Introduction to Machine Learning
Assignment 2
Linear Regression

Group 05
Bora Yilmaz (s3903125) & Felix Zailskas (s3918270)

September 30, 2021

Contents

Linear Regression

1. Introduction

In contrast to classifying algorithms, in this implementation the linear regression is considered for machine learning. Instead of being a classifier, this algorithm regresses, aiming to provide a predictive model, for guessing future values and/or reducing error in data. This algorithm aims to calculate the best fit line between two variables in given data, under the assumption of a dependence $y = w * x$ where $w, x \in \mathbb{R}^{25}$ and $y \in \mathbb{R}$. In this algorithm, an optimal weight vector w is calculated based on a test set of feature and label vectors. After thorough plotting and observation of resulting error rate and estimates, the behaviour and accuracy of the linear regression algorithm will be discussed.

The data set on which this linear regression is executed on contains two sets of 500 feature vectors with 25 dimensions and two sets of 500 continuous values representing the target labels to the feature vectors.
One of these set combinations will be used for training the linear regression coefficients, the other will be used to test the linear regression coefficients on novel data.

2. Method

To estimate the linear regression coefficients we first assume that there is a linear relationship between the feature vectors x^μ and the labels y^μ. We use the mean square error function to calculate the error E of our linear regression as follows

$$E = \frac{1}{2P} \sum_{\mu=1}^{P} \left[w^T \cdot x^\mu - y^\mu \right]^2 \tag{1}$$

where P is the amount of feature vectors we have.
We see that the error is reduced if the dot product of a feature vector and the weight vector is equal to the label of that feature vector.
Given that $P > N$, where N is the dimension of the feature vectors, we can find the best possible weight vector w by using the gradient of E in respect to w. We get

$$\nabla_w E = \sum_{\mu=1}^{P} \left[w^T \cdot x^\mu - y^\mu \right] x^\mu \tag{2}$$

We now define the matrices

$$\begin{aligned} Y &= [y^1, y^2, ..., y^P]^T \in \mathbb{R}^P \\ X &= [x^1, x^2, ..., x^P]^T, dim(N \times P) \end{aligned} \tag{3}$$

to rewrite 2 as

$$\nabla_w E = X^T (Xw - Y) \tag{4}$$

Since we want to minimize the error (set it to 0) we can extract the values for w in the following way

$$\begin{aligned} 0 &= X^T (Xw - Y) \\ &\Leftrightarrow \\ w &= [X^T X]^{-1} X^T Y \end{aligned} \tag{5}$$

The assignment from 5 can be neatly packed into this python code

```
x_pinv = np.linalg.inv(np.transpose(train_x) @ train_x + (delta * np.
    identity(train_x.shape[1]))) @ np.transpose(train_x)
w = np.dot(x_pinv, train_y)
```

We calculated the weights and both training error and testing error for a variety of different test set sizes P. For the error analysis we used 30, 40, 50, 75, 100, 200, 300, 400, and 500 for values of P. For the weight analysis we used 30, 40, 50, 75, 100 and 500 for values of P.

3. Results

In this section we will present the results our linear regression analysis has given.

3.1. Error Analysis

The first result we will look at is the influence of the cardinality P of the training set.

Figure 1: *Plot of both the training error (green) and testing error (red) for different values of P.*

We can see that both training error and testing error seem to converge to the same number (ca. 0.5) as P increases. This means that for bigger test data sets, the error made on novel data is decreasing. At the same time we can see that bigger test data sets increase the testing error. Furthermore, at $P = 30$, the lowest cardinality tested, the testing error has a value of 5.3 and the training error has a value of 0.04.

3.2. Weight Analysis

Now we will investigate the weights generated for the presented P values.

3.2.1. $P = 30$

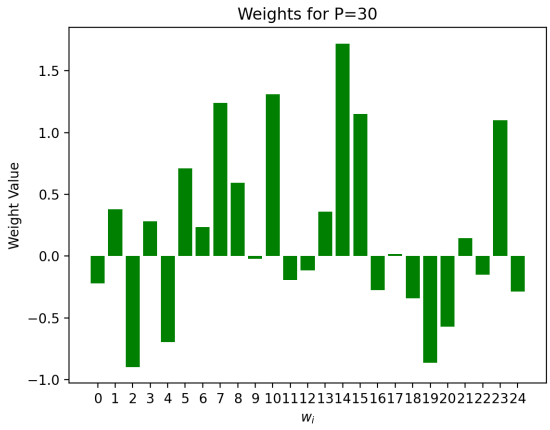

Figure 2: *Histogram of best weight vector for* $P = 30$.

The weight with the maximal value is $w_{14} = 1.72$. The weight with the minimal value is $w_2 = -0.90$ Furthermore, weights $w_2, w_4, w_{16}, w_{18}, w_{19}, w_{20}$ and w_{24} have a value below -0.25 while weights $w_1, w_3, w_5, w_7, w_8, w_{10}, w_{13}, w_{14}, w_{15}, w_{23}$ have a value above 0.25. This gives us a total of 17 extreme weights.

3.2.2. $P = 40$

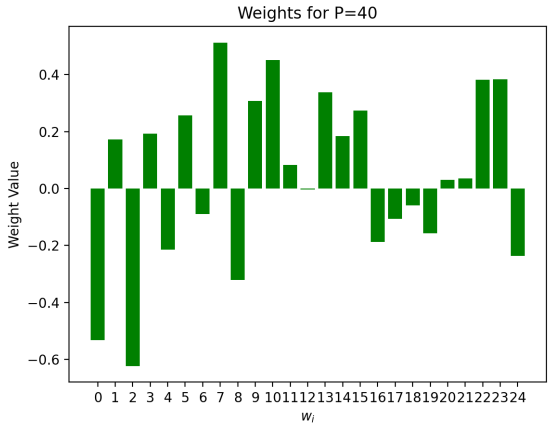

Figure 3: *Histogram of best weight vector for* $P = 40$.

The weight with the maximal value is $w_7 = 0.51$. The weight with the minimal value is $w_2 = -0.62$ Furthermore, weights w_0, w_2 and w_8 have a value below -0.25 while weights $w_5, w_7, w_9, w_{10}, w_{13}, w_{15}, w_{22}$ and w_{23} have a value above 0.25. This gives us a total of 11 extreme weights.

3.2.3. $P = 50$

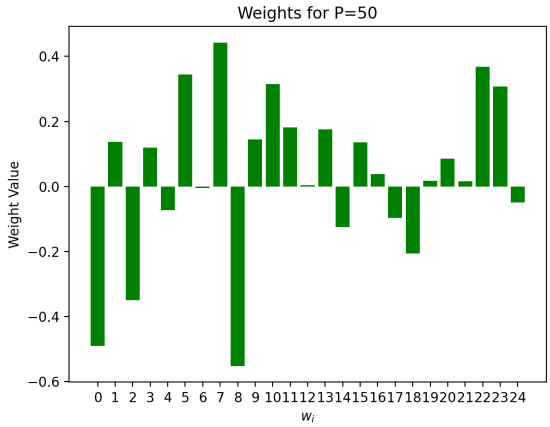

Figure 4: *Histogram of best weight vector for* $P = 50$.

The weight with the maximal value is $w_7 = 0.44$. The weight with the minimal value is $w_8 = -0.55$ Furthermore, weights w_0, w_2 and w_8 have a value below -0.25 while weights w_5, w_7, w_{10}, w_{22} and w_{23} have a value above 0.25. This gives us a total of 8 extreme weights.

3.2.4. $P = 75$

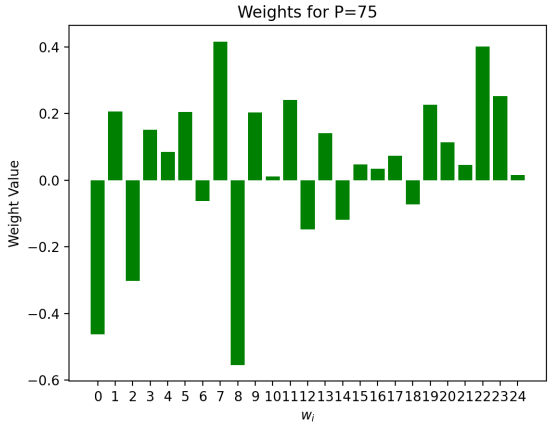

Figure 5: *Histogram of best weight vector for* $P = 75$.

The weight with the maximal value is $w_7 = 0.42$. The weight with the minimal value is $w_8 = -0.55$ Furthermore, weights w_0, w_2 and w_8 have a value below -0.25 while weights w_7, w_{22} and w_{23} have a value above 0.25. This gives us a total of 6 extreme weights.

3.2.5. $P = 100$

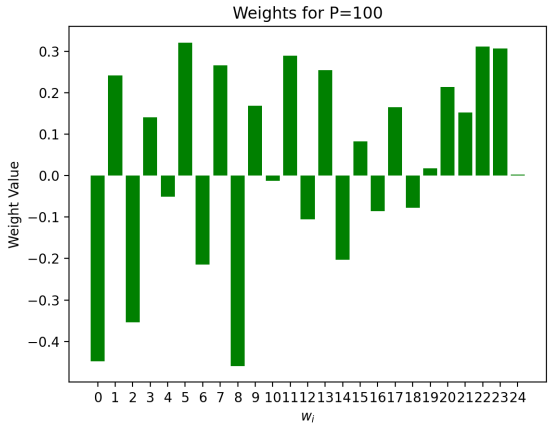

Figure 6: *Histogram of best weight vector for* $P = 100$.

The weight with the maximal value is $w_5 = 0.32$. The weight with the minimal value is $w_8 = -0.46$ Furthermore, weights w_0, w_2 and w_8 have a value below -0.25 while weights $w_5, w_7, w_{11}, w_{13}, w_{22}$ and w_{23} have a value above 0.25. This gives us a total of 9 extreme weights.

3.2.6. $P = 500$

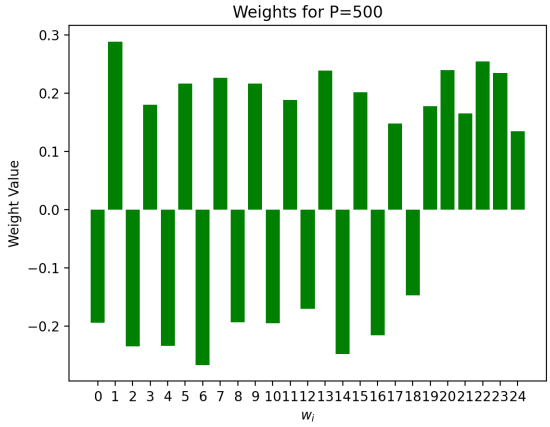

Figure 7: *Histogram of best weight vector for* $P = 500$.

The weight with the maximal value is $w_1 = 0.29$. The weight with the minimal value is $w_6 = -0.27$ Only weight w_6 has a value below -0.25 while weights w_1 and w_{22} have a value above 0.25. This gives us a total of 3 extreme weights.

4. Discussion

4.1. Error Analysis

For the error analysis via Figure 1, there are two critical observations that are clear via observing the lines; the initial error(s) at $P = 30$ and the convergence of the training and testing error lines at approximately 0.5.
For $P = 30$, the testing errors (red line) really high initial value, is simply tied into the fact that with a training model working with such a small size of 30, it is not possible to reach a good estimate model. The weights in w^* will not be a good guesser of the testing data. Thus, the error sits at a high value, especially when compared to the error with higher P. The initial training error on the other hand, sits low at 0.004. This is then connected to the fact that with such a small training size, the model is too specifically trained for those 30 values, and thus results in a very low error, meaning it is only a good model for the data it was trained with, which obviously is not the aim of the training and model.
For $P = 40$, we see the drop in testing error. It is the case that even when the training data size increased by 10 elements, the error decreases immensely. It is the case that those 10 elements allowed the weights to scale to a way better value.
For higher P values, $P \geq 100$, the convergence of the red and green line is observed. Starting from $P = 100$, both errors converge into an plateau error of approximately 0.5. This is a clear sign that the training provided a good representative regression model. The testing error, decreased into the plateau, meaning that the model reaches a low error rate as P increases. At $P = 500$, it becomes a straight line, indicating the plateau. So, larger training data sets lead to better estimates. Training error on the other hand, increases up to this plateau instead of decreasing into it, because it's error is respective to itself. Even though it's self-error increases, it provides a lower error for novel data. The converging plateau signifies that the model reached a best model for $P \geq 400$.

4.2. Weight Analysis

From figures 2 to 7, P increases with each figure and the weights are plotted on bar plots. The pattern observed is that initially, for example at Figure 2, the weights are not evened out and while some weights are near 0, some are extremely high (compared to further plots).For $P = 40, 50, 75$, it can be said that some weights keep their values as P increases, while some keep changing, from positive to negative even. These changes that happen to each weight w_i is caused by the new elements that are added as P changes.
At $P = 500$, compared to all previous P values, shows an evened out bar plot. With a data size of 500, it is the case that the weights are closing in on their true values. Even though we can not guess exactly what the vector w that generated the data is, the weights that we get by training in $P = 500$ and assumed in higher P, allows the approximation of such vector w. With the original data and high value of P, the approximation gets closer and closer, such that the difference between the approximate and the true vector should become minimal.
For the further part of the analysis we define weight w as extreme if $|w| > 0.25$.

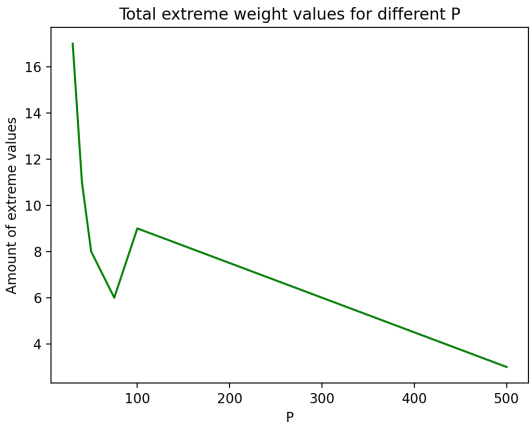

Figure 8: *Amount of extreme values x where $|x| > 0.25$*

.

From Figure 8 and Figure 1 combined, it can be seen how as the number of extreme values decrease, the error also decreases. Also, with a training size of $P = 30$, it is expected that some weights will be extreme and overfitted to the testing data. This is because it is highly likely that only a few weights will be relevant in a data set in which there are not enough data points. As P increases, it is less likely that a certain weight or two are as relevant and high in value as when there were fewer points.
One last observation to mention is how the pattern of negative then positive weight follows for weights w_0 to w_{19} for $P = 500$. The first weight is negative, then positive, then negative, and this pattern goes on until w_{19}. This pattern is seen only in Figure 7. In previous P plots, this pattern is not clearly seen and also in some cases, the pattern of alternating between negative and positive disappears. It can be said that at Figure 7, the weights approximate towards their true values thanks to the better trained model, and the alternating pattern is just a coincidence or nature of the data itself.

5. Contribution

Both team members attended the lab session for this assignment and both agree that the contribution to this assignment was equal and fair.

5.1. Code

The development of the code for the assignment was done in a lab session. Therefore, the contribution to the code base was entirely equal for both group members.

5.2. Report

Sections were split in half and thus the report was worked on by both members. After looking over the report together one last time, it was finalized and ready.

6. Code Appendix

6.1. linreg.py

```
import pandas as pd
import numpy as np
import plotting as pl
from matplotlib import pyplot as plt


def print_weight_analysis():
    below_th = -0.25
    above_th = 0.25
    total_extr = []
    for i in range(len(idx)):
        weights = W[idx[i]]
        print("--------------------")
        print(f"Current: P={P[idx[i]]}")
        maximum = [-1, 0]
        minimum = [-1, 0]
        below = []
        above = []
        for j in range(len(weights)):
            w = weights[j]
            if w > maximum[1]:
                maximum = [j, w]
            if w < minimum[1]:
                minimum = [j, w]
            if w < below_th:
                below.append([j, w])
            if w > above_th:
                above.append([j, w])
        print("Maximum:", maximum)
        print("Minimum:", minimum)
        print(f"Below {below_th}:", below)
        print(f"Above {above_th}:", above)
        print("Total extreme:", len(above)+len(below))
        total_extr.append([P[idx[i]], len(above)+len(below)])
        i += 1
    return total_extr


def calc_error(w, x, y, delta=0):
    assert len(w) == x.shape[1] and x.shape[0] == len(y)
    sum = 0
    for i in range(0, x.shape[0]):
        sum += 0.5 * ((np.dot(w, x[i]) - y[i])**2)
    sum += 0.5 * delta * np.dot(w, w)
    return sum / len(y)


def get_errors_and_weight(train_x, train_y, test_x, test_y, delta=0):
    x_pinv = np.linalg.inv(np.transpose(train_x) @ train_x + (delta *
   np.identity(train_x.shape[1]))) @ np.transpose(train_x)
    w = np.dot(x_pinv, train_y)
```

```
    return calc_error(w, train_x, train_y, delta), calc_error(w,
   test_x, test_y, delta), w


# read in the data
# training
train_x = np.loadtxt(open("data/xtrain.csv", "r+"), delimiter=",")
train_y = np.loadtxt(open("data/ytrain.csv", "r+"), delimiter=",")

# testing
test_x = np.loadtxt(open("data/xtest.csv", "r+"), delimiter=",")
test_y = np.loadtxt(open("data/ytest.csv", "r+"), delimiter=",")

# setting the regularization term
delta = 0

# 0: 30, 1: 40, 2: 50, 3: 75, 4: 100, 5: 200, 6: 300, 7: 400, 8: 500
P = [30, 40, 50, 75, 100, 200, 300, 400, 500]
idx = [0, 1, 2, 3, 4, 8]
W = []
train_errors = []
test_errors = []
for i in range(0, len(P)):
    train_error, test_error, w = get_errors_and_weight(train_x[0:P[i]],
    train_y[0:P[i]],

   test_x[0:P[i]], test_y[0:P[i]], delta)
    train_errors.append(train_error)
    test_errors.append(test_error)
    W.append(w)


pl.plot_total_extreme_values(print_weight_analysis())
pl.plot_error(train_errors, test_errors, P)
pl.plot_W(W, P, idx)
plt.show()
```

6.2. PLOTTING.PY

```
from matplotlib import pyplot as plt
import numpy as np


def plot_error(train_err, test_err, P):
    plt.figure()
    plt.title(f"Training and Testing error for different P values")
    plt.xlabel("P")
    plt.ylabel("Training and Test Error")
    plt.plot(P, train_err, color='green', label='Training Error')
    plt.plot(P, test_err, color='red', label='Testing Error')
    plt.legend()
```

```
def plot_W(W, P, idx):
    ticks = np.arange(len(W[0]))
    for i in idx:
        plt.figure()
        plt.title(f"Weights for P={P[i]}")
        plt.xlabel('$w_{i}$')
        plt.ylabel("Weight Value")
        plt.xticks(ticks)
        plt.bar(ticks, W[i], color='green', align='center')


def plot_total_extreme_values(vals):
    plt.figure()
    plt.title("Total extreme weight values for different P")
    plt.xlabel('P')
    plt.ylabel("Amount of extreme values")
    plt.plot([val[0] for val in vals], [val[1] for val in vals], color
   ="green")
```