Applying deep learning to basketball trajectories

Rajiv Shah & Rob Romijnders
tinyurl.com/traj-rnn

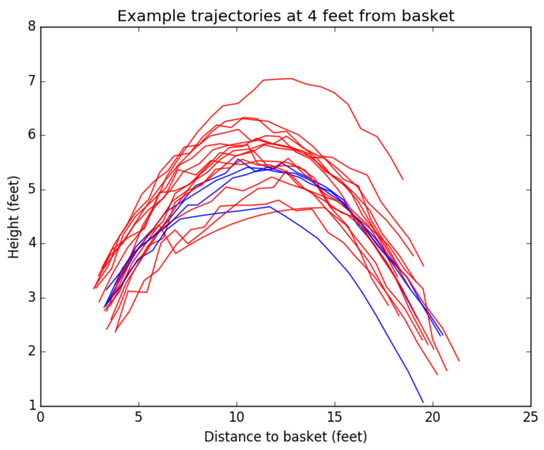
Problem:
Predict whether a three point shot will be made

We seek to teach a computer to predict whether a shot would be good or a miss. The computer would learn the same way we do, by watching lots of shots. (We don't teach the computer about acceleration, gravity, spin . . . all the factors that actually affect the motion of the ball.)

We start with two key questions:

Can we use deep learning to predict the success of a three pointer? — The challenge here is the predict the success of a shot when the ball is far from the basket.

Can we use deep learning to create new trajectories from such model? — This builds from the success of other RNNs models to generate text and art.

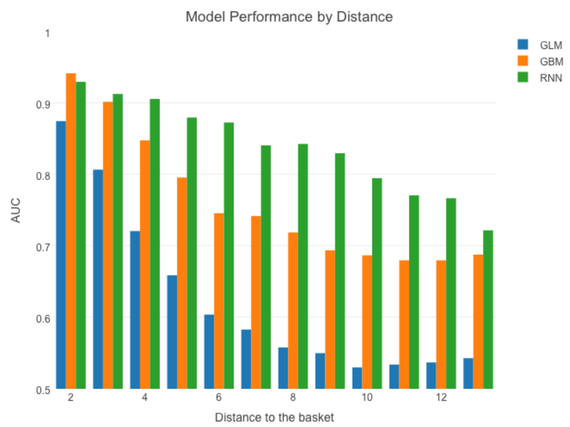


Results:
Recurrent neural network beats feature rich models

Our RNN model predicts whether a basketball shot will be a make or a miss. We developed twelve models based on the distance of the ball from the basket (2-13 feet). The RNN only uses xyz positional data along with the game clock (no feature engineering).

To assess the performance of the model, we use Area Under the Curve (AUC), which is a typical metric for a binary classification problem.

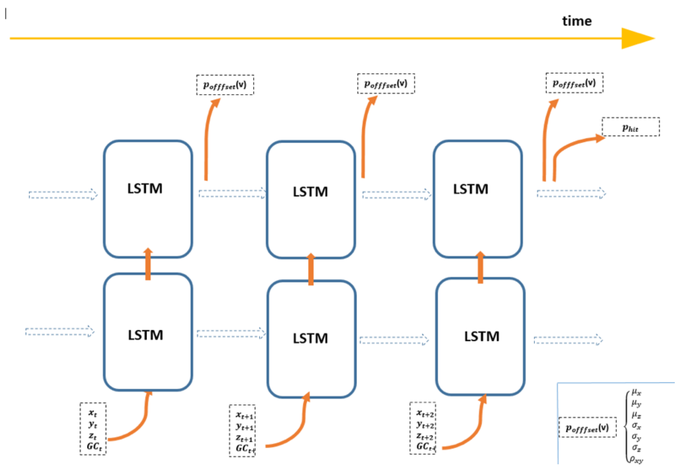
As the chart shows, the RNN performed much better than the baseline generalized linear model (GLM) and gradient boosted machine (GBM) model. The GLM and GBM relied on feature engineering (including position, speed, distance, and angle to the basket).



Approach:
Use a recurrent neural network with a mixture density network

The inputs consisted of the xyz position of the ball and the game clock over time. The final model used a Mixture Density Network (MDN). The model was a 2 layer LSTM network with 64 hidden nodes and implemented in Tensorflow.

For our model, we created a distribution with a mixture of Gaussian distributions. Gaussians are defined by their mean and covariance matrix. Every time step, the LSTM outputs parameters that define this Gaussian. Every Gaussian requires seven parameters:



Generative:
Create trajectories using the recurrent neural network model

This plots shows a trajectory generated by the computer. This is accomplished by recognizing that with the MDN, every time-step has a distribution for the offset to the next time-step. Here we sample from this distribution. If done for consecutive time steps, we can *generate* the full trajectory.

Bias and a priming sequence can improve the quality of the generated trajectories. The bias uses a simple heuristic to bias the predictions towards the mode of the distribution. A priming sequence uses several points of a true trajectory for the initial steps. These initial steps help to prime or warm up the LSTM, resulting in more realistic trajectories.