The Immaculate Reception

Using autoencoded receiver routes to optimize yardage

BIG DATA BOWL

Abstract

Because receiver routes are represented as long, variable-length chains of coordinates, we cannot simply feed them into a model and try to predict a play outcome. There are too many dimensions, and any output would be mostly meaningless.

So, receiver routes need to be represented in a way that is more conducive to modeling.

In this presentation, we show off three ways that we attempted to represent route data before ultimately settling on one method and using it in a final model to predict play yardage.

Our resulting model struggles in terms of predicting play outcomes, but is informative nonetheless - despite attempting a whole host of ways to model this situation, it seems that routes have little impact.

This is a claim that would require far more research (and data) to support, but we find little evidence to support the idea that routes are key to play success. More likely coaches are already using routes that are at least "good enough", and most of the variation in plays comes from randomness and player talent.

Data Cleaning

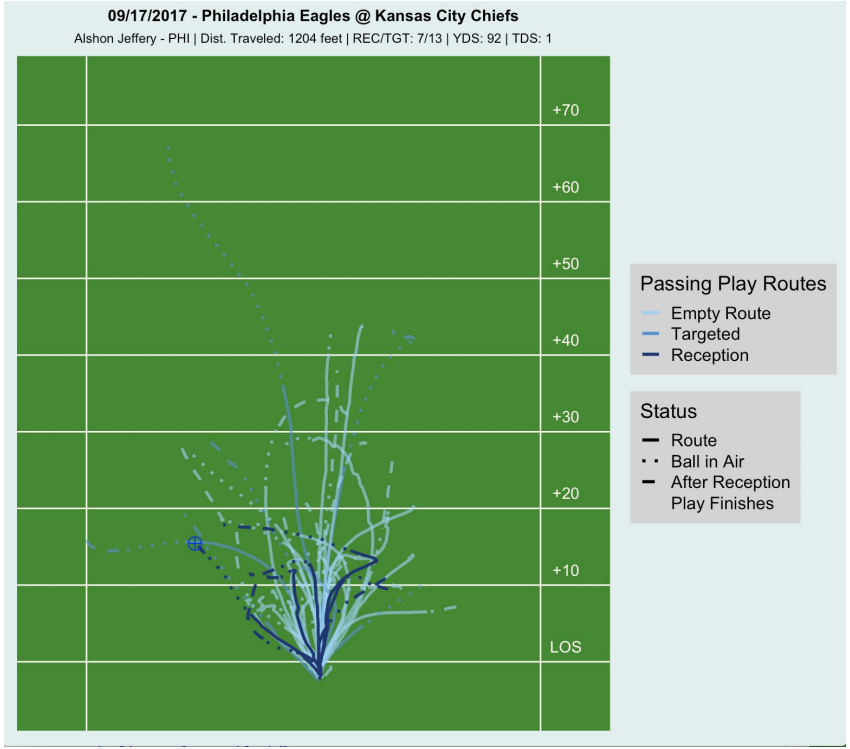
In order to treat the routes in a consistent manner, we have to decide to standardize the coordinates and route details.

We elected to start the route when the ball is snapped and end the route when the pass is either caught or deemed incomplete.

Additionally, all routes are linearly transformed to begin at the position (0,0). (For visualization purposes, we rotated the routes in the image shown to the right)

In an attempt to control for play calling preferences at different downs and distances, we elected to only use plays that occurred on first and ten.

We use the classification ‘receiver’ as running backs, wide receivers, and tight ends on passing plays.



All routes run by Alshon Jeffery, standardized using out data cleaning process. PDFs do not play GIFs, so the animation can be seen at this link: (https://gph.is/2CDugtQ)

Attempt 1: Time Series Clustering

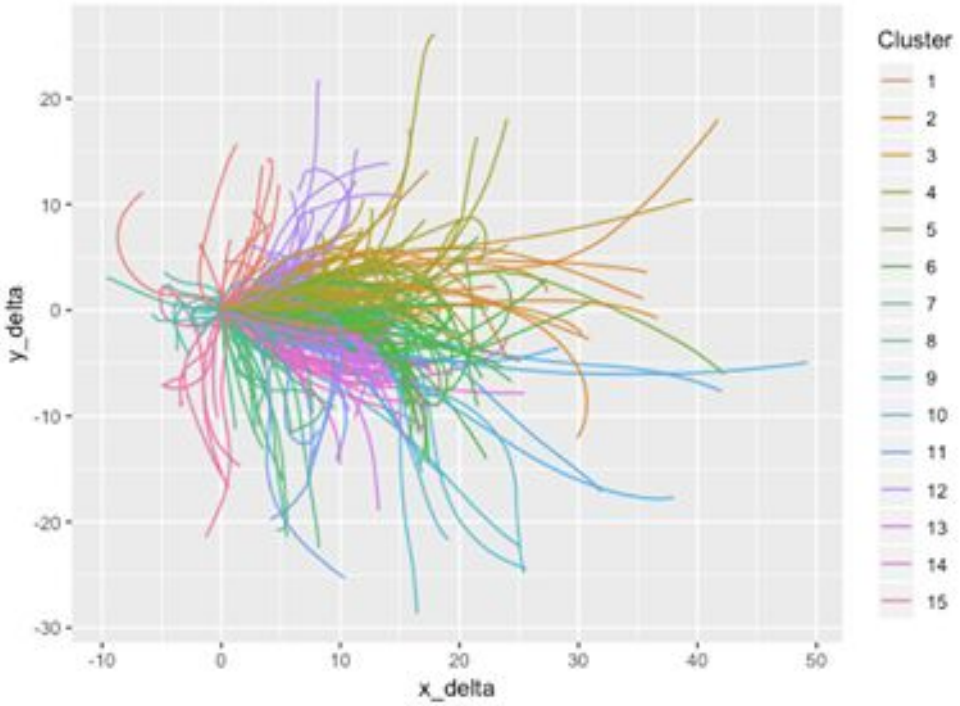
We tried to use clustering on the coordinates to identify routes that are similar to one another.

However, it is well known that clustering does not work all that well for time series[(1)].

We used Dynamic Time Warping to alleviate some of the issues, but we basically found that the clusters were providing little information.

Clearly, the clusters are only learning the general direction and distance of the routes - there is very little consideration given to the shape of the play.

That is extremely limiting, especially because the shape is probably what matters most.



Route clusters for one game; each cluster is a different color, and each line is a route.

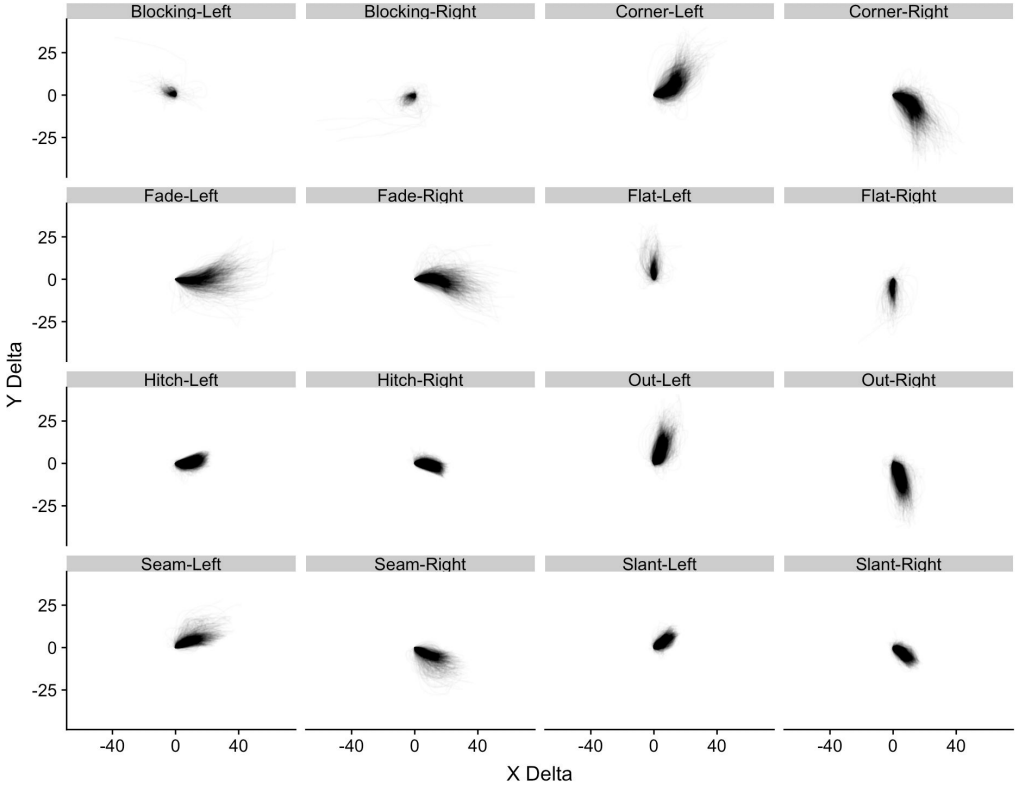
Attempt 2: Shape-Based Clustering

As clustering worked poorly on the spatio-temporal data, we decided to calculate some actual route features and used them to create new clusters.

From the route data, we generated several features meant to represent the "curviness" of the line. We measured a route's deviation from a straight line, and also count the number of inflection points.

Along with other additional shape features, these allow us to group routes in terms that better represent their actual form.

However, it still seems that the shape is not totally being picked up. There is still a considerable amount of variability within each group: so, we opted for other techniques.



Route Representations

Instead, we chose to turn routes from a time series into an “image”.

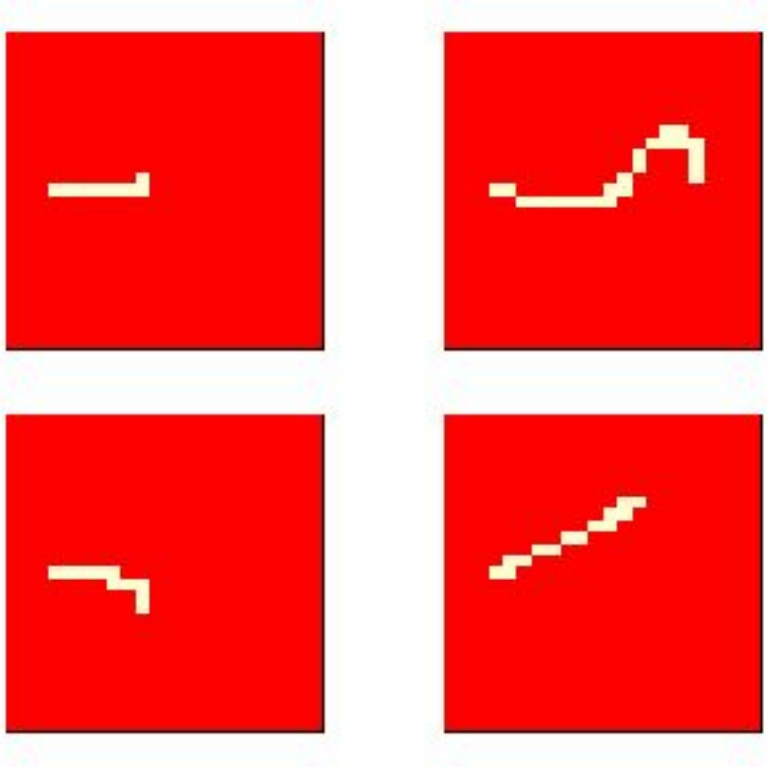
This is done by gridding up the field as pixels, and placing a 1 or 0 in the place for each pixel to illustrate the receiver’s path.

Our matrix (22x27) of 1s and 0s is then flattened into a 594-item vector that represents a single receiver route.

These 594-item vectors are still of too high dimension to be fed into a model, so some additional processing needs to be done.

What we would like to do is automatically reduce these vectors down into a distilled set of “traits” that still represents the route accurately.

We could cluster the vectors, but clustering works poorly with many dimensions and binary values[2]. The more effective thing to do is to use an **autoencoder**.



Routes go from left to right; the sidelines are on the top and bottom.

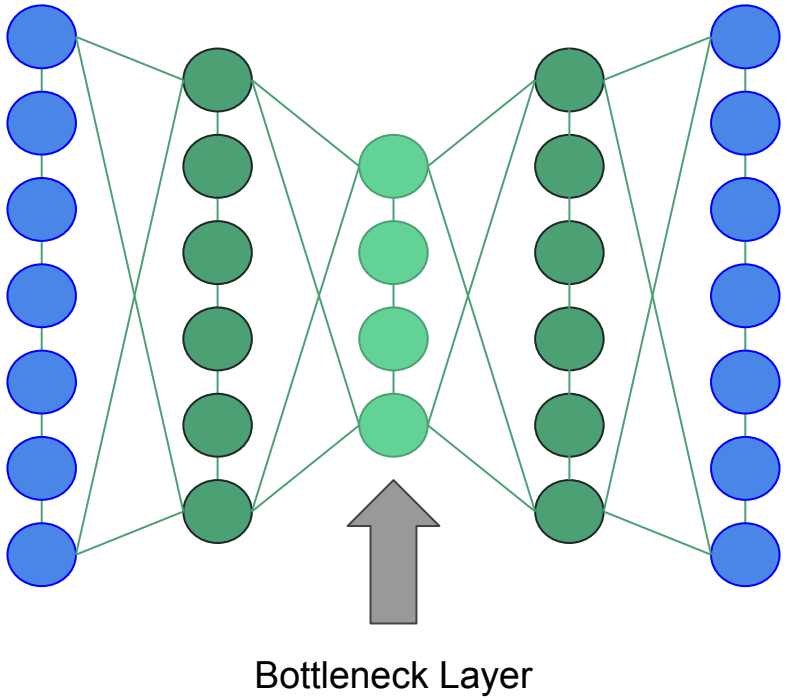
Attempt 3: Autoencoding

An **autoencoder** is a particular architecture of deep neural networks that works well for clustering-like tasks.

The autoencoder takes in a vector (like our 594-item input) and then shrinks it down to one small hidden layer. It then uses this "bottleneck" layer to generate a predicted vector that most closely resembles the input[3].

Extracting the neuron weights from the "bottleneck" layer leaves us with a lower dimension representation of the original routes. The autoencoded routes are essentially compressed versions of the original routes, but this compression is not done losslessly. The details of the routes are thus abstracted away.

This is useful since similar plays will end up having very similar values, which can then be used as variables in a final model.



Schematic structure of an autoencoder neural network.

Modeling

Boosting, multiple regression, random forests, and a **feed-forward deep neural network with backpropagation** were applied to data derived from **time series clustering**, **autoencoding**, and **shape-based clustering**. The models were tuned and then evaluated on test data. The boosted model (XGBoost) using the autoencoder data performed the best among our models. We illustrate our general modeling framework below using our XGBoost/autoencoder model as an example.

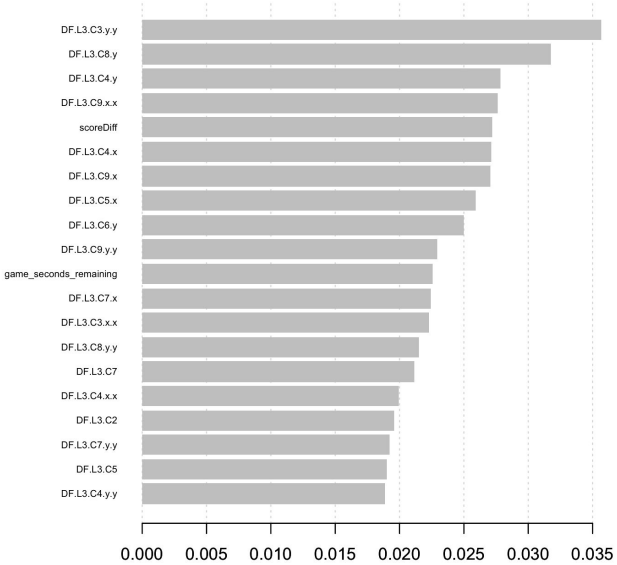
For the model, we represented the number of yards gained on a play based on the outputs of the bottleneck layer of **auto-encoded routes** for up to five receivers.

We also included the actual position of those receivers (for instance, if he is actually a running back), yard line, offensive formation, number of defenders in box, defensive formation, quarterback name, and score differential.

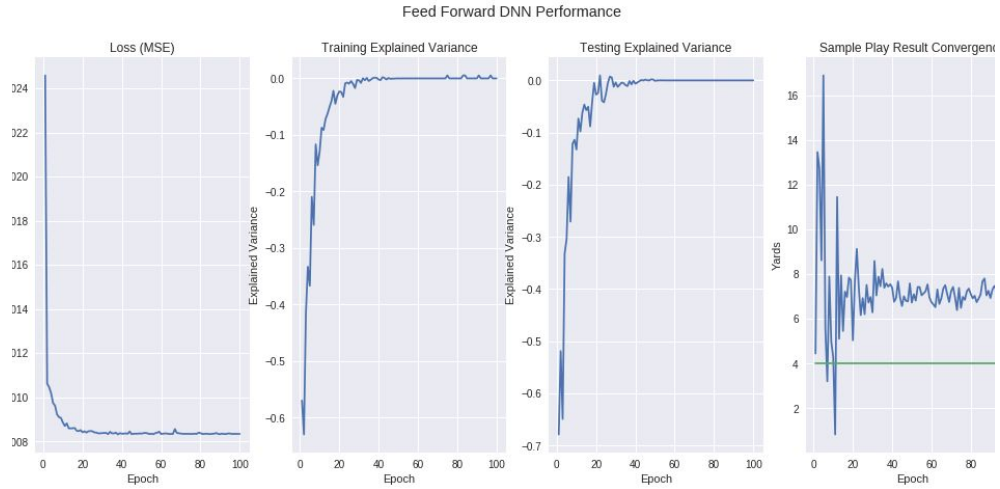
We then used XGBoost to forecast the yards gained on each play based on the above factors.

Despite our findings that the XGBoost model performed best (lowest RMSE) out of the models tested, these differences were not practically significant, and no model showed strong predictive power.

Example Model Experiments



Variable Importance plot of the boosting algorithm.
Most of the top variables are nodes from the autoencoded routes (i.e. DF.L3.C4.y), showing that they are more important than the passer or the game-state data we used in the model.



Feed-forward neural network with backpropagation.
The network converges, but explains almost no variance in the yards gained on the pass. The mean absolute error is 7.3 yards, meaning that the model struggles to predict the play outcome accurately.

Results

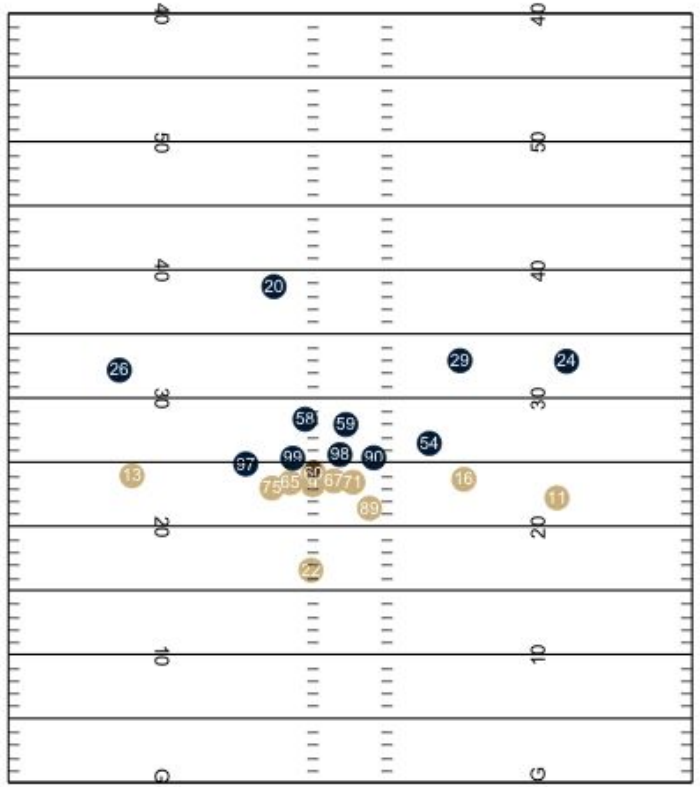
On test data, the model is able to explain 1.7% of the variance in yards gained (XGBoost with autoencoder layers before clustering).

The shape of the routes and the combinations chosen matter, but our model shows that other factors tend to matter far more.

Perhaps it is not the shape of the route that is key - maybe the speed and agility of the receiver matters more, or maybe other factors are dominant.

Based on our analysis, long routes, with a single receiver cutting across the middle, work quite well. This may be because the long routes from widely-dispersed receivers spread out the defense, opening the center of the field for a cut.

Despite the variety of our approaches, we find little evidence that routes have any effect on play success. Future research may be directed towards other areas. Perhaps more data would unlock new possibilities, but for the time being, our analysis did not yield positive insights.



The play with the second highest predicted yardage: Brees to Ingram for 25 yards. Model predicts 22 yards. (https://giphy.com/gifs/5bo7YxJAWLwvcAOQcH)

Credits

Jake Flancer - jflancer@wharton.upenn.edu

Jack Soslow - jsoslow2@gmail.com

Eric Dong - ericdong@seas.upenn.edu

Andrew Castle - castla@wharton.upenn.edu

Special thanks to Professor Abraham J. Wyner of the Wharton School for his advice and assistance.

References:

1. Keogh, Eamonn, and Jessica Lin. "Clustering of time-series subsequences is meaningless: implications for previous and future research." Knowledge and information systems 8.2 (2005): 154-177.
2. Steinbach, Michael, Levent Ertöz, and Vipin Kumar. "The challenges of clustering high dimensional data." New directions in statistical physics. Springer, Berlin, Heidelberg, 2004. 273-309.
3. Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." Proceedings of the 25th international conference on Machine learning. ACM, 2008.