Uppsala University

Introduction to Machine Learning, Big Data, and AI

Assignment 4

General information

- The recommended tool in this course is R (with the IDE R-Studio). You can download R **here** and R-Studio **here**.
- Report all results in a single, *.pdf -file. *Other formats, such as Word, Rmd, or similar, will automatically be failed.*
- The report should be submitted to the Studium.
- 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 Recurrent Neural Networks

As a first step, we will try using Recurrent Neural Networks on a text classification task. Here Chapter 6 - 6.2 in Chollet and Allaire (2018) will be the primary reference.

1.1 Text classification using Keras

Start by downloading the IMBD dataset and preprocess it as follows.

```
library(tensorflow)
library(keras)
imdb <- dataset_imdb(num_words = max_features)
c(c(input_train, y_train), c(input_test, y_test)) %<-% imdb
input_train <- pad_sequences(input_train, maxlen = maxlen)
input_test <- pad_sequences(input_test, maxlen = maxlen)
```

1. Create a simple ordinary neural network for text classification in Keras using an embedding layer of dimension 16 and a hidden layer with 32 hidden states (and an output layer with one unit). Use a validation split of 20% observations in the validation set and a batch size of 128. Train your network using rmsprop and use binary cross-entropy as the loss function. Print the model and return your validation accuracy after ten epochs. Most of these settings we will reuse later in the assignments.

2. Explain what the embedding layer is doing.

3. Now setup a simple Recurrent Neural Network with 32 hidden units based on the same embedding layer. Print the model and present your result.

4. The model you have implemented now. Which recurrent network does best describes your model you have now implemented out of Fig. 10.3 and 10.5 in Goodfellow et al. (2017)? Motivate why.

5. Explain the role of the parameters U, W in the recurrent net (see Goodfellow et al., 2017 Section 10.2). How large are these parameters in your model specified above, and where are they included in the Keras output?

6. Now, explain the role of the parameters V (see Goodfellow et al., 2017 Section 10.2). How many parameters are included in V, and where are they included in the Keras output?

7. We are now going to implement an LSTM network. Implement a standard LSTM network with 32 hidden units using Keras. Print your model and your validation accuracy.

8. Extend your network in ways you think fit. Report at least two other extensions/-modifications of your network and why you chose them. Report the Keras model output and validation accuracy.

9. Reason why we cannot get as good performance as we could get with the MNIST dataset. Is the problem bias, variance, or the Bayes error? Why?

2 Transformers and Attention

In this assignment, we will implement a multi-head attention transformer layer. These layers are currently used within all transformer based models, such as GPT and different BERT-models. A good read for this assignment is the Illustrated transformer by Jay Alammar, which can be found [here](http://jalammar.github.io/illustrated-transformer/).

Start by loading the parameters that we are going to use for the implementation as follows.

```
library(uuml)
data("transformer_example")
```

1. Start out by computing the query, key and value matrices for one attention head by implementing it in a function you call `qkv()`. The function should work like this.

```
# Pick out the matrix for the first attention head
Wq <- transformer_example$Wq[,,1]
Wk <- transformer_example$Wk[,,1]
Wv <- transformer_example$Wv[,,1]

# Pick out the first three words and their embeddings
X <- transformer_example$embeddings[1:3,]

qkv(X, Wq, Wk, Wv)

## $Q
##             [,1]        [,2]       [,3]
## the    0.4722259  0.04995783 -0.5350845
## quick -0.3662435  0.12144160  0.3454785
## brown -0.1029677 -0.12728414  0.1817097
##
## $K
##                [,1]          [,2]       [,3]
## the    0.094360579 -0.203807092 -0.1851229
## quick -0.033313240  0.279012100  0.2530560
## brown -0.004457052  0.001013468  0.0133802
##
## $V
##                [,1]        [,2]        [,3]
## the    0.317318525 -0.35023010  0.13284078
## quick  0.009929565  0.04208206 -0.15412097
## brown -0.316413241  0.27717408  0.02725089
```

2. Now, based on your query, key and value, compute the attention of that given attention head for the three chosen tokens.

```
attention(res$Q, res$K, res$V)
## $Z
##                [,1]          [,2]         [,3]
```

```
## the    0.012395453 -0.0212420459  0.009404870
## quick -0.003759269 -0.0008360029 -0.005108890
## brown  0.002412222 -0.0088974612  0.001147999
##
## $attention
##              the     quick     brown
## the   0.3601932 0.3080896 0.3317172
## quick 0.3088780 0.3582373 0.3328847
## brown 0.3300375 0.3360583 0.3339042
```

3. Interpret the attention values. What does the second row mean?
4. Now we have everything in place for implementing it all as a multi-head attention layer. The layer will take in embeddings and then return a 3-dimensional embedding per word. Run your code on all words in the included example.

```
multi_head_self_attention(X,
                          transformer_example$Wq,
                          transformer_example$Wk,
                          transformer_example$Wv,
                          transformer_example$WO)
##               [,1]          [,2]         [,3]
## the   -0.014189613 -0.0040299008 -0.006756286
## quick -0.009963516 -0.0010724342 -0.001996524
## brown -0.006394562 -0.0006626115 -0.002219108
```

3 Taking BERT out for a spin (Optional assignment)

Unfortunately, there is no simple way to run BERT directly from R yet. There are two ways you can try to use BERT. There are two ways you can try BERT using Keras.

1. Using reticulate in R. Here is a tutorial.
2. Run BERT using Python in Colab here.

The second approach is probably more straightforward, and there you could try out running BERT using a GPU. In both cases, it is built upon Keras, and you will understand most parts.

The (optional) assignment is to run through the tutorial and summarize your preferred method's experience and thoughts on using BERT.