# Advanced Machine Learning: Lab 22 - Individual Report

Shashi Nagarajan (shana299)

```
library(HMM)
library(entropy)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

**1**

We can model the observed reading from the sensor at time $t$ as $X^t$, and the latent position of the robot at time $t$ as $Z^t$. Per the problem description:

- Sample space of both $X^t$ and $Z^t$ is $\{0, 1, 2, ..., 9\}$
- $X^t|Z^t = i \sim DiscreteUnif([(i-2)\ mod\ 10, (i+2)\ mod\ 10])$
- $Z^{t+1}|Z^t = i \sim DiscreteUnif(i, (i+1)\ mod\ 10)$

This yields the following graphical model:

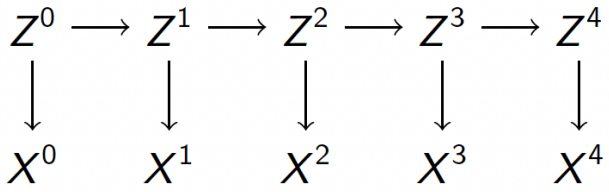


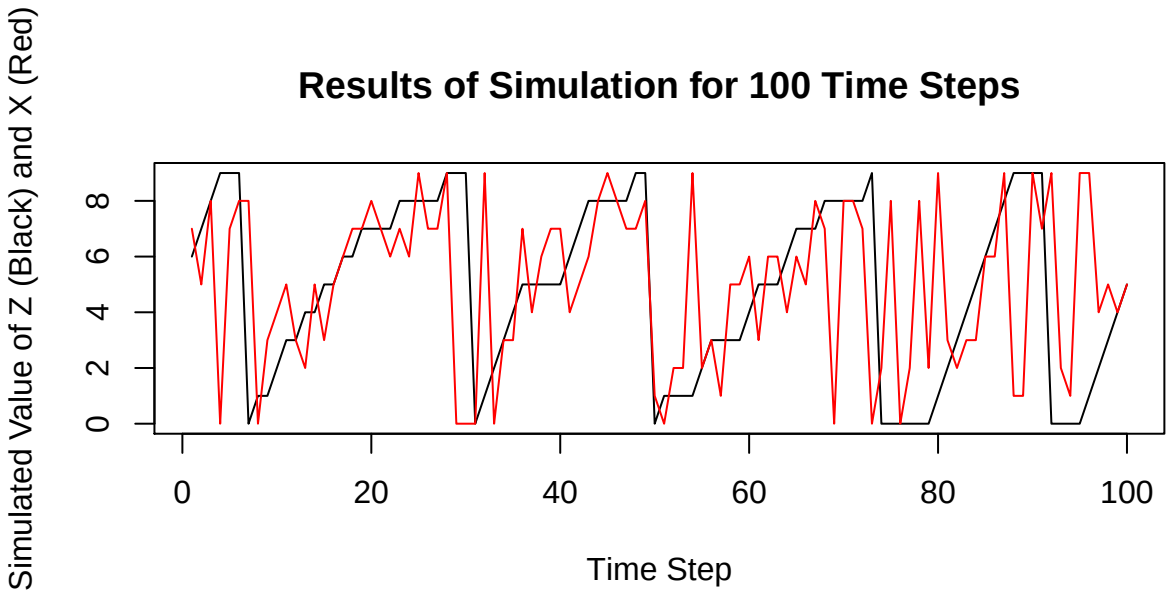

Figure 1: Graphical Model

**2**

Requested simulation was performed; results plotted below.

```
Z <- vector("numeric", 100)
X <- vector("numeric", 100)

# seed
Z[1] = sample(1:10, 1)

# start simulation

X[1] = sample(c((Z[1]-2) %% 10, (Z[1]-1) %% 10, Z[1], (Z[1]+1) %% 10, (Z[1]+2) %% 10), 1)

for (i in 2:100) {

  Z[i] = sample(c(Z[i - 1], (Z[i - 1] + 1) %% 10), 1)
  X[i] = sample(c((Z[i]-2) %% 10, (Z[i]-1) %% 10,
```

```
                Z[i], (Z[i]+1) %% 10, (Z[i]+2) %% 10), 1)

}

# plot results
plot(1:100, Z, type = 'l', xlab = "Time Step",
     ylab = "Simulated Value of Z (Black) and X (Red)",
     main = "Results of Simulation for 100 Time Steps",
     ylim = c(0, 9))
lines(1:100, X, col = 'red')
```




**3**

As asked, the simulated observations (of sensor readings) were used to compute the filtered and smoothed probability distributions for each of the 100 time points. The most probable path was also computed. Workings below

```
robot_hmm <- initHMM(States = as.character(0:9), Symbols = as.character(0:9),
               startProbs = rep(0.1, 10),
               transProbs = diag(0.5, 10) + diag(0.5, 10)[, c(10, 1:9)],
               emissionProbs = t(diag(0.2, 10)[, c(9:10, 1:8)]) +
                 t(diag(0.2, 10)[, c(10, 1:9)]) + diag(0.2, 10) +
                 diag(0.2, 10)[, c(10, 1:9)] + diag(0.2, 10)[, c(9:10, 1:8)])

forward_probs <- exp(forward(robot_hmm, as.character(X)))
backward_probs <- exp(backward(robot_hmm, as.character(X)))
product_probs <- forward_probs*backward_probs

filtered <- sapply(1:100, function(i)
  forward_probs[, i]/sum(forward_probs[, i]))
smoothed <- sapply(1:100, function(i)
  product_probs[, i]/sum(product_probs[, i]))

most_probable_path <- viterbi(robot_hmm, as.character(X))
```

**4**

The requested accuracy measures have been computed as shown below.

```
filtered_most_probable_path <- apply(filtered, 2, which.max) - 1
smoothed_most_probable_path <- apply(smoothed, 2, which.max) - 1

print(paste("Accuracy of the most probable states",
            "using the filtered distribution was",
            sum(filtered_most_probable_path == Z), "%"))
```

```
## [1] "Accuracy of the most probable states using the filtered distribution was 51 %"
```

```
print(paste("Accuracy of the most probable states",
            "using the smoothed distribution was",
            sum(smoothed_most_probable_path == Z), "%"))
```

```
## [1] "Accuracy of the most probable states using the smoothed distribution was 72 %"
```

```
print(paste("Accuracy of the most probable states",
            "as obtained through the Viterbi algorithm was",
            sum(most_probable_path == Z), "%"))
```

```
## [1] "Accuracy of the most probable states as obtained through the Viterbi algorithm was 51 %"
```

**5**

In 100 simulations, the most probable paths estimated using the smoothed distributions ('smoothed paths') yielded better results than both those estimated using filtered distributions ('filtered paths') and the Viterbi algorithm ('Viterbi paths') most often.

**Filtered vs. Smoothed Paths** From the graphical model shown in Q(1), we can see the future observed states (sensor readings) can be meaningful in estimating hidden state (actual robot position) likelihoods. At any given time $t$, there are open paths from future states (i.e. $X^{t+1},\ i \in \{1, 2, 3, ..\}$) to $Z^t$.

Between Smoothed and Filtered distributions, only Smoothed uses both historical and future observations (sensor readings) to work out the maximum likelihood path. The former therefore can be expected to perform better than the latter.

**Viterbi vs. Smoothed Paths** Whereas Viterbi paths are based on estimates which maximise the joint likelihood of all the hidden states given the observed states, Smoothed paths are based on estimates that maximise the marginal likelihood of each hidden state given the observed states.

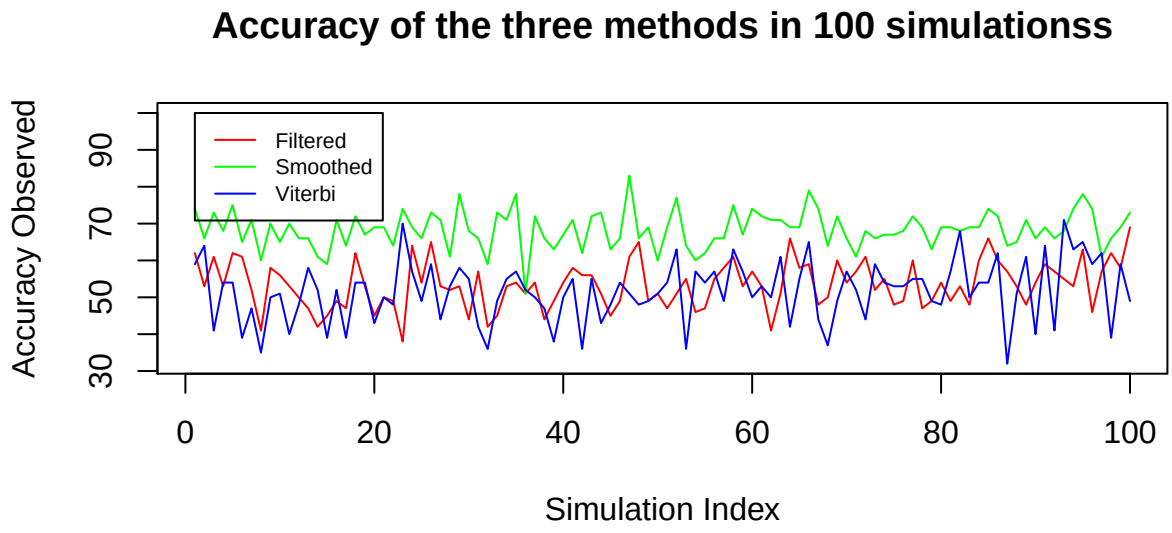
Given two paths, one can potentially evaluate how close each is to the other by observing how many sub-sequences of arbitrary size are present in the same location of either paths.

The performance metric of our choice, namely 'accuracy' viz. 'percentage of the true hidden states guessed' only looks at sub-sequences of length 1 (i.e. each hidden state in isolation).

In such a scenario, the Smoothed distribution is less constrained by the joint

```
filtered_acc <- vector("numeric", 100)
smoothed_acc <- vector("numeric", 100)
viterbi_acc <- vector("numeric", 100)

for (i in 1:100) {

  sim <- simHMM(robot_hmm, 100)
  forward_probs <- exp(forward(robot_hmm, sim$observation))
```

```
  backward_probs <- exp(backward(robot_hmm, sim$observation))
  product_probs <- forward_probs*backward_probs

  filtered <- sapply(1:100, function(i)
    forward_probs[, i]/sum(forward_probs[, i]))
  smoothed <- sapply(1:100, function(i)
    product_probs[, i]/sum(product_probs[, i]))

  filtered_most_probable_path <- apply(filtered, 2, which.max) - 1
  smoothed_most_probable_path <- apply(smoothed, 2, which.max) - 1
  most_probable_path <- viterbi(robot_hmm, sim$observation)

  filtered_acc[i] <- sum(filtered_most_probable_path == sim$states)
  smoothed_acc[i] <- sum(smoothed_most_probable_path == sim$states)
  viterbi_acc[i] <- sum(most_probable_path == sim$states)

}

plot(1:100, filtered_acc, type = 'l', col = 'red',
     ylim = c(min(filtered_acc, smoothed_acc, viterbi_acc), 100),
     xlab = "Simulation Index", ylab = "Accuracy Observed",
     main = "Accuracy of the three methods in 100 simulationss")
lines(1:100, smoothed_acc, col = 'green')
lines(1:100, viterbi_acc, col = 'blue')
legend(1, 100, col = c('red', 'green', 'blue'),
       legend = c('Filtered', 'Smoothed', 'Viterbi'), lty = 1, cex = 0.7)
```




**6**

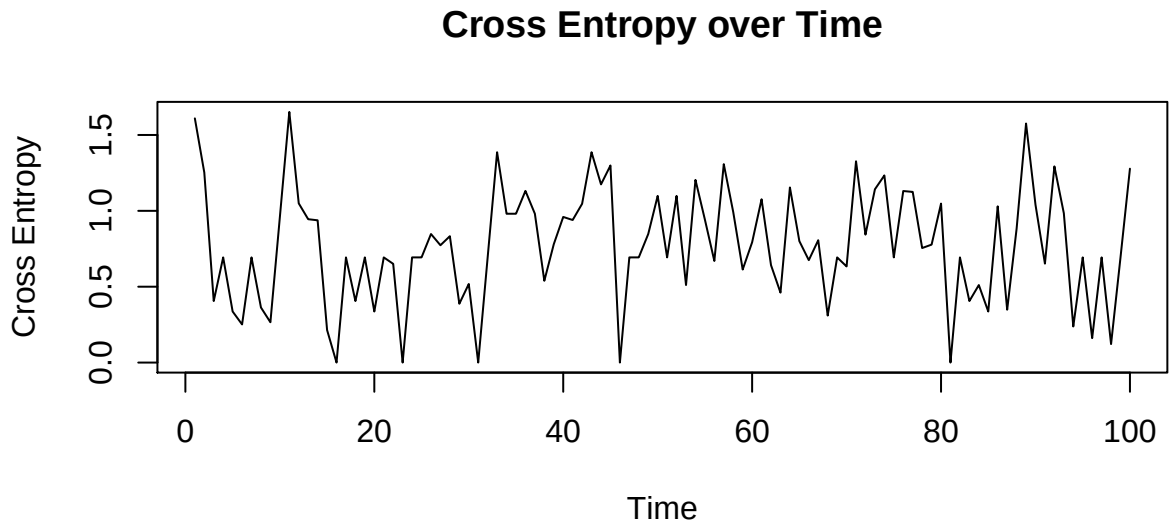
For this question, we start by using results from the last simulation in Q(5). Also since the question refers to where the robot *is*, we concern ourselves with the filtered distribution alone.

We see no evidence suggesting that as time goes, the accuracy of the filtered distribution improves.

Since the assertion has already been disproved here, we don't perform more simulations.

```
# prepare one-hot encoding of simulated latent states
states_df <- data.frame(id = 1:100, var = sim$states)
```

```r
states_one_hot <- predict(dummyVars(" ~ .", data=states_df),
                          newdata =states_df)[, -1]

# (log) filtered distributions
log_filtered <- t(log(filtered))

# cross-entropy
plot(1:100, -rowSums(states_one_hot*log_filtered, na.rm = T),
     ylab = "Cross Entropy", xlab = "Time",
     main = "Cross Entropy over Time", type = 'l')
```




**7**

For this question, we once again consider the results from the last simulation in Q(5).

$$p(z^{101}|x^{1:100}) = \sum_{z_{100}} p(z^{101}, z^{100}|x^{1:100}) = \sum_{z_{100}} p(z^{101}|z^{100}, x^{1:100}) p(z^{100}|x^{1:100})$$

From the graph, we can observe that $Z^{100}$ separates $Z^{101}$ from $X^{1:100}$. Thus, $p(z^{101}|z^{100}, x^{1:100}) = p(z^{101}|z^{100})$, which is given by the transition probability matrix.

Also, $p(z^{100}|x^{1:100})$ is the filtered distribution of $Z^{100}$.

Thus the conditional distribution of $Z^{101}|X^{1:100}$ is given by the product of the filtered distribution at time 100 and the transition probability matrix.

```r
print(filtered[, 100] %*% robot_hmm$transProbs)
```

```
##       to
##                 0 1 2 3 4 5         6         7     8         9
##   [1,] 0.1106719 0 0 0 0 0 0.01432806 0.1536561 0.375 0.3463439
```