Hidden Markov Models

Deadline For grade **A** you must latest have your implementations accepted by Kattis (and for the duckhunt with the required score) by **Wed Sept 18 2019, 11:59pm sharp** and present this in an "in-time" presentation slot Sept 19, 2019. Any assignment accepted by Kattis/presented after these deadlines will result in max. grade B.

1 INTRODUCTION TO HIDDEN MARKOV MODELS

A Hidden Markov Model (HMM) is a powerful statistical model used in robotics, computational biology and many other disciplines. HMMs build on the idea that observations are generated by unknown or hidden states. For example, a GPS signal indicating your location is generated by your actual position. Since many processes, ranging from DNA sequences to satellite coordinates, can only be observed indirectly and under high uncertainty, HMMs are useful approximations of these systems.

HMMs belong to the class of Markov processes that describe memoryless dynamical systems. Any Markov process fulfils the Markov assumption, i.e. that the future is independent of the past given the current state at time t. In terms of HMMs, given the knowledge of the hidden state $\mathbf{X_{t-1}}$, the current hidden state $\mathbf{X_t}$ is independent of all past hidden states $\mathbf{X}_\tau$ with $\tau < t-1$. Similarly, given the current state $\mathbf{X_t}$ the current observation $\mathbf{O_t}$ is independent of all past states and observations.

In this notation, $\mathbf{X_t}$ and $\mathbf{O_t}$ are random variables which means that their value depends on a probability distribution. Realizations of these random variables, which means choosing values according to these probability distributions, are denoted by $\mathbf{x_t}$ and $\mathbf{o_t}$. Each random variable can take values from a given set of outcomes, discrete or continuous. In this assignment, we will only be dealing with discrete random variables.

As an example of that, the outcome of a coin toss can be denoted by $\mathbf{O_t}$ which can take any value in the discrete set $\{head, tail\}$. The event of observing $head$ is a realization $\mathbf{o_t}$ and has probability $P(\mathbf{O_t} = \mathbf{o_t}) = P(\mathbf{O_t} = head) = 0.5$. The probability of observing any outcome, i.e. either $head$ or $tail$, has to sum to $\sum_{\mathbf{o_t} \in \{\mathbf{head}, \mathbf{tail}\}} P(\mathbf{O_t} = \mathbf{o_t}) = 1$.

Considering more than one time step, the notation $\mathbf{O_{1:T}} = \mathbf{o_{1:T}}$ describes an observed series of realizations of the random variable. In terms of the coin example, if we threw the coin three times and observed $head, head, tail$, then we would get $\mathbf{O_1} = head$, $\mathbf{O_2} = head$ and $\mathbf{O_3} = tail$.

Equipped with this notation, we can define the joint probability of a sequence of made observations $\mathbf{o_{1:T}}$ and all hidden states in the time interval $[1, T]$ as

$$P(\mathbf{O_{1:T}} = \mathbf{o_{1:T}}, \mathbf{X_{1:T}}) = P(\mathbf{X_1}) P(\mathbf{O_1} = \mathbf{o_1} | \mathbf{X_1}) \prod_{t=2}^{T} P(\mathbf{X_t} | \mathbf{X_{t-1}}) P(\mathbf{O_t} = \mathbf{o_t} | \mathbf{X_t})$$

Since the hidden states are unknown to us, they must be modelled under uncertainty. This is achieved by modelling the probability distribution over the transition from one state to another and the probability distribution over the observations given the current state. Thus, $P(\mathbf{X_t} = \mathbf{x_t} | \mathbf{X_{t-1}} = \mathbf{x_{t-1}})$ denotes

the probability of being in a specific state at time t given that the system was in a specific state at time $t-1$. $P(\mathbf{O_t} = \mathbf{o_t}|\mathbf{X_t} = \mathbf{x_t})$ is the probability of observing $\mathbf{o_t}$ given that the system is in hidden state $\mathbf{x_t}$.

At each time step $t \in [1, T]$ in the interval from 1 to T an HMM can thus be characterized by the following components:

$$\mathbf{X_t} = \mathbf{x_i}, i \in \{1, 2, ...N\} : N \text{ possible hidden states}$$
$$\mathbf{O_t} = \mathbf{o_k}, k \in \{1, 2, ...K\} : K \text{ possible observations}$$
$$\pi_i = P(\mathbf{X_1} = i) : \text{the initial probability vector } \pi \in \mathbb{R}^{1,N} \text{ with elements } \pi_i$$
$$a_{i,j} = P(\mathbf{X_{t+1}} = j|\mathbf{X_t} = i) : \text{the transition probability matrix } \mathbf{A} \in \mathbb{R}^{N,N} \text{ with elements } a_{i,j}$$
$$b_i(k) = P(\mathbf{O_t} = k|\mathbf{X_t} = i) : \text{the observation probability matrix } \mathbf{B} \in \mathbb{R}^{N,K} \text{ with elements } b_{i,k} = b_i(k)$$

As discussed above, both the hidden states and observations are random variables for which we denote the realizations by $\mathbf{X_t} = \mathbf{x_i}, i \in \{1, 2, ...N\}$ and $\mathbf{O_t} = \mathbf{o_k}, k \in \{1, 2, ...K\}$. The transition probability matrix $\mathbf{A}$ describes the probability of being in state j at time $t+1$ given that the system has been in state i at time t. Equivalently, the observation probability matrix $\mathbf{B}$ describes the probability of observing k at time t given that the system is in state i at time t. The matrix notation of the parameters simplifies the computations as the implementation of this statistical model are reduced to pure algebraic manipulations.

HMMs can be cast into different problem settings. First, if we assume that the parameter set $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ as defined above is known and that we have observed the realization $\mathbf{O_{1:T}} = \mathbf{o_{1:T}}$, we can predict the next observation. The auto-completion function for text in a mobile phone can be based on this operation - given the letters that have been observed so far, what is the most probable next letter or complete word? In this example, both the observations and hidden states are the letters of the alphabet.

Second, the probability of a given observation sequence can be estimated. Since the HMM represents the statistics of hidden states and observations, the observation of "good morninf" will have a lower probability than "good morning" and an auto-correction program can detect the mistake.

Third, in order to correct the mistake, we need to decode the actual intention of the writer. Thus, we need to determine the most likely hidden state sequence that might have produced the observation sequence.

Fourth, and finally, if the parameters are not known, the set $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ needs to be learned with help of made observations. Both the Markov assumption and the algebraic notation render this problem solvable.

1.1 Getting started

In this lab, you are required to understand and implement the different HMM problems. Along with the description of your tasks, we pose a number of questions, marked by a frame in the text. You are required to answer these in written form and present them to a teaching assistant in a lab session.

For the grades E and D, this implies to implement the above mentioned problems of HMMs, i.e. the evaluation, decoding and learning problem.

For grade C, you need to have passed the E-D level and you are furthermore required to investigate HMMs in empirical and theoretical terms.

Finally, for A and B, you need to have passed the E-D and C level. In the final assignment, you are required to apply your knowledge to an applied problem setting. We provide you with observations of the direction of flight of different birds over time. The task is to develop a system that can determine different bird species based on their flying pattern and to shoot the birds.

2 GRADE LEVEL E AND D

2.1 OVERVIEW

As introduced above, HMMs consist of probability distributions over hidden states and observations. In order to infer e.g. future observations or the most likely sequence of hidden states, it is essential to take any uncertainty about the system into account. This is achieved by marginalizing over states and observations at all relevant time points. For the discrete case, marginalization is achieved as follows:

In general terms, let $\mathbf{O} \sim P(\mathbf{O})$ and $\mathbf{X} \sim P(\mathbf{X})$ be random variables in a discrete space that can take the values $\mathbf{O} \in \{\mathbf{o}_1, \mathbf{o}_2, ...\mathbf{o}_K\}$ and $\mathbf{X} \in \{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_N\}$ respectively. The joint probability distribution over $\mathbf{O}$ and $\mathbf{X}$ is denoted by $P(\mathbf{O}, \mathbf{X})$ and the conditional distribution of $\mathbf{O}$ given $\mathbf{X}$ by $P(\mathbf{O}|\mathbf{X})$. Then the marginal probability mass distribution of $P(\mathbf{O} = \mathbf{o}_i)$ is given by

$$P(\mathbf{O} = \mathbf{o}_i) = \sum_{j=1}^{m} P(\mathbf{O} = \mathbf{o}_i, \mathbf{X} = \mathbf{x}_j) = \sum_{j=1}^{m} P(\mathbf{O} = \mathbf{o}_i | \mathbf{X} = \mathbf{x}_j) P(\mathbf{X} = \mathbf{x}_j) \tag{2.1}$$

Here we are marginalizing over the random variable $\mathbf{X}$, i.e. in the discrete setting we sum over all its values.

As an example, imagine that we have two coins, representing the hidden states $\mathbf{X}_i$, denoted by $\mathbf{c}_1$ and $\mathbf{c}_2$, depicted in Figure 2.1. We know that coin $\mathbf{c}_1$ is biased and will show $head$ with probability 0.9 and $tail$ with probability 0.1, while coin $\mathbf{c}_2$ shows $head$ and $tail$ each with a probability of 0.5. Now assume that our observation at each time point consists of the result of a toss of one of these coins, so either $\mathbf{o}_1 = head$ or $\mathbf{o}_2 = tail$ but we do not know which of the two coins was used. In this example, the hidden states $\mathbf{X}_t$ are the identity of the coin at time t and the observations $\mathbf{O}_t$ are the observed outcome of $head$ or $tail$. Let us assume that the probability of selecting any coin is 0.5, both initially and at every next time step. In order to determine the probability of observing e.g. $head$ at time t we need to take into account that any of the two coins could produce this observation. Therefore, we

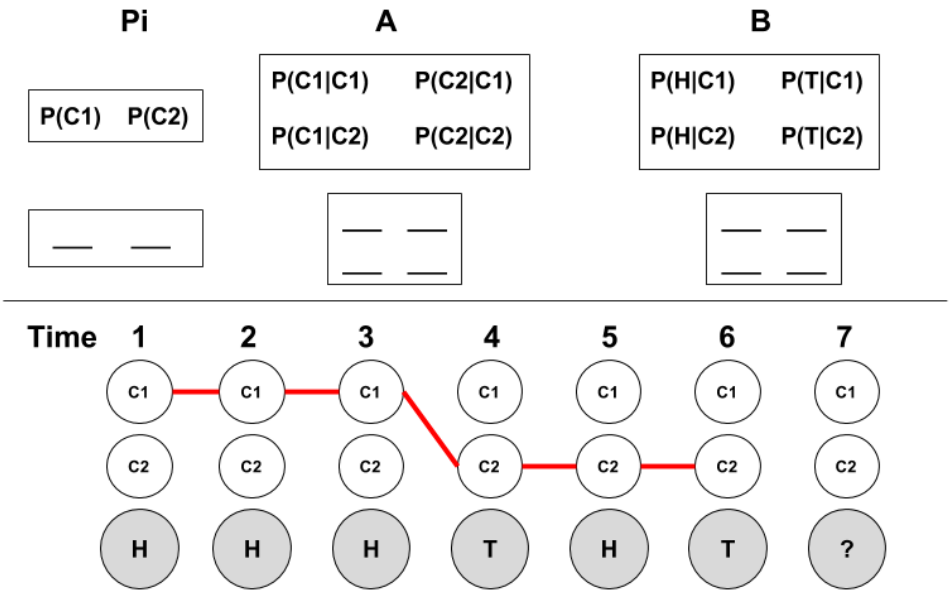

Figure 2.1

need to marginalize over this uncertainty. For the initial time step, $t = 1$, this gives us e.g.

$$P(\mathbf{O}_1 = head) = \sum_{i=1}^{2} P(\mathbf{O}_1 = head, \mathbf{X}_1 = \mathbf{c}_i) \tag{2.2}$$

$$= \sum_{i=1}^{2} P(\mathbf{O}_1 = head | \mathbf{X}_1 = \mathbf{c}_i) P(\mathbf{X}_1 = \mathbf{c}_i) \tag{2.3}$$

$$= P(\mathbf{O}_1 = head | \mathbf{X}_1 = \mathbf{c}_1) P(\mathbf{X}_1 = \mathbf{c}_1) + P(\mathbf{O}_1 = head | \mathbf{X}_1 = \mathbf{c}_2) P(\mathbf{X}_1 = \mathbf{c}_2) \tag{2.4}$$

$$= 0.9 * 0.5 + 0.5 * 0.5 = 0.7 \tag{2.5}$$

Question 1 This problem can be formulated in matrix form. Please specify the initial probability vector π, the transition probability matrix $\mathbf{A}$ and the observation probability matrix $\mathbf{B}$.

2.2 HMM 0 - Next observation Distribution

Imagine that we have an estimate of the current state distribution, i.e. we know which of the coins we have used at the last time step with a certain probability. In Figure 2.1, we are located at time step 7 and know the probability of using coin one or two at the previous time step. This knowledge is represented in a row vector, in which each row represents the probability to be in a certain state, here the coin we use, and the sum of all entries is 1. Given this knowledge, we can compute the probability of observing each observation, here head or tail. First, we need to multiply the transition matrix with our current estimate of states.

Question 2 What is the result of this operation?

In the following, the result must be multiplied with the observation matrix.

Question 3 What is the result of this operation?

Implement these steps in order to solve the HMM 0 problem (found in Kattis: `https://kth.kattis.com/problems/kth.ai.hmm0`). It might be a good idea to implement basic matrix and vector operations such as the multiplication of a matrix and a vector. Test these sufficiently before spamming Kattis. Keep in mind that the problems on Kattis have nothing to do with the coin example but represent arbitrary HMM models. For testing, download the files from the homepage for the problem. You can test your java code outside Kattis by running

java MyHMM < file.in

and then make sure that the output matches that of file.ans in this case. Replace MyHMM with whatever your java program is called. If you are using C++ you would run something like

./myHMM < file.in.

2.3 HMM 1 - Probability of the observation sequence

After predicting the next observation, we want to estimate what the probability of the made observation sequence $\mathbf{O}_{1:T} = [\mathbf{O}_1 = \mathbf{o}_1, \mathbf{O}_2 = \mathbf{o}_2, ..., \mathbf{O}_T = \mathbf{o}_T]$ is. For example, given our model of the coin problem, it will be extremely unlikely to observe only tails all the time.

In order to solve this problem, we can make use of the so called forward algorithm or α-pass algorithm. This procedure iteratively estimates the probability to be in a certain state i at time t **and** having observed the observation sequence up to time t for $t \in [1, ..T]$.

We start off by computing the probability of having observed the first observation $\mathbf{o}_1$ and having been in any of the hidden states. The latter probability is provided by the initial state distribution

vector π. Thus, we initialize $\alpha_1(i)$, where the subscript 1 indicates the time step and i the state, as

$$\alpha_1(i) = P(\mathbf{O}_1 = \mathbf{o}_1, \mathbf{X}_1 = \mathbf{x}_i) \qquad \text{for } i \in [1,..N] \tag{2.6}$$
$$= P(\mathbf{O}_1 = \mathbf{o}_1 | \mathbf{X}_1 = \mathbf{x}_i) P(\mathbf{X}_1 = \mathbf{x}_i) \tag{2.7}$$
$$= b_i(\mathbf{o}_1)\pi_i \tag{2.8}$$

In the following steps, we need to take into account that any of the other hidden states at time $t-1$ could have preceeded the state at the current time t. In the coin example this means that we need to take into account the probability of having used any of the coins at the last time step in order to estimate the probability of the coin that is currently used. If we would not take this probability into account, we would always guess that coin $\mathbf{c}_1$ produced a *head* observation, even if the last 6 observations have been $tail$. **Any information available to us needs to be taken into account!** Thus, at time t we need to marginalize over the probability of having been in any other state at $t-1$ and multiply this estimate with the matching observation probability as follows

$$\alpha_t(i) = P(\mathbf{O}_{1:t} = \mathbf{o}_{1:t}, \mathbf{X}_t = \mathbf{x}_i) \qquad \text{for } i \in [1,..N] \tag{2.9}$$
$$= P(\mathbf{O}_t = \mathbf{o}_t | \mathbf{X}_t = \mathbf{x}_i, \mathbf{O}_{1:t-1}) P(\mathbf{X}_t = \mathbf{x}_i, \mathbf{O}_{1:t-1}) \tag{2.10}$$
$$= P(\mathbf{O}_t = \mathbf{o}_t | \mathbf{X}_t = \mathbf{x}_i) P(\mathbf{X}_t = \mathbf{x}_i, \mathbf{O}_{1:t-1}) \tag{2.11}$$
$$= P(\mathbf{O}_t = \mathbf{o}_t | \mathbf{X}_t = \mathbf{x}_i) \Big[\sum_{j=1}^{N} P(\mathbf{X}_t = \mathbf{x}_i | \mathbf{X}_{t-1} = \mathbf{x}_j) P(\mathbf{X}_{t-1} = \mathbf{x}_j, \mathbf{O}_{1:t-1}) \Big] \tag{2.12}$$
$$= b_i(\mathbf{o}_t) \Big[\sum_{j=1}^{N} a_{j,i} \alpha_{t-1}(j) \Big] \tag{2.13}$$

Question 4 Why is it valid to substitute $\mathbf{O}_{1:t} = \mathbf{o}_{1:t}$ with $\mathbf{O}_t = \mathbf{o}_t$ when we condition on the state $\mathbf{X}_t = \mathbf{x}_i$?

This step has to be computed iteratively for every $t \in [1,..T]$. Finally, we can compute the probability of having observed the given observation sequence $\mathbf{O}_{1:T}$. For this, we again have to marginalize over the hidden state distribution such that

$$P(\mathbf{O}_{1:T} = \mathbf{o}_{1:T}) = \sum_{j=1}^{N} P(\mathbf{O}_{1:T} = \mathbf{o}_{1:T}, \mathbf{X}_T = \mathbf{x}_j) \tag{2.14}$$
$$= \sum_{j=1}^{N} \alpha_T(j) \tag{2.15}$$

Implement these steps in order to solve the HMM 1 problem (found in Kattis: `https://kth.kattis.com/problems/kth.ai.hmm1`). You might want to store the α values in matrix form. Be careful to index all matrices and vectors correctly and to maintain the temporal order of summation and product as in the formulas. As before, HMM 1 is not connected to the coin example but represents arbitrary HMM problems.

2.4 HMM 2 - Estimate Sequence of States

The forward algorithm marginalizes over the hidden state distribution. In contrast, we can also compute the most likely sequence of hidden states given the observations. Let us denote this sequence with $\{\mathbf{X}^*\}_{1:T} = (\mathbf{X}_1^*, \mathbf{X}_2^*, ...\mathbf{X}_T^*)$. Instead of assuming that any state has led to the current estimate, we only take the most likely predecessor into account. The most likely hidden state sequence can be valuable information when estimating the actual dynamics underlying our noisy observations. In the coin example, you might want to know which coin has most likely been used at each time point. In Figure 2.1, such a path is indicated by the red line.

This problem is solved with help of the Viterbi algorithm. The procedure is divided into two steps.

At first, we need to compute the probability of having observed $\mathbf{O}_{1:t} = \mathbf{o}_{1:t}$ and being in a state $\mathbf{X}_t = \mathbf{x}_i$ given the most likely preceding state $\mathbf{X}_{t-1} = \mathbf{x}_j$ for each t. This quantity is denoted by $\delta_t(i)$, defined in equations (2.16 - 2.18). We start with the initial values

$$\delta_1(i) = P(\mathbf{O}_1 = \mathbf{o}_1, \mathbf{X}_1 = \mathbf{x}_i) \qquad \text{for } i \in [1,..N] \tag{2.16}$$

$$= b_i(\mathbf{o}_1)\pi_i. \tag{2.17}$$

The subsequent steps update the δ_t as follows

$$\delta_t(i) = max_{j\in[1,..N]}\ P(\mathbf{X}_t = \mathbf{x}_i|\mathbf{X}_{t-1} = \mathbf{x}_j)P(\mathbf{O}_{1:t-1}, \mathbf{X}^*_{1:t-2}, \mathbf{X}^*_{t-1} = \mathbf{x}_j)P(\mathbf{O}_t = \mathbf{o}_t|\mathbf{X}_t = \mathbf{x}_i) \quad \text{for } i \in [1,..N] \tag{2.18}$$

$$= max_{j\in[1,..N]}\ a_{j,i}\delta_{t-1}(j)b_i(\mathbf{o}_t). \tag{2.19}$$

In order to be able to trace back the most likely sequence later on, it is convenient to store the indices of the most likely states at each step.

$$\delta^{idx}_t(i) = argmax_{j\in[1,..N]}\ a_{j,i}\delta_{t-1}(j)b_i(\mathbf{o}_t) \text{ for } i \in [1,..N] \tag{2.20}$$

So, if the algorithm determined $\delta_t(i)$ for state i at time t to have been preceded by state k, then $\delta^{idx}_t(i) = k$.

When arriving at the end of the observation sequence, T, the probability of the most likely hidden state sequence is given by

$$P(\{\mathbf{X}^*\}_{1:T}, \mathbf{O}_{1:T} = \mathbf{o}_{1:T}) \ = \ max_{j\in[1,..N]}\ \delta_T \tag{2.21}$$

In the second step, we can backtrack the sequence $\{\mathbf{X}^*\}$ by setting

$$\mathbf{X}^*_T \ = \ argmax_{j\in[1,..N]}\ \delta_T(j) \qquad \text{and} \tag{2.22}$$

$$\mathbf{X}^*_t \ = \ \delta^{idx}_t(\mathbf{X}^*_{t+1}) \tag{2.23}$$

Implement these steps in order to solve the HMM 2 problem (found in Kattis: `https://kth.kattis.com/problems/kth.ai.hmm2`). You might want to store the δ and δ^{idx} values in matrix form.

> **Question 5** How many values are stored in the matrices δ and δ^{idx} respectively?

2.5 HMM 3 - Estimate Model Parameters

Up to this point, we assumed that the HMM parameters were given. In a real world scenario, this will rarely be the case. Instead, the initial distribution and transition and observation matrices must be approximated with help of data samples. This can be done efficiently with help of the Baum-Welch algorithm. Luckily, we have already implemented a part of this method - the α-pass algorithm in the HMM 1 problem.

Recall that the α-pass algorithm determined the joint probability of each state at time t and all observations up to t. In order to be able to determine the probability of any state at any time point given **all** observations, we need assess the probability of a state at time t and all future observations $[t+1:T]$. For this, we can make use of the β-pass algorithm that works similar to the α-pass algorithm but traverses the observations in the opposite direction - from the last to the first. The β is defined as

$$\beta_t(i) = P(\mathbf{O}_{t+1:T} = \mathbf{o}_{t+1:T}|\mathbf{X}_t = \mathbf{x}_i) \qquad \text{for } i \in [1,..N], \tag{2.24}$$

$$\tag{2.25}$$

i.e. the probability of observing all future observations $\mathbf{O}_{t+1:T}$ given the current state $\mathbf{X}_t = \mathbf{x}_i$. Again, we need to determine β values for all time steps and all states. For this, we initialize

$$\beta_T(i) = 1 \qquad \text{for } i \in [1,..N], \tag{2.26}$$

$$\tag{2.27}$$

as we do not intend to favour any final state over the other. In the following, we iterate backwards through the observation sequence and compute the β values, defined in the following equation:

$$\beta_t(i) = P(\mathbf{O}_{t+1:T} = \mathbf{o}_{t+1:T} | \mathbf{X}_t = \mathbf{x}_i) \qquad \text{for } i \in [1,..N], \tag{2.28}$$

$$= \sum_{j=1}^{N} P(\mathbf{O}_{t+2:T} = \mathbf{o}_{t+2:T} | \mathbf{X}_{t+1} = \mathbf{x}_j) P(\mathbf{O}_{t+1} = \mathbf{o}_{t+1} | \mathbf{X}_{t+1} = \mathbf{x}_j) P(\mathbf{X}_{t+1} = \mathbf{x}_j | \mathbf{X}_t = \mathbf{x}_i) \tag{2.29}$$

$$= \sum_{j=1}^{N} \beta_{t+1}(j) b_j(\mathbf{o}_{t+1}) a_{i,j}. \tag{2.30}$$

Now that we have computed both α and β, we can estimate the probability of being in state i at time t and being in state j at time $t+1$ given all made observations and the probability of being in state i at time t given all made observations. These probabilities are given by the di-gamma function

$$\gamma_t(i,j) = P(\mathbf{X}_t = \mathbf{x}_i, \mathbf{X}_{t+1} = \mathbf{x}_j | \mathbf{O}_{1:T} = \mathbf{o}_{1:T}) \tag{2.31}$$

$$= \frac{\alpha_t(i) a_{i,j} b_j(\mathbf{O_{t+1}}) \beta_{t+1}(j)}{\sum_{k=1}^{N} \alpha_T(k)} \tag{2.32}$$

and the gamma function

$$\gamma_t(i) = P(\mathbf{X}_t = \mathbf{x}_i, | \mathbf{O}_{1:T} = \mathbf{o}_{1:T}) \tag{2.33}$$

$$= \sum_{j=1}^{N} \gamma_t(i,j). \tag{2.34}$$

Question 6 Why we do we need to divide by the sum over the final α values for the di-gamma function?

Finally, we can estimate $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ by determining the expected value of the probabilities. We have the transition estimates given by

$$a_{i,j} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad \text{for } i, j \in [1,..N], \tag{2.35}$$

the observation estimates given by

$$b_j(k) = \frac{\sum_{t=1}^{T-1} \mathbf{1}(\mathbf{O}_t = k) \gamma_t(j)}{\sum_{t=1}^{T-1} \gamma_t(j)} \qquad \text{for } j \in [1,..N], k \in [1,..K], \tag{2.36}$$

where $\mathbf{1}(\mathbf{O}_t = k)$ is the indicator function that is 1, when the argument is true and 0 otherwise.

At last, the initial probabilities are given by

$$\pi_i = \gamma_1(i) \qquad \text{for } i \in [1,..N]. \tag{2.37}$$

Now, we have everything in place in order to learn the parameters of our HMM with the Baum-Welch Algorithm:

1. Initialize $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ (2.38)
2. Compute all $\alpha_t(i), \beta_t(i), \gamma_t(i,j), \gamma_t(i)$ values (2.39)
3. Re-estimate $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ (2.40)
4. Repeat from 2. until convergence (2.41)

Implement these steps in order to solve the HMM 3 problem (found in Kattis: `https://kth.kattis.com/problems/kth.ai.hmm3`). For more details, especially regarding the initialization of the parameters (!), please refer to the Stamp tutorial (A Revealing Introduction to Hidden Markov Models by Mark Stamp).

3 GRADE LEVEL C

3.1 OVERVIEW

This part extends the HMM implementation by more empirical investigations. Given your complete Baum-Welch algorithm, you are supposed to both investigate different properties of its performance and to test different parameter settings. To this end, we provide you with an HMM model which can be used for data generation. Thus, this assignment focuses on theoretical and empirical understanding of HMMs.

3.2 EMPERICAL INVESTIGATIONS

We provide you with a file that contains a data sequence (T = 1000, 10000) generated from the following HMM dynamics:

$$\mathbf{A} = \begin{pmatrix} 0.7 & 0.05 & 0.25 \\ 0.1 & 0.8 & 0.1 \\ 0.2 & 0.3 & 0.5 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 0.7 & 0.2 & 0.1 & 0 \\ 0.1 & 0.4 & 0.3 & 0.2 \\ 0 & 0.1 & 0.2 & 0.7 \end{pmatrix} \qquad \pi = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \tag{3.1}$$

Your task is to train an HMM on a varying number of these observations with help of your Baum-Welch implementation.

Question 7 Train an HMM with the same parameter dimensions as above, i.e. **A** should be a 3 times 3 matrix, etc. Initialize your algorithm with the following matrices:

$$\mathbf{A} = \begin{pmatrix} 0.54 & 0.26 & 0.20 \\ 0.19 & 0.53 & 0.28 \\ 0.22 & 0.18 & 0.6 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 0.5 & 0.2 & 0.11 & 0.19 \\ 0.22 & 0.28 & 0.23 & 0.27 \\ 0.19 & 0.21 & 0.15 & 0.45 \end{pmatrix} \quad \pi = \begin{pmatrix} 0.3 & 0.2 & 0.5 \end{pmatrix}$$

Does the algorithm converge? How many observations do you need for the algorithm to converge? How can you define convergence?

Question 8 Train an HMM with the same parameter dimensions as above, i.e. A is a 3x3 matrix etc. The initialization is left up to you.

How close do you get to the parameters above, i.e. how close do you get to the generating parameters in Eq. 3.1? What is the problem when it comes to estimating the distance between these matrices? How can you solve these issues?

Question 9 Train an HMM with different numbers of hidden states.

What happens if you use more or less than 3 hidden states? Why?

Are three hidden states and four observations the best choice? If not, why? How can you determine the optimal setting? How does this depend on the amount of data you have?

Question 10 Initialize your Baum-Welch algorithm with a uniform distribution. How does this effect the learning?

Initialize your Baum-Welch algorithm with a diagonal **A** matrix and $\pi = [0,0,1]$. How does this effect the learning?

Initialize your Baum-Welch algorithm with a matrices that are close to the solution. How does this effect the learning?

4 Grade level B and A

4.1 Overview

In this part of the assignment you will implement an agent that can classify and hunt ducks. The aim is to get a deep understanding of Hidden Markov Models and for you to show that you can make use of them to solve a less structured problem. You will practice identifying the number of states, observations, etc. You will have to learn the HMM model and use it for classification.

If you have not played the game until now, go here and try your hand at it: `http://www.silvergames.com/duck-hunt`.

Though this is a one person version of the game, the game was originally meant for 2-6 players and was designed as a Nintendo TV-video game. The controllers were plastic guns with light sensors that could be aimed at the television to "shoot" the ducks on the TV screen.

Our version of ducks hunt is a generalized version of the original game: There is a sky with birds flying. The birds fly in the sky until they are shot down. The player has to observe the flight pattern of the birds and predict the next move in order to shoot them down. If the prediction is correct, the player will hit the bird and gain points. If the prediction is wrong the player will miss the bird and lose points. The game is over when all birds are shot or when the time runs out.

We have also added different species of birds which behave differently in the air. Most birds give one point when shot, but one of the species is very rare and will hurt your score seriously if you shoot it. The species are identified after each round, in order to prioritize your targets you will need to identify them during flight. You may also get extra points for guessing the correct species before the truth is revealed.

AI Perspective

The game can be broken down into three main topics, which will all be addressed in your code:

1. Predicting the flight trajectory of the birds so that they can be shot down.
2. Identifying the bird species to avoid forbidden targets and maximizing the score.
3. Making decisions to shoot or not based on the confidence of prediction of bird species and flight trajectory.

Tip: First focus on predicting the movement of a single bird using a HMM. This is the building block to solve the entire homework.

4.2 Duck hunt - General information

This assignment requires thorough understanding of HMMs. Therefore it is **highly recommended** to have a look at the Stamp tutorial (A Revealing Introduction to Hidden Markov Models by Mark Stamp)! It will give you more details about numerically stable computations and other important factors for your implementation.

You find valuable information regarding the dynamics of the game on Kattis `https://kth.kattis.com/problems/kth.ai.duckhunt`.

Provided code

A basic code skeleton is provided for you in each of the supported programming languages. The program is fully functional as it is provided, but the program never shoots nor guesses. The code can be found on Kattis: `https://kth.kattis.com/problems/kth.ai.duckhunt`. **Please remember that you are not allowed to use the implementation we provided for the 2015 edition of this course!**

The birds and the environments provided for testing are different from the ones that are used in Kattis.

You should modify the player class and you may also create any number of new classes and files. The files included in the skeleton may be modified locally but keep in mind that they will be overwritten on kattis (except for Player.cpp/hpp/java).

Your interface with the judge is the player class. Avoid using stdin and stdout. (Use stderr for debugging.)

TEST ENVIRONMENT

There are five test environments available to you. Each environment comes in two versions, with and without opponents. The default environment is SouthEmissions.in. You can play other environments by using the "load" option to the server, for example

./duckhunt server load ParadiseEmissions.in < player2server | ./duckhunt verbose > player2server

GRADING DETAILS AND TASKS

You are required to have a **Kattis score of at least 370**. This homework has not as an aim that you optimize your code toward Kattis but that you get an **understanding of the theory behind HMMs**. Therefore, you have to be able to reason about your solution and to answer theoretical questions regarding HMMs. **You need to solve both the shooting and guessing problem. Having solved them means being able to both shoot and guess reasonably, all using an HMM framework.**