RNNs Part 3: Longer Predictions and Graphs

Eckel, TJHSST AI2, Spring 2024

Background & Explanation

Mean squared error is just a number; it's hard to tell how close to the real sequences our predictions are getting unless we see it. So let's see it!

In this assignment, we'll visualize the performance of our RNNs by predicting further into the future – predicting the next 10 values, instead of just the next 1 – and graphing a few different treatments, seeing how much they differ.

Make a copy of your RNNs Part 2 code to modify for this assignment. Make sure you have a working **RNNs 2** that you leave alone, and **don't** modify.

Predicting Further into the Future

What if we want to use our networks to predict the next 10 values, instead of the next 1?

For starters, we'll need a different training set; we'll need length 60 sequences, now, with the first 50 values as the input and the last 10 as the output. That's not too hard to do – just modify our sequence generator accordingly.

Then, for the predictions themselves, there are a few ways of going about this. Here are all the treatments we'll consider:

- Naïve prediction – use the actual value at step #50 to predict steps #51-60
- DNN prediction – use a [50, 10] perceptron network, trained as you'd expect (feel free to use the initialization radius trick from the previous assignment)
- RNN step-by-step prediction – here, we'll use the [1, 6, 1] network from last assignment and train it just like we did in the previous assignment, discarding steps #52-60 and using only step #51 as our target. Then, we'll use this network to on our testing set to take the first 50 steps and generate 10 more steps by feeding it values #1-#50, then as it produces each output after that we'll send that output back into the network as well, producing 10 more predictions one prediction at a time.
- RNN all-at-once prediction – here, we'll use a [1, 15, 10] network (to roughly match the number of parameters in a [50, 10] DNN) to predict all 10 of the next 10 values at once. Be sure to let this train for a while; 50-100 epochs is not unreasonable. If you get big spikes in error, use a smaller lambda.

To calculate mean squared error on an output with 10 values, **average the error you get across all 10 values** to get the mse at each particular input, then average all those mse values to get the mse for the whole data set.

Go ahead and code up all four of these treatments, and send me the mse you get from each one. As a gut check, to know you're in the right ballpark, the DNN mse should be somewhere around 0.015. Note this is much worse than the value in the previous assignment, but of course it is; this is harder! I won't tell you the rest though – how do YOU think these four treatments will compare to each other? Find out if you're right!

Graphing

After you send me the mse values you got on the previous page, to verify you're on the right track, it's time to make graphs.

The goal here is to make a 3x3 grid of subplots, line plots specifically, that show the first 50 steps given and then after that, in different colors, show all four different predictions for the next 10 steps along with the correct next 10 steps actually generated. Then we can visually compare how well the algorithms do. Since we're generating our data set randomly, this means that after we train each algorithm, we can just run it on whatever the first 9 items in our testing set turn out to be.

There are a few different ways to keep track of all the data. You could do some experimentation and figure out an appropriate number of epochs for each of the treatments on the prior page, then have your code set to run all of them, get results, and make some graphs, all in one run. Or, you could train the networks separately and pickle each one, then have code that loads the networks, runs them, gets the results on the first 9 items, and graphs them. Or you could train separately and pickle the specific results on the first 9 items in the testing set, and just write code that reads those results and graphs them. Up to you.

Specification

Either show me your 9 graphs on your computer, or send me a screenshot on Mattermost; no code submission necessary. Here's what I want to see.

For each of the first 9 sequences in a testing set, you should have a line plot that shows:

- The first 50 steps, in black
- The last 10 steps, correctly, also in black
- The predicted last 10 steps, from each of the four treatments on the previous page, in different colors that are labelled somewhere
- Underneath each graph, the mse of each of the four treatments on that particular sequence