ECCE 635 Deep Learning Systems Design

Fall 2020

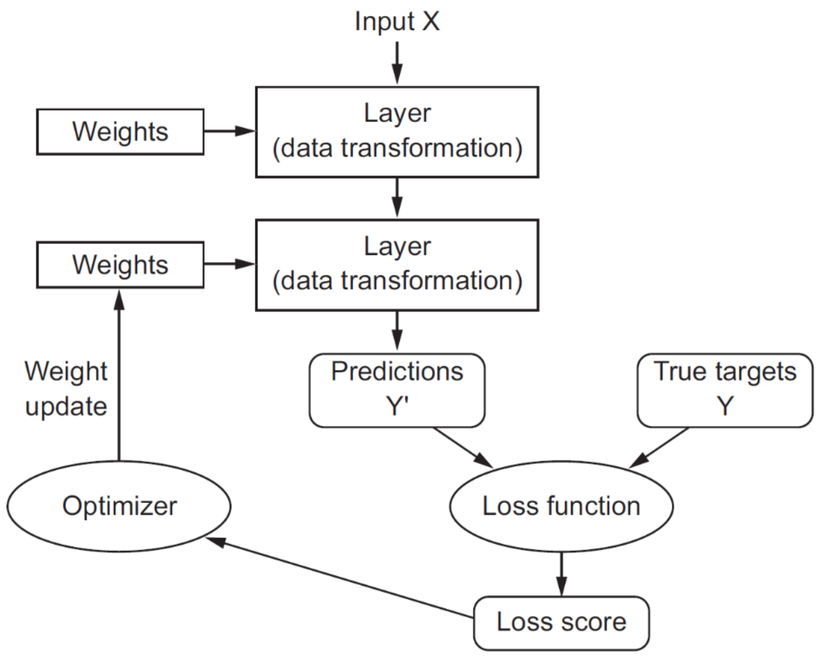
Assignment 1

Due Date: Sunday - September 13, 2020 (12:00 midnight)
Submission: You must submit two files through Blackboard:

- A PDF file containing your writeup titled Student_ID_Name_writeup_Assignment1.pdf
- One code file titled Student_ID_Name_code_Assignment1.py or Student_ID_Name_code_Assignment1.ipynb.

Part 1: Conceptual Questions

1. What is a deep learning model? How is it different from machine learning models?
2. What is the difference between training, validation, and test sets?
3. What is the difference between classification and regression tasks? Give an example.
4. What do we mean by one-hot encoding? Give an example.
5. How does splitting a dataset into training, validation, and test sets help identify overfitting?
6. Training a neural network revolves around different objects, what are they? Provide a brief description of each object. Refer to the below diagram for your answer. List three objects.



Part 2: Programming Assignment

1. Build a function that returns the sigmoid of x. x could be scalar (0D tensor), Vector (1D tensor) or Matrices (2D tensors). The data structures we use in numpy to represent these tensors are called numpy arrays. Use numpy.exp(x) for the exponential function.

Reminder: sigmoid(x) = $\sigma(x) = \frac{1}{1+e^{-x}}$ is sometimes also known as the logistic function. It is a non-linear function used not only in Machine Learning (Logistic Regression), but also in Deep Learning.

<u>Example of Input:</u>

```
(1)  x = np.array(12)
(2)  x = np.array([12,4,2,0])
(3)  x = np.array([[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]])
```

2. Implement the function that compute the gradient of the sigmoid function with respect to its input x. you will need to compute gradients to optimize loss functions using backpropagation.

Reminder: sigmoid derivative (x) = $\grave{\sigma(x)}$ = $\sigma(x)\ (1 - \sigma(x))$

3. Implement the function that calculate the Numpy vectorized version of the *L1* loss. The loss is used to evaluate the performance of your model. The bigger your loss is, the more different your predictions ($\hat{y}$) are from the true values (y). In deep learning, you use optimization algorithms like Gradient Descent to train your model and to minimize the cost.

Reminder: L1 loss is defined as:

$$L_1(\hat{y}, y) = \sum_{i=0}^{m} \left| y^{(i)} - \hat{y}^{(i)} \right|$$

<u>Example of Input:</u>

```
(1)
  yhat = np.array([.3, 0.2, 0.1, .8, .7])
  y = np.array([1, 0, 0, 1, 1])

(2)
  yhat = np.array([0, 0, 0.1, 0.6, 0.2])
  y = np.array([1, 0, 0, 1, 1])
```

4. Implement the function that calculate the Numpy vectorized version of the *L2* loss. The loss is used to evaluate the performance of your model. Use (numpy.dot() ; numpy.dot(x,x) = $\sum_{j=0}^{n} x_j^2$)

Reminder: L2 loss is defined as:

$$L_2(\hat{y}, y) = \sum_{i=0}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

Example of Input:

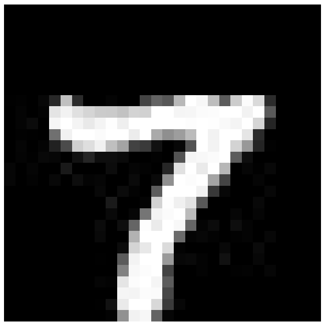
(1)
```
yhat = np.array([.3, 0.2, 0.1, .8, .7])
y = np.array([1, 0, 0, 1, 1])
```

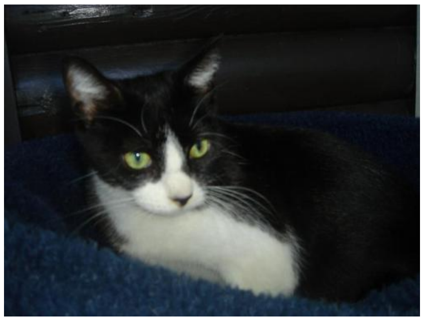
(2)
```
yhat = np.array([0, 0, 0.1, 0.6, 0.2])
y = np.array([1, 0, 0, 1, 1])
```

5. Implement a function using python scikit-image package to read img_1.jpg and img_2.jpg. The function takes an image directory as input and return image Numpy arrays representation. For each image, report image shape and the shape of the vectorize representation of the image. Comment on the main difference between the two images.

Reminder: Two common numpy functions used in deep learning are numpy.shape and numpy.reshape().

- X.shape is used to get the shape (dimension) of a matrix/vector X.
- X.reshape(...) is used to reshape X into some other dimension.


img_1.jpg


img_2.jpg