Uppsala University

Introduction to Machine Learning, Big Data, and AI

Assignment 8

General information

- The recommended tool in this course is R (with the IDE R-Studio). You can download R **here** and R-Studio **here**. You are allowed to use Python and Jupyter Notebooks, although the assignments may use data available only through the R package, a problem you would need to solve yourself.
- Report all results in a single, `*.pdf`-file. *Other formats, such as Word, Rmd, Jupyter Notebook, or similar, will automatically be failed.* Although, you are allowed to first knit your document to Word and then print the assignment as a PDF from Word if you find it difficult to get TeX to work.
- The report should be submitted to the **Studium**.
- To pass the assignments, *all questions should be answered*, although minor errors are ok.
- A report that do not contain the general information (see template) will be automatically rejected.
- When working with R, we recommend writing the reports using R markdown and the provided **R markdown template**. The template includes the formatting instructions and how to include code and figures.
- If you have a problem with creating a PDF file directly from R markdown, start by creating an HTML file, and then just print the HTML to a PDF.
- Instead of R markdown, you can use other software to make the pdf report, but the same instructions for formatting should be used. These instructions are also available in **the PDF produced from the R markdown template**.
- The course has its own R package `uuml` with data and functionality to simplify coding. To install the package just run the following:

  1. `install.packages("remotes")`
  2. ```
     remotes::install_github("MansMeg/IntroML",
     subdir = "rpackage")
     ```

- We collect common questions regarding installation, and technical problems in a course Frequently Asked Questions (FAQ). This can be found **here**.
- Deadline for all assignments is **Sunday at 23.59**. See the course page for dates.
- If you have any suggestions or improvements to the course material, please post in the course chat feedback channel, create an issue, or submit a pull request to the public repository!
     ```

1 Variational Autoencoders

We are now going to implement a variational autoencoder in R using Tensorflow. A lot of the code needed for this assignment can be found **here**.

Note! Running Keras can be computationally heavy, and I suggest not to run the code in `markdown`. Instead, run the code in R and copy in the results as output (see the assignment template for an example).

1. Start by loading in the MNIST data in R.
2. Now, implement a one-layer (encoder and decoder) feed-forward variational autoencoder with two latent dimensions. Both the encoder layer and the decoder layer should have 200 hidden units. You should end up with a variational autoencoder with roughly 310 000 - 320 000 parameters.
3. Print the model and include it in your report. How many weights are used to compute μ and σ^2 for the latent variables? What layer represent the latent variables?
4. Now train your variational autoencoder on the MNIST data for 50 epochs. Visualize the latent state for the different numbers. How do you interpret this result? What numbers are better represented by the latent state?
5. Finally, encode all the 2:s in the MNIST test dataset to the latent state using your encoder. What is the mean of the digits "2" in the two latent dimensions? *Hint!* See `y_test` for the MNIST numbers.
6. Visualize this value of the latent state as a 28 by 28-pixel image using your decoder.

2 Variational Autoencoders using Convolutional Neural Networks

As we have seen previously, for images, we can get better performance using Convolutional Neural Networks. Hence we are going to repeat the exercise above using a convolutional neural network as encoder and decoder. You can find detailed code **here**.

1. Now implement a four-layer (encoder and decoder) convolutional neural network with two latent dimensions. There should be 50 filters in each convolutional layer. Also, a dense layer should be included as the last step in the encoder and the first step in the decoder. These layers should have 100 hidden units. You should end up with a variational autoencoder with roughly 2M parameters.
2. Print the model and include in your report. How many weights are used to compute μ and σ^2 for the latent variables? What layer represent the latent variables?
3. Now train your CNN variational autoencoder on the MNIST data for five epochs. Visualize the latent state for the different numbers. How do you interpret this result? Compare these result with the results from the feed-forward autoencoder.

4. Finally, encode all the 2:s in the MNIST test dataset to the latent state using your decoder (Hint!, see `y_test` for numbers). What is the mean of the digits "2" in the two latent dimensions?

5. Visualize the mean value of the digit 2 of the latent state as a 28 by 28-pixel image using your decoder.

3 Topic Models

We are now going to analyze the classical book Pride and Prejudice by Jane Austen using a probabilistic topic model. If you have not read the book, **here** you can read up on the story of this classical book.

Reading instructions . For this part of the assignment, Griffith and Steyvers (2004) is the main reference. I would also recommend reading Blei (2012) before starting with the assignment.

We are now going to implement a Gibbs sampler to estimate ten different topics occurring in Pride and Prejudice and study where they occur. A tokenized version of the book and a `data.frame` with stopwords can be loaded as follows:

```
library(uuml)
library(dplyr)


##
## Attaching package:  'dplyr'
## The following objects are masked from 'package:stats':
##
##    filter, lag
## The following objects are masked from 'package:base':
##
##    intersect, setdiff, setequal, union

data("pride_and_prejudice")
data("stopwords")
```

1. As a first step we will remove stopwords (common english words without much semantic information):

```
pap <- pride_and_prejudice
pap <- anti_join(pap, y = stopwords[stopwords$lexicon == "snowball",])


## Joining, by = "word"
```

2. Then we will remove rare words. Here we remove words that occur less than five times.

```
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, by = "word"
```

3. Now we have a corpus we can used to implement a probabilistic topic model. We do this by using the `topicmodels` R package. As a first step we will compute a document term matrix using the `tm` package, where we treat each paragraph as a document. How many documents and terms (word types) do you have?

```
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)
```

4. To compute a topic model with ten topics, we use a Gibbs sampling algorithm. Below is an example of how we can run a Gibbs sampler for 1000 iterations. Run your topic model for 2000 iterations.

```
library(topicmodels)
K <- 10
# Note: delta is beta in Griffith and Steyvers (2004) notation.
control <- list(keep = 1, delta = 0.1, alpha = 1, iter = 2000)
tm <- LDA(m, k = K, method = "Gibbs", control)
```

5. In the `uuml` R package you have three convenience functions to extract Θ, Φ and the log-likelihood values at each iteration. This is the parameter notation used in Griffiths and Steyvers (2004).

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

6. As a first step, check that the model has converged by visualizing the log-likelihood over epochs/iterations. Does it seem like the model have converged?

7. Extract the 20 top words for each topic (i.e. the words with the highest probability in each topic). Choose two topics you find to be coherent/best (the top words seem to belong together). Interpret these two topics based on the storyline of the book. What have these two topics captured?

8. Visualize these two topics evolve over the paragraphs in the books by plotting the θ parameters for that topic over time (paragraphs) in the book. How do these two topics evolve? If you want, you can take a rolling mean of the theta parameters to more easily show the changes in the topic over the book.