Struggling with part (b) or maybe (a) still

question posted 8 days ago by gontia0

I don't know if my part (a) is correct or not as part (b) is essentially part a in code. The answer I am getting with my code is [0.7115389, 0.06189829], clearly not even summing to 1. I even did this manually and still getting the same answer, which leads me to believe, I made a mistake in part (a)

My approach was as follows

$$p_{X,Y_0,Y_1\ldots Y_{N-1}} = p_X(x) \cdot \prod_{n=0}^{N-1} p_{Y|X}(y_n|x)$$

Since,

$$p_{X|Y_0,\ldots Y_{N-1}} = \frac{p_{X,Y_0,\ldots Y_{N-1}}}{p_{Y_0,\ldots Y_{N-1}}}$$

Therefore, on substituting and using the fact that Y_i

are i.i.d

$$p_{X|Y_0,\ldots Y_{N-1}} = \frac{p_X(x) \cdot \prod_{n=0}^{N-1} p_{Y|X}(y_n|x)}{\prod_{n=0}^{N-1} p_Y(y_n)}$$

Where am I going wrong?

Also, I realized that normalizing these two values gets me the answer (or maybe very close to it, since I only have so many precision digits). But I don't understand why I need to normalize, if that is to be done.

Related to: Mini-project 1 / Mini-project: Movie Recommendations

This post is visible to everyone.

Add a Response

1 response

jheadwood

8 days ago

same problem here

I don't understand why I should normalize the pX|Y0..YN-1

posted 8 days ago by **jheadwood**

Typesetting math: 100%

Support

Edit: my claim happens to be wrong. $p_Y(y) \neq \prod_n p_{Y_n}(y_n)$, because $p_{Y_n}(y_n)$ are not independent. Independent (iid) are $p_{Y_n|X}(y_n|x)$

You are perfectly right.
$\prod_n p_Y(y_n) = \prod_n \Sigma_x p_{Y,X}(y_n, x) = \prod_n \Sigma_x p_{Y|X}(y_n|x) \cdot p_X(x) = \Sigma_x p_x(x) \cdot \prod_n p_{Y|X}(y_n|x)$
.

1. Marginalizing join distribution
2. Product rule
3. Change the order of summation and multiplication

And the last equation is a sum of numerator (normalizing constant).

posted 8 days ago by Mark_B2 Community TA

That is as close as I've gotten and beyond that I haven't a clue: log_answer = np.log(likelihood[y, :]).sum(axis = 0) + log_prior

log_answer=log_answer / scipy.misc.logsumexp(np.dot(np.log(likelihood[y, :]),log_prior))

posted 8 days ago by seanedXacc

$A = \frac{B}{C}$ in log domain looks like $log(A) = log(B) - log(C)$, thus it's more like

```
log_answer = log_answer - scipy.misc.logsumexp(log_answer)
# or
log_answer -= scipy.misc.logsumexp(log_answer)
```

posted 8 days ago by Mark_B2 Community TA

ya when I switched from divide to minus I got [0.00127637 0.00011119] so I must be doing something very wrong.

posted 8 days ago by **seanedXacc**

I recommend you take the main example with calculator (or python interpreter) and try to understand what are you doing. To check log_answer you should exponentiate it first np.exp(log_answer).

posted 8 days ago by **Mark_B2** Community TA

Can you clarify for me: prior gives 2 P(x) probabilities (0.6, 0.4), likelihood gives a joint probability table P(x,y) with four values for P(x|y), and y gives the observed Y=y values for 3 experiments, Y = y [0, 0, 1].

So is it .7 P(x=0|y=0) and .3 P(x=1|y=0) etc?

posted 7 days ago by **seanedXacc**

likelihood gives P(y|x)

posted 7 days ago by **gontia0**

@Mark_B2 thanks for the help. But I have another problem. What I did was find first the distribution $p_{x,y}(x, y)$ and marginalising it to find p_y and then I used this p_y in the denominator of the expression I showed above. What am I doing wrong? Did I find the distribution incorrectly or is perhaps my approach incorrect?

posted 7 days ago by **gontia0**

$$p_{X,Y}(x=0,Y) = p_X(x=0)\cdot p_{Y|X}(y=0|x=0)\cdot p_{Y|X}(y=0|x=0)\cdot p_{Y|X}(y=1|x=0)$$
$$= 0.6\cdot 0.7\cdot 0.7\cdot 0.3 = 0.0882$$

$$p_{X,Y}(x=1,Y) = p_X(x=1)\cdot p_{Y|X}(y=0|x=1)\cdot p_{Y|X}(y=0|x=1)\cdot p_{Y|X}(y=1|x=1)$$
$$= 0.4\cdot 0.98\cdot 0.98\cdot 0.02 = 0.0076832$$

After normalzation $p_{X|Y} = \frac{p_{X,Y}}{p_Y} = \{0.919869174161897, 0.08013082583810303\}$

posted 7 days ago by **Mark_B2** Community TA

---

are you getting P(y= 0) = 0.812 when you did that Px,y joint probability table gontia?

posted 7 days ago by **seanedXacc**

@gontia0 You are right, corrected my mistake. See above.

posted 7 days ago by **Mark_B2** Community TA

@seanedXacc Yes i get 0.812

posted 7 days ago by **gontia0**

@Mark_B2 Thanks. But I do not understand how you interchanged the summation and the product. If you manually do calculations for both, the results would be different. That is the reason we are getting different answers.

At least, it is clear to me now, that the mistake I'm making, is in calculating the $\prod_n p_Y(y_n)$.

posted 7 days ago by **gontia0**

Typesetting math: 100%

Support

Summation is by x and multiplication is by n in y_n. They are interchangeable.

posted 7 days ago by Mark_B2 Community TA

But that would mean

$$\prod_n a y_n = a \cdot \prod_n y_n$$

posted 7 days ago by gontia0

$p_X(x)$ doesn't depend on y

posted 7 days ago by Mark_B2 Community TA

welp you got me close Mark, I've managed to get the right answer doing manual inputs.
Now to try get the np.logs and such to work :O

posted 7 days ago by seanedXacc

Please don't post answers (even if they are wrong) before the deadline.

posted 7 days ago by RADUGROSU Community TA

Sorry Radugrosu, I deleted it. I have the numerators properly done in np.log python code form using the numpy arrays, I confirmed that by printing out the np.exp of my numerators. When I converted my numerators using np.exp and did the division I get the right answer, but I cannot figure out how to get the right answer in logspace. I tried using - instead of divide but its giving me 130.15 and 11.337 as the answer when I do that lol

posted 7 days ago by seanedXacc

@seanedXacc You try to mix memberwise and nd-array attitude. It will not work. If you managed to get right answer in memberwise access, do the same in log domain (still memberwise). Then we'll see ...

posted 7 days ago by **Mark_B2** Community TA

Mark you should be a community TA for this subject! I'm guessing I have to convert this algorithm into a loop which iterates i = 0 i++ through the n values for the x and y arrays so that it can receive the input from the large data set. I'm not sure what you mean by memberwise and nd-array attitude, but I am researching memberwise now.

Somehow its the pre np.exp numerators that are giving me problems. I tried: denom1 = scipy.misc.logsumexp(num1 - (num2 + num1)) denom2 = scipy.misc.logsumexp(num2 - (num1 + num2)) I tried these both without logsumexp also and get the same answer. log_answer = denom1, demon2 posterior = np.exp(log_answer) and I get 130 & 11. Cannot figure out how to normalize them.

Same thing happens when I use nd array language like: log_answer = np.array([num1, num2]) log_answer = np.subtract(log_answer, np.sum([num1,num2]))

if I do np.exp(num1) and np.exp(num2) and then use those with division I get the right answer. So its the num1 - num1 - num2 that isn't working out and I cannot figure out why.

posted 7 days ago by **seanedXacc**

What do you get for P(y= 2)? Like gontia at the start of this thread I am close at 0.7115389. Numerator is right and P(y=0) and P(y=1) at 0.812 but P(y=2) is at 0.188

Got! - Agree with the above - Mark - very big thank you

posted 6 days ago by **marklepla**

@Mark_B2

Just to see we are in the same page, let me tell you my approach

I calculated the denominator this way -

Typesetting math: 100%

Support

I calculated the denominator this way

$$p_{Y_0,Y_1,\ldots,Y_{N-1}} = \prod_n p_Y(y_n)$$

$$\prod_n p_Y(y_n) = \prod_n \sum_x p_X(x) \cdot p_{Y|X}(y_n|x)$$

Now, I understand that you interchanged the summation and the product, getting

$$\prod_n p_Y(y_n) = \sum_x \prod_n p_X(x) \cdot p_{Y|X}(y_n|x)$$

and then took $p_X(x)$ out of the product since it is a constant w.r.t n getting

$$\prod_n p_Y(y_n) = \sum_x p_X(x) \cdot \prod_n p_{Y|X}(y_n|x)$$

However, I'm still against this one step of interchanging the product and summation signs. Because, let's say I have a term $a_x \cdot b_n$ in which a_x depends only on the index $x \in \{1,2,3\}$ and b_n depends only on its index $n \in \{1,2,3\}$ Then,

$$\prod_n \sum_x a_x \cdot b_n$$

would be

$$\prod_n b_n \cdot (a_1 + a_2 + a_3) = b_1 \cdot b_2 \cdot b_3 \cdot (a_1 + a_2 + a_3)^3$$

and If I were to just simply interchange the symbols, I would get,

$$\sum_x \prod_n a_x \cdot b_n$$

which equals

$$\sum_x a_x^3 \cdot b_1 \cdot b_2 \cdot b_3 = b_1 \cdot b_2 \cdot b_3 \cdot \sum_x a_x^3$$

which is equal to

$$b_1 \cdot b_2 \cdot b_3 \cdot (a_1^3 + a_2^3 + a_3^3)$$

Clearly, both the expressions are not equal. Is there some property I am missing ? or maybe some detail in the question itself?

posted 6 days ago by **gontia0**

Typesetting math: 100%

You are right, my derivation is flowed. In such equation $\sum$ and $\prod$ are **not** interchangeable. I tried to shoe derivation to known result.

On the other hand, your initial premise that y_i's are i.i.d and therefore

$$p_Y = \prod_{n=0}^{N-1} p_Y(y_n)$$

is wrong. They are conditionally independent on given x (see here) and thus marginalizing joint distribution

$$p_Y = \sum_x p_{X,Y}(x,y) = \sum_x p_X(x) \cdot \prod_{n=0}^{N-1} p_{Y|X}(y_n|x)$$

As aside note this case could be good example why it's so important for me to participate in forum discussions. Only trying to explain issue one can check his own understanding.

posted 6 days ago by **Mark_B2** Community TA

Support

Typesetting math: 100%

Support

Glad it worked out for you Mark. As for me, I have got the correct answer/formula in logspace but when the posterior np.exp runs on my result I get the unnormalized answer of the expected answer. If I normalize my answer manually I get the expected answer. I tried using logsumexp() on my array and on my variables before putting them into the array but it will not normalize them, I keep getting the unnormalized answer.

If I do this I get no answer at all when I run my program: log_answer = scipy.misc.logsumexp(log_answer). I am stuck.

posted 6 days ago by **seanedXacc**

@seanedXacc I ment take this 6 numbers with a pen and a paper (and calculator) and try to get to the right answer.

posted 6 days ago by **Mark_B2** Community TA

I have the right answer I just cannot get it to normalize from My Answer: [[130.15410246 11.33786848]] to Expected answer: [[0.91986917 0.08013083]] using scipy.misc.logsumexp or even just doing the normalisation division manually.

posted 6 days ago by **seanedXacc**

@Mark_B2 oh that clears it. Thanks for your support.

posted 6 days ago by **gontia0**

---

omg I finally figured it out.. I was using logsumexp on the wrong thing.....

posted 6 days ago by **seanedXacc**

Typesetting math: 100%

Support

@Mark_B2

Thanks so much this thread was a huge help. I think I might actually have eventually gotten it on my own but I had so little confidence that I knew what I was doing that I couldn't get myself to even start coding and even considered dropping the class at one low point. But this thread helped immensely and I just got the expected answer from the little test of compute_posterior() on the first run of my code. I could not believe it. It's only the first baby step but at least I'm finally started.

And after writing the code the concepts seems clearer and in a way simpler than seemed after re-watching some of the videos a few times, googling for explanations, and juggling equations with pencil and paper. I guess that just goes to shows that I've been programming too long and have gotten to the point where I can only understand things when they're written out in code. ;)

posted 5 days ago by **pymike**

A journey of a thousand miles begins with a single step

posted 5 days ago by **Mark_B2** Community TA

@Mark_B2

$$p_{X,Y}(x=0,Y) = p_X(x=0)\cdot p_{Y|X}(y=0|x=0)\cdot p_{Y|X}(y=0|x=0)\cdot p_{Y|X}(y=1|x=0)$$
$$= 0.6\cdot 0.7\cdot 0.7\cdot 0.3 = 0.0882$$

Why are we multiplying 0.7 twice here ?

posted 3 days ago by **bicepjai**

Because vector $Y = \{0,0,1\}$.

posted 3 days ago by **Mark_B2** Community TA

Part b: I got the correct answer for the simple 2x2 test case without using logs. I understand that for bigger calculations, using logs is critical for reasons mentioned in the assignment,. I can see that logs would be easy to apply for the numerator as it is just one product term. I am struggling to figure out how to apply the log to the denominator which is a sum of potentially many products. Would appreciate any hints on how to address log(A+B+C....). Is the approach to use logs to calculate the products - then exponentiate them to get A, B, C - add them and then take the log of the sum? Thanks.

posted 2 days ago by **SudipChahal**

Typesetting math: 100%

You want this beautiful little function Sudip

```
scipy.misc.logsumexp
```

posted 2 days ago by **kiwitrader** Community TA

See wiki how to use it.

posted 2 days ago by **Mark_B2** Community TA

Thank you for the help - greatly appreciated.

posted a day ago by **SudipChahal**

Add a comment

Support

Showing all responses

Add a response:

Preview

Typesetting math: 100%

Submit

Support