Algorithms in Probabilistic Modelling
Hidden Markov Models

This honors project introduces a fundamental probabilistic model called the hidden Markov model. Hidden Markov models have a particular structure that makes the construction of polynomial inference algorithms feasible using dynamic programming. There are many expositions of hidden Markov model and I have included two such texts on HuskyCT. This assignment is meant to introduce the concept but the reading should be referred to for specific details.

Introduction to probabilistic modelling

In probabilistic modelling, we are often presented with many correlated random variables which represent the data themselves or the unknown (hidden) generating process for the data. Examples of correlated random variables are words in an email, whether or not a section of the genome represents a gene, or user movie ratings. We capture the relationship among the random variables within a directed graphical model, also known as a Bayesian network (or Bayes net for short). Bayes nets are graphs where random variables are represented by vertices and an edge from $u \to v$ signifies v is conditionally dependent on u. Shaded nodes denote *observed* random variables while unshaded nodes are unobserved (hidden).

Conditional independence and model representation

In any probabilistic model, we must reason about the conditional independence among the random variables, that is, which variables are independent of each other when conditioning on another set of random variables. Formally, we say X is conditionally independent of Z given Y if $P(X, Z|Y) = P(X|Y)P(Z|Y)$. We can represent the random variables and their conditional dependencies in a graph representation where random variables are nodes and edges indicate conditional dependence (Figure 1).

Hidden Markov models

A Markov model assumes that the immediate past z_{t-1} captures everything we need to know about $z_{1 \ldots t-1}$ for the computation of z_t. Markov models are useful for modelling sequential data, which can include time-based data like weather forecasts or spacial data like genome sequences. A *hidden Markov model* (HMM) is a Markov model where we assume a number of hidden states that underlie the generation of the observed data (Figure 1). For example, if the observed data x were features from an audio signal extracted from a speech, then z may be the word that was spoken.

To specify an HMM we require a specification of the hidden states, an observation model, and a transition model. In Figure 1, this amount to specifying the probability distribution on z, $x|z$, and $z_t|z_{t-1}$ respectively. Let us expand the joint distribution of the hidden and observed random variables using the chain rule and the conditional independencies of a hidden Markov model.

$$P(z_{1:T}, x_{1:T}) = p(z_1) \prod_{t=2}^{T} p(z_t|z_{t-1}) \prod_{t=1}^{T} p(x_t|z_t) \tag{1}$$

Notice how this equation contains the three elements we require to specify an HMM: hidden state specification, observation model, transition model.

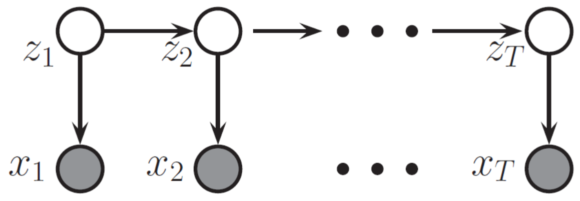

Figure 1: A simple Hidden Markov model with observations x and hidden random variables z.

Inference in HMMs: An occasionally dishonest Casino

A dishonest casino uses either a fair or a biased die (Figure 2). The fair die has an equal probability of presenting numbers 1 through 6. The biased die has an equal probability of presenting numbers 1 through 5 but a probability of 1/2 for presenting a 6. The casino dealer transitions from bias-bias with probability 0.9, bias-fair with probability 0.1, fair-bias with probability 0.05, fair-fair with probability 0.95 The biased coin comes out heads with probability $\frac{4}{5}$ and tails with probability $\frac{1}{5}$. The probability of transitioning from a biased to a fair coin and from a fair to a biased coin is $\frac{1}{5}$. The casino is equally likely to start in the fair or biased state.

An example of observed rolls and hidden die state (L for loaded, F for fair) is

```
Rolls: 664153216162115234653214356634261655234232315142464156663246
Die:   LLLLLLLLLLLLLFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFLLLLLLLL
```

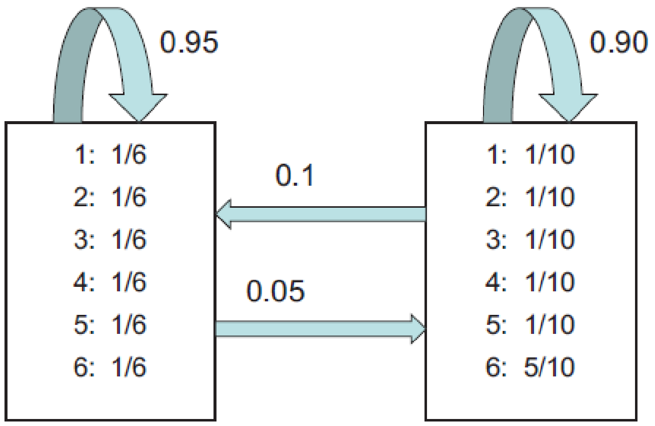

Figure 2: The dishonest casino has two dice for which they switch given the probability distribution in this figure.

The die state is hidden but we do observe the die roll. We observe that when the die state changes, the distribution of the numbers change as well. We will be concerned with two types of inference:

1. *Maximum a posteriori* (MAP) inference, or, the sequence of hidden states that gives the maximum probability conditional on the data: $\arg\max_{z_{1:T}} p(z_{1:T}|x_{1:T})$

2. *Marginal state probability*, or, the probability of a hidden state at a single location conditional on all of the data: $p(z_i|x_{1:T})$. Note that we can combine these marginals to compute the maximizer of the posterior marginals: $\hat{z} = (\arg\max_{z_1} p(z_1)|x_{1:T}) \dots \arg\max_{z_t} p(z_t)|x_{1:T}))$. Note that *this is not necessarily the same as the most probable sequence of states.*

In both cases, we use dynamic programming. We first state the algorithm to compute the MAP estimate $\arg\max_{z_{1:T}} p(z_{1:T}|x_{1:T})$. A useful construct for explaining both algorithms is the trellis graph (Figure 3). In the trellis graph, we expanded the set of possible hidden states for each observation (in the case of the dishonest casino, the only possible hidden states are *fair* or *loaded*). Combinatorially, we see that every possible path through the trellis graph indicates a valid sequence of hidden states, each of these with a particular probability. How do we compute the maximum via dynamic programming?

A key to this construction is the observation that the probability of a path can be factored as $p(z_1|x_1)p(z_2|x_2, z_1)p(z_3|x_3, z_2)\dots$ due to the Markov property. Let (j, t) indicate some state j (for which there are N possible states) at time t. Let $DP(j, t)$ be the dynamic programming table entry for (j, t) representing the maximum probability of any path ending at state (j, t). The only way to visit (j, t) is through one of the nodes $(i, t-1)$ for $i = 1 \dots N$. If we have already computed the maximum probability of reaching each of the nodes in $(i, t-1)$ for

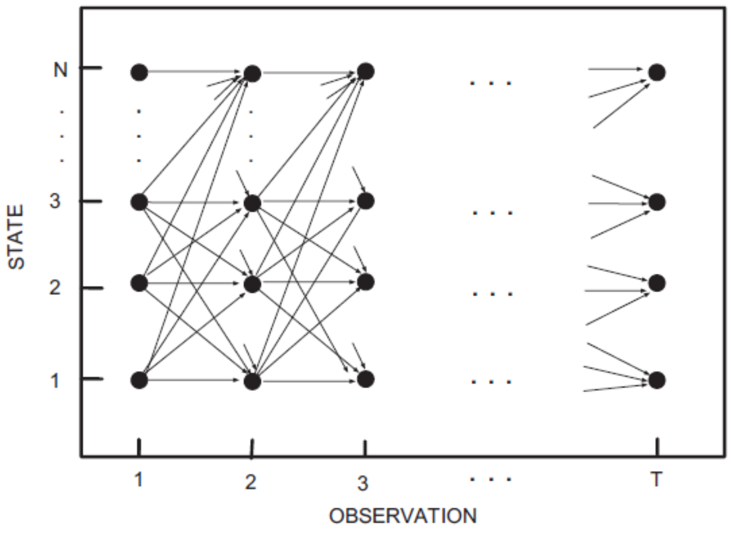

Figure 3: The trellis graph of states vs time in a Markov chain.

$i = 1 \dots N$ (which we assume is in the DP table), and we know the transition probabilities, then we can compute $DP(j,t) = \max_{i=1\dots N} DP(i,t-1) \cdot \psi(i,j)\phi(j)$ where $\psi(i,j)$ is the transition probability from state i at $t-1$ to j at t and $\phi(j)$ is the emission probability of the observation at t for hidden state j. Continuing this process for $t = 1, \dots T$, the maximum value in the final column of the DP table T will give the probability of the MAP. So, the path with the highest probability ends in column T at the state with the largest probability.

A couple of implementation notes. We keep track of the back-pointers so we can traceback the final sequence through the hidden states. We should also work with log-probabilities to avoid underflow, e.g., $log(P(X) \cdot P(Y)) = log(P(X)) + log(P(Y))$. The algorithm to compute the MAP estimate for a Markov chain is referred to as the Viterbi algorithm; more generally it is referred to as the max-product algorithm.

Next, we state the algorithm to compute the marginal state probability $p(z_i|x_{1:t}, z_{-t})$ for all $z_i \in \{z_1, \dots, z_t\}$ and where z_{-t} is the set of all hidden variables that are not z_t. Similar to the Viterbi algorithm, we store probabilities in a DP table DP. Unlike the Viterbi algorithm, the interpretation of DP is the marginal probability, so the sum of the probabilities for all paths to reach that state. To compute the marginal probability at a single time t, we need to compute the probability of paths from $1, \dots, t-1$ and $t+1, \dots, T$ that read a state z_i.

Let's focus on a single time point.

$$p(z_t|x_t, z_{-t}) = p(z_1|x_1)p(z_2|x_2, z_1) \dots p(z_t|x_t, z_{t-1})p(z_t|x_t, z_{t+1})p(z_{t+1}|x_{t+1}, z_{t+2}) \dots p(z_T|x_T) \quad (2)$$

We provide interpretation for these probabilities. $p(z_1|x_1)$ and $p(z_T|x_T)$ start the chain off from the first and last observation respectively. Then we have transitions from previous states to the next state from both directions. Functionally, we change the max in the Viterbi recursion with a summation: $DP_F(j,t) = \sum_{i=1\ldots N} DP(i,t-1) \cdot \psi(i,j)\phi(j)$ where $\psi(i,j)$ is the transition probability from state i at $t-1$ to j at t and $\phi(j)$ is the emission probability of the observation at t for hidden state j. Backward probabilities are calculated similarly for transitions from states at $t+1$ to t.

We observe that we do not have to compute this for each state separately because we would be duplicating work. Instead we compute all of the forward probabilities in a table, say, DP_F, and all backward probabilities in another table DP_B. To get the marginal probability of (j,t) we need only multiply $DP_F(j,t)$ by $DP_B(j,t)$ and then normalize by $\sum_{i=1}^{N} DP_F(i,t)DP_B(i,t)$.

Your task

Implement the Viterbi and forward-backward algorithms using the previously defined HMM to find the most probable explanation of hidden states that generate the series of dice rolls on HuskyCT. Compare your solutions with the ground truth. In general, discuss when you would prefer Viterbi over forward-backward and vice-versa. Is it always the best idea to return the most likely explanation? Why or why not? We included two chapters on Hidden Markov models on HuskyCT for additional perspectives on HMMs.