University of Cape Town
Department of Computer Science

CSC3022F
ML Assignment 3 - Artificial Neural Networks

June, 2021

In this Assignment, you are given two tasks. The first involves solving the XOR problem using only Perceptrons. The second task has you designing an ANN that can classify handwritten digits from the MNIST10 dataset. This assignment is designed to test your ability to design Neural Networks capable of solving different problems. As such, you are allowed to use *Pytorch* and a *Perceptron.py* base class has been provided for you (from lectures). You are free to modify this file as you please.

You will also be asked to write a report detailing the designs of your two solutions. This **MUST** be a pdf that you submit along with your code. (See below for more info)

1 Part 1: The XOR Problem

As we know from lectures, Single Perceptrons are only capable of solving linearly separable problems. The XOR Gate (See table 1) is not linearly separable but can be implemented by chaining together individual Perceptrons.

Your task is to solve the XOR problem using only Perceptrons that implement OR, AND or NOT gates. These Perceptrons must use a threshold activation function and the final network must be noise tolerant. As in lectures, any input > 0.75 should be treated as an on signal and your network should be able to detect that (See table 2).

Your XOR gate solution should be implemented in a file called *XOR.py* and should be invoked as follows:

```
python XOR.py
```

x1	x2	XOR
0	0	0
1	0	1
0	1	1
1	1	0

Table 1: Truth table for the XOR gate.

x1	x2	XOR
0.2	-0.15	0
0.76	0	1
0.1	0.86	1
1.1	0.83	0

Table 2: XOR gate with noisy inputs.

When a user executes this code, it should train your perceptrons using training sets that you have generated yourself, construct your network and, allow a user to enter two inputs as follows:

```
> python XOR.py
Training GATE_0...
Training GATE_1...
...
Training GATE_N...
Constructing Network...
Done!
Please enter two inputs:
> 0.2 -0.1
XOR Gate: 0
Please enter two inputs:
> 1.0 0.5
XOR Gate: 1
Please enter two inputs:
> exit
Exiting...
```

Your training data can be generated before file execution and stored in a data file or you can generate it at runtime.

Given that we are giving you a lot freedom when designing the Network, you must write a brief report detailing the following:

- The topology of your network
- The training data used to train the Perceptrons, how you generated said data and why you used that method to generate the training examples.
- How different learning rates affect the accuracy of your Network (if at all).

All of this information should be included in Part 1 of your Design Doc pdf that you will submit alongside your code.

Completing this part of the assigned work will earn you up to 50% for the Assignment.

2 Part 2: Image Classification

For Part 2 of the Assignment, you must design an ANN that is capable of classifying hand written digits from the MNIST10 dataset. The dataset has been provided for you under Assignment Resources so you **may NOT download the dataset yourself**. You will be penalized if you do this. **Do not submit the dataset either,** the tutors will have access to it.

You are free to use *Pytorch* and *Torchvision* for this Part of the Assignment. This includes all of the different activation functions, loss functions and the built-in gradient descent and backpropagation functionality. The main goal of this assignment is to get you to design a neural network that can successfully (to some degree of accuracy) classify handwritten digits.

Your ANN should be implemented in a file called *Classifier.py* and should be invoked as follows:

```
python Classifier.py
```

When a user executes this code, it should build your network, define any necessary functions, preprocess your data (if needed), train your network and, once training has completed, allow a user to enter a path to an image from the MNIST10 dataset:

```
> python Classifier.py
Pytorch Output...
...
Done!
Please enter a filepath:
> ./path/to/file_1
Classifier: 1
Please enter a filepath:
> ./path/to/file_8
Classifier: 8
Please enter a filepath:
> exit
Exiting...
```

Given that this is a commonly solved problem in Machine Learning, a number of valid solutions exist. The bulk of your marks for this section will come from your design document (The same one mentioned in Part 1) where you will need to describe (with sufficient reasoning / motivation):

- The topology of your network (Including the number of layers, activation functions and connectedness of said layers)
- Any preprocessing steps performed on the training set.
- The loss function used.

- Any optimizers that were used.
- How the network is trained / validated
- Any other miscellaneous details that you feel are relevant to your ANN.

It is worth noting that simply saying you used a ReLU activation function or that your loss function was the Cross Entropy Loss function will not get you marks. You must motivate all of your design decisions and demonstrate that you understand your own solution.

Completing this part of the assigned work will earn you up to 50% for the Assignment.

In a compressed archive named as your student number (e.g. GWRBRA001.zip/tar.gz/tar.xz), place your Python scripts, Design Document, makefile, git repo, readme, and any other applicable files. Upload that archive to the assignments tab on **Vula before 6.00 PM, 16 June, 2021**

Please Note the following for ALL ML Assignments:

1. A working Makefile must be submitted. If the tutor cannot build a python virtual environment on nightmare.cs by typing make, you will only receive **50%** of your final mark.

2. You must use version control from the get-go. This means that there must be a .git folder alongside the code in your project folder. A **10%** penalty will apply should you fail to include a local repository in your submission.

   With regards to git usage, please note the following:

   -10% - usage of git is absent. This refers to both the absence of a git repo and undeniable evidence that the student used git as a last minute attempt to avoid being penalized.

   -5% - Commit messages are meaningless or lack descriptive clarity. eg: "First", "Second", "Histogram" and "fixed bug" are examples of bad commit messages. A student who is found to have violated this requirement for numerous commits will receive this penalty.

   -5% - frequency of commits. Git practices advocate for frequent commits that are small in scope. Students should ideally be committing their work after a single feature has been added, removed or modified. Tutors will look at the contents of each commit to determine whether this penalty is applicable. A student who commits seemingly unrelated work in large batches on two or more occasions will receive this penalty.

   Please note that all of the git related penalties are cumulative and are capped at -10% (ie: You may not receive more than -10% for git related penalties). The assignment brief has been updated to reflect this new information.

   We cannot provide a definitive number of commits that determine whether or not your git usage is appropriate. It is entirely solution dependent and needs to be accessed on an individual level. All we are looking for is that a student has actually taken the time to think about what actually constitutes a feature in the context of their solution and applied git best practices accordingly.

3. You must provide a README file explaining what each file submitted does and how it fits into the program as a whole. The README file should **not** explain any theory that you have used. The README is used by the tutors if they encounter any problems.

4. Do **not** hand in any binary files. Do **not** add binaries (.o files and your executable) to your local repository.

5. Please ensure that your tarball works and is not corrupt (you can check this by trying to downloading your submission and extracting the contents of your tarball - make this a habit!). Corrupt or non-working tarballs will not be marked - **no exceptions.**

6. A 10% penalty per day will be incurred for all late submissions. No hand-ins will be accepted if later than 3 days.

7. **DO NOT COPY. All code submitted must be your own.** ***Copying is punishable by 0 and can cause a blotch on your academic record.*** **Scripts will be used to check that code submitted is unique.**

8. You are **NOT** allowed to simply download python packages to solve your coding problems. Unless specifically stated in the Assignment brief, we expect all of your code to be your own.