22AIE304 Deep Learning Lab Sheet 1

Fifth Semester BTech CSE(AI)

Department of Computer Science and Engineering

Amrita School of Computing

Exercise 1: Refresh NumPy

Lab1(Practice)

- Shape and type of the array
- Access specific elements of a 2D NumPy array
- Show how to slice a NumPy array to get all rows, but only the first two columns.
- Reshape a 1D array of size 12 into a 2D array with 3 rows and 4 columns?
- Perform matrix multiplication between two 2D arrays using NumPy.
- Compute the mean, median, and standard deviation of a NumPy array
- Perform vertical and horizontal stacking in NumPy
- Flatten a 3 x 4 NumPy array into a 1D array.
- Generate a random array of size n with values drawn from a normal distribution
- Perform element-wise addition, subtraction, multiplication, and division

Exercise 2: Practice Open CV

Use the practice sheet in the below link to understand basic image Operations in OpenCV and do the following:

2.1 Load an image using OpenCV and display it using Matplotlib

2.2 Resize the image to half of its original size using OpenCV
2.3 Crop a specific region (e.g., the top-left quarter) of an image using NumPy slicing in OpenCV.
2.4 Convert an image from BGR (OpenCV default) to grayscale.
2.5 Display the image using Matplotlib instead of OpenCV
2.6 Find the dimensions (width, height, and channels) of an image using OpenCV
2.7 Flip the image horizontally and vertically using OpenCV.

2.8 Apply a 5x5 averaging (box) filter to smoothen the image.

2.9 Apply a sharpening kernel to enhance the edges and details in the image.

2.10 Apply the Sobel operator to detect edges in both the horizontal and vertical directions.

Exercise 3: Practice PyTorch

3.1 Create two random 3x3 tensors and perform matrix multiplication. Compute the matrix product and use Py Torch's autograd to calculate the gradient of the result with respect to one of the input tensors.

3.2 Perform element-wise operations on tensors with broadcasting. Create a 3x1 tensor and a 1x3 tensor. Use broadcasting to add them and multiply the result by another tensor of shape 3x3.Explore how broadcasting works in PyTorch and understand how it simplifies tensor operations.

3.3 Create a 2D tensor of shape (6, 4) and reshape it into a tensor of shape (3, 8). Extract specific slices from the reshaped tensor (e.g., select all rows but only the first two columns).

3.4 Create a NumPy array, convert it into a PyTorch tensor, perform some operations (e.g., multiplication by a scalar), and convert the result back to a NumPy array.

3.5 Initialize a 5x5 tensor with random values sampled from a uniform distribution between 0 and 1. Initialize another 5x5 tensor with random values sampled from a normal distribution with a mean of 0 and a standard deviation of 1. Multiply the two tensors elementwise. Compute the mean and standard deviation of the resulting tensor. Reshape the result into a 1D tensor of size 25. Compute the sum of all elements in the reshaped tensor.

Exercise 4:

4.1 Build a function that returns the sigmoid of a real number x. Use math.exp(x) for the exponential function.

> **Note:** sigmoid(x)=1/(1+e^-x) is sometimes also known as the logistic function. It is a non-linear function used in Machine Learning (Logistic Regression) and Deep Learning.
>
> use **np.exp()** to Implement the sigmoid function using NumPy. see why np.exp() is preferable to math.exp().

4.2 Implement the function sigmoid_grad() to compute the gradient of the sigmoid function with respect to its input x. The formula is: sigmoid_derivative(x)=σ’(x)=σ(x)(1−σ(x))

4.3 Implement image2vector() that takes an input of shape (length, height, 3) and returns a vector of shape (length*height*3, 1).

4.4 Implement normalizeRows() to normalize the rows of a matrix. After applying this function to an input matrix x, each row of x should be a vector of unit length (meaning length 1).

4.5 Implement the L1 and L2 loss functions:

> L1 loss is defined as:

$$L_1(\hat{y}, y) = \sum_{i=0}^{m-1} |y^{(i)} - \hat{y}^{(i)}|$$

L2 loss is defined as:

$$L_2(\hat{y}, y) = \sum_{i=0}^{m-1} (y^{(i)} - \hat{y}^{(i)})^2$$

Exercise 5: Towards neural network from logistic regression

Build a logistic regression model to classify images as either cat or non-cat.

5.1 Download dataset

5.2 Load and display the first image from the training dataset, print its shape and verify that the image is correctly loaded as an RGB image.

5.3 Implement Logistic regression for image classification