Uppsala University

Machine Learning

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**. 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**. Deadlines for all assignments are **Sunday 23.59**. See **Studium** for dates. Assignments will be graded within 10 working days from the assignment deadline.
- To pass the assignments, *you should answer all questions not marked with **, and get at least 75% correct.
- To get VG on the assignment, *all questions should be answered, including questions marked with a **, although minor errors are ok. VG will only be awarded on the first deadline of the assignment.
- 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**.
- You are not allowed to show your assignments (text or code) to anyone. Only discuss the assignments with your fellow students. The student that show their assignment to anyone else could also be considered to cheat. Similarly, on zoom labs, only screen share when you are in a separate zoom room with teaching assistants.
     ```

- The computer labs are for asking all types of questions. Do not hesitate to ask! The purpose of the computer labs are to improve your learning. We will hence focus on more computer labs and less on assignment feedback. *Warning!* There might be bugs in the assignments! Hence, it is important to ask questions early on so you dont waste time of unintentional bugs.
- 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

Keras, Tensorflow, and Google Colab

This assignment can be computationally heavy. If your own an old computer, you might want to run this task in the computer room in the Department or on Google Colab. Google Colab also enable GPU support if you want to try that.

We have setup a Jupyter Notebook with R and Tensorflow at MansMeg/IntroML/additional_material/colab_nb. We suggest *not* to run the code in `markdown`. Instead, run the code in R/Colab and copy the results as output (see the assignment template for an example).

As a first step, you must install `tensorflow` and `keras` on your local computer. You can find detailed information on how to install `tensorflow` and `keras` **[here]**. You can find details on different architecture components in the `keras` reference found **[here]**.

1 Convolutional neural networks using Keras

We are now going to implement a convolutional neural network using Keras. Here Ch. 5.1-5.3 in Chollet and Allaire (2018) and the following tutorial might be useful, especially for details on how to load the data. Remember to load the `tensorflow` R package before loading the `keras` R package.

If you have installed `keras`, you can load the MNIST dataset. This dataset contains data on handwritten digits that we want to classify. To access the data, we use `keras` as follows.

```
library(keras)
mnist <- dataset_mnist()
mnist$train$x <- mnist$train$x/255
mnist$test$x <- mnist$test$x/255
```

1. As a first step, we visualize a couple of digits. Include the first three you can find in the training dataset in the report as an image.

   ```
   idx <- 1
   im <- mnist$train$x[idx,,]
   # Transpose the image
   im <- t(apply(im, 2, rev))
   image(1:28, 1:28, im, col=gray((0:255)/255), xlab = "", ylab = "",
         xaxt='n', yaxt='n', main=paste(mnist$train$y[idx]))
   ```

2. Implement a Convolutional Neural Network for the MNIST dataset. The network should have two convolutional layers as follows.

   ```
   ## ________________________________________________________________________
   ## Layer (type)                        Output Shape                    Param #
   ## ========================================================================
   ## conv2d (Conv2D)                     (None, 26, 26, 32)              320
   ## ________________________________________________________________________
   ```

```
## max_pooling2d (MaxPooling2D)      (None, 13, 13, 32)          0
## _________________________________________________________________
## conv2d (Conv2D)                   (None, 11, 11, 32)          9248
## _________________________________________________________________
## flatten (Flatten)                 (None, 3872)                0
## _________________________________________________________________
## dense (Dense)                     (None, 64)                  247872
## _________________________________________________________________
## dense (Dense)                     (None, 10)                  650
## =================================================================
## Total params: 258,090
## Trainable params: 258,090
## Non-trainable params: 0
```

3. Explain why there are 320 parameters in the first layer. How many are kernel weights (and why), and how many biases?

4. Train the network using Keras. What is your loss and accuracy on the MNIST dataset?

5. Choose at least three hyper-parameters you would like to evaluate (Goodfellow et al. (see Table 11.1 in 2016)). Use random search Goodfellow et al. (see Ch. 11.4.4 in 2016) and train 10 different models with random hyper-parameter settings and store the validation accuracy of your network. Based on these result, choose the best hyper-parameter setting for your network. Print the code and the accuracy of your final model. You can choose the marginal distribution for the hyper-parameters yourself.

6. As the next step, we will implement a convolutional neural network for the CIFAR-10 dataset using `dataset_cifar10()`. See the tutorial for details on how to load data. Implement a similar CNN as in the tutorial. That is:

```
## Model: "sequential"
## _________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =================================================================
## conv2d (Conv2D)              (None, 30, 30, 32)        896
## _________________________________________________________________
## max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
## _________________________________________________________________
## conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
## _________________________________________________________________
## max_pooling2d_1 (MaxPooling2D) (None, 6, 6, 64)        0
## _________________________________________________________________
## conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
## _________________________________________________________________
## flatten (Flatten)            (None, 1024)              0
## _________________________________________________________________
## dense (Dense)                (None, 64)                65600
## _________________________________________________________________
## dense_1 (Dense)              (None, 10)                650
## =================================================================
```

```
## Total params: 122,570
## Trainable params: 122,570
## Non-trainable params: 0
## _________________________________________________________________
```

Note! This problem is more complex than the MNIST problem, so don't be surprised if you get lower accuracy than you got on the MNIST dataset.

7. Visualize one image of your choice from the CIFAR dataset together with its class/-category.

8. Why do we now have 896 parameters in the first convolutional layer?

9. Try out and compare at least three different networks for the CIFAR data. Describe the networks (include the keras model output), why you choose them, and the Keras model output. Goodfellow et al. (2016, Ch. 11-11.5) and (Chollet and Allaire, 2018, Ch. 5.3) might give some suggestions.

2 * Implementing a convolutional layer

To get VG on this assignment, you can choose to make this task OR the task on transfer learning below. You are also free to do both, if you want. As long as you pass one of the two VG assignments you will get a VG point. Although, please only write down the time it took to do the VG assignment for one of the tasks.

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 a first step, we will implement a layer of a convolutional neural network with one filter. We will here not train the layer, only implement it to understand the inner workings. You are not allowed to use `convolve()` in R in the final solution, but you can use it to debug your code if you want to.

Some good material for this exercise is Figure 9.1 in Goodfellow et al. (2016) and the video Ng (2017).

Start by importing examples from the MNIST dataset as follows.

```
library(uuml)
data("mnist_example")
```

To visualize an image use:

```
im <- mnist_example[["5"]]
image(1:ncol(im), 1:nrow(im), im,
      xlab = "", ylab = "",
      xaxt='n', yaxt='n', main="4")
```

1. Visualize the MNIST example digit 4.
2. Implement a convolution function called `convolution(X, K)` that takes an input MNIST image (`X`), an arbitrary large square kernel (`K`) and returns a valid feature map. Below is an example of how it should work.

```
X <- mnist_example[["4"]][12:15,12:15]
X
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   56  250  116    0
## [2,]    0  240  144    0
## [3,]    0  198  150    0
## [4,]    0  143  241    0
```

```
K <- matrix(c(-0.5, -0.5, -0.5,
              1,  1,  1,
              -0.5, -0.5, -0.5), nrow = 3, byrow = TRUE)
K
```

```
##      [,1] [,2] [,3]
## [1,] -0.5 -0.5 -0.5
## [2,]  1.0  1.0  1.0
## [3,] -0.5 -0.5 -0.5
```

```
convolution(X, K)
```

```
##      [,1] [,2]
## [1,]   -1   27
## [2,]  -36  -36
```

3. Visualize the feature map of MNIST example digit 4 using the above two by two kernel `K`.

4. Now implement all steps in a `convolutional_layer(X, K, b, activation)` function that takes the kernel, bias and activation function. It should work as follows.

```
relu <- function(x) max(0, x)
convolutional_layer(X, K, 100, relu)
```

```
##      [,1] [,2]
## [1,]   99  127
## [2,]   64   64
```

5. Run your convolutional layer on MNIST example digit 4 with bias -150. Visualize the feature map as you visualize the original image. What does the filter seem to capture?

6. Now transpose your filter and run your convolutional layer on MNIST example digit 4 with bias -150. Visualize the feature map. What does that transposed filter seem to capture?

7. As the last step in our convolutional layer, implement a two by two, two stride max-pooling layer. It should work as follows.

```
X <- mnist_example[["4"]][12:15,12:15]
maxpool_layer(X)
```

```
##      [,1] [,2]
## [1,]  250  144
## [2,]  198  241
```

8. Now put it all together and visualize the final output of your own convolutional layer. Visualize the feature map.

```
X <- mnist_example[["4"]]
relu <- function(x) max(0, x)
output <- maxpool_layer(convolutional_layer(X, K, -370, relu))
```

3 * Transfer learning using VGG16

To get VG on this assignment, you can choose to make this task OR the task on implementing a convolutional layer above. You are also free to do both, if you want. As long as you pass one of the two VG assignments you will get a VG point. Although, please only write down the time it took to do the VG assignment for one of the tasks.

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.

Note! This task can be a computationally heavy.

As the last step, we will look into using transfer learning as a quick way of improving the prediction accuracy on the cifar-10 dataset. Here Ch. 5.3 in Chollet and Allaire (2018) or this tutorial might be helpful.

You can find the current state-of-the-art neural networks for the cifar-10 dataset here. Feel free to read some of the papers and be inspired on how to improve your neural network.

1. Using Keras, download the VGG16 convolutional neural network convolutional base. Just download the convolutional base. We are going to use the network on the cifar-10 dataset so change the `input_size` to `c(32, 32, 3)`.

2. Now add a dense layer with 64 hidden nodes (as in the previous exercise). Include the Keras model output in the assignment report. How many parameters do you have in the dense top layer (compared to the CNN in the previous part)?

3. Now, freeze the convolutional base and train the dense top layer of your network on the cifar-10 dataset. Run it for five epochs. *Note!* This will take time. Don't be surprised if you need roughly 200s per epoch or more. Also, remember to freeze the convolutional base!

4. Report your final accuracy. How does this compare to the previous model for the cifar data?

References

François Chollet and Joseph J Allaire. *Deep Learning with R*. Manning, 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Andrew Ng. C4w1l07 one layer of a convolutional net, 2017. URL https://www.youtube.com/watch?v=jPOAS7uCODQ&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF&index=8.