Assignment 3

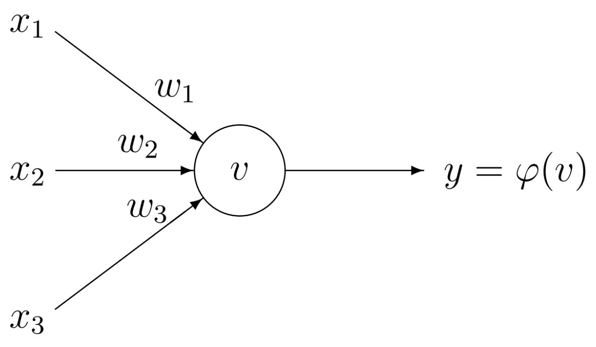
Due on December 13, 2019 (23:59:59)

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:



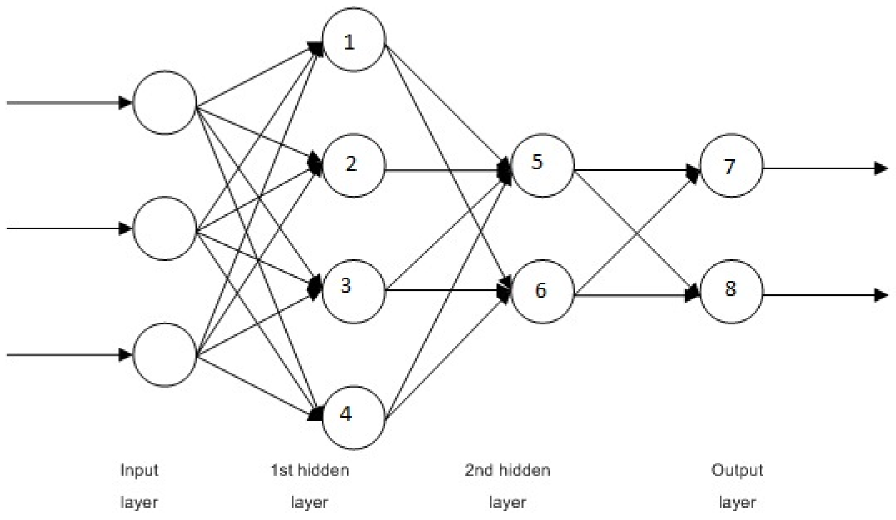
Asssume that the weights for the neuron are $w_1 = 3$, $w_2 = -5$ and $w_3 = 2$ with activation function below:

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

find the output y values for the input patterns below:

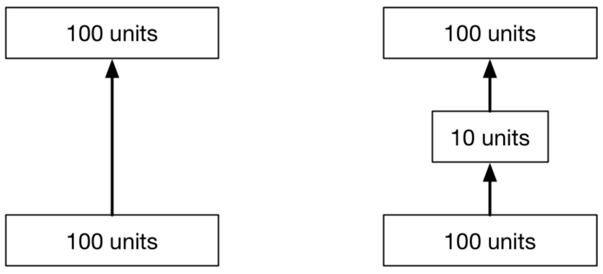
Input	I_1	I_2	I_3	I_4
x_1	1	0	1	1
x_2	0	1	0	1
x_3	0	1	1	1

2. Consider the multi-layer neural network below:



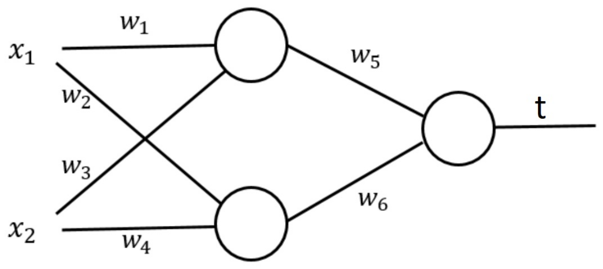
- Find how many weight variables the network has in total (Ignore bias values). Show your calculations.
- Find how many weight variables the network has in total if the network is considered as fully connected (Ignore bias values). Show your calculations.
- State the dependency information for nodes 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 below:

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

where q is a fixed parameter.

Finally assume that error is measured by the formula below:

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

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, also explain your reason:
   - 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 Herbs using Neural Network

For this assignment, you will implement a single layer and convolutional neural network architecture to classify the examples in the dataset mentioned below.

1 Dataset

- You will use a small version of Herbarium Challenge Dataset [1] contains 34.225 images of herbarium specimens and corresponding 683 labels. (See Figure 1)

  Training and test dataset will provided later and be announced from Piazza group.

Figure 1: Different species from the dataset

2 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. Actually, you will implement a back-propagation algorithm to train a neural network.

In this step, you will implement the network given in Figure 2 and train the network feeding by given training set as 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:

$$o_i = w_{ij}x_j + b_i \tag{2}$$

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 $O_0, O_1, \ldots, O_n$.

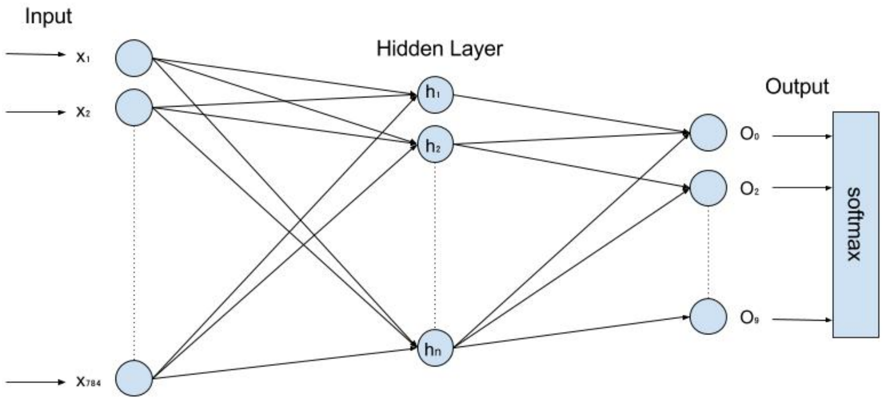


Figure 2: Multi layer neural network.

You will use Herbarium Challenge Dataset which is explained in Section 1, for both classification methods.

3 Convolutional Neural Network

In this part of the assignment, you have to implement convolutional neural network (CNN) for classification. In other words your network consists of n convolutional layers, m fully-connected 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.

Training a network

- You should determine the number of units in your convolutional layers and fully connected layers.
- You should determine convolutional layers width, height, depth parameters.

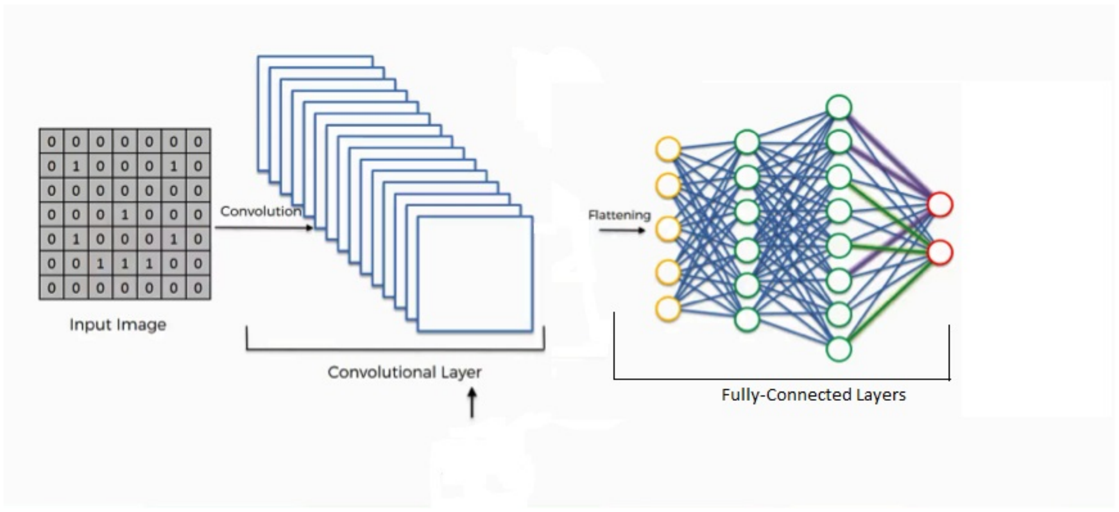


Figure 3: An example convolutional neural network.

- You should determine batch size as you learned in the class.
- You should determine a 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 loss. You can see if it converges or if it needs a different parameter setting.
- **You should discuss about your each experiment in the report. Comment about their effects.**
- Save your trained models to use later in test time.

Implementation Details:

1. For implementing CNN, you can utilize libraries from the PyTorch (`https://pytorch.org/`) and you can use transfer learning which means you can fine-tune a pre-trained CNN model for the classification.
2. You can also look into the links below for further analyze:
   - `https://blog.algorithmia.com/convolutional-neural-nets-in-pytorch/`

- `http://cs231n.github.io/convolutional-networks/`
- `https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`
- `https://pytorch.org/docs/stable/nn.html`

Notes:

- You will implement single layer neural network and run experiments on Herbarium Challenge Dataset. You will change parameters (activation func., objective func. etc.) and report results with a table format in your reports. **Obligatory**
- You will implement a neural network which contains one hidden layer. You will change the mentioned parameters (unit number in the hidden layer, activations function etc.) and report the results. **Obligatory**
- Then you will change your architecture and use a network that contains two hidden layers. Repeat same experiments and comment about the results. **Obligatory**
- You will also implement a convolutional neural network which contains one convolutional layer and one fully connected layer. You'll change the mentioned parameters (unit number in the hidden layer, activations function etc.) and report the results. **Obligatory**
- Then you'll change your architecture and use a network that contains two convolutional layers and two fully connected layers. Repeat same experiments and comment about the results. **Obligatory**
- You can also try different layer size (convolutional or fully-connected) for your normal neural network or CNN model. But, you should not restrict yourself with the obligatory models above (especially for CNN), as you try different combinations with layers size and other parameters, your chance to get full point from the analysis report increases.
- You have to comment about results and parameters' effects on different values(learning rate, batch size, layer size, hidden neuron size in the layers etc...).
- Your implementation should be reproducible, in other words, do not write separate code for each architecture. If you use n 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.
- **You should also compare your classic neural network and convolutional neural network with respect to accuracy, parameter (weight size) size and training-test error curve plots and you must state every experiment**

you accomplish and related proper explanation/conclusion about why you obtain such a result in your analysis report in order to get full points on analysis report.

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.

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_and_code.ipynb (PDF file containing your report and also 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 (60 points) : 35 points for multi-layer neural network implementations, 25 points for convolutional neural network implementations.
- Report(40 points): Theory part: 20 points, Analysis of the results for classification: 20 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 convolutional 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]

References

[1] This reference is hidden due to the obvious reasons.

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