GOVERNMENT OF KERALA

DEPARTMENT OF TECHNICAL EDUCATION

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(GOVT. ENGINEERING COLLEGE)

KOTTAYAM - 686501

20MCA241 Data Science Lab

List of Experiments

Assignment 1
Review of Python Programming

Problem Statement

Write Python code to explore and practice with the basic data types, containers, functions, and classes of Python.

1. Start by creating variables of various numeric data types and assigning them values.
2. Print the data types and values of these variables.
3. Perform mathematical operations on these variables.
4. Update the values of these variables.
5. Create boolean variables with True or False values.
6. Print the data types of these boolean variables.
7. Perform Boolean operations on these boolean variables.
8. Create string variables with text values.
9. Print the contents and lengths of these string variables.
10. Concatenate strings.
11. Format strings with variables.
12. Use string methods to manipulate strings by capitalizing, converting to uppercase, justifying, centering, replacing substrings, and stripping whitespace.
13. Create and use Python lists. Perform tasks like appending elements, indexing, slicing, and iterating through the list.
14. Create and use Python tuples. Perform tasks like indexing, slicing, and concatenation.
15. Create and use Python sets. Perform tasks like accessing, adding, deleting set elements.
16. Create and use Python dictionaries. Perform tasks like adding, updating, and removing key-value pairs, and accessing values.
17. Define simple functions with parameters and return values.
18. Call functions with different arguments and use the returned results.
19. Write functions that accept other functions as arguments.

20. Define and use Python classes. Include tasks like creating a class, defining methods, and creating instances.

21. Implement class inheritance and method overriding.

22. Create a class with class variables and instance variables, and demonstrate their usage.

Use the Jupyter notebook file to experiment with the code.
You can read a lot more about Python classes in the documentation.

Assignment 2
Vectorized Computations using Numpy

Problem Statement

Implement the following computations using NumPy:

1. Create a matrix U of shape (m, n) with input values where m and n are input positive integers.
2. Compute X as the transpose of U.
3. Create a matrix Y of shape (1, m) with random values $\in [0, 1]$.
4. Create a matrix $W1$ of shape (p, n) with random values $\in [0, 1]$ where p is an input positive integer.
5. Create a vector $B1$ of shape (p, 1) with random values $\in [0, 1]$.
6. Create a vector $W2$ of shape (1, p) with all zeros.
7. Create a scalar $B2$ with a random value $\in [0, 1]$.
8. Perform the following computations iteratively 15 times:
   (a) $Z1 = W1 \cdot X + B1$ (Matrix Multiplication)
   (b) $A1 = f(Z1)$ where f is a function that returns 0 for negative values and the input value itself otherwise.
   (c) $Z2 = W2 \cdot A1 + B2$
   (d) $A2 = g(Z2)$ where g is a function defined as $g(x) = \frac{1}{1+e^{-x}}$.
   (e) $L = \frac{1}{2}(A2 - Y)^2$
   (f) $dA2 = A2 - Y$
   (g) $dZ2 = dA2 \circ gprime(Z2)$ where *gprime (x)* is a function that returns $g(x) \cdot (1 - g(x))$ and $\circ$ indicates element-wise multiplication
   (h) $dA1 = W2^T \cdot dZ2$
   (i) $dZ1 = dA1 \circ$ *fprime(Z1)* where *fprime* is a function that returns 1 for positive values and 0 otherwise and $\circ$ indicates element-wise multiplication.
   (j) $dW1 = \frac{1}{m} \cdot dZ1 \cdot X^T$
   (k) $dB1 = \frac{1}{m} \sum dZ1$ (sum along the columns)
   (l) $dW2 = \frac{1}{m} \cdot dZ2 \cdot A1^T$

(m) $dB2 = \frac{1}{m} \sum dZ2$ (sum along the columns)

(n) Update and print $W1$, $B1$, $W2$, and $B2$ for $\alpha = 0.01$:

i. $W1 = W1 - \alpha \cdot dW1$

ii. $B1 = B1 - \alpha \cdot dB1$

iii. $W2 = W2 - \alpha \cdot dW2$

iv. $B2 = B2 - \alpha \cdot dB2$

Reference: Jupyter notebook file.

Check out the numpy reference to find out much more about numpy.

Additionally, you can read the documentation.

Assignment 3
Vectorized Computations using TensorFlow

Problem Statement

Implement the following computations using TensorFlow:

1. Create a matrix U of shape (m, n) with input values where m and n are input positive integers.
2. Compute X as the transpose of U.
3. Create a matrix Y of shape (1, m) with random integer values $\in [0, 9]$.
4. Create a matrix $W1$ of shape (p, n) with random values $\in [0, 1]$ where p is an input positive integer.
5. Create a vector $B1$ of shape (p, 1) with random values $\in [0, 1]$.
6. Create a matrix $W2$ of shape (10, p) with all zeros.
7. Create a scalar $B2$ with a random value $\in [0, 1]$.
8. Perform the following computations iteratively 15 times:
   (a) $Z1 = W1 \cdot X + B1$ (Matrix Multiplication)
   (b) $A1 = ReLU(Z1)$ where $ReLU(x)$ is a function that returns 0 for negative values and the input value itself otherwise.
   (c) $Z2 = W2 \cdot A1 + B2$
   (d) $A2 = softmax(Z2)$ where $softmax(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$
   (e) $dZ2 = A2 - one\_hot\_Y$ where one_hot_Y is the one-hot encoded form of Y.
   (f) $dA2 = W2^T \cdot dZ2$
   (g) $dW2 = \frac{1}{m} \cdot dZ2 \cdot A1^T$
   (h) $dB2 = \frac{1}{m} \sum dZ2$ (sum along the columns)
   (i) $dZ1 = dA2 \circ ReLU\_deriv(Z1)$ where $ReLU\_deriv(x)$ returns 1 for positive values and 0 otherwise, and $\circ$ indicates element-wise multiplication.
   (j) $dA1 = W1^T \cdot dZ1$
   (k) $dB1 = \frac{1}{m} \sum dZ1$ (sum along the columns)
   (l) $dW1 = \frac{1}{m} \cdot dZ1 \cdot X^T$
   (m) Update and print $W1$, $B1$, $W2$, and $B2$ for $\alpha = 0.01$:

i. $W1 = W1 - \alpha \cdot dW1$

ii. $B1 = B1 - \alpha \cdot dB1$

iii. $W2 = W2 - \alpha \cdot dW2$

iv. $B2 = B2 - \alpha \cdot dB2$

Reference: Tensors and Operations official documentation.

Assignment 4
Implementing an FCNN from Scratch using TensorFlow

Problem Statement

Implement the following computations using TensorFlow:

1. Load the the MNIST dataset from tensorflow as x_train, y_train, x_test and y_test.

   The Modified National Institute of Standards and Technology (MNIST) dataset contains grayscale images of handwritten digits. The training set consists of 60,000 images and the test set contains 10,000 images. The label of each image is a digit between 0 and 9. Each image has a size of 28×28, consisting of 784 pixel values, where each pixel value $\in [0, 255]$ with 0 corresponds to black, 255 to white, and values in between representing various shades of gray.

2. Form a matrix U of shape (m, n) using TensorFlow by reshaping the images in x_train to be 1D arrays of 784 (28×28) pixel values (Flatten the images) where $m = 60,000$ is the number of training examples (training images) and $n = 784$ is the number of features (no. of pixel values)

3. Compute X as the transpose of U.

4. Normalize the pixel values of X to $[0, 1]$ by dividing by 255.

5. Form a matrix Y of size m corresponding to the labels $\in [0, 9]$ of images by transposing y_train.

6. Form a matrix V by reshaping the images in x_test to be 1D arrays of 784 (28×28) pixel values (Flatten the images).

7. Compute $Xtest$ as the transpose of V.

8. Normalize the pixel values of $Xtest$ to $[0, 1]$ by dividing by 255.

9. Form a matrix $Ytest$ of size m corresponding to the labels $\in [0, 9]$ of images by transposing y_test.

10. Select an image from X and display it. Also, display the corresponding label from Y.

11. Set the hyper parameters: $p = 10$, the no. of neurons in hidden layer, $q = 10$, the no. of neurons in output layer (corresponding 10 labels in one-hot encoding format), learning rate $\alpha = 0.01$ and the number of training epochs (iterations over the dataset) as 1000.

12. Create a matrix $W1$ of shape (p, n) and initialize it as $W1 = \mathcal{N}(0, 1) \times \sqrt{\frac{1}{n}}$, where $\mathcal{N}(0, 1)$ represents a matrix of random values drawn from a normal distribution with mean 0 and standard deviation 1.

13. Initialize the vector $B1$ of shape $(p, 1)$ to zeros.

14. Initialize the matrix $W2$ of shape (q, p) as $W2 = \mathcal{N}(0, 1) \times \sqrt{\frac{1}{p}}$.

15. Initialize the vector $B2$ of shape $(q, 1)$ to zeros.

16. Perform the following forward propagation and backpropagation computations iteratively (No. of epochs=1000):

    (a) $Z1 = W1 \cdot X + B1$ (Matrix Multiplication)

    (b) $A1 = ReLU(Z1)$ where $ReLU(x)$ is a function that returns 0 for negative values and the input value itself otherwise.

    (c) $Z2 = W2 \cdot A1 + B2$

    (d) $A2 = softmax(Z2)$ where $softmax(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$

    (e) Get the predicted labels from the output of $A2$ (index of the maximum value).

    (f) Find the accuracy of the predictions by comparing them to the true labels Y and print the progress in every 100 epochs.

    (g) Compute the cross-entropy loss using TensorFlow's *tf.nn.softmax_cross_entropy_with_logits* function.

    (h) $dZ2 = A2 - one\_hot\_Y$ where one_hot_Y is the one-hot encoded form of Y.

    (i) $dA2 = W2^T \cdot dZ2$

    (j) $dW2 = \frac{1}{m} \cdot dZ2 \cdot A1^T$

    (k) $dB2 = \frac{1}{m} \sum dZ2$ (sum along the columns)

    (l) $dZ1 = dA2 \circ ReLU\_deriv(Z1)$ where $ReLU\_deriv(x)$ returns 1 for positive values and 0 otherwise, and $\circ$ indicates element-wise multiplication.

    (m) $dA1 = W1^T \cdot dZ1$

    (n) $dB1 = \frac{1}{m} \sum dZ1$ (sum along the columns)

    (o) $dW1 = \frac{1}{m} \cdot dZ1 \cdot X^T$

    (p) Update and print $W1$, $B1$, $W2$, and $B2$ for $\alpha = 0.01$:

        i. $W1 = W1 - \alpha \cdot dW1$

        ii. $B1 = B1 - \alpha \cdot dB1$

        iii. $W2 = W2 - \alpha \cdot dW2$

        iv. $B2 = B2 - \alpha \cdot dB2$

17. Use tensorflow GradientTape() to automatically calculate the gradients from steps (h) to (o) and redo the training steps.

18. Select one test image from $Xtest$, display it, reshape it to $n \times 1$, perform forward propagation computations and predict the label. Check whether the prediction is correct.

19. Use the entire $Xtest$ and perform the forward propagation computations and predict the accuracy of the model.

References:

- FCNN from Scratch
- Tensors and Operations official documentation.
- Xavier Initialization

Assignment 5
Explore Data and Create Linear Regression Model

Problem Statement

Implement the following computations using Pandas and TensorFlow:

1. Load the dataset and import it into a Pandas DataFrame. Use the dataset
2. Display the first five rows and the last three rows of the dataset.
3. Get the dimensions (number of rows and columns) of the dataset.
4. Generate descriptive statistics (mean, median, standard deviation, five-point summary, IQR, etc.) for the data.
5. Print a concise summary of the dataset as information on data types (schema) and missing values.
6. Add a new column named "X22" by converting the "house age" from years to days.
7. Delete the column "X22" from the dataset.
8. Create three new instances synthetically and add them to the dataset.
9. Delete the newly inserted three instances from the dataset.
10. Update the "house price of unit area" to 110, provided it is currently greater than the amount.
11. Find the latitude and longitude of the houses whose prices are less than or equal to 20.
12. Add the missing convenience store values of instances by calculating the average number of convenience stores.
13. Find the normalized distance to the nearest train station by performing:
    (a) Z-score normalization.
    (b) Min-max normalization.
    (c) Decimal scaling.
14. Generate the following basic visualizations using Seaborn. Customize your visualizations by adding titles, labels, legends, and appropriate color schemes.
    (a) Create a histogram for the ''Y house price of unit area" attribute.

(b) Create a box-and-whisker plot for the ‘’Y house price of unit area” attribute.

(c) Create a scatter plot showing house prices against house age.

(d) Add a second scatter plot showing house prices against distance to the nearest MRT station.

15. Form the Design Matrix X of shape $m \times n+1$ in order to apply normal equation method where m is the number of training examples and n is the number of input features. Only use the two normalized input features 'X2 house age' and 'X3 distance to the nearest MRT station' from the dataset as second and third columns respectively and all 1 s as the first column. Also, form output vector Y of shape $m \times 1$.

16. Find the parameter vector W using the normal equation method as $W = (X^T X)^{-1} X^T Y$.

17. Implement the gradient descent algorithm with the following steps.

- Form the Design Matrix X of shape $n \times m$. Only use the two normalized input features 'X2 house age' and 'X3 distance to the nearest MRT station' and the output vector Y of shape $1 \times m$
- Initialize the parameter vector W of shape $1 \times n$ and bias b (scalar).
- Repeat the following steps to a certain number of iterations with learning rate $\alpha = 0.01$, and print the final parameter values.
  (a) Calculate the prediction $\hat{Y} = WX + b$.
  (b) Compute loss $L = \frac{1}{2} \times (\hat{Y} - Y)^2$
  (c) Compute error $E = \hat{Y} - Y$
  (d) Compute the gradient with respect to W as $dW = \frac{1}{m} E \cdot X^T$ and with respect to b as $db = \frac{1}{m} \times E$ (sum over the columns)
  (e) Update $W = W - \alpha dW$ and $b = b - \alpha db$
- Use tensorflow GradientTape() to automatically calculate the gradients in the above step (d) and redo the training steps and print the final parameter values.

18. Define a class to create a Linear Regression model with methods fit and predict. Use the above iterative process to implement the model's training within the fit method.

Reference: Pandas Tutorial.
Implementation of Linear Regression.

Assignment 6
Building Machine Learning Models with Scikit-learn

Problem Statement 1: Classification on the Iris Dataset

1. Load the Iris dataset from Seaborn's dataset module into a Pandas DataFrame.
2. Split the dataset into training and test data.
3. Train and evaluate the following classification models:
   (a) K-Nearest Neighbors (KNN)
   (b) Gaussian Naive Bayes
   (c) Decision Tree
   (d) Random Forest
   (e) Support Vector Machine (SVM)
4. For each model:
   (a) Print the labels predicted by the model on the test data and compare them with the actual labels.
   (b) Form and display the confusion matrix using the test data.

Problem Statement 2: K-Means Clustering on the Wine Dataset

5. Load the Wine dataset from Scikit-learn's dataset module into a Pandas DataFrame.
6. Preprocess the data by selecting relevant features and handling any missing values if present.
7. Apply K-Means clustering to the dataset with the number of clusters equal to the number of unique wine classes in the dataset.
8. Print the cluster labels assigned to each data point by the K-Means algorithm.
9. Compare the cluster labels with the actual wine class labels by calculating the accuracy of the clustering.
10. Visualize the clusters using a scatter plot or pairplot with different colors representing different clusters.

Problem Statement 3: Linear Regression on the California Housing Dataset

11. Load the California Housing dataset from Scikit-learn's dataset module into a Pandas DataFrame.
12. Preprocess the data by selecting relevant features and handling any missing values if present.
13. Split the dataset into training and test data.
14. Train a Linear Regression model on the training data.
15. Evaluate the model by predicting housing prices on the test data and calculating performance metrics such as Mean Squared Error (MSE) and R-squared.
16. Print the coefficients of the model to understand the influence of each feature on the target variable.