Experiment - 8

6/4/23

Implementation of Learning Algorithms.

Aim: To implement and apply learning algorithms for real world problems.

Algorithm:

1. collect the dataset and split it into two parts → training and testing sets.
2. Normalize or standardize the input
3. Initialize the parameters of the logistic regression model.
4. Implement sigmoid function to transform the linear regression O/P into probability value between 0/1.
5. Use the likelihood function to calculate the cost (loss) of the model.
6. use gradient descent algorithm to minimize the cost function and optimize cost as well as parameters (weights sets.)

⑦ Use trained model, evaluate the testing dataset and calculate output using standardised metrics.

⑧ Regularize model using L1, L2 regularisation to prevent overfitting.

Result: Learning algorithms were implemented and applied successfully.

```python
import numpy as np
import matplotlib.pyplot as plt

# Generate some sample data
np.random.seed(0)
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define the linear regression model
class LinearRegression:
    def __init__(self, lr=0.01, n_iters=1000):
        self.lr = lr
        self.n_iters = n_iters
        self.weights = None
        self.bias = None

    def fit(self, X, y):
        n_samples, n_features = X.shape

        # Initialize weights and bias
        self.weights = np.zeros((n_features, 1))
        self.bias = 0

        # Gradient descent
        for _ in range(self.n_iters):
            y_pred = np.dot(X, self.weights) + self.bias
            dw = (1 / n_samples) * np.dot(X.T, (y_pred - y))
            db = (1 / n_samples) * np.sum(y_pred - y)

            # Update weights and bias
            self.weights -= self.lr * dw
            self.bias -= self.lr * db

    def predict(self, X):
        y_pred = np.dot(X, self.weights) + self.bias
        return y_pred

# Train the model using the training data
lr = LinearRegression()
```

Manual calculation / Output :

```
from sklearn.linear_mode
import logistic regression
classification.score (x_test, y_test)
```

⇒ 0.9768

```
(m = confusion_matrix ( y_test, y_pre
print (m)
```

$$\begin{bmatrix} [13 & 0 & 0] \\ [0 & 15 & 1] \\ [0 & 0 & 9] \end{bmatrix}$$