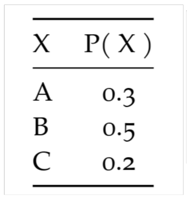
CSCI 4511/6511 - Exam Prep 6

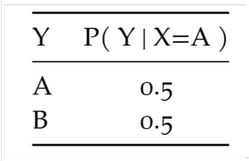
Instructions:
This is ungraded exam prep.

1 Bayesian Update

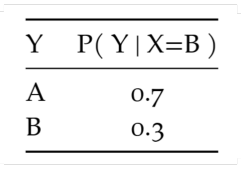
Give the following probabilities:



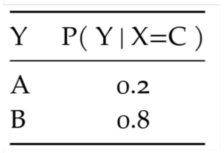
X	P(X)
A	0.3
B	0.5
C	0.2



Y	P(Y | X=A)
A	0.5
B	0.5



Y	P(Y | X=B)
A	0.7
B	0.3



Y	P(Y | X=C)
A	0.2
B	0.8

1.1

Compute $P(X,Y)$

1.2

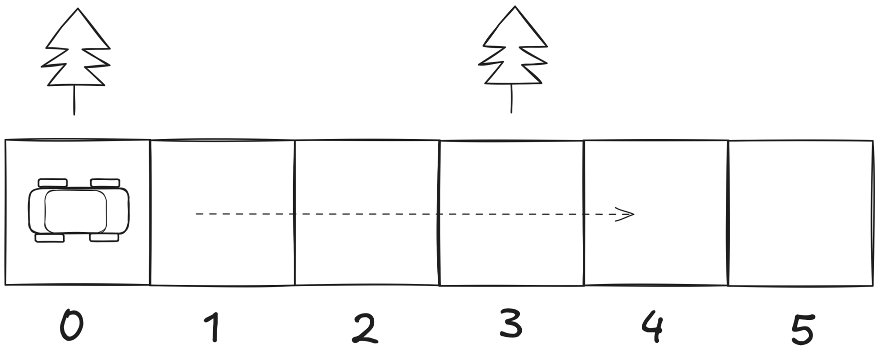
Compute $P(Y)$

1.3

Compute $P(X|Y)$

2 Discrete State Filter

A self-driving car moves through city blocks, some of which are adjacent to parks. Parks occur every three blocks, and are represented in the diagram below by trees:



- At each time step,[1] the car *attempts* to move forward one block. It actually moves forward with probability 0.9, and remains stationary with probability 0.1.
- If the car is adjacent to a park, it observes a park with probability 0.8 and does not observe a park with probability 0.2
- If the car is not adjacent to a park, it observes a park with probability 0.1 and does not observe a park with probability 0.9

2.1 No Observations

Ignore the possibility of observations. What is the probability distribution over positions if the car starts in position zero and three time steps elapse?

[1]Time is discretized into arbitrary-length "steps" for this problem.

2.2 Observations

Stop ignoring the possibility of observations. What is the probability distribution over positions if the car starts in position zero, three time steps elapse, and the follow sequence of observations is observed: {No Park, Park, Park} ?

3 The More You Sample

3.1 `move_car`

Write a Python function `move_car` that uses a call to `random.random()` as a source [2] of randomness, and which simulates the movement of the car at a time step. Your function should return `True` if the car moves, and `False` if it doesn't.

- Use only one comparison
- A `random.random()` value less than 0.05 should return `True`

3.2 `weighted_sample`

Write a function `weighted_sample` that uses a call to `random.random()` as a source of randomness, and which samples from a list of weighted values. Each item in the list will be a tuple, in the format `[(state0, weight0), (state1, weight1), ...etc .]`. The function should return *one* new state.

[2] `random.random()` returns a float uniformly randomly distributed between 0 and 1

3.3 Randomness

Use these values for each call to `move_car`:

Particle	Time Step	Random Value
A	0	0.8467518088104002
B	0	0.7146373451674315
C	0	0.41584110620899706
D	0	0.7812495155120517
E	0	0.884257718376851
—	—	—
A	1	0.27484050847267993
B	1	0.9112617351411147
C	1	0.8422494653415566
D	1	0.8523684011924184
E	1	0.27612766183353366
—	—	—
A	2	0.8477253993755229
B	2	0.8585095769033313
C	2	0.797864118588275
D	2	0.880904752314263
E	2	0.43107299999957605

Use these values for each call to `weighted_sample`:

Particle	Time Step	Random Value
A	0	0.7289570801647197
B	0	0.6206331687900658
C	0	0.8557638172230831
D	0	0.7222662675029506
E	0	0.5038410037791823
—	—	—
A	1	0.6733044776687852
B	1	0.6453856411241969
C	1	0.8540311221772551
D	1	0.7229956208750197
E	1	0.23682063509754958
—	—	—
A	2	0.3149872725045424
B	2	0.04021201432488919
C	2	0.28151458691996367
D	2	0.7322482515505646
E	2	0.37113472084694754

4 Particle Filter

We will use the following particle filter algorithm:

Algorithm 1 Particle Filter

```
function UpdateBelief($b_t, a, o$)
    $b'_{t+1} \leftarrow \emptyset$
    for $i \in \{0, 1, ...\text{Size}(b_t)\}$ do
        $s'_i \sim G(s_i, a)$
        $w_i \leftarrow O(o|s'_i, a)$
        $b'_{t+1} = b'_{t+1} + (s'_i, w_i)$
    $b_{t+1} \leftarrow \emptyset$
    for _ $\in \{0, 1, ...\text{Size}(b_t)\}$ do
        $k \leftarrow \text{Sample}(b'_{t+1})$
        $b_{t+1} = b_{t+1} + k$
    return $b_{t+1}$
```

- $G(s, a)$ is the `move_car` function (the action is constant)
- $O(o|s, a)$ is the observation probability
- Sample(b') is the `weighted_sample` function

Create a belief state b_0 of five particles (call them A, B, C, D, and E) that represent the car's position. Initialize each particle in state 0 at time 0.

4.1 Generate Forward

Use calls to `move_car` to generate each particle forward in time by one time step, from b_0 to b_1'. For each call, use the random numbers previously specified for time 0.

- Write the new positions below. Use a table, labeling each particle (A through E), the old state, and the new state.
- For each particle, calculate a weight $P(b|O)$ based on the observation("No Park").[3]

[3]Probability of the belief conditioned on the observation: for this filter, you'll use $P(O|s)$, where s is the state for each particle.

4.2 Resample

Use calls to `weighted_sample` to resample each particle based on the previous distribution, updating from b'_1 to b_1. Use the random numbers previously specified for time 0.

Formatting note: For a hypothetical belief state with three particles:

Particle	State	Weight
A	1	0.4
B	0	0.1
C	2	0.5

The list format for `weighted_sample` would be `[(1, 0.4), (0, 0.1), (2, 0.5)]`

4.3 Forward

Perform two more complete belief updates: generate forward, weight, resample. Show your results using tables for b'_2, b_2, b'_3 and b_3.

This is the back of the exam prep.