

# 732A96: Labs

## Advanced Machine Learning

Carles Sans Fuentes

October 16, 2017

---

```
1
2 trans_probs <- diag(1/2, 10) +
3   diag(1/2, 10)[, c(10, 1:9)]
4 emission_probs <-
5   diag(1/5, 10)[, c(3:10, 1:2)] +
6   diag(1/5, 10)[, c(2:10, 1)] +
7   diag(1/5, 10) +
8   diag(1/5, 10)[, c(10, 1:9)] +
9   diag(1/5, 10)[, c(9:10, 1:8)]
10
11 emission_density <- function(x, z) {
12   return(emission_probs[z, x])
13 }
14
15 transition_density <- function(z, previous_z) {
16   return(trans_probs[previous_z, z])
17 }
18
19 transition_density2 <- function(z,previous_z){
20   if( z == zt){
21
22     return(0.5)
23
24   } else if( (z + 1) == zt){
25     return(0.5)
26   } else return(0)
27 }
28 }
29
30 get_alpha_scalar <- function(zt, xt, previous_alpha, previous_z) {
31   # Args:
32   #   zt Scalar, hidden state at which to compute alpha.
33   #   xt Scalar, observed state.
34   #   previous_alpha Vector, alpha for all z_{t-1}.
35   #   previous_z      Vector, all z_{t-1}.
36
37   summation_term <- 0
38   for (i in 1:length(previous_z)) {
39     summation_term <- summation_term +
40       previous_alpha[i] * transition_density(zt, previous_z[i])
41   }
42
43   alpha <- emission_density(xt, zt) * sum(summation_term)
44   return(alpha)
45 }
46
47 get_alpha <- function(Zt, xt, previous_alpha, previous_z) {
48   # Args:
49   #   Zt Vector, hidden states at which to compute alpha.
50   #   xt Scalar, observed state.
51   #   previous_alpha Vector, alpha for all z_{t-1}.
52   #   previous_z      Vector, all z_{t-1}.
53
54   alpha <- sapply(Zt, function(zt) {
55     get_alpha_scalar(zt, xt, previous_alpha, previous_z)
56   })
57
58   return(alpha)
59 }
60
61 get_beta_scalar <- function(zt, next_x, next_beta, next_z) {
62   # Args:
63   #   zt      Scalar, hidden state at which to compute alpha.
64   #   next_x   Scalar, observed next state.
65   #   next_beta Vector, alpha for all z_{t+1}.
66   #   next_z   Vector, all z_{t+1}.
```

```

67
68 summation_term <- 0
69 for (i in 1:length(next_z)) {
70   summation_term <- summation_term +
71     next_beta[i] * emission_density(next_x, next_z[i]) * transition_density(next_z[i], zt)
72 }
73
74 #  $P(z_{(t+1)} | z_t) =$ 
75 # 0.5 if  $z_t = z_{(t+1)}$ 
76 # 0.5 if  $z_t = z_{(t+1)} + 1$ 
77 # 0 otherwise
78
79
80 return(summation_term)
81 }
82
83 get_beta <- function(Zt, next_x, next_beta, next_z) {
84   # Args:
85   #   Zt      Vector, hidden states at which to compute alpha.
86   #   next_x  Scalar, observed next state.
87   #   next_beta Vector, alpha for all  $z_{\{t+1\}}$ .
88   #   next_z  Vector, all  $z_{\{t+1\}}$ .
89
90   beta <- sapply(Zt, function(zt) {
91     get_beta_scalar(zt, next_x, next_beta, next_z)
92   })
93
94   return(beta)
95 }
96
97 fb_algorithm <- function(
98   observations,
99   emission_density,
100   transition_density,
101   possible_states,
102   initial_density) {
103
104   t_total <- length(observations)
105   cardinality <- length(possible_states)
106
107   # Alpha
108   alpha <- matrix(NA, ncol=cardinality, nrow=t_total)
109
110   for (i in 1:cardinality) {
111     alpha[1, i] <-
112       emission_density(observations[1], possible_states[i]) * initial_density[i]
113   }
114
115   for (t in 2:t_total) {
116     alpha[t, ] <- get_alpha(possible_states, observations[t], alpha[t - 1, ], possible_states)
117   }
118
119   # Beta
120   beta <- matrix(NA, ncol=cardinality, nrow=t_total)
121
122   beta[t_total, ] <- 1
123
124   for (t in (t_total - 1):1) {
125     beta[t, ] <- get_beta(possible_states, observations[t + 1], beta[t + 1, ], possible_states)
126   }
127
128   return(list(alpha = alpha, beta = beta))
129 }
130
131 filtering <- function(alpha) {
132   alpha / rowSums(alpha)
133 }
134
135 smoothing <- function(alpha, beta) {
136   alpha * beta / rowSums(alpha * beta)
137 }
138
139
140
141
142
143
144
145
146 robotHMM <- HMM::initHMM(
147   States = 1:10,
148   Symbols = 1:10,
149   transProbs = trans_probs,
150   emissionProbs = emission_probs
151 )
152
153 # Create a wrapper for simHMM to assign class to the output

```

```

154 simHMM <- function(hmm, length) {
155   simulation <- HMM::simHMM(hmm, length)
156   return(structure(simulation, class="HMMSimulation"))
157 }
158
159 # Simulate
160 nSim <- 100
161 robotSimulation <- simHMM(hmm=robotHMM, length=nSim)
162
163 #debugonce(fb_algorithm)
164
165 alphabeta <- fb_algorithm(observations = robotSimulation$observation,
166                           emission_density = emission_density,
167                           transition_density = transition_density,
168                           possible_states = 1:10,
169                           initial_density = rep(0.1, 10))
170
171 filtering(alphabeta$alpha)
172 smoothing(alphabeta$alpha, alphabeta$beta)
173
174
175 plot(apply(filtering(alphabeta$alpha), 1, which.max), type = "l")
176 plot(apply(smoothing(alphabeta$alpha, alphabeta$beta), 1, which.max), type = "l")
177 lines(x = 1:100, robotSimulation$states, type = "l", col = "green")
178
179
180
181
182 # # # Test
183 # zt <- 5
184 # xt <- 6
185 # previous_alpha <- rep(0.1, 10)
186 # previous_z <- 1:10
187 # transition_density(zt, previous_z[5])
188 # get_alpha_scalar(zt, xt, previous_alpha, previous_z)
189
190
191 # # Test
192 # zt <- 1:10
193 # xt <- 6
194 # previous_alpha <- rep(0.1, 10)
195 # previous_z <- 1:10
196 # transition_density(zt, previous_z[5])
197 # get_alpha(zt, xt, previous_alpha, previous_z)
198 #
199
200
201 # # Test
202 # Zt <- 1:10
203 # next_x <- 6
204 # next_beta <- rep(0.1, 10)
205 # next_z <- 1:10
206 # transition_density(zt, previous_z[5])
207 # get_beta_scalar(5, next_x, next_beta, next_z)
208 # get_beta(zt, next_x, next_beta, next_z)
209
210
211 # Define the transition, emission and initialization probabilities -----
212
213 emission_probs <- matrix(c(.2, .2, .2, 0, 0, 0, 0, 0, .2, .2,
214                             .2, .2, .2, .2, 0, 0, 0, 0, 0, .2,
215                             .2, .2, .2, .2, .2, 0, 0, 0, 0, 0,
216                             0, .2, .2, .2, .2, .2, 0, 0, 0, 0,
217                             0, 0, .2, .2, .2, .2, .2, 0, 0, 0,
218                             0, 0, 0, .2, .2, .2, .2, .2, 0, 0,
219                             0, 0, 0, 0, .2, .2, .2, .2, .2, 0,
220                             0, 0, 0, 0, 0, .2, .2, .2, .2, .2,
221                             .2, 0, 0, 0, 0, 0, .2, .2, .2, .2,
222                             .2, .2, 0, 0, 0, 0, 0, .2, .2, .2), byrow=TRUE, nrow=10)
223
224 transition_probs <- matrix(c(.5, .5, 0, 0, 0, 0, 0, 0, 0, 0,
225                              0, .5, .5, 0, 0, 0, 0, 0, 0, 0,
226                              0, 0, .5, .5, 0, 0, 0, 0, 0, 0,
227                              0, 0, 0, .5, .5, 0, 0, 0, 0, 0,
228                              0, 0, 0, 0, .5, .5, 0, 0, 0, 0,
229                              0, 0, 0, 0, 0, .5, .5, 0, 0, 0,
230                              0, 0, 0, 0, 0, 0, .5, .5, 0, 0,
231                              0, 0, 0, 0, 0, 0, 0, .5, .5, 0,
232                              0, 0, 0, 0, 0, 0, 0, 0, .5, .5,
233                              .5, 0, 0, 0, 0, 0, 0, 0, 0, .5), byrow=TRUE, nrow=10)
234
235 tProbDensity <- function(zt, zt_1) {
236   return(transition_probs[zt_1, zt])
237 }
238
239 eProbDensity <- function(xt, zt) {
240   return(emission_probs[zt, xt])

```

```

241 }
242
243 initProbDensity <- function(z0) {
244   return(dunif(z0, min=1, max=10))
245 }
246
247
248 # Simulate data -----
249
250 library(HMM)
251
252 robotHmm <- HMM::initHMM(
253   States = 1:10,
254   Symbols = 1:10,
255   transProbs = transition_probs,
256   emissionProbs = emission_probs
257 )
258
259 simHMM <- function(hmm, length) {
260   simulation <- HMM::simHMM(hmm, length)
261   return(structure(simulation, class="HmmSimulation"))
262 }
263
264 nSim <- 100
265 robotSimulation <- simHMM(hmm=robotHmm, length=nSim)
266
267 X <- robotSimulation$observation
268 Z <- robotSimulation$states
269
270 # Implement Viterbi -----
271
272 possibleStates <- 1:10
273 get_omega <- function(Z, Omega, Z_next, x_next) {
274   sapply(Z_next, function(z_next) {
275     term1 <- log(eProbDensity(x_next, z_next))
276
277     term2 <- sapply(Z, function(z) {
278       log(tProbDensity(z_next, z))
279     }) + Omega
280
281     return(term1+ max(term2))
282   })
283 }
284
285 get_phi <- function(Z, Z_next, Omega) {
286   sapply(Z_next, function(z_next) {
287     term <- sapply(Z, function(z) {
288       log(tProbDensity(z_next, z))
289     }) + Omega
290     return(Z[which.max(term)])
291   })
292 }
293
294 viterbi <- function(observations, possibleStates) {
295   cardinality <- length(possibleStates)
296   t_total <- length(observations)
297
298   omega_0 <- vector("numeric", length = cardinality)
299   for (i in 1:cardinality) {
300     omega_0[i] <- log(initProbDensity(possibleStates[i])) + log(eProbDensity(observations[i],
301       possibleStates[i]))
302   }
303
304   omega <- matrix(NA, nrow=t_total, ncol=cardinality)
305   phi <- matrix(NA, nrow=t_total, ncol=cardinality)
306   omega[1, ] <- omega_0
307
308   for (i in 1:(t_total-1)) {
309     omega[i+1, ] <- get_omega(possibleStates, omega[i, ], possibleStates, observations[i+1])
310     phi[i+1, ] <- get_phi(possibleStates, possibleStates, omega[i, ])
311   }
312
313   mpp <- rep(NA, t_total)
314   mpp[t_total] <- possibleStates[which.max(omega[t_total, ])]
315   for (t in (t_total - 1):1) {
316     mpp[t] <- phi[t + 1, possibleStates[mpp[t + 1]] == possibleStates]
317   }
318
319   return(list(path = mpp, omega = omega, phi = phi))
320 }
321
322
323 results <- viterbi(X, possibleStates)
324 results$path
325
326 results_HMM <- HMM::viterbi(robotHmm, X)

```

```
327  
328 cbind(results$path, results_HMM)
```