Using NBA Data to Help the Wizards Win Games

**Background**

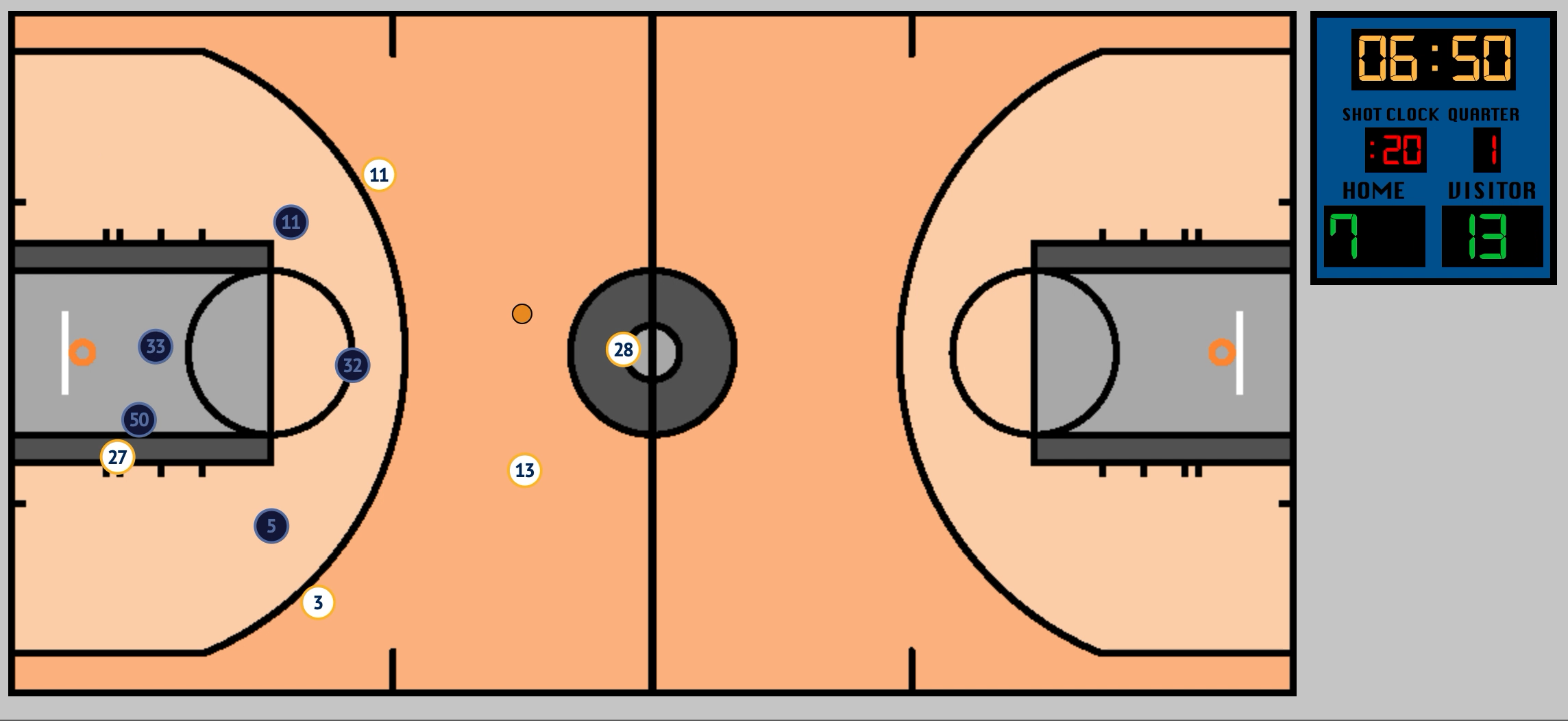
***Introduction*:**

NBA teams have access to more accurate and granular data than ever before and are devoting more resources to data science. The use of data science to drive strategy has become increasingly prevalent. For this project, I aimed to use NBA data to help the Washington Wizards win more games. Winning more benefits the Wizards organization and management for obvious reasons – they would sell more tickets and merchandise, attract better players, and so on. This was done from the perspective of a data scientist working for the Wizards hoping to provide better information to Wizards coaches to help the team win.

It is important that this information be actionable for the coaches. Some things are out of the coaches’ control – for example; which players are on the team and which opponents they play against. The information should be useful given the Wizards’ available resources and situation. With this in mind, the project focuses on half-court offensive possessions. This term will be defined below, but these plays tend to be situations where the offensive team has a lot of control, so strategy can be very impactful.

***Data and Definitions:***

In basketball, a “possession” refers the period of time starting when a team gets the ball and ending when the opponent gets the ball. Informally, people use “half-court offense” to refer to offensive plays where the action occurs after both the offense and defense are set up and prepared for the start of the play. For the purposes of this project, a half-court possession is one that lasts for at least 8 seconds after all players have crossed the half-court line. The project will only use data from the first six seconds after all players have crossed half-court. The images below show the start of one of these half-court possessions. Often, it is in these periods when offensive teams execute strategies designed to give them an opportunity to score – usually a specific series of passes, cuts, screens, and movements.

*Figures 1 and 2: Left: A snapshot of the player tracking data at the moment that the half-court possession started. #28 in white has just crossed half-court, making this the first moment of the half-court possession. Right: The game telecast at the same point in time*

The data used in this project comes from two sources: SportVU player tracking data from the 2015-16 season1 and NBA play-by-play data2, accessed using 3 from the same season. The SportVU player tracking data is what is enabling much of the newest and most interesting work in the field. Cameras are set up in every NBA arena that record the exact positions (x,y coordinates) of all players and the ball 25 times per second. The play-by-play data contains detailed information on every “event” that occurs during a game – every made or missed shot, foul, violation, timeout, etc. Together, this data was used to identify half-court possessions, how many points were scored on each possession, and get the x-y coordinates of the players and ball at each moment (25 times per second) during the possession.

***Relevant Research:***

There are many pieces of relevant research, some of which similar data or have similar goals. However, none of them used the methods that were used in this project. In “An Analysis of NBA Spatio-Temporal Data”4, Robertson used player tracking and play-by-play data to create a model that predicts the likelihood of a made shot using the player tracking data (and some additional variables) from the five seconds prior to the shot. Robertson’s analysis was interesting, but the results did not seem directly actionable in a strategic sense. In “Predicting Shot Making in Basketball Learnt from Adversarial Multiagent Trajectories”5, Harmon, Lacey, and Klabjan have a similar objective (predicting the likelihood of shot success given tracking data from the previous five seconds). This was notable because of the authors’ methodology; they used convolutional and feed-forward neural networks.

Wang and Zemel’s “Classifying NBA Offensive Plays Using Neural Networks”6 was the research that was arguably most similar to mine. Their dataset consisted of plays that were labeled in advance, belonging to one of 11 classes (play types). All plays that did not fall into one of these types were discarded. They then trained a classifier to determine which of the 11 types a given play belonged to. The paper did not include any analysis of the effectiveness of the different plays, which made it less directly useful to a coach then what this project aimed to accomplish.

***Objectives:***

My goal was to categorize or cluster all half-court possessions into play types and measure the effectiveness of each play type. This clustering would have to be completely unsupervised – my possessions did not have any labels. Then, this information could be used to answer questions like: Are there plays that the Wizards use frequently that are ineffective? Are there plays that the Wizards do not use often that are effective? How does the Wizards’ usage and effectiveness of the different play types compare to other teams’? The answers to these could help coaches choose plays that lead to more points in these half-court possessions, and therefore more wins for the Wizards.

**Methods**

***Data Preprocessing:***

Data from the two sources was preprocessed, cleaned, loaded, and joined together using Python and PostgreSQL. Several Python libraries, including Django and Pandas, were used in this process. Many possessions could not be used for various reasons – for example, some did not have player tracking data for the entirety of the possession, others had multiple sets of player tracking coordinates for the same points in time. Some others were excluded for other reasons – for example, possessions starting with less than one minute remaining in the game, half, or quarter were excluded because odd strategies, which may be inefficient in a vacuum, are sometimes employed in these situations. The play-by-play data was used to identify when possessions started and ended, and how many points were scored on each. The player tracking data was used to identify when all players and the ball had crossed half-court, and therefore whether or not the possession should count as a half-court possession.

***Clustering:***

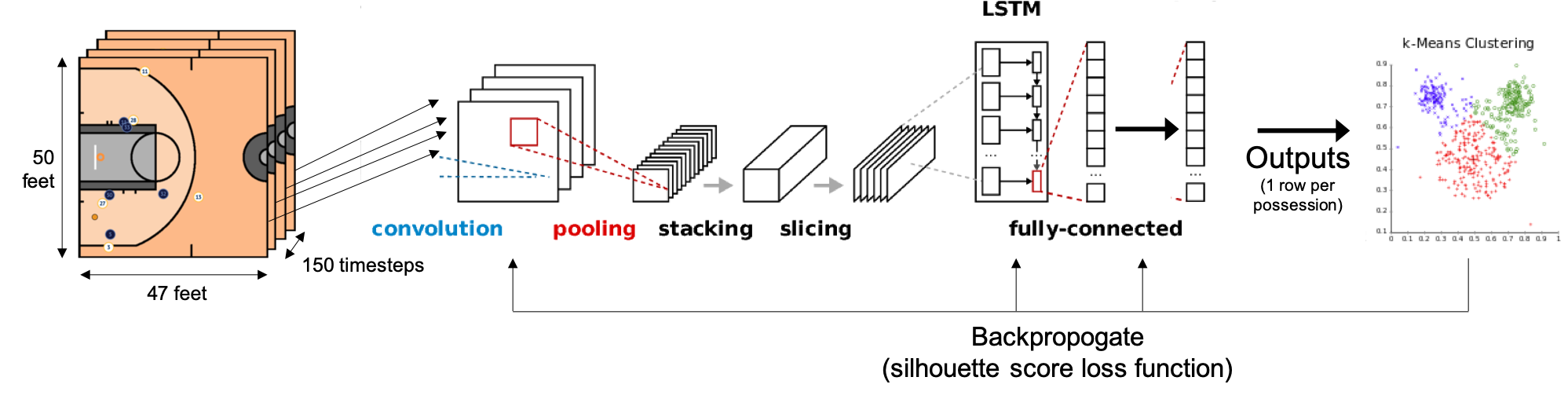
Before choosing a method or algorithm to cluster the possessions, some measure of cluster “goodness” (quantitative or otherwise) needed to be defined. Otherwise there would be no way of assessing the success or failure of my methods. In many unsupervised clustering algorithms (including K-means), the sum of squared errors is used as the objective or loss function that the algorithm tries to minimize. In this case, the “error” for each sample is the distance between the sample and the center of the cluster it belongs to. The silhouette score is another commonly used measure of the consistency of clusters. Each sample’s silhouette coefficient is equal to: , where *a* is the average distance between that sample and each of the other samples in their cluster, and *b* is the average distance between that sample and each of the samples in the next-closest cluster. The silhouette score is the average of each sample’s silhouette coefficient.

These scores are both useful in assessing the success of unsupervised clustering, but for the purposes of this project, it was most important that the clusters grouped the plays in a way that would make sense to a coach. In order to make recommendations based on the results, the plays in each cluster must be able to be described to a coach or player, so that the appropriate strategies can be used (or avoided).

Most clustering algorithms, like K-means, require one row of N features for each sample. This is necessary so that the distance between two samples can be calculated using these features. K-means, for example, uses an iterative process with assignment and update steps. After initialization in which samples are placed in initial clusters randomly or otherwise, each sample is re-assigned to the cluster that it is “closest” to – as defined by the Euclidean distance between the sample and the centroid of that cluster. Then, the new centroids are calculated. This process repeats until some stopping criteria is reached. In order to calculate Euclidean distance between points A and B , it is generally necessary to have “flat” rows of N features for each sample.

Because of this, clustering the plays using their player tracking data was difficult due to the complexity of the data. Each possession consisted of a sequence of 150 “images”, or sets of 11 x,y coordinates – for the five players on each team plus the ball. To cluster this data with a conventional clustering algorithm it would be necessary to somehow turn it into a flat row of features. First, I attempted to use a combination of neural networks and K-means clustering. A similar approach has been used in some research on unrelated topics, such as in Fan et al.’s “Neural Feedback Text Clustering with BiLSTM-CNN-KMeans”7. A combination of recurrent and convolutional neural networks seemed to be a good fit for the problem, because recurrent neural networks are often used to handle time-series data and convolutional networks are used for image data.

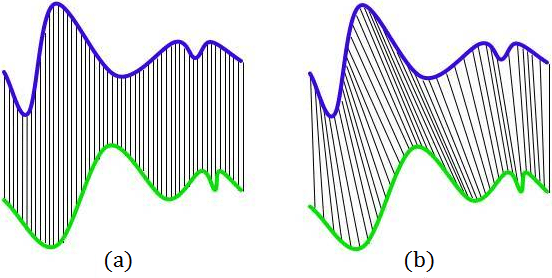
A network was designed that would use convolution and pooling layers to extract information from each image. The outputs of the convolution and pooling layers would be passed into an LSTM (long short term memory network). These types of networks retain some information from each timestep to the next, and contain neurons and gates that determine what information should be retained (“remembered”) and what should not. The LSTM outputted one row of features per possession, which is what K-means and other clustering algorithms require. Normally, these outputs would be used to make a prediction or classification, which would be compared to the true outcome for that sample. A loss function would be used to calculate the network’s performance for this batch of samples, and the weights throughout the network would be updated. This is not a prediction or classification problem, though – the plays do not have known outcomes or labels such as “pick-and-roll” or “post-up”. In my network, the features outputted by the LSTM were instead used for unsupervised clustering with K-means. Then, the silhouette score was calculated based on the results of the clustering, and this was used as the loss.



*Figure 3: Diagram of neural network consisting of convolutional and LSTM layers, in which the outputs are clustered and their silhouette score is used as the loss function*

Unfortunately, this approach was unsuccessful, in part due to computational limitations. The machine that was used could only process batches of 100 samples at a time. This meant that the LSTM outputs for 100 samples were clustered with K-means. Using between 8 and 16 clusters, this meant that some clusters would only have a few samples. The silhouette score fluctuated highly from batch to batch, and the network’s performance on test data failed to improve over time. Whether for this reason or some other, the network wasn’t able to learn.

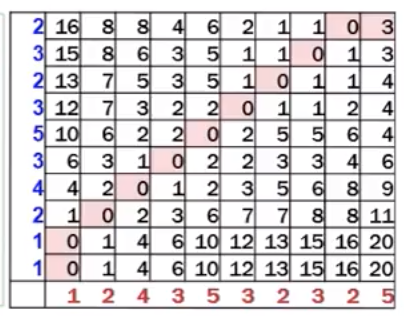
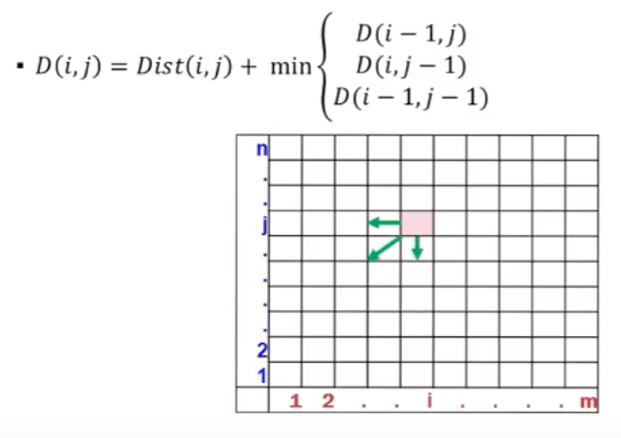
In lieu of a neural network-based approach, I utilized a clustering method specifically designed for time series data. K-means can be implemented using non-Euclidean measures of distance as well, including dynamic time warping. Dynamic time warping is a measure of similarity between two time series. It was originally designed to be used in speech recognition, to deal with varying speaking speed. When comparing two time series, Euclidean distance compares only data from the exact same point in time. Dynamic time warping compares each point to data from the other time series at many different points in time, and looks for the minimum distance of these comparisons.



*Figures 3 and 4: (a): two time series compared using Euclidean distance. (b): two time series compared using dynamic time warping8*

When comparing two time series A and B (of lengths M and N) with dynamic time warping, an MxN “cost matrix” is created. Each cell at (i,j) in the matrix is assigned a value equal to:

, where *Dist(i,j)* refers to the Euclidean distance between time series A at time i and time series B at point j.



*Figures 5 and 6: Cost matrix for dynamic time warping between two time series. In the case of the cost matrix on the right, the DTW similarity is 3 (the sum of all the numbers in the “warp path”)9*

The value of each cell in the matrix is dependent on the values to its left, bottom, and bottom-left; so the matrix must be constructed starting at (0,0) in the bottom-left corner. Then, starting at the bottom-left, a “warp path” is constructed going up and right all the way to the top-right. At each step along the way, the smallest adjacent (above and/or to the right) number is chosen as the next step. Finally, the dynamic time warping similarity of the two time series is equal to the sum of all of the numbers in this path.

The time series shown in figures 3 and 4 consist of only a single time-varying variable, but mine will use many variables. Features were extracted from each timestep, including the angles and distances between each offensive player and the other offensive players, ball, and basket. This totaled to 57 variables. For performance reasons, the number of timesteps was reduced from 150 to 30 – instead of one timestep/image every .04 seconds, now only one timestep every .20 seconds was used. So, each possession has 57 variables for each of 30 timesteps. Time-series K-means can now be used on this data. It works similarly to the K-means methodology described earlier, but uses dynamic time warping similarity rather than Euclidean distance.

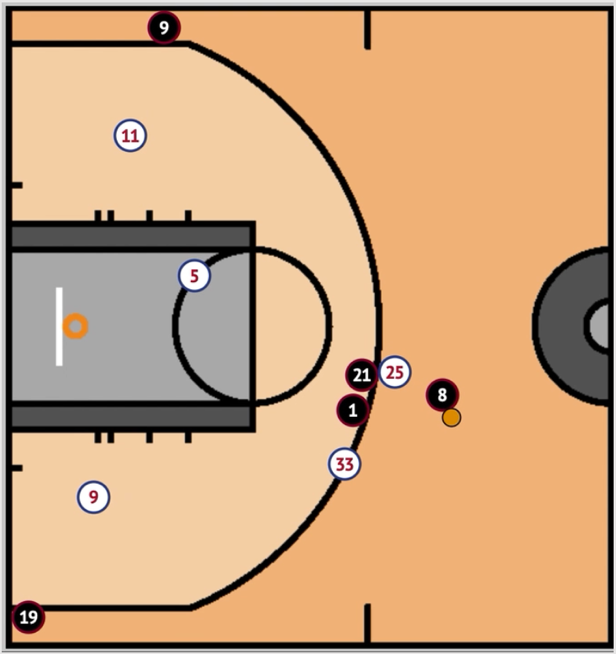
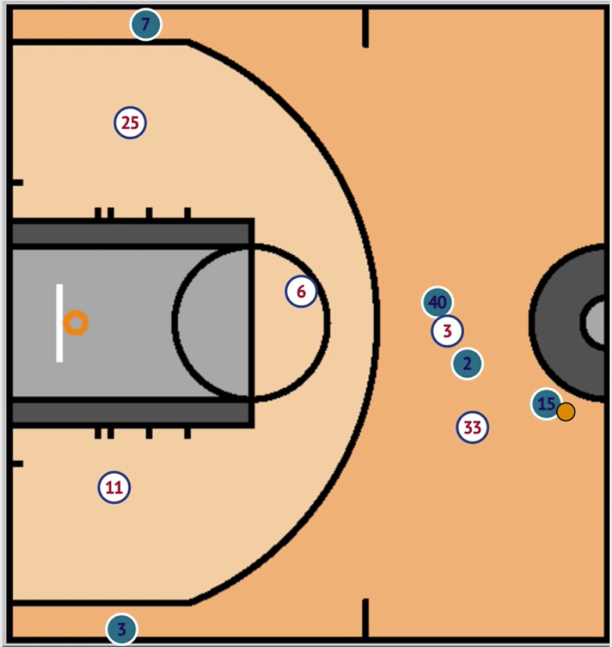
 

*Figures 7 and 8: Illustration of the angles and distances used as variables at each time step*

**Results and Conclusions**

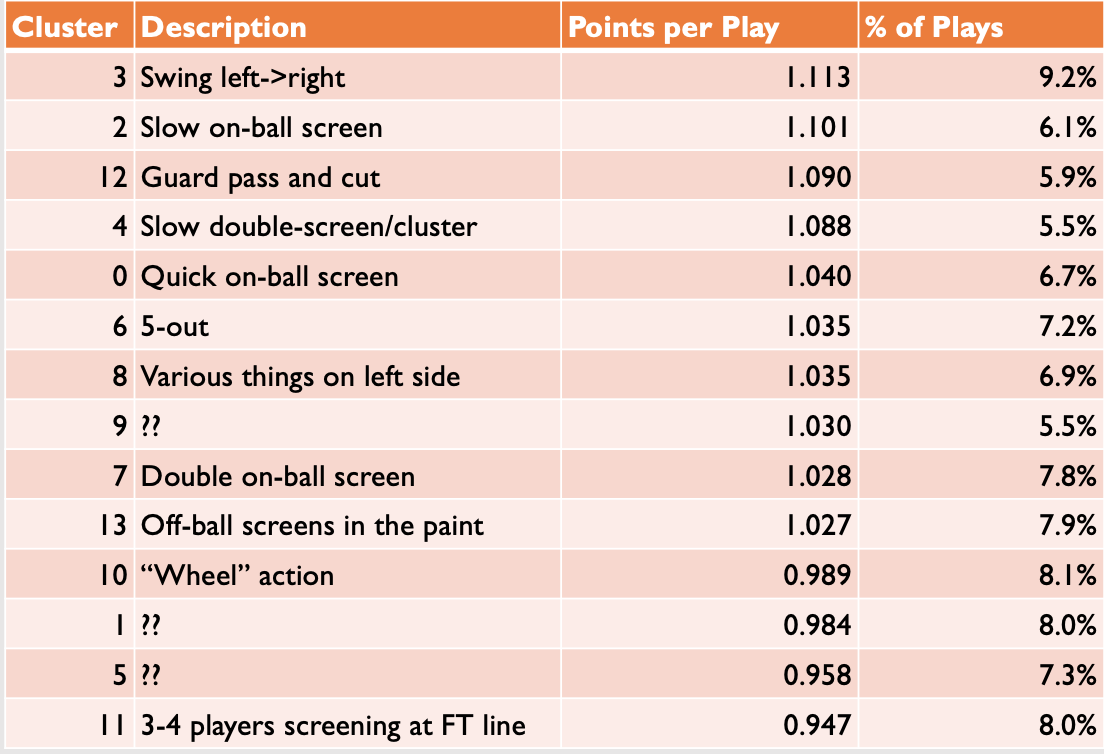
As described above, time series K-means clustering was performed on the 7,931 half-court plays in the dataset. It was run using every number of clusters between 8 and 16. The run using 14 clusters had the highest silhouette score, so this is what was used in the final analysis. The process is totally unsupervised, so it was not guaranteed that the clusters would contain plays that looked, to a human’s eyes, to be similar. From each cluster, the 5 plays (sometimes more) that were closest to their cluster’s center, in terms of DTW distance, were observed using animated visualizations of the player tracking data. The plays that are closest to the cluster centers should be the ones that are most representative of the bulk of the plays in that cluster.

I was able to identify a clear pattern in the plays in most, but not all, of the clusters. Given the unsupervised nature of the method, I viewed this as a success. The “Description” column in the tables below contains my own subjective short description of the pattern exhibited in the plays I observed from each cluster. There clusters are simply described as “??”. I was not able to identify a pattern in these clusters. They contained some broken plays, in which the offense’s strategy wasn’t able to be completed – or sometimes, even started – due to some error or odd occurrence. Defenses deflecting passes and offensive players throwing the ball out of bounds are some of the things that caused such plays.



*Figures 9 and 10: Snapshots from two plays that were in the same cluster, which I described as “double on-ball screen”*

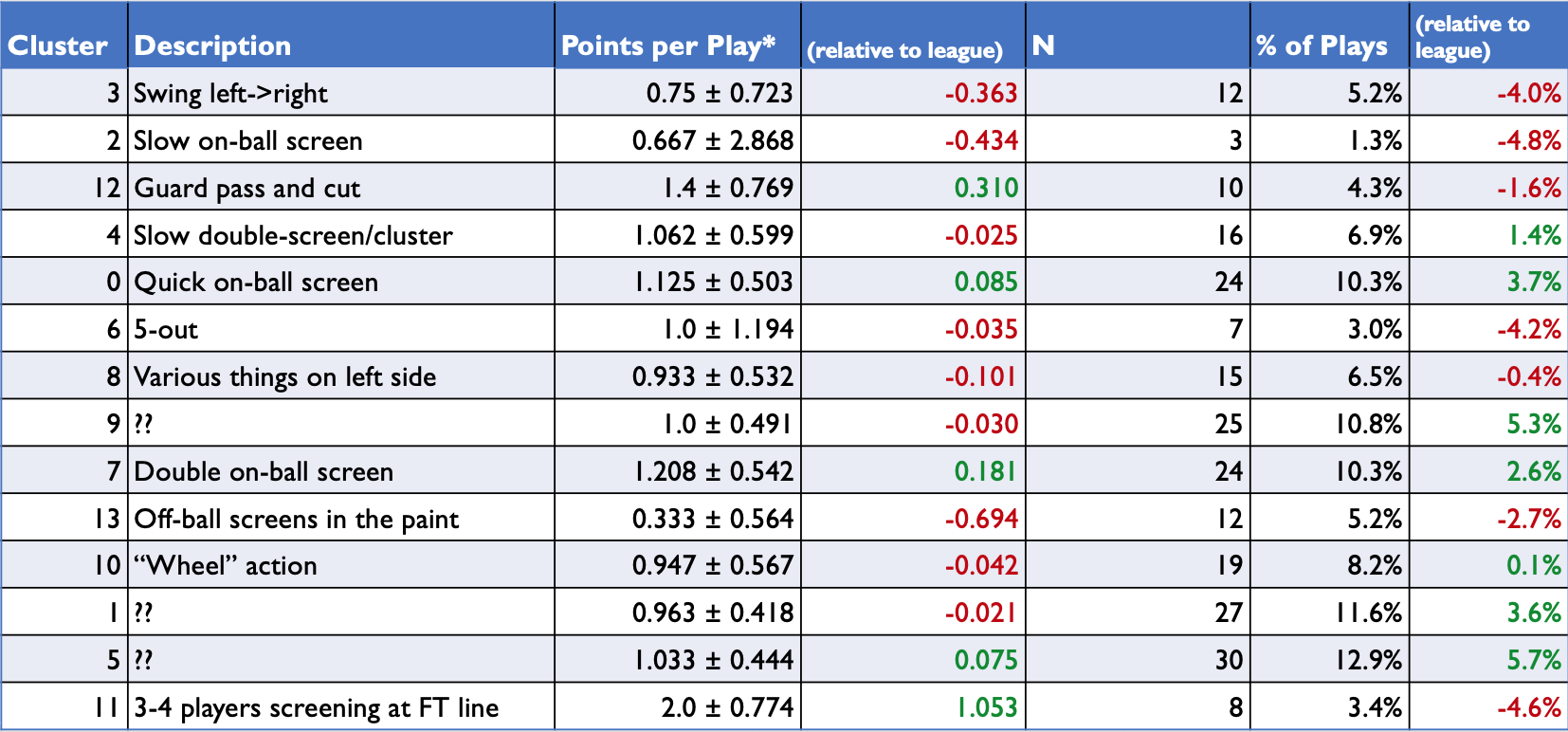
For each cluster, the tables show the average number of points scored on those plays and the percentage of plays that belong to that cluster. First, the results for the entire league:

**

*Figure 11: Play cluster effectiveness and frequency across all NBA teams. N=7,931*

This shows that, for example, slow-developing on-ball screen plays tended to be fairly successful; the second most successful cluster. They are run less frequently than some other clusters, though there are also other clusters of somewhat similar on-ball screen plays. Plays in which multiple players are gathered or screening near the free throw line were very ineffective, scoring only 0.947 points per play.

There is a reasonably large difference in points per play (PPP) between the most and least effective clusters. Especially when considered in the context of what good and bad offense means in terms of PPP – in 2015-16, the best offense in the league averaged 1.14 PPP and the worst averaged 0.99 PPP. This difference, compounded over many plays and many games, makes the difference between a champion and a bottom-dweller. With this in mind, we now look at the Wizards’ usage and effectiveness for these same clusters of plays, and compare it to the entire league’s.



*Figure 12: Play cluster effectiveness and frequency for the Wizards, compared to the entire league. N=232, points per play numbers include a 95% confidence interval which shows how little certainty there is in those numbers*

With only 232 plays in the dataset, it is difficult to draw too many conclusions from the results in this table. But, although the points per play numbers are certainly not reliable, the frequencies of the play types may shine some light on some of the Wizards’ strategy. The Wizards run several of the plays that are most effective for the league as a whole less frequently than other teams. In particular, plays where the ball is passed multiple times from the left to right side of the court, slow-developing on-ball screens, and point guard passes and cuts. The Wizards may be underutilizing these plays.

The Wizards also have more “??” plays than other teams, which are often broken plays when the intended strategy could not be executed. It is unclear whether this is a reflection of Wizards’ strategy or of the quality of their players, but if these situations could be avoided it could lead to better overall half-court offense.

**Discussion**

There are many things that could be improved on in this analysis, but the most obvious is to use more data. Only half of the 2015-16 season was included in the SportVU player tracking data I had access to, and only about 1/3rd of this was used in my project, due to time and space constraints. With more time and some more efficient code, more data could be used. This would be particularly useful for giving some more credence to the Wizards-specific points per play numbers, which have a very small sample size.

Another point to consider is the comparison between the Wizards’ plays and the entire league’s. Maybe it would be better to compare the Wizards’ strategies to only “good” teams, because these are the teams whose success and strategies they hope to emulate. “Good” could be defined by some threshold like making the playoffs that season.

Originally, I’d planned on using exclusively neural networks for this project. I was not planning on using the time series K-means clustering method, but it proved to be very effective and was very interesting to learn about. One sort of hybrid approach that could be tried would be to use a convolutional neural network to transform each image/timestep, then use these outputs as the variables in a time series K-means clustering. Then, the network could use silhouette score as the loss function as described previously. It is possible that the time series K-means could be more effective than using LSTM and then a traditional “flat” K-means.

1. Neilmj. “Neilmj/BasketballData.” *GitHub*, https://github.com/neilmj/BasketballData.
2. “NBA Stats.” *NBA Stats*, https://stats.nba.com/.
3. Swar. “Swar/nba\_api.” *GitHub*, 20 July 2019, https://github.com/swar/nba\_api.
4. Robertson, Megan. “An Analysis of NBA Spatio-Temporal Data.” *An Analysis of NBA Spatio-Temporal Data*, 2017, https://dukespace.lib.duke.edu/dspace/bitstream/handle/10161/15261/Robertson\_duke\_0066N\_14012.pdf.
5. Harmon, Mark, et al. “Predicting Shot Making in Basketball Learnt from Adversarial Multiagent Trajectories.” *ArXiv*, https://arxiv.org/pdf/1609.04849.pdf.
6. Wang, Kuan-Chieh, and Richard Zemel. “Classifying NBA Offensive Plays Using Neural Networks.” *MIT Sloan Sports Analytics Conference*, http://www.cs.toronto.edu/~zemel/documents/1536-Classifying-NBA-Offensive-Plays-Using-Neural-Networks.pdf.
7. Fan, Yang, et al. “Neural Feedback Text Clustering With BiLSTM-CNN-Kmeans.” *IEEE Access*, vol. 6, 2018, pp. 57460–57469., doi:10.1109/access.2018.2873327.  
     
   Images from:
8. Fick, Carlien. 2017, https://pdfs.semanticscholar.org/e532/1ae5a320327024521b8a48a5294cb8d43a63.pdf.
9. Di Rosso, Simone, director. *Dynanmic Time Warping*. *YouTube*, https://www.youtube.com/watch?v=tfOevFKQIjQ.