Griffith UNIVERSITY
Queensland, Australia

Assignment 2

2802ICT Intelligent System
School of ICT, Griffith University
Trimester 1, 2021

Assignment 2 (50%)
Due on 11:59pm Monday 31 May 2021

with demonstrations to be held on Week 12

Instructions:

- **Due:** 11:59pm, 31 May 2021
- **Marks:** 50% of your overall grade
- **Late submission:** Late submission is allowed but a penalty applies. The penalty is defined as the reduction of the mark allocated to the assessment item by 5% of the total weighted mark for the assessment item, for each working day that the item is late. A working day is defined as Monday to Friday. Assessment items submitted more than five working days after the due date will be awarded zero marks.
- **Extensions:** You can request for an extension of time on one of two grounds:
  - Medical
  - other (e.g. special family or personal circumstances, unavoidable commitments).

  All requests must be made through the myGriffith portal.
- **Individual Work:** You must complete this assignment individually. You are encouraged to discuss the assignment with your fellow students, but eventually this should be your own work. Anyone caught plagiarising will be penalised and reported to the university.
- **Presentation:** You must present/demonstrate your work on Week 6 to a teaching team member. Your work will not be marked if you do not present it. Indeed, the final marks will be based on the interview results as follows:

finalMark = interviewMark * submissionMark + oralExaminationMark

Note: **submissionMark** is your mark of the submitted code and report. **interviewMark** is out of 1 based on how well the code and report are explained. **OralExaminationMark** is your mark of your answers to the open questions you'll be asked during the interview.

- **Objectives:**

  ✓ Implement learning algorithms

  ✓ Evaluate the performance of a simple learning system on a real-world dataset
- **Overview:**

In this assignment you will be required to study the properties of supervised learning algorithms as well as compare their prediction accuracy with the provided datasets.
The assignment includes 3 tasks and algorithms you will be implementing are **K-Nearest-Neighbours, Decision Tree** and **Neural Networks**.

Your code should be written in Python. If you wish to use another programming language please contact Liat for an approval. Note that - you are **not allowed** to use libraries such as sklearn, pytorch, tensorflow to implement the algorithms. You are allowed to use libraries for data structures, math functions and the libraries mentioned in the tasks description below.

- **Maximum marks are:**
- **10** for **Task 1**
- **10** for **Task 2**
- **20** for **Task 3**
- **10** for **open questions**

Total: 50 marks

Task 1 - k Nearest Neighbours

Implementation Requirements:

a. Implement the K-Nearest-Neighbours algorithm. Your code should include at least the following functions:
   1. Read the **iris.csv** dataset, which includes description of three types of the Iris flower (Setosa, Versicolor, and Virginica) using four features describing their sepals and petals (their length and width in cm).
   2. **split_data** function**:** takes a percentage value as a parameter, which represents the relative size of the testing set. The function should randomly split the dataset into two groups: testing and training.
   3. For example, if the dataset includes 100 data items, then the function call `split_data(0.3)` should return two groups of data items: one that includes 70 random selected items for training, and the other includes the other 30 items for testing.
      Note: You may use the Python function random_sample to split the data set.
   4. **euclidean_distance** function: measures the distance between two flowers based on their features.
   5. **kNN** function: takes a training set, a single flower and an integer k, and returns the k nearest neighbours of the flower in the training set.
   6. A classification function that finds the type of the flower. Your function should return the type based on the majority of its k nearest neighbours.
   7. A function that returns the prediction **accuracy**, i.e. the **percentage** of the flowers in the test set that were correctly identified.

b. The output of your program should include:
   1. For each **sample** in each group (training and testing) print its real type, the classifier prediction and whether the prediction was correct (true/false). For each **group** print the prediction accuracy.

   For example:

```
…
sample class = Iris-versicolor, prediction class = Iris-versicolor, prediction correct: True
sample class = Iris-versicolor, prediction class = Iris-virginica,  prediction correct: False
…
Training set accuracy: 90.47619047619048 %
```

c. Run your algorithm using different k values and find the optimal k for the given dataset.

d. Plot a graph that shows the **accuracy** of both sets (training and testing) in respect to k.
   Note: To make plots, you can use the Python library matplotlib.

e. Try to use a different distance function (replacing the euclidean_distance from (4.) above). Does it change the results? In what why? (Improve or worsen the accuracy). The results should be included in the report.

Report Requirements:

Your report should include:

a. Software design: information about each function and data structure.

b. Results:
   - You should add the accuracy graph from (d.) above to your report and explain which part of the graph indicates of overfitting and which on underfitting and why.
   - Also mention what is the optimal k and any other interesting things you have learned.
   - Present and explain the distance function you tested in (e.) and the corresponding results.

c. Conclusions.

Task 2 - Decision Tree

Implementation Requirements:

a. Read the **votes.csv** dataset, which includes a list of US Congress members, the name of their party and features that represent their voting patterns on various issues.

b. Use the data from (a) to build a decision tree for predicting party from voting patterns. Your implementation should include:
   1. **split_data** function which works in a similar way to Task 2.
   2. Build a decision tree from the training set according to the algorithm learned in the lectures.

c. **Run** the kNN algorithm from Task1 on the votes dataset and compare the two classifiers:
   - **kNN** with optimal k
   - **Decision Tree**

   For each classifier calculate and **print**:
   - **precision**, **recall** and **F1** values
   - Plot the **Learning curve** (accuracy as a factor of percentage of learning example) i.e. show how the accuracy changes while learning.

Report Requirements:

Your report should include:

a. Software design: information about each function and data structure.

b. The results and graphs from (c.) above.

c. Conclusions.

Task 3 - Neural Networks

In this task, you are to implement a popular machine learning algorithm, Neural Networks. You will train and test a simple neural network with the datasets provided to you and experiment with different settings of hyper parameters.

The dataset provided to you is of Zalando's research, and consists of images of fashion items. A sample of the dataset is given below:

The training and test datasets are provided to you together with this assignment in the following files:

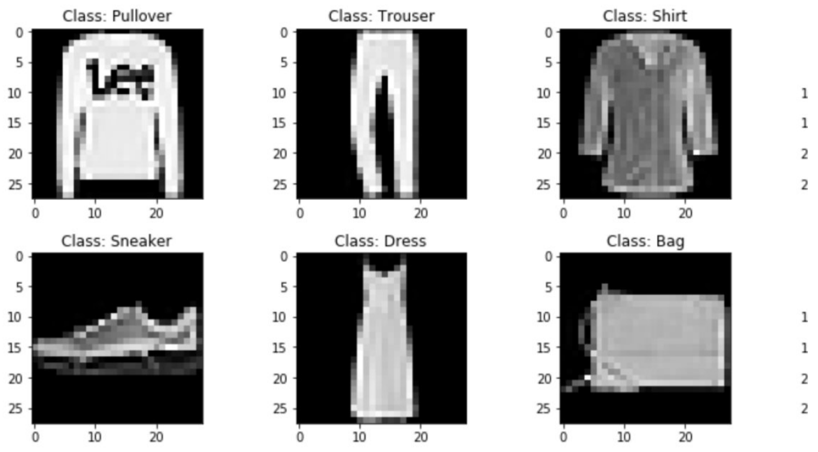
Training examples:	`fashion-mnist_train.csv.gz`
Test examples:	`fashion-mnist_test.csv.gz`

Data
There are 60,000 training examples and 10,000 test examples. Each example is a grayscale image, associated with a label from 10 classes.
Each image is 28x28 pixels, i.e consists of 784 pixels in total. Each pixel has a grayscale value represented as an integer between 0 and 255, with 0 indicates white and 255 indicates black.
The training and test data sets have 785 columns. The first column consists of the class labels (see below), and represents the article of clothing. The rest of the 784 columns contain the pixel-values of the associated image.

Labels
Each training and test example is assigned to one of the following labels:

0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

An example of some samples with there classes can be seen below:



Implementation requirements

Your task is to implement a neural network learning algorithm that creates a neural network classifier based on the given training dataset. Your network has 3 layers: an input layer, one hidden layer and an output layer.

You should name your main file as **`nn.py`** which accepts **five arguments**. Your code should be run on the command line in the following manner:

```
> python nn.py NInput NHidden NOutput fashion-mnist_train.csv.gz fashion-
mnist_test.csv.gz
```

Where the meanings of each argument are:

`NInput`:	number of neurons in the input layer
`NHidden`:	number of neurons in the hidden layer
`NOutput`:	number of neurons in the output layer
fashion-mnist_train.csv.gz:	the training set
fashion-mnist_test.csv.gz:	the test set

You should set the default value of number of **epochs** to 30, size of **mini-batches** to 20, and **learning rate** to 3, respectively.

Note: you can use numpy.loadtxt() and numpy.savetxt() methods to read from or write to a csv.gz. There is no need to un-compress the file before use. To making plots, you can use the Python library matplotlib.

The **nonlinearity** used in your neural net should be the basic sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

The main steps of training a neural net using stochastic gradient descent are:

1. Assign random initial weights and biases to the neurons. Each initial weight or bias is a random floating-point number drawn from the standard normal distribution (mean 0 and variance 1).
2. For each training example in a mini-batch, use backpropagation to calculate a gradient estimate, which consists of following steps:
    a. Feed forward the input to get the activations of the output layer.
    b. Calculate derivatives of the cost function for that input with respect to the activations of the output layer.
    c. Calculate the errors for all the weights and biases of the neurons using backpropagation.
3. Update weights (and biases) using stochastic gradient descent:

$$w \to w - \frac{\eta}{m} \sum_{i=1}^{m} error_i^w$$

where m is the number of training examples in a mini-batch, $error_i^w$ is the error of weight w for input i, and η is the learning rate.

4. Repeat this for all mini-batches. Repeat the whole process for specified number of epochs. At the end of each epoch evaluate the network on the test data and display its accuracy.

For this part of the assignment, use the **quadratic cost function**:

$$C(w, b) = \frac{1}{2n} \sum_{i=1}^{n} ||f(x_i) - y_i||^2$$

where

w: weights

b: biases

n: number of test instances

x_i: i^th test instance vector

y_i: i^th test label vector, i.e. if the label for x_i is 8, then y_i will be [0,0,0,0,0,0,0,0,1,0] (see below)

$f(x)$: Label predicted by the neural network for an input x

For this fashion images recognition assignment, we will encode the output (a number between 0 and 9) by using **10 output neurons**. The neuron with the highest activation will be taken as the prediction of the network. So the output number y has to be represented as a vector of 10 binary digits, all of them being 0 except for the entry at the correct digit.

You should do the following:

1. Create a neural network of size [784, 30, 10]. This network has three layers: 784 neurons in the input layer, 30 neurons in the hidden layer, and 10 neurons in the output layer. Then train it on the training data with the following settings: epoch = 30, minibatch size = 20, η = 3.

   Test your code on the test data and make plots of test accuracy vs epoch. Report the maximum accuracy achieved.

2. Train new neural networks with the same settings as in (1.) above but with different learning rates η = 0.001, 0.1, 1.0, 10, 100. Plot test accuracy vs epoch for each η on the same graph. Report the maximum test accuracy achieved for each η. Remember to create a new neural net each time so its starts learning from scratch.

3. Train new neural nets with the same settings as in (1.) above but with different mini-batch sizes = 1, 5, 10, 20, 100. Plot maximum test accuracy vs mini-batch size. Which one achieves the maximum test accuracy? Which one is the slowest?

4. Try different hyper-parameter settings (number of epochs, η, and mini-batch size, etc.). Report the maximum test accuracy you achieved and the settings of all the hyper parameters.

5. An alternate cost function: Now replace the quadratic cost function by a cross entropy cost function:

$$C(w,b) = -\frac{1}{n}\sum_{i=1}^{n} y_i \ln\left[f(x_i)\right] + (1-y_i)\ln[1-f(x_i)]$$

   Train a neural network as before and try different hyper-parameter settings. What is the test maximum accuracy achieved?

Report requirements

Your report should include:

a. Software design: information about each function and data structure.
b. All experimental results and graphs mentioned above, and detailed explanations of these results. i.e. try to explain the logic behind the results you achieved.
c. Conclusions.

What to hand in

You should hand in **three zip files**, one for each task. Each zip file should include:

- well commented python script
- a report
- a text file which includes all your python code (just copy-paste your code to a single .txt file).

Submission should be done through the link provided in L@G by the due date.

Marking scheme

Please refer to the marks mentioned above.
A detailed marking rubric will be provided in L@G.

About collaboration:

You are encouraged to discuss the assignment with your fellow students, but eventually this should be your own work. Anyone caught plagiarising will be penalised.