10/14/2017

Where's Croc

Artificial Intelligence Course

Kim Hyojung & Nevine Gouda
UPPSALA UNIVERSITY

Table of Contents

Introduction

Overview on Markov Models

Markov Models is a stochastic model that can be used to represent a constantly changing system and that the system's next state doesn't depend on previous states, except the current state. And Markov Models can be split into 4 different models depending on whether the states are observable or hidden, and whether the system is controlled or not. And these models are the following:

1- Markov Chains

2- Hidden Markov Models

3- Markov decision process

4- Partially observable Markov decision process.

Where the Markov Model depends on the systems states and a transition matrix for the states. Where this transition matrix basically holds the probabilities of moving from one state to another.

$$X_{t+1} = X_t T^n$$

And due to the nature of the game in this exercise, we will only focus on the Hidden Markov Models. Since the game is considered to be autonomous and the states to be partially observable (more into that later).

Markov Models are proven to be the most helpful in systems where we want to compute the probability for the occurrence of a sequence of specific events.

Overview on Hidden Markov Models

Hidden Markov Model is statistical model for the Markov Model where the system is considered to be autonomous and the states to be partially observable, or hidden states. In other words, the system state is considered hidden, however there are observations relatable to the states that can help determine the system's current state. Where these observations are the input probabilities for the system's probabilistic model. Where these observations depend only on the system's current state (i.e. conditionally independent of all other states and observations). Therefore, HMM is considered to be quite useful when the sequence of events that we are interested in are not directly observable, i.e. hidden.

Inference on Hidden Markov Models

HMM has a couple of inference problems depending on the state of the system/application as the following:

1- Filtering
   a. It tries to calculate the model's belief on the states based on a sequence of events. In other words, it produces the most likely state at the current time step given the observed sequence.
   b. Where this problem is best approached using the forward algorithm.

2- Smoothing
   a. It tries to calculate the probability distribution for the system's current state, given a all the sequence of observations $O = O_1, O_2, O_3, \dots O_T$.
   b. Where this problem is best approached using the forward-backward algorithm.

3- Most-Probable
   a. Where it tries to reach the most probable sequence of events/states given a sequence of observations.
   b. Where this problem is best approached using Viterbi algorithm.

The HMM system's probabilistic model consists of discrete states and a single discrete observation variable that is observed across time.

In order to formalize the HMM algorithm, we have the following:

1. N number of States where each state can have one of N values.
   - Where a State i at time j is written as: $S_{i,j}$ $where\ 1 \leq i \leq N\ and\ 0 \leq j \leq T$
2. M number of observations.
   - Where an Observation i at time j is written as:

$$O_{i,j}\ where\ 1 \leq i \leq M\ and\ 0 \leq j \leq T$$

3. An initial State is written as $P\left(S_{i,0}\right)$. Where its actual value should be known to all states i.
4. The evolution of the system, or in other words the transition matrix is written as $P\left(S_{i,j+1} \mid S_{k,j}\right)$. Where its actual value should be known to all states i and k.

5. The emission probabilities, or the observation probabilities is written as $P\left(O_{i,j} \mid S_{k,j}\right)$. Where its actual value should be known to all i and k.
6. Where the $P\left(S_{i,j} \mid O, S_{i,j-1}\right) \alpha\, P\left(O_{i,j} \mid S_{k,j}\right). P\left(S_{i,j+1} \mid S_{k,j}\right)$.

Algorithm Overview

Where’s Croc game overview

The ***Where’sCroc*** game is a graph type game. Where it consists of 40 connected states/waterholes with a crocodile, the player, and 2 backpackers randomly located initially in one of these nodes. And as time passes by the crocodile and 2 backpackers keep moving in the graph. And they are only allowed to move from one waterhole to its connected waterholes. And the purpose of the game is for the player (computer) to find the crocodile before it goes unavailable and loses the game. Putting in mind that if the crocodile goes to the same waterhole as a backpacker, then the crocodile eats him and he dies. In addition, the player is only allowed to move from one waterhole to its connected waterholes. And the player can make 2 actions at a time before the others (the crocodile and the backpackers) move. Where these 2 actions can be the following:

1- Move 2 steps ahead.
2- Search the current waterhole and move only 1 step ahead.
3- Move only 1 step ahead and search the new waterhole.

Where information is given about the waterholes to help find croc. Where in each waterhole there are 3 water sensors that records the hole’s salinity, phosphate and nitrogen levels. And the crocodile only sends the sensors values from his current waterhole.
Since the position of the crocodile is unknown, while having the waterhole’s sensors observables, and we need to go to the position of the crocodile in order to find it. Therefore, we can classify this game as a Hidden Markov Model and a search graph problem.

Where we will iteratively re-calculate the probabilistic models after each move since the observations change after each iteration. And after calculating the most probable location of the crocodile after each iteration, we have to perform a searching algorithm. That is to find the most optimum way to reach the crocodile and only performing 2 actions towards that most probable location (as previously mentioned).

Following is the detailed information about both algorithms, the Hidden Markov Model and the searching algorithm.

Implemented Hidden Markov Model inference

As explained earlier, the HMM algorithm has more than 1 inference, but due to the nature of the game we used the forward inference algorithm.

Where the breakdown of the algorithm is as the following:

Where the breakdown of the HMM system's probabilistic model is as the following:

1. N is the number of States and in this game N = 40 and the States has values from 1 to 40.
   - Where a State i at time j is written as: $S_{i,j}\ where\ 1 \leq i \leq 40\ and\ 0 \leq j \leq T$
2. M is number of observations and in this game M = 3. Where the observations are the mean and standard deviation of salinity, phosphate and nitrogen.
   - Where an Observation i at time j is written as:

$$O_{i,j}\ where\ 1 \leq i \leq 3\ and\ 0 \leq j \leq T$$

3. The initial State in this game can be written as $P\left(S_{i,0}\right)$. Where in this game it is a vector of size 40, and each element has equal probability of $1/_{40} = 0.025$. In other words, the initial state is as the following:

$$P\left(S_{i,0}\right) = [0.025, 0.025, 0.025 \dots, 0.025]_{1\,x\,40}.$$

4. The transition matrix can be written as $P\left(S_{i,j+1} \mid S_{k,j}\right)$. Where the matrix's size is $40\ x\ 40$. Where each row represents a state and each column also represent a state. And row i will be = 0 to all cells except the ones the respective state is directly connected to in the graph. Where they will have the value of $1/_{n}$ where n is the number of directly connected states to the current state (including the state itself).

   For example, in the game since waterhole 1 is directly connected by an edge to waterholes 1, 2, 4 and 6. Then the first row in the transition matrix will be

$$[0.25, 0.25, 0, 0.25, 0, 0.25, 0, 0, \dots, 0]_{1\,x\,40}$$

5. The emission probabilities, or the observation probabilities is written as $P\left(O_{i,j} \mid S_{k,j}\right)$. Where in order to calculate it, we had to do the following:
   i. We gathered the previously mentioned observations, the mean and standard deviation for the salinity, phosphate and nitrogen.
   ii. We calculate the probability that the observation sent from the crocodile from the salinity sensor came from waterhole 1,2,3…40 given the mean and standard

deviation for the waterholes 1, 2, 3, …40 for their respective salinity sensor. And the same for the phosphate sensor as well as the nitrogen sensor. We computed this probability using an R built-in function called "dnorm" that simply calculates the probability density function for a standard normal distribution.

iii. The result of the previous step is a matrix of size $40\ x\ 3$ where the columns represent the salinity, phosphate and nitrogen probability distribution respectively.

iv. Then we transform the previous matrix to a vector of size $40\ x\ 1$ where the new column is obtained by multiplying the previous matrix's 3 columns together, using the following equation:

$$P_{state1} = {P_{state1}}^{salinity} . {P_{state1}}^{phosphate} . {P_{state1}}^{nitrogen}$$

6. And the probability that the crocodile is at state I at time j can be calculated by the following equation:

$$P\left(S_{i,j} \mid O, S_{i,j-1}\right) \alpha\, P\left(O_{i,j} \mid S_{k,j}\right). P\left(S_{i,j+1} \mid S_{k,j}\right)$$

Which in turn is normalized across all states, where the summation of probabilities obtained for all the waterholes/states should be equal to 1. Then the waterhole which the highest probability is the most probable waterhole that holds the crocodile. Then it is time to find the shortest path between the player's position and that most probable waterhole.

Search/Route finding Algorithm

After finding the most probable location of the crocodile, we need to find the shortest path towards that waterhole. Thus, we use a simple Breadth First Search algorithm that finds the shortest path between the current player's location and the goal (the most probable location of the hole). After that is calculated, we only take a maximum of 2 steps towards that goal, then the game continues towards the next time step and everything is recalculated.

Due to the nature of the problem, we can see that this is an unweighted, undirected path finding problem. And in order to find the shortest path, we inherited a Breadth First Search algorithm to find the shortest path. And since the node we are searching for is guaranteed too exist in the graph then the algorithm is complete and it guarantees to find the shortest path.

Tweaks and Improvements

In order to perform well/better in the game we did the following:

1- If the player is at a state n and the most probable state for the location of the crocodile has a probability greater than 0.8 and is at least 2 steps ahead then get as closer to it as possible by jumping a step. In other words, the actions to be taken are move to the next 2 states, without search. Where the search for the second state will be performed in the next iteration.

2- If the player is at a state n and the most probable state for the location of the crocodile has a probability greater than 0.8 and is at least 2 steps ahead then get as closer to it as possible by jumping a step.

3- If any of the backpackers were just eaten, then calculate the shortest path towards that state instead of the most probable.