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**. 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.
- 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 a convolutional kernel/feature map? (max 1 paragraph)

2. Reason about when to gather more data in an ML application? (1-2 paragraphs)

3. Describe a good debugging test for an ML algorithm. (max 1 paragraph)

4. Describe why we primarily focus on fine-tuning layers higher up in neural networks when fine-tuning large pre-trained neural networks. (1-2 paragraphs)

.

2 Implementing a convolutional layer

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(1, 1, 0, 0), nrow = 2)
K
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    1    0
```

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

```
##      [,1] [,2] [,3]
## [1,]   56  490  260
## [2,]    0  438  294
## [3,]    0  341  391
```

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)
X <- mnist_example[["4"]][12:15,12:15]
K <- matrix(c(1, 1, 1,
              0, 0, 0,
              0, 0, 0), nrow = 3, byrow = TRUE)
convolutional_layer(X, K, -370, relu)
```

```
##      [,1] [,2]
## [1,]   52    0
## [2,]   14   14
```

5. Run your convolutional layer on MNIST example digit 4 with bias -400. 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 -450. 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.

```r
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.

```r
X <- mnist_example[["4"]]
relu <- function(x) max(0, x)
K <- matrix(c(1, 1, 1,
              0, 0, 0,
              0, 0, 0), nrow = 3, byrow = TRUE)
output <- maxpool_layer(convolutional_layer(X, K, -370, relu))
```

3 Convolutional neural networks using Keras

Note! This assignment can be a little heavy computationally. If your own an old computer, you might want to run this task in the computer room in the Department.

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 `tesorflow` R package before loading the `keras` R package.

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. 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
```

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

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

4. As the next step, we will implement a similar 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.

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

6. Try out at least three different networks, describe the networks, why you chose them, and the Keras model output. Reading Ch. 11-11.5 in Goodfellow et al. (2016) might give some good inspiration and guidance. For example, you can add a `dropout_layer` (see Chollet and Allaire, 2018, , Ch. 5.3).

7. Can you improve over the previous network concerning accuracy? You can also use regularization techniques and batch normalization from the previous lab.

4 * Transfer learning using VGG16

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 assignment can be a little heavy computationally. If your own an old computer, you might want to run this task in the computer room in the Department.

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.

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 networks.

1. Using Keras, download the VGG16 convolutional neural network. 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.
4. Report your final accuracy. How does this compare to the previous model for the cifar data?

References

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.

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