William Power
Milestone 1 - Pitch Predicting

Algorithm Description

Data

This project's goal is to correctly predict the next pitch in an at-bat, based on the past pitches and various additional features. To experiment on possible models, we need a corpus of data that contains information that may be useful for possible features descriptions.

The data was composed by tying together two sources. First, the MLBgame API, second the Lahman baseball database. This was needed because neither contained both sequence data, and handedness data. Data was collected in two phases. First, the MLBgame API was used to iterate over every game in a given year. For each game, each player in the roster was iterated over, and a query was made to the lahmanDB (running locally on a mysql server) to match the players names and insert the mlbgameID of the player into a new column.

With this done, the sequences can be built in a similar manner. For each game, each at bat is iterated over. The pitcher and batters mlbgamIDs are used to find the handedness from the lahmanDB. These are all composed into a single example and saved to a file.

BasicRNN

The ultimate goal of this project is to compare multiple models. First, the most basic RNN was attempted. Hopefully it will be easy to extend this to include more complicated features, and more complicated/useful cells.

The basic RNN contains a layer of neurons that produce an output, and a state vector. The RNN makes a prediction in the forward direction by reading in each character, and using the state of the previous input as the state of the current input, along with the next item in the sequence.

In this model, the RNN contains a set number of 'hidden' neurons, which are then fully connected to a 'projection' layer, which outputs a tensor the same size as the input.

The seq2seq loss builtin is used to calculate the loss from a prediction. The prediction here is the output of that fully connected layer.

The AdamOptimizer is then used to minimize the loss with respect to the weights in the model.

Goals

As mentioned before, the goal is to compare different types of feature vectors. However, I might experiment with more complex cells for the RNN.

Roadblocks

Data

Much of my assumptions about the available data were wrong. For starters, I naively assumed that the player data available from MLBgame would include the pitching and throwing hand of the player. This was not true. This required the additional work in gathering data.

Also, not every game included pitch types in their description of the at-bats. These needed to be identified and removed. It's not that the entries did not have a sequence, it was that each element was simply a blank space, instead of a id.

Model

The BasicRNN was difficult to implement, but the content of the past lecture helped get a basic graph going. The next roadblock is how to handle the addition of the new features into the sequences. I have a partial attempt now that concatenates another 'oneshot-like' vector to the end of the sequence, representing the batters position, along with a small array of two values that correspond to the handedness of the batter and pitcher.

In addition, I have a concern about the behaviour of the network as it learns weights, given that the concatenated features will never change from time step to time step. I was considering having the prediction rely on a subset of the output. That is, have a tensor that masks out all the non-sequence fields, and take the max of those to choose the prediction.

A final roadblock is the task of making a custom feature. One of the discussed goals was to find a way to incorporate some other feature that was 'custom made'. One idea was based on the idea of the 'types' of pitchers and batters that people refer to. If I could find additional data in either of the current datasets I've built that describes some attributes related the pitching and batting behaviour that are more physical, maybe these could be used in an unsupervised learning method.

That is, I could chose some data points like {average fastball speed, %fastballs thrown, %CU's thrown, …}, and see if there were groups that emerged with a clustering algorithm. These groups could then be used as a feature of their own, after classifying the pitcher/batter in the at-bat.

Experimental Plan

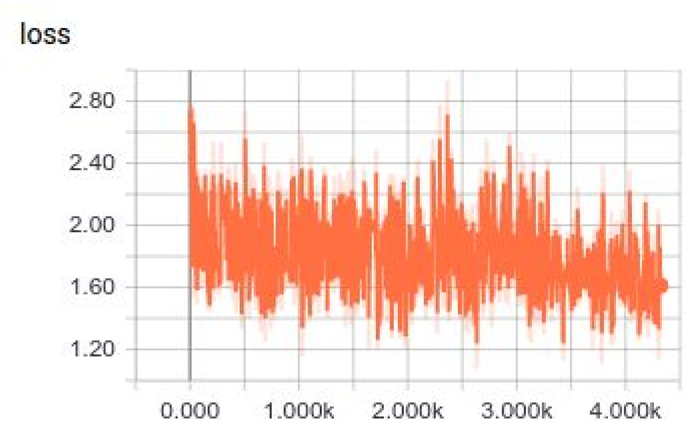
- Compare different possible additional feature vectors
  - Hands + Pos
  - Hands + Pos + Base Data (b1, b2, b3)
- For Each
  - Cross Validate Search for Learning Rate
  - Cross Validate Search for Neuron Count

Completed

Included in the zipped folder is a notebook directory containing jupyter notebooks describing the first two models. There is also a data directory which should contain just the pitches_2016.p file. Finally, there is a src directory which contains the scripts used to update the LahmanDB and to build the actual datasets. In addition, there is a file that contains just the code for the BasicRNN model.

If its easier, you can view the content of the jupyter notebooks on the repository set up for the project. Available here: https://github.com/wpower12/pitch-predictor-2000/tree/master/notebooks

- Data
  - Completed lahmanDB update through 2014
  - Gathered all of 2016 sequences
    - Sequence only
    - Basic Feature (pos, hands) + Sequence
- Model
  - Have a working graph for the BasicRNN model
  - Graph is training and recording a loss value to a tensorboard file
    - Sample loss graph below
  - Outline for the basic feature + sequence model.


loss
1.000k
2.000k
3.000k
4.000k