Using Bayesian Networks to Predict March Madness Brackets and Results

Kenneth Stewart II
Rochester Institute of Technology
kis7255@g.rit.edu

Marko F. Galesic
Rochester Institute of Technology
mfg1071@g.rit.edu

ABSTRACT

Generally, we will use standard statistical basketball indices for team performance to create a model that can predict whether a team will win March Madness given the season statistics and tournament seeds. We intend to develop an algorithm to parse the initial dataset and derive features from the data to create a table of statistical indices for each team in a the given season. The seasons are then aggregated together to come up with an overall performance history of each team. The statistical indices will then be used as input to a Baysian Network to create probabilities for the winner of each game in the tournament until a tournament winner is chosen.

1. BAYESIAN NETWORKS LITERATURE REVIEW

Bayesian (Belief) Networks is a probabilistic graphical model used to model knowledge about an uncertain domain. Each graph node represents a random variable and each edge between nodes represents the probabilistic dependencies among the corresponding nodes[1]. The edges are often estimated using various statistical and numerical methods such as observations made by the Chain Rule (discussed later).[2, 3] Bayesian Networks are structured as directed acyclic graphs (DAGs) which enables an intuitive, rigorous model that effectively represents and facilitates the computation of the joint probability distribution over a set of random variables[1]. These graphs are structured as two sets, one for nodes and one for directed edges. An edge from node 'i' to node 'j' means the value taken by node 'j' depends on the value of node 'i'. 'i' is said to *influence* node 'j', and one of the *parents* of node 'j'. 'j' is then a *child* of node 'i'[1]. Nodes that represent variables are evidence nodes, otherwise they are latent or hidden nodes. We've reviewed literature that goes through various techniques available in WEKA, the data mining and learning tool, for construction and learning Bayesian Networks.[4]

2. DATA PREPROCESSING

2.1 Data Cleaning

Since the data primarily consists of ordinal or nominal attributes and ranges are given, we can use clustering techniques to validate the data and find relationships. Google refine will be used to handle this task. The source documented three (3) records that were added in order to create a complete the dataset that was representative of all the current teams eligible for the NCAA Division I tournament in 2014. We chose to remove these entries from the set as they may compromise our model. These instances are outliers since their values for certain important attributes occur in no other instances in the dataset.

2.2 Stratified Sampling

The dataset is not large (several megabytes), but we can still split on seasons to make processing the data faster. Instead of processing each game of every season. A subset of the first 'n' games for each team for each season will be used. This effectively reduces the data by a constant factor each time 'n' decreases. This also effectively helps reduce the time for pre-processing while still accurately representing the model as long a significant amount of games are used.

2.3 Aggregation \Feature Creation

We plan to use the following measures and indices to predict teams' chances of winning the NCAA Division I championship: Rating Percentage Index (RPI), Average Margin of Victory, Pythagorean Expectation, and Close Won Games. Each index provides some unique information on a team. These features are then used to construct the Bayesian Network to model the joint probability distribution of the variables. These features will be calculated per team over all seasons giving us a total of 356 records. The records then become input to a Bayesian Network to generate probabilities to calculate the winner of each game starting from the first round all the way to the championship game to determine a winner. Following is the rationale behind each metric.

2.3.1 Rating Percentage Index (RPI)

While we've read that RPI is not a good indicator of team performance since it does not include a team's defensive fitness[5], several contestants for the March Machine Learning Madness competition on Kaggle.com have mentioned that they are using RPI in their algorithms.[6, 7, 8] It is a common indicator of team performance and is essentially

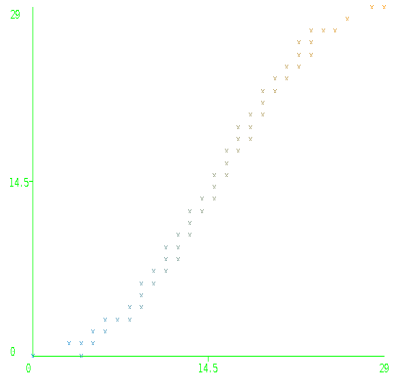

Figure 1: A scatterplot of Average Margin of Victory(X) vs. Pythagorean Expectation(Y)

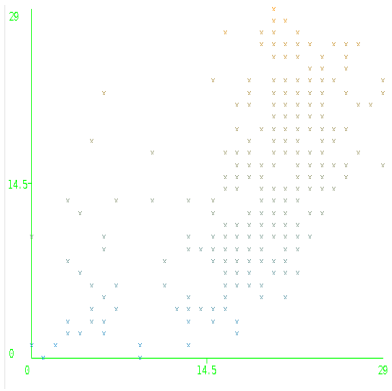

Figure 2: A scatterplot of Close Won Games(X) vs. Pythagorean Expectation(Y)

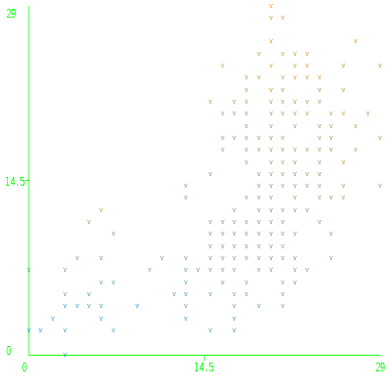

Figure 3: A scatterplot of Close Won Games(X) vs. Pythagorean Expectation(Y)

a weighted average of a team's winnings along with its opponents winnings. The formula is described below for a Team A:

$$RPI_{Team_A} = WP * 0.25 + OWP * 0.50 + OOWP * 0.25 \quad (1)$$

$$OWP_{Team_A} = \{OWP_1, OWP_2, ..., OWP_n\} \quad (2)$$

$$OWP(Team_A) = \frac{\sum_{i=1}^{|OWP_{Team_A}|} OWP_i}{|OWP_{Team_A}|} \quad (3)$$

In order to get the the OOWP, we use Eq. 3 for each opponent that is an opponent of Team A except we do not divide by the total opponents for each opponent but for all opponents of opponents.

2.3.2 *Average Margin of Victory*

Average Margin of Victory will give us information on how much, on average, a team performed better than another team. It is a relative measure of team fitness or "goodness" that is more specific than RPI. It also has an interesting relationship with pythagorean expectation (Fig. 1).

$$AverageMarginOfVictory_{Team_A} = \frac{\sum_{g=1}^{n} tpf_g - tpa_g}{totalgames} \quad (4)$$

tpf(g) would be total points a team scored in a game (total points *for*), and tpa would be the number of points the opposing team scored (total points *against*). g would be a game in set of games with the cardinality n.

2.3.3 *Pythagorean Expectation*

This index was developed by Daryl Morey, a sports executive, who had much success in turning around a losing team into a successful team.[9] Pythagorean expectation seems to have its roots in the Pythagorean theorem and may be rooted better theoretically than RPI. RPI and Pythagorean Expectation are similar and are roughly correlated. The equation is described below:

$$PE_{Team_A} = \frac{tpf^{13.91}}{tpf^{13.91} + tpa^{13.91}} \quad (5)$$

2.3.4 *Close Won Games*

We will consider close won games to be a game in which the margin of victory for a given team is 7 points. Close Won Games are then a summation of all matchups between a team and its opponents where the stated condition is met. This metric does not seem to be correlated with either RPI, Pythagorean expectation, or Average Margin of Victory, meaning that we'd get more information from close won games than from just RPI, Pythagorean expectation, or Average Margin of Victory. Fig. 2 and 3 show for Close Won Games versus RPI or Pythagorean Expectation that there is little correlation between these attributes. The graph of Close Won Games versus Average Margin of Victory (not shown) is similar. This should not be a surprise since Average Margin of Victory has a correlation with the Pythagorean Expectation.

3. STRUCTURE OF TRAINING \TEST SETS

3.1 Hight Level Structure of Training and Test Sets

Out of the eighteen (18) seasons given in the dataset, 11 of those seasons will be used for training data, the remaining 7 will be used for the test data (a 60/40 split). Our model will represent the regular season and the Division 1 Championship. Since teams come and go, it would be unwise

to make specific models off teams. A larger practical concern for a more granular split (e.g. if we modeled individual teams) would be that we would end up having to possibly train 350 different models. On the other hand, we chose not to take into account schedule strength or the strength of a teams' conference as the NCAA ranking system does. This choice was made to reduce complexity.

3.2 Use of Stratified Sampling

In order to get variability in the dataset, stratified sampling will be used to sample the 18 seasons worth of data we are given to create the training and test sets. We now describe our specific method of stratified sampling for this dataset. Given the seasons are in chronological order, the season are separated into three (3) groups that spans six (6) seasons each. This is done to help prevent bias in the model as a contemporary instance of a team 'A' may play differently than past a team instance of 'A' due to changes in rules, medicine, training, and strategy. These variations must be captured during the training process or the classification of a winner may be incorrectly chosen based on data that no longer accurately represents the model. Lastly, four (4) samples from two (2) groups will be selected and three (3) samples from the remaining group will be selected as training data (11 seasons total). The rest of the data will be used for testing.

3.3 Discretizing Sample Data to Make Constructing the Bayes Network Easier

The data will then be binned to discretize the continuous variables to better fit how our model (a Bayesian Network) operates. Bayesian Networks generally operate on discrete variables. Continuous variables can be modelled but must still be discretized.[10] Discretization will let us create tables of local probability distributions for a particular node (a node represents a feature or attribute) given the node's parents. We plan on using a number of bins to discretize the distribution of the data into well defined classes and possibly vary binning per continuous attribute to improve performance and perhaps avoid bias. Since most of the data has a normal distribution, equal width binning is the binning method of choice.

4. BAYESIAN NETWORK

4.1 Constructing the Network

Constructing a Bayesian Network is a multistep process with many variations depending on the application domain. We will be creating a predictive model and have focused our attention on algorithms that allow us to do so; however, the first step in the process is always to identify the variables of interest that the network is to model. We have outlined the variables (features) we are to use, statistical indices, in the previous sections.

The next step in the process is to create the actual structure of the network (the DAG). There are various ways to do this. One of the two methods in [1] outlined an algorithm that takes a set of variables

$$X = \{x_1, x_2, ..., x_n\} \tag{6}$$

in a particular order and computes the network based on the observations of the Chain Rule of Probability which says

$$p(x) = \prod_{i=1}^{n} p(x_i \mid x_1, ..., x_{i-1}) \tag{7}$$

from [1]. This is saying that the probability of a variable in the set X is equal to the product of each variable given all the variables that come before it. This identifies conditionally independent nodes (nodes with no parents) in the network as well as the connections between the conditionally dependent nodes. Thus, if a variable 'a' is conditionally dependent on a set of variables B then 'a' will have all variables in B as parents. This approach is the most straightforward; however, it has serious limitations. The ordering of variables is very sensitive and improper order can cause incorrect causal relationships to be formed which can lead to poor results.

The second approach is based on two observations: 1. People are able to readily assert causal relationships in the variables. 2. Causal relationships usually correspond to assertions of conditional dependence [1]. Using these observations the person constructing the network can explicitly create nodes and connections. This usually preserves the observations from the Chain rule of Probability [1].

The last step in the algorithm is to compute the local conditional probability distributions at each node in the graph. This is done by calculating, for each node, a single distribution for every combination of the configuration of the node's parents. The local conditional probability tables for each node will be calculated from data. This is accomplished by parsing the binned data, locating positives classes (wins), and determining the probability based on the number of positive observations divided by the number of observations. This is done for each combination of the binned attributes calculated after features have been extracted.

4.2 Probabilistic Inference

Probabilistic inference is the process of calculating a probability of interest from the model. There are two types of inference, approximate and exact. Exact inference is an NP-Hard problem. Due to this nature we chose to use approximate inference as it is less computationally expensive to compute. We are currently looking into algorithms that will allow efficient computation of an approximate value to used

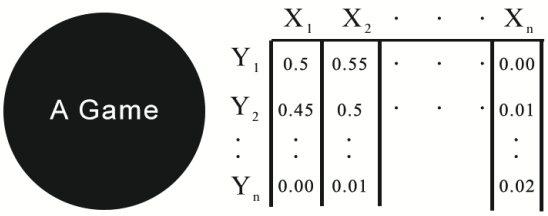

Figure 4: A scatterplot of Close Won Games(X) vs. Pythagorean Expectation(Y)

as the probability estimate. [1] has pointed us to additional resources which may give us a better understanding of inference and the algorithms to compute probabilities of random variables.

5. NEXT STEPS AND THOUGHTS

5.1 A Representation of a Game Node

Our initial thought was the represent the March Madness tournament as a collection of game nodes. Each game would connect to a next round game (until the final game). In the initial 32 games, for example, one of the second round nodes might look like Fig. 4. All of the possible victors from round one in, say, game one would be on the 'top' of a conditional probability table and be represented by

$$X_i \epsilon \{X_1, X_2, ..., X_n\} \tag{8}$$

All of the victors from game two in the same round would be on the 'side' of the same table, represented by

$$Y_i \epsilon \{Y_1, Y_2, ..., Y_n\} \tag{9}$$

Where 'i' would be either that victor's seed, or some performance metric; 1 being the lowest score and n being the highest.The table would then represent all possible combinations from two games of victor matchups. Appropriately, the probability of any victor facing another victor with a higher (lower) seed would be less (more) likely to win in that matchup. This model, of representing each game as its own set of probabilities, lends itself to scaling each victor's winning probability - giving us flexibility when predicting the winner based at least in part on any heuristics.

5.2 "Cinderella" and "Fatigue" Factors

We may weight each outcome with a "Cinderella" factor which would be a distribution we'd create based on low seed teams beating higher seed teams. This distribution could be put into either the table or as a scalar going in. We may also use a function to represent how a given team would perform given the number of consecutive days said team has played. The function, we call it Fatigue, would scale a team's probability of winning against any other team down the more consecutive games they have played. In March Madness, for example, the First Four are played almost a week in advance of the first round 32 games, but then teams that have won in that first round of the first 32 games would play another team within two days. Fatigue would be dependent on a team's baseline performance, a higher performing team would likely get fatigued slower than a lower performing team. A high seeded and low seeded team's fatigue graph might look like Fig. 5.

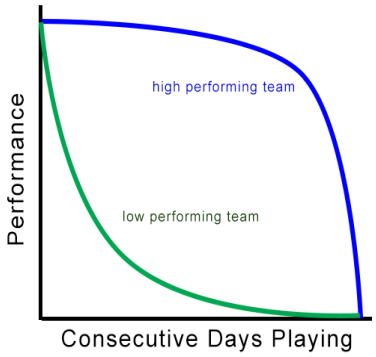

Figure 5: A scatterplot of Close Won Games(X) vs. Pythagorean Expectation(Y)

5.3 Probabilistic Inference Techniques

One of our next goals is to really understand how to calculate probabilities from the Bayesian Network. We have encountered sources that lead to more descriptive explanations of probabilistic inference in Bayesian Networks. The next step is to review these sources and see what techniques fit our application domain well and make a decision of what to use.[11, 12]

5.4 Learning Model

Another short-term goal is to determine how the network is fine-tuned and the learning model is used. Both the parameters and probabilities of a network can be fine-tuned [1]. We are working to determine how.

6. REFERENCES

[1] D. Heckerman, "A Tutorial On Learning With Bayesian Networks," *Microsoft Research Advanced Technology Division Technical Report*, March 1995. http://tinyurl.com/lx7u8tt;Retrieved Mar. 2, 2014.

[2] S. Wagner, "A Bayesian Network Approach To Assess and Predict Software Quality Using Activity-Based Quality Models," *Proceedings of International Conference on Predictor Models in Software Engineering*, vol. 5, no. 6, p. 9, 2009. DOI=10.1145/1540438.1540447http://tinyurl.com/pgracp9.

[3] V. K. Pang-Ning Tang, Michael Steinbach, *Introduction to Data Mining*. Addison-Wesley, 2005.

[4] R. R. Bouckaert, "Bayesian Network Classifiers In Weka For Version 3-5-7," *Dept. of Computer Science at The University of Waikato*, May 2008. http://tinyurl.com/od8fu7r;Retrieved Mar. 2, 2014.

[5] J. Mozell, "Bracket Prediction: Highlighting Historical Indicators Of March Madness Success," *The Bleacher Report*, pp. 1–12, 2013. http://tinyurl.com/p6bwyxh;Retrieved Feb. 20, 2014.

[6] J. Sonas, "About the RPI Benchmark," *Kaggle.com*, 2013. http://tinyurl.com/nofpkgy;Retrieved March 6, 2014.

[7] C. M, "Local Scoring," *Kaggle.com*, 2014. http://tinyurl.com/pndjuew;Retrieved March 6, 2014.

[8] Covalytics, "Upset Identification," *Kaggle.com*, 2014. http://tinyurl.com/otgty7z;Retrieved March 6, 2014.

[9] D. Feschuk, "Morey'S Moneyball Approach Paying Off," *The Star*, 2013. http://tinyurl.com/pgw5sjv;Retrieved March 6, 2014.

[10] A. S. Barry R. Cobb, Rafael Rumi, "Bayesian Network Models With Discrete and Continuous Variables," tech. rep., Virginia Military Institute Department of Statistics and Department of Applied Mathematics University of Almeria, March 2014. http://tinyurl.com/nx3ayda.

[11] C. Ruiz, "Illustration Of the K2 Algorithm For Learning Bayes Net Structures," *Department of Computer Science, WPI*, March 2014. http://tinyurl.com/kkcm765;Retrived Mar. 2, 2014.

[12] F. F. Fabrizio Ruggeri, Ron Kenett, *Bayesian Networks In Encyclopedia Of Statistics In Quality and Reliability*.