Personal Open source Business Explore Pricing Blog Support This repository Search Sign in Sign up

ApurvaDubey / EdX.ComputationalProbability.MIT6.008 Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Pulse Graphs

Branch: master EdX.ComputationalProbability.MIT6.008 / hmm_robot_toy_example.py Find file Copy path

ApurvaDubey Update hmm_robot_toy_example.py 3b62089 2 days ago

1 contributor

85 lines (59 sloc) 3.38 KB Raw Blame History

```
## Suppose you send your robot, Inquisition, to Mars. Unfortunately, it gets stuck in a canyon while landing and most of its sensors break.
## Every hour, it tries to move forward by one area (i.e. from Area 1 to Area 2, or Area 2 to Area 3). It succeeds with probability
## and fails with probability. If it fails, it stays where it is. If it is in Area 3, it always stays there (and waits to be rescued).
##
## The temperature sensor still works. Every hour, we get a binary reading telling us whether the robot's current environment is hot or col
## We have no idea where the robot initially got stuck.
##
## Let's construct an HMM for this problem

import numpy as np

########################################################################
# Inputs

# Emission probabilities
em_idx= {'hot':0, 'cold':1} # this dictionary is to be able to map observed state back to the index in the 'emission' matrix
emission = np.array([[1,0],[0,1],[1,0]])
```

```
18
19  # Transition probabilities
20  transition = np.array([[0.25,0.75,0],[0,0.25,0.75],[0,0,1]])
21
22  # Prior Distribution
23  prior = np.array([1/3.0,1/3.0,1/3.0])
24
25  # Observations
26  observations = ['hot','cold','hot']
27  ########################################################################
28
29  ########################################################################
30  # parameters for forward/backward algorithm
31  alpha = np.zeros(shape=(3,3)) # timesteps x hidden states
32  beta = np.zeros(shape=(3,3)) # timesteps x hidden states
33
34  print "Initial value of alpha and beta"
35  print alpha, '\n', beta
36
37  # initialize alpha and beta
38  alpha[0] = prior
39  beta[2] = 1.0/len(observations)
40
41  ########################################################################
42  print "Running forward algorithm"
43
44  # Go forward
45  for i in range(len(observations)-1):
46
47      alpha[i+1] = np.matrix(np.array(alpha[i]) # prior message
48                   * np.array(emission[:,em_idx[observations[i]]])) * np.matrix(transition)
49                  # emision probability x transition probability
50
51      alpha[i+1] = alpha[i+1]/sum(list(alpha[i+1])) # normalize
52
```

```
53      print "Iteration : " + str(i) + "\n" + str(alpha)
54
55  print "Final alpha is: \n ------------------------"
56  print alpha
57
58  # Go backward
59  print "Running backward algorithm"
60  for i in range(len(observations)-2,-1,-1):
61
62      beta[i] = np.matrix(np.array(beta[i+1]) # prior message
63                  * np.array(emission[:,em_idx[observations[i+1]]])) * np.transpose(np.matrix(transition))
64                 # emision probability x transition probability
65
66      beta[i] = beta[i]/sum(list(beta[i])) # normalize
67
68  print "Final beta is: \n ------------------------"
69  print beta
70  ########################################################################
71
72  ########################################################################
73  print "Calculating marginals"
74
75  marginals = np.zeros(shape=(3,3)) # timesteps x hidden states
76
77  for i,obs in enumerate(observations):
78
79      marginals[i] = (alpha[i]) * (beta[i]) * np.array(emission[:,em_idx[obs]]) # martignal = alpha x beta x emission probability
80      marginals[i] = marginals[i]/sum(list(marginals[i])) # normalize
81
82  print marginals
83  ########################################################################
84
```

© 2016 GitHub, Inc. Terms Privacy Security Status Help

Contact GitHub API Training Shop Blog About