Midterm Project

Feb 5, 2021

In the midterm project, you will calculate the value of information using an adapted ocean drilling example.

Background story

You are the boss of an oil drilling company. Your company found N potential oil wells, but one and only one of these wells contains oil. You decide the buy one well, hoping it contains oil so that you can make a big fortune. You did some research and formed a prior of which well contains oil. Then a microbiologist comes to you, saying that she has a test to analyze microbes in the ocean. The task can be only performed in one well chosen by you. If there is oil in the examined well, a certain type of microbe is more likely to be detected. She asks you how much you want to pay for the test.

To calculate the value of information, we need to be able to 1) perform Bayesian Inference given a piece of evidence and 2) calculate expected utility.

Module 1 - Bayesian Inference

In this module, you are supposed to complete the 5 functions: **getUnnormalizedPosterior**, **normalize**, **getSumOfProbability**, **getPosterior**, **getMarginalOfData**.

Task 1: Calculate the Posterior The formula is

$$P(s|e) = \frac{P(e|s)P(s)}{\sum_s P(e|s)P(s)}$$

.

You are supposed to complete the functions **getUnnormalizedPosterior**, **normalize**, and **getPosterior** in the *valueOfInfo.py* file.

1. Calculate unnormalized posterior $P(e|s)P(s)$

There are two input parameters for the function **getUnnormalizedPosterior**.

(a) prior - A dictionary representing the prior of the state. The keys of dictionary are the possible states s and the values of the dictionary are the corresponding probability $p(s)$.
(b) likelihood - A dictionary representing the likelihood $P(e|s)$ - the probability of collecting evidence e given the state is s. The keys of the dictionary are the possible states s and the values of the dictionary are the corresponding likelihood $P(e|s)$.

The return value of the function **getPosterior** is a dictionary representing the unnormalized posterior $P(e|s)P(s)$. The keys of the dictionary are the possible states s and the values of the dictionary are the posterior probabilities $PP(e|s)P(s)$.

2. Normalization

Perform a normalization on a unnormalized probability distribution. $P(x) = \frac{P_{unnormalized}(x)}{\sum_x P_{unnormalized}(x)}$.

The function *normalize* takes in one parameter, which is a dictionary representing a probability distribution. It returns a dictionary, the normalized version of the input probability distribution.

3. Calculate the posterior:

Combine *getUnnormalizedPosterior* and *normalize* to complete the function **getPosterior**.

There are two input parameters for the function **getPosterior**.

(a) prior - A dictionary representing the prior of the state. The keys of dictionary are the possible states s and the values of the dictionary are the corresponding probability $p(s)$.

(b) likelihood - A dictionary representing the likelihood $P(e|s)$ - the probability of collecting evidence e given the state is s. The keys of the dictionary are the possible states s and the values of the dictionary are the corresponding likelihood $P(e|s)$.

The return value of the function **getPosterior** is a dictionary representing the normalized posterior $P(s|e)$. The keys of the dictionary are the possible states s and the values of the dictionary are the posterior probabilities $P(s|e)$.

Task 2: Calculate the marginal probability of evidence The formula is

$$P(e) = \sum_s P(e|s)P(s)$$

.

You are supposed to complete the functions **getSumOfProbability** and **getMarginalOfData** in the *valueOfInfo.py* file.

1. Sum up the probability

Sum up the probabilities in a probability distribution. $\sum_x P_{unnormalized}(x)$.

The function *getSumOfProbability* takes in one parameter, which is a dictionary representing a probability distribution. It returns a scalar, the sum of the probabilities in the input probability distribution.

2. Calculate the marginal probability of evidence

Combine *getUnnormalizedPosterior* and *getSumOfProbability* to complete the function **getMarginalOfData**..

There are two input parameters for the function **getPosterior**.

(a) prior - A dictionary representing the prior of the state. The keys of dictionary are the possible states s and the values of the dictionary are the corresponding probability $p(s)$.
(b) likelihood - A dictionary representing the likelihood $P(e|s)$ - the probability of collecting evidence e given the state is s. The keys of the dictionary are the possible states s and the values of the dictionary are the corresponding likelihood $P(e|s)$.

The return value of the function **getMarginalOfData** is a scalar representing the marginal probability $P(e)$.

Module 2 - Calculate expected utility

In this module, you are supposed to complete the 2 functions: **getEU**, **getMaxEUFull**.

Task 3: expected utility of an action The formula is

$$EU(a) = \sum_s R(a,s)P(s)$$

.

Since there is no state transition in our problem, we need to calculate expected utility of an action a by calculating the expected reward received by taking the action.

You are supposed to complete the function **getEU** in the *valueOfInfo.py* file.

There are three input parameters for the function **getEU**.

(a) action - a string indicating which action a is being evaluated.
(b) sDistribution - A dictionary representing the probability distribution of the current state s. The keys of dictionary are the possible states s and the values of the dictionary are the corresponding probability $p(s)$.
(c) rewardTable - A dictionary representing the reward from taking action a in state s. It is represented as $\{s : \{a : R(a,s)\}\}$.

The return value of the function **getEU** is a scalar representing the expected utility of the input action a.

Task 4: maximum utility given a piece of evidence The formula is $E(\alpha|e) = \max_a E(a|e)$.

Calculate the expected utility for each possible action and choose the maximum.

You are supposed to complete the function **getMaxEUFull** in the *valueOfInfo.py* file.

There are five input parameters for the function **getMaxEUFull**.

(a) evidence - A string indicating the evidence e you collected. It can be input as None. **If e_j is input as None, use the prior to calculate** $E(\alpha)$.
(b) prior - A dictionary representing the probability distribution before any new evidence is collected. The keys of dictionary are the possible states s and the values of the dictionary are the corresponding probability $p(s)$.
(c) likelihoodTable - A nested dictionary representing the likelihood for each evidence $P(e|s)$. The dictionary is in the format of $\{e : \{s : \{P(e|s)\}\}$.
(d) rewardTable - A dictionary representing the reward from taking action a in state s. It is represented as $\{s : \{a : R(a,s)\}\}$.
(e) actionSpace - A list of all possible actions

The return value of the function **getMaxEUFull** is a scalar representing the maximum expected utility after receiving e_j. $E(\alpha|e_j)$. You should use the **getEU** function.

The **getMaxEUFull** function will be transformed in the **main** function to only take parameters “test” and “evidence”.

Module 3 - Calculate the value of information

Task 5: value of information of a given test The formula is

$$VPI(E_j) = \sum_{e_{jk}} P(e_{jk}) EU(\alpha|e_{jk}) - EU(\alpha)$$

You are supposed to complete the function **getValueOfInformationFull** in the *valueOfInfo.py* file.

The **getValueOfInformationFull** function takes four parameters:

(a) prior - A dictionary representing the probability distribution before any new evidence is collected. The keys of dictionary are the possible states s and the values of the dictionary are the corresponding probability $p(s)$.
(b) evidenceSpace - A list of all possible evidence you can get from the test.
(c) getMarginalOfEvidence - A function that calculates the marginal probability of each evidence $P(e)$. Its input argument is a piece of evidence in evidenceSpace. Its return value is a scalar representing the marginal probability $P(e)$
(d) getMaxEU - A function that calculates the maximum expected utility you can get after receiving a piece of evidence $EU(\alpha|e)$. Its input argument is a piece of evidence in evidenceSpace, or None if you calculate $E(\alpha)$ for the prior. Its return value is a scalar representing the maximum expected utility $EU(\alpha|e)$ or $EU(\alpha)$.

The return value of the function **getValueOfInformationFull** is a scalar, representing the value of the given test.

The **getValueOfInformationFull** function will be transformed in the **main** function to only take parameter “test” so that it can calculate value of information for different tests.

Information in main function

The parameters of one example problem are provided in the main function.

The prior - There are 4 potential oil wells. You think that the probability that Well 2 contains oil is 0.4. The probability for each other well to contain oil is 0.2.

The action space - You have 4 possible actions: buy one of the wells.

The reward table - If you buy the well that contains oil, you earn 100 millions. If you buy a well without oil, you do not earn or lose money. The reward table is a nested dictionary $\{s : \{a : R(s, a)\}\}$, with the value being the reward from taking action a in state s.

The test space - You have 4 possible tests to take. You can try each of them with your algorithm.

The evidence space - Each test can give you 2 potential results: the microbe detected or not detectd.

The likelihood table: given one state to be true, the probability of collecting the evidence through the test. It is a nested dictionary $\{test : \{evidence : \{s : P(e|s)\}\}\}$. If there is oil in the tested well, the probability of microbes being detected is 0.8, the probability that they are not detected is 0.2. If there is no oil in the tested well, the probability of microbes being detected is 0.1, the probability that they are not detected is 0.9.

Submission

Please submit a completed *valueOfInfo_YourLastName_YourFirstName.py* file on CCLE before due. **The due date and time of this homework assignment is Monday, 02/21/2021 11:59pm**.