Assignment 2

2802ICT Intelligent System
School of ICT, Griffith University
Trimester 1, 2023

Instructions:

- **Due:** Monday 29th May 2023, 11:59 PM with demonstrations to be held on Week 12.
- **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 or special circumstances ((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 and submit an authentic work. This should be your own work. Anyone caught plagiarising will be penalised and reported to the university.
- **Presentation:** You must present/demonstrate your work. Your work will not be marked if you do not present it.

Maximum marks are:

- **25** for **Task 1**
- **40** for **Task 2**
- **20** for the **report**
- **15** for the **interview (demonstration + open questions)**

Total: 100 marks

Objectives:

✓Implement learning algorithms
✓Evaluate the performance of a simple learning system on a real-world dataset

Your code should be written in Python. If you wish to use another programming language please contact the course convenor 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.

Task 1 - Decision Trees

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.
c. You should be able to randomly split the dataset into testing and training set.
d. Test your classifier and print:
    1. The size of the training and testing sets
    2. Total accuracy
    3. Confusion matrix
    4. precision, recall and F1-score values for each class, together with the macro-average and weighted -average
    5. Plot the Learning curve (accuracy as a factor of percentage of learning example) i.e. show how the accuracy changes while learning.

    For example, if the training set includes 1000 samples, you may stop building the tree after 100 samples and test with the current tree, then continue building the tree and stop again to test after 200 samples, and so on until you have the final decision tree which is built using the full training set.

   Note: To make plots, you can use the Python library matplotlib.

Report Requirements

Your report should include:

a. Software design: a detailed description of your code and how it works. Include details on key functions and data structures you are using in your program.
b. The results and graphs from paragraph (d.) above.
c. Discuss the results, include any observations or insights you gained from running tests.
d. Write a conclusion paragraph that summaries and explains your findings.

Submission

For task 1 you should submit the following:

1. zip file that includes your code (.py files). Name the file: `firstname_surname_Snumber_A2T1.zip` (for example, `Bart_Simpson_s123456_A2M1.zip`)
2. A copy of your source code in one of the following formats: Microsoft Word (.doc/.docx), Plain text (.txt), or Adobe PDF (.pdf).

* Make sure your code is well documented and include comments to explain key parts.

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

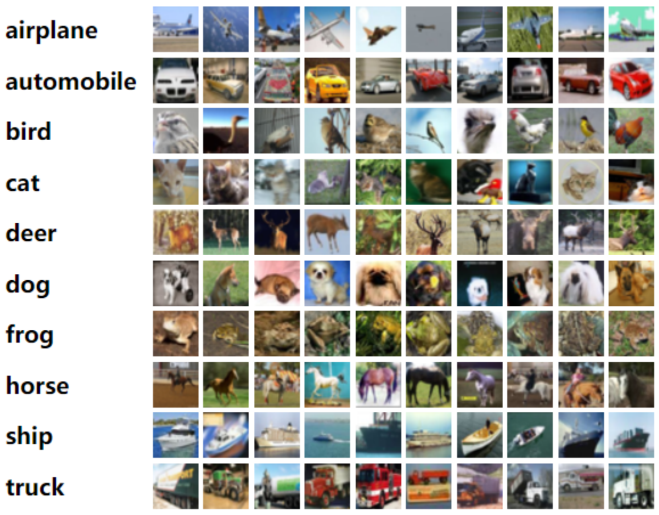
Marking scheme

A detailed marking rubric will be provided in L@G.

Task 2 - Classification using Neural Network

In this task, you need 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.

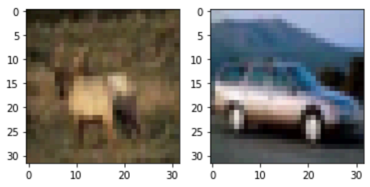
The dataset to learn is called **the CIFAR-10 dataset** and is a subset of the 80 million tiny images dataset. This dataset consists of 60,000 32x32 colour images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images.



The train and test datasets can be downloaded from here. Make sure you download the CIFAR-10 python version. You will also find there instructions on how to open the files as a python dictionary.

Data

Each image is 32x32 pixels, which is 1,024 pixels in total. Each pixel has an RGB (Red,Green and Blue) value represented as an integer between 0 and 255. Hence, an image in the dataset is represented as an array of size 3,072 (32x32x3), in which the first 1024 entries contain the red channel values, the next 1024 the green, and the final 1024 the blue.



You will notice that the dataset is divided into five training batches and one testing batch, each contains 10,000 examples.

Labels
Each training and testing image is assigned to one of the following labels:

0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

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 will have three layers: an input layer, one hidden layer and an output layer.

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 neurones. 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 the 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 neurones 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, 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: ith test instance vector

y_i: ith 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 images recognition assignment, we will encode the output (a number between 0 and 9) by using **10 output neurones**. The neurone 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.

It is important that your code enable simple and easy adjustment of key parameters such as the number of neurones in the input, hidden and output layers, as well as the number of epochs, mini-batch size and learning rate.

Your code should also run on a small neural network for testing purposes. See below.

You should do the following:

CIFAR-10 dataset:

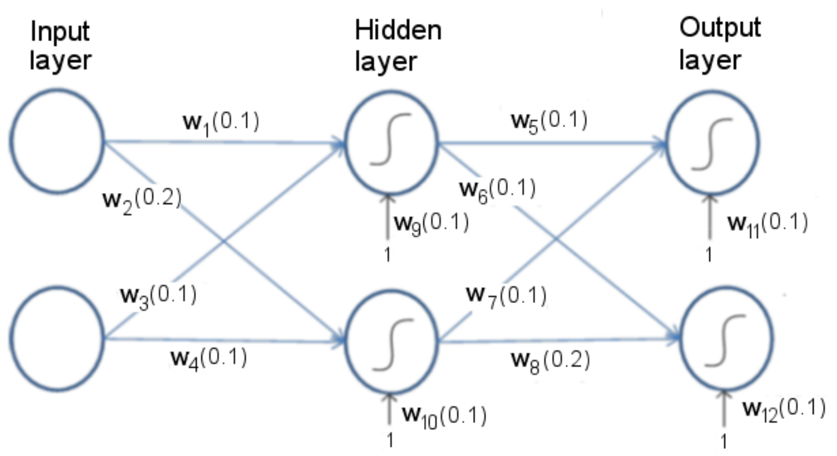
1. Create a neural network of size [3072, 30, 10], i.e. 3072 neurones in the input layer, 30 neurones in the hidden layer, and 10 neurones in the output layer. Then train it on the training dataset with the following settings: epoch = 20, mini-batch size = 100, η = 0.1.

   Test your code with the testing dataset and draw a graph of **test accuracy vs epoch**. Print the **maximum accuracy** achieved.

2. **Learning rate:** Train a new neural network with the same settings as in (1.) but with different learning rates η = 0.001, 0.01, 1.0, 10, 100. Plot a graph of **test accuracy vs epoch for each η on the same graph**. Print the **maximum accuracy achieved for each η**. Remember to create a new neural net each time so its starts learning from scratch.

3. **Mini-batch size:** Train new neural net with the same settings as in (1.) above but with different mini-batch sizes = 1, 5, 20, 100, 300. Plot **maximum test accuracy vs mini-batch size**. Which one achieves the maximum test accuracy? Which one is the slowest?

Testing with a small neural network:



The above small network has two neurones in the input layer, two neurones in the hidden layer and two neurones in the output layer. Weights and biases are marked on the figure. W9, w10, w11, w12 are weights for the bias term.

There are two training samples: X1 = (0.1, 0.1) and X2 = (0.1, 0.2). The label for X1 is 0, so the desired output for X1 in the output layer should be Y1 = (1, 0). The label for X2 is 1, so the desired output for X2 in the output layer should be Y2 = (0, 1).

You should update the weights and biases for this small neural net using stochastic gradient descent with back-propagation using batch size of 2 and a learning rate of 0.1 in one epoch, i.e. run forward pass and backward pass for X1, then run forward pass and backward pass for X2, then update the twelve weights based on the values you calculated from X1 and X2.

Output: Print the new (updated) twelve weights.

A sample output for the small network will be available on L@G to allow you to test your implementation.

Report requirements

Your report should include:

a. Software design: information about key functions 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 those results.
c. Write a conclusion paragraph that summaries and explains your findings.

Submission

For task 2 you should submit the following:

4. zip file that includes your code (.py files). Name the file: `*firstname_surname_Snumber*_A2T2.zip` (for example, `Bart_Simpson_s123456_A2M2.zip`)
5. A copy of your source code in one of the following formats: Microsoft Word (.doc/.docx), Plain text (.txt), or Adobe PDF (.pdf).
6. **Report**: You can include task 1 and 2 in the same report or submit two seperate reports.

Submission should be done through the link provided in L@G by the due date. Make sure you submit at least five files: two zip files, two copies of the code, and a report.

Marking scheme

A detailed marking rubric will be provided in L@G.