CMP5133 Artificial Neural Networks 16/17

NBA Players All Star Classification

Samet Yazak
Department of Computer Engineering
University of Bahçeşehir
Istanbul, Turkey
yazaksamet@gmail.com

In this article, I will give details about NBA (National Basketball Association) players All-Star prediction according to their yearly statistics. I will compare three classification algorithms, Multi-Layer Perceptron with Backpropagation, Radial Basis Functions, K-Nearest Neighbors. I will share results with unbalanced/balanced data and full feature set, best feature set results.

NBA, All-Star, multi-layer perceptron, backpropogation, radial basis function, k-nearest neighbor

I. INTRODUCTION

All-Star players are best players in NBA that is selected by fans and coaches every year for exhibition match. It is prestigious for a player to be selected as all-star. The goal here is to predict whether or not a player will be selected as all-star. It is a binary classification problem.

II. DATA

Data that used in this experiment has 20 fields and 4120 records where 215 players are categorized as all-stars. That means, data is unbalanced. To overcome this problem, I applied following methods:

- Duplicate minority class until sufficient balance is obtained.
- Delete majority class randomly until sufficient balance is obtained.

Sufficient balance percentage has been considered as 40%.

A. Data Collection

For this classification, I need yearly statistics of players and all-star player lists in each year. By collecting recent a few years old data, classification task may lead incorrect results since all-star players' statistics change every year. So, I collected last 17-year data (since 2000). Data is exported from http://www.basketball-reference.com in csv format. Data is not well shaped, so after extraction, a pre-processing step is needed.

B. Data Fields

Field Name	**Explanation**	**Data Type**
Player	player name	identity
Games	games played	numerical
Min	average minutes per game	numerical
Pts	average point per game	numerical
OReb	average offensive rebounds per game	numerical
Dreb	average defensive rebounds per game	numerical
Reb	average total rebounds per game	numerical
Ast	average assists per game	numerical
Stl	average steals per game	numerical
Blk	average blocks per game	numerical
TO	average turnovers per game	numerical
PF	average personal fouls per game	numerical
FGM	average field goals made per game	numerical
FGA	average field goals attempt per game	numerical
FG%	average field goal percentage	numerical
3PTM	average 3-point shot made per game	numerical
3PTA	average 3-point shot attempt per game	numerical

3PT%	average 3-point percentage	numerical
FTM	average free throws made per game	numerical
FTA	average free throws attempt per game	numerical
FT%	average free throw percentage	numerical

Table 1 – Data fields

C. *Feature Selection*

For feature selection part, firstly I calculated correlation matrix of fields according to target class. According to these correlation values I fed fields to KNN algorithm to which fields generate better results together. Below, you can find the matrix. Bolded fields are final results of feature selection procedure.

Field	Correlation
Games	0.19362651954
Minutes	0.37118485219
Points	0.51761020897
Rebound	0.34957348974
Assist	0.36374357265
Steal	0.32176438550
Block	0.24253395097
Turnover	0.43403979948
Personal Foul	0.17889546228
Field Goal Made	0.50121407031
3-points Made	0.19633734464
F.Throw Made	0.54251463647

Table 2 – Correlation Matrix

D. *Pre-Processing*

The extracted data has some duplicated or missing values. For first thing to do is to remove/adjust these values. Data also has some inconsistent values with players have same name and surname. It is important to distinguish these players because in most of them, father is the all-star but we have statistics of son since we only deal with 2000 or newer years.

Final data has 20 numeric fields which should be scaled before running classification algorithms.

III. CLASSIFICATION

For this binary classification task I implemented three classification algorithms and compared these algorithms with balanced/unbalanced data. Finally, I ran algorithms after applying feature selection methods to data.

A. *Backpropagation*

I implemented backpropagation algorithm for 2-class discrimination and tested the algorithm with both balanced/unbalanced and full feature set / selected set data. It is time consuming to test multi-layer perceptron with all features involved. It takes 1 hour to complete 10 epochs which is not sufficient to train a network. So, for this approach, I will share results with selected features according to correlation values to output field.

Following, there are plots of different set of features and different number of hidden neurons.

1) *Balancing with minority duplication*

As mentioned before, when number of instances increase, computational requirements also increases dramatically. So, for this example I will give result with only 10 iterations (epoch) completed.

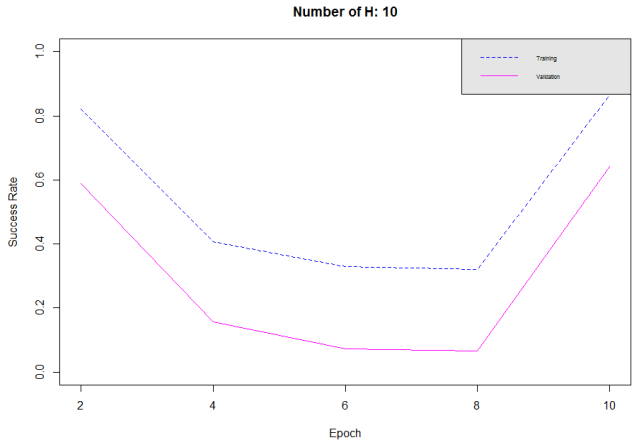

Figure 1 – Backpropagation with minority duplication

This limited result shows that; error fluctuates too much. This is because training procedure just started and also the nature of data. I will give detail on this later in conclusion section.

2) *Balancing with majority deletion*

Best results in both performance and training/validation error are obtained with majority deletion method. In this method, random records from majority class are deleted until data is balanced. Below, there are plots according to different number of hidden units (H) in hidden layer.

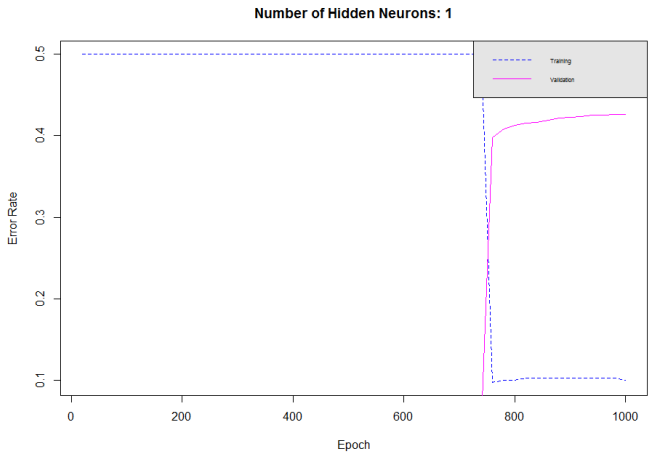

Figure 2 – Backpropagation with majority deletion

With one hidden neuron, both training and validation errors are not satisfying since around 700 epochs, validation error starts to improve as training error decreases.

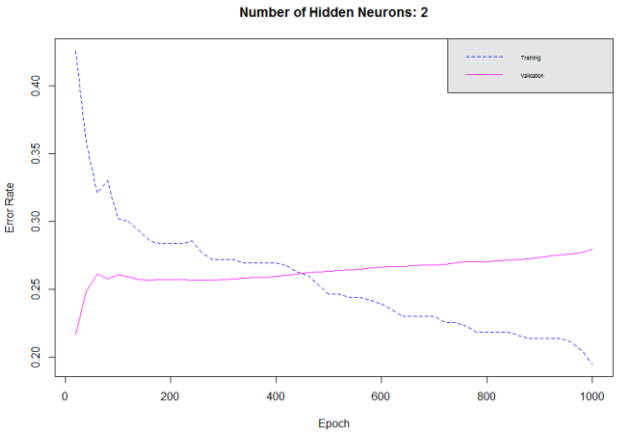

Figure 3 – Backpropagation with 2 hidden neurons

With two hidden neurons, training and validation error seem to get smother. But still it is not good enough to say model has been set.

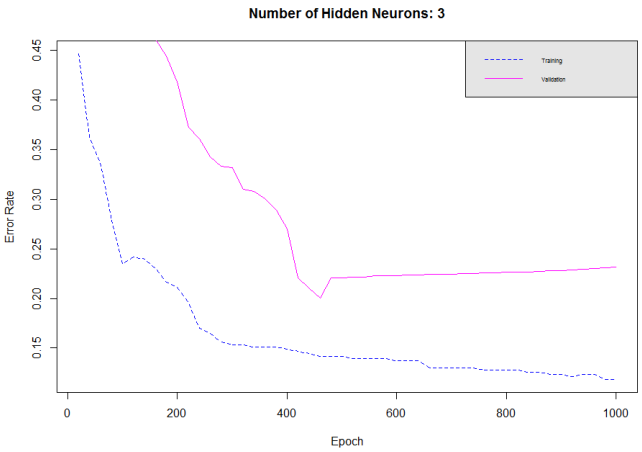

Figure 4 – Backpropagation with 3 hidden units

With three hidden neurons, errors tend to decrease until 0.20 when epoch is reached around 400. After that point validation error starts to increase although training error decreases which can cause an overfitting.

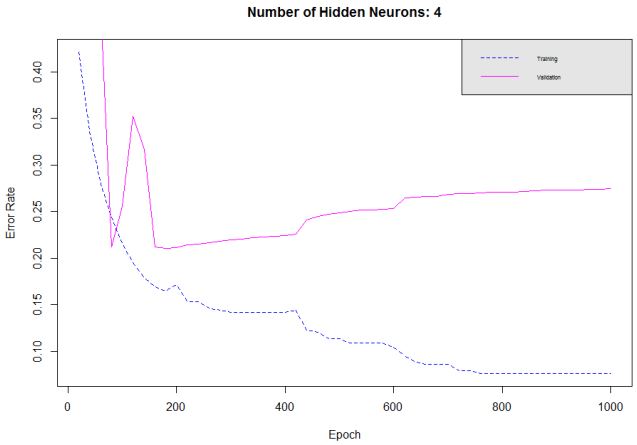

Figure 5 - Backpropagation with 4 hidden units

With four hidden neurons, we get an overfitting as we had with three hidden neurons. But, this result causes an overfitting earlier.

B. Radial Basis Function

RBF has been used in this experiment as hybrid learning method. In the unsupervised part, k-Means algorithm, in the supervised part single layer perceptron has been implemented. Similar to MLP-Backpropagation, clusters generated from k-Means has been used as hidden layers.

Radial Basis Functions require less computational power than multi-layer perceptron, so I will share more detailed results compared to MLP.

1) Balancing with minority duplication

Unlike MLP, I had time to finish RBF with data that is duplicated with minority class. I ran the process 7 times K (number of clusters) value from 3 to 9. They produced very similar outputs to each other. Below, there are success rate plots of K value 3 and 4, as I mentioned other results are very similar. You can full plots from URL in additional information section

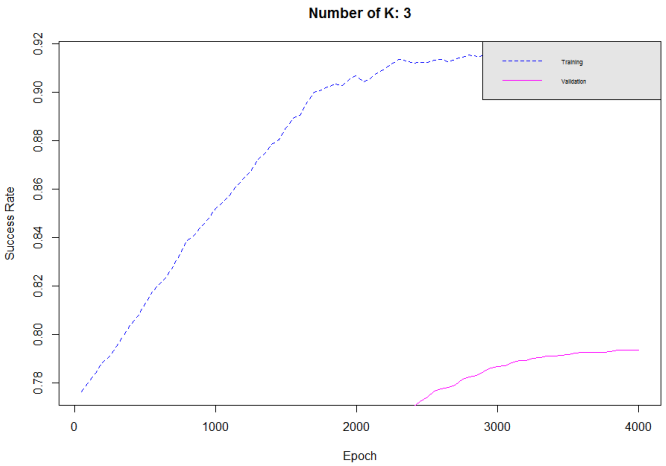

Figure 6 – RBF network with 3 clusters

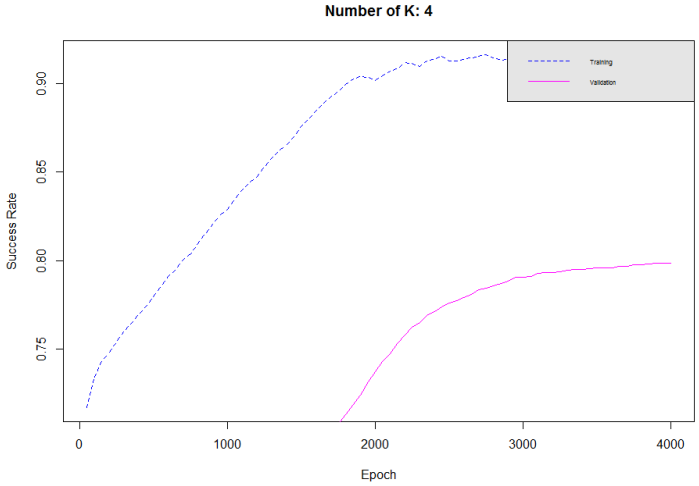

Figure 7 – RBF network with 4 clusters

2) Balancing with majority deletion

Duplicating minority class has some drawbacks like elapsed computational time. To overcome this problem, I deleted random non-All Star records to even data set. This procedure has advantage of having small dataset but this is also a disadvantage since more samples mean better learning.

This procedure also executed with some selected highly correlated attributes which are average minutes per game, average points per game, average rebounds per game and average assists per game. These fields have been extracted using correlation matrix in Section II.

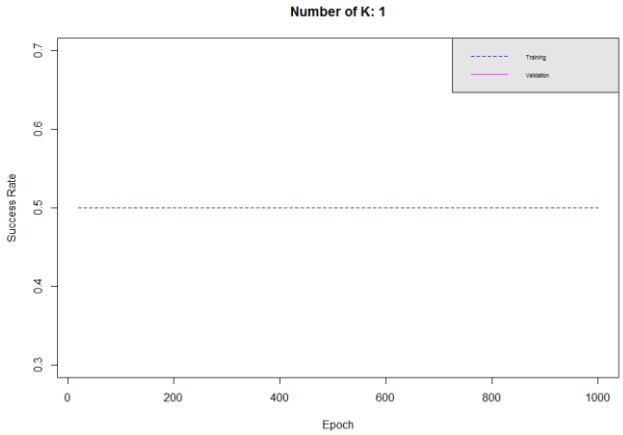

Figure 8 – RBF with majority deletion – K:1

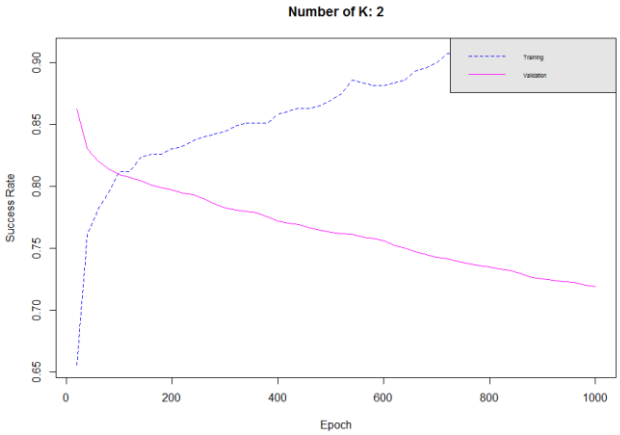

Figure 9 - RBF with majority deletion – K:2

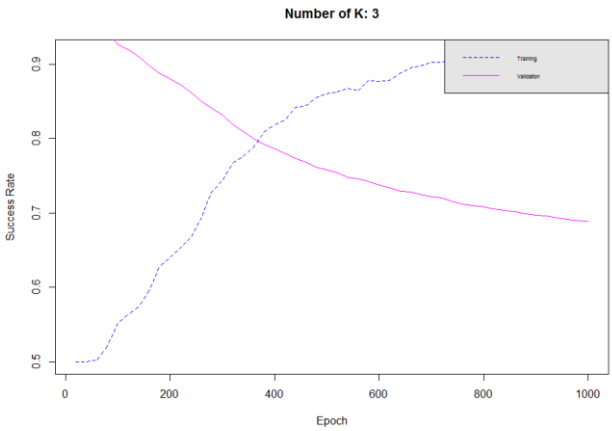

Figure 10 - RBF with majority deletion – K:3

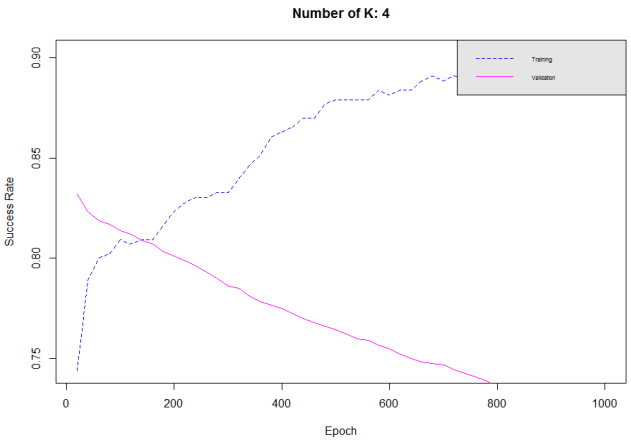

Figure 11 - RBF with majority deletion – K:4

These results show some interesting statistics. Although validation success seems to be decreasing as iteration increases, model does learn how to classify a player as an all-star. First let's look at the truly/falsely classified records.

K	epoch	TP	FP	TN	FN	%
3	20	0	0	215	215	0,5
3	40	0	0	215	215	0,5
3	60	1	0	215	214	0,502326
3	80	9	0	215	206	0,52093
3	100	22	0	215	193	0,551163
3	120	27	0	215	188	0,562791
3	140	32	0	215	183	0,574419
3	160	41	0	215	174	0,595349
3	180	55	0	215	160	0,627907
3	200	60	0	215	155	0,639535
3	220	67	1	214	148	0,653488
3	240	74	2	213	141	0,667442
3	260	85	2	213	130	0,693023
3	280	100	2	213	115	0,727907
3	300	107	2	213	108	0,744186
3	320	117	2	213	98	0,767442
3	340	122	3	212	93	0,776744
3	360	127	3	212	88	0,788372
3	380	136	3	212	79	0,809302
3	400	141	4	211	74	0,818605

Table 3 – RBF Training Classification Counts

K	epoch	TP	FP	TN	FN	%
3	20	0	0	3904	216	0,947573
3	40	0	0	3904	216	0,947573
3	60	110	77	3827	106	0,955583
3	80	149	164	3740	67	0,943932
3	100	160	245	3659	56	0,926942
3	120	176	287	3617	40	0,920631
3	140	184	340	3564	32	0,909709
3	160	189	392	3512	27	0,898301
3	180	195	439	3465	21	0,88835
3	200	204	482	3422	12	0,880097
3	220	210	522	3382	6	0,871845
3	240	211	565	3339	5	0,86165
3	260	213	615	3289	3	0,85
3	280	214	654	3250	2	0,840777
3	300	214	690	3214	2	0,832039
3	320	214	745	3159	2	0,818689
3	340	215	783	3121	1	0,809709
3	360	215	822	3082	1	0,800243
3	380	215	856	3048	1	0,79199
3	400	215	877	3027	1	0,786893

Table 4 - RBF Validation Classification Counts

As we examine these tables, we can see that, success rate of validation is because of dominance of non-all-star players. After several iterations, although success rate decreases, number of correctly classified as an all-star player increases. There are 215 all-star players in data set, all classified correctly. There are some misclassified records that cause success rate to decrease and I will discuss these records in conclusion part.

3) UnBalanced Data

Balancing data before feeding it to RBF gives good results. Now let's look at a success rate of a model without balancing.

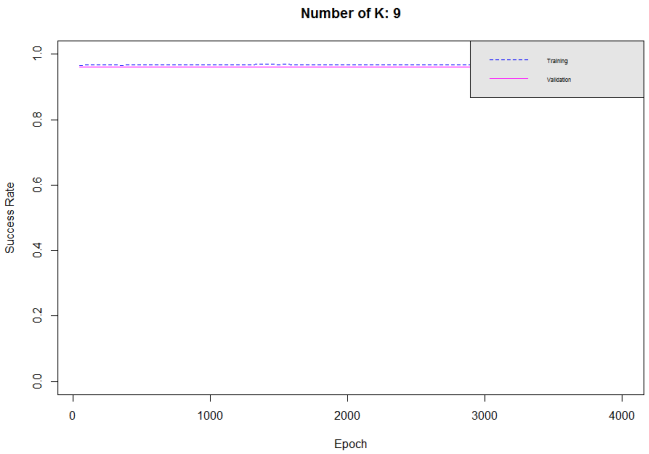

Figure 12- RBF with unbalanced data

It seems it has a great success rate, considering 1.0 as the highest. But, the result is caused by dominant non-all-star players. As seen in the plot, success rate doesn't fluctuate at all, this means it does not learn how to classify an all-star player.

C. *K-Nearest Neighbor*

Compared to previous two algorithms we discussed, K-Nearest Neighbor algorithm is quite simple to implement. Reason for including this algorithm to experiment is to compare its results with the complex algorithms.

Although its simplicity, learning a large data set with KNN algorithm takes a long time to finish. For this reason, I will only share the results with small balanced dataset with selected features that constructed from feature selection process that is described in Section II.

Below, there are results of KNN algorithm with K value from 1 to 10.

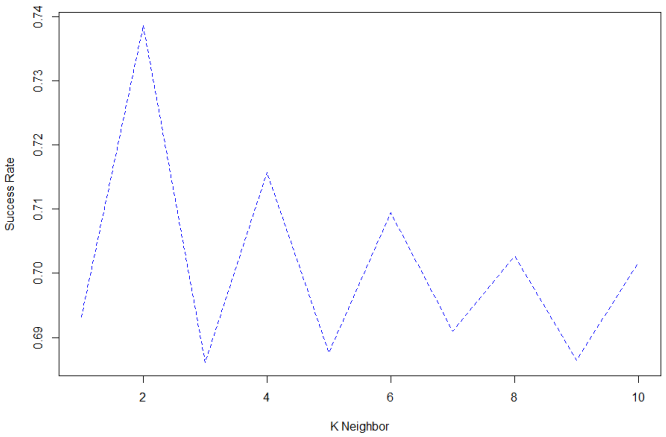

Figure 13 – KNN with K from 1 to 10

KNN produces good results but not as good as RBF or MLP.

IV. Conclusion

To sum up, all results of all experiments are better and faster with data balancing. Both of the balancing method we discussed results can be used for unbalanced data.

Feature selection process helped my algorithms to converge sooner than with full data. Especially in MLP, feature selection part is essential if you are running algorithms with limited hardware on personal computers.

One interesting result I had in this experiment is that RBF network that is trained with balanced data (majority deletion) has high success rate on start of the validation. Also, success rate tends to decrease although iteration count increases. Besides success rate, when I examined classified records, they are all classified non-all-star and non-all-star population is dominant. This causes model to produce high success rate. When the model gets trained, it learns to classify an all-star player with some errors which leads total success to be lower than start phase. One another reason that some records classified false positive is that; in a year only 12 players selected as all-star although many others have all-star statistics. These players are causing some experiments I had have lower success rates.

As I had time to finish the experiments I had, I can say that RBF networks are faster than MLP and KNN. RBF and MLP seem to produce better results but KNN models can also be preferred for good success rates and their simplicity

V. Additional Information

All material that is produced from this experiment can be accessed from https://github.com/yazaksamet/ANN_Term.

VI. References

[1] Ethem Alpaydin, Introduction to Machine Learning 2e. MIT Press, Cambridge, Massachusetts, London, England.