Include your name and sign the honor code:

On my honor as a UNT student, I, Thomas Lee have neither given nor received unauthorized assistance on this work.

A Neural Network MLB Game Predictor

Thomas Lee, University of North Texas, Department of Computer Science

Thomaslee2@my.unt.edu

**Abstract**

*Prior research has shown that predicting the winner of a baseball game is a difficult task with the most accurate methods correctly predicting a winner about 60% of the time. I believe the performance of other teams can be attributed to their feature selection or the algorithm(s) they chose. I believe that my neural network and the features I selected enabled me to more accurately predict the winner of games. I propose that using individual player statistics based on season and previous five game performance as an input vector into a neural network. I believe that using predicted player statistics as input to a game-outcome predictor will perform worse than the game-outcome predictor that uses actual player statistics, but I believe it will still perform better than the other papers examined.*

**Introduction**

What is the problem you are addressing?

* I believe that baseball player performance and baseball game outcomes can be accurately predicted using neural networks.

What claim do you hope to make when your work is complete?

* I hope to predict particular player statistics (strikeouts, hits, at-bats, etc) perfectly at least 70% of the time. For statistics that occur infrequently, like homeruns and triples, I will calculate a “non-zero” accuracy percentage as well as traditional accuracy. This measurement will show often my algorithm perfectly predicts a value above zero over the total number of times the actual value is above zero. I will use these predicted player statistics as part of the input vector to a neural network to predict a winner of a game. I hope to predict the winner of the game correctly 90% of the time.

Who cares?

* Las Vegas odds makers
* Baseball general managers
* Baseball fans

Why do they care?

* In 2016, $1 Billion was bet on Major League Baseball.
* For general managers, job performance is often based on wins per dollar amount.

What is lacking from previous attempts to solve the problem (in particular, mention weakness that your paper is going to overcome)?

* There have been a few attempts at using machine learning to predict game outcomes. Some of these attempts use what, I think, are irrelevant features such as previous season team record. While other attempts use less suitable machine learning algorithms or too small datasets.

What do you know about previous attempts to solve it? List key publications that you are aware of.

* [Predicting the Major League Baseball Season by Randy Jia, Chris Wong, and David Zeng](http://cs229.stanford.edu/proj2013/JiaWongZeng-PredictingTheMajorLeagueBaseballSeason.pdf)
* [Applying Machine Learning to MLB Prediction & Analysis by Gregory Donaker](http://cs229.stanford.edu/proj2005/Donaker-MLBPredictionAndAnalysis.pdf)

What have previous results been?

* Less than 60% accuracy on predicting the winner of a baseball game.
* When broken down by month, accuracy rises towards the end of the season, this is due to the erratic swings in statistics during the early months of the season.

The key contributions of this work are:

* The use of a neural network to predict winners as suggested by previous papers
* Including a measurement of “streakiness” – how well a player has done recently – as opposed to only season-long statistics.

**Background/Preliminaries, Definitions, Data, Examples, Key issues/problems, Problem framework**

Definitions

* At-Bat (AB) – a plate appearance of a particular batter.
* Strikeout (SO) - when a batter is retired from his at-bat after three called strikes or missed swings.
* Double – a hit that results in the batter reaching second base safely
* Triple – a hit that results in the batter reaching third base safely
* Homerun (HR) – a hit that leave the playing field in fair territory, or a hit that results in the player running all the bases and returning to home safely.
* Walk – a player if allowed to move to first base after four pitches outside the strike zone

My dataset was derived from the data collected by RetroSheet.org. It contains plate appearance data for every AB for each player from 1921 to 2016.

[Link to Data](http://www.retrosheet.org/game.htm)

I wrote various python programs to transform this data into two datasets. One for player data, and one for game data. One example from the player dataset will include game measurements such as temperature and day of the week, player statistics such as average strikeouts per game for the season and average strikeouts per game for the past five games, and includes the player’s performance for that particular game which is removed before training and used as the actual value vector. The following are the preprocessing steps take to prepare the data.

* First *BGAME.EXE***,** *BEVENT.EXE*, and compressed files of all the data from 1921 to 2016 were downloaded.
* **Filterprog.bat** was used to compile data with the appropriate contents from the downloaded data. The results were saved as text files.
* Then **combineTraining.py** was used to combine all examples into two files. One containing every event (at-bat) statistic, and one containing every game statistic.
* Then the event file was run through **getAB.py**. This program removed irrelevant entries from the file such as an at-bat that never concludes because a runner is caught stealing and is the third out of the inning.
* Then both files (game and event) are run through **getStats.py**. This file calculates the season and “last 5 game” statistics for every player for every year and adds them to each event entry. Each of these stats is also added to the corresponding “game” example for the appropriate starting 18 batters of that game. The output of the event file is split into two parts to save processing, but they are eventually recombined.
* Then the game file is run through **getRecord.py**. This program calculates the current record of each team for any particular game in any particular season. This information is added to the appropriate game data example.

An example from the game dataset would include the same game measurements as with the player dataset, but it would also include the player statistics of all 18 starting batters from the player dataset. The example will also include the current win-loss record of each team.

For feature engineering and feature selection, I split the data up with 10 seasons in the training set, one season as a validation set, and one season as the test set. I repeated this split, adding the validation set of the previous iteration to the training set, the test set becomes the validation set, and the new test set becomes the season after the previous iterations test set.

The final algorithm is performed on a subset of the data. I had to shrink my dataset considerably to compensate for a lack of resources. The training set includes the 1979 season to the 1999 season. The validation set includes the 2000 season to the 2014 season, and the test set includes the 2015 and 2016 seasons. For all steps, after initial training and evaluation on the validation set, the validation set is added to the training set, the algorithm is retrained, and then a final evaluation (prediction step) is performed on the test dataset.

**Approach / Methods**

To train the classifier, the dataset is first, transformed into a vector. This means that features with nominal values are transformed into a numeric representation. For example, if a feature has seven possible values, then for every example in the dataset, seven floating-point values will be added. One nominal value will be represented as [1,0,0,0,0,0,0] and another as [0,1,0,0,0,0,0] and so on. These vector representations are fed into scikit learns MLPClassifier method.

I’ve used several neural networks in my program. One for predicting each player statistics (eight total), one for predicting game scores, and one for predicting the winner of a game. Many iterations of training and validation were used to refine the number and density of the hidden layers of each of the neural networks. Generally, the player statistic NNs performed best with two hidden layers. Most of these hidden layers contained *d, 2d, or 2log2(d)* where *d* is the dimensionality of the input vector.

I wanted to use neural networks because I believe that there are complex relationships between team performance and individual player performance that cannot be accurately approximated by other machine learning algorithms. I think the flexibility and potential complexity of neural networks can capture and approximate this relationship.

My different neural networks use various hidden layer attributes, but the rest of the attributes are the default values of the classifier seen [here](http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html).

I also used the mean absolute value function provided by scikit learn as a measurement of preciseness for player attributes and predicted score.

For predicted score, I broke down predictions into categories: perfect prediction, within one run, within two runs, and more than two runs, and calculated the percentage breakdown of these categories.

**Experiments**

**Design / Methods**

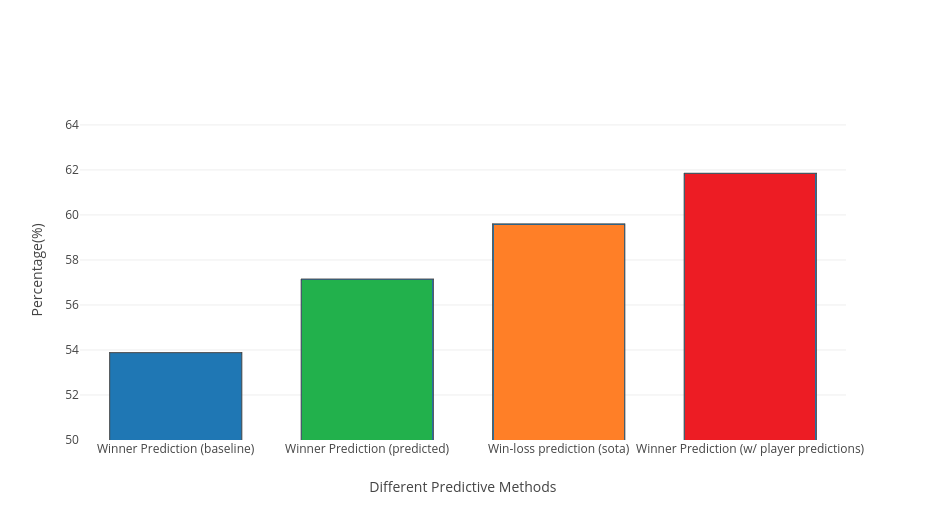
I generate two python files to calculate baseline predictions - **getPlayerBaseline.py** and **getGameBaseline.py**. For player statistics, I used the “last 5 game” average of each statistic as the predicted value. For game statistics, I used each team’s average run total as the baseline score, and for winner predictions I picked always the “home” team. This home team baseline proved to be more accurate than more complex baseline methods.

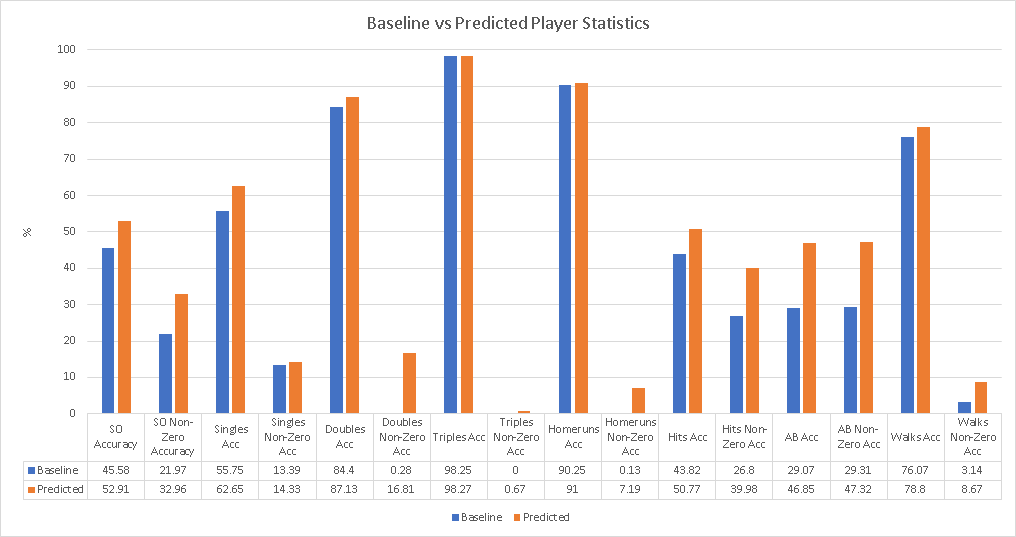
During the actual experimentation, I broke the problem down into three parts – predicting player performance, predicting game outcomes, and finally, using predicted player performance to influence predicted game outcomes. My main program, **learn.py**, was chopped up in various way throughout the experiment often having irrelevant pieces commented out. When predicting player performance, I commented out the section of the code responsible for computing game outcomes, and when predicting game outcomes, the player predictors were commented out. Only when I moved on to the third stage of my experiment, did I use the entire **learn.py** as it was intended. Also, there are certain helper methods that are used for certain steps of the experiment. **featureExtract.py** is a file that removes certain irrelevant-for-training data from dataset examples. There is one method for when I’m just predicting game outcomes (phase 2) – **featureSnWPrep**, and one method that performs an almost identical function – **finalGamePrep** – but is used when I’m predicting game outcomes influenced by predicted player statistics (phase 3).

As stated earlier, the training set is the 1979 season to the 1999 season, the validation set is the 2000 season to the 2014 season and the test set is the 2015 season and 2016 season. This is true for player predictions, game predictions, and game predictions influenced by player predictions. In each step, after training an algorithm on the training set (and after evaluating on the validation set) the validation set is added to the training set, and the algorithm is retrained. At no time were the 2015 season and 2016 used to train any machine learning algorithm.

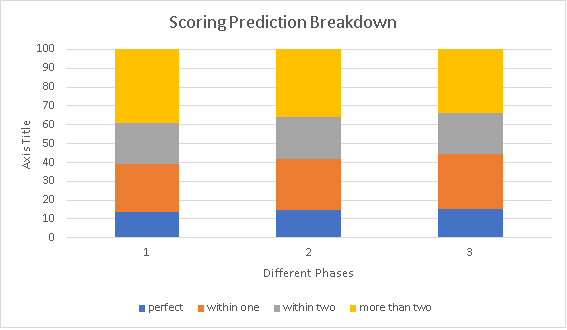
Also describe the baseline solution and state-of-the-art solutions to which you will be comparing your performance and indicate why these are reasonable or good enough.

**Results** (tables, figures and 1-3 paragraphs)

The baseline accuracy for predicting the winner of a game was 53.55%. The neural network algorithm predicted the winner of a game 57.15% of the time. And the neural network supplemented with predicted player values predicts 61.85% of the time. The below chart shows the three values mentioned above along with the best predictive accuracy of previous works.

The below graph shows the accuracy and non-zero accuracy of baseline player statistics versus predicted player statistics. Since some of the statistics associated with player performance have a value of zero for any particular game, I also used the measurement of non-zero accuracy that is an accuracy measurement on stats when they are greater than zero.

I also developed a neural network that predicts the amount of runs a team will score based on their lineup’s statistics. The baseline was achieved by selecting the run average for the team for that season.

****

**Discussion**

Although my algorithm outperformed other reported work, statistical analysis didn’t show that the increase in accuracy was significant. However, one of the papers I read stated that Las Vegas odds makers accurately predict winners about 58% of the time, so outperforming them is interesting.

In the related works, I found they significantly increased their accuracy by breaking predictions down by month – to almost 70%. With more time, I would like to analyze my months accuracy breakdown. Baseball game statistics are very erratic in the early months of the season for several reasons including the small samples size and players not necessarily being in their peak form.

In the future, I think that by including pitching related statistic would greatly increase my algorithms accuracy. I would like to include pitching statistics such as Earned Run Average and Strikeouts per nine innings.

**Conclusion** (2-3 paragraphs)

I believe I am on the right track using neural networks to analyze baseball games. Baseball is a difficult sport to predict with even the best teams losing nearly 40% of their games. This noisy dataset makes predictions difficult, but I believe neural networks offer the flexibility needed to accurately approximate this noisy data. Since baseball has a plethora of statistics, including a wider variety of features could increase the accuracy of the algorithm.

I was disappointed that my algorithm didn’t perform better, but outperforming Las Vegas, and other works, even if by a small percentage, is encouraging. I believe, pitching statistics would greatly increase the accuracy.

**References** …

Donaker, Gregory. “Applying Machine Learning to MLB Prediction & Analysis.” *Http://cs229.Stanford.edu*, cs229.stanford.edu/proj2005/Donaker-MLBPredictionAndAnalysis.pdf.

Jia, Randy, Wong, Chris, Zeng, David. “Predicting the Major League Baseball Season.” *cs229.Stanford.edu*, cs229.stanford.edu/proj2013/JiaWongZeng-PredictingTheMajorLeagueBaseballSeason.pdf.

Retrosheet.” Retrosheet. N.p., 2017. Web. 12 Sept. 2017.