Machine Learning Bootcamp Report

Outline

- Overview
- Problem Statement
- Brief Description
  - Implementations
  - Problems Faced in algorithm
  - Proposed Solution to overcome the issue
- Performance testing
- Code Link
- Resources

Overview

Student: Surya Prakash Tejasvee
Mob. No: 7004273032
Email: 23je0998@iitism.ac.in

Mentor: Anant Upadhyay & Yogita Singh
Club: Cyberlabs (IIT ISM Dhanbad)
Division: Machine Learning

Problem Statement

Developed a Machine Learning Algorithm Library encompassing sophisticated implementations of Linear (including Polynomial) Regression, Logistic Regression, K-Nearest Neighbors (KNN), K-Means Clustering, and an n-layer Neural Network, all from scratch.

Brief Description

---

1. **Linear Regression Algorithm :**
    - Supervised learning algorithm
    - Establish a linear relationship between the input features and the target variable, enabling the prediction of the target variable for new, unseen data.
    - For multiple linear regression with n features:

      $Y = b_0X^0 + b_1X_1 + b_2X_2 + \ldots\ldots + b_nX_n$

        - Where b_0 : bias term
        - b_i (i from 1 to n) : weights
    - Cost Function (MSE) :

      $$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

    - Gradient :

      $$\frac{\partial J}{\partial \theta_1} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

2. **Polynomial Regression Model :**
    - Supervised learning algorithm
    - Finds the best-fitting polynomial curve that captures the underlying relationship between the input features and the target variable.
    - Equation of the form :

      $$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \ldots + \beta_n X^n$$

    - Cost Function :

      $$J(\beta_0, \beta_1, \ldots, \beta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\beta(x^{(i)}) - y^{(i)})^2$$

    - Gradient :

      $$\frac{\partial J}{\partial \beta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\beta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

    - Regularisation term :

      $$\lambda \sum_{j=1}^{n} \beta_j^2$$

3. **Logistic Regression Algorithm :**
   - Supervised learning algorithm and classification algorithm
   - Predicts the probability of an instance belonging to the positive class (usually labelled as 1).
   - Uses two functions :
     - Sigmoid Function : Generally for binary class classification

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

     - Softmax Function : For multi-class classification

$$\text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

   - Here, z = $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$
   - Cost Function :

$$J(\beta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(\hat{p}^{(i)}) + (1-y^{(i)})\log(1-\hat{p}^{(i)})\right]$$

   - Gradient :

$$\frac{\partial J}{\partial \beta_j} = \frac{1}{m}\sum_{i=1}^{m}(\hat{p}^{(i)} - y^{(i)})x_j^{(i)}$$

   - Regularisation term :

$$\frac{\lambda}{2m}\sum_{j=1}^{n}\beta_j^2$$

4. **K-Nearest Neighbors (KNN) Algorithm :**
   - Supervised learning algorithm
   - Used for both classification and regression tasks.
   - Finds the K nearest points in the training dataset and uses their class to predict the class or value of a new data point.
   - Euclidean Distance :

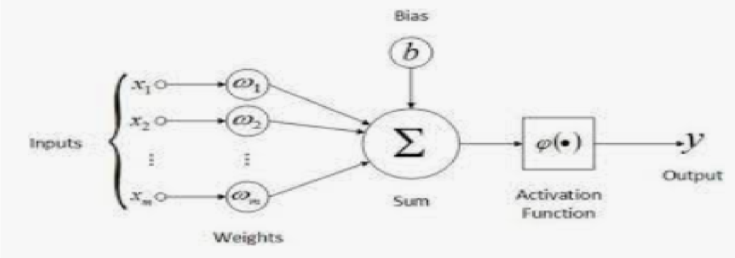
$$\text{Distance} = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2}$$

   - Manhattan Distance :

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i|$$

5. **K-means Clustering Algorithm :**
    - Unsupervised learning algorithm
    - Computes centroids and repeats until the optimal centroid is found
    - The number of clusters found from data by the method is denoted by the letter 'K' in K-means.
    - Mathematically, assign each data point x_i to the cluster j for which $||x_i - \mu_j||^2$ is minimized, where μ_j is the centroid of cluster j.
    - Update Centroids:

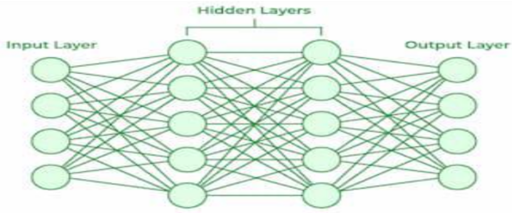
$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

6. **Deep Learning (Neural-Network) Algorithm :**
    - Can be used for both supervised and unsupervised learning tasks
    - Used for various tasks, including pattern recognition, classification, regression, and more
    - Consist of interconnected nodes or neurons that process and learn from data, enabling tasks such as pattern recognition and decision making
    - Hidden Layers: Each hidden layer neuron processes inputs by multiplying them by weights, adding them up, and then passing them through an activation function.
    - Activation Function: Model non-linearity is introduced by activation functions, which enables the network to recognize intricate patterns.
        - ReLU: f(x) = max(0,x)
        - 

    - Cost Function :

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2$$

    - Working of a Neural Network:



Implementations

---

1. **Multiple Linear Regression:**
    - Importing Libraries:
    - Reading .csv file
    - Setting the seed value and shuffling the data row-wise
    - Converting pandas dataframe to numpy array
    - Split the data into features and target variables
    - Adding column of ones in features for bias term
    - Splitting the data in (80:20) for cross validation
    - Setting the hyperparameters (no. of iterations & learning rate)
    - Defining function for weight initialization, cost calculation and gradient descent
    - Checking R-2 Score
    - Predicting value of new dataset

2. **Polynomial Regression:**
    - Importing Libraries:
    - Reading .csv file
    - Setting the seed value and shuffling the data row-wise
    - Converting pandas dataframe to numpy array
    - Split the data into features and target variables
    - Generating polynomial function up to degree n along with interaction terms between features
    - Normalising the features for smooth performance
    - Adding column of ones in features for bias term
    - Splitting the data in (80:20) for cross validation
    - Setting the hyperparameters (lambda, no. of iterations & learning-rate)
    - Defining function for weight initialization, cost calculation, regularisation and gradient descent
    - Checking R-2 Score
    - Predicting value of new dataset

3. **Logistic Regression:**
    - Importing Libraries:
    - Reading .csv file
    - Setting the seed value and shuffling the data row-wise
    - Converting pandas dataframe to numpy array
    - Split the data into features and target variables
    - Normalising the features for smooth performance
    - Adding column of ones in features for bias term
    - Generating multiple classes for target variable
    - Splitting the data in (80:20) for cross validation
    - Setting the hyperparameters (lambda, no. of iterations & learning-rate)
    - Defining Softmax function for predicting probabilities for different classes
    - Defining function for weight initialization, cost calculation and gradient descent
    - Checking Accuracy
    - Predicting value of new dataset

4. **K-Nearest Neighbors (KNN):**
    - Importing Libraries:
    - Reading .csv file
    - Setting the seed value and shuffling the data row-wise
    - Converting pandas dataframe to numpy array
    - Split the data into features and target variables
    - Normalising the features for smooth performance
    - Splitting the data in (80:20) for cross validation
    - Defining function for Euclidean distance for calculating distance between train data and test point.
    - Setting the hyperparameter (K)
    - Defining function for calculating target of K nearest neighbors of test point and predicting most occurred value.
    - Checking Accuracy
    - Predicting value of new dataset

5. **K-means Clustering:**
    - Importing Libraries:
    - Reading .csv file
    - Converting pandas dataframe to numpy array
    - Normalising the features for smooth performance
    - Setting the hyperparameter (K)
    - Defining function for calculating K centroids of K clusters and labelling them.

6. **Neural Network:**
    - Importing Libraries:
    - Reading .csv file
    - Setting the seed value and shuffling the data row-wise
    - Converting pandas dataframe to numpy array
    - Split the data into features and target variables
    - Normalising the features for smooth performance
    - Adding column of ones in features for bias term
    - Generating multiple classes for target variable
    - Splitting the data in (80:20) for cross validation
    - Setting the hyperparameters (lambda, no. of iterations & learning-rate)
    - Defining Activation Function (Relu)
    - Defining Softmax function for predicting probabilities for different classes
    - Defining Forward Propagation function for interconnecting input, hidden and output layers.
    - Defining Backward Propagation function for training the model.
    - Defining cost function, regularisation and gradients in Backward Propagation function
    - Checking Accuracy
    - Predicting value of new dataset

Problems Faced in Algorithm

---

1. **Code Error:**
   - Index Error: Upon using numpy attributes without converting pandas dataframe to numpy arrays.

```
InvalidIndexError                               Traceback
Cell In[8], line 1
----> 1 features = df[:, 1:]

File ~\AppData\Local\Programs\Python\Python312\Lib\s
Frame.__getitem__(self, key)
   3891 if self.columns.nlevels > 1:
   3892     return self._getitem_multilevel(key)
-> 3893 indexer = self.columns.get_loc(key)
   3894 if is_integer(indexer):
   3895     indexer = [indexer]
```

   - Dimensional Error: Upon doing dot product between X and error of different dimensions.

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[31], line 3
      1 iteration = 2000
      2 learning_rate = 0.06
----> 3 theta, cost_list = linear_regression(X_train, y_train, alpha = learning_rate, itr =ite
n)

Cell In[30], line 14, in linear_regression(X, y, alpha, itr)
     11 cost_list.append(cost)
     13 # Gradient Descent
---> 14 gradient = np.dot(X,error) / m
     15 theta = theta-alpha * gradient
     18 if (i+1) % 100 == 0 or i==0:

ValueError: shapes (40000,21) and (40000,1) not aligned: 21 (dim 1) != 40000 (dim 0)
```

2. **Unexpected Variation in Cost:**
    - Experienced negative cost upon iteration in Logistic Regression
    - Cost continuously increasing and decreasing upon iteration in Neural-Network

3. **Setting the hyperparameters(alpha & lambda):**
    - Setting the hyperparameters and then going through the iterations and if accuracy is less then again setting the hyperparameters was a bit difficult.

4. **Setting the size of hidden layers:**
    - Finding the appropriate size of hidden layers for good accuracy was also a bit difficult as continuously checking different hidden layers for different accuracies.

5. **Low Performance:**
    - Low R-2 Score:
        - In case of Polynomial Regression for including only individual terms up to degree *n.*
    - Low Accuracy:
        - In case of Logistic Regression upon using sigmoid function to predict the values.
        - In the case of Neural-Network upon using tanh as an activation function.

Proposed Solution to overcome the issue

---

1. To avoid dimensional errors, I subsequently printed 'shape' of variables to take a continuous look into it.

2. Taking interaction terms into account along with individual terms, the low performance issue of Polynomial Regression was resolved.

3. Upon using the softmax function after encoding the target variable into multi-class, I was able to overcome cost variation and low R-2 score problems in logistic regression.

4. Upon using ReLU as 'Activation Function', cost variation & low performance issues of Neural-Network were resolved.

5. With continuous tests, I found decent hyperparameters for my models.

Performance Testing

---

1. **Linear Regression:**
   R-2 Score: 0.9999999999246252

2. **Polynomial Regression:** (Degree=4)
   R-2 Score: 0.9102795240143028

3. **Logistic Regression:**
   Accuracy: 95.82%

4. **K-Nearest Neighbors (KNN):** (K=3)
   Accuracy: 98.00%

5. **K-means Clustering:** (Different clusters along with their centroids are plotted)

6. **Neural Network:**
   Accuracy: 93.33%

Code Link

GitHub Repository Link

.

Resources

---

1. Coursera Course: "Machine Learning Specialization"
2. YouTube Videos: (@Siddhardhan and @CodingLane)
3. Book: "Machine Learning with Python by Cookbook Publication"