CPS109 Fall 2016
Assignment 2: Machine Learning-Computer Vision

Due Date: 11:59pm December 5th, 2016

You are to **work alone** when writing your code. You can discuss general ideas with your classmates, but you cannot copy code or develop code together (changing identifiers and rearranging code does not help) nor take code from the web. We will be using the Measure of Software Similarity (MOSS) to identify cases of possible plagiarism; see the following link for details: http://theory.stanford.edu/~aiken/moss. The Department of Computer Science takes the act of plagiarism very seriously. **Those caught plagiarizing (both originators and copiers) will be sanctioned.** Please see Ryerson University's Policy 60 for possible penalties and consequences: http://ryerson.ca/senate/policies/pol60_procedures.pdf. The due date and time is firm. No assignments will be accepted after the deadline.

You will implement an application for image recognition, where the goal is to automatically assign a categorical label to an image based on its contents. In this assignment, your program will be provided an image containing a digit between 0-9 and will determine what digit the image contains. The image recognition system is based on an artificial neural network (ANN).

Artificial Neural Networks are behind many of the recent exciting advances in the field of computing[1], such as voice recognition, image recognition and self-driving cars. ANNs are a branch of machine learning, where the goal is to have a computer learn from experience (i.e., data) rather than being explicitly programmed[2].

What is an artificial neural network? ANNs are a computational approach based on a large collection of connected computational units called artificial neurons or perceptrons. This model is loosely inspired by the way the brain solves problems

---

[1] http://www.nytimes.com/2012/11/24/science/scientists-see-advances-in-deep-learning-a-part-of-artificial-intelligence.html.

[2] https://en.wikipedia.org/wiki/Machine_learning

through a large cluster of biological neurons connected by axons. At an abstract level, you can think of an ANN as a non-linear mathematical function that takes in a set of numbers performs some computation and outputs another set of numbers.

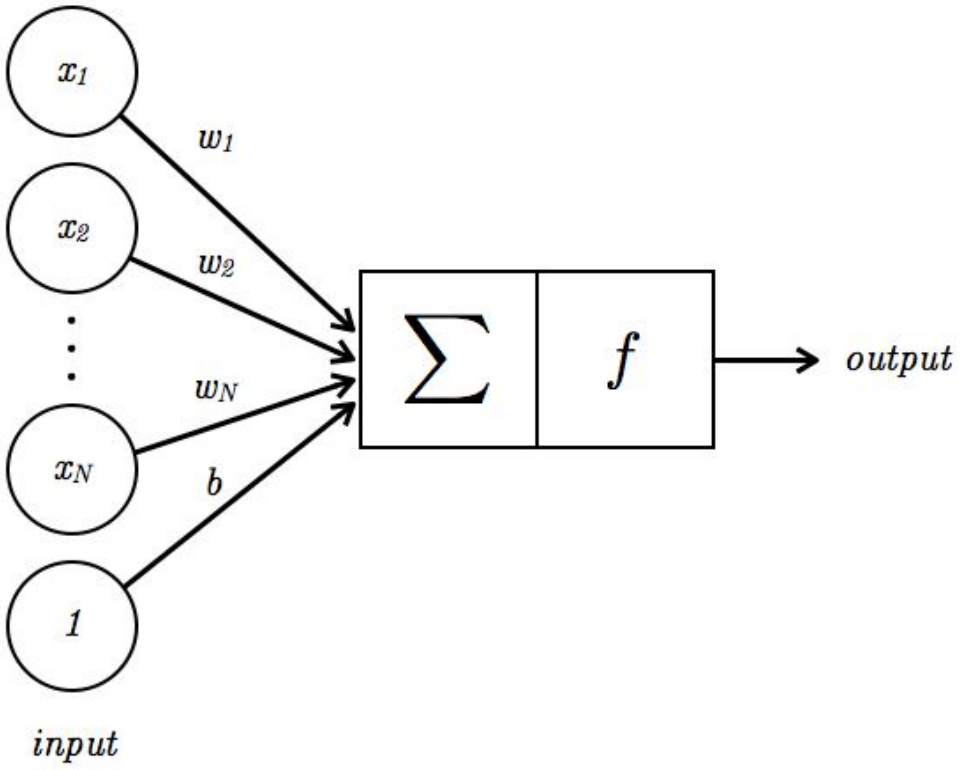
The fundamental building block of an ANN is the perceptron. A perceptron takes as input a set of numbers, multiplies each input by a number, sums up the results and then applies a non-linear function to yield the final result. Formally, the output, r, of a perceptron is given as follows:

$$r_{\text{perceptron}} = f(\sum_{i=1}^{N} x_i w_i + b)$$

,

where x_i is the input number, w_i is the corresponding weight and b is an additive bias term, i.e., a number.

An illustration of a perceptron is shown in the figure below:



For example, assume that the image contains four pixels, x_1=10, x_2=55, x_3=155 and x_4=201. The four corresponding weights are given by w_1 = 0.5, w_2 = 1, w_3 = 3.2 and w_4 = 1.1. The bias term is given by b = -5. The weighted sum is then:

$$\sum_{i=1}^{4} x_i w_i + b = 10 * 0.5 + 55 * 1 + 155 * 3.2 + 201 * 1.1 - 5 = 772.1$$

.

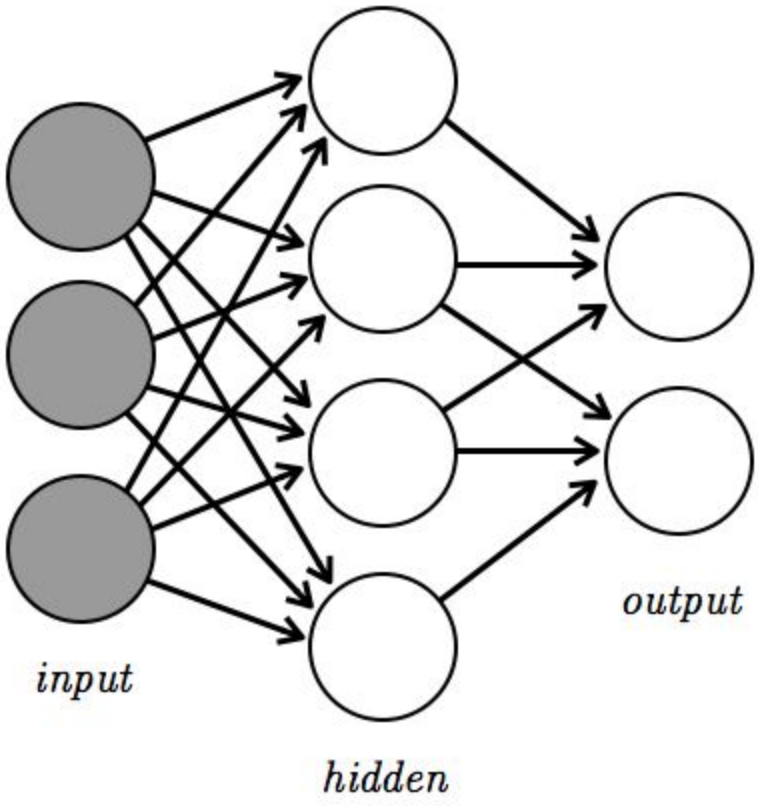
There are a multitude of possible options for the non-linear function, f. For this assignment, the non-linear function, f, will be a sigmoid which is defined by:

$$f(z) = \frac{1}{1 + e^{-z}}.$$

Continuing on with our example and now applying the sigmoid function, the final perceptron result is:

$$f(772.1) = \frac{1}{1 + e^{-772.1}}.$$

The overall ANN is composed of a series of layers, starting with an input layer containing the input image data and then a sequence of layers each containing a set of perceptrons. Each perceptron has its own unique set of weights and bias. In this assignment, you will be implementing a fully connected network, where each unit is connected to every unit in the preceding layer. There will be one input layer, followed by a layer of perceptrons termed the hidden layer and ending with a layer of perceptrons called the output layer. An illustration of a (fully connected) ANN is shown in the figure below:

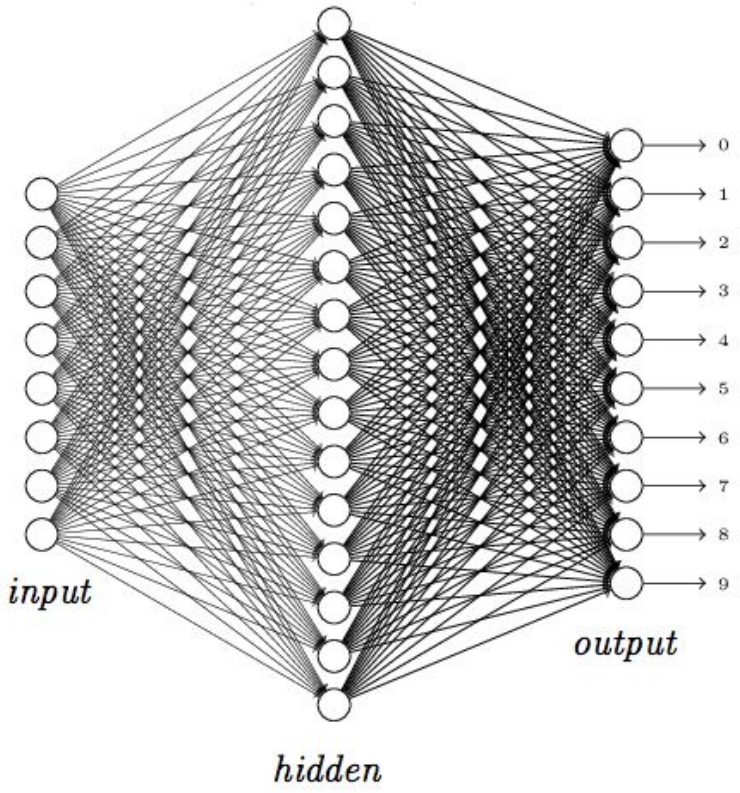


The weights, w, and bias, b, are “learned” through a process known as backpropagation. In short, this process is an efficient way of computing the “gradient” of our function using the “chain rule” from calculus. For this assignment, you will not be responsible for implementing the learning procedure. Instead, the weights will be provided to you in a file. For those interested in understanding the “gradient” and “chain rule”, and machine learning in general, it is highly recommended to take MTH 330 Calculus and Geometry.

The input in this assignment will be a greyscale image with dimensions 28x28 and values are assumed to range between 0 and 1, where 0 and 1 represent black and white, respectively, and anything in between is a shade of grey. An example input image of the digit 4 is shown in the following figure:

You can think of the input image as a one dimensional array of 28x28=784 intensity values by stacking each row of the image one after the other. The hidden layer contains 300 neurons and the output layer contains 10 neurons, where each one indicates the likelihood that the input is one of the digits. When computing the class of the input image, one computes the perceptron outputs layer by layer, using the previous layer as input. The decision for the final class is made based on the output node that contains the largest value. This is called the feedforward pass.

As an illustrative example, the following figure shows an input image with 8 pixels in total, 15 neurons in the hidden layer and 10 output neurons in the output:



In this assignment you will implement the feedforward pass. The steps to implement are as follows:

1. Read in the weights and biases for both the hidden and output layers from the files `hidden-weights.txt` and `output-weights.txt`, respectively. Each row of the files contains the weights for a neuron, with the last value corresponding to the bias. In `hidden-weights.txt` there should be 300 rows with 785 values per row (784 weights + 1 bias). In `output-weights.txt`, there should be 10 rows by 301 values per row (300 weights + 1 bias).

2. Read in image (the filename is to be provided at the command line).

   The Java standard library package `java.awt.image` contains a class `BufferedImage` whose objects represent two-dimensional pixel raster images. To read in the image and place it in a one dimensional array (each image row concatenated after the previous one) use the following code:

```
img = ImageIO.read(new File("src/number.png"));

// get pixel data
double[] dummy = null;
double[] X = img.getData().getPixels(0, 0, img.getWidth(),
   img.getHeight(), dummy);
```

   You will have to include the following packages to read in an image:

```
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.*;
```

3. You will next need to rescale the values of the input image from its original range of 0 to 255 to the required range 0 to 1.

4. Given the input image, compute the outputs for the hidden layer.

5. Given the results of the hidden layer as input, compute the results of the output layer.

6. Print to the screen the label of the output neuron with the highest value, this corresponds to the prediction of your neural network based on the input image. Your code should print to the screen the following: “The network prediction is XX.”, where XX corresponds to the digit class with the highest ANN output value.

Submission:

You have to submit to BrightSpace the following files (and no more):

1. `feedforward.java` (your code)

2. `comments.txt` (optional, contains comments you want the markers to read about your submission)

Marking rubric (out of 8):

- 2 marks for proper coding style (proper indentation of the code and meaningful variable names used in the program)

- at most 2 marks for a plausible attempt at a solution

- 4 marks for code that compiles and executes correctly, i.e., computes and outputs the correct solution

Bonus:

We have provided you a folder containing 100 test images and a text file, `labels.txt`, containing the filename of each image and its corresponding label. For the bonus you are to do the following:

1. Read in `labels.txt`

2. For each row in the text file, read in the filename and corresponding label

    a. Compute the neural network output for the image

    b. Compare the predicted label with the label in the file

3. Count the number of correct predictions

4. Report the classification rate given by the number of correct predictions divided by the number of test images; do not hardcode the value for the number of images, it may change when we are evaluating your code