CS 188 Summer 2020 Introduction to Artificial Intelligence

Final Review

HMM, PF and Viterbi Solutions

Q1. Partying Particle #No Filter(ing)

Algorithm 1.1 Particle Filtering

```
1: procedure PARTICLE FILTERING(T, N)        ▷ T: number of time steps, N: number of sampled particles
2:     x ← sample N particles from initial state distribution P(X_0)        ▷ Initialize
3:     for t ← 0 to T − 1 do        ▷ X_t: hidden state, E_t: observed evidence
4:         x_i ← sample particle from P(X_{t+1}|X_t = x_i) for i = 1, ..., N        ▷ Time Elapse Update
5:         w_i ← P(E_{t+1}|X_{t+1} = x_i) for i = 1, ..., N        ▷ Evidence Update
6:         x ← resample N particles according to weights w        ▷ Particle Resampling
7:     end for
8:     return x
9: end procedure
```

Algorithm 1.1 outlines the particle filtering algorithm discussed in lecture. The variable x represents a list of N particles, while w is a list of N weights for those particles.

(a) Here, we consider the unweighted particles in x as approximating a distribution.

(i) After executing line 4, which distribution do the particles x represent?

● $P(X_{t+1}|E_{1:t})$ ○ $P(X_{1:t+1}|E_{1:t})$ ○ $P(X_{t+1}|E_{1:t+1})$ ○ None

(ii) After executing line 6, which distribution do the particles x represent?

○ $P(X_{t+1}|X_t, E_{1:t+1})$ ○ $P(X_{1:t+1}|E_{1:t+1})$ ● $P(X_{t+1}|E_{1:t+1})$ ○ None

1. Before line 4, x estimated $P(X_t \mid E_{1:t})$. Line 4 is the time elapse update, sampling from the distribution $P(X_{t+1}|X_t = x)$. After the update, x estimates $P(X_{t+1} \mid E_{1:t})$. Note the second choice is estimating too many states, and the third choice depends on E_{t+1} that has yet to be incorporated.

2. (ii) After we sum by state, normalize, and then re-sample, our new states x approximate $P(X_{t+1}|E_{1:t+1})$ which is the distribution we want for HMM's.

(b) The particle filtering algorithm should return a sample-based approximation to the true posterior distribution $P(X_T \mid E_{1:T})$. The algorithm is **consistent** if and only if the approximation converges to the true distribution as $N \to \infty$. In this question, we present several modifications to Algorithm 1.1. For each modification, indicate if the algorithm is still **consistent** or **not consistent**, and if it is **consistent**, indicate whether you expect it to be **more accurate** in general in terms of its estimate of $P(X_T \mid E_{1:T})$ (i.e., you would expect the estimated distribution to be closer to the true one) or **less accurate**. Assume unlimited computational resources and arbitrary precision arithmetic.

(i) We modify line 6 to sample 1 or $2N - 1$ particles with equal probability $p = 0.5$ for each time step (as opposed to a fixed number of particles N). You can assume that $P(E_{t+1}|X_{t+1}) > 0$ for all observations and states. This algorithm is:

○ Consistent and More Accurate
○ Consistent and Less Accurate
● Not Consistent

This algorithm is not consistent because we will sample only one particle, with probability tending to one. Every time we end up sampling 1 particle, our algorithm can no longer represent the posterior distribution with arbitrary precision even as $N \to \infty$.

(ii) Replace lines 4–6 as follows:
$4'$: Compute a tabular representation of $P(X_t = s|E_{1:t})$ based on the proportion of particles in state s.
$5'$: Use the forward algorithm to calculate $P(X_{t+1}|E_{1:t+1})$ exactly from the tabular representation.
$6'$: Set x to be a sample of N particles from $P(X_{t+1}|E_{1:t+1})$.
This algorithm is:

● Consistent and More Accurate ○ Not consistent
○ Consistent and Less Accurate

This algorithm is consistent. It is more accurate since the particle filtering algorithm is a consistent approximator while the forward algorithm computes the exact distribution. The forward algorithm might be computationally intractable for large HMMs (a key reason to use particle filtering), but we are instructed to ignore resource limitations.

(iii) At the start of the algorithm, we initialize each entry in w to 1s. Keep line 4, but replace lines 5 and 6 with the following multiplicative update:
$5'$: For $i = 1, \ldots, N$ do
$6'$ $\quad w_i \leftarrow w_i * P(E_{t+1}|X_{t+1} = x_i)$.
Finally, **only** at the end of the T iterations, we resample x according to the cumulative weights w just like in line 6, producing a list of particle positions. This algorithm is:

○ Consistent and More Accurate ○ Not Consistent
● Consistent and Less Accurate

This algorithm is consistent. Indeed, it is equivalent to likelihood weighting, which is consistent. One can also see this result by comparison to the particle filtering algorithm. The time elapse updates are unmodified. The observation updates are weighted by $P(E_t|X_t)$ as in particle filtering. The only difference is that normalization and resampling happens at the end of the loop rather than as part of each observation update. It is less accurate since in practice, the error for fixed N grows exponentially with T, but for fixed T it still converges as N goes to ∞.

(c) Suppose that instead of particle filtering we run the following algorithm on the Bayes net with T time steps corresponding to the HMM:

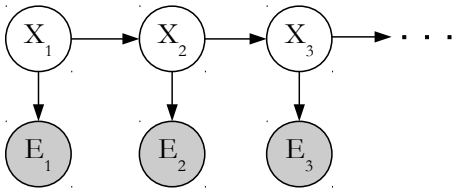
1. Fix all the evidence variables $E_{1:T}$ and initialize each X_t to a random value x_t.
2. For $i = 1, \ldots, N$ do
   - Choose a variable X_t uniformly at random from $X_1, \ldots, X_T$.
   - Resample X_t according to the distribution $P(X_t|X_{t-1} = x_{t-1}, X_{t+1} = x_{t+1}, E_t = e_t)$.
   - Record the value of X_T as a sample.

Finally, estimate $P(X_T = s|E_{1:T} = e_{1:T})$ by the proportion of samples with $X_T = s$. This algorithm is:

● Consistent ○ Not Consistent

This is just Gibbs sampling. It will converge to the true posterior distribution $P(X_{1:T} \mid E_{1:T})$, even though the sample only changes when X_T is sampled.

Q2. Most Likely Estimates in HMMs



The Viterbi algorithm finds the most probable sequence of hidden states $X_{1:T}$, given a sequence of observations $e_{1:T}$. Throughout this question you may assume there are no ties. Recall that for the canonical HMM structure, the Viterbi algorithm performs the following **dynamic programming**[1] computations:

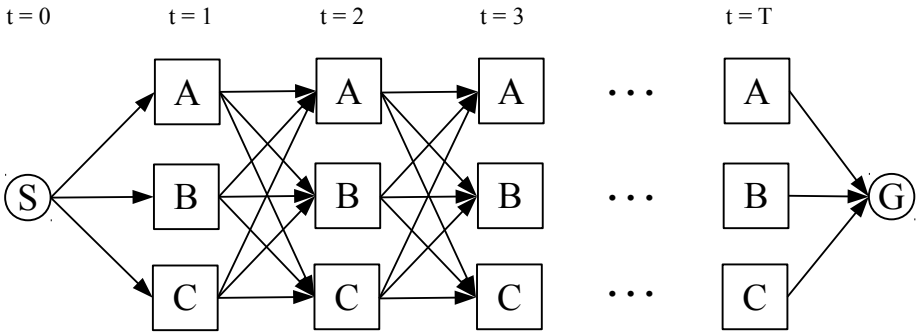
$$m_t[x_t] = P(e_t|x_t) \max_{x_{t-1}} P(x_t|x_{t-1}) m_{t-1}[x_{t-1}]$$

(a) For the HMM structure above, which of the following probabilities are maximized by the sequence of states returned by the Viterbi algorithm? Mark **all** the correct option(s).

- ○ $P(X_{1:T})$
- ○ $P(X_T|e_T)$
- ● $P(X_{1:T}|e_{1:T})$
- ● $P(X_{1:T}, e_{1:T})$
- ● $P(X_1)P(e_1|X_1)\prod_{t=2}^{T} P(e_t|X_t)P(X_t|X_{t-1})$
- ○ $P(X_1)\prod_{t=2}^{T} P(X_t|X_{t-1})$
- ○ None of the above

The sequence of states returned by the Viterbi Algorithm maximizes the conditional $= P(X_{1:T}|e_{1:T})$. Since, $P(e_{1:T})$ is just a constant, $P(X_{1:T}, e_{1:T}) \propto P(X_{1:T}|e_{1:T})$. Hence, the full joint $P(X_{1:T}, e_{1:T})$ is also maximized. The third option $P(X_1)P(e_1|X_1)\prod_{t=2}^{T} P(e_t|X_t)P(X_t|X_{t-1}) == P(X_{1:T}, e_{1:t})$ due to the conditional independences implied by the HMM structure; therefore, this is also maximized.

(b) Consider an HMM structure like the one in part (a) above. Say for all time steps t, the state X_t can take on one of the three values $\{A, B, C\}$. Then, we can represent the state transitions through the following directed graph, also called a *Trellis Diagram*.



We wish to formulate the most probable sequence of hidden state query as a **graph search problem**.

Note in the diagram above, dummy nodes S and G have been added to represent the *start* state and the *goal* state respectively. Further, the transition from the starting node S to the first state X_1 occurs at time step $t = 0$; transition from X_T (the last HMM state) to the goal state G occurs at time step $t = T$.

(The questions for this section are on the following page)

[1] If you're not familiar with dynamic programming, it is essentially a recursive relation in which the current value is defined as a function of previously computed values. In this case, the value at time t is defined as a function of the values at time $t - 1$

Definition : Let $w^t_{Y\to Z}$, be the cost of the edge for the transition from state Y at time t to state Z at time $t+1$. For example, $w^1_{A\to B}$ is the cost of the edge for transition from state A at time 1 to state B at time 2.

(i) For which **one** of the following values for the weights $w^t_{Y\to Z}$, $1 \le t < T$, would the minimum cost path be exactly the same as most likely sequence of states computed by the Viterbi algorithm.

- ○ $w^t_{Y\to Z} = -P(X_{t+1} = Z|X_t = Y)$
- ○ $w^t_{Y\to Z} = -P(e_{t+1}|X_{t+1} = Z)P(X_{t+1} = Z|X_t = Y)$
- ○ $w^t_{Y\to Z} = -\log(P(X_{t+1} = Z|X_t = Y))$
- ● $w^t_{Y\to Z} = -\log(P(e_{t+1}|X_{t+1} = Z)P(X_{t+1} = Z|X_t = Y))$
- ○ $w^t_{Y\to Z} = \dfrac{1}{P(X_{t+1} = Z|X_t = Y)}$
- ○ $w^t_{Y\to Z} = \dfrac{1}{P(e_{t+1}|X_{t+1} = Z)P(X_{t+1} = Z|X_t = Y)}$

We want the solution to maximize the joint $P(X_{1:T}, e_{1:T}) = P(X_1)P(e_1|X_1)\prod_{t=2}^{T} P(e_t|X_t)P(X_t|X_{t-1})$. Since, a search algorithm **minimizes** the cost, we want to pose this problem as a minimization problem. Hence, we can equivalently say that we want to minimize $\dfrac{1}{P(X_1)P(e_1|X_1)\prod_{t=2}^{T} P(e_t|X_t)P(X_t|X_{t-1})}$.
A search algorithm in its native form can only work with **additive** costs. Therefore, to turn the above products of probabilities into sums, we take the log.

Hence, we wish to minimize :

$$\log\left(\frac{1}{P(X_1)P(e_1|X_1)\prod_{t=2}^{T} P(e_t|X_t)P(X_t|X_{t-1})}\right) = -\log\left(P(X_1)P(e_1|X_1)\prod_{t=2}^{T} P(e_t|X_t)P(X_t|X_{t-1})\right)$$

$$= -\log\left(P(X_1)P(e_1|X_1)\right) - \sum_{t=2}^{T}\log\left(P(e_t|X_t)P(X_t|X_{t-1})\right).$$

If for $t > 1$, the edge cost is set to $-\log P(e_{t+1}|X_{t+1} = Z)P(X_{t+1} = Z|X_t = Y)$, we get the above as the total cost of the path.

(ii) The initial probability distribution of the state at time $t = 1$ is given $P(X_1 = Y), Y \in \{A, B, C\}$.

Which **one** of the following should be the value of $w^0_{S\to Y}, Y \in \{A, B, C\}$ — these are the cost on the edges connecting S to the states at time $t = 1$?

- ○ $w^0_{S\to Y} = -P(X_1 = Y)$
- ○ $w^0_{S\to Y} = -P(e_1|X_1 = Y)P(X_1 = Y)$
- ○ $w^0_{S\to Y} = -\log(P(X_1 = Y))$
- ● $w^0_{S\to Y} = -\log(P(e_1|X_1 = Y)P(X_1 = Y))$
- ○ $w^0_{S\to Y} = \dfrac{1}{P(X_1 = Y)}$
- ○ $w^0_{S\to Y} = \dfrac{1}{P(e_1|X_1 = Y)P(X_1 = Y)}$
- ○ $w^0_{S\to Y} = \alpha, \alpha \in \mathbb{R}$: (some constant)

The reasoning is essentially the same as the previous question. One exception is that for the first node X_1, there is no state transition probability. Instead, we use the prior $P(X_1)$. Therefore, we modify the answer to the previous question to use the prior probability instead.

(iii) Which **one** of the following should be the value of $w^T_{Y\to G}, Y \in \{A, B, C\}$ — these are the cost on the edges connecting the states at last time step $t = T$ to the goal state G?

- ○ $w^T_{Y\to G} = -P(X_T = Y)$
- ○ $w^T_{Y\to G} = -P(e_T|X_T = Y)P(X_T = Y)$
- ○ $w^T_{Y\to G} = -\log(P(X_T = Y))$
- ○ $w^T_{Y\to G} = -\log(P(e_T|X_T = Y)(P(X_T = Y))$
- ○ $w^T_{Y\to G} = \dfrac{1}{P(X_T = Y)}$
- ○ $w^T_{Y\to G} = \dfrac{1}{P(e_T|X_T = Y)P(X_T = Y)}$
- ● $w^T_{Y\to G} = \alpha, \alpha \in \mathbb{R}$: (some constant)

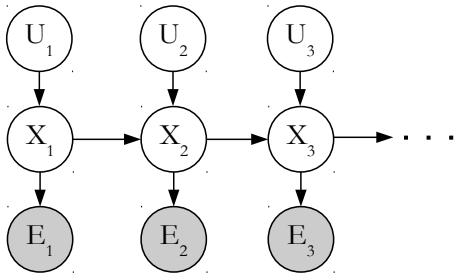
As long as the costs on all the three edges : $w^T_{A\to G}, w^T_{B\to G}, w^T_{C\to G}$ is the same, we will get the optimal answer. Note, it does not matter if the constant α is postive or negative because the total probability is only scaled by a positive constant $= e^{\alpha}$ (that is, the total cost of the path is $(\texttt{sum of all log probabilities}) + w^T_{Y\to G}(== \alpha)$. When you exponentiate this to get the probabilities, you get $e^{\text{sum of log probabilities}} * e^{\alpha}$ — which is just a scaling by some positive number).

(c) We consider extending the Viterbi algorithm for finding the most likely sequence of states in modified HMMs.

For your convenience, the computations performed by the Viterbi algorithm for the canonical HMM structure, like in part (a), are repeated below:

$$m_t[x_t] = P(e_t|x_t) \max_{x_{t-1}} P(x_t|x_{t-1}) m_{t-1}[x_{t-1}]$$

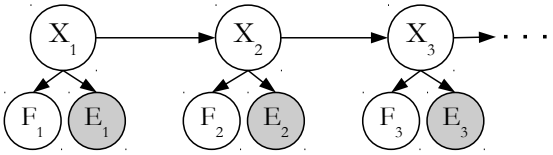
(i) Consider the HMM below with additional variables U_t. The HMM can be interpreted as : The state X_t at time t is caused due to some action U_t and previous state X_{t-1}. The state X_t emits an observation E_t. Both U_t and X_t are unobserved.



We want to find the most likely sequence of states $X_{1:T}$ and actions $U_{1:T}$, given the sequence of observations $e_{1:T}$. Write a dynamic programming update for $t > 1$ **analogous** to the one for the canonical HMM structure.

$m_t[x_t, u_t] =$ $P(e_t|x_t)P(u_t) \max_{x_{t-1}, u_{t-1}} P(x_t|u_t, x_{t-1}) m[x_{t-1}, u_{t-1}]$

(ii) Consider the HMM below with two emission variables at each time step F_t and E_t. E_t is observed while X_t and F_t are unobserved.

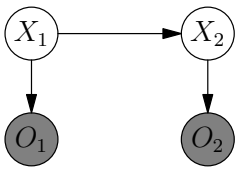


We want to find the most likely sequence of states $X_{1:T}$ and the unobserved emissions $F_{1:T}$, given the sequence of observations $e_{1:T}$. Write a dynamic programming update for $t > 1$ **analogous** to the one for the canonical HMM structure.

$m_t[x_t, f_t] =$ $P(e_t|x_t)P(f_t|x_t) \max_{x_{t-1}, f_{t-1}} P(x_t|x_{t-1}) m[x_{t-1}, f_{t-1}]$

Q3. Hidden Markov Models

Consider the following Hidden Markov Model.



X_1	$\Pr(X_1)$
0	0.3
1	0.7

X_t	X_{t+1}	$\Pr(X_{t+1}\|X_t)$
0	0	0.4
0	1	0.6
1	0	0.8
1	1	0.2

X_t	O_t	$\Pr(O_t\|X_t)$
0	A	0.9
0	B	0.1
1	A	0.5
1	B	0.5

Suppose that $O_1 = A$ and $O_2 = B$ is observed.

(a) Use the Forward algorithm to compute the probability distribution $\Pr(X_2, O_1 = A, O_2 = B)$. Show your work. You do not need to evaluate arithmetic expressions involving only numbers.

X_1	$\Pr(X_1, O_1 = A)$
0	$0.3 \cdot 0.9$
1	$0.7 \cdot 0.5$

X_2	$\Pr(X_2, O_1 = A, O_2 = B)$
0	$0.1 \cdot [0.4 \cdot (0.3 \cdot 0.9) + 0.8 \cdot (0.7 \cdot 0.5)] = 0.0388$
1	$0.5 \cdot [0.6 \cdot (0.3 \cdot 0.9) + 0.2 \cdot (0.7 \cdot 0.5)] = 0.1160$

(b) Use the Viterbi algorithm to compute the maximum probability sequence X_1, X_2. Show your work.

X_1	$\Pr(X_1, O_1 = A)$
0	$0.3 \cdot 0.9$
1	$0.7 \cdot 0.5$

X_2	$\max_{x_1} \Pr(X_1 = x_1, X_2, O_1 = A, O_2 = B)$	$\arg\max$
0	$0.1 \cdot \max(0.4 \cdot (0.3 \cdot 0.9), 0.8 \cdot (0.7 \cdot 0.5)) = 0.1 \cdot \max(0.108, 0.28) = 0.028$	$X_1 = 1$
1	$0.5 \cdot \max(0.6 \cdot (0.3 \cdot 0.9), 0.2 \cdot (0.7 \cdot 0.5)) = 0.5 \cdot \max(0.162, 0.07) = 0.081$	$X_1 = 0$

Thus, in the maximum probability sequence, $X_2 = 1$ and $X_1 = 0$.

For the next two questions, use the specified sequence of random numbers $\{a_i\}$ generated independently and uniformly at random from $[0, 1)$ to perform sampling. Specifically, to obtain a sample from a distribution over a variable $Y \in \{0, 1\}$ using the random number a_i, pick $Y = 0$ if $a_i < \Pr(Y = 0)$, and pick $Y = 1$ if $a_i \geq \Pr(Y = 0)$. Similarly, to obtain a sample from a distribution over a variable $Z \in \{A, B\}$ using the random number a_i, pick $Z = A$ if $a_i < \Pr(Z = A)$, and pick $Z = B$ if $a_i \geq \Pr(Z = A)$. Use the random numbers $\{a_i\}$ in order starting from a_1, using a new random number each time a sample needs to be obtained.

(c) Use likelihood-weighted sampling to obtain 2 samples from the distribution $\Pr(X_1, X_2 | O_1 = A, O_2 = B)$, and then use these samples to estimate $\mathrm{E}[\sqrt{X_1 + 3X_2} | O_1 = A, O_2 = B]$.

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
0.134	0.847	0.764	0.255	0.495	0.449	0.652	0.789	0.094	0.028

Sample 1: Sampling $X_1 = 0$ since $a_1 = 0.134 < 0.3$. Sampling $X_2 = 1$ since $a_2 = 0.847 \geq 0.4$. Weight is $0.9 \cdot 0.5 = 0.45$.

Sample 2: Sampling $X_1 = 1$ since $a_3 = 0.764 \geq 0.3$. Sampling $X_2 = 0$ since $a_4 = 0.255 < 0.8$. Weight is $0.5 \cdot 0.1 = 0.05$.

$\mathrm{E}[\sqrt{X_1 + 3X_2}] = \frac{.45}{.5}\sqrt{1(0) + 1(1)} + \frac{.05}{0.5}\sqrt{1(1)1(0)} = 1.$

Weights are calculated as the product of the probabilities $p(O_t | X_t)$ of the evidence variables given the parents.

(d) [*true* or *false*] In general, particle filtering using a single particle is equivalent to rejection sampling in the case that there is no evidence. Explain your answer.

Both will produce samples from the prior. Rejection sampling without evidence never rejects, so it is equivalent to prior sampling.

(e) [*true* or *false*] Performing particle filtering twice, each time with 50 particles, is equivalent to performing particle filtering once with 100 particles. Explain your answer.

False. Due to re-weighting, it is not equivalent. (Consider the case of 1 particle vs. 2 particles.)

(f) [*true* or *false*] Variable elimination is generally more accurate than the Forward algorithm. Explain your answer.

They both perform exact inference.