Homework 01

IANNWTF 21/22

no submission deadline

This week, you do not have to hand in anything. But you should make sure you have everything set up on your computer to get started. And also make sure to have everything in your brain to get started: we will give you some hints and tasks to recap programming and maths skills.

1 Setup your computer

Try to first do the setup on your own, we will try and give you a good guideline. But in doubt, drop by at our Coding Support if you need help or more information.

1.1 Installing Anaconda

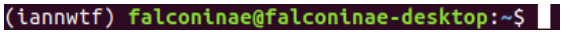
For this course we are using the python distribution Anaconda. Follow the installation instructions for Anaconda for your respective operating system under https://docs.anaconda.com/anaconda/install/

1.2 Setting up a virtual environment

Next, we want to set up a virtual environment with anaconda. We talk you through everything in detail below (while explaining it, so you are able to recreate and fix anything as necessary later), but if you encounter anything surprising, or just want to educate yourself more on this, check out this guide.

Think of a virtual environment as a airtight container on your computer, which makes sure nothing from the inside can come to the outside: Specifically (python) packages installed in a container are not installed on your whole machine, but only inside this container (and the other way around!). This is really important, because different projects you might be working on might

- require different python packages, which are not compatible with each other
- might require different versions of the same package (which you can't have installed at the same time
- he container and your computer might even have different python versions installed



Activated 'iannwtf' environment

We now create such an environment with the following command (in bash/your command line[1]):

```
$ conda create -n iannwtf tensorflow
```

Let's quickly unpack what this does:

1. `conda create`
   is the command for creating such an virtual environment with anaconda

2. `-n iannwtf`
   sets the name flag: -n or equally
   `--name`
   is the flag and the subsequent element will be treated as the parameter, in the case of this flag the name parameter, which simply sets the name for our virtual conda environment

3. `tensorflow`
   will already install tensorflow and all the dependencies needed, including the correct python version [2]

So now that you have created a virtual environment, we have to activate it (sort of entering the container - indeed you will have to enter it every time you want to use it, e.g. when you are working on your homework). Run the conda activate command for this:

```
$ conda activate iannwtf
```

If you named your virtual environment different than iannwtf, of course you will have to use this respective name here. If the command did run successfully, in your command line you should now see this very clearly, as the line should now begin with

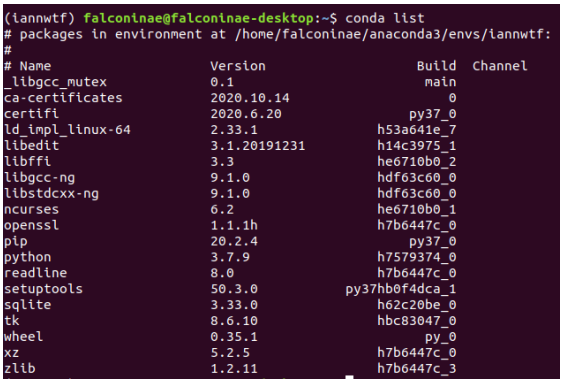
```
(iannwtf)$
```

to remind you that your virtual environment is activated.

Although, tensorflow should now be installed (you can check all installed packages by typing *conda list*), you might want to install some additional packages (e.g. matplotlib for plotting). To install packages there are two good ways: onda install comes with anaconda and most of the time get's the job done - but here we want to stick to

[1] We will refer to this as command line, in ubuntu this is typically bash (else you probably know what you are doing anyways and don't really need our help setting up a virtual env), but Windows cmd and mac Terminal pretty much do the same thing here - and for typical users they are probably better known under the name command line.

[2] If you are fancy enough to have a computer with a cuda-enabled graphics card, you can also use
`tensorflow-gpu`
to directly install tensorflow with GPU support.



Is pip installed?

```
$ pip install
```

because typically pip is better kept up to date than conda for package updates. So first of all we want to install pip by running:

```
(iannwtf)$ conda install pip
```

Make sure to do this inside your virtual environment (which should be indicated by having it's name in brackets prepended to your cursor in the command line). *conda install pip* should have installed the package installer pip **inside** your virtual environment. Before we use pip, we want to make sure that it is indeed installed in your virtual env, again you can check by viewing the output of

```
(iannwtf)$ conda list
```

Before you advance, please confirm that this list (should still be rather empty) contains pip now. We need to make sure of this, because when subsequently we use pip to install further packages, if we do not have pip in our virtual conda env, this will use your system's pip to install python packages into your user, instead of the virtual env's pip to install packages into your virtual env. Now having confirmed this, we use pip to update pip:

```
(iannwtf)$ pip install --upgrade pip
```

If you run

```
(iannwtf)$ conda list
```

again now, you should see your pip updated to a really recent version, 20.x.x (20.2.4 probably). Finally, we simply have to use pip to install the packages we need for this course:

```
(iannwtf)$ pip install matplotlib
```

This should now have the plotting library matplotlib, but also any other packages those depend on (like the very important numpy). Quickly confirm this via

```
(iannwtf)$ conda list
```

again.

1.3 Test Tensorflow

Run the test_tensorflow.py file in your environment to check if your tensorflow installation works. For this you can either use the code editor of your choice or navigate to the folder where the test file is stored, using the terminal and then execute the script with the command

```
(iannwtf)$ python test_tensorflow.py
```

If you do not get any errors tensorflow works as intended.

2 Jupyter Notebook and Google Colab

For your homework, you will have the choice between using python scripts (.py files) or jupyter notebooks (.ipynb files). Code in notebooks is organized in cells, which can be executed independently. They also allow for markdown cells, which are a convenient way to add explanations. Jupyter notebook is automatically installed with Anaconda. Just activate your environment and type *jupyter notebook* into the console to start a notebook sever. If you are not familiar with jupyter notebooks you can check out this guide.

Google Colab is a browser tool for notebooks with free GPU computing. You do not have to use it if you don't want to support Google but if your laptop or computer is slow it might be a good idea. If you prefer python scripts over notebooks you can also import your scripts into Colab to have access to the GPU. The major drawback is that you have to be online to access Colab. Check out this notebook for an overview of the basic features.

In general, we recommend trying to script in python script and modules for good practice. Also, if you have many classes and predefined functions, you will soon notice that jupyter notebooks can be a pain in the ass. However, due to easy markdown cells and the interactiveness, jupyter notebooks are very powerful when it comes to visualization / playing with results etc. We thus recommend sticking to python modules for the actual coding and using notebooks for the interactive parts. For the homework however, we will accept everything - you might just not get an 'outstanding' ;).

3 Test yourself!

This part of the homework is a **voluntary self test** for your basic coding and maths skills. We generally assume you have some basic python coding skills and some basic high school level understanding of maths, and here want to make sure nothing we use anywhere seems like black magic to you.

IF you notice that a read-up is good for you, check out the section on **Prepare Yourself** in **Lecture 0** on Courseware - we provide you with some videos and links

that could prove useful to you. Pay special attention to Numpy - it will come in very handyK

3.1 Python modules

This first task asks you to write two distinct python "modules" to check whether you are familiar with both (1) python objects and python modules. If you are new to python: A python module is (a bit simplified) just a python file, i.e. a file that has the ending ".py". Create two such files named "cat.py" and "kittyconcert.py".

In "cat.py" define a cat class. This class should have:

- *nitializer* (think constructor), where you can also name your cat
- *Method: greet* (think function bound to a class), where the cat introduces herself with her own name, and then greets another cat by that cat's name, e.g.:

  "Hello I am Kittosauros Rex! I see you are also a cool fluffy kitty Snowball IX, let's together purr at the human, so that they shall give us food." [3]

In "kittyconcert.py" just simply import "cat.py", create two different named cats, and let them both greet each other.

3.2 List comprehension

In this very quick task, we want to make sure you are familiar with list comprehension. This clever concept lets you define lists on the fly in python, but might look a bit terrifying for new python users at first glance. Your task is to define a list with squares of all numbers from 0 to 100 **in just one line**.

If you are unfamiliar with the concept, google "python list comprehension".

Bonus: Only include those squares that are divisible by 2, but still keep it in one line.

3.3 Generators

Generators are an important tool for implementing Deep Learning training procedures in python, because they can be used to iterate over data without cluttering your computer's memory too much. Your task is to implement a generator that on each call returns a string with multiple "meows" in it, specifically always twice as many as in the last call, e.g.:

1. "Meow"
2. "Meow Meow"
3. "Meow Meow Meow Meow"
4. "Meow Meow Meow Meow Meow Meow Meow Meow"

[3] Tip: this method should take an additional argument (the name of the other cat

5. etc.

You may define the generator in a function and just directly print out the first 10 calls to it in a loop below.

If you are unfamiliar with the concept, google "python yield" and "python generator" to get familiar with the concept.

3.4 Numpy and slicing

Numpy is an awesome library for matrix, arrays and generally math based applications in python.

Your task here is to

1. create a 5x5 numpy array with normal distributed random numbers ($\mu = 0$, $\sigma = 1$)
2. replace all entries of this array with the original entry squared only if the original entry larger than 0.09 with the number 42
3. print the fourth column of this array

[4]

3.5 Derivatives

Consider the following function:

$$f(x, z, a, b) := y = (4ax^2 + a) + 3 + \sigma(z) + (\sigma(b)^2)$$

σ here is the sigmoid-function - feel free to just google either it's definition and work yourself towards its derivative (we will need this function and its derivative next week, so putting in this work might not be in vain), or simply also google its derivative.

Calculate the derivative of this function for all of x,z,a,b, specifically:

- $\frac{\partial y}{\partial x}$
- $\frac{\partial y}{\partial z}$
- $\frac{\partial y}{\partial a}$
- $\frac{\partial y}{\partial b}$

There is one additional piece of math you might want to look up for next week, if you are super motivated: Learn about the "Del" "Nabla" operators and the idea of a gradient on your own (google and youtube are your friends). With the solutions from above that should enable you to calculate the gradient ∇f w.r.t $[x, z, a, b]$ quite quickly.

[4] If anything here troubles you, just google

1. numpy random
2. numpy boolean array indexing
3. numpy slicing