Machine Learning Fall 2017 Homework 5

General Instructions

Homework must be submitted electronically on Canvas. Make sure to explain your reasoning or show your derivations. Except for answers that are especially straightforward, you will lose points for unjustified answers, even if they are correct.

General Instructions

You are allowed to work with at most one other student on the homework. With your partner, you will submit only one copy, and you will share the grade that your submission receives. You should set up your partnership on Canvas as a two-person group by joining one of the preset groups named "HW5 Group n" for some number n.

Submit your homework electronically on Canvas. We recommend using LaTeX, especially for the written problems. But you are welcome to use anything as long as it is neat and readable.

For the programming portion, you should only need to modify the Python files. You may modify the iPython notebooks, but you will not be submitting them, so your code must work with our provided iPython notebook.

Relatedly, cite all outside sources of information and ideas.

Written Work: Project status update

With your project group, submit a short report on the following three aspects of your project. This report should not be much longer than one or two pages.

1. (5 points) Write about a paragraph on literature related to your project. Describe what you have learned since the project proposal.
2. (5 points) Briefly describe preliminary results from your project. This point can include insights from exploration of the data or the models and software you plan to use. Or ideally, it can discuss results from initial experiments using baseline methods or early versions of your idea.
3. (5 points) Your plan moving forward. At the time you submit this status report, you will have roughly three weeks until the project is due, with Thanksgiving Break in the middle. Your plan should have a timetable and responsibilities of your group members that includes the research, the experimentation, the analysis, and creating the video presentation and website.

Programming Assignment

For this programming assignment, we have provided a lot of starter code. Your tasks will be to complete the code in a few specific places, which will require you to read and understand most of the provided code but will only require you to write a small amount of code yourself.

You will build a tool that will train a Markov chain from text data, be able to generate semi-realistic text, and identify text from different authors. All the parts you need to complete are in the file `markov.py`. To see how each bit of code should work, see how we use it in `run_text_experiments.ipynb` and the unit tests in `test_mm.py`.

The type of data we will be using is sequence data, where we a collection of data $\{x_1, \ldots, x_n\}$, and each example x_i is actually a sequence of tokens $x_i[1], \ldots, x_i[t_i]$. For the primary application of this tool, each token represents a possible word of text. (We won't use fancy text parsing, so we'll consider any string of characters that are not whitespace to be a token, so we'll treat strings like "hello" and "hello!" as distinct tokens.) The function `parse_text` reads in a text file as a list of paragraphs, treating each paragraph as a sequence.

1. (5 points) Complete the first missing part (1 of 2) of the function `train_markov_chain`. You should compute the probability of a token starting each sequence. Compute this by finding the fraction of times a sequence/paragraph begins with a particular token. Store these estimated probabilities in the dictionary `prior`. Don't store entries for tokens that never start a sequence. Since we're dealing with a large vocabulary of possible tokens, we'll need these dictionaries to be sparse for us to be able to compute with them efficiently.

2. (5 points) Complete the second missing part (2 of 2) of the function `train_markov_chain`. Here you should compute the probability of each token following a previous token. To do this, again, compute the fraction of times each token follows a future token. Do this efficiently by scanning through the training data only once in order, keeping count of how often each "state transition" from token to token occurs. Then normalize those counts to form proper conditional probabilities that you store in the dictionary called `transitions`. Again, do not store entries for zero-probability events. Be aware that the parsing function treats the end of each sequence specially, appending a "stop token" (newline) as a special token to signal that the we've reached the end of a sequence.

3. (5 point) Complete `sample_markov_chain`. Given a model output by the training function, sample a series of sequences from the Markov chain. The generated sequences should sample from the prior distribution first to get a starting token, then sample from the transition probabilities to get each subsequence token until either it has sampled the maximum length tokens or it samples a stop token. Once you complete this function, the iPython notebook should allow you to import the example text files and generate text that resembles the source material. Feel free to import other files to see if it generates realistic output when training on other text you're interested on trying.