Uppsala University

Introduction to Machine Learning, Big Data, and AI

Assignment 7

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 can 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 first to knit your document to Word or HTML and then print the assignment as a PDF from Word or HTML if you find it difficult to get TeX to work.
- The report should be written in English.
- If a question is unclear in the assignment. Write down how you interpret the question and formulate your answer.
- You should submit the report to **Studium**.
- To pass the assignments, *you should answer all questions not marked with **, although minor errors are ok.
- To get VG on the assignment, *all questions should be answered, including questions marked with a **, although minor errors are ok.
- A report that does not contain the general information (see the **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.
- Instead of R markdown, you can use other software to make the pdf report, but you should use the same instructions for formatting. 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**.
- Deadlines for all assignments are **Sunday 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 **here**, or submit a pull request to the public repository.

Contents

1 General Questions

You will be able to answer the following questions based on the reading assignments for this assignment. See the course plan for detailed reading **here**.

1. What is the relationship between PCA and linear autoencoders? (max 1 paragraph)
2. When we maximize the ELBO, what two things do we optimize in a VAE? (max 1 paragraph)
3. What is a topic in a latent Dirichlet topic model? (max 1 paragraph)
4. What is a mixed membership model? (max 1 paragraph)

2 Variational Autoencoders

We are now going to implement a variational autoencoder in R using Tensorflow. You can find a lot of the code needed for this assignment **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 the results as output (see the assignment template for an example).

1. Start by loading the MNIST data in R. See the link above or previous assignments for details.
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.
   (a) How many weights (parameters) are used to compute μ and σ^2 for the latent variables?
   (b) What layer represent the latent variables?
   (c) What does the lambda layer do in the model?
4. Now train your variational autoencoder on the MNIST data for 50 epochs. Visualize the latent state for the different numbers.
   (a) How do you interpret this latent state?
   (b) What numbers are better represented by the latent state?
   (c) What number is less well 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. *Note!* Do not retrain your VAE. Just use the one you have already trained.
6. Visualize this value of the latent state as a 28 by 28-pixel image using your decoder.

3 Topic Models

We will now 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.

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

We will use 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?

```r
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 2000 iterations. Run your topic model for 2000 iterations.

```r
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).

```r
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 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. Think of this as the time-line of the book. On the y-axis, you should plot θ_i for your chosen topic i and the x-axis should be the paragraph number (first paragraph has number 1 and so forth).
9. How do these two chosen topics evolve over the course in the book? 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. *Hint!* Here `zoo::rollmean()` might be a good function to use.

4 * Variational Autoencoders using Convolutional Neural Networks

This task is only necessary to get one point toward a *pass with distinction* (VG) grade. Hence, if you do not want to get a *pass with distinction* (VG) point, you can ignore this part of the assignment.

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. *Note!* 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 it in your report. How many weights (parameters) 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 results 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.

References

Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235, 2004.

David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.