Machine Learning Assignment Report

PARNA PRATIM MITRA(G24AIT2015) & MD AZAM(G24AIT2073)

Q1: Diabetes Dataset Model Comparison – Summary

1. Using PCA and cumulative explained variance, it was found that out of 8 components, 7 accounted for 95% of the variance.

2. Two models were chosen: Logistic Regression and Random Forest Classifier, as the dataset was a classification problem indicating whether a patient had diabetes or not.

3. After data cleaning, training, and testing, classification metrics were calculated, including:
   - Accuracy
   - R-squared (R^2)
   - Mean Squared Error (MSE)
   - Mean Absolute Error (MAE)

4. **Comparison Report:**
   - **Logistic Regression**
     - **Full Data:** Accuracy = 77.56%, R^2 = 0.0035
     - **Reduced Data:** Accuracy = 76.77%, R^2 = -0.0315
     - **Observation:** Slight drop in accuracy and R^2 after dimensionality reduction, indicating minor loss of predictive power.
   - **Random Forest Regression**
     - **Full Data:** Accuracy = 75.20%, R^2 = -0.1014
     - **Reduced Data:** Accuracy = 71.65%, R^2 = -0.2587
     - **Observation:** Significant drop in accuracy and R^2, suggesting that Random Forest struggles more with reduced features.

5. Logistic Regression performed better with PCA compared to the Random Forest Classifier.

Q2: K-Means Clustering on MNIST Fashion Dataset - Summary

1. The task was to perform K-Means clustering on the MNIST fashion dataset.

2. The dataset contained 60,000 rows and 785 columns. The first column represented the actual class label, while the remaining 784 columns represented the pixel values of a 28x28 image.

3. The data was clustered into 10 groups. Initially, 10 random cluster centers were chosen, and the K-Means algorithm was applied.

4. In the second approach, 10 cluster centers were chosen from each of the 10 different classes before applying K-Means.

5. PCA was used to reduce the dimensionality to 2D for visualization of both clustering approaches.

6. Comparing inertia values for both methods showed that clustering in the second approach was better because the initial random points in the first approach were not as effective.

Q3: Neural Networks. - Summary

1) Data Loading and Preprocessing

The `load_mnist()` function loads and preprocesses the MNIST dataset using OpenML or Keras as a fallback. It normalizes the pixel values to the range [0, 1], reshapes each image into a 784-dimensional vector, and splits the dataset into training and testing sets.

2) Parameter Initialization

The `initialize_parameters` function sets up weights (W_1, W_2) and biases (b_1, b_2) for a 3-layer neural network:

- **Input to Hidden Layer:** W_1 of shape `(input_size, hidden_size)` is initialized with small random values.
- **Hidden to Output Layer:** W_2 of shape `(hidden_size, output_size)` is initialized similarly.
- **Biases:** b_1, b_2 are initialized to zeros.

These parameters are used during forward and backward propagation in the NumPy-based implementation.

3) Training the Neural Network

The `train_nn` function trains the network with the following steps:

1. **Initialization:** Parameters are initialized via `initialize_parameters`. `np.random.seed(42)` is set for reproducibility.
2. **Training Loop:** Training runs for a number of epochs (default: 20). Data is shuffled at each epoch to avoid ordering bias. The training data is split into mini-batches (default: 128 samples/batch) for efficient computation.
3. **Forward and Backward Pass:** Forward pass computes activations (a_1, a_2). Loss is calculated using cross-entropy via `compute_loss`. Gradients are computed through backpropagation, and parameters are updated via `gradient_descent` with learning rate lr.

4. **Metrics and Hyperparameters:** The model tracks training and validation loss and accuracy for each epoch. Key hyperparameters include:

    - `hidden_size` (default: 128)
    - `lr` – learning rate (default: 0.1)
    - `batch_size` (default: 128)

    Validation accuracy is printed every 5 epochs to monitor progress.

4) Final Accuracy (Manual Implementation)

After training, the manual NumPy-based implementation achieved a testing accuracy of **97.14%**.

5) Sklearn Implementation Comparison

Using similar hyperparameters with `sklearn.neural_network.MLPClassifier`, we obtained the following results:

- **Manual NumPy Implementation Test Accuracy:** 0.9714
- **Sklearn MLPClassifier Test Accuracy:** 0.9768

6) Accuracy Plot

We plotted the accuracy across various epochs for both the Manual and Sklearn implementations using Matplotlib:

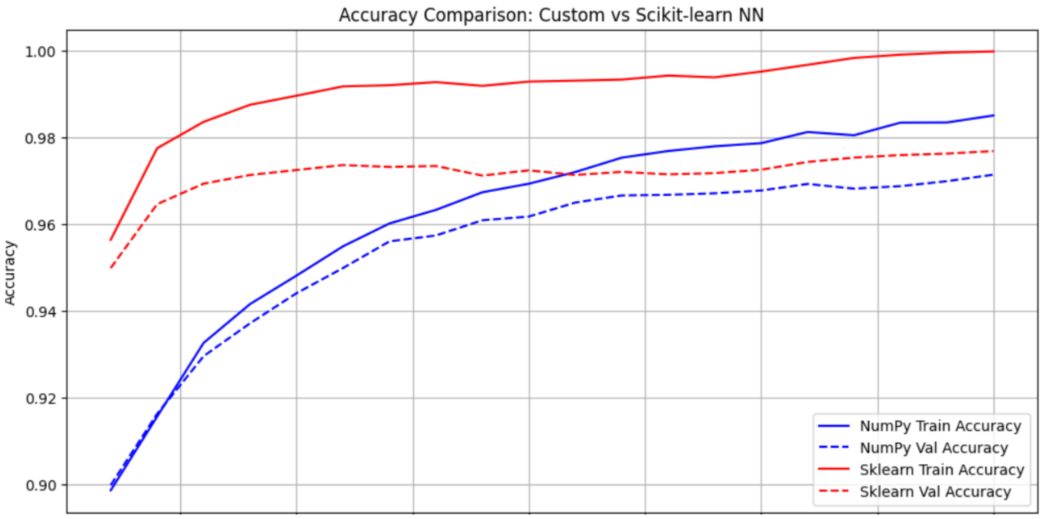

Figure 1: Accuracy Comparison of Manual vs Sklearn Neural Network Implementation

Q4: ML Project Titanic Passenger Survival Prediction. - Summary

1. The Titanic dataset was chosen for passenger survival prediction.
2. The dataset was cleaned, addressing missing values for Age, Cabin, and Embarked.
3. After exploratory data analysis, Age values were imputed based on Passenger class and median age.
4. Cabin had too many missing values and was dropped, while Embarked had only one missing value.
5. Feature Engineering was performed to derive family size based on siblings and parents.
6. Categorical columns like Gender and Embarked were one-hot encoded.
7. Non-essential columns like Passenger Name were removed.
8. Logistic Regression was used for prediction, and a simple Gradio app was developed as a Proof of Concept.
9. Titanic App Demo on Hugging Face