Predicting outcomes of the NCAA March Madness with an Artificial Neural Network

By Mohammad Azim

**Abstract:**

The NCAA march madness brackets are picked by millions of American across the country every year as a popular low stake gambling entertainment. However, with 9.2 quintillion possibilities, and potential for upsets, these picks provide a challenge for the most sophisticated algorithms. Neural networks have had great successes in a large domain of problems, therefore it is an ideal candidate for this difficult problem. This paper is interested in its ability of neural networks to predict the march madness brackets, and how to format the neural networks to get the best results.

**Introduction:**

The main aim of this paper is to create a model to predict the NCAA Men’s division one basketball tournament (March Madness) brackets, and to do so with a Data Mining approach. In almost all sporting matches the fans have a vested interest in trying to estimate the winners beforehand, however in this tournament that desire supersedes all other sporting evert. In recent years, more than 9 billion dollars is spent per year in gambling to pick the march madness brackets. This is not just in gambling institutions but mainly in office work places. Despite all this attention, it is hard to find a publication that implements data mining to this problem. A search in google scholar shows mainly statistical approaches or papers focusing on another aspect of the game. Most related methods for brackets are Formulas like RPI Rating percent index, BPI Basketball Power Index (an adjustment to RPI), the team seeds (ranking each team gets that determines the order of the games), or FiveThirtyEight’s model (which is based on an aggregation of many polls weighted by accuracy with a bit of adjustments).

This paper intends to satisfy this need for a data mining predictive model for determining the brackets of march madness. Beyond just classifying which team will win, we’ll try to gauge how much a team will win by. This approach has benefits in both the building and application of the model, which will be explained in a later section. The algorithm of choice will be an artificial neural network (ANN), which has been used to solve numerous problems of varying difficulty with great results, and performed better than other data mining techniques in the regular season of NCAA basketball (Shi, Moorthy, Zimmermann, 2013).

**Background:**

March Madness is a tournament at the end of every regular season of college basketball. Each game is a one game elimination matches colligate sporting event, and it gives mid-level colleges a chance to compete and even win against the best. On top of being the most popular colligate sporting event internationally, it is also the most popular gambling event. With 9.2 quintillion possible outcomes, and potential for upsets, predicting the winners is difficult task. To understand this analysis, a fundamental understanding of the tournament is necessary.

The tournament starts with 68 teams, 31 of the 68 teams are selected by division 1 champions. From the Ivy League, which comprises of the first few colleges that formed the NCAA, one team is chosen. The remaining 36 is selected by the selection committee on numerous complicated factors, but the goals of all those factors is to pick the best teams. The first 4 games are against the 4 lowest automatic picked (division 1) teams vs the 4 lowest selected teams, which aren’t necessarily the 8 lowest teams overall. After 4 teams are eliminated a seeding process is done for the remaining 64.

To schedule the games, teams have to be split into 4 regions and in each region they are ranked best to worst from 1 to 16, known as the seed numbers. The process is complicated but the goal is to divide the best teams to the region closest to them, however they also try to not have the team’s play in home court, and they try to match teams that don’t usually play against each other in the regular season. They are split into 4 regions north, south, mid-west, west, which doesn’t align well with each college’s actual location. Then they are matched against each other with team 16 vs team 1, then 15 vs 2, and so on.

At this point fans select the brackets, which is a prediction of which teams will win and lose in all the rounds. The first week the first 32 games are played and next 16 game is played. The first 32 games eliminate 32 teams next 16 eliminated 16 more until there is one winner. Seeds are a popular method for determining the better team, along with performances of these teams in the regular season. Vast amount of data is collected gauging everything from offense, defense, coach’s ability, the team’s historical performance, etc. The dataset used in this analysis had 96 such attributes, with the scores of each team, the dependent variable, being of interest.

**Related Work:**

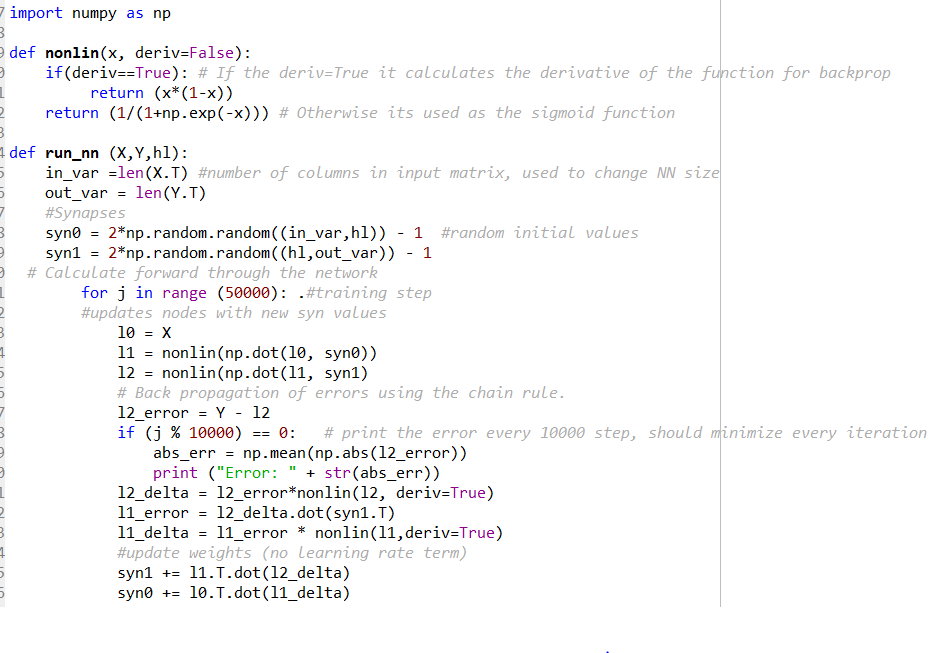
The data journalism site Fivethirtyeight has their predictions out every year, and their initial model for 2015 (they refine their model as the tournament is played) results had 70% of the brackets correctly or 44/63 correct (McCann, Reuben Fischer-Baum and Allison, 2015). Their work is done by aggregating many other ranking systems and weighting it by accuracy, and this is usually better than average of the individual, in most years. This accuracy should be taken as a baseline for the statistical methods accuracy, and hopefully a data mining approach can perform better.

Shi, Moorthy, Zimmermann (2013) have tried similar predictive task in determining the winners of the NCAA Basketball in the regular season, and they have some interesting results that can guide this analysis. Their modeling was performed all in WEKA using its default implementation, and they evaluated C4.5, Ripper, Multi-Layer Perceptron same as an ANN, Naïve Bayes, and Random Forest. Of these they found Multi-Layer Perceptron to have the best results, they found that explicitly creating features that displays the difference between two teams doesn’t improve predictability, and they also claim a celling of predictability of around 74%, and they justify it by giving examples of numerous papers predicting sporting events with similar accuracy.

**Experiment**

There are a few designee decisions in creating a ANN to make this model transparent to others interesting in replicating the results, the below python code displays the exact implementation of the ANN used for this analysis.

**Figure 1:** Neural Network Algorithm in Python



The model was specified to predict the score difference. This is the score of team 1 minus the score of team 2. Another option was to set the winners as 1 and losers as 0 and try to make this a classification problem, however this approach has advantages in both the building and application of the model. If a game in which one team beat its opponent by 13 points vs another game in which the winner one by 1 point an ANN will surely be able to weight its synapsis better with the score difference as the Y variable. Information is lost when the severity of the win is reduced. Also, in betting application like the Las Vegas spread, the payout for betting on a team is weighted by the demand to bet on that team. So a gauge of how sever the win or loss is by will aid in betting wisely. Transformations had to be made to the data set to run the algorithm. Data set had team 1 was always the winning, if not randomized, the model would always select team 1 slot as the winner. A random binary generator assigned each row either a 0 or 1. In the rows assigned 1s the values of team 1 and 2 were switched. All the variables were set to min\_max scale to make all the data to be from 0 to 1 (formula below). ANN’s are very sensitive to scale, and min\_max is standard practice. For all of the below tests the training years were 2002 to 2004, 2005 &2008, 20010 to 2012, and 2014. The test set was 2005, 2009, and 2013. Finally, the 2015 data would be the one to evaluate against other models.

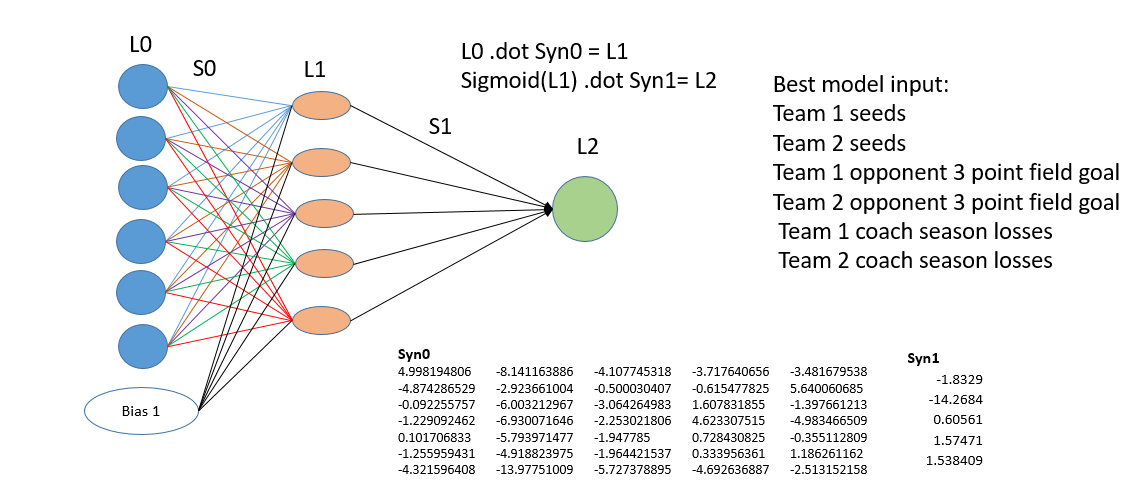
Min\_Max Scale = X(i)-Min(X)/(Max(X)-Min(X))

There were two attempts to choose the variables through brute force methods. The first was an attempt to pick the best 4 variables of 64 variables. The main objective was to lower the mean absolute error between the actual Y variable and the predicted Y or Y-hat variable. And to try an iteration of that model with 6 hidden nodes then 5, then 4, then 3, then 2. A few heuristics we’re added to the brute force search to minimize the search space. One of the rules were to combine the input variables for the two teams. Each attribute had a corresponding attribute for the other team. For example, if team1\_FG3pt (average 3 point shots per game for the team1) was selected to be in the model, the corresponding team2\_FG3pt would be selected as well. The logic behind it is if a variable is important in predicting the performance of a team, it should be just as important to predict the performance of the other team to get the score difference. This decreased the search space from 64 choose 4, which is over 635,000 to 32 choose 2, or 496 models. The second was to eliminate variables, due to more than 20 missing data points, or if the variable was redundant. For example, if there was a variable for offensive efficiency then another for adjusted offensive efficiency I would eliminate the former.

The last model was chosen in the same fashion as the above except 32 choose 3 or 4960 different choices. The time frame increases further since more hidden nodes were iterated though, from 6… 3 to 9… 3. Since all available outputs would’ve taken too long to search though a subset was searched though, with the goal of beating the results achieved in the 4-variable input, which had a mean absolute error rate of .0833. The best results achieved in the subset that was viewed had a mean absolute error that was slightly less than .0830. This result was after 10 hours of search of over a 1000 different variable inputs.

**Results:**

**Figure 2:** Depiction of the Final Model



The final model had an accuracy of close to 70% in the training set, and an accuracy of close to 70% (or 135/195 games) in the overall test set. The accuracy of the model of its training and test set being close to each other is an odd result. This is because the ANN optimized for the mean absolute error instead of accuracy. When you compare the mean absolute error of the test set which is .0954 to the training .0830, the result makes more sense. Accuracy in the prior section is different than the accuracy of the brackets. In the prior context, each game is independent but when selecting brackets the error from each of the previous rounds compound. For 2015, the brackets selected by the model had 41/63 correct or 65% correct.

**Conclusion:**

In conclusion, the results of this experiment were not the best, and compared worse than the typical statistical model. Where FiveThirtyEight had 44/63 games the ANN had 41/63. It may be why the statistical approaches are dominant in the field, but only one of the many data mining algorithm was attempted and still there was a lot of improvements that could’ve made. For future research on this topic it would be interesting to get granular data and try to use Deep Learning to see if that can improve the results. As of now most of the works seems to plateau at a similar accuracy. This may be due to everyone using a similar data set that doesn’t have the information necessary for this task. With deep learning, we might be able to extract the necessary information in the second or third hidden layer if we combine it with more granular data, such as the attributes of each player in the games.

**Works Cited:**

Shi, Zifan, Sruthi Moorthy, and Albrecht Zimmermann. "Predicting NCAAB match outcomes using ML techniques–some results and lessons learned." *ECML/PKDD 2013 Workshop on Machine Learning and Data Mining for Sports Analytics, Prague, Czech Republic*. 2013.

McCann, Reuben Fischer-Baum and Allison. "How FiveThirtyEight’s NCAA Tournament Forecasts Did." *FiveThirtyEight*. N.p., 15 Mar. 2016. Web. 1 Dec. 2016.<http://fivethirtyeight.com/features/how-fivethirtyeights-ncaa-tournament-forecasts-did/>