MACHINE LEARNING ASSIGNMENT

SUBJECT: MACHINE LEARNING CSC602

GROUP MEMBERS:

NAME	REGISTRATION NUMBER
ANUSHTUP GANGULY	978
SAYAN MANDAL	1004

PROBLEM STATEMENT:

Topic: Handwritten Digit Classification using Machine Learning Based Classifier.

The study utilizes the MNIST dataset, a widely used benchmark dataset for handwritten digit classification.

What is MNIST?

A set of 70000 small images of digits handwritten by high school students & employees of the US Census Bureau.

- All images are labelled with the respective digit they represent.
- There are 70000 images & each image has 784 features.
- Each image is 28 x 28 pixels, and each feature simply represents one pixel's intensity from 0 (white) to 255 (black).

ALGORITHMS USED:

We have used four algorithms in this project for handwritten digit classification.The four algorithms are **Custom Logistic Regression Model using Batch Gradient Descent(Logistic Regression - Part 1 (BGD)), Decision Tree, Naive Bayes Classifier, Logistic Regression model with Stochastic Average Gradient Descent(Logistic Regression - Part 2 (SGD)).**

Custom Logistic Regression Model using Batch Gradient Descent(Logistic Regression - Part 1 (BGD)):

This is a manually implemented logistic regression model using batch gradient descent.

Key Components:

1. **Sigmoid Function**

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

2. **Prediction Function**

$$\hat{y} = \sigma(Wx + b)$$

3. **Loss Function (Binary Cross-Entropy)**

$$L = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

4. **Batch Gradient Descent Updates**

$$W := W - \alpha \cdot \nabla_W$$
$$b := b - \alpha \cdot \nabla_b$$

Where gradients are calculated as:

$$\nabla_W = \frac{1}{n} X^T (\hat{y} - y)$$
$$\nabla_b = \frac{1}{n} \sum (\hat{y} - y)$$

Decision Tree:

Overview:

A Decision Tree builds a model in the form of a tree structure. It splits the dataset into subsets based on the value of input features. This continues recursively, creating branches until a stopping criterion is met, such as when all samples in a node belong to the same class or a maximum tree depth is reached.

Key Concepts:

- Root Node: Represents the entire dataset.

- Decision Nodes: Internal nodes where decisions/splits are made.

- Leaf Nodes: Terminal nodes representing predicted class labels.

- Impurity Measures: Gini Index and Entropy are used to evaluate the quality of splits.

Mathematical Formulas:

1. Gini Impurity:

$$Gini(D) = 1 - \sum_{i=1}^{n} p_i^2$$

Where p_i is the probability of class i in dataset D.

2. Information Gain (using Entropy):

$$Entropy(D) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

$$Gain(D, A) = Entropy(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} Entropy(D_v)$$

Where:

- D is the dataset,
- A is an attribute,
- D_v is the subset of D where attribute $A = v$.

Naive Bayes Classifier:

Overview:

Naive Bayes is a probabilistic classifier based on Bayes' Theorem. It assumes independence between features given the class label — an assumption that is "naive" but often effective in practice.

It is particularly useful for high-dimensional datasets and text classification problems.

1. Bayes' Theorem

The foundation of Naive Bayes is **Bayes' Theorem**:

$$P(y|x) = \frac{P(x|y) \cdot P(y)}{P(x)}$$

Where:

- $P(y|x)$: Posterior probability of class y given the input features x
- $P(x|y)$: Likelihood of features x given class y
- $P(y)$: Prior probability of class y
- $P(x)$: Marginal probability of features x (acts as a normalization constant)

2. Naive Assumption (Conditional Independence)

Assuming features $x_1, x_2, ..., x_n$ are conditionally independent given class y:

$$P(x|y) = \prod_{i=1}^{n} P(x_i|y)$$

So the **posterior** becomes:

$$P(y|x_1, x_2, ..., x_n) = \frac{P(y) \cdot \prod_{i=1}^{n} P(x_i|y)}{P(x)}$$

For classification, we only need the **numerator**, as $P(x)$ is constant across all classes:

$$\hat{y} = \arg\max_y \ P(y) \cdot \prod_{i=1}^{n} P(x_i|y)$$

3. Gaussian Naive Bayes (for continuous data)

If the feature x_i follows a **normal distribution**, then:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \cdot \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Where:

- μ_y and σ_y^2 are the **mean** and **variance** of feature x_i for class y

Logistic Regression model with Stochastic Average Gradient Descent(Logistic Regression - Part 2 (SGD))

Overview:

This implementation of logistic regression uses the Stochastic Average Gradient (SAG) Descent optimization algorithm. Unlike traditional batch gradient descent, SAG improves convergence speed by combining the benefits of both batch and stochastic methods. It maintains an average of gradients computed on individual samples, offering faster and more stable convergence.

Key Concepts:

- **Sigmoid Function**:

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

- **Prediction Function**:

$$\hat{y} = \sigma(Wx + b)$$

- **Loss Function (Binary Cross-Entropy)**:

$$L = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

- **Stochastic Average Gradient Update Rule**:

  At each iteration:

  - Randomly pick a training example (x_i, y_i)
  - Compute its gradient and update the stored gradient for that sample
  - Update the weights and bias using the average of all stored gradients:

$$W := W - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} g_i$$

$$b := b - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)$$

  Where:

  - α is the learning rate
  - g_i is the gradient of the i^{th} training example
  - The method uses memory to keep track of gradients from each sample

Confusion Matrix and Evaluation Metrics (Classification Report)

To evaluate the performance of each classifier, we use the confusion matrix along with precision, recall, and F1-score. These metrics provide a deeper insight into how well the model performs across different digit classes (0 to 9).

Confusion Matrix:

The confusion matrix for a multi-class classification problem is a square matrix of size $n \times n$, where $n = 10$ for MNIST (digits 0–9).

Each element $C_{i,j}$ in the matrix represents the number of instances of class i that were predicted as class j.

For each classifier:

- Let y_{true}: Ground truth labels
- Let $\hat{y}_{\text{model}}$: Predicted labels

Confusion Matrices for each classifier:

- **Logistic Regression (Batch GD):**

$$\text{Confusion\_Matrix}_{\text{log\_batch}} = \text{confusion\_matrix}(y_{\text{true}}, \hat{y}_{\text{log\_batch}})$$

- **Logistic Regression (SGD - SAGA):**

$$\text{Confusion\_Matrix}_{\text{log\_sgd}} = \text{confusion\_matrix}(y_{\text{true}}, \hat{y}_{\text{log\_sgd}})$$

- **Naïve Bayes Classifier:**

$$\text{Confusion\_Matrix}_{\text{nb}} = \text{confusion\_matrix}(y_{\text{true}}, \hat{y}_{\text{nb}})$$

- **Decision Tree Classifier:**

$$\text{Confusion\_Matrix}_{\text{dt}} = \text{confusion\_matrix}(y_{\text{true}}, \hat{y}_{\text{dt}})$$

Precision, Recall and F1-Score

Let:

- TP_i: True Positives for class i
- FP_i: False Positives for class i
- FN_i: False Negatives for class i

Then:

- **Precision**:

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i}$$

- **Recall**:

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i}$$

- **F1-score**:

$$F1_i = 2 \times \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

CLICK HERE FOR GOOGLE COLLAB NOTEBOOK ACCESS

RESULTS:

We can see that Logistic Regression model with Stochastic Average Gradient Descent(Logistic Regression - Part 2 (SGD)) gives the best possible prediction while Custom Logistic Regression Model using Batch Gradient Descent(Logistic Regression - Part 1 (BGD)) gives the worst prediction.

The Training and Testing accuracy for Custom Logistic Regression Model using Batch Gradient Descent(Logistic Regression - Part 1 (BGD)) is 11.10% each.

The Training and Testing accuracy for Decision Tree Algorithm is 100.00% and 87.55% respectively.

The Training and Testing accuracy for Naive-Bayes Classifier is 53.64% and 52.40% respectively.

The Training and Testing accuracy for Logistic Regression model with Stochastic Average Gradient Descent(Logistic Regression - Part 2 (SGD)) is 94.09% and 92.64% respectively.

Discussions (Limitations and Future Scope)

Limitations:

Naive Bayes Assumptions: The assumption of feature independence is violated in image data, leading to poor performance.

Lack of Deep Learning Models: The project uses traditional ML classifiers, which limits the potential accuracy compared to Convolutional Neural Networks (CNNs) / Artificial Neural Networks (ANNs).

Custom Logistic Regression Model using Batch Gradient Descent(Logistic Regression - Part 1 (BGD)): A **simple linear model** is not expressive enough for this high-dimensional task. This is why it is giving the worst accuracy.

Future Scope of Work:

Deep Learning Integration: Employing ANNs/CNNs for better feature extraction and classification performance.

Data Augmentation: Applying rotation, scaling, and translation to make models robust to variations.

Deployment: Create a web or mobile application interface for real-time digit recognition (using frameworks like Streamlit).