Elizabeth Brennan

Machine Learning – Individual Project

Viterbi and Baum-Welch

My Individual project for Machine Learning was a study of the algorithms of Baum-Welch and Viterbi, and resulted in Matlab functions for both algorithms. Both Baum-Welch and Viterbi are “table-filling” processes, in which dynamic programming techniques are used to make inferences about a Hidden Markov Model (HMM) and a given observation sequence. These algorithms come up in the context of Expectation Maximization, a Machine Learning approach in which some process provides an estimate of sought-after information, such as the probabilities of a state transition matrix for a HMM, and the estimation process is run again, this time using the probabilities generated from the past iteration. Eventually, the iteration process will cease to make meaningful improvements on the estimates, and converge to yield the final values for the problem.

The Baum-Welch algorithm is also known as the “Forward-Backward” process because it achieves its purpose of estimating a HMM’s state transition matrix and observation probabilities for each state by generating a table of “forward probabilities” and a table of “backward probabilities”. In keeping with the philosophy of EM, the new estimates of the HMM parameters from the tables are put back into the Forward-Backward algorithm, to continue the re-estimations until convergence. This process provides a way to acquire a transition matrix and observation probabilities for a HMM given only a sequence of observations. One important piece to note is that Baum-Welch does not find the global maximum for a model. Therefore, the estimates for the HMM that are produced by Baum-Welch should be checked against other estimates from other initial parameters through “hill climbing” approaches to achieve an optimal solution.

The Viterbi algorithm is similar to the table-filling process of the Forward probability generation of Baum-Welch, but has a different goal: to find the most probable path through the hidden states of the HMM given the observation sequence. I particularly liked using Matlab for Viterbi because its vector functionalities made it easy to find the maximum probability at each timestep and keep track of the most probable path.

I am satisfied with the results of this project. I purposefully wrote the logic of the Matlab functions to support reuse with a variety of datasets, and I especially like that the Baum-Welch program facilitates the case when the HMM parameters are completely unknown by setting random (actually, a uniform distribution) values for the initial parameters. The Hidden Markov Model requires a very specific set-up of hidden states and observation states, and “real life” problems that could benefit from HMM analysis rarely present themselves in the exact form required for the HMM. By merely having a set of observations and the hidden states for which you want to classify them, the Baum-Welch approach can fill in the necessary information and create an estimate for a full-fledged Hidden Markov Model.

Developing the algorithms was a labor-intensive task. The coding process itself was not any more interesting than usual, but troubleshooting and checking that the equations were referencing the correct variables required an intimate knowledge of the intent of each step of the algorithm and careful attention to nomenclature, especially in the Baum-Welch table-filling and parameter re-estimation processes. Fortunately, there are many schools/professors who publish lecture slides on the topics of this project, and I was able to use them to gain a deep understanding of the algorithms and the goals they achieve. One of the most helpful online resources was an Excel spreadsheet example of Baum-Welch by Jason Eisner of John Hopkins University. By implementing his example HMM scenario in my program, I was able to use his forward-backward table calculations to check my work. By the end of the project, I had read so many professors’ lecture slides and worked through so many examples of Baum-Welch that I felt like I could teach the subject.

A note should be made about the lack of a dataset associated with this project submission. I wrote my original project design as a classification problem on data generated by the robot I am using for ECE department research. At the time project proposals were due, I thought it would be easy to extract datasets from runs of the robot, although I had not implemented the code to actually do so. In the weeks before the project, I developed a C++ program that “eavesdrops” on the ROS (Robot Operating System) messages passed between processes on the robot. I became stuck in the development cycle when, despite extensive research, my message consumer could not receive any messages from the relevant process. When the final week of regular classes came and I was still unable to overcome the problem, I decided to switch gears and apply Machine Learning to another dataset I compiled and used in a previous project.

My “senior project” for my bachelor’s degree was a study of population migration between the City and the nearby Metro counties in Saint Louis over the years since 1980. My data was compiled from US Census reports and IRS tax returns for the Missouri-side of the Saint Louis Metro Area. The data originally came in a variety of files: Census reports came from PDFs, and IRS data came from annual Excel spreadsheets. The areas were usually identified in Census reports by name, and in IRS reports by zip code. I had to create a program in R to identify the zip-codes as corresponding to the counties of St. Louis City, St. Louis County, Jefferson County, St. Charles County, Lincoln County, and Warren County. Other formatting eventually resulted in a dataset of 7 tables (one for each county/area) with 30 years’ worth of annual counts of the number of households leaving a particular county and their destination county (within the “closed” system of the Metro area).

When I decided to use this dataset as a new Machine Learning topic, I combined the 7 tables into a Markov Model, taking the averages of migration over the years to condense the data into a 7x7 matrix of transition probabilities. At this point, I tried to implement Viterbi on the Markov Model by creating a “fake” observation sequence and the hidden state categories of “I Will Move This Year” and “I Will Not Move This Year”. I asked for help on the set-up, and it was determined that the dataset did not really fit the Hidden Markov Model mold, and therefore should not be forced into an algorithm designed specifically for an HMM. At this point, it was recommended that I implement Viterbi and Forward-Backward in an easy-to-use, flexible program.

“Lessons Learned” from this project include the importance of getting to know a dataset before choosing an algorithm to analyze the data. I could have saved time and effort by noticing that the population data was a straight-forward Markov Model, and would not naturally support a Hidden Markov Model analysis. I also worked through the Viterbi and Baum-Welch algorithms, and became comfortable enough with them to understand the motivations behind each step. Even though I understand the “how”s and “why”s of Baum-Welch, the end result – the convergence of the new parameter estimates - still feels a little like magic, and I am glad to have worked on this project.