CAP 6619 Deep Learning

2024 Summer

Homework 2 [15 Pts, Due: June 2, 2024. Late Penalty: -2/day]

[If two homework submissions are found to be similar to each other, both submissions will receive 0 grade]

[Homework solutions must be submitted through Canvas. No email submission is accepted. If you have multiple files, please include all files as one zip file, and submit zip file online (only zip, pdf, or word files are allowed). You can always update your submissions. Only the latest version will be graded.]

Question 1 [2 pts]: Figure 1 shows some samples in a two-dimensional feature space (X_1, X_2). Each symbol (circle, triangle, plus, diamond) denotes a sample, and each shape denote one type of samples. Please design a neural network architecture which can learn to classify samples in Figure 3 into correct types.

- Show your network input dimension, number of hidden nodes (layers), and output node(s) [1 pt].
- Explain why you use this network architecture [1 pt]

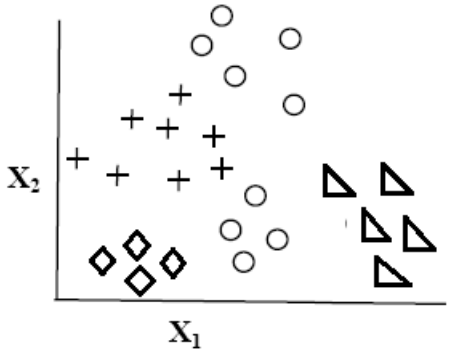


Figure 1

Question 2 [1 pt] Figure 2 shows three neural network structures. (a) is a single layer neural network, (b) is a one hidden layer neural network, and (c) is a two hidden layer neural network, where each rectangle box denotes a one hidden layer network similar to (b).

- Please draw decision regions for each network, respectively [0.5 pt],
- Explain how the networks learn to classify samples into different categories [0.5 pt]

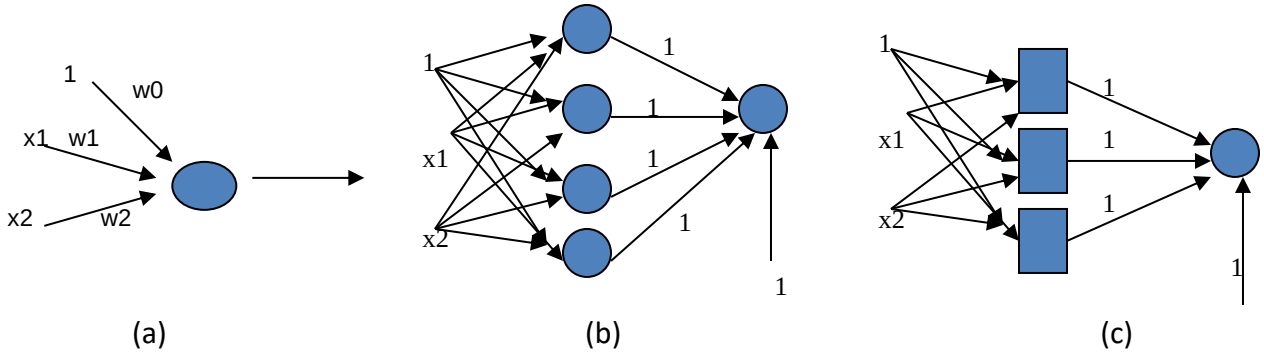


Figure 2: Feedforward neural networks

Question 3 [2 pts]. Given a neuron with *n* dimensional input features [x_1, x_2, …, x_n], assuming the output from the neuron is defined by

$$o = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$$

- Please draw the structure of the neuron (show input, output, and the weight) [0.5 pt]
- Given a batch of N training instances (x(1), d(1)), (x(2), d(2)), …, (x(N), d(N)), where d(i) denotes label of instance x(i), show gradient descent learning objective function of the neuron [0.5 pt]
- Derive gradient descent learning weight updating rule for weight w_i [1 pt].

Question 4 [2 pts]: In Figure 3, a one hidden layer neural network is trained to predict an input as “dog” or “cat”. For each input, if the label of the input is “dog”, its output from node “e” is expected to be 1, otherwise the output is expected to be 0. Similarly, if the label of the input is “Cat”, its output from node “f” is expected to be 1, otherwise the output from node “f” is 0 (using sigmoid activation function, and parameter a=1).

Given an input [x0, x1, x2]=[1, 0.5, 0.5] which is labeled as “dog”, where x0 denotes bias,

- Calculate its actual output from node “e” and “f”, respectively. [0.5 pt]
- Calculate network error of the instance. [0.5 pt]
- Calculate local gradient of the instance with respect to node “e” and “f”, respectively. [0.5 pt]
- Calculate local gradient of the instance with respect to node “a” and “b”, respectively [0.5 pt]

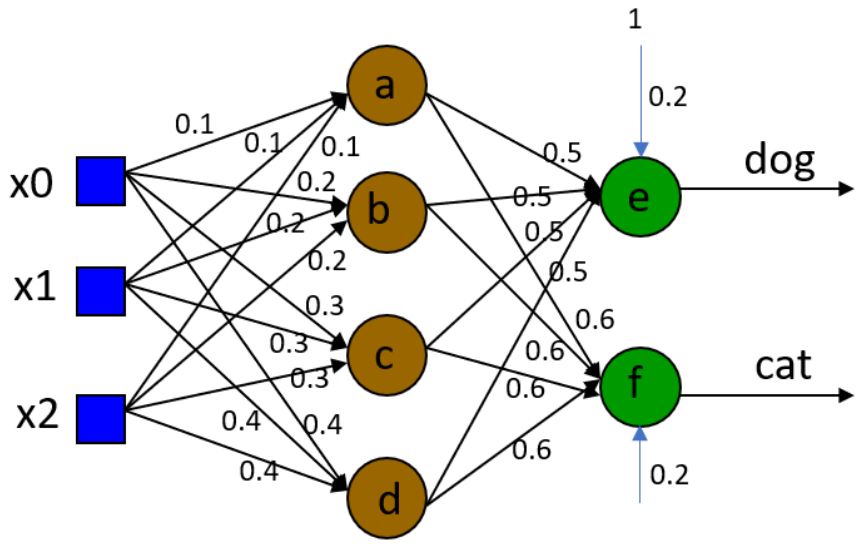


Figure 3: A one hidden layer neural network

Question 5 [3 pts]: Given the following feedforward neural network with one hidden layer and one output layer, assuming the network initial weights are

$\begin{bmatrix} w_{0a} \\ w_{1a} \\ w_{2a} \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}$; $\begin{bmatrix} w_{0b} \\ w_{1b} \\ w_{2b} \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}$; $\begin{bmatrix} w_{0c} \\ w_{ac} \\ w_{bc} \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}$. All nodes use a sigmoid activation function with value a=1.0, and the learning rate η is set to 0.1. The expected output is *1* if an instance’s class label is “True”, otherwise, the expected output is *0*. Given the following three instances I_1, I_2, I_3 which are labeled as “True”, “True”, and “False”, respectively,

- Please calculate actual outputs of each instance from the network? [1 **pt**]
- Calculate mean squared error of the network with respect to the three instances? [**1 pt**]
- Assuming instance I_1 is fed to the network to update the weight, please update the network weight (using backpropogation rule) by using this instance (list major steps and results). [**1 pt**]

$$I_1 = \begin{bmatrix} 1.0 \\ 1.0 \\ 0.5 \end{bmatrix}; \quad I_2 = \begin{bmatrix} 1.0 \\ 0.0 \\ 1.0 \end{bmatrix}; \quad I_3 = \begin{bmatrix} 1.0 \\ 0.5 \\ 0.5 \end{bmatrix}$$

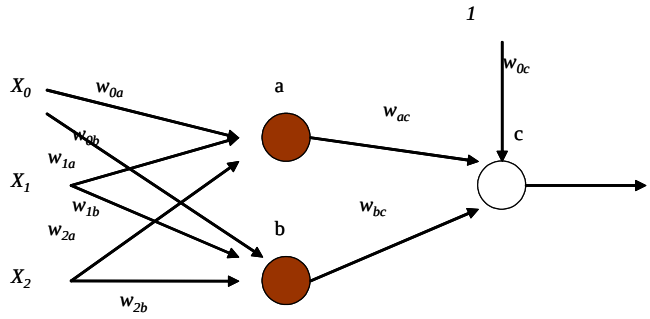


Figure 4: A hidden layer neural network

Questions 6 and 7 are programming tasks

For all programming tasks, solutions must be submitted as notebook (html or pdf) files for grading (your submission must include scrips/code and the running results of the script).

If you are not familiar with Python programming (and want to use Python for the coding tasks), please check Python Plotting notebook and Python Simple Analysis notebook posted in the Canvas, before working the coding tasks.

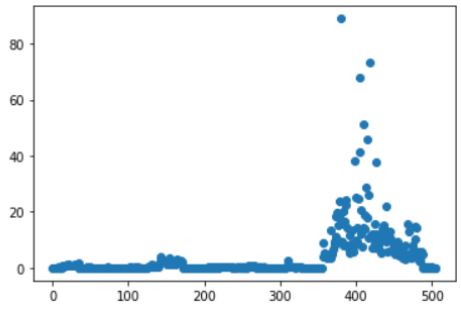
For each subtask, please use task description (requirement) as comments, and report your coding and results in following format:

```
# Report all samples with respect to the Crim index on a plot (the x-axis shows the index of the sample, and the y-axis
# shows the crim index of the sample).

y = boston['Crim']
x=np.arange(y.shape[0]) # generate x index

plt.scatter(x, y, marker='o')
```

```
<matplotlib.collections.PathCollection at 0x1c0d3ceebc8>
```



Question 6 [2 pts] The **Dense Layer Classification [Notebook, html]** in the Canvas shows detailed procedures about how to use Oivetti face dataset (from AT&T) to train Neural Network classifiers for face classification. The dataset in the Canvas also provides "olivetti_faces.npy" and "olivetti_faces_target.npy", which includes 400 faces in 40 classes (40 different person). Please implement following face recognition task using Neural Networks.

- Please show at least one face images for each class in the Oivetti face dataset100. Randomly split the dataset into 60% training and 40% test samples. Train a one-hidden layer neural network with 10 hidden nodes. Report the classification accuracy of the classifier on the test set [1 pt]
- Please use one time 10-fold cross validation to compare the performance of different neural network architectures, including (1) one-hidden layer NN with 10 hidden nodes, (2) one-hidden layer NN with 50 hidden nodes, (3) one-hidden layer NN with 500 hidden nodes, and (4) two-hidden layer NN with 50 hidden nodes (1^{st} layer) and 10 hidden nodes (2n layer). Please report and compare the cross-validation accuracy of the four neural networks, and conclude which classifier has the best performance [1 pt].

Question 7 [3 pts]: Please follow "Dog Cat Dense Layer Classification **[Notebook, html]** " to create a dense layer network image classifiers and validate its performances.

- Please download the cat vs. dog images (dogvscat1000.zip) from Canvas (You can also download whole image dataset from the Kaggle site (https://www.kaggle.com/c/dogs-vs-cats/data)
- Unzip the downloaded (zip) file. There are 1000 dog and cat images (500 for each category) in the unzipped folder.
- Use 10-fold cross validation to split the 1000 images into training vs. test set. Train neural networks using training set, and report their performance on the test set.

- Create a feedforward neural network with one or two hidden layers (you can determine the number of hidden nodes for each layer, but the number of hidden nodes for each layer should be at least 50). Train the network on the training set and report its performance on the test. Report at least one loss plot with respect to the number of epochs. Report 10-fold cross validation accuracy, including mean and standard deviation [1 pt]
- For the same network structure created above, please change the network structure by adding batch normalization and/or dropout layer (at different layers). Report 10-fold cross validation performance of the network by adding batch normalization and dropout layer, respectively (you can decide which layer(s) to add batch normalization and/or dropout). Conclude batch normalization and drop-out impact on the neural network performance. [2 pts]