Assignment 3

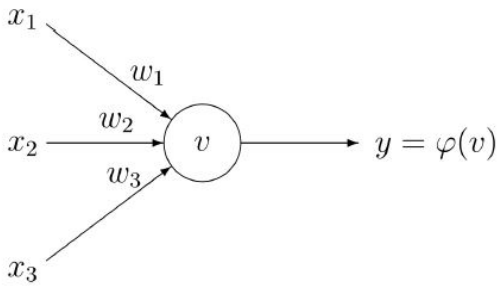
Due on April 30, 2021 (23:59:59)

Click here to accept your Assignment 3

Instructions. There are two parts in this assignment. The first part involves a series of theory questions and the second part involves coding. The goal of this problem set is to make you understand and familiarize with Neural Network algorithm.

PART I: Theory Questions

1. Consider the simple neuron structure below:



Asssume that the weights for the neuron are $w_1 = 3$, $w_2 = -5$ and $w_3 = 2$ with activation function below:

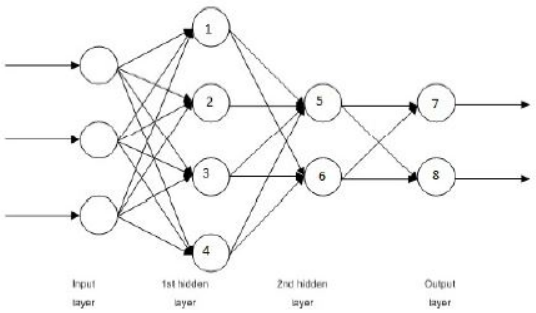
$$\varphi(v)) = \begin{cases} 1 & v \geq 0 \\ 0 & otherwise \end{cases} \tag{1}$$

find the output y values for the input patterns below:

Input	I_1	I_2	I_3	I_4
x_1	1	0	1	1
x_2	0	1	0	1
x_3	0	1	1	1

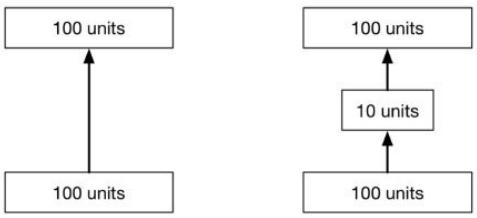
2. Consider the multi-layer neural network below:

   - Find how many weight variables the network has in total (Ignore bias values). Show your calculations.



- Find how many weight variables the network has in total if the network is considered as fully connected (Ignore bias values). Show your calculations.
- State the dependency information for nodes given number values, which are about which node takes information from which previous node. State also these dependencies for both forward and back-propagation streams.

3. Assume that you have two networks (one in left and one in right) in the figure below:

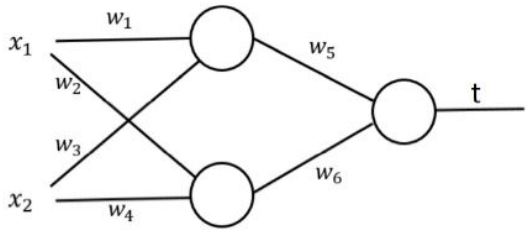


- Specify an advantage of network in the left over one in the right. Explain your answer.
- Specify an advantage of network in the right over one in the left. Explain your answer.

4. Consider the network structure below:
Also you are given the activation function $f(u)$ below for the nodes in the structure above:

$$f(u) = \begin{cases} (u + |u|)^q & u \geq 0 \\ (u - |u|)^q & otherwise \end{cases} \tag{2}$$

where q is a fixed parameter.



Finally assume that error is measured by the formula below:

$$E = \frac{1}{N}\sum_{i=1}^{N}(y-t)^2 \tag{3}$$

Where y is the desired output and t is the actual output.

- Write the gradient formulas of the error with respect to the all weight parameters. Show your steps properly.
- For $q = 1$, state that whether the model becomes to a linear regression model or not. Explain why or why not.

5. Fill the blanks with T (True) or F (False) for the statements below:
   - In every condition, a perceptron network perfectly learns a linearly separable function through a finite number of training steps. ( )
   - Single perceptron can compute the XOR function. ( )
   - In backpropagation learning, the model should start with a small learning parameter and slowly increase it while it is in the learning process. ( )

PART II: Classification of Natural Scenes using Neural Network

For this assignment, you will implement a single layer and multilayer neural network architecture to classify natural scenes around the world into 6 classses.

1 Dataset

- Image data of Natural Scenes contains 150x150 images of natural scenes and corresponding labels. The pictures are divided into six classes: buildings, forest, glacier, mountain, sea, street.

(a) buildings (b) forest (c) glacier (d) mountain (e) sea (f) street

Figure 1: Sample images from each class in Natural Scenes dataset

- Dataset and the code to load it are provided to you. You can donwload them from link. The dataset is split into three sets: Training (14034 images), test 3650 and validation 3000 images).
- When you load the dataset, you need to resize images to 30x30 and convert them to gray images. After resizing, you'll see that every image is represented with [900x1] vector and has a label. You'll use this vectorized pixel representation (do not have to extract new features).

2 Single Layer Neural Network

In the first step, you will implement the network given in Figure 2 and train the network feeding by given training set as 900 dimensional gray-level image values. It is important to normalize image values $(0 - 255)$ to between 0 and 1. We can express this network mathematically as:

$$o_i = w_{ij}x_j + b_i \tag{4}$$

As loss function, you will use sum of negative log-likelihood of the correct labels. Write a python function to compute loss function. Then you will update network parameters w and b to minimize the loss function using gradient descent algorithm. You will implement a function which computes the derivative of the loss function with respect to the parameters. To make sure your function is correct, you must also implement numerical approximation of gradients.

Write a function to minimize your cost function using mini-batch gradient descent. You should try different learning rate$(0.005 - 0.02)$ and batch sizes$(16 - 128)$. Make a table to show learning performance for each setting you tried.

Finally, you will visualize the learned parameters as if they were images. Visualize of the each set of parameters that connect to $O_0, O_1, \ldots, O_5$. Please discuss the visualization of parameters that your model learned. You have sample code for how to visualize images.

You will use Natural scenes dataset which is explained in Section 1, for both classification methods.

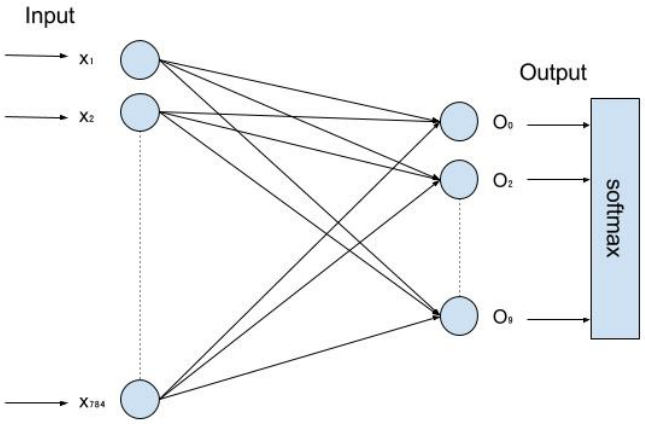


Figure 2: Single layer neural network.

3 Multi Layer Neural Network

In this part of the assignment, you have to implement multi layer neural network for classification. In other words your network consists of one input layer, n hidden layer(s) and one output layer. You will implement forward and backward propagations with the loss function and learning setting as explained in the previous section. Actually, you will implement a back-propagation algorithm to train a neural network.

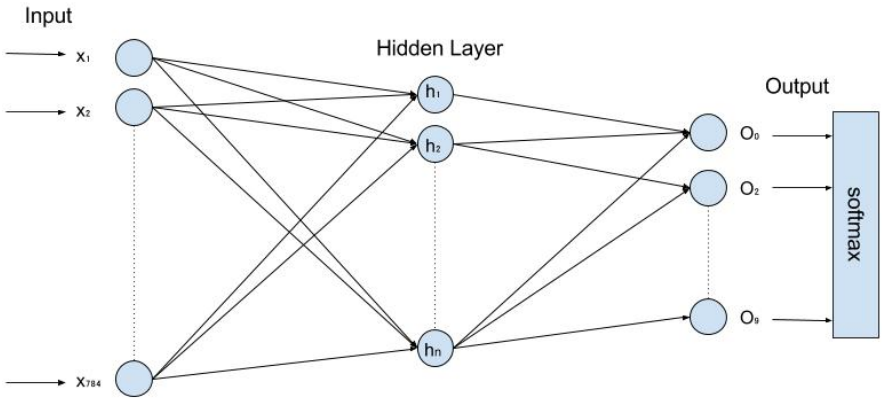


Figure 3: Multi layer neural network.

4 Implementation

Training a network

- You should determine the optimal number of units in your hidden layer.
- You should determine the optimal batch size as you learned in the class.
- You should determine an optimal learning rate for your gradient descent method.
- Remember, learning rate parameter may be a problem (too big - may not converge, too small - very slow convergence). For this reason you can define a learning rate decay parameter. You will start with a learning rate value and after each epoch you will reduce the learning rate by multiplying it by a decay rate. This operation can deal with mentioned problem.
- You can use different activations functions: Sigmoid, tanh, ReLU etc.
- You can use different objective functions: Cross-entropy error, sum of squared error (SSE) etc.
- You can control your implementation by plotting the loss. You can see if it converges or if it needs a different parameter setting.
- You should discuss your observations from each experiment in the report. Comment about their effects.
- Save your trained models to use later in test time.

Notes:

- You will implement a single layer neural network and run experiments on natural scenes dataset. You'll change parameters (activation func., objective func. etc.) and report results with a table format in your reports.
- You will implement a neural network which contains one hidden layer. You'll change the mentioned parameters (unit number in the hidden layer, activations function etc.) and report the results.
- Then you'll change your architecture and use a network that contains two hidden layers. Repeat same experiments and comment about the results.
- You have to comment about your results and the effect of parameters.
- Your implementation should be reproducible, in other words, do not write separate code for each architecture. If you use n hidden layers, your method should create a n-layer network and learn the classifier.)
- Comment your code with corresponding mathematical functions and explain what is going on your code in the code scripts.

Running:
Someone manage to run your code as shown:
python train.py -data_path /path/to/train/data
python test.py -data_path /path/to/test/data -model_path /path/to/trained/data
You are free to add arguments for loss function, hidden layer size, activation function, etc.(see "argparser" library in Python). Please give information about your arguments and how to run your code. Make sure you saved the trained parameters to use in test time.

5 Calculation of Accuracy

You will compute the accuracy of your model to measure the success of your classification method:

$$\textbf{Accuracy} = 100 * (\frac{\textbf{number of correctly classified examples}}{\textbf{number of examples}}) \tag{5}$$

NOTE: To enter the competition, you have to register Kaggle in Class with your department email account. The webpage of the competition will be announced later. Top 5 assignment will earn extra points.

6 (Bonus) Using VGG-19 Features for classification

Extract deep image features of VGG-19 net for classification of images and apply single and multi layer neural network.

Submit

You are required to submit all your code with a report in ipynb format (should be prepared using Jupyter notebook). The codes you will submit should be well commented. Your report should be self-contained and should contain a brief overview of the problem and the details of your implemented solution. You can include pseudocode or figures to highlight or clarify certain aspects of your solution. Finally, prepare a ZIP file named **name-surname-pset3.zip** containing

- report.ipynb (PDF file containing your report)
- code/ (directory containing all your codes as Python file .py)
- model/ containing your learned parameters to use in test time (you can use numpy.save and numpy.load)

The ZIP file will be submitted via Github Classroom.Click here to accept your Assignment 3

Grading

- Code (53) : Single-layer neural network: 18, Multi layer neural network: 30, Accuracy: 5, Bonus: 10
- Report(47): Theory part: 12 points, Analysis of the results for classification: 35 points.

  Notes for the report: You should analyse the method you employed. How did you improve your results? Explain every step you choose. Comment about the results. Compare single layer and multi layer neural network. Comment about the activation functions, loss functions etc. that you used for your experiments. Your reports have to include your classification accuracy.

Late Policy

You may use up to four extension days (in total) over the course of the semester for the three problem sets you will take. Any additional unapproved late submission will be weighted by 0.5. You have to submit your solution in (rest of your late submission days + 4 days), otherwise it will not be evaluated.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else. [1]

[1] This assignment is adapted from http://www.cs.toronto.edu/ guerzhoy/321/proj2/