SPL-1 Project Report, 2019

Hidden Markov Model

SE 305 : Software Project Lab I

Submitted by

Md. Ibrahim Khalil

Roll No. : BSSE 1009

BSSE Session: 2017-2018

Supervised by

Dr. Ahmedul Kabir

Designation: Assistant Professor

Institute of Information Technology

Institute of Information Technology

University of Dhaka

29-05-2019

Table of Contents

1.Introduction

Hidden Markov Models (HMM) are a class of probabilistic graphical model that allow us to predict a sequence of unknown (hidden) variables from a set of observed variables. A simple example of an HMM is predicting the weather (hidden variable) based on the type of clothes that someone wears (observed). An HMM can be viewed as a Bayes Net unrolled through time with observations made at a sequence of time steps being used to predict the best sequence of hidden states.

The below diagram from Wikipedia shows an HMM and its transitions. The scenario is a room that contains urns **X1, X2 and X3**, each of which contains a known mix of balls, each ball labeled **y1, y2, y3 and y4**. A sequence of four balls is randomly drawn. In this particular case, **the user observes a sequence of balls y1,y2,y3 and y4 and is attempting to discern the hidden state which is the right sequence of three urns that these four balls were pulled from.**

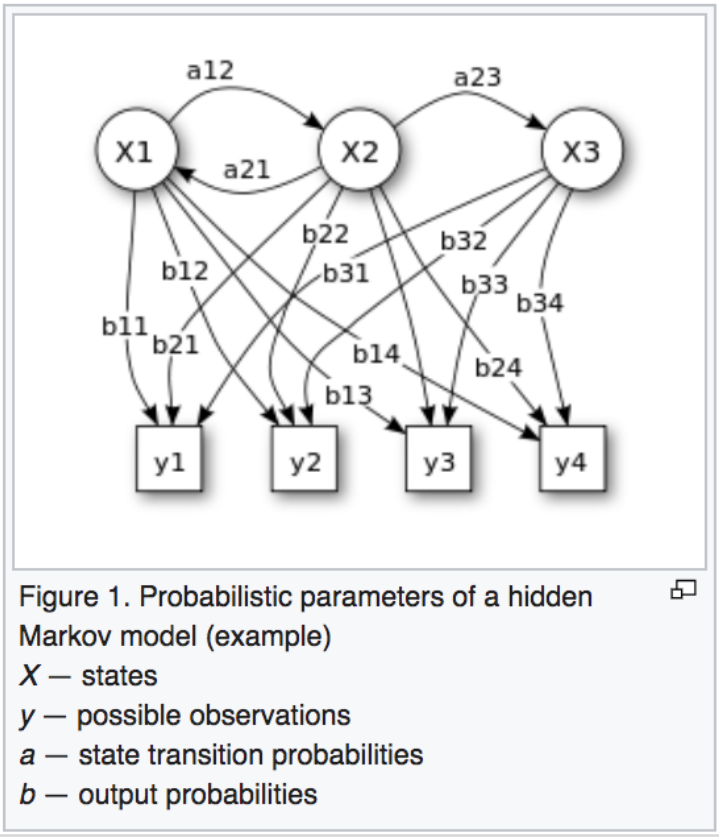


Figure 1. Probabilistic parameters of a hidden Markov model (example)
X — states
y — possible observations
a — state transition probabilities
b — output probabilities

Figure 1: HMM hidden and observed states

1.1 Background study

To Implement Hidden Markov Model I had studied :

Markov Model :

Markov chains (and HMMs) are all about modelling sequences with discrete states. Given a sequence, we might want to know e.g. what is the most likely character to come next, or what is the probability of a goven sequence. Markov chains give us a way of answering this question. To give a concrete example, you can think of text as a sequence that a Markov chain can give us information about e.g. 'THE CAT SA'. What is the most likely next character? Another application is determining the total likelyhood of a sequence e.g. which is more likely: 'FRAGILE' or 'FRAMILE'?

Markov chains only work when the states are discrete. Text satisfies this property, since there are a finite number of characters in a sequence. If you have a continuous time series then Markov chains can't be used.

The Viterbi Algorithm :

The goal of the Viterbi algorithm is to discover the optimal sequence of states given an HMM and a sequence of observations. Here, the term optimal is defined from a probabilistic perspective. In other words, the Viterbi algorithm seeks the sequence of HMM states that maximizes the joint probability of observations given the HMM.

An example is chosen from wikipedia : Consider a village where all villagers are either healthy or have a fever and only the village doctor can determine whether each has a fever. The doctor diagnoses fever by asking patients how they feel. The villagers may only answer that they feel normal, dizzy, or cold.

The doctor believes that the health condition of his patients operate as a discrete Markov Chain(Markov model). There are two states, "Healthy" and "Fever", but the doctor cannot observe them directly; they are *hidden* from him. On each day, there is a certain chance that the patient will tell the doctor he/she is "normal", "cold", or "dizzy", depending on their health condition.

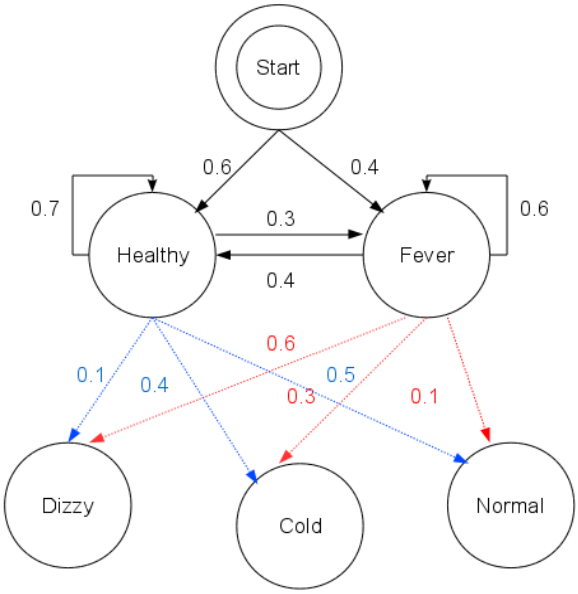


Fig : 2 Graphical Representation of this Example (HMM)

If the Doctor Observe a sequence ['normal', 'cold', 'dizzy'] and try to decoding the Hidden States {Healthy, Fever} then he should follow viterbi algorithm.

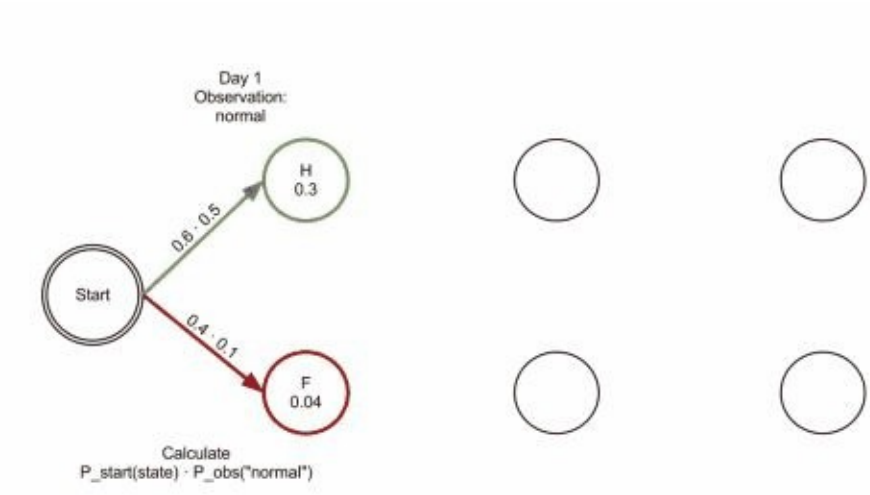


Fig : 3 Viterbi algorithm,s working procedure

Using Viterbi alogorithm the doctor can find the most optimal hidden State sequence `['Healthy', 'Healthy', 'Fever'].`

Forward Algorithm:

In Forward Algorithm (as the name suggested), we will use the computed probability on current time step to derive the probability of the next time step. Hence the it is computationally more efficient $O(N2.T)$.

We need to find the answer of the following question to make the algorithm recursive:

Given a a sequence of Visible state VT , what will be the probability that the Hidden Markov Model will be in a particular hidden state s at a particular time step `t`.

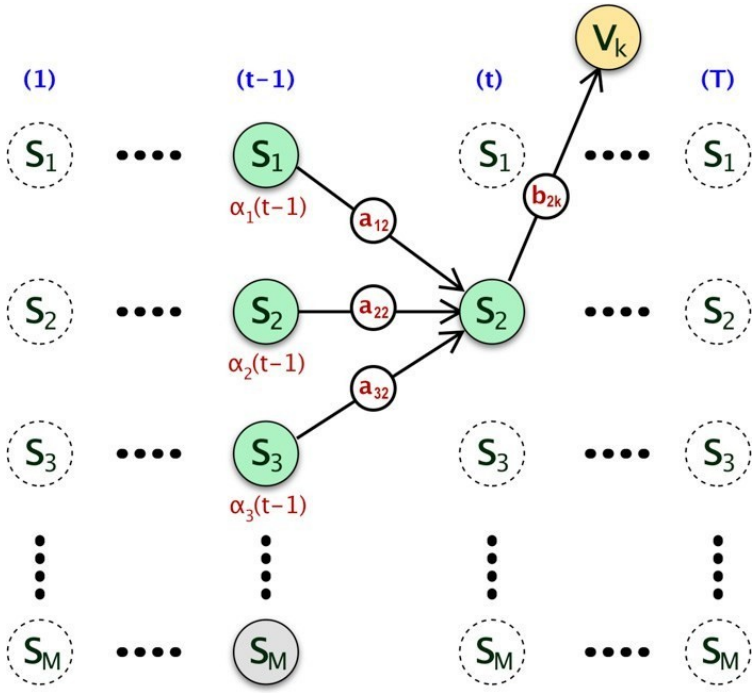


fig :4 Forward Algorithm

here,

S = possible Hidden State

t = time

V = visible state

a(i,j) = transition probability of S(i) to a S(j)

b(i,k) = Emission probability of S(i) to V (k)

Dynamic programming :

Dynamic programming is both a mathematical optimization method and a computer programming method.In both contexts it refers to simplifying a complicated problem by breaking it down into simpler sub-problems in a recursive manner. While some decision problems cannot be taken apart this way, decisions that span several points in time do often break apart recursively. Likewise, in computer science, if a problem can be solved optimally by breaking it into sub-problems and then recursively finding the optimal solutions to the sub-problems, then it is said to have optimal substructure.

If sub-problems can be nested recursively inside larger problems, so that dynamic programming methods are applicable, then there is a relation between the value of the larger problem and the values of the sub-problems .In the optimization literature this relationship is called the Bellman Equation.

1.2 Challenges

Implementing a new software solution carries with it a number of challenges. The process can be

overwhelming, confusing and lenthy. For implementing this project there are lot of challenges

that I have faced. Some of them are:

- Handling large code for the first time
- Learning and understanding algorithm
- Understand and design the project implementation
- Calculate Joint probability
- Mapping Observable Data and Hidden Data and Their Probability
- GUI Design and connect to the Project
- Train the Hidden Markov model

2. Project Overview

I have divided my whole project into three different parts. They are

- Implementation of Viterbi Algorithm
- Implementation of Forward and Backward algorithm
- Connect A Hidden Markov Model with the Algorithm

2.1 Implementation of Viterbi Algorithm

I follow the Pseudo Code of Wikipedia and implement the algorithm in Java. Here is the Source code :

```
public class viterbi {

    public static int[] compute(int[] obs, int[] states, double[] start_p, double[][] trans_p, double[][] emit_p, int
    {
        double[][] V = new double[nObs][nStates];
        int[][] path = new int[nStates][nObs];

        for (int y = 0; y < nStates; y++)
        {
            V[0][y] = start_p[y] * emit_p[y][obs[0]];
            path[y][0] = y;
        }

        for (int t = 1; t < nObs; ++t)
        {
            int[][] newpath = new int[nStates][nObs];

            for (int y = 0; y < nStates; y++)
            {
                double prob = -1;
                int state;
                for (int y0 = 0; y0 < nStates; y0++)
                {
                    double nprob = V[t - 1][y0] * trans_p[y0][y] * emit_p[y][obs[t]];
                    if (nprob > prob)
                    {
                        prob = nprob;
                        state = y0;
                        //
                        V[t][y] = prob;
                        //
                        System.arraycopy(path[state], 0, newpath[y], 0, t);
                        newpath[y][t] = y;
                    }
                }
            }

            path = newpath;
        }

        double prob = -1;
        int state = 0;
        for (int y = 0; y < nStates; y++)
        {
            if (V[nObs - 1][y] > prob)
```

2.2 Implementation of Forward and Backward Algorithm

Here the Code of Forward Algorithm in Java.

```java
public class ForwardProccesing {

	public static double[][] forwardProc(int[] obs, double [] start_p, double[][] trans_p, double[][] emit_p , int nState
	{

		double[][] probability_table = new double[nStates][obs.length];
		//double [][] kola = new double [11][12];

		//System.out.println(kola.length);
		int n = probability_table.length;

		for (int l = 0; l < n ; l++)
		{
			probability_table[l][0] = start_p[l] * emit_p[l][obs[0]];
		}

		for (int i = 1; i < obs.length; i++)
		{
			for (int k = 0; k < n; k++)
			{
				double sum = 0;

				for (int l = 0; l < nStates; l++)
				{
					sum += probability_table[l][i-1] * trans_p[l][k];
				}

				probability_table[k][i] = sum * emit_p[k][obs[i]];
			}
		}

		return probability_table;
	}
}
```

Here the Code of Backward Algorithm in Java.

```
package application.Hmm;

public class BackwardProccesing {

        protected double[][] backwardProc(int[] o , int numStates, double [][] trans_p, double [][] emit_p) {

                int T = o.length;

                double[][] bwd = new double[numStates][T];

                for (int i = 0; i < numStates; i++)
                        bwd[i][T - 1] = 1;
                for (int t = T - 2; t >= 0; t--)
                {
                        for (int i = 0; i < numStates; i++)
                        {
                                bwd[i][t] = 0;
                                for (int j = 0; j < numStates; j++)
                                        bwd[i][t] += (bwd[j][t + 1] * trans_p[i][j] * emit_p[j][o[t + 1]]);

                        }
                }
                return bwd;
               }

}
```

2.3 Connect A Hidden Markov Model with the Algorithm

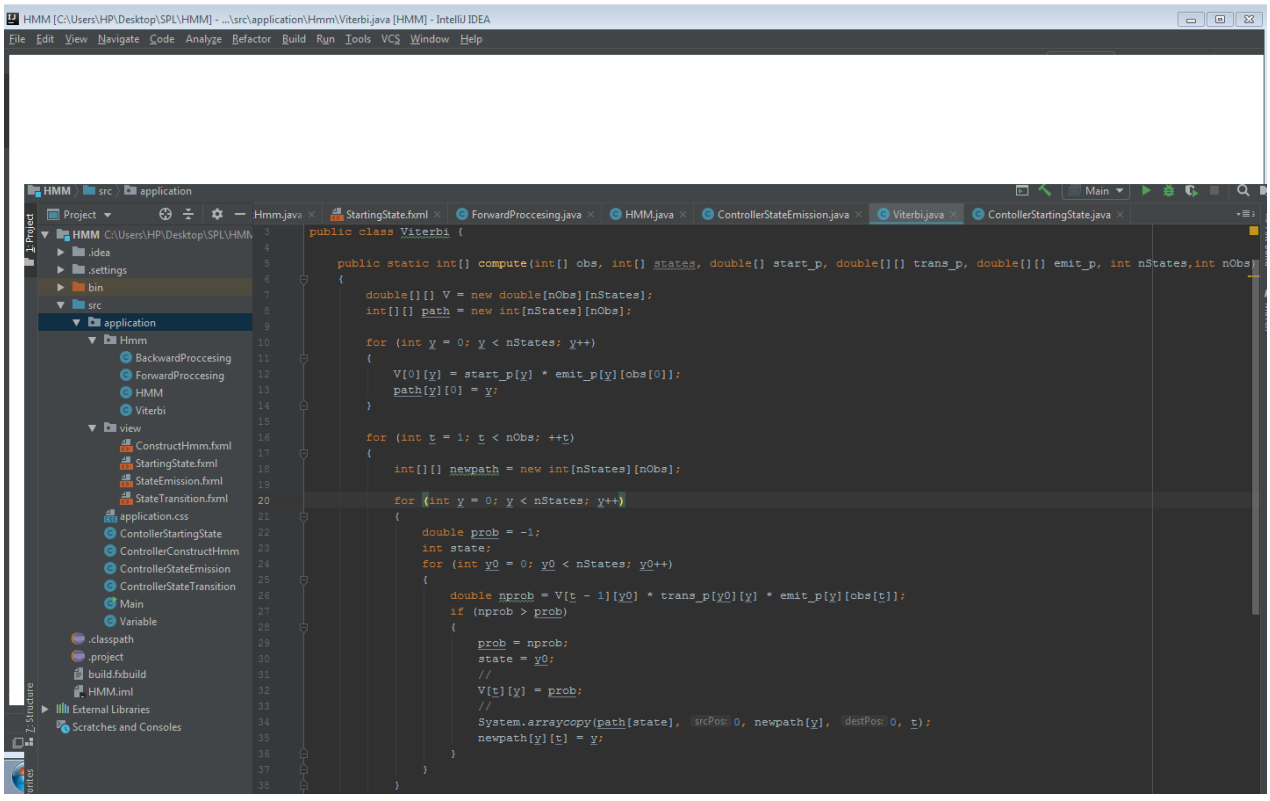
Given a Hidden Markov Model and a Observable sequence Calculate

Probability of Observation by the HMM

Most likely Hidden State Sequence

Estimate the Next Hidden State

Estimate the Current Hidden State



3. User manual

3.1 Open the executable file

Double click on the .exe file and user can see a Graphical User Interface like bellow :

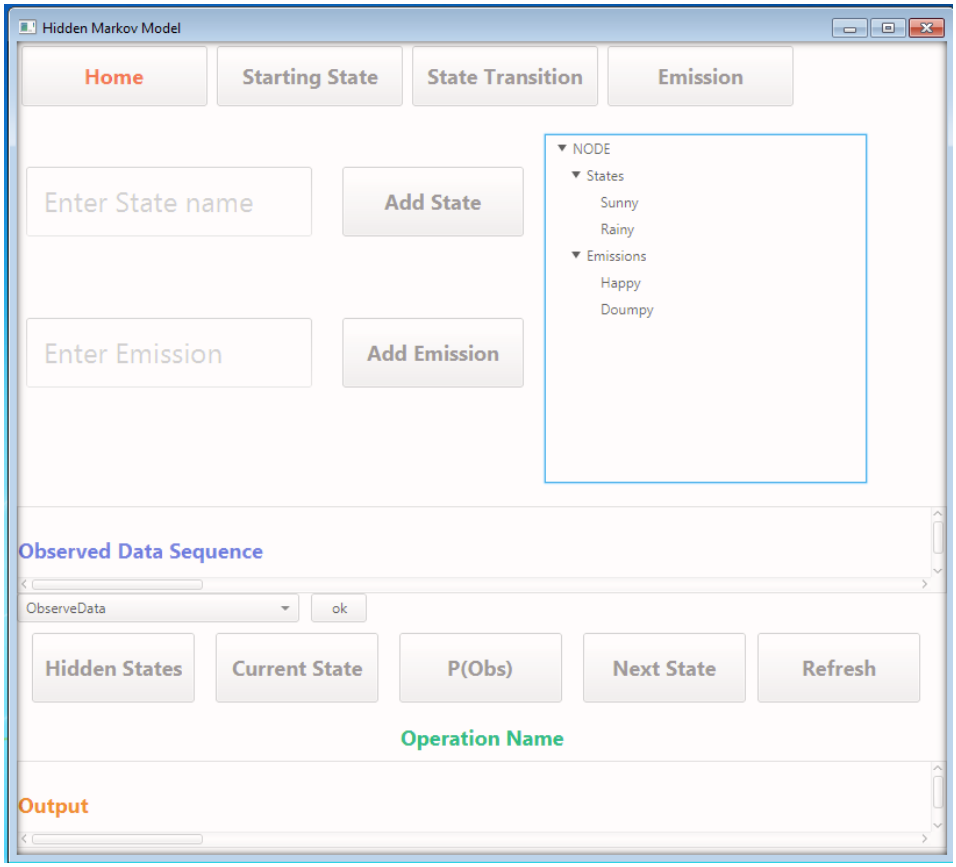
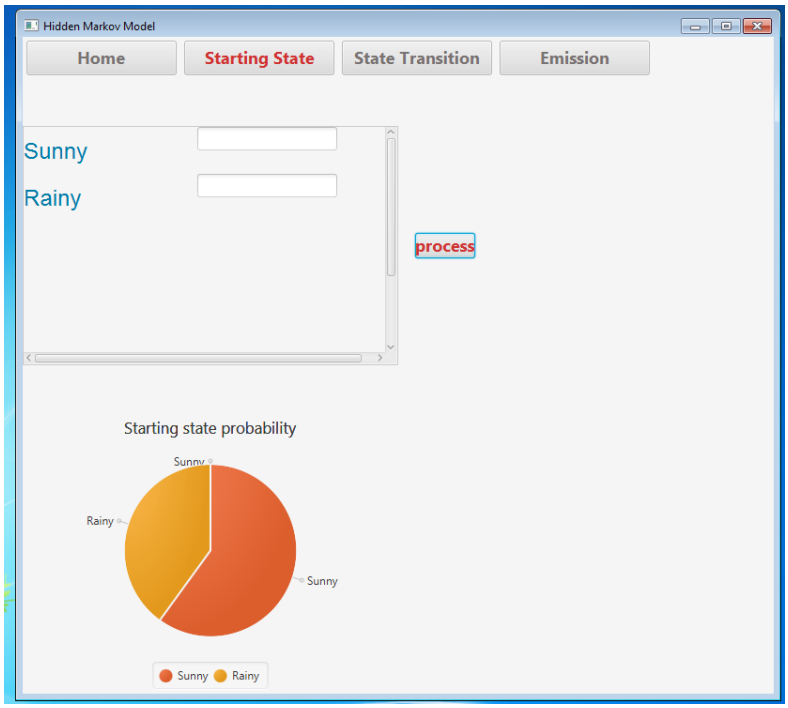


Fig : 6 GUI of HMM

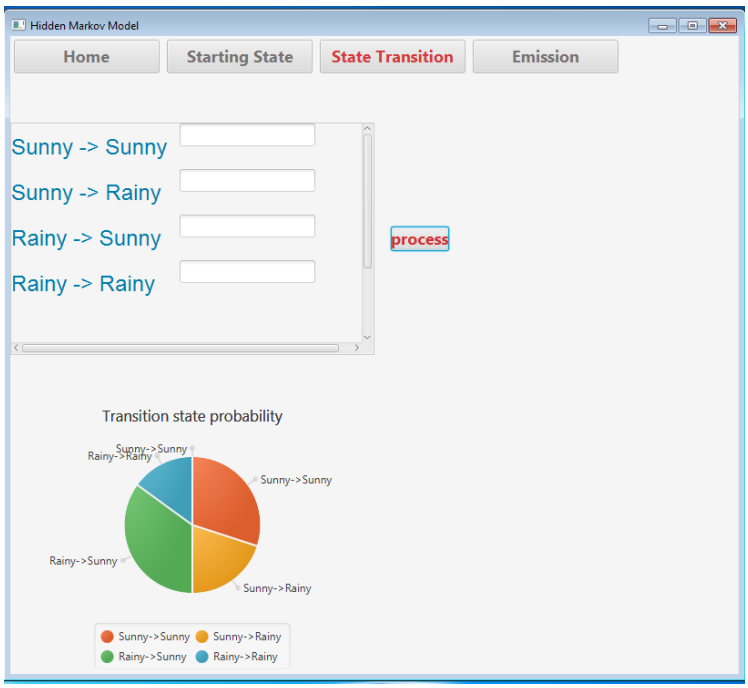
3.2 Build a HMM

Input hidden state name and observation state name in corresponding text field and press ENTER. Click on the Tree View user can see the state name.(See Fig : 6)

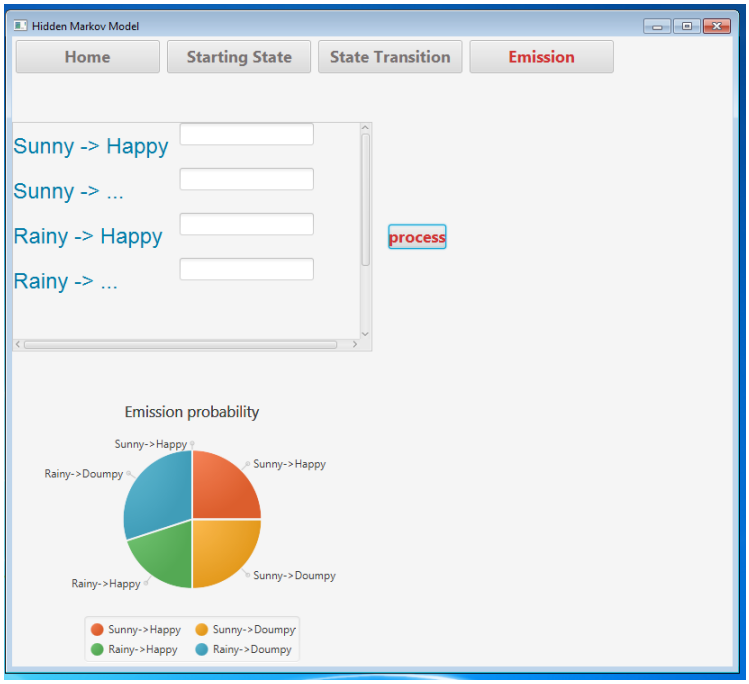
Then press the starting probability button and fill the starting probability table. Then press process button. The pie_chart show the starting state probability distribution.



Then press the Transition probability button and fill the Transition probability table. Then press process button. The pie_chart show the Transition state probability distribution.



Then press the Emission probability button and fill the Emission probability table. Then press process button. The pie_chart show the Emission probability distribution.



3.3 Input And Output

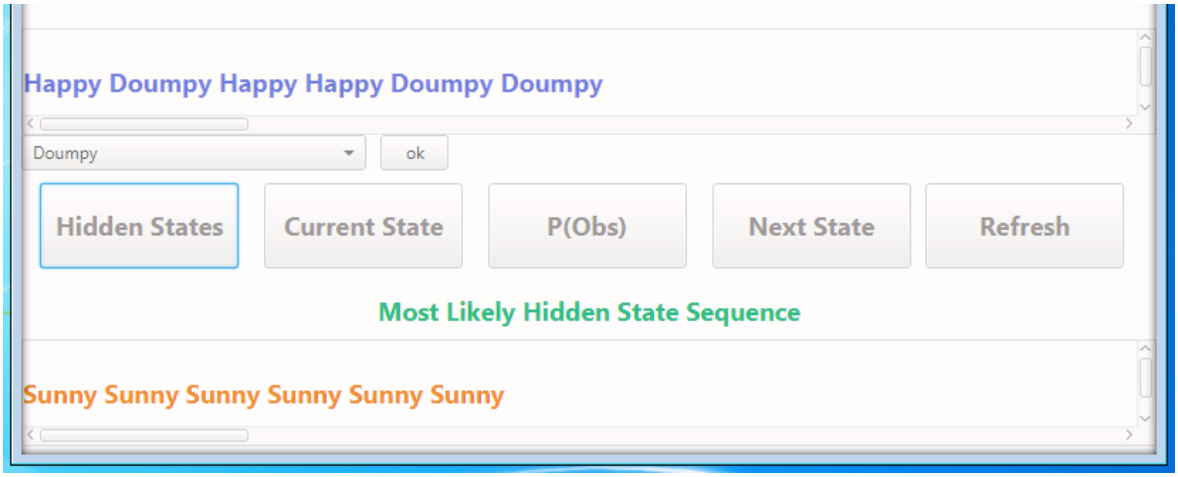
Input an Observation sequence by using ComboBox. And press the button :

Probability of Observation to Calculate probability of Observable sequence.

Current State to Estimate Current State.

Hidden State to Estimate most Likely Hidden State sequences

Next State to Estimate Next State of hidden state



4. Conclusion

Implementing Hidden Markov Model algorithm helps me to improve my coding skill and I have learned to handle large code for the first time. I hope it will help me to deal with difficulties in future. This project was quiet challenging and I gained a lot of experience from it. I have gained lots of impressions and interest about Machine learning and Artificial Intelligence . I want continue my project and build a real life Software (Sign language recognition). I want to thank my supervisor for guiding me a lot during this project.

5. Appendix

In this project I implement the Hidden Markov Model Algorithm. But here is missing the training part of Hidden Markov model. Within a Few days I want to implement the training part and use this algorithm in sign language detection.

6. Reference

- ✔ https://en.wikipedia.org/wiki/Viterbi_algorithm
- ✔ http://practicalcryptography.com/miscellaneous/machine-learning/hidden-markov-model-hmm-tutorial/
- ✔ https://www.nature.com/articles/nbt1004-1315
- ✔ http://www.adeveloperdiary.com/data-science/machine-learning/forward-and-backward-algorithm-in-hidden-markov-model/