Programming Assignment 3- Linear Chain Conditional Random Field

Overview

My linear chain conditional random field implementation achieves 83% accuracy using both the Character and Character2 feature sets. I incremented the batch size linearly and decayed the learning rate exponentially for each pass over the data. I did not manage to achieve higher than 83% accuracy for any test runs using the vanilla Viterbi algorithm.

Experiment results

I decided to run the first experiment. I modified my maximum entropy classifier from PA2 to accommodate sparse vectors and sequence data. I essentially just took each sequence of characters and flattened them out, then trained the classifier on single characters. Doing so yielded unexpectedly accurate results. I achieved 88% and 93% on the Character and Character2 feature sets respectively.

Not only was the accuracy much higher for my maximum entropy classifier than my linear chain conditional random field, but the performance was much better as well. Both took 5 total iterations over the data with no short-stopping, but maximum entropy took 1-2 minutes, where linear chain conditional random field took 5-6 minutes.

I am not sure why maximum entropy outperformed linear chain CRF by so much. It could be because of a bug in my linear chain CRF, poorly tuned hyper-parameters (learning rate, batch size, etc.), the nature of this sequence labeling task, or something else altogether. Either way, it appears that maximum entropy demonstrates more-than-acceptable accuracy performance while not taking too long computationally.

Biggest challenges

The first challenge I faced was getting myself to understand the meaning of all the indexes in the context of the transition matrices and the alpha/beta/viterbi trellises. I ran into several bugs along the way, some of which were easy to debug, and some of which were more difficult.

The next big bug I had to deal with was that I was returning an appended string for decode, rather than a list of numbers, and was consequently getting 0% accuracy. This took me far longer than it should have to properly debug.

The final challenge I ran into was that the model would seem to reach ideal accuracy on the dev-set, but then start to decrease. I thought about implementing a short-stopping algorithm to detect these decreases and store the last best model parameters, but instead decided to take a different approach. I made the batch-size scale linearly upward and the learning rate decay exponentially for each iteration through the data. I also set the fixed number of passes through the entire data set to 5.