate whether or not our model learns to recognize this symmetry or not. See Figure 2 for an example of mirrored scenarios.

Our data does not include any scenarios where the next move is "forced." This means that the second player has three tokens in a row and could currently place a 4th and win the game. Thus the first player is forced to place their token to block their opponent. Our model was not trained



(A) Original Data point (B) Mirror image

FIGURE 2. Example of a "forced" scenario and its mirror

on any data of this type, but it has an identical format to our current data and we can use the already existing AI for the game to classify any scenario as a win loss or draw for optimal play. We crafted 63 "forced" data points to see how the model would handle this new data. Two examples of scenarios included in the forced data set can be seen in Figure 2.

## 4. RESULTS

4.1. **Artificial Neural Network.** When designing our artificial neural network we used 40 neuron hidden layer(s) and a softmax output. We chose 40 neurons because it is similar to the number of features (42) in our data, but this choice was somewhat arbitrary and tuning this hyperparameter offers an avenue for further study. When comparing the results between $1, 2$ and 3 hidden layers we used a 20/80 training/testing split. We trained 100 models on 100 epochs and averaged the testing accuracy to get the results in the table below. For the 3 hidden layer case, because of computation time, we only trained 30 models instead of 100. As seen in the Table 1, the 2 hidden

| **Hidden Layers** | **Average Testing Accuracy** | **Standard Deviation** |
|---|---|---|
| 1 | 74.54% | .701% |
| **2** | **75.51%** | **.609%** |
| 3 | 75.44% | .592% |

TABLE 1. ANN Architecture Comparison

layer architecture gave the best testing accuracy. It also had a much shorter run time than the 3 layer version so it was the best choice all around. In what follows, by ANN model we mean the version with 2 hidden layers.

4.2. **Optimal Training/Testing Split.** With the optimal number of layers chosen, we now consider the ANN model in more depth. Specifically, we consider how the accuracy of the model changes as the percentage of data it is trained on increases. For each percentage of data $10\%, 20\%, \ldots 90\%$ we plot the average accuracy of 100 models trained with that percentage of data and different train/test splits in Figure 3. In Figure 4 we can see the standard deviations of the accuracy of the models trained on each percentage of the data. This information allows us to answer one of our questions of interest. Increasing the percentage of data used to train the model only increases the

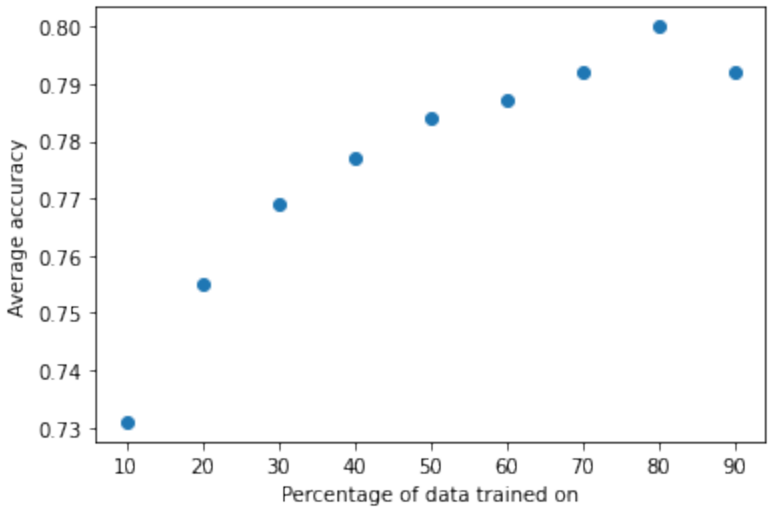



FIGURE 3. Average accuracy of 100 models trained at each percentage of data

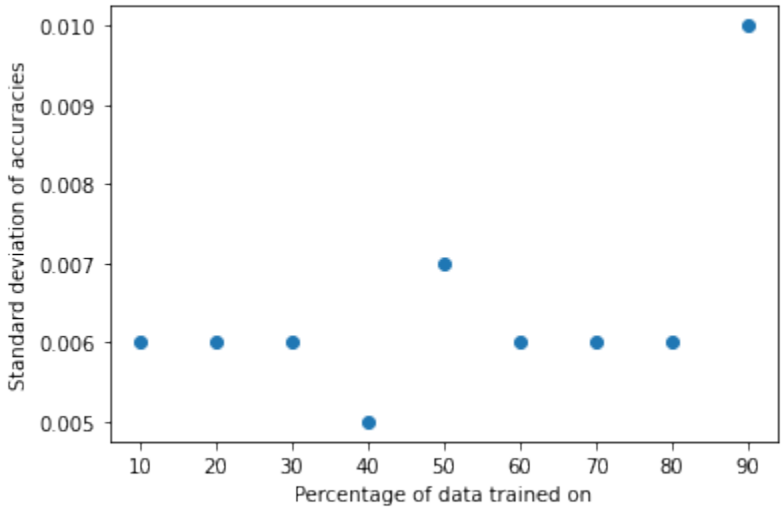



FIGURE 4. Standard deviation of the accuracy of the models at each percentage level

accuracy about 7% before the model begins to overfit and accuracy decreases. It also allows us to train models until we find an average one, at least according to the metric of accuracy and conduct further analysis on it.

4.3. **Other Methods.** Having investigated the ANN model thoroughly, we wanted to see how other model types would perform on out data. First we trained 10 SVM and averaged their accuracy so we could better compare to the average accuracy of the ANN model. We used a one versus one scheme since it is no more complicated than one versus all for a 3 class problem. We used a linearized kernel to keep things simple. On a 40/60 training/testing split we found a 65.85% average testing accuracy. This is significantly higher than our baseline, but still not nearly as good as our ANN model. (See Table 2 below for comparison of all model types)

We also considered a decision tree architecture. For this we used 5-fold cross validation to tune the hyperparameters for maximum depth and minimum samples split. We ran 50 models to find the average accuracy and hyper-parameters. On a 40/60 split we found that the average maximum

depth was 32 and the minimum samples split was 2. These gave us a 73.06% testing accuracy. This is better than the baseline or the SVM model but is still not as good as the ANN model.

| Model | Average Testing Accuracy |
|---|---|
| Baseline | 45.0% |
| SVM | 65.85% |
| Decision Tree | 73.06% |
| ANN | 77.69% |

TABLE 2. Testing Accuracy comparison with 40/60 split

With further confirmation that the ANN model is our best option, we now come back to our question of how our model performs differently on the different classes of data (wins, losses and draws). Using an 80/20 training testing split we trained models until we found one that had within one standard deviation of the average training accuracy: 80% $\pm$ .006. We then calculated the percentage of wins, losses and draws it correctly predicted from the test set. These percentages are in Table 3. Given the skew in the data with many more wins and losses than draws, it is not

| | Percentage Correctly Classified |
|---|---|
| Wins | 92.47% |
| Losses | 73.75% |
| Draws | 9.15% |

TABLE 3. Percentage classified correctly by label by average model with 80/20 split

unexpected that the model is far more accurate on these labels. Noting that, depending on the type of data (win, loss, or draw) being classified, the success of the model in accurate classification varies greatly. Particularly, the very poor accuracy in classifying draws stands out. It is tempting to assume that draws are inherently difficult to classify as their scenarios hang somewhere between win and loss. However, what stands out is that the proportion of draws in the overall data set is dramatically small. One could therefore question whether increasing the proportion of a certain class of data will increase the accuracy of the model in classifying that type of data. Four models were therefore trained using subsets of the overall data set so that each class made up different proportions. We explore this further by training more models on different training data with a similar number of wins, losses and draws. Table 4 shows the difference win/loss/draw splits used.

| | Data Set 0 | Data Set 1 | Data Set 2 | Data Set 3 |
|---|---|---|---|---|
| Wins | 6449 | 16635 | 25000 | 40000 |
| Losses | 6449 | 16635 | 16635 | 16635 |
| Draws | 6449 | 6449 | 6449 | 6449 |

TABLE 4. Data Set Compositions

Using these different data sets we trained models on an 80/20 split we trained models on these new data sets and the average results for each category of data are show in Figure 5.

It can be seen in Figure 5 that draws are in fact more accurately classified when they make up a larger proportion of the training and validation sets. While this accuracy, at 48.56% is still low compared to the overall accuracy, it is much higher than the 9.15% accuracy observed when using the entire set and is more accurate than if points were classified randomly. This accuracy appears

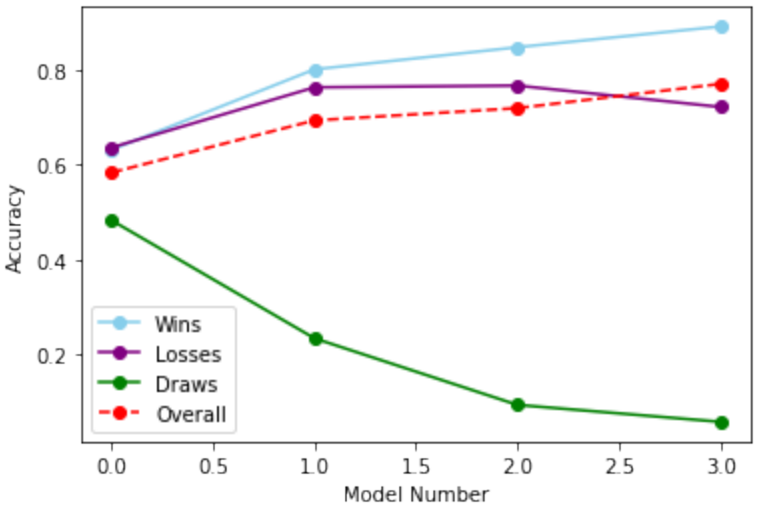



FIGURE 5. Accuracy of Classification Under Different Data Set Compositions

to drop exponentially as the proportion of draws in the training and validation sets decreases. As games are most likely to result in a win for player one, it is not surprising that a higher proportion of wins in the data sets leads to the model being more accurate overall. It is curious that optimal accuracy for classifying losses appears to be around 73%, occurring when losses and wins take up a roughly equal share in the data set.

4.4. **Mirrored and Forced Data Points.** We investigated how our average ANN model would perform on the "forced" data set we made. After training a model on the original data set using an 80/20 split we then tested it on the "new" data. This gave a 42.66% accuracy for the 63 data points. This is much worse than the around 80% accuracy it was getting on the original testing set. This may be because these scenarios are inherently more complicated or different than the non-forced scenarios. It could also be due to the win/loss/draw split being different from the original data. A larger number of a category of data point is positively correlated with that category being accurately classified. So, it is possible draws and losses will be more accurately classified if more draws and losses are included in the training set. The data set we made has a 25/22/16 win/loss/draw split which is 40% wins 35% losses and 25% draws while as we have discussed above the original data has 65%/24%/9%. Repeating the analysis of breaking the accuracy down for each label for this type of data is an opportunity for further investigation.

Considering a single board of Connect-4 and its mirror image across the y-axis, these two boards should, regarding game results, come to the same ending. It is important in evaluating the effectiveness of a model to observe whether the model classifies mirror images the same way. Ten artificial neural networks with identical parameters were trained using an 80/20 training/testing split. These models were then used to classify mirror images of all data points in the set. Through the ten models, it was observed that, on average, mirrored pairs were given the same classification about 80% of the time. While this shows that the model strongly takes into account the symmetry in the data, it does not perfectly recognize it. Essentially, mirror images are likely to be given the same classification, but the softmax function does not assign identical values to mirrored pairs.

## 5. Discussion and Conclusions

In conclusion, we found that the ANN model with 2 hidden layers outperformed other ANN architectures, SVM models and decision trees for this data set. None of the models perform particularly well but all outperform our baseline. This problem is inherenently difficult as the data

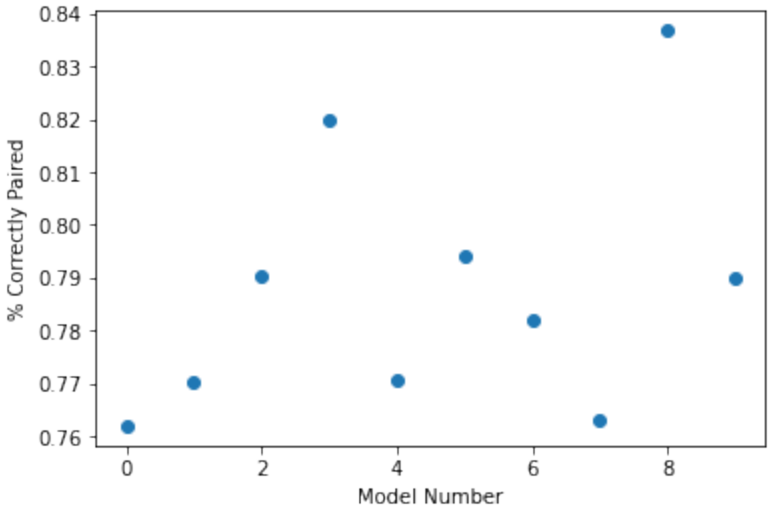



FIGURE 6. Percentage of Mirrored Data Pairs Classified Correctly

only represents the first 8 moves of a possible 42 and tries to predict the outcome. With that being said the amount of data trained on does not impact the accuracy as much as one might expect. We saw only a 7% increase in accuracy between training on 10% and 90%.

We found that accuracy for our model is greatly dependent on the label of the data point and that when we change the composition of the data set it effects the accuracy on the different classes of data. We found that our ANN model recognizes the symmetry of the data fairly well while it does not perform well on the new "forced" scenarios.

In future study, one could investigate the architecture of the ANN model further and tune the number of nodes as well as the number of hidden layers. One could also go further into investigating forced scenarios and create a larger "forced" data set as well as look into the details of how the label of the data effects the accuracy for that set.

## REFERENCES

[All88] Louis Victor Allis. A knowledge-based approach of connect-four. *J. Int. Comput. Games Assoc.*, 11(4):165, 1988.

[Inc20] OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences. `https://oeis.org/A212693`, 2020.

[Tro95] John Tromp. UCI Machine Learning Repository Connect-4 Data Set, 1995.