|  |  |
| --- | --- |
|  | **DEPARTMENT OF COMPUTER ENGINEERING** |

Experiment No. 06

|  |  |
| --- | --- |
| Semester | B.E. Semester VIII – Computer Engineering |
| Subject | Deep Learning Lab |
| Subject Professor In-charge | Prof. Kavita Shirsat |
| Academic Year | 2024-25 |

|  |  |
| --- | --- |
| Student Name | Deep Salunkhe |
| Roll Number | 21102A0014 |

**Title:** Batch Gradient Descent

 **Overview**  
This program implements **Batch Gradient Descent (BGD)** for linear regression using a dataset provided in a CSV file. The goal is to find optimal weights (theta values) that minimize the Mean Squared Error (MSE).

 **Key Components of the Code**

* **Function: printThetas(vector<float>& thetas)**
  + Prints the current theta values (weights).
* **Function: computeMSE(vector<float>& y\_cap, vector<float>& target)**
  + Computes Mean Squared Error (MSE).
  + Formula:

MSE = (1/N) \* Σ (y\_actual - y\_predicted)^2

where N is the number of data points.

* **Function: gradientDescent(vector<float>& thetas, vector<vector<float>>& data, vector<float>& target, int tc, int tl, float lr, int epochs)**
  + Performs batch gradient descent:
    1. **Prediction Step:**

y\_cap[i] = theta\_0 + theta\_1 \* x1 + theta\_2 \* x2 + ... + theta\_n \* xn

* + 1. **Gradient Update Rule:**

theta\_j = theta\_j - (lr \* (1/N) \* Σ (y\_actual - y\_predicted) \* x\_j)

* + - * lr = Learning Rate
      * N = Number of training samples
* **Function: readCSV(string filename, vector<float>& target)**
  + Reads the CSV file, extracts feature values and target labels.

 **Main Execution (main())**

* Prompts user for CSV file path.
* Reads data and initializes weights (theta values) to zero.
* Accepts **learning rate (lr)** from the user.
* Runs **gradient descent for 10 epochs** and prints updated weights and MSE after each epoch.

 **Expected Behavior**

* If learning rate is **too high**, weights may diverge, leading to inf values.
* If learning rate is **too low**, convergence may be too slow.
* A properly tuned learning rate leads to decreasing MSE.

**Implementation:**

**#include** <iostream>

**#include** <vector>

**#include** <fstream>

**#include** <sstream>

**#include** <cmath>

**using** **namespace** std;

**void** printThetas(vector<**float**>**&** thetas) {

**for** (**float** theta : thetas) {

        cout **<<** theta **<<** " ";

    }

    cout **<<** endl;

}

**float** computeMSE(vector<**float**>**&** y\_cap, vector<**float**>**&** target) {

**float** mse **=** 0.0;

**int** n **=** target.size();

**for** (**int** i **=** 0; i **<** n; i**++**) {

        mse **+=** pow(target**[**i**]** **-** y\_cap**[**i**]**, 2);

    }

**return** mse **/** n;

}

**void** gradientDescent(vector<**float**>**&** thetas, vector<vector<**float**>>**&** data, vector<**float**>**&** target, **int** tc, **int** tl, **float** lr, **int** epochs) {

**for** (**int** epoch **=** 1; epoch **<=** epochs; epoch**++**) {

        vector**<float>** y\_cap(tl);

**for** (**int** i **=** 0; i **<** tl; i**++**) {

**float** c\_y\_cap **=** thetas**[**0**]**;

**for** (**int** j **=** 0; j **<** tc **-** 1; j**++**) {

                c\_y\_cap **+=** data**[**i**][**j**]** **\*** thetas**[**j **+** 1**]**;

            }

            y\_cap**[**i**]** **=** c\_y\_cap;

        }

**for** (**int** i **=** 0; i **<** tc; i**++**) {

**float** sum **=** 0.0;

**for** (**int** j **=** 0; j **<** tl; j**++**) {

**if** (i **==** 0) {

                    sum **+=** target**[**j**]** **-** y\_cap**[**j**]**;

                } **else** {

                    sum **+=** (target**[**j**]** **-** y\_cap**[**j**]**) **\*** data**[**j**][**i **-** 1**]**;

                }

            }

            thetas**[**i**]** **-=** (lr **\*** (**-**sum **/** tl));

        }

**float** mse **=** computeMSE(y\_cap, target);

        cout **<<** "Epoch " **<<** epoch **<<** " MSE: " **<<** mse **<<** endl;

        cout **<<** "Weights: ";

        printThetas(thetas);

    }

}

vector<vector<**float**>> readCSV(string filename, vector<**float**>**&** target) {

    vector**<**vector**<float>>** data;

    ifstream file(filename);

    string line;

    // Read and discard the first line (feature names)

    getline(file, line);

**while** (getline(file, line)) {

        stringstream ss(line);

        vector**<float>** row;

        string value;

**while** (getline(ss, value, ',')) {

            row.push\_back(stof(value));

        }

        target.push\_back(row.back());

        row.pop\_back();

        data.push\_back(row);

    }

**return** data;

}

**int** main() {

    string filename;

    cout **<<** "Enter CSV file path: ";

    cin **>>** filename;

    vector**<float>** target;

    vector**<**vector**<float>>** data **=** readCSV(filename, target);

**int** tl **=** data.size();

**int** tc **=** data**[**0**]**.size() **+** 1;

    vector**<float>** thetas(tc, 0.0);

**float** lr;

**int** epochs **=** 10;

    cout **<<** "Enter learning rate: ";

    cin **>>** lr;

    gradientDescent(thetas, data, target, tc, tl, lr, epochs);

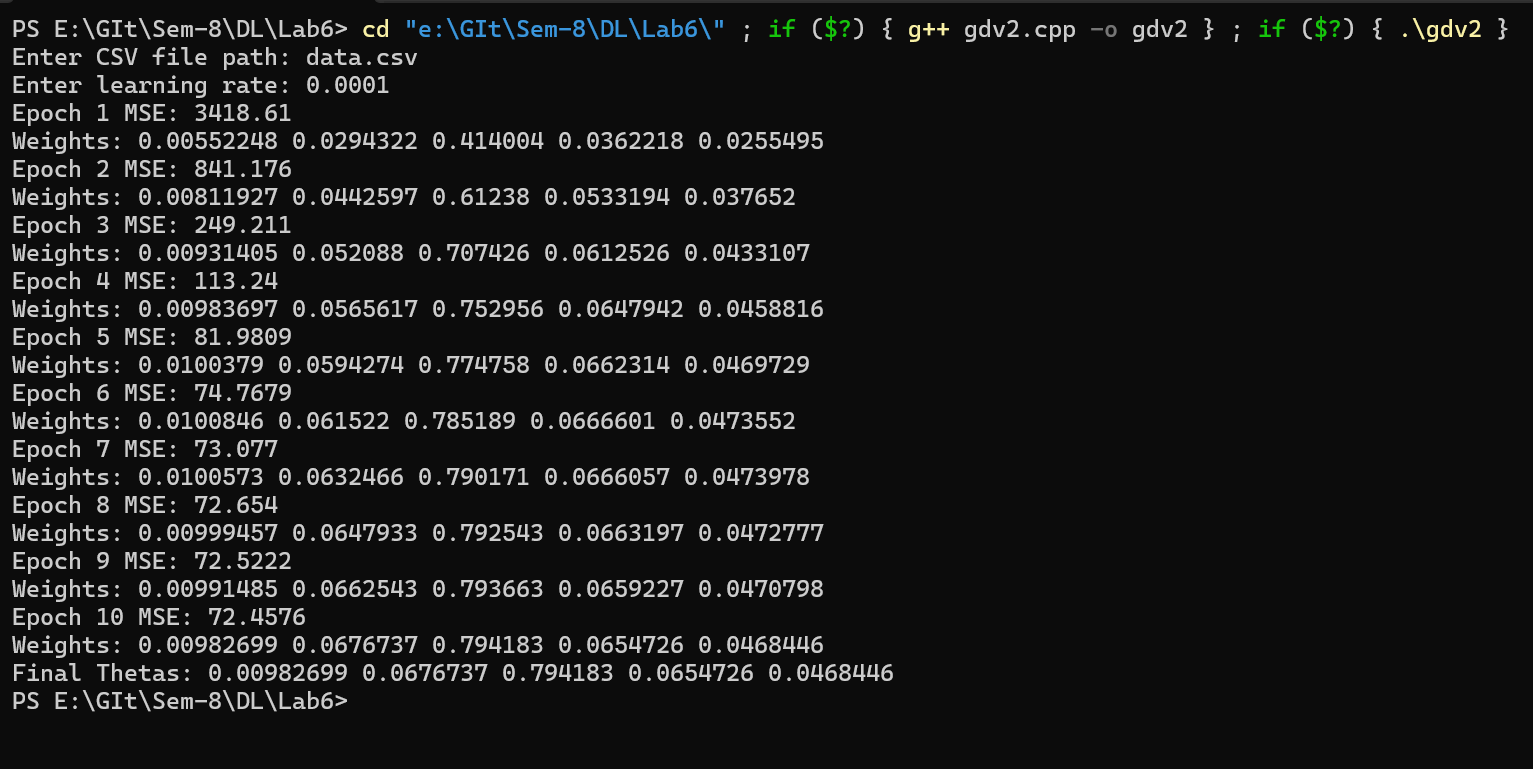
    cout **<<** "Final Thetas: ";

    printThetas(thetas);

**return** 0;

}

**Output:**

****

**Conclusion from Output of Batch Gradient Descent**

1. **Initial MSE and Rapid Decrease:**
   * The initial **Mean Squared Error (MSE)** starts at **3418.61**, indicating a high error at the beginning.
   * As training progresses, MSE drops significantly, reaching **72.4576** by epoch 10.
   * This suggests that the model is **learning effectively** and adjusting the weights to reduce prediction error.
2. **Weight (Theta) Convergence:**
   * The theta values (**weights**) start from **0.0** and gradually update.
   * The weights **increase sharply** in the initial epochs but **stabilize** towards the end.
   * The final theta values:

0.00982699, 0.0676737, 0.794183, 0.0654726, 0.0468446

* + The small updates in later epochs suggest the model is nearing **convergence**.

1. **Learning Rate Impact:**
   * The chosen learning rate (**0.0001**) is **slow but stable**, leading to a smooth decrease in MSE.
   * A **higher learning rate** might have resulted in faster convergence but could also risk divergence.
   * A **lower learning rate** would slow convergence further.
2. **Final Observations:**
   * The model **successfully reduced error** over 10 epochs.
   * However, **MSE is still 72.4576**, which may indicate that:
     + More training epochs might further reduce error.
     + Feature scaling (normalization) might improve efficiency.
     + A slightly higher learning rate could speed up convergence.