Programming Assignment

Automata Theory Monsoon 2025, IIIT Hyderabad

August 14, 2025

Total Points: 150 Points
Deadline: 9 September 2025

General Instructions: Read the input and output formats very carefully since the evaluation will be **scripted**. The goal is not to stress test your code but to see if you have understood the underlying concepts used in the implementation. There will be an evaluation/viva after the submission, and a part of the grade would depend on it.
Language constraints: The submission must be in Python. For any further use of other libraries please refer to one of the TAs for approval. Usage of any other language will lead to no evaluation.

---

1 Hidden Markov Model

A Markov chain is a model that tells us something about the probabilities of sequences of states, each of which can take on values from some set. These sets can be words, or symbols representing anything, like the weather. Markov chains assume that future outcomes depend only on the current state the system is in and not on what states were traversed earlier. Real life scenarios involve events that are hidden, and we have access to only a certain set of observables. In such cases, we can still draw useful information by modelling the setting as a Hidden Markov Model.

In class we have seen Deterministic Finite Automatas (DFAs), where the next state is determined by the state the DFA is in and the symbol it reads. When these transitions are probabilistic, then we call it a Probabilistic Finite Automata (PFA). Similarly, we can also define a model of computation called the finite state transducer (Refer pg. 87, Ex. 1.24, 3rd edition Introduction to Theory of Computation, Michael Sipser).

We can use Probabilistic Transducers to model Hidden Markov Models with the following 5-tuple: $\langle Q, A, O, B, q_0 \rangle$ where,

1. Q = set of N states
2. A = an $N \times N$ transition probability matrix, where $A_{i,j}$ is the probability of moving from state i to state j such that $\sum_{j=1}^{N} A_{i,j} = 1$ for all i.
3. O = set of M observables, the values printed on the tape of the transducer
4. B = an $N \times M$ matrix with $B_{i,j}$ = probability that observation j is made when we are in state i. So here $\sum_{j=1}^{M} B_{i,j} = 1$. These are also called emission probabilities.
5. $q0$ = starting state in Q.

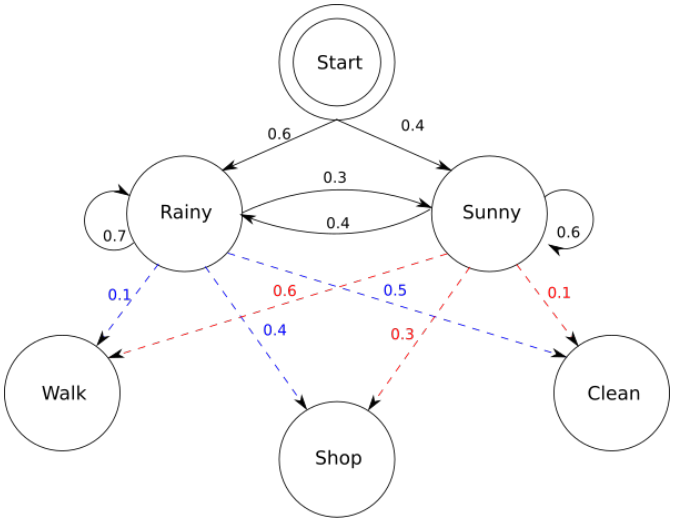

Figure 1: Hidden Markov Model Example

Note how the way we have modelled HMMs here, the transitions only contain probabilities and not symbols of an alphabet (this can be also be defined if required, but for our purpose we will avoid this additional information).

For example, diagram 1 represents a Hidden Markov model where the set of states is $\{Start, Rainy, Sunny\}$. The set of observables is $\{Walk, Shop, Clean\}$.

The transition $A_{Rainy,Sunny}$ tells us the probability that the next day will be Sunny given that the current day is Rainy (which is 0.3). The transition $B_{Rainy,Walk}$ tells us the probability that the person will go for a walk on a Rainy day (which is 0.1). If the person is at Start on day 0, then the probability that they will clean their house on the second day is:

$$0.6 * 0.7 * 0.5 + 0.6 * 0.3 * 0.1 + 0.4 * 0.4 * 0.5 + 0.4 * 0.6 * 0.1 = 0.332$$

Note that the person starts on Start in Day 0, and on Days 1 or 2, the state could be Rainy or Sunny.

Tasks

1. [20 points] **Constructing the Hidden Markov Model**

   Input Format:

   Each file contains multiple data points. The first line contains an integer T, which is the number of data points in that file. Each data point has 2 lines.

   1. The first line contains D integers representing the states involved in that run.
   2. The second line contains D integers representing the observables for each of the states in the run.

   The number of states involved in each run could be different, that is, for each data point, the value of D might change depending on how many days were recorded in that data point.

   The first integer in both these lines is always 0 as we always start at the start state, and we define the observable 0 to be observed if and only if we are at the start state. The walk length (number of days) for each data point can be different.

Your task is to read this data and construct the Hidden Markov Model according to the frequency of transitions observed. For example, if a run contains a transition from $1 \to 3$, then $A_{1,3}$ should be computed using the formula,

$$A_{1,3} = \frac{\#(1 \to 3)}{\sum_{i \in Q} \#(1 \to i)}$$

where $\#(i \to j)$ is the number of times the HMM transitions from state i to state j.

For example, consider the following dataset. Here we have 2 data points, each with walk length = 4.

```
2
0 1 2 1
0 2 2 1
0 3 2 2
0 1 1 1
```

Here, you need to infer that $Q = \{0, 1, 2, 3\}, q0 = 0$ and $O = \{0, 1, 2\}$. Then you need to compute A and B and store them in a text file according to the output format.

Output Format:

Output the matrices A (which is $N \times N$) and B (which is $N \times M$) as follows: The first N lines describe the matrix A, where each row has N space separated elements of that row in A. Similarly, the next N lines describe B, where each row has M elements.

For example, the solution to the dataset given above is,

```
0 0.5 0 0.5
0 0 1 0
0 0.5 0.5 0
0 0 1 0
1 0 0
0 0.5 0.5
0 0.66667 0.33333
0 1 0
```

You are given 2 datasets. The first is corresponding to diagram 1, while the second one corresponds to a much larger HMM.

1. Dataset 1 [5 points] Save your output as "sol-1-1.txt".
2. Dataset 2 [15 points] Save your output as "sol-1-2.txt".

Link to the datasets: [Link]

During evaluation, we will test your outputs of A, B with the actual solution to a precision of 5 digits.

2. [20 points] **Making Predictions**
Given a series of observables over different days, find the underlying states the Probabilistic transducer was present in, with the highest probability. For example, if the observables over 2 days is Walk, Walk, what was the weather on the 2 days? Let us suppose in the HMM described by the image, Walk corresponds to 1, Shop is 2 and clean is 3. Similarly, Start is 0, Rainy is 1, Sunny is 2.

Sample Input and Output:
You program (*predictions.py*) should take as input a file (*input.txt*) containing multiple test cases. And the program is expected to output a file ($sol-2.txt$) where line i is your output file is the answer to test case i

```
Input:
1       // this number will be only 1 or 2. This represents
        // the HMM you will be using. You will use A, B
        // calculated for dataset 1 in this case.
2       // number of test cases
2       // number of observations for test case 1
1 1     // This means the observables on day 1 and 2 are Walk and Walk
1       // number of observations for test case 1
2       // This means the observables on day 1 is Shop

Output:
2 2     // Underlying states on day 1 and 2 were Sunny and Sunny (with highest probability)
1       // Day 1 was Rainy
```

Explanation

Day1	Day2	Probability
Sunny	Sunny	$P(S_1) \times P(W_1\|S_1) \times P(W_2\|S_2, S_1) \times P(S_2\|S_1)$ $= 0.4 \times 0.6 \times 0.6 \times 0.6 = 0.0864$
Sunny	Rainy	$P(S_1) \times P(W_1\|S_1) \times P(W_2\|R_2, S_1) \times P(R_2\|S_1)$ $= 0.4 \times 0.6 \times 0.1 \times 0.4 = 0.0096$
Rainy	Sunny	$P(R_1) \times P(W_1\|R_1) \times P(W_2\|R_1, S_2) \times P(S_2\|R_1)$ $= 0.6 \times 0.1 \times 0.6 \times 0.3 = 0.0108$
Rainy	Rainy	$P(R_1) \times P(W_1\|R_1) \times P(W_2\|R_1, R_2) \times P(R_2\|R_1) =$ $0.6 \times 0.1 \times 0.1 \times 0.7 = 0.0042$

Here R_i = It was Rainy on Day i, S_i = It was Sunny on Day i, W_i = The person went to walk on day i.

The most probable combination of the underlying states given the list of observations is Sunny, Sunny, because from the table, it is clear that this specific combination has the highest probability among the 4. This is called Maximum a-posteriori decoding.

2 Compiler with tokenization and CFG parsing

1. [50 points] **Syntactic Analysis**
   Link to the boilerplate

   You are tasked with building a compiler for a simple programming language. The language supports the following token types: identifiers, keywords, integers, floating-point numbers, and symbols. **NOTE: You have been provided a boilerplate code which already parses the program.**

You are expected to implement a function that takes as parameters the list of tokens and performs syntactic analysis

Tokenization: Your compiler should perform tokenization (lexical analysis) of the source code using Finite State Automata (FSAs) for each token type. The compiler should be able to recognize and classify the tokens as identifiers, keywords, integers, floating-point numbers, and symbols present in the source code.

Token Hierarchy: If the source code contains any keywords, identifiers, or numbers that match the same patterns, prioritize the matching based on token hierarchy. For example, if 'if' is a valid identifier, but it is also a keyword, it should be recognized as a keyword.

Syntactic Analysis: After tokenizing, the compiler should also be able to perform some syntactical analysis according to the grammar provided in the question.

2.1 Rules for Syntactic Analysis

1. S → statement
2. statement → if (A)| (statement)(statement) | y
3. y ∈ statement alphabets [$\Sigma_{statement}$]
4. A → (cond)(statement) | (cond)(statement)(else)(statement)
5. cond → (x)(op1)(x) | x
6. op1 → + | - | * | / | ^ | < | > | =
7. x → $\mathbb{R}$ | cond | y

NOTE: $\Sigma_{statement}$ = numbers ∪ keywords ∪ identifiers - 'if' and 'else'. It does NOT include operations ("if 2+print print 5" is valid)
The brackets in the above grammar are only for reference, you can implement the compiler without the brackets as well

2.2 I/O format

When the program is run, it should ask for an input statement

2.2.1 Input

It is a single line of text

Output

2.2.2 No Error

If there are no errors (Syntactical or Lexical), then on each line, output
Token Type: <token type>, Token value <token value>

Token Type ∈ Identifier, Keyword, Integer, Float, Symbol
Token Value is the token itself

2.2.3 Lexical Error

Raise the error and give a brief description about the error
Eg: ValueError: Invalid identifier: 2xi
Identifier can't start with a number

2.2.4 Syntactic Error

When the compiler detects a code which does not follow the mentioned grammar, raise an error.

2.3 Sample Test Cases

Input

```
if 2+xi > 0 print 2.0 else print -1;
```

Output

```
Token Type: KEYWORD, Token Value: if
Token Type: INTEGER, Token Value: 2
Token Type: SYMBOL, Token Value: +
Token Type: IDENTIFIER, Token Value: xi
Token Type: SYMBOL, Token Value: >
Token Type: INTEGER, Token Value: 0
Token Type: KEYWORD, Token Value: print
Token Type: FLOAT, Token Value: 2.0
Token Type: KEYWORD, Token Value: else
Token Type: KEYWORD, Token Value: print
Token Type: SYMBOL, Token Value: -
Token Type: INTEGER, Token Value: 1
Token Type: SYMBOL, Token Value: ;
```

Input

```
if 2xi > 0 print 2.0 else print -1;
```

Output

```
ValueError: Invalid identifier: 2xi
Identifier can't start with digits
```

Input

```
else print 2.0 if 2>0 print 11;
```

Output

```
SyntaxError: 'else' occurs before 'if'
```

3 Grading and Submission Details

3.1 Grading Scheme

1. Question 1 (50 points)
   (a) Task 1 (20 points)
      i. Dataset 1 (5 points)
      ii. Dataset 2 (15 points)
   (b) Task 2 (20 points)
   (c) Viva (10 points)

2. Question 2 (100 points)
   (a) Syntactic Analysis (80 points)
   (b) Viva (20 points)

Submission details

A new portal will be created for the Programming Assignment. All your files should be placed in 1 zip file named as "<Roll Number>.zip". The folder "<Roll Number>" **must** follow this format for the automated grader to work.

```
<Roll_Number>/

 Question-1/
    construct.py        # Task 1 implementation
    predictions.py      # Task 2 implementation
    sol-1-1.txt         # Task 1 solution file
    sol-1-2.txt         # Task 1 solution file

 Question-2/
    q2.py               # Implementation for Question 2

 README.py               # Brief explanation of solutions
```