Programming Project 4: Hidden Markov Models

CS 181, Fall 2023

Code Out: Nov. 27 **Application Out:** Dec. 1
Due: Dec. 9, 11:59 PM

1 Task

In this assignment, you will implement the Viterbi algorithm for inference in hidden Markov models.

2 HMM Specifications

You will implement the Viterbi algorithm to identify the maximum likelihood hidden state sequence. You will be given a transition matrix, an initial state probability vector, and an observation sequence. Your program shall then output the maximally likely sequence of hidden states that produced the observation sequence, and on a second line, the probability of the model emitting such a sequence and following the maximally likely sequence of hidden states.

2.1 Setup

To grab the support code, see the website. We've provided an example shell script, `viterbi.sh` and a few test cases.

2.2 Reminders About Programming Language

Our Gradescope autograder is currently configured to accept solutions written in Python 2, Python 3, Java, R, and Julia.

Your solutions generally should not require the installation of any packages that do not come in the standard installations of your chosen programming language. However, if you are using Python, you will also have access to numpy and pandas.

To facilitate anonymized & automated grading, each of your solutions must be accompanied by a shell script. Make sure each problem is able to output the correct result using the shell script provided. The shell script you turn in must explicitly call your language's compiler or interpreter. If you are using Python 3, your shell script must run your program using the command "python3" rather than "python", as "python" will run your code with Python 2.

Your shell scripts should print exactly what is shown in the examples given for each problem. If you print any extra text, you will fail our autograder and lose points. This means that if you are coding in R, you may need to print text using "cat()" instead of "print()".

Be sure to print any terminal output to stdout, which is the channel that the standard print functions write to in most programming languages. If you print to stderr, your solution will be interpreted as an error message and fail the autograder.

2.3 Shell Scripts

You must provide a shell script, `viterbi.sh`, that accepts two arguments and can be run using the following command:

```
> sh viterbi.sh config obs
```

... where `config` will be the path to a configuration file, described below, and `obs` will be the path to an observation sequence file.

The program you hand in should not throw exceptions for any valid inputs.

2.4 Input and Output Format

The configuration file will have the following format:

- Line one contains the number of hidden states, a comma, and the number of observations.
- Line two contains a comma separated list of hidden state names.
- Line three contains a comma separated list of observation names.
- Line four contains the initial state probabilities.
- Line five contains the transition matrix, columns comma separated and rows semicolon separated. Transitions go from the row's hidden state to the column's hidden state.
- Finally, line six contains the observation matrix, similarly separated. Each row represents a hidden state, while each column represents an observation.

In your output, all probabilities should be printed in scientific notation rounded to 4 decimal places. For example, a probability of 1 should be printed as "1.0000e+00". If you are working in Python, we recommend using exponent-format (e-format) strings.

An example model and several input sequences and solutions follow. You should make your own tests and perform additional testing, with particular emphasis on edge cases.

`config`

```
2, 4
A, B
A, T, G, C
0.5, 0.5
0.75, 0.25; 0.25, 0.75
0.3, 0.3, 0.2, 0.2; 0.25, 0.25, 0.25, 0.25
```

`obs1`

```
ATATAAAGCACCGTTGCG
```

Output:

```
> ./viterbi.sh config obs1
AAAAAAABBBBBBBBBBB
6.5325e-14
```

`obs2`

```
ATATAAAGCACCGTTGC
```

Output:

```
> ./viterbi.sh config obs2
AAAAAAAAAAAAAAAAA
3.7877e-13
```

`obs3`

```
TATAAAGCACCGTTGC
```

Output:

```
> ./viterbi.sh config obs3
AAAAAAAAAAAAAAAA
1.6834e-12
```

`obs4`

```

```

Output:

```
> ./viterbi.sh config obs4

1.0000e+00
```

2.5 README

You must include a README file. Include the following information:

- A description of any known bugs.
- Anything you want the TAs to know about your project.

3 ExploreInformatics: Part 2

In this applications section, you will train a Hidden Markov Model to perform the important task of motif recognition. This process utilizes the Baum-Welch algorithm - a variant of the Expectation-Maximization algorithm.

This portion of the project is currently set to be released on **Friday, December 1st.** It will still have the same due date of **Saturday, December 9th**. Please include any relevant information in a file labeled **application.pdf**. This is **required** for us to grade your project.

4 Handin

To hand in this assignment, upload your shell scripts along with all files needed to run your code on Gradescope. If you have any subdirectories or folders that you would like to preserve in your handin, you will need to compress your submission into a zip file and upload the zip file to Gradescope. However, to ensure that the autograder runs your handin properly, make sure that all shell scripts are present at the root of your handin; in other words, do not place your shell scripts inside any folders.

When you hand in your solution, our autograder will immediately run on your submission. We have made the test cases provided in the project handout immediately visible to you so that you can ensure your solution runs properly with the autograder. If you fail the autograder and cannot determine why, notify a TA and we will help you to diagnose the problem. If your final submission is not compatible with our autograder, it will be very difficult for us to give you credit for this assignment.

5 Grading

We will grade your handin using pre-generated test cases with a certain number of points allocated per test case. Test edge cases extensively!

ExploreInfromatics: Part 2 will also make up at around 30-45% (TBD) of your grade for Project 4. Please do not wait to the last moment to start it.