Homework 01

IANNWTF 20/21

27th October 2020

This week you do not have to hand in anything. But you will have to make sure you have everything set up on your computer to get started. Below there are also some recap programming and maths tasks.

1 Setup

We will also go through the setup quickly in the session this Thursday. If you still have problems making it work come into the coding sessions on Friday or Tuesday.

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

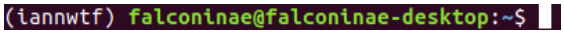
Next we want to set up a virtual environment with anaconda. We talk you through everything in detail below (while explaining it, so you are able to recreate and fix anything as necessary later), but if you encounter anything surprising, or just want to educate yourself more on this, check out this guide. 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 (1) require different python packages, which are not compatible with each other, (2) might require different versions of the same package (which you can't have installed at the same time), (3) the container and your computer might even have different python versions installed. We now create such an environment with the following command (in bash/your command line[1]):

[1] We will refer to this as command line, in ubuntu this is typically bash (else you probably now what you are doing anyways and don't really need our help setting up a virtual env), but Windows cmd

conda create -n iannwtf python=3.7
Let's quickly unpack what this does:

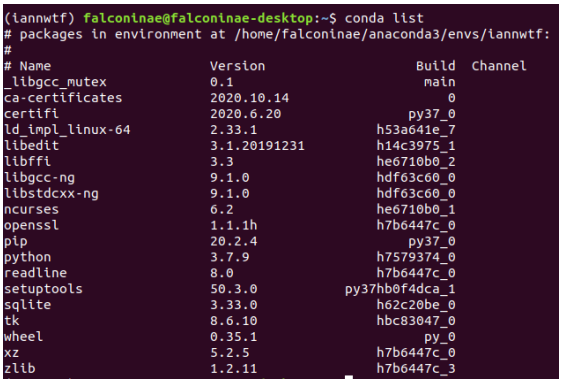
- *conda create* is the command for creating such an virtual environment with anaconda
- *-n iannwtf* sets the name flag: *-n* or equally *–name* is the flag and the subsequent element will be treated as the parameter, in the case of this flag the name parameter, which simply sets the name for our virtual conda environment
- *python* = 3.7 makes sure our conda environment is created with python version 3.7, which we are going to use for this course

So now that you have created a virtual environment, we have to turn it on (and indeed you will have to turn it on 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.



Now we want to install some python packages (e.g. tensorflow), so please make sure that for the subsequent steps indeed your virtual environment is activated (that means you should see the name of your environment in brackets to the left in your bash/command line). To install packages there are two good ways: *conda install* comes with anaconda and most of the time get's the job done - but here we want to stick to *pipinstall* 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:
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, so we run:
conda list
to get a list of all installed packages in our virtual environment.

and mac Terminal pretty much do the same thing here - and for typical users they are probably better known under the name command line



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:

pip install –upgrade pip

If you run *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:

pip install tensorflow

pip install matplotlib

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

We will also have a short introduction to notebooks and colab in the session this thursday.

3 Test yourself!

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

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 "kitty.py" and "kittyconcert.py".

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

- Initializer (think constructor), where you also can 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 they shall give us food."*

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

To import "cat.py" in "kittyconcert.py" you will have to create a third file named "__init__.py". If you are unfamiliar with this, just google it quickly.

3.2 List comprehension

Done!

In this very quick task we want to make sure you are familiar with list comprehension. This clever concept let's 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"
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, tensor 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) Next replace all entries, where the squared entry is larger than 0.1 with the number 42.
(3) Finally print the 3rd column of this numpy array.

If anything here troubles you, just google (1) "numpy random", (2) "numpy boolean array indexing" and (3) "numpy slicing".

3.5 Derivatives

Consider the following function:

$$f(x, z, a, b) := y = (2ax^2 + a) + 5 + \sigma(z) + (\sigma(b)^2)$$

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

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

- $\frac{\delta y}{\delta x}$

- $\frac{\delta y}{\delta z}$
- $\frac{\delta y}{\delta a}$
- $\frac{\delta y}{\delta b}$

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.