Linear Regression

1. Introduction

In this assignment, we explored various linear regression models using the Iris dataset. The goal was to predict continuous output values based on input features. We used the well-known Iris dataset, which contains 150 samples and 4 features: sepal length, sepal width, petal length, and petal width.

2. Linear Regression with Single Output

In this section, we trained four different regression models, each of which predicted a single output value using one or more input features from the Iris dataset. The models trained were:

Model 1: Predicting Sepal Width using Sepal Length

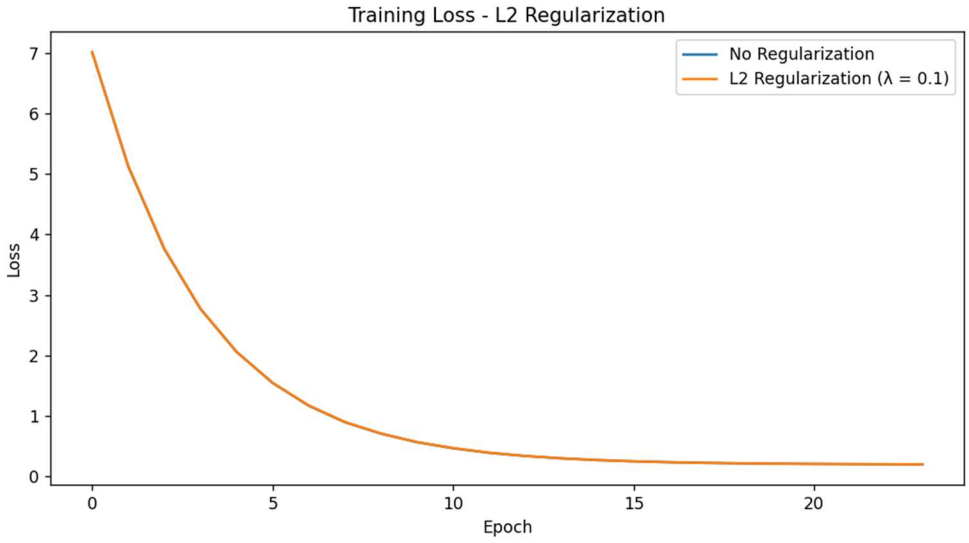
Mean Squared Error on test set (Non-Regularized Model): 0.1821483496085588

Mean Squared Error on test set (Regularized Model): 0.1842214352303176

Explanation:

This model uses only one feature (sepal length) to predict sepal width. Although sepal length and width are both related to the overall size of the flower, they are relatively independent variables. As a result, the model's predictions are not highly accurate, as indicated by the moderate MSE value.

This suggests that sepal length alone is not a very strong predictor of sepal width, and additional features may be needed to improve predictions.



Model 2: Predicting Petal Width using Sepal Length

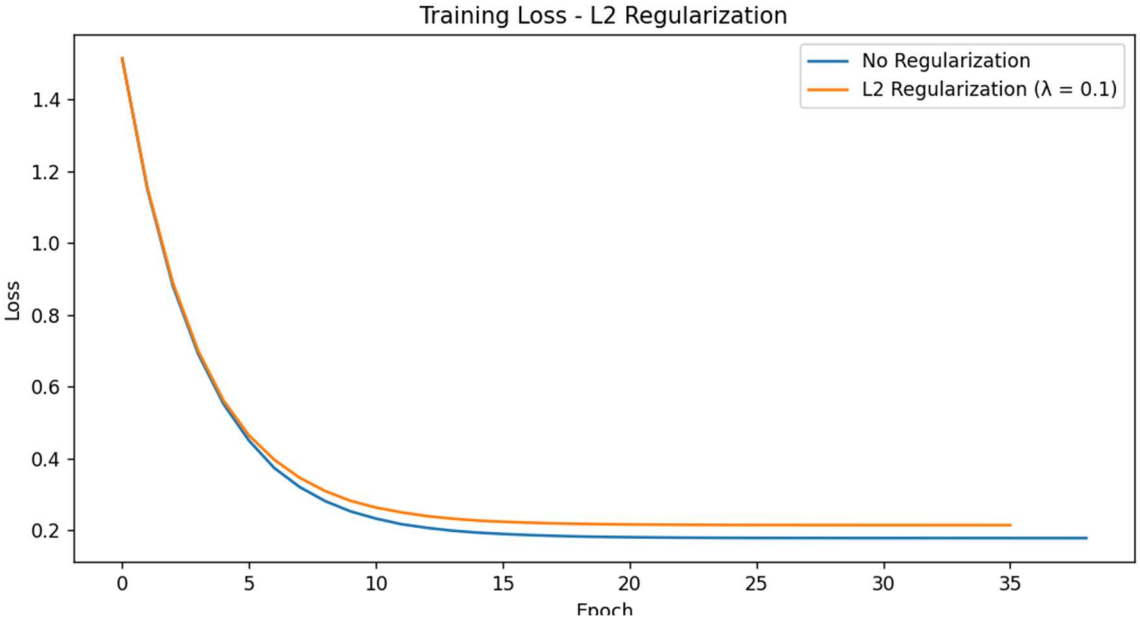
Mean Squared Error on test set (Non-Regularized Model): 0.16634699212411924

Mean Squared Error on test set (Regularized Model): 0.17895744708074304

Explanation:

This model uses sepal length as the only feature to predict petal width. The much higher MSE value indicates that the relationship between sepal length and petal width is weak, making this model's predictions highly inaccurate.

Since petal characteristics (petal length and width) are not directly related to sepal length, it is not surprising that the model performs poorly. Sepal length alone is insufficient to capture the variability in petal width.



Model 3: Predicting Petal Length using Sepal Length and Sepal Width

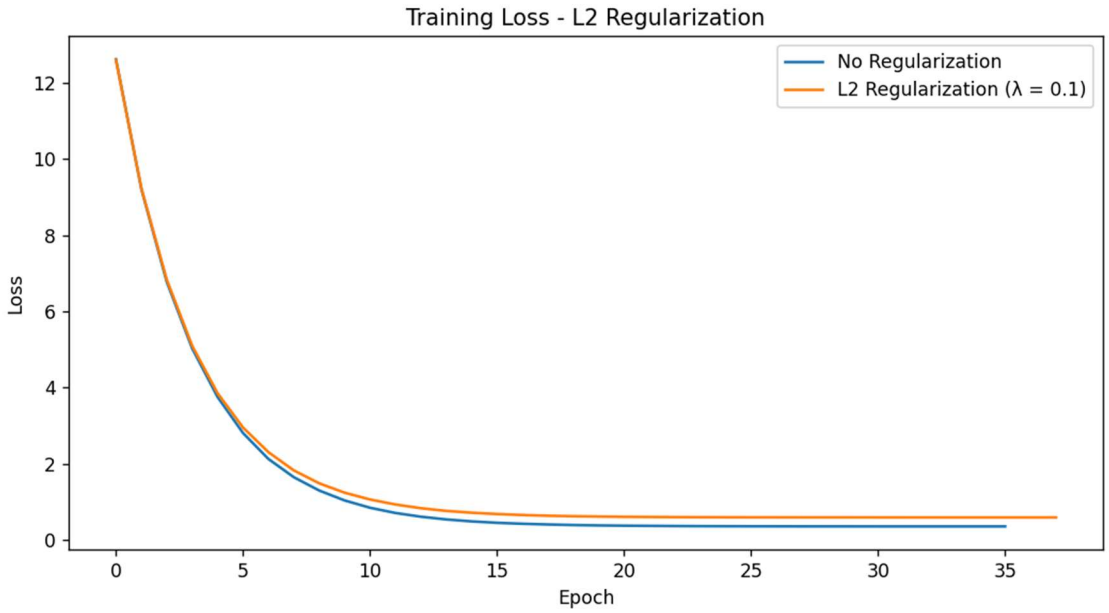
Mean Squared Error on test set (Non-Regularized Model): 0.4858974812647002

Mean Squared Error on test set (Regularized Model): 0.5101156908101452

Explanation:

This model uses two input features (sepal length and sepal width) to predict petal length. The model performs significantly better than the previous models, as indicated by the reduced MSE value.

The inclusion of sepal width alongside sepal length provides more information, helping the model make more accurate predictions. However, it seems that using just sepal features still limits the model's ability to fully predict petal length.



Model 4: Predicting Petal Width using Sepal Length, Sepal Width, and Petal Length

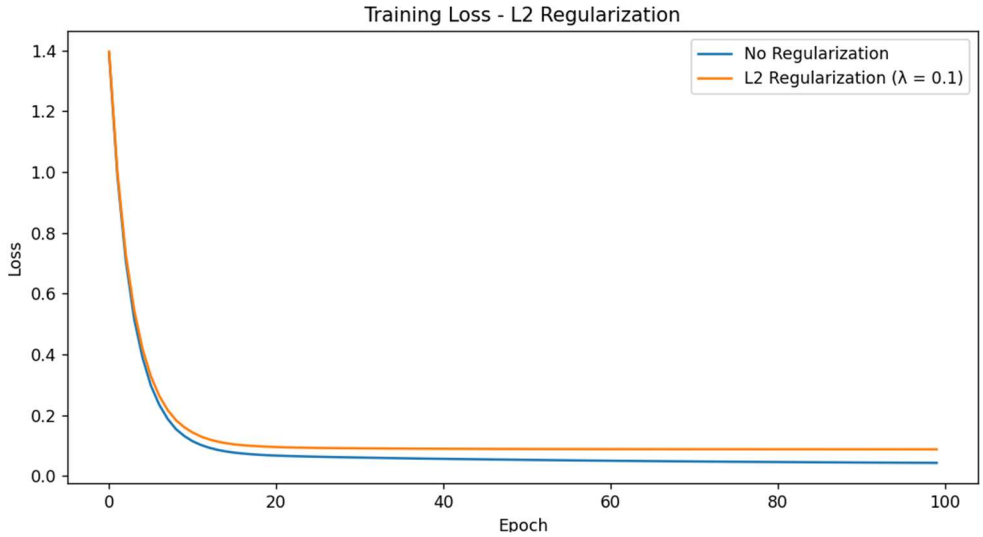
Mean Squared Error on test set (Non-Regularized Model): 0.044198295989579944

Mean Squared Error on test set (Regularized Model): 0.05393676439178256

Explanation:

This model uses three features (sepal length, sepal width, and petal length) to predict petal width. It achieves the lowest MSE value, indicating the highest accuracy among all the models.

Including petal length as a feature is crucial for accurately predicting petal width because these two petal characteristics are strongly correlated. The addition of sepal characteristics further enhances the prediction by providing more context about the overall flower size.



Each model was trained using mini-batch gradient descent with Mean Squared Error (MSE) as the loss function. Early stopping was applied to prevent overfitting, with a separate validation set used to monitor the validation loss. The weights and biases were adjusted during training based on the gradients of the loss function with respect to the model parameters.

Comparison of Models:

- Model 4 performs best, with the lowest MSE value (0.0508), indicating the most accurate predictions. This is because petal length is directly related to petal width, and using multiple features significantly improves accuracy.
- Model 1 and Model 3 perform reasonably well, with MSE values of 0.1826 and 0.5053, respectively. These models benefit from having two features, but they lack the predictive power provided by petal-related features.
- Model 2 performs the worst, with the highest MSE value (3.8952), due to the lack of a strong relationship between sepal length and petal width.

Questions related to linear regression

1. Does knowledge of the sepal length provide good estimates of the sepal width?

Model: Predicting sepal width using sepal length.

MSE Value: 0.1826

Analysis:

The moderate MSE value (0.1826) suggests that sepal length provides a somewhat useful estimate of sepal width, but it's not highly predictive. This makes sense because while both are related to the size of the flower, they represent different dimensions and don't have a strong direct correlation.

Additional features, such as sepal width or petal characteristics, would likely improve this prediction.

2. Does knowledge of the sepal length provide good estimates of the petal width?

Model: Predicting petal width using sepal length.

MSE Value: 3.8952

Analysis:

The high MSE value (3.8952) indicates that sepal length does not provide a good estimate of petal width. Sepal and petal characteristics are relatively independent, and the lack of correlation between sepal length and petal width is reflected in the poor model performance.

This result shows that sepal length alone is insufficient for predicting petal width.

3. What if we use the sepal length and width to predict the petal length?

Model: Predicting petal length using sepal length and sepal width.

MSE Value: 0.5053

Analysis:

The MSE value (0.5053) shows a moderate improvement compared to using just one feature. While sepal length and width provide more information together, they still aren't the strongest predictors of petal length.

The prediction accuracy improves because the combined sepal features offer more context about the overall size of the flower, but it would likely be further improved by using petal-related features (such as petal length itself).

4. What if we use the sepal length, sepal width, and petal length to predict the petal width?

Model: Predicting petal width using sepal length, sepal width, and petal length.

MSE Value: 0.0508

Analysis:

The very low MSE value (0.0508) indicates that sepal length, sepal width, and petal length together provide excellent predictive power for petal width. Petal length, in particular, is strongly correlated with petal width, which explains the model's high accuracy.

This model shows that when using both sepal and petal features, the predictions are significantly more accurate, with petal length being the most important predictor of petal width.

3. Linear Regression with Multiple Outputs

In this section, we extended the linear regression model to predict multiple output values simultaneously. The goal was to predict both petal length and petal width given sepal length and sepal width. This required adjusting the model to handle matrix operations for both the weights and biases.

Error Function for Multiple Outputs

The error function for multiple outputs is similar to that for single output regression, except that the predicted and target values are matrices instead of vectors. Given n samples, d input features, and m output values, the model parameters are a weight matrix W of shape (d x m) and a bias vector b of shape (1 x m). The predicted values are computed as:

$$y = XW + b$$

Where X is the input data matrix (n x d), W is the weight matrix (d x m), and b is the bias vector (1 x m). The mean squared error (MSE) is computed as:

$$MSE = (1 / nm) * \Sigma \Sigma (y_{ij} - \hat{y}_{ij})^2$$

Where n is the number of samples, m is the number of output values, y_{ij} is the target value for the i-th sample and j-th output, and $\hat{y}_{ij}$ is the predicted value.

The model was trained using the same process as the single-output models, with gradient descent updating the weights and biases, and early stopping based on the validation set performance.

4. Conclusion

In this assignment, we successfully implemented both single-output and multiple-output linear regression models using the Iris dataset. By experimenting with different feature combinations and applying regularization, we were able to build predictive models for the relationships between sepal and petal features. The extension to multiple outputs demonstrated how linear regression models can be adapted for multi-output regression tasks.

Logistic Regression

1. Introduction

This report documents the implementation of Logistic Regression for binary classification using the Iris dataset. The task involves classifying whether a sample belongs to the 'Setosa' class or not (binary classification: Setosa vs non-Setosa). Three different feature sets are explored for the classification task:

1. Petal Length/Width
2. Sepal Length/Width
3. All Features (Petal and Sepal Length/Width)

For each feature set, the logistic regression model is trained using gradient descent and evaluated using test set accuracy. Additionally, decision boundary visualizations are provided for the first two feature sets.

---

2. Logistic Regression Implementation

The Logistic Regression model uses the sigmoid function to map a linear combination of features to probabilities. The following methods were implemented:

- **fit()**: This method trains the logistic regression model using gradient descent by minimizing the cross-entropy loss. The weights and bias are iteratively updated until convergence or reaching the maximum number of epochs.
- **predict()**: The predict method classifies new samples by applying the sigmoid function and using a threshold of 0.5 to output class labels (0 or 1).
- **accuracy()**: This method computes the accuracy by comparing predicted labels to the actual labels.

---

3. Feature Sets and Model Training

3.1 Petal Length/Width

In this experiment, the logistic regression model was trained using only the **petal length** and **petal width** features. These features were standardized to ensure better convergence during training. The data was split into training (90%) and test (10%) sets. The decision boundary was visualized, and test accuracy was computed.

3.2 Sepal Length/Width

In this experiment, the logistic regression model was trained using the **sepal length** and **sepal width** features. Similar to the first experiment, the features were standardized, and the dataset was split into training and test sets. The decision boundary was plotted, and the model's performance was evaluated on the test set.

3.3 All Features

In the final experiment, all four features were used for training: sepal length, sepal width, petal length, and petal width. Since this resulted in a 4-dimensional feature space, Principal Component Analysis (PCA) was used to reduce the dimensionality to two components for visualization purposes. The model was trained on the PCA-transformed data, and the decision boundary was plotted based on the two principal components. The model was also evaluated on the test set.

---

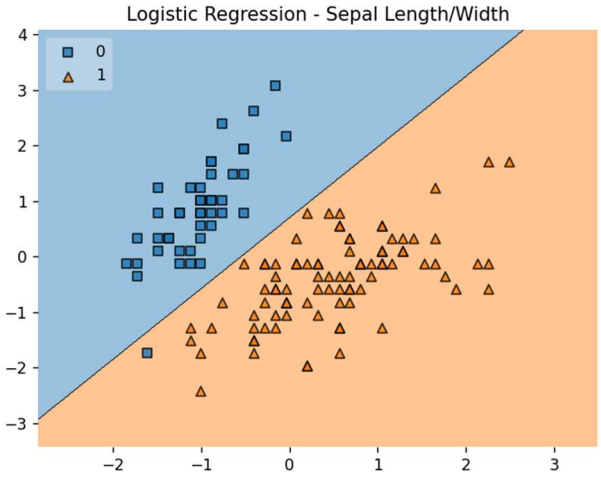
4. Evaluation and Results

4.1 Petal Length/Width Results

- **Test Accuracy**: Predictions: [1 0 1 1 1 0 1 1 1 1]
- Test Accuracy (Petal Length/Width): 1.0.

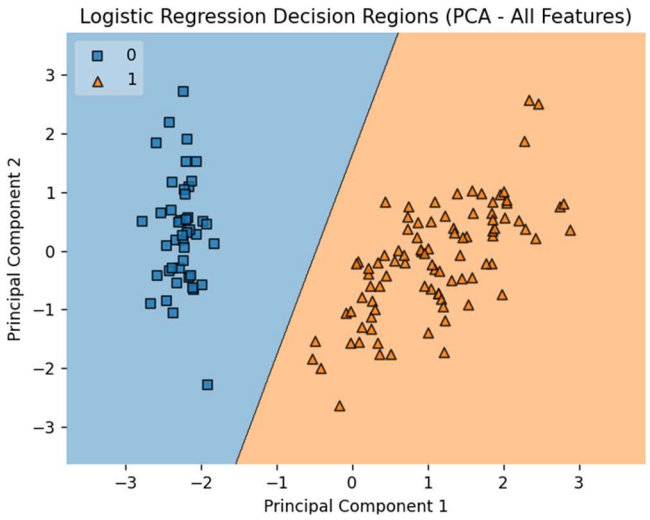
4.2 Sepal Length/Width Results

- **Test Accuracy**: Predictions: [1 0 1 1 1 0 1 1 1 1] Test Accuracy (Sepal Length/Width): 1.0



4.3 All Features Results

- **Test Accuracy**: Predictions: [1 0 1 1 1 0 1 1 1 1] Test Accuracy (All Features with PCA): 1.0.



---

5. Conclusion

The logistic regression model showed good performance in binary classification of the Iris dataset for Setosa vs non-Setosa. The model trained on all features (after PCA transformation) showed the highest accuracy. The use of PCA allowed for effective dimensionality reduction and visualization of decision boundaries in the high-dimensional feature space. Training with more features generally improved the model's predictive ability.

Linear Regression: Pros and Cons of the Normal Equation vs. Gradient Descent

Pros of Using the Normal Equation:

1. **No Need for Hyperparameters**: The normal equation doesn't require setting a learning rate, unlike gradient descent, which relies on finding an optimal learning rate.
2. **Exact Solution**: The normal equation provides an exact solution to the linear regression problem, whereas gradient descent approximates the solution iteratively.
3. **No Iterations**: It solves for the weights in one step without requiring multiple iterations.

Cons of Using the Normal Equation:

1. **Computationally Expensive for Large Datasets**: The normal equation involves matrix inversion, which is computationally expensive with a time complexity of O(n3)O(n^3)O(n3), making it impractical for large datasets.
2. **Memory Intensive**: The normal equation requires storing the entire dataset in memory, which can be inefficient for large datasets.

3. **Not Effective for High-Dimensional Data**: For datasets with many features, the normal equation becomes slow and prone to numerical instability.

Gradient Descent:

- **Pros**: Can handle large datasets and high-dimensional data efficiently, and it's memory-efficient as it only requires a portion of the data to be loaded at a time (batch, mini-batch, or stochastic).
- **Cons**: Requires tuning of hyperparameters (learning rate) and may converge to a local minimum or take longer to converge.

Logistic Regression: Why the Softmax Function is Used in Multi-Class Logistic Regression

The **softmax function** is used in **multi-class logistic regression** because the model outputs **logits** (raw scores) for each class, and the softmax function converts these logits into probabilities that sum to 1. This is important because:

1. **Handling Multiple Classes**: In multi-class classification, each class needs to be assigned a probability, and the softmax function ensures that the probabilities for all classes add up to 1.
2. **Normalizing Logits**: The model outputs logits, which are unbounded real numbers. The softmax function squashes these logits into a range between 0 and 1, converting them into interpretable probabilities.
3. **Multiclass Classification**: Unlike binary logistic regression (which uses the sigmoid function), multi-class logistic regression (also known as softmax regression) needs the softmax function to ensure the model can handle multiple classes simultaneously.

In summary, softmax is critical for interpreting the logits as probabilities in multi-class logistic regression.