Backward Messages: Python Dictionary Approach

discussion posted 3 days ago by Upul

I'm really struggling with `forward_backward` calculation and with the help of forum posts, finally managed to correctly implement forward messages.

However, I could not still figure out, how to implement backward messages correctly. Presently, I'm using the following algorithm for calculating backward messages but it looks like it has some flaws.

1. Initialize the last message of the `backward_messages` with the `prior_distribution` (i.e. `backward_messages[num_time_steps -1 ] = prior_distribution`)
2. For each time step starting from num_time_steps -2 to 0
3. Initialize new `distribution = robot.Distribution()` and `previous_step = backward_messages[time_step + 1]`.
4. for each state in previous_step
5. `observation = observation_model(state)` and `transition = transition_model(state)`
6. For each `next_state` in `transition`:

Mp2: Samples: observation & transition matrices, plus elements of 1 forward & 1 reverse message 33
Pinned

Mini-project 2 deadline extended until Nov 9, 5pm Eastern Time (beware: daylight savings ends!) 6
Pinned Staff

Viterbi part(c)-(d) 23

Mp2: Tips for part (d) 66

checking implementations in real life 5

Forward Message Implementation with Dictionaries (not numpy!) 9 new 11

marginal calculation part a 12

Dictionary Tips & Tricks, Struggling 7

fwd message correct, back msg & marginals WRONG 20

Backward Messages: Python Dictionary Approach 8

Mp2 robot: Getting started... 10

Stuck in part a: forward and backward messages (almost) never match 18 new 26

Support

7. `trans_val = transition[next_state]`, `obser_val = observation[observations[time_step + 1]]` and `pre_val = previous_step[state]`
8. Next update `distribution[next_state]` with `trans_val * obser_val* pre_val`
9. Finally, renormalize `distribution` and add it to the `backward_messages`

I ran the python inference.py --load=test.txt and inspected some backward messages but messages and bit off from expected messages. For instance:

time step, i = 3 (what I'm getting)

```
(6, 7, 'left'):0.8099999999999998
(7, 7, 'left'):0.024999999999999994
(9, 7, 'right'):0.024999999999999994
(8, 7, 'stay'):0.024999999999999994
(8, 6, 'up'):0.024999999999999994
(7, 7, 'stay'):0.09
```

But expected output:

```
 (8, 6, 'down'): 0.13385885327582356,
(7, 7, 'left'): 0.01338588532758236,
 (8, 7, 'left'): 0.0037183014798839887,
 (9, 7, 'left'): 0.013385885327582357,
 (9, 7, 'stay'): 0.0037183014798839887,
 (8, 7, 'stay'): 0.009295753699709971,
 (7, 7, 'right'): 0.026771770655164714,
(7, 7, 'down'): 0.13385885327582359,
 (8, 7, 'down'): 0.037183014798839883,
 (8, 7, 'right'): 0.0037183014798839887,
(7, 7, 'stay'): 0.03718301479883989,
(6, 7, 'stay'): 0.12047296794824124,
(8, 6, 'stay'): 0.02974641183907191,
 (6, 7, 'right'): 0.4337026846136685
```

Hidden Markov Models --

So you help regarding this is highly appreciated!

Related to: Mini-project 2 / Mini-project: Robot Localization

This post is visible to everyone.

Add a Response

1 response

Support

Upul

3 days ago

Any help regarding this would be greatly appreciated.

The most fruitful way is to solve Formulating HMM's example https://courses.edx.org/courses/course-v1:MITx+6.008.1x+3T2016/courseware/2__Inference_in_Graphical_Models/hmm_marg/ by your program. Try it.

posted 3 days ago by Mark_B2 Community TA

I believe you do not have the logic right for finding the site potentials (nd also the transitions between states). You are given Xj+1 and Yj but you need Xj. So you need to find all possible Xj such that transformation_model(Xj)= one of your Xj+1 keys and Xj is in the nearest neighbor cross of Yj. In many cases, this only hurts you at the boundaries because the site potentials are always the same no matter where you are except at the boundaries. This is why it is hard to debug. My advice is be sure you have completely correct formulas you are evaluating for the backward trace, and be sure you find all of the information you need to determine the specific values. For me, if I set up this problem using just dictionaries, I would set up two of them. One for the forward pass and one for the backward, because it is so easy to make a mistake in the backward pass, but this is not an option for us the way the code and grader are set up... But if you persevere you will get it to work.

posted 3 days ago by Jim_Freericks

Support

Hi - I'm at a similar stage. That is using dictionaries with the forward messages working - but debugging the backward messages. I have a question on the boundary condition and I see it mentioned in other posts. However, I don't fully understand the concern. From what I can see the functions for returning the observation and transition probabilities already account for boundary conditions. For example if a enter a (X,Y) coordinate near the centre of the grid the possible hidden state probs will each have .2. If I enter a (X,Y) coordinate near a boundary the possible hidden state probs will each have .25 etc. So if we use this functions we don't need to account for the boundary conditions ourselves. Or am I missing something? Any inputs appreciated.

posted 3 days ago by jmoranrun

You have to be sure to call it with the coordinates of X the hidden variable and not of Y the observation. The initial inclination is to do the opposite (at least it was for me). These give different answers depending on whether X or Y or both are on the boundary (can have X off boundary but Y on boundary, etc.) it is rare, but does happen and leads to small errors in the messages.

posted 3 days ago by Jim_Freericks

Thanks Jim. I see this comment in the function: def observation_model(state): # given a hidden state, return the Distribution for its observation What I had been doing is inserting the observed value into this function and using the resultant dictionary to look up site potentials using the hidden variable as a key. I now changed my implementation to insert the hidden variable into the function and use the observed value as a key in the resultant dictionary. My backward messages are looking better!

posted 2 days ago by jmoranrun

Debugging this can be painful. Stick with it!

posted a day ago by Jim_Freericks

Add a comment

Support

Showing all responses

Add a response:

Preview

Submit

© 2016 edX Inc. All rights reserved except where noted. EdX, Open edX and the edX and Open EdX logos are registered trademarks or trademarks of edX Inc.

POWERED BY OPEN edX

Download on the App Store
ANDROID APP ON Google play

Support