AML Lab 2: Hidden Markov Models

Joshua Hudson

14 September 2017

(1)

The first step was to construct the Hidden Markov Model, by defining the unobserved/hidden variable states (named "states" in the HMM package), observed variable states ("symbols"), transmission model and emission model. The hidden variables states were the true positions of the robot, from sectors 1 to 10. The observed variable states were the positions of the robot as given by the robot's tracking device. The transition model defines probability distribution between the hidden states over a time step, i.e. the probabilities of the robot moving from one sector to another (or staying in the same one). Here we were told that the robot will with equal probability stay in the same sector or move to the next. The emission model defines the probability distribution between the hidden variable state and the observed variable state at the same time step, i.e. the probabilities that the robot will be reported by the tracking device as being in a sector j when it is actually in sector i. Note that B could be A if the tracking device is correct in that instance. We were told that if the tracking device can give a reading of up to ± 2 sectors off the true position with equal probability. I decided to start the robot in sector 1. Using this information we obtained the following HMM, as shown by the output of the `initHMM()` function:

```
print(hmm)
```

```
## $States
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $Symbols
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $startProbs
##  1  2  3  4  5  6  7  8  9 10
##  1  0  0  0  0  0  0  0  0  0
##
## $transProbs
##      to
## from   1   2   3   4   5   6   7   8   9  10
##   1  0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   2  0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   3  0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0
##   4  0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0
##   5  0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0
##   6  0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0
##   7  0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0
##   8  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0
##   9  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5
##   10 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5
##
## $emissionProbs
##        symbols
## states   1   2   3   4   5   6   7   8   9  10
##      1  0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2
##      2  0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2
```

```
##      3  0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0
##      4  0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0
##      5  0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0
##      6  0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0
##      7  0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0
##      8  0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2
##      9  0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2
##      10 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2
```

(2)

We then used this model to simulate from for 100 timesteps. Using the `simHMM()` function, we obtained the following observed and hidden path:

```
print(simpath)
```

```
## $states
##   [1]  1  1  1  1  2  3  3  4  5  5  6  6  7  7  7  7  7  7  7  8  8  8  9
##  [24]  9  9 10 10 10  1  2  2  3  3  4  4  4  5  5  5  6  7  7  8  9 10  1
##  [47]  2  3  3  4  5  5  6  6  7  7  8  8  8  8  9 10 10 10 10  1  1  2  2
##  [70]  2  2  2  3  3  3  4  5  5  5  6  7  8  8  8  8  8  9  9  9 10 10 10
##  [93]  1  1  1  1  1  1  2  3
##
## $observation
##   [1]  2 10  3 10  1  3  5  6  3  3  4  7  9  7  9  6  6  5  8  7 10 10  9
##  [24] 10 10  9  9 10  2 10  2  5  3  2  6  6  4  7  7  6  5  9  7 10 10  3
##  [47]  3  1  3  3  6  5  4  7  7  9  9 10  6  9 10  2  9  9  8  1  3  2  3
##  [70]  4  3  2  5  4  4  2  4  6  4  6  8 10  8  7  6  6  7  8  9 10  1  9
##  [93]  2  2  3  9  2 10  4  1
```

(3)

For this assignment, we put the hidden states aside and considered only the observed states obtained by simulation. Using these, we computed the filtered and smoothed posterior probability distributions. The filtered posterior at time t uses only the data uptil that timepoint so is given by: $p(z^t|x^{0:t})$. The smoothed posterior on the contrary uses all the data, uptil the last timestep T: $p(z^t|x^{0:T})$.

In order to obtain the filtered distribution, we used the `forward()` function to calculate the joint distribution of observing the observed states uptil time t and of the hidden states at this time t. As this function outputs the results in the logarithmic scale, we took the exponential of each probability. In order to get the filtered posterior, we used the `prop.table()` function to divide the joint by the Bayesian constant term. This is shown by the equations below:

$$P(z^t|x^{0:t}) = \frac{P(z^t, x^{0:t})}{P(x^{0:t})} = \frac{P(z^t, x^{0:t})}{\sum_i P(z_i^t, x^{0:t})}$$

Below is the filtered distribution for the last 10 timesteps as an example:

```
print(filterpost[, 91:100])
```

```
##       index
## states       91    92      93      94       95 96  97      98  99   100
##      1  0.322581 0.500 0.46667 0.36667 0.267857  1 0.5 0.33333 0.0 0.000
##      2  0.161290 0.000 0.26667 0.36667 0.392857  0 0.5 0.66667 0.6 0.375
##      3  0.032258 0.000 0.00000 0.13333 0.267857  0 0.0 0.00000 0.4 0.625
```

```
##     4  0.000000 0.000 0.00000 0.00000 0.071429  0 0.0 0.00000 0.0 0.000
##     5  0.000000 0.000 0.00000 0.00000 0.000000  0 0.0 0.00000 0.0 0.000
##     6  0.000000 0.000 0.00000 0.00000 0.000000  0 0.0 0.00000 0.0 0.000
##     7  0.000000 0.000 0.00000 0.00000 0.000000  0 0.0 0.00000 0.0 0.000
##     8  0.000000 0.000 0.00000 0.00000 0.000000  0 0.0 0.00000 0.0 0.000
##     9  0.161290 0.125 0.00000 0.00000 0.000000  0 0.0 0.00000 0.0 0.000
##     10 0.322581 0.375 0.26667 0.13333 0.000000  0 0.0 0.00000 0.0 0.000
```

Finding the smoothed posterior distribution was a little more straight-forward as the function `posterior()` computes this directly. Below is the smoothed posterior for the last 10 timesteps as an example:

```
print(smoothpost[, 91:100])
```

```
##       index
## states      91      92      93      94 95 96    97   98   99   100
##     1  0.13333 0.26667 0.46667 0.73333  1  1 0.625 0.25 0.00 0.000
##     2  0.00000 0.00000 0.00000 0.00000  0  0 0.375 0.75 0.75 0.375
##     3  0.00000 0.00000 0.00000 0.00000  0  0 0.000 0.00 0.25 0.625
##     4  0.00000 0.00000 0.00000 0.00000  0  0 0.000 0.00 0.00 0.000
##     5  0.00000 0.00000 0.00000 0.00000  0  0 0.000 0.00 0.00 0.000
##     6  0.00000 0.00000 0.00000 0.00000  0  0 0.000 0.00 0.00 0.000
##     7  0.00000 0.00000 0.00000 0.00000  0  0 0.000 0.00 0.00 0.000
##     8  0.00000 0.00000 0.00000 0.00000  0  0 0.000 0.00 0.00 0.000
##     9  0.33333 0.13333 0.00000 0.00000  0  0 0.000 0.00 0.00 0.000
##     10 0.53333 0.60000 0.53333 0.26667  0  0 0.000 0.00 0.00 0.000
```

We also used the Viterbi algorithm to compute the most probable path using the implemention inside the `viterbi()` function. The resulting path is shown below:

```
print(optpath)
```

```
##   [1]  1  1  1  1  1  2  3  4  4  4  5  6  7  7  7  7  7  7  8  9 10  1  1
##  [24]  1  1  1  1  1  1  1  2  3  3  3  4  4  4  5  5  6  7  8  9 10  1  1
##  [47]  1  1  2  3  4  4  4  5  6  7  7  8  8  8  9 10 10 10 10  1  1  1  1
##  [70]  2  2  2  3  3  3  3  3  4  5  6  7  8  8  8  8  8  9 10  1  1  1  1
##  [93]  1  1  1  1  1  1  2  2
```

(4)

Next we recovered the hidden states obtained through simulation in (2) in order to compute the accuracy of our filtered and smoothed posteriors and most probable path. We assumed that the robot took the position with the highest probability at each timesteps to obtain a path of states to compare to the true hidden states (majority voting). We then used the `table()` function to calculate the accuracy as a percentage.

Table 1: Accuracy of the 3 Methods (%)

	Filtered	Smoothed	Viterbi
TRUE	56	75	57

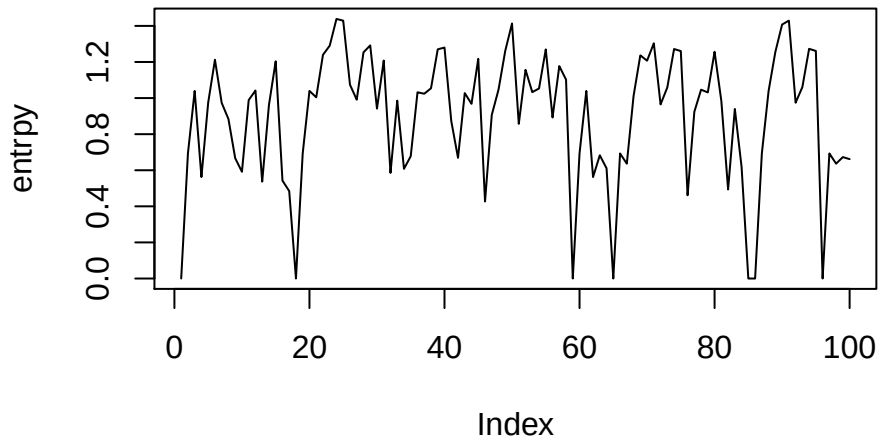
For this simulation, it is clear that the Smoothed Posterior is the most accurate by far, while the Viterbi and Filtered Posterior are almost equivalent in this respect.

(5)

Table 2: Accuracy of the 3 Methods (%) for 10 simulations

	Filtered	Smoothed	Viterbi
1	56.0	75.0	57
2	44.0	65.0	54
3	47.0	79.0	62
4	56.0	68.0	53
5	62.0	78.0	67
6	53.0	64.0	39
7	49.0	69.0	53
8	52.0	67.0	55
9	53.0	52.0	26
10	42.0	64.0	44
Mean	51.4	68.1	51

(6)



(7)

Joint

$$p(Z^{101}, Z^{100}|x^{1:100}) = p(Z^{100}|x^{1:100})p(Z^{101}|Z^{100})$$

marginalise out Z100