Project 3. Viterbi algorithm

March 10, 2024

Contents

Problem 6

Consider the HMM represented by the above graph, where X_n are observed variables and Z_n are hidden variables. Each hidden variable can take on K different values and the observed variables can take on M different realisations. The HMM is now parameterized as in the lecture on slide 18.

a) What is the maximum number of free parameters to define the HMM?

Parameters of HMM:

- Initial state probabilities: $I_k = P(Z_1 = k)$ -> vector of length k, probabilities sum up to 1 so k-1 free parameters
- Transition probabilities: $T_{kl} = P(Z_n = l|Z_{n-1} = k)$ from state k to l. Generally, it would be a matrix of size $k \times k$ (from any state k to any other state), with $k \cdot (k-1)$ free parameters
- Emission probabilities: $E_{kx} = P(X_n = x|Z_n = k)$. With k states and m random variables x, thus a matrix of size $k \times m$, with $k \cdot (m-1)$ free parameters

The overall sum of parameters is: (k-1) + (k * (k-1)) + (k * (m-1))

b) Assume K = 2 and the transition matrix is

$$T = \begin{pmatrix} 0.3 & 0.7 \\ 0.2 & 0.8 \end{pmatrix} \tag{1}$$

What is the stationary distribution π?

The definition of stationary distribution satisfies the following:

$$\pi = \pi T$$

Considering K is equal to 2, that means that the stationary distribution can be represented as following vector with k-1=1 parameters:

$$\pi = \begin{bmatrix} p & (1-p) \end{bmatrix} \tag{2}$$

This leads to the following equations:

$$0.3p + 0.2(1-p) = p$$
$$0.7p + 0.8(1-p) = 1-p$$

Both this equations lead to the solutions of $p = \frac{2}{9}$. Thus the stationary distribution is:

$$\pi = \begin{bmatrix} \frac{2}{9} & , \frac{7}{9} \end{bmatrix} \tag{3}$$

Problem 7

Setting up the environment

```
suppressPackageStartupMessages(library(dplyr))
set.seed(42)
```

```
DSSP <- c("B", "C", "E", "G", "H", "I", "S", "T")
AA <- c("A", "C", "D", "E", "F", "G", "H", "I", "K",
    "L", "M", "N", "P", "Q", "R", "S", "T", "U",
    "V", "W", "X", "Y"
)
data_folder <- "../data/"
```

Functions

Viterbi algorithm was provided.

```
#' @param E n times m matrix with the emission log probabilities of the n latent variables and m observ
#' @param Tr n times n matrix with the transition log probabilities
#' @param I numeric vector of length n with the initial log probabilities of the n latent variables
#' @param p a data.frame with a Variable named AminoAcids which holds the amino acid sequence as a char
viterbi <- function(E, Tr, I, p) {
    .as.array <- function(.) stringr::str_split(., "")[[1]]
    unique.ss <- c("B", "C", "E", "G", "H", "I", "S", "T")
    unique.aa <- c("A", "C", "D", "E", "F", "G", "H", "I",
                   "K", "L", "M", "N", "P", "Q", "R", "S",
                   "T", "U", "V", "W", "X", "Y")
    for (k in seq(nrow(p))) {
        sequence <- p$AminoAcids[k]
        aa.vec <- .as.array(sequence) %>% match(unique.aa)
        P      <- matrix(0, nrow(E), length(aa.vec))
        Ptr    <- matrix(0, nrow(E), length(aa.vec))

        ## sets the paths
        for (i in seq(length(aa.vec))) {
```

```
            if (i == 1) {
                P[, i] <- I + E[, aa.vec[i]]
            } else {
                for (j in seq(nrow(E))) {
                    p.loc    <- P[, i - 1] + Tr[, j] + E[j, aa.vec[i]]
                    P[j, i] <- max(p.loc)
                    Ptr[j, i] <- which.max(p.loc)
                }
            }
        }

        ## backtrace: computes the most likely path
        Phi <- vector(mode="integer",   length=length(aa.vec))
        Phi[length(Phi)] <- which.max(P[, ncol(P)])
        ## we start at the back, just as with Needleman-Wunsch or Smith-Waterman
        for (i in seq(from=length(aa.vec), to=2)) {
            Phi[i - 1] <- Ptr[Phi[i], i]
        }

        states <- unique.ss[Phi]
        p$PredictedStructure[k] <- paste(states, collapse="")
    }
    return(p)
}

loading_data <- function(data_file, sec_struct) {
    col_names <- c('SeqID', 'AminoAcids')
    if(sec_struct) { col_names <- append(col_names, 'SecondaryStructure')}
    data_df <- read.csv(data_file, header=FALSE, sep="\t")
    names(data_df) <- col_names
    return(data_df)
}

assign_row_names <- function(M) {
    row.names(M) <- DSSP
    return(M)
}

assign_col_names <- function(M, aa=TRUE) {
    if (aa) {
        colnames(M) <- AA
    } else {
        colnames(M) <- DSSP
    }

    return(M)
}

init_matrices <- function(data) {
    # Initialisation of I, T, and E matrices using maximum likelihood

    # I - 8 x 1 matrix - initial state probabilities
    I <- matrix(0, nrow=length(DSSP), ncol=1)
```

```
    # T - 8 x 8 matrix - transition probabilities
    T <- matrix(0, nrow=length(DSSP), ncol=length(DSSP))
    # E - 8 x 22 matrix - emission probabilities
    E <- matrix(0, nrow=length(DSSP), ncol=length(AA))

    I <- assign_row_names(I)
    T <- assign_row_names(T)
    E <- assign_row_names(E)

    T <- assign_col_names(T, aa=FALSE)
    E <- assign_col_names(E)

    I <- initial_states(data, I)
    T <- transition_states(data, T)
    E <- emission_states(data, E)

    I <- log(I)
    T <- log(T)
    E <- log(E)

    return(list(I=I, T=T, E=E))
}

initial_states <- function(data, M) {
    # Retrieving DSSP profiles
    sec_struct <- data$SecondaryStructure

    # Computing frequencies of each letter in the first position
    frequency <- table(substr(sec_struct, 1, 1))

    # Computing relative frequencies
    M[row.names(M) %in% names(frequency), 1] <- as.numeric(frequency)/sum(frequency)

    # Return initial states matrix
    return(M)
}

transition_states <- function(data, M) {
    # Retrieving DSSP profiles
    sec_struct <- data$SecondaryStructure

    pairs <- lapply(sec_struct,
        function(x) substring(x, first=1:(nchar(x)-1), last=2:nchar(x))
    )
    frequencies <- table(unlist(pairs))

    # Assigning frequencies
    for (i in seq(nrow(M))) {
        for (j in seq(ncol(M))) {
            row_name <- row.names(M)[i]
            col_name <- colnames(M)[j]
            freq_name <- paste0(row_name, col_name)
            if (is.na(frequencies[freq_name])) {
```

```
                M[i, j] <- 1
            } else {
                M[i, j] <- as.numeric(frequencies[freq_name])+1
            }
        }
    }
    # Relative frequencies
    M <- M/rowSums(M)

    # Return transition states matrix
    return(M)
}

emission_states <- function(data, M) {
    # Retrieving profiles
    aa <- data$AminoAcids
    dssp <- data$SecondaryStructure

    aa_list <- lapply(aa,
        function(x) substring(x, first=1:(nchar(x)), last=1:nchar(x))
    )
    dssp_list <- lapply(dssp,
        function(x) substring(x, first=1:(nchar(x)), last=1:nchar(x))
    )

    frequencies <- table(paste(unlist(dssp_list), unlist(aa_list), sep=""))

    # Assigning frequencies
    for (i in seq(nrow(M))) {
        for (j in seq(ncol(M))) {
            row_name <- row.names(M)[i]
            col_name <- colnames(M)[j]
            freq_name <- paste0(row_name, col_name)
            if (is.na(frequencies[freq_name])) {
                M[i, j] <- 1
            } else {
                M[i, j] <- as.numeric(frequencies[freq_name])+1
            }
        }
    }
    # Relative frequencies
    M <- M/rowSums(M)

    # Return initial states matrix
    return(M)
}

eigen_stationary_distribution <- function(T) {
    # T transposed to get left eigenvectors of T
    e <- eigen(t(exp(T)))
    e_vec <- e$vectors
    e_val <- e$values
    e_val_one_index <- which(abs(e_val-1) < 1e-12)
```

```
    stat_dist <- matrix(e_vec[, e_val_one_index]/
        sum(e_vec[, e_val_one_index]), nrow=1, ncol=length(DSSP)
    )
    colnames(stat_dist) <- DSSP
    print(paste("Eigenvalue:", e_val[e_val_one_index]))
    print("Stationary distribution of transmission probabilities (eigenvalue approach): ")
    return(stat_dist)
}

brute_force_stationary_distribution <- function(T) {
    T <- exp(T)
    while (sum(abs(T[1,]-T[2,])) > 1e-12) {
        T <- T %*% T
    }
    print("Stationary distribution of transmission probabilities (brute-force approach): ")
    print(T[1,])
}

run_viterbi_predictions <- function(data, params) {
    pred_df <- data.frame(AminoAcids=data$AminoAcids, PredictedStructure=NA)
    pred_df <- apply(pred_df, 1, function(row) {
        viterbi(params$E, params$T, params$I, data.frame(AminoAcids=row["AminoAcids"]))
    })
    pred_df <- data.frame(t(sapply(pred_df, function(x) x[1:max(lengths(pred_df))])))
    data$PredictedStructure <- pred_df$PredictedStructure
    data <- apply(data, 2, as.character)
    return(data)
}

run_random_predictions <- function(data, params) {
    rand_struct <- apply(data, 1, function(row) {
        paste0(sample(DSSP, nchar(row["AminoAcids"]), replace=TRUE), collapse="")
    })
    data <- cbind(data, RandomStructure=rand_struct)
    return(data)
}

save_to_tsv <- function(data, file_path) {
    write.table(data, file=file_path, sep="\t", row.names=FALSE, col.names=FALSE)
}

get_accuracies <- function(pred_df, column_name) {
    accuracies <- apply(pred_df, 1, function(row) {
        sum(
            strsplit(row["SecondaryStructure"], "")[[1]] == strsplit(row[column_name], "")[[1]])/
            nchar(row["SecondaryStructure"])
    })
    print(summary(accuracies))
    return(accuracies)
}
```

(a) Data loading

Read `proteins_train.tsv`, `proteins_test.tsv` and `proteins_new.tsv` into the memory and store each in a data.frame.

```
# Loading data
prot_train_df <- loading_data(paste0(data_folder, "proteins_train.tsv"), sec_struct=TRUE)
prot_test_df <- loading_data(paste0(data_folder, "proteins_test.tsv"), sec_struct=TRUE)
prot_new_df <- loading_data(paste0(data_folder, "proteins_new.tsv"), sec_struct=FALSE)
```

(b) Estimating parameters

Estimate the vector of initial state probabilities I, the matrix of transition probabilities T and the matrix for emission probabilities E by using maximum likelihood, i.e., by counting how often each transition and each emission happened and how often each secondary structure was found at the beginning of a protein, respectively.

$$I_i = P(Z_1 = i) = \frac{\#latent\ sequences\ start\ with\ i}{\#sequences}$$

$$E_{ij} = P(X_n = j | Z_n = i) = \frac{\#latent\ state\ i\ emits\ observable\ j}{\#latent\ state\ i\ emits\ anything}$$

$$T_{ij} = P(Z_n = j | Z_{n-1} = i) = \frac{\#latent\ state\ i\ transits\ to\ state\ j}{\#latent\ state\ i\ transits\ to\ any\ other\ state}$$

```
# Initialisation of parameters
params <- init_matrices(prot_train_df)
```

(c) Stationary distribution

Eigenvalues

We need left eigenvector for the stationary distribution:

$$\pi = \pi T$$

```
eigen_stationary_distribution(params$T)
```

```
## [1] "Eigenvalue: 1"
## [1] "Stationary distribution of transmission probabilities (eigenvalue approach): "
```

```
##               B         C         E          G         H            I          S         T
## [1,] 0.01169951 0.2041994 0.2094351 0.03433775 0.3394477 0.0001497634 0.08894147 0.1117894
```

Brute force

```
brute_force_stationary_distribution(params$T)
```

```
## [1] "Stationary distribution of transmission probabilities (brute-force approach): "
##            B            C            E            G            H            I            S
## 0.0116995148 0.2041993650 0.2094350926 0.0343377464 0.3394476727 0.0001497634 0.0889414725 0.1117893
```

(d) Predictions

```
# Making predictions
prot_test_df <- run_viterbi_predictions(prot_test_df, params)
prot_new_df <- run_viterbi_predictions(prot_new_df, params)

# Saving predictions
save_to_tsv(prot_new_df, paste0(data_folder, "proteins_new_pred.tsv"))

# Computing prediction accuracies
print("Accuracy statistics of Viterbi predictions:")
```

```
## [1] "Accuracy statistics of Viterbi predictions:"
```

```
viterbi_acc <- get_accuracies(prot_test_df, column_name="PredictedStructure")
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.007752 0.226253 0.322917 0.319801 0.407240 0.857143
```

```
# Making random "predictions"
prot_test_df <- run_random_predictions(prot_test_df, params)
print("Accuracy statistics of random 'predictions':")
```

```
## [1] "Accuracy statistics of random 'predictions':"
```

```
rand_acc <- get_accuracies(prot_test_df, column_name="RandomStructure")
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1051  0.1213  0.1225  0.1398  0.2857
```

(e) Bootstrapping

```
# Number of bootstrap runs
n_bt <- 1000
# Define storage dataframes
I_bt <- matrix(0, length(DSSP), n_bt)
E_bt <- array(0, dim=c(length(DSSP), length(AA), n_bt))
T_bt <- array(0, dim=c(length(DSSP), length(DSSP), n_bt))

# Bootstrapping
for (i in 1:n_bt){
    # Resample the dataframe
    df_bt <- prot_train_df[sample(nrow(prot_train_df), replace=T),]

    est <- init_matrices(df_bt)

    # Estimate current run
    I_est <- exp(est$I)
    E_est <- exp(est$E)
    T_est <- exp(est$T)

    # Concatenate
    I_bt[, i] <- I_est
    E_bt[, , i] <- E_est
    T_bt[, , i] <- T_est
```

```
    if (sum(is.na(T_est)) > 0){
        break
    }
}
```

```
I_ci <- apply(I_bt, 1, function(x) {
  down <- quantile(x, 0.025)
  up <- quantile(x, 0.975)
  return(c(down, up))
})

E_ci <- apply(E_bt, c(1, 2), function(x) {
  down <- quantile(x, 0.025)
  up <- quantile(x, 0.975)
  return(c(down, up))
})

T_ci <- apply(T_bt, c(1, 2), function(x) {
  down <- quantile(x, 0.025)
  up <- quantile(x, 0.975)
  return(c(down, up))
})

I_ci
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## 2.5%     0    1    0    0    0    0    0    0
## 97.5%    0    1    0    0    0    0    0    0
```

```
head(E_ci)
```

```
## , , 1
##
##             [,1]       [,2]       [,3]       [,4]      [,5]       [,6]       [,7]       [,8]
## 2.5%  0.03649427 0.06158530 0.05521285 0.09921299 0.1215074 0.02941176 0.05086419 0.06004591
## 97.5% 0.05406883 0.06744052 0.06043168 0.11665884 0.1282808 0.24489796 0.05791220 0.06714667
##
## , , 2
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.02557546 0.01424455 0.02169162 0.01337151 0.01099009 0.01666667 0.01301568 0.01028668
## 97.5% 0.03976589 0.01751170 0.02538252 0.02080558 0.01405029 0.04545455 0.01789918 0.01323517
##
## , , 3
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.05480050 0.06945877 0.02614289 0.08932105 0.05096491 0.02222222 0.08120260 0.07307416
## 97.5% 0.07715708 0.07531605 0.02964338 0.10410661 0.05426017 0.22222222 0.09105812 0.08150564
##
## , , 4
##
##             [,1]      [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.01785694 0.0460772 0.03793854 0.07570073 0.08373449 0.01666667 0.04812489 0.05651190
## 97.5% 0.03229040 0.0521777 0.04194061 0.08985016 0.08887383 0.04545455 0.05617271 0.06362472
```

```
##
## , , 5
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]      [,7]       [,8]
## 2.5%  0.03369594 0.03217176 0.04970949 0.03013586 0.03488724 0.01666667 0.0208113 0.02122520
## 97.5% 0.05116176 0.03561375 0.05525977 0.03937356 0.03826327 0.04545455 0.0258616 0.02567428
##
## , , 6
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]      [,7]      [,8]
## 2.5%  0.04170958 0.07912638 0.04531599 0.05720275 0.03334553 0.01666667 0.1361762 0.1930842
## 97.5% 0.06177554 0.08578075 0.05090377 0.06917045 0.03661246 0.04545455 0.1477139 0.2050565
##
## , , 7
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.02525296 0.02384385 0.02558821 0.02617671 0.02002470 0.01666667 0.02054714 0.01895803
## 97.5% 0.04054089 0.02806185 0.02946365 0.03485827 0.02380629 0.04545455 0.02571667 0.02368500
##
## , , 8
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.05206276 0.04130862 0.08851788 0.02740374 0.05291752 0.02173913 0.02483991 0.02266594
## 97.5% 0.07238536 0.04523890 0.09779674 0.03659869 0.05675165 0.09375000 0.03040234 0.02687818
##
## , , 9
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.03668320 0.05753851 0.05543738 0.05325271 0.07028363 0.01666667 0.06801146 0.07426971
## 97.5% 0.05514953 0.06463510 0.06006484 0.06541824 0.07551643 0.04545455 0.07680915 0.08451294
##
## , , 10
##
##             [,1]       [,2]      [,3]       [,4]      [,5]       [,6]       [,7]       [,8]
## 2.5%  0.06290486 0.06704022 0.1018011 0.05228375 0.1096429 0.03703704 0.04746181 0.04673258
## 97.5% 0.08738356 0.07374590 0.1085023 0.06465623 0.1161957 0.19565217 0.05416124 0.05278027
##
## , , 11
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.01077877 0.02167431 0.02207374 0.01181385 0.02833617 0.02173913 0.01314625 0.01163744
## 97.5% 0.02338885 0.02465660 0.02535943 0.01844824 0.03111367 0.09375000 0.01700754 0.01457211
##
## , , 12
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.04430371 0.05390784 0.02215284 0.04695472 0.03477586 0.01666667 0.05069293 0.06380421
## 97.5% 0.06587051 0.06011531 0.02583954 0.05808596 0.03813470 0.04545455 0.05906697 0.07095243
##
## , , 13
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.03439350 0.07910986 0.01632859 0.05877748 0.01805750 0.01666667 0.05441141 0.06418135
## 97.5% 0.05133497 0.08664032 0.01925271 0.07027915 0.02049846 0.04545455 0.06074693 0.07194532
```

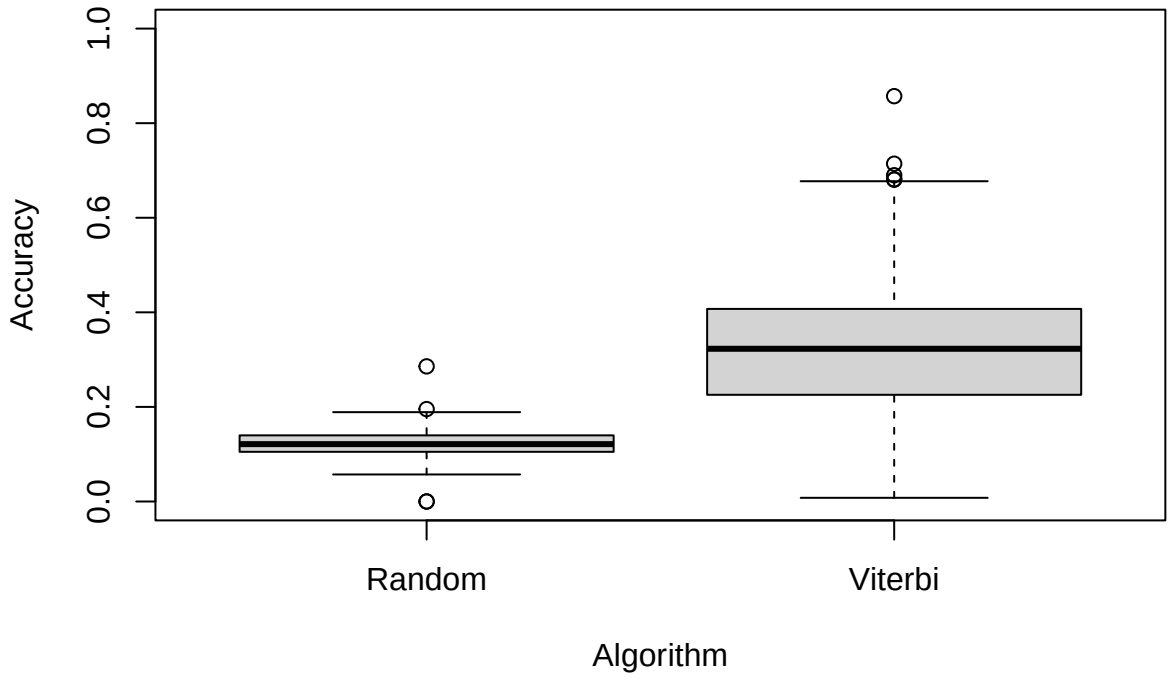
```
##
## , , 14
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.02263273 0.02949378 0.02913929 0.03125437 0.03964978 0.01666667 0.02819695 0.03435538
## 97.5% 0.03767021 0.03330648 0.03345541 0.04002958 0.04300839 0.04545455 0.03372068 0.03971283
##
## , , 15
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.03769675 0.04496360 0.03625093 0.03448247 0.06160769 0.02083333 0.04993055 0.03835882
## 97.5% 0.05567474 0.04979351 0.04063134 0.04390842 0.06658288 0.15555556 0.05716525 0.04391404
##
## , , 16
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.03663811 0.07683403 0.04858513 0.06227866 0.03850884 0.01666667 0.08332725 0.05995569
## 97.5% 0.05462616 0.08281403 0.05537585 0.07661173 0.04252290 0.04545455 0.09256604 0.06932631
##
## , , 17
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.06614006 0.06886244 0.07446137 0.03719875 0.04202283 0.01666667 0.06836274 0.04116116
## 97.5% 0.09457072 0.07511246 0.08115276 0.04848396 0.04514804 0.04545455 0.07613655 0.04747090
##
## , , 18
##
##               [,1]         [,2]         [,3]         [,4]         [,5]       [,6]         [,7]
## 2.5%  0.0004716981 2.725379e-05 2.678730e-05 0.0001659999 1.663225e-05 0.01666667 6.376219e-05 5.103
## 97.5% 0.0005491488 3.019996e-05 3.068617e-05 0.0007721665 1.869569e-05 0.04545455 7.105364e-05 5.641
##
## , , 19
##
##            [,1]       [,2]      [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.1149506 0.04851073 0.1210934 0.02060298 0.06980848 0.02083333 0.03617125 0.01816138
## 97.5% 0.1467033 0.05362861 0.1314308 0.02841960 0.07368739 0.08888889 0.04278154 0.02262071
##
## , , 20
##
##              [,1]        [,2]       [,3]       [,4]       [,5]       [,6]       [,7]        [,8]
## 2.5%  0.005116862 0.007162380 0.01824446 0.02094082 0.01210010 0.02083333 0.01435023 0.008839036
## 97.5% 0.013854674 0.009207843 0.02134023 0.02860709 0.01402983 0.08888889 0.01810591 0.011510470
##
## , , 21
##
##               [,1]         [,2]         [,3]         [,4]         [,5]       [,6]         [,7]
## 2.5%  0.0004807635 0.0005161066 2.678730e-05 0.0001626267 1.663225e-05 0.01666667 6.43914e-05 5.1474
## 97.5% 0.0037394162 0.0015884559 3.068617e-05 0.0001865002 1.869569e-05 0.04545455 8.97756e-04 3.8473
##
## , , 22
##
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
## 2.5%  0.05357968 0.02615564 0.05271038 0.04365302 0.02798338 0.01666667 0.02220757 0.02222699
## 97.5% 0.07754166 0.02961028 0.05791758 0.05449145 0.03133652 0.04545455 0.02762013 0.02652420
```

```r
head(T_ci)
```

```
## , , 1
##
##             [,1]       [,2]        [,3]        [,4]         [,5]       [,6]       [,7]       [,8]
## 2.5%  0.01349827 0.02694777 0.003378155 0.005789082 0.0002925743 0.02173913 0.02354641 0.01611129
## 97.5% 0.02423141 0.03090281 0.004517958 0.010229955 0.0006389966 0.12500000 0.02873810 0.02001761
##
## , , 2
##
##            [,1]      [,2]      [,3]      [,4]       [,5]       [,6]      [,7]      [,8]
## 2.5%  0.5870119 0.5038996 0.1045434 0.1053153 0.01685508 0.02173913 0.3751841 0.2210258
## 97.5% 0.6257567 0.5269629 0.1119504 0.1183659 0.01883349 0.12500000 0.3895853 0.2334833
##
## , , 3
##
##             [,1]      [,2]      [,3]       [,4]         [,5]       [,6]       [,7]       [,8]
## 2.5%  0.02373146 0.1064581 0.8078762 0.01560642 0.0001602061 0.02173913 0.08042008 0.06532819
## 97.5% 0.03819293 0.1161312 0.8168627 0.02222278 0.0005058731 0.12500000 0.09126838 0.07349410
##
## , , 4
##
##             [,1]       [,2]        [,3]      [,4]        [,5]       [,6]       [,7]       [,8]
## 2.5%  0.01440015 0.02354322 0.003828153 0.6940130 0.002441762 0.02173913 0.01608261 0.01146981
## 97.5% 0.02805486 0.02678631 0.005300806 0.7001462 0.003267694 0.12500000 0.02055086 0.01436029
##
## , , 5
##
##             [,1]       [,2]        [,3]       [,4]      [,5]       [,6]       [,7]       [,8]
## 2.5%  0.01748027 0.07950875 0.004688498 0.02912687 0.9089374 0.03225806 0.06218135 0.03729350
## 97.5% 0.03071166 0.09028703 0.006743756 0.03770840 0.9118885 0.15000000 0.07280317 0.04327075
##
## , , 6
##
##               [,1]         [,2]         [,3]         [,4]         [,5]      [,6]         [,7]
## 2.5%  0.0004748338 2.787212e-05 2.679735e-05 0.0001629979 1.688962e-05 0.1250000 6.444462e-05 5.17698
## 97.5% 0.0005534034 3.096107e-05 3.069935e-05 0.0001869884 7.212656e-05 0.7021277 2.775851e-04 2.18978
##
## , , 7
##
##            [,1]      [,2]       [,3]       [,4]       [,5]       [,6]      [,7]      [,8]
## 2.5%  0.1403848 0.1333075 0.02753708 0.05239020 0.01508257 0.02173913 0.3480118 0.1160330
## 97.5% 0.1685191 0.1447695 0.03138037 0.06331324 0.01744083 0.12500000 0.3650542 0.1251127
##
## , , 8
##
##            [,1]       [,2]       [,3]       [,4]       [,5]      [,6]       [,7]      [,8]
## 2.5%  0.1289314 0.09223128 0.03417253 0.06658213 0.05022219 0.0500000 0.05981712 0.5074345
## 97.5% 0.1597192 0.09993996 0.03794007 0.07932027 0.05347118 0.1714286 0.06777946 0.5158857
```

(g) Comparison

```
# Plot box plot of accuracies
boxplot(rand_acc, viterbi_acc, ylim=c(0, 1), ylab="Accuracy", xlab="Algorithm",
    names=c("Random", "Viterbi")
)
```



Render this .rmd into a pdf

```
library(rmarkdown)
render("viterbi.Rmd", pdf_document(TRUE), "viterbi.pdf")
```