Lab 2 - Hidden Markov Models

Arvid Edenheim - arved490

2019-08-12

1 - Model

The transition probability $p(x_{t+1}|x_t) = 0.5$ & $p(x_t|x_t) = 0.5$.

The emission/observation probability $p(y_{t-2}|x_t) = p(y_{t-1}|x_t) = p(y_t|x_t) = p(y_{t+1}|x_t) = p(y_{t+2}|x_t) = 0.2$.

This translates into the following matrices:

Transition matrix

	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

Emission matrix

	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 - Simulation

Simulation of 100 timesteps was trivial using the HMM library.

```
############### 2 ###############

nSteps <- 100
sims <- simHMM(HMM, nSteps)
```

3

```r
############### 3 ###############

# Exponential due to forward/backward returning log values
alpha <- exp(forward(HMM, sims$observation))
beta <- exp(backward(HMM, sims$observation))

# Normalizing at each time step to prevent underflow (?)
# and to, you know, normalize to obtain a prob. distr.
# Basically performs col[,i] / sum(col[,i]) to get the normalized distr.
filter <- prop.table(alpha, margin = 2)
smoother <- prop.table(alpha*beta, margin = 2)

# Evaluate most probable path using viterbi.
viterbiPath = viterbi(HMM, sims$observation)

# Generate path by maximizing the marginal probability for each individual timestep
filterPath <- apply(filter, MARGIN = 2, which.max)
smootherPath <- apply(smoother, MARGIN = 2, which.max)
```

4

```r
############### 4 ###############

accuracy <- function(path, states) {
  return(sum(diag(table(path, states)) / length(path)))
}

# 0.49
viterbiAccuracy = accuracy(viterbiPath, sims$states)

# 0.63
filterAccuracy =  accuracy(filterPath, sims$states)

# 0.75
smootherAccuracy =  accuracy(smootherPath, sims$states)
```

5

Repeated trials yields the same result as in 4). The smoothed distribution appears to have a higher accuracy than both the most probable path and the filtered distribution. This is natural since the smoother is the joint probability of both future and past observations, thus containing more information than the filtered distribution. The viterbi algorithm only calculates the most probable sequence, which intuitively always will be less accurate than the path obtained by maximizing the marginal probability for each individual timestep, regardless of neighboring states.

```
############### 5 ###############

sims <- simHMM(HMM, nSteps)

alpha <- exp(forward(HMM, sims$observation))
beta <- exp(backward(HMM, sims$observation))

filter <- prop.table(alpha, margin = 2)
smoother <- prop.table(alpha*beta, margin = 2)

viterbiPath = viterbi(HMM, sims$observation)
filterPath <- apply(filter, MARGIN = 2, which.max)
smootherPath <- apply(smoother, MARGIN = 2, which.max)

# 0.55
viterbiAccuracy = accuracy(viterbiPath, sims$states)

# 0.58
filterAccuracy =  accuracy(filterPath, sims$states)

# 0.73
smootherAccuracy =  accuracy(smootherPath, sims$states)

# Smoothing appears to perform better at evaluating the hidden states
```

6

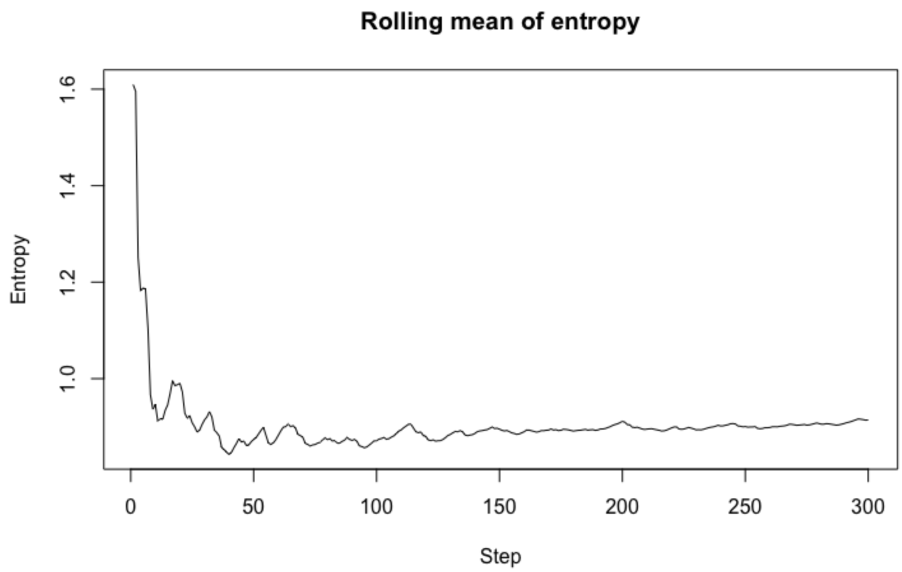

Figure 1: Entropy

I calculated the entropy of the filtered distributions over 300 timesteps and evaluated a rolling mean from timestep 1 to $i \in [1 : 300]$. After repeated trials, the mean appeared to converge after 70-100 iterations, which would imply that further observations don't necessarily provide additional information.

```
################ 6 ################

# With 300 time steps
nSteps <- 300
sims300 <- simHMM(HMM, nSteps)
alpha300 <- exp(forward(HMM, sims300$observation))
filter300 <- prop.table(alpha300, margin = 2)

entropy300 <- apply(filter300, MARGIN = 2, entropy.empirical)

# Evaluate rolling mean from step 1 to current step i
rollingMeanEntropy <- rep(0, nSteps)
for(i in 1:nSteps) {
  rollingMeanEntropy[i] <- mean(entropy300[1:i])
}

plot(rollingMeanEntropy,
     type='l',
     main="Rolling mean of entropy",
     ylab = 'Entropy',
     xlab='Step')

# Comment: Converges after approx 70-100 steps.
# More observations does not appear to provide further information
```

7

The state vector for the filtered distribution at timestep 100

$X_{100} =$

$$\begin{bmatrix} 0.654 & 0.345 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{bmatrix}$$

Which, when multiplied with the transition matrix from 1) yields the probability distribution for the hidden state at timestep 101,

$X_{101} =$

$$\begin{bmatrix} 0.327 & 0.500 & 0.173 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{bmatrix}$$

```
############### 7 ###############

hiddenStates101 <- colSums(filter[,100] * transProbs)

# P(X_101 = 1) = 0.33, P(X_101 = 2) = 0.50, P(X_101 = 3) = 0.17
# given filter[,100] = [0.65, 0.35, 0, ...]
```