Assignment Report

Deepak Singh , MA23M006

September 22, 2024

1 Introduction

The objective of this assignment is to explore various regression techniques, including least squares, ridge regression, and kernel regression. By analyzing the dataset in `FMLA1Q1Data_train.csv`, which contains 1000 training data points (each consisting of two features and a corresponding y-value) and `FMLA1Q1Data_test.csv` containing 100 testing points , the following tasks will be demonstrated:

1. Solving the least squares regression problem using the analytical solution.
2. Implementing gradient descent for least squares regression and observing its convergence.
3. Implementing stochastic gradient descent and comparing its behavior to standard gradient descent.
4. Solving ridge regression using gradient descent and cross-validating for the best choice of regularization parameter, λ.
5. Implementing kernel regression and evaluating its performance compared to least squares.

2 Task 1: Analytical Solution to Least Squares Regression

Objective: Finding w_{ML} using the analytical solution to the least squares regression problem. The solution for w_{ML} is given by:

$$w_{ML} = (X^T X)^{-1} X^T y$$

Where:

- X is the matrix of input features (1,000 points with 2 features each).
- y is the vector of output values.

Code Implementation:

```python
def least_squares(X, y):
    X_T_X_inv = np.linalg.inv(X.T @ X)  #computing (X^T X)^(-1)
    wML = X_T_X_inv @ X.T @ y           #compute wML = (X^T X)^(-1) X^T y
    return wML

wML = least_squares(X, y)  #computing the least squares solution
print("Least squares solution wML:", wML)
```

```
Least squares solution wML: [9.89400832 1.76570568 3.5215898 ]
```

Figure 1: This is analytical solution.

3 Task 2: Gradient Descent for Least Squares

Objective: Implement gradient descent to minimize the least squares cost function and observe the convergence of the weight vector w_t to the analytical solution w_{ML}. The norm $||w_t - w_{ML}||^2$ is plotted as a function of iterations.

Gradient Descent Algorithm:

$$w_t = w_{t-1} - \eta \nabla J(w_{t-1})$$

Where η is the learning rate and $\nabla J(w_t) = X^T(Xw_t - y)$ is the gradient of the cost function.

Output:

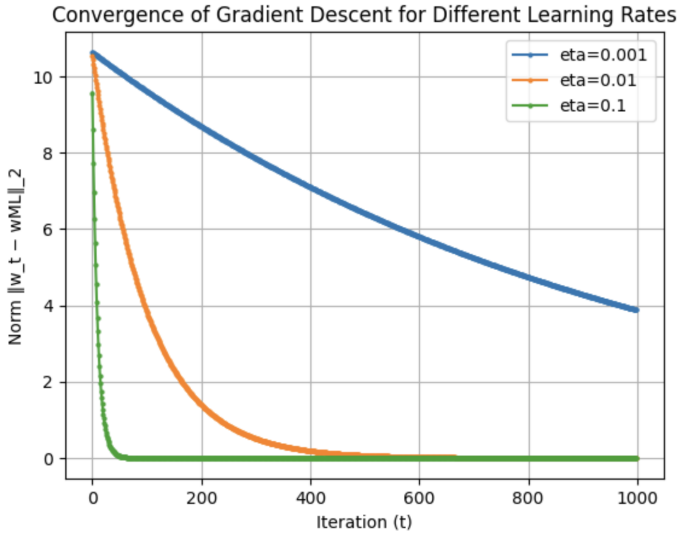


Figure 2: Graph for different step length (eta).

Observations: 1. As the number of iterations increases, $||w_t - w_{ML}||^2$ gradually decreases, demonstrating the convergence of gradient descent towards

the least squares solution. The rate of convergence depends on the step size η.

2. Depending on the step size of eta , the rate of convergence can vary. If the step size is too large, the algorithm might not converge, while if it is too small, convergence will be slow.

4 Task 3: Stochastic Gradient Descent (SGD)

Objective: Stochastic gradient descent (SGD) is implemented using a batch size of 100. Similar to Task 2, $||w_t - w_{ML}||^2$ is plotted against the number of iterations.

Output:

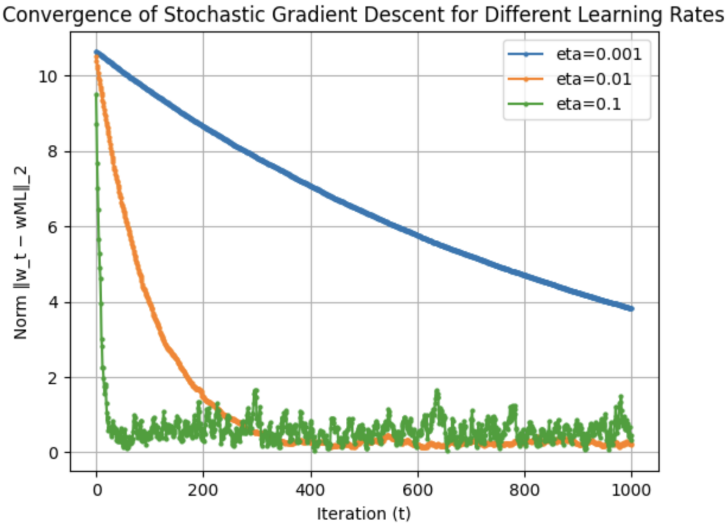


Figure 3: Graph for different step length (eta).

Observations:

1. Initially, the norm $||w_t - w_{ML}||^2$ will decrease rapidly, indicating that the weights are getting closer to the optimal solution.

2. Due to the randomness introduced by stochastic gradient descent, the curve may not be as smooth as in full-batch gradient descent. It may oscillate a bit because of the random sampling of batches, but overall, it should still trend downwards.

3. Depending on the learning rate eta , the batch size, and the total number of iterations, you may observe faster convergence compared to full-batch gradient descent.

Notes:

Step Size (eta) :If the step size is too large, the algorithm may not converge. If it's too small, convergence will be slow.

Batch Size: A smaller batch size introduces more stochasticity, which can lead to faster but noisier convergence. A larger batch size makes the gradient more accurate but slower to compute.

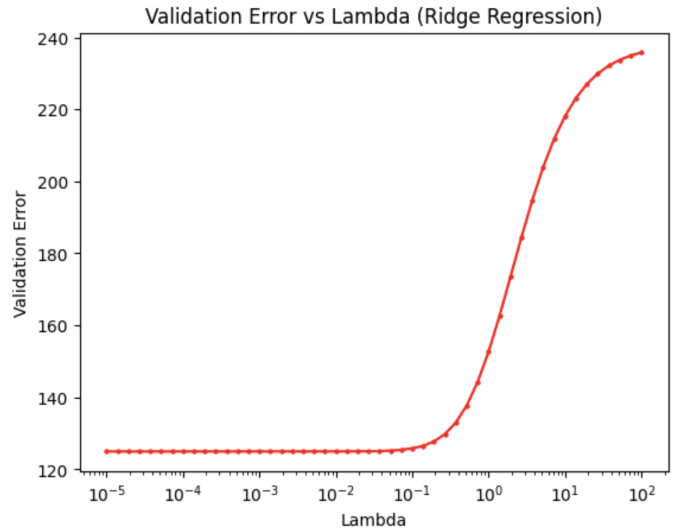
5 Task 4: Ridge Regression with Gradient Descent

Objective: Ridge regression introduces a regularization term to prevent overfitting. The weight vector w_R is obtained by minimizing the following cost function:

$$J(w) = ||Xw - y||^2 + \lambda||w||^2$$

Where λ controls the strength of regularization.

Output:




```
Best lambda: 0.003727593720314938
wR: [9.85701426 1.75760067 3.50973469]
```

```
Test error for wR (Ridge): 65.94804114527419
Test error for wML (Least Squares): 66.00545933461238
Ridge regression performs better.
```

Figure 4: Graph for different step length (eta).

Comparison of w_R and w_{ML}:

1. Ridge Regression tends to perform better when the features are highly collinear or the model suffers from overfitting, as the regularization term helps reduce variance.

2. Least Squares Regression (wML) might perform better if the data does not suffer from overfitting or multicollinearity.

3. As lambda increases, Ridge Regression adds more regularization, shrinking the weights and potentially reducing overfitting.

6 Task 5: Kernel Regression

Objective: Kernel regression is implemented to capture non-linear relationships in the data. The radial basis function (RBF) kernel is chosen for its flexibility in modeling complex data:

$$K(x, x') = \exp\left(-\gamma ||x - x'||^2\right)$$

Output: Comparison with Least Squares: Why the RBF Kernel is

```
# Kernel regression parameters
lambda_val = 0.0037275937203149 38  # Regularization term
sigma = 1.0        # Kernel bandwidth (hyperparameter)

# Predict the test data using kernel regression
y_pred_kernel = kernel_ridge_regression(X_train, y_train, X_test, lambda_val, sigma)

# Compute the test error for kernel regression
test_error_kernel = np.mean((y_test - y_pred_kernel) ** 2)

# Compute the test error for least squares regression (wML)
X_train_with_bias = np.hstack([np.ones((X_train.shape[0], 1)), X_train])

X_test_with_bias = np.hstack([np.ones((X_test.shape[0], 1)), X_test])


print("Test error for Kernel Regression (RBF):", test_error_kernel)
print("Test error for Least Squares (wML):", test_error_ml)
```

```
Test error for Kernel Regression (RBF): 0.19410260921652334
Test error for Least Squares (wML): 66.00545933461238
```

Figure 5: Comparison with Least Squares

Better:

1. Nonlinear relationships: The RBF kernel captures nonlinear relationships between the features, which might be crucial if the dataset has patterns that cannot be modeled by a simple linear function.

2. Localized impact: Each data point influences only the nearby points due to the exponential decay in the RBF function. This makes the model robust to outliers or irrelevant data points, unlike linear least squares regression, where all points influence the model equally.

Conclusion:

If kernel regression with the RBF kernel gives a lower test error, it indicates that the relationship between features and target values is likely nonlinear, and the RBF kernel successfully captured these patterns.

If least squares regression performs better, it suggests that the dataset's underlying relationship is close to linear, and the added complexity of kernel methods may not be necessary.

7 Conclusion

This report demonstrated the implementation of least squares regression, gradient descent, stochastic gradient descent, ridge regression, and kernel regression. Each method was tested and compared based on its effectiveness and computational properties. Kernel regression provided additional flexibility in capturing non-linear relationships, making it more effective for certain types of data.