Machine Learning 2ST129 26605 HT2023 Assignment 7

Anonymous Student

December 31, 2023

Contents

General Information

- Time used for reading: 3 hours
- Time used for basic assignment: 16 hours
- Time used for extra assignment: NA
- Good with lab: I think that in general it was a good overall structure of the different tasks. For example, that we had to implement something, plot it and also interpret it, which I felt was good for learning.
- Things improve with lab: In my opinion, it would have been good if there was some more information about the code, especially for task 1. For example, now I mostly copy pasted the code from the linked guide to get it to work, but I do not really know what some of that parts are supposed to represent.

Task 1

1.1

First we load the required libraries as well as the mnist data

```
#Libraries
 library(tidyverse)
 library(xtable)
 library(tensorflow)
 library(keras)
```

```
mnist <- dataset_mnist()
x_train <- mnist$train$x/255
x_test <- mnist$test$x/255
x_train <- array_reshape(x_train, c(nrow(x_train), 784), order = "F")
x_test <- array_reshape(x_test, c(nrow(x_test), 784), order = "F")
```

1.2

Now we want to implement a a one-layer (encoder and decoder) feed-forward variational autoencoder with two latent dimensions such that both the encoder and decoder layers should each have 200 hidden units. The following code is pretty much all based on the example as provided by the variational_autoencoder.R guide on GitHub. Though with some minor differences such that we change the dimensions.

```
if (tensorflow::tf$executing_eagerly())
  tensorflow::tf$compat$v1$disable_eager_execution()
batch_size <- 100L
original_dim <- 784L
latent_dim <- 2L
intermediate_dim <- 200L
epochs <- 50L
epsilon_std <- 1.0

# Model definition ----------------------------------------------------------

x <- layer_input(shape = c(original_dim))
h <- layer_dense(x, intermediate_dim, activation = "relu")
z_mean <- layer_dense(h, latent_dim)
z_log_var <- layer_dense(h, latent_dim)

sampling <- function(arg){
  z_mean <- arg[, 1:(latent_dim)]
  z_log_var <- arg[, (latent_dim + 1):(2 * latent_dim)]

  epsilon <- k_random_normal(
    shape = c(k_shape(z_mean)[[1]]),
    mean=0.,
    stddev=epsilon_std
  )

  z_mean + k_exp(z_log_var/2)*epsilon
}


z <- layer_concatenate(list(z_mean, z_log_var)) %>%
  layer_lambda(sampling)

# we instantiate these layers separately so as to reuse them later
decoder_h <- layer_dense(units = intermediate_dim, activation = "relu")
```

```
decoder_mean <- layer_dense(units = original_dim, activation = "sigmoid")
h_decoded <- decoder_h(z)
x_decoded_mean <- decoder_mean(h_decoded)

# end-to-end autoencoder
vae <- keras_model(x, x_decoded_mean)

# encoder, from inputs to latent space
encoder <- keras_model(x, z_mean)

# generator, from latent space to reconstructed inputs
decoder_input <- layer_input(shape = latent_dim)
h_decoded_2 <- decoder_h(decoder_input)
x_decoded_mean_2 <- decoder_mean(h_decoded_2)
generator <- keras_model(decoder_input, x_decoded_mean_2)


vae_loss <- function(x, x_decoded_mean){
  xent_loss <- (original_dim/1.0)*loss_binary_crossentropy(x, x_decoded_mean)
  kl_loss <- -0.5*k_mean(1 + z_log_var - k_square(z_mean) - k_exp(z_log_var), axis = -1L)
  xent_loss + kl_loss
}

vae %>% compile(optimizer = "rmsprop", loss = vae_loss)
```

1.3

```
print(vae)
```

```
Model: "model"
__________________________________________________________________________________
 Layer (type)           Output Shape             Param   Connected to
                                                  #
==================================================================================
 input_1 (InputLayer)   [(None, 784)]            0       []
 dense (Dense)          (None, 200)              15700   ['input_1[0][0]']
                                                 0
 dense_1 (Dense)        (None, 2)                402     ['dense[0][0]']
 dense_2 (Dense)        (None, 2)                402     ['dense[0][0]']
 concatenate (Concaten  (None, 4)                0       ['dense_1[0][0]',
 ate)                                                     'dense_2[0][0]']
 lambda (Lambda)        (None, 2)                0       ['concatenate[0][0]']
 dense_3 (Dense)        (None, 200)              600     ['lambda[0][0]']
 dense_4 (Dense)        (None, 784)              15758   ['dense_3[0][0]']
                                                 4
==================================================================================
Total params: 315988 (1.21 MB)
Trainable params: 315988 (1.21 MB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________________________
```

As for how many weights there are used to compute μ and σ^2 for the latent variables, we have set the variable original_dim to 784 and the variable intermediate_dim to 200. Hence in total we have (784+1) * 200 weights.

The layer repersenting the latent variable is the object **z** in the code, which is obtianed by combining z_mean and z_log_var.

The lambda layer is used to sample from the specified distribution from z, since we get both the mean

and variance. Then this is used as the latent representation in the variational autoencoder.

1.4

Now we train the VAE on the MNIST data for 50 epochs and visualize some of the results.

```
results <- vae %>% fit(
  x_train, x_train,
  shuffle = TRUE,
  epochs = epochs,
  batch_size = batch_size,
  validation_data = list(x_test, x_test)
)
```

First we can simply plot the history over the epochs.

```
knitr::include_graphics("vae_hist.png")
```

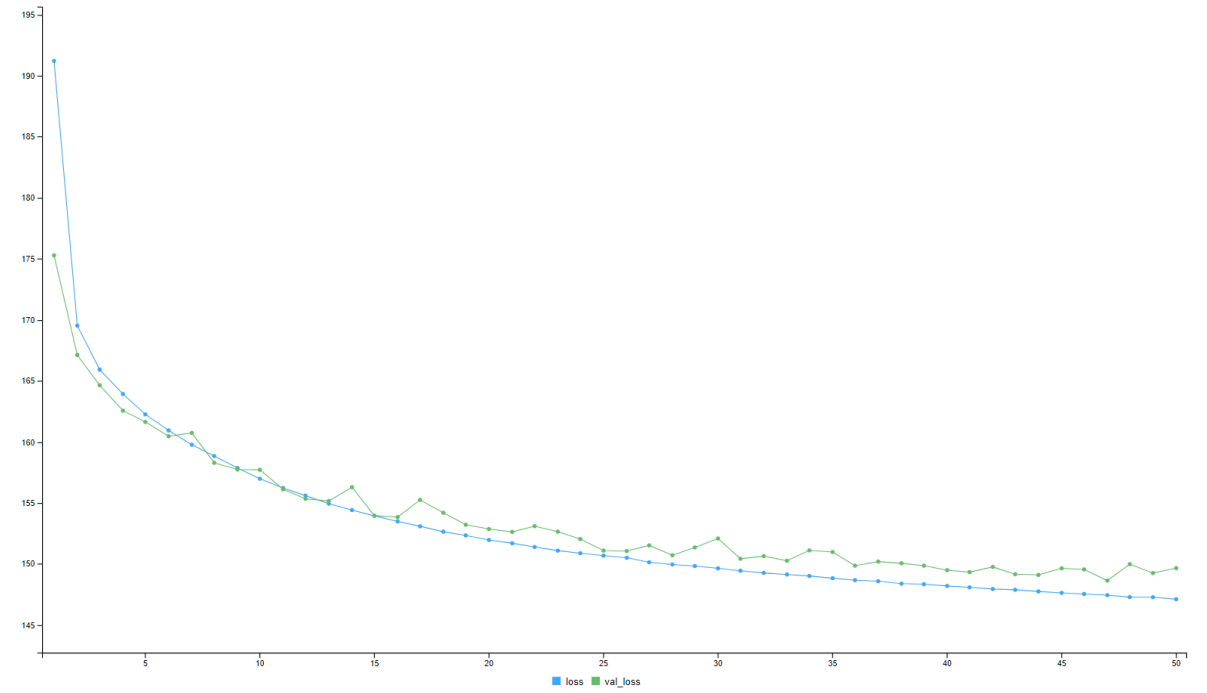

Figure 1: epochs history

Next plot latent state for the different numbers.

```
x_test_encoded <- predict(encoder, x_test, batch_size = batch_size)

x_test_encoded %>%
  as_data_frame() %>%
  mutate(class = as.factor(mnist$test$y)) %>%
  ggplot(aes(x = V1, y = V2, colour = class)) + geom_point()
```

```
knitr::include_graphics("vae_scatter.png")
```

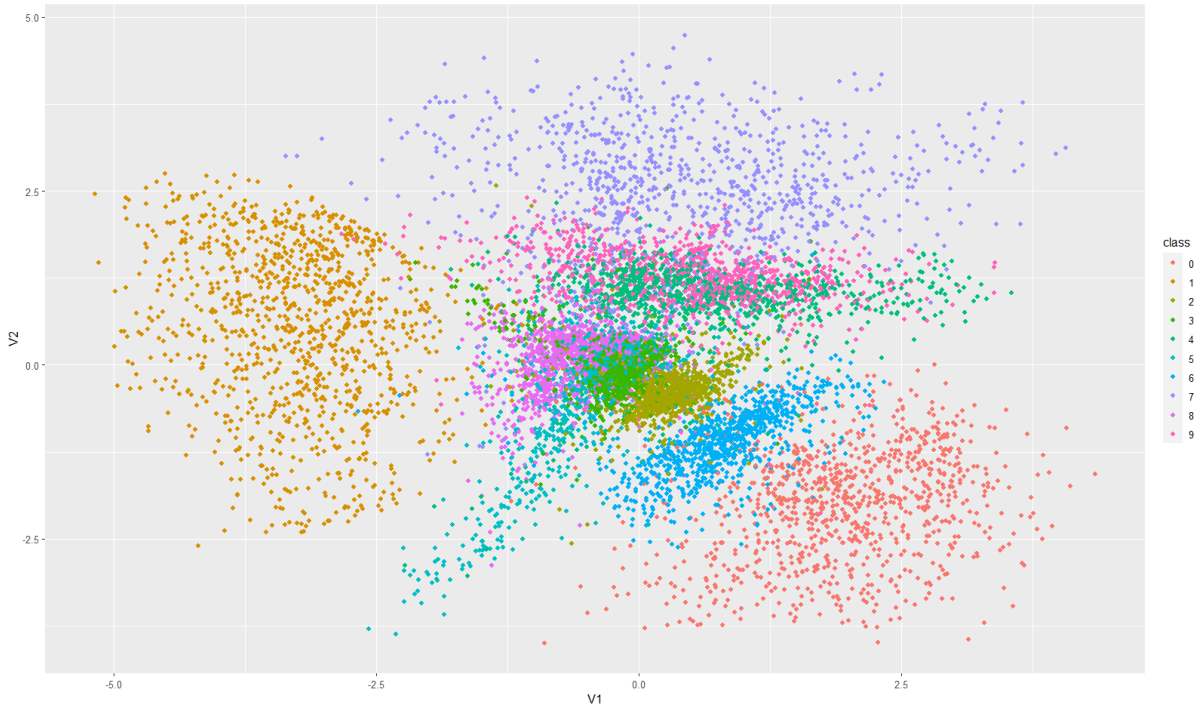

Figure 2: latent states for different numbers

As can be seen from the output. This scatter plot shows how the different classes (digits) are distributed for the hidden states. In general the classes that are close to each other and clustered are better represented, whereas big overlaps means that it is not as good represented. Looking at the plot, then we can see for example that the digit 2 is very centered in the middle and thus well represented, whereas for example the digit 1 is more spread out on the left and not as well represented.

1.5

Now to encode all the 2s in the test dataset to the latent state using the encoder.

```
x_digit_2 <- x_test[mnist$test$y == 2, , drop = FALSE]
digit_2_LS <- predict(encoder, x_digit_2)
latent_states_mean <- colMeans(digit_2_LS)
latent_states_mean
```

```
[1]  0.1862775 -0.4990619
```

1.6

And lastly we plot this as an image using the decoder.

```
matrix(latent_states_mean, nrow=1) %>%
  predict(generator, .) %>%
  array_reshape(dim=c(28,28), order = c("F")) %>%
  image(., col = gray.colors(n =  256))
```

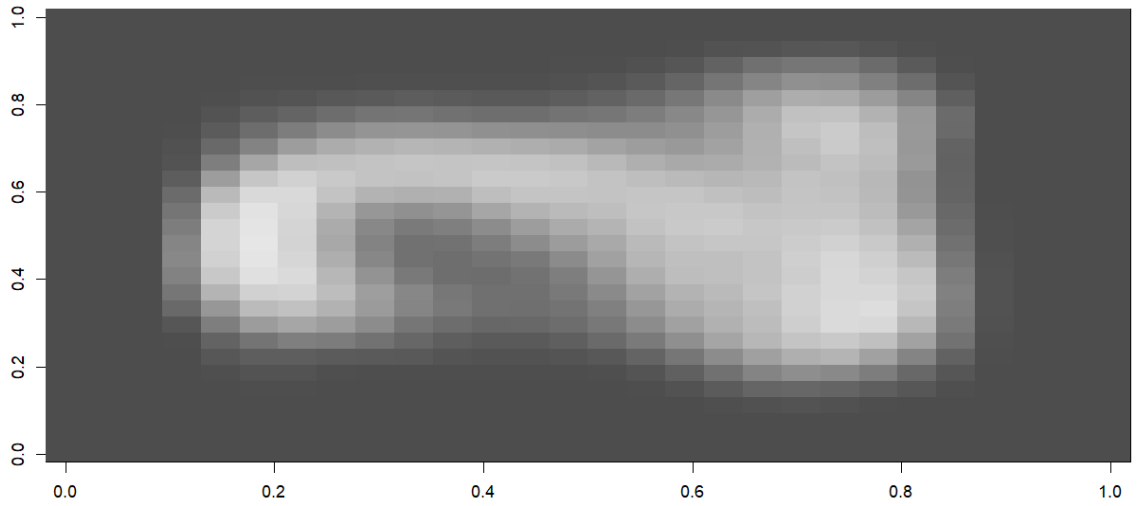

Figure 3: 28 x 28 pixel image of the latent state

Task 2

2.1

First we load the data and remove the stopwords.

```
library(uuml)
library(dplyr)
library(tidytext)
data("pride_and_prejudice")
data("stopwords")
```

```
pap <- pride_and_prejudice
pap <- anti_join(pap, y = stopwords[stopwords$lexicon == "snowball",])
## Joining with `by = join_by(word)`
```

2.2

Next we remove rare words

```
word_freq <- table(pap$word)
rare_words <- data.frame(word = names(word_freq[word_freq <= 5]),
                         stringsAsFactors = FALSE)
pap <- anti_join(pap, y = rare_words)
## Joining with `by = join_by(word)`
```

2.3

Now to compute a document term matrix where each paragraph is treated as a document.

```
library(tm)
crp <- aggregate(pap$word, by = list(pap$paragraph), FUN = paste0, collapse = " " )
names(crp) <- c("paragraph", "text")
s <- SimpleCorpus(VectorSource(crp$text))
m <- DocumentTermMatrix(s)
print(s)
```

```
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 2051
```

```
print(m)
```

```
<<DocumentTermMatrix (documents: 2051, terms: 1689)>>
Non-/sparse entries: 42130/3422009
Sparsity           : 99%
Maximal term length: 15
Weighting          : term frequency (tf)
```

Here we have 2051 documents and 1689 terms.

2.4

For this step we runt the Gibbs sampler for 2000 iterations.

```
library(topicmodels)
K <- 10
control <- list(keep = 1, delta = 0.1, alpha = 1, iter = 2000)
tm <- LDA(m, k = K, method = "Gibbs", control)
```

2.5

Next we extract some parameters.

```
lls <- extract_log_liks(tm)
theta <- extract_theta(tm)
phi <- extract_phi(tm)
```

2.6

And now to check if the log likelihood has converged.

```
plot(lls, ylab="Log-likelihood", xlab="Epoch")
```

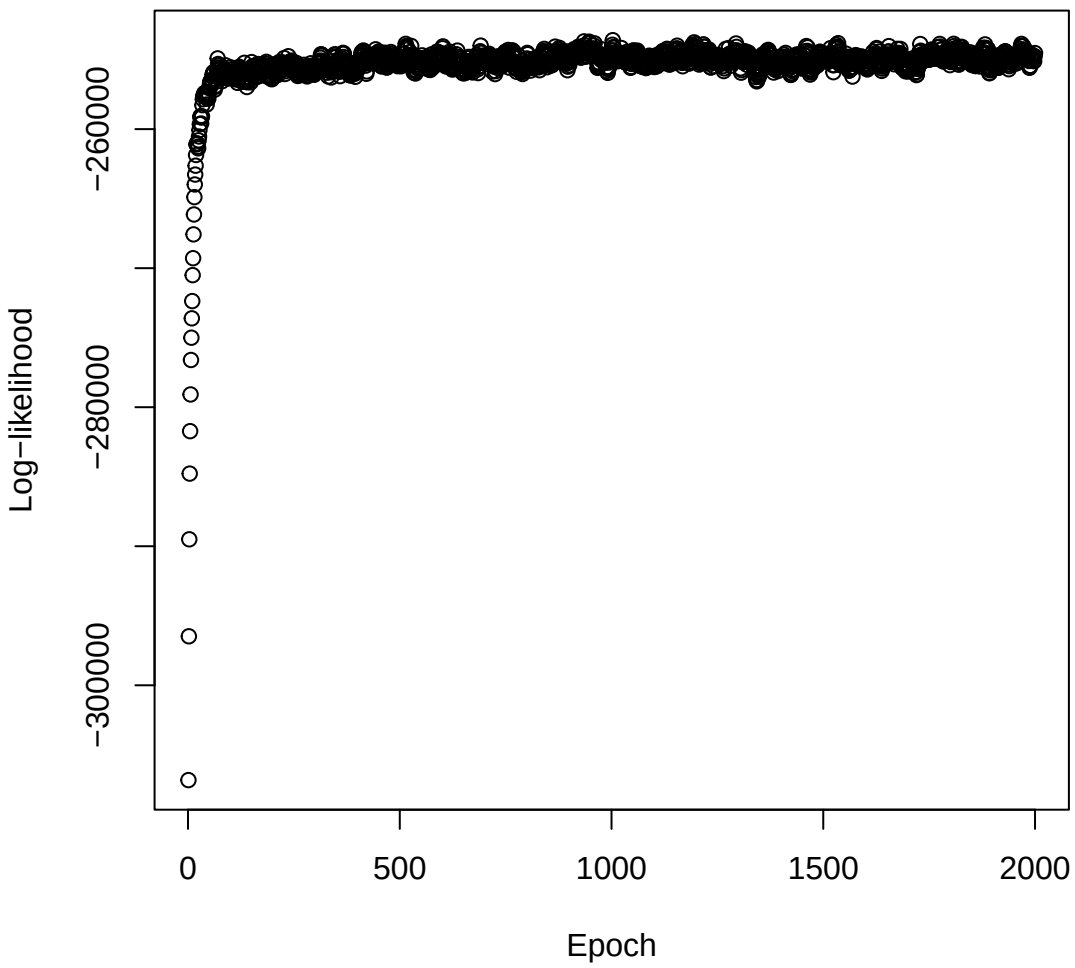

Figure 4: Log likelihood for epochs

Looking at the figure, it seems as if the log-likelihood has converged.

2.7

Now we extract the top 20 words for each of the 10 topic.

```
#function to extract n number of top words given topic
extract_func <- function(topic_index, phi_parameter, n_words){
  words <- colnames(phi_parameter)[order(-phi_parameter[topic_index,])]
  return(words[1:n_words])
}

#using the function over all 10 topics
sapply(1:10, FUN =function(i){extract_func(topic_index=i, phi_parameter = phi, n_words=20)})
```

```
      [,1]        [,2]        [,3]      [,4]          [,5]          [,6]
 [1,] "now"       "lady"      "lydia"   "mrs"         "elizabeth"   "will"
 [2,] "letter"    "collins"   "good"    "bennet"      "darcy"       "can"
 [3,] "nothing"   "catherine" "wickham" "day"         "said"        "may"
 [4,] "soon"      "great"     "well"    "two"         "made"        "shall"
```

```
 [5,] "wickham"   "sir"       "never"   "soon"        "little"      "know"
 [6,] "give"      "charlotte" "said"    "time"        "looked"      "dear"
 [7,] "without"   "time"      "know"    "morning"     "manner"      "said"
 [8,] "done"      "lucas"     "mother"  "longbourn"   "speak"       "say"
 [9,] "first"     "first"     "aunt"    "next"        "answer"      "think"
[10,] "ill"       "make"      "sure"    "till"        "eyes"        "must"
[11,] "present"   "young"     "kitty"   "netherfield" "might"       "lizzy"
[12,] "known"     "rosings"   "father"  "went"        "pemberley"   "one"
[13,] "since"     "way"       "gone"    "daughters"   "spoke"       "believe"
[14,] "last"      "family"    "away"    "might"       "mind"        "yes"
[15,] "received"  "little"    "meryton" "husband"     "felt"        "let"
[16,] "knew"      "daughter"  "going"   "home"        "continued"   "hope"
[17,] "long"      "however"   "now"     "leave"       "elizabeth's" "replied"
[18,] "hardly"    "much"      "place"   "coming"      "without"     "well"
[19,] "therefore" "cousin"    "married" "family"      "whether"     "indeed"
[20,] "told"      "william"   "thought" "added"       "last"        "upon"
      [,7]        [,8]        [,9]        [,10]
 [1,] "miss"      "man"       "must"      "elizabeth"
 [2,] "bingley"   "never"     "every"     "room"
 [3,] "darcy"     "think"     "elizabeth" "house"
 [4,] "jane"      "good"      "might"     "said"
 [5,] "much"      "always"    "though"    "mrs"
 [6,] "sister"    "young"     "happiness" "saw"
 [7,] "elizabeth" "many"      "less"      "every"
 [8,] "friend"    "world"     "affection" "party"
 [9,] "though"    "much"      "make"      "ladies"
[10,] "enough"    "must"      "love"      "soon"
[11,] "see"       "father"    "still"     "began"
[12,] "nothing"   "life"      "marriage"  "conversation"
[13,] "two"       "anything"  "yet"       "walk"
[14,] "bingley's" "ever"      "almost"    "took"
[15,] "one"       "little"    "without"   "walked"
[16,] "sisters"   "manners"   "happy"     "towards"
[17,] "say"       "means"     "really"    "turned"
[18,] "well"      "pride"     "hope"      "civility"
[19,] "however"   "often"     "regard"    "door"
[20,] "asked"     "agreeable" "wish"      "entered"
```

Looking at the topics, then I would say that perhaps topic 1 since it contains some words that reflects some key concepts and general state of the novel. For example, that the idea is the everyone must find someone to marry in order to bring happiness, but in the process of doing so the sisters reflect on the situation and general feelings on whether or not they actually want to go thorugh with it or not . Annother topic is perhaps topic 4 since it contains some of the names of the characters in the novel, or the relationships between the chaaracetsr suc has bennet, bingley, Elizabeth or netherfield, as well as some descriptions such as daughters, sisters or mother and so on.

2.8

Next we visualize how these evolve over the paragraphs.

```
plot_topics <- function(theta_matrix, col_index1, col_index2) {
  paragraphs <- 1:nrow(theta_matrix)
  topic1 <- theta_matrix[, col_index1]
  topic2 <- theta_matrix[, col_index2]

  new_df <- tibble(paragraphs, topic1, topic2)
  name1 <- paste("Topic no. ", col_index1)
  name2 <- paste("Topic no. ", col_index2)
```

```
  new_df %>%
    ggplot(aes(x = paragraphs)) +
    geom_line(aes(y = topic1, color = name1), size = 1) +
    geom_line(aes(y = topic2, color = name2), size = 1) +
     scale_color_discrete(name="") +
    labs(
      title = "Topics over time on paragraphs",
      x = "Paragraph",
      y = expression(theta)
    )
}

plot_topics(theta, 1, 4)
```

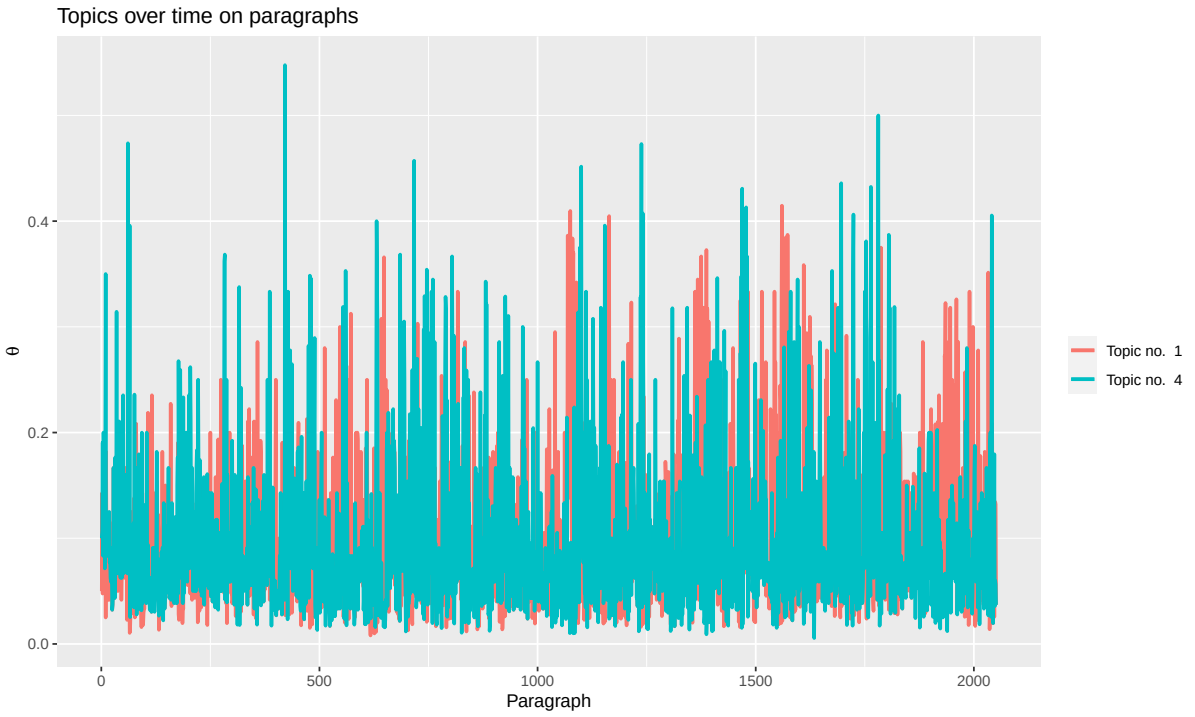

Figure 5: vizualisation of topics over time

2.9

Lastly, we visualize it again as a rolling mean with k=5.

```
head(zoo::rollmean(theta[,1], k=5))
```

```
         3          4          5          6          7          8
0.09142857 0.09681319 0.08157509 0.08633700 0.10300366 0.08133700
```

```
theta_rolling_m <- zoo::rollmean(x=theta, k =5)
plot_topics(theta_rolling_m, 1, 4)
```

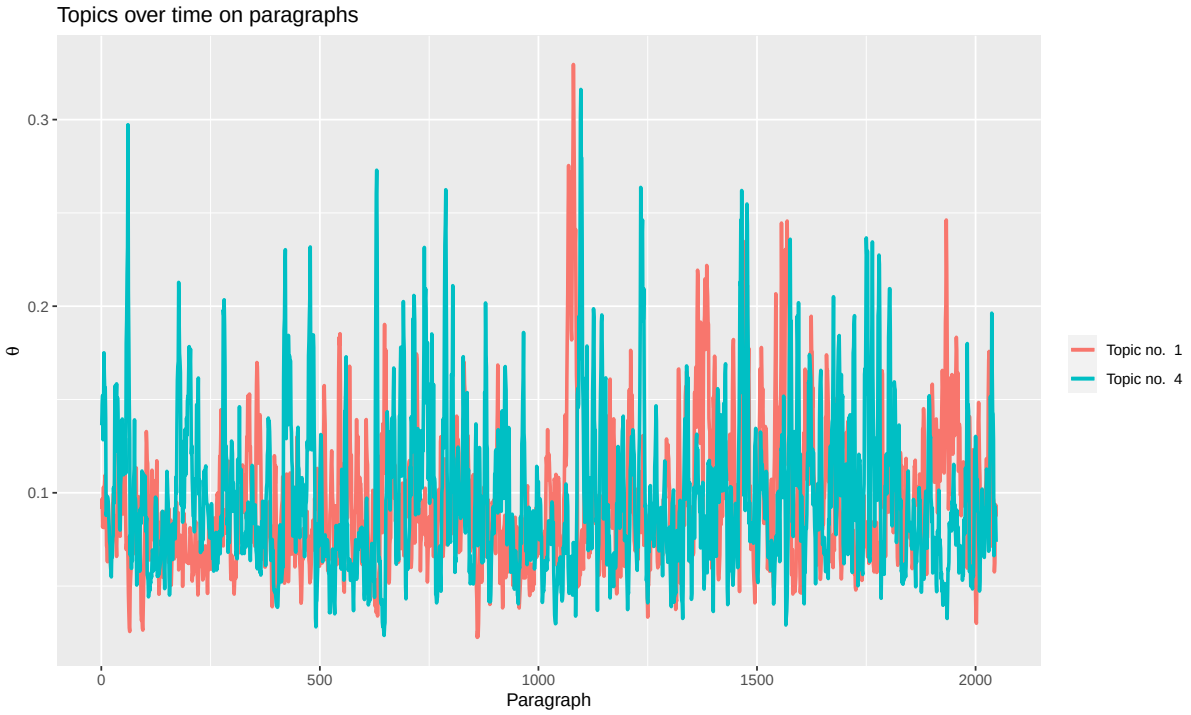

Figure 6: Topic rolling means

Looking as this figure, we see that in general Topic number 4 seems to always somewhat relevant for the different paragraphs such that we see the value for theta jumping back and fort between 0.5 - 0.25, but it does so somewhat consistently across the different paragraphs, albeit there being a decrease in the middle for a short while. Looking at topic number 1 , we see that it is not as prevalent in the first quarter of the paragraphs, but then evolves to be more relevant in the middle and the end , which perhaps can be attributed due to the fact that it contained many names that becomes more relevant during that period.