NeuRayl

A neural network based baseball projection system

Joshua Imbriani

1 Abstract

The problem this paper and system seek to answer is the problem of how to predict upcoming baseball season statistics given past data. This problem is interesting precisely because of the reason it is needed: massive sums of money are now commonplace in baseball. Teams as well as fans seek to more accurately predict player seasons so as to avoid bad contracts and seek new, good ones. This paper approaches this problem by applying neural networks to this novel problem. A one-hidden-layer and a two-hidden-layer neural network are applied to historical baseball data in order to learn career trajectories and generate predictions. The predictions are strong but fail to surpass the current state of the art proprietary systems. In conclusion, this approach shows great promise and in the future if more time is spent on the underlying statistics as well as the configuration of the neural network, better results can be achieved, even surpassing the state of the art.

2 Introduction

With the advent of sabermetrics, baseball front offices across the league have adopted advanced statistics and computers. Team after team have hired more and more Ivy League economists, statisticians and computer scientists at the expense of who used to make the decisions, the so-called "baseball people" that did things by eye or instinct. At the same time as this revolution in America's game came another one: a massive influx of cash. Buoyed by a new wave of lucrative TV deals, the highest team payroll went from $92,538,260 in 2000 to $223,352,402 in 2016. Player salaries likewise shot through the roof but as can be assumed not all of these new contracts were great deals. For every good big money contract, there seems to be five regrettable ones.

With this new financial climate and this new brain trust came the advent of projection system that would seek to prevent bad signing by projecting player stats through yet unplayed seasons. The more accurate the projection system, the more likely to avoid an albatross of a contract as well as the more likely to spot a bargain.

This report will evaluate some previous work on the topic, lay out the proposed neural network based system and describe the results, comparing them to the current state of the art. The neural network proposed in this work is a simple one layer neural network with the linear activation function. A neural network with two hidden layers is also evaluated. They are both compared each other and the current state of the art systems.

Overall, the results were strong. The system was able to predict the upcoming seasons of players to within a small amount of each of the important metrics for players with a godd amount of data while it was not able to do the same for players with sparse data. This system demonstrates strong potential and with some more rigorous statistical adjustment of the data and by continuing to configure the neural network itself, this system can continue to improve and become a state of the art system itself.

Previous projection systems have run the gamut from purely statistical to a neural network implementation (from 1997!). The state of the art projection systems on the market today are PECOTA by Baseball Prospectus, Steamer by Cross et. al and ZiPS by Dan Szymborski. PECOTA, developed by Nate Silver, is a proprietary secret but is believed to be a nearest neighbors based implementation weighing each players upcoming season and past seasons against similar a window of years for past players. Steamer weights past performance and incorporates the important concept of regression to league average. ZiPS is similar to Steamer in its implementation. These projection systems are all secrets and no articles exist on their inner workings. The data isn't readily available so they are impossible to compare against on a large scale, forcing me to compare individual players to them by hand.

Academic works have also approached this problem through various means. Jiang and Zhang used a Bayesian estimator to moderately good results[1]. They limited their system to predicting the batting average only and compared it only with other statistical prediction methods, not against the state of the art. Lyle used multiple methods, including neural networks, to tackle this problem to very good results [2]. He converted all statistics to rate statistics over the course of a 162 game season, not predicting whether a player would stay healthy based on previous data.

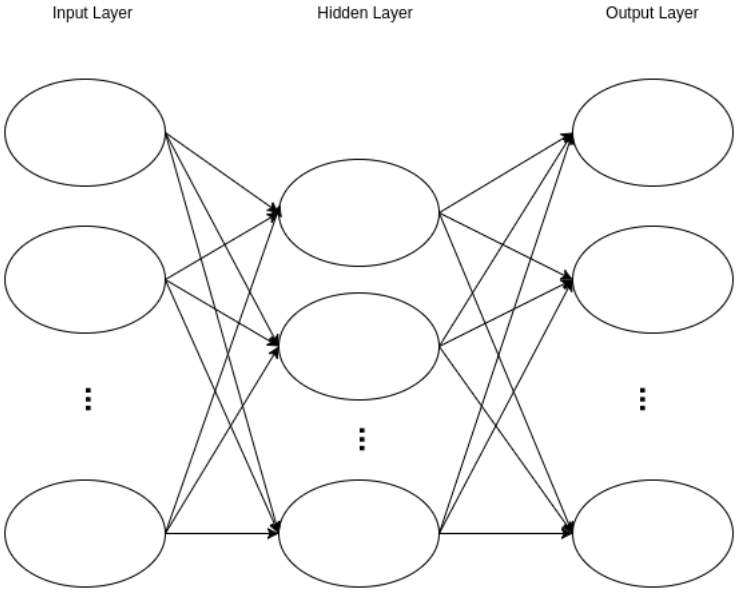
3 Approach

A description of the approach must start with a discussion of the data. The data for this system comes from the Lahman Database. This data is the project of Sean Lahman who, with the help of many volunteers, has digitized and collected every baseball statistic from nearly the beginning of the game in the late 1800s to the present day.

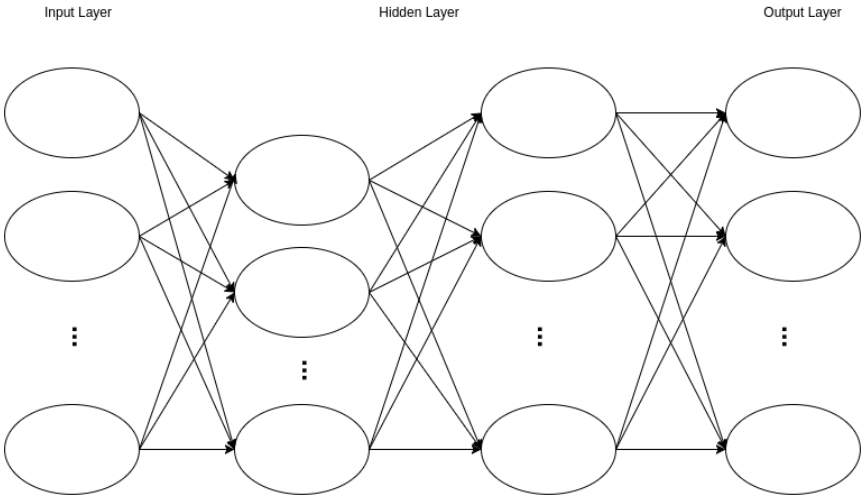
This data also had some preprocessing applied to it before it is ready for prime time. Baseball statistics cannot exactly be compared apples to apples right away. In baseball there exists the concept of park and era adjustment. In baseball, there have existed a few distinct eras such as the Deadball Era in the 1920s and the Steroid Era in the early 00s and the average player from each of these eras looks very different. In addition to this, different parks lead to different statistics due to the fact that each park has unique dimensions. Before statistics can be compared, they must be normalized. Luckily the Lahman database contains a statistic known as the park factor that measures how many runs are scored or given up in comparison with the rest of the teams that year.

Also I will apply a "league factor" to the data that compares the total runs scored in the year vs previous years which reflects the current offensive climate of the game. After this post processing, the data is ready to be used.

This system will use a neural network. It will take in a feature vector that differs thus producing three different versions of the algorithm. The first iteration (1B) will use all of the counting stats from the Lahman database for the past year as well as the players age and years played in the league (this is to represent the wear and tear of the game on players). The next iteration (3B) will use the past three years as inputs with the age and years played and the last iteration (5B) will use the past five years. Players without this amount of data will be filled with the league average values.



For this project I am adopting a simple fully connected neural network with one hidden layer. This seemed like a good simple but still powerful configuration to run (other configurations are covered in Future Work). I am using the Tensorflow library for code for my NN. I am using this library because it already has robust, efficient implementations of all of the architectures that I want and even has code for running it on a GPU, which I have and will greatly speed up the learning process over a CPU.



After seeing the learning results from the first neural network configuration, I included a second configuration of the network in order to see if I could get better learning results. For this second configuration, I adopted the architecture shown above, adding a second hidden layer so as to map this function better.

There are a few parameters that must be set. The activation function in the hidden layer is linear. The regression output layer is optimizing the output via Stochastic Gradient Descent. It is calculating loss as Mean Squared Error.

The output of the NN will be the amount of counting stats the system projects for each player. From these counting stats, rate stats such as the batting average or on base percentage can be computed. In order to get results, I will project the results of the 2016 season based on 2015 data and then compare that with the actual data from the year to see how well the projection system did. I will also compare the results to how well the state of the art systems did.

4 Results

The 1 year back, single layered neural network showed the lowest mean squared error while each of them showed individual strengths and weaknesses when evaluated on a single player.

The results from our running of the dataset follow this brief explanation. The results for the entire data set are presented in the first table. I trained my model on around 95% of the data and tested it on the rest of the 5% of the data. 1000 epochs were run and the mean squared error results are presented below.

Mean Squared Error on Single Hidden Layer NN		
1 Year Back	3 Years Back	5 Years Back
16972.3783	34358.7043	55774.0151

After seeing these results, I added a second layer to the network, in order to see if the function lends itself better to a two layer approximation. I ran the same datasets on the two layer neural network for 1000 epochs and got the following results:

Mean Squared Error on Two Hidden Layer NN		
1 Year Back	3 Years Back	5 Years Back
23658.7260	28137.2197	37333.3320

After running the full dataset, NeuRayl was compared to the current state of the art. Since there exists no publicly available dataset that gives full predictions for every year for each projection system, I am evaluating the systems by showing the projected and actual statistics for a few players for the 2016 season. Mike Trout was chosen because his statistics were relatively similar between 2015 and 2016 and Bryce Harper and Evan Longoria were chosen because their stats differed greatly in two different directions from their stats in 2015, testing whether the system can respond correctly.

Mike Trout							
Model Name	Games	At Bats	Hits	Home Runs	RBIs	Stolen Bases	Walks
Actual	159	549	173	29	100	30	116
NeuRayl (1L, 1B)	179.89	619.55	160.83	30.82	97.77	12.17	88.85
NeuRayl (1L, 3B)	157.17	674.80	184.10	30.84	103.08	16.23	92.82
NeuRayl (1L, 5B)	106.48	-29.60	-31.46	104.96	-22.98	-85.56	-171.51
NeuRayl (2L, 1B)	152.46	554.97	141.96	29.87	92.48	12.26	82.92
NeuRayl (2L, 3B)	155.15	568.02	153.37	31.14	95.29	20.14	88.70
NeuRayl (2L, 5B)	138.14	531.59	138.18	28.91	87.38	19.61	85.82
ZiPS	-	-	169	37	101	19	96
STEAMER	138.36	-		35.91	104.02	14.54	89.31

Bryce Harper							
Model Name	Games	At Bats	Hits	Home Runs	RBIs	Stolen Bases	Walks
Actual	147	506	123	24	86	21	108
NeuRayl (1L, 1B)	180.86	605.67	163.13	32.89	104.99	8.06	112.94
NeuRayl (1L, 3B)	152.26	633.33	171.99	26.96	93.18	9.66	96.63
NeuRayl (1L, 5B)	-48.51	-48.11	-31.62	-142.20	-67.12	81.78	29.44
NeuRayl (2L, 1B)	155.43	546.79	146.52	33.64	101.87	10.25	106.66
NeuRayl (2L, 3B)	145.43	517.10	139.90	26.84	84.96	14.93	87.43
NeuRayl (2L, 5B)	125.05	466.10	119.89	23.19	72.92	14.93	77.80
ZiPS	-	-	147	34	79	5	97.85
STEAMER	143.45	-	153.01	33.37	96.81	6.92	94.4438

Evan Longoria							
Model Name	Games	At Bats	Hits	Home Runs	RBIs	Stolen Bases	Walks
Actual	160	633	173	36	98	0	42
NeuRayl (1L, 1B)	164.66	568.99	140.27	17.31	73.56	4.82	52.36
NeuRayl (1L, 3B)	144.45	637.80	166.55	20.55	83.63	6.25	54.84
NeuRayl (1L, 5B)	-82.82	123.83	-13.67	-45.43	-33.90	120.15	29.22
NeuRayl (2L, 1B)	140.21	515.29	125.95	19.46	71.62	4.18	47.48
NeuRayl (2L, 3B)	141.33	528.15	134.79	20.95	75.58	7.33	51.88
NeuRayl (2L, 5B)	127.25	494.12	120.84	19.93	70.70	5.82	53.20
ZiPS	-	-	146	22	77	3	52.12
STEAMER	146.72	584.09	148.09	23.59	83.01	2.66	55.09

5 Discussion

As can be seen from the data, across both configurations of the data, the single year back data configuration significantly outperformed the three and five year back datasets. This is because of the increased zeroes in the three and five year dataset. In the MLB, only a small amount of players get the majority of the at bats and plate appearances. This results in many players being called up from the minor leagues due to injury or other reasons that might cause the MLB

regulars to not play, getting a few at bats and then getting called down to the minor leagues when the star gets healthy. This is a problem with how the data was preprocessed. In order to maximize the number of training samples, players without a full one, three or five years of history were padded with zeros. So these players were skewing the weights for the neural network, by obscuring the true changes that MLB regulars get between seasons.

In addition to this, the model is affected by the lack of minor league numbers used to fill in for years that players are missing. With the current data configuration, a rookie's first year will receive a very small prediction since their past few seasons were all zeroes. A rookie with regular playing time will nearly always outperform this.

I was initially disappointed with the error results from my models. I had hoped to receive a much lower mean squared error number for the simple, single layer network. However after considering the shortcomings of the data, these results show promise; the system is learning well and the bad predictions can be blamed more on the data than the system.

After seeing the one hidden layer system try with some success to predict statistics, I set up a system with two hidden layers to see if the extra hidden layer could help it handle the intricacies of the data better. As can be seen from the results, the two layer network was mixed. It was outperformed by the single layer one year back model by 40% but it pretty significantly improved on the three and five year back models, improvements of 22% and 49%. I hypothesize that the two hidden layer system is better with the extra zero padded data, explaining why it can significantly outperform the three and five year back models but be outperformed itself by the one-year back model.

By comparing the results of the different configurations of NeuRayl to the current state of the art, we can see that NeuRayl performs comparatively. In some cases, such as in the vase of at-bats and hits for Evan Longoria, Neurayl (1L, 3B) predicts to within 4 and 7 respectively whereas the state of the art is about 50 and 25 respectively. The system does not perform all positively, though, with the (1L, 5B) configuration routinely getting nonsensical answers. This is probably due to the extensive amount of zeroes within the input vectors due to sparse data. There does not appear to be a clear across the board superiority for either system, which is why I classify NeuRayl as comparative to the state of the art systems.

What does NeuRayl do well compared to the state of the art and what does it do poorly? Positively, it tends to regress each player's stats to their career mean, an important tenet of baseball and sabremetrics. The other two state of the art models tend to overemphasize recent data whereas NeuRayl treats each past year with the same wight. Negatively, it sometime receives nonsensical answers, with some negative values and some values that are impossible in the game (¿ 162 games). This could potentially be fixed and capped in the future.

If these individual players look relatively similar, why is the squared error so poor? It mostly comes from players with sparse data. The system has a hard time with what to do with players with only a few career at bats and games. Whereas it learned the career trajectories of MLB regulars fairly well, it cannot

predict players with sparse data very well, hugely driving up the mean squared error.

6 Conclusion

In conclusion, this system shows great promise, comparable with the current state of the art systems for projections of players with much data. For players with sparse data, the system performs poorly, which skews the mean squared error of the system as a whole. The data configuration appears to be the main problem with zeroes filling in for unknown statistics, which teaches the system on erred data. With some work, however, this system can very easily reach or even surpass the current state of the art system and be of great use for a major league team (and maybe even get the author a job with one).

7 Future Work

Future work will be centered on a few areas of improvements. First will be on the statistics side. The way I do park and league adjustments are very rudimentary. Each of these adjustments also contain some noise in them and are thus imperfect. Future work will center on perfecting the statistical side.

The other side of future work centers on the neural network side. First, I would like to join my current configurations together, averaging their values together. Each of the neural networks seemed to demonstrate mastery of a different statistic and if they could be combined, they could better predict player statistics as a whole. There are also a few more configurations I would like to try out: most notably the General Regression Neural Network[3]. I also would tweak my simple NN more as well, adding another layer or two and varying different parameters. This is work I will continue to do as I tweak my work more.

This model also has no concept of prospect pedigree or takes into account minor league stats. Where I pass in the 0s, I could potentially pass in adjusted minor league data so as to more accurately predict how a prospect will do given how well they performed in the minor leagues. This avenue will be explored in the future.

References

[1] Wenhua Jiang, Cun-Hui Zhang, et al. Empirical bayes in-season prediction of baseball batting averages. In *Borrowing Strength: Theory Powering Applications–A Festschrift for Lawrence D. Brown*, pages 263–273. Institute of Mathematical Statistics, 2010.

[2] Arlo Lyle. Baseball prediction using ensemble learning. *Athens, Gerogia*, 2007.

[3] Donald F Specht. A general regression neural network. *IEEE transactions on neural networks*, 2(6):568–576, 1991.