COMP9417: Homework Set #1

z5113817

University of New South Wales — June 21, 2021

Question 1

a)

We know from the normal equations for a the gradient of a minimised linear regression is:

$$\hat{\beta}_1 = \frac{\bar{XY} - \bar{X}\bar{Y}}{(\bar{X^2}) - (\bar{X})^2}$$

Which can expand into:

$$\hat{\beta}_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

So now substituting in the transformation, we get:

$$\hat{\beta}_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(\tilde{X}_i - \bar{X})(Y_i - \bar{Y})}{\frac{1}{n}\sum_{i=1}^{n}(\tilde{X}_i - \bar{X})^2} \tag{1}$$

$$\hat{\beta}_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(c(X_i + d) - \bar{X})(Y_i - \bar{Y})}{\frac{1}{n}\sum_{i=1}^{n}(c(X_i + d) - \bar{X})^2} \tag{2}$$

and expand:

$$\hat{\beta}_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(c * X_i + c * d - \bar{X})(Y_i - \bar{Y})}{\frac{1}{n}\sum_{i=1}^{n}(c * X_i + c * d - \bar{X})^2} \tag{3}$$

$$\hat{\beta}_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(c * X_i + c * d - \bar{X})(Y_i - \bar{Y})}{\frac{1}{n}\sum_{i=1}^{n}(c * X_i + c * d - \bar{X})^2} \tag{4}$$

I expanded this on paper and could not work out how to represent it in terms of the original expression.

b)

So we have:

$$\hat{\beta}_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

Let "t" be the number of treatments and "T" be the set in [1..n] corresponding to the data point in which a treatment was supplied (i.e equals 1).

Likewise, let "p" the number of placebo and "P" be the set in [1..n] corresponding to the data point in which a placebo was supplied (i.e equals 0).

Note that T union P makes up the complete set of [1..n].

$$\hat{\beta}_1 = \frac{\frac{1}{n}\sum_{i\in T}^{n}(1 - \frac{t}{n})(Y_i - \frac{\bar{Y_T} * t + \bar{Y_P}(n-t)}{n}) + \frac{1}{n}\sum_{i\in P}^{n}(0 - \frac{t}{n})(Y_i - \frac{\bar{Y_T} * t + \bar{Y_P}(n-t)}{n})}{\frac{1}{n}\sum_{i=1}^{n}(n-t)^2(-\frac{t}{n}) + (t)(1 - \frac{t}{n})^2}$$

Now lets remove the sigma for terms which aren't dependant on "i".

$$\hat{\beta}_1 = \frac{\frac{n-t}{n^2}\sum_{i\in T}^{n}(Y_i - \frac{\bar{Y_T}*t+\bar{Y_P}(n-t)}{n}) + \frac{-t}{n^2}\sum_{i\in P}^{n}(Y_i - \frac{\bar{Y_T}*t+\bar{Y_P}(n-t)}{n})}{\frac{1}{n}(n)(n-t)^2(-\frac{t}{n}) + (t)(1-\frac{t}{n})^2}$$

and simplify:

$$\hat{\beta}_1 = \frac{\frac{n-t}{n^2}\sum_{i\in T}^{n}(Y_i - \frac{\bar{Y_T}*t+\bar{Y_P}(n-t)}{n}) + \frac{-t}{n^2}\sum_{i\in P}^{n}(Y_i - \frac{\bar{Y_T}*t+\bar{Y_P}(n-t)}{n})}{t(1-n) + t^2(2-\frac{2}{n}) + t^3(\frac{1}{n^2} - \frac{1}{n})}$$

This is an expression for B1 in terms of the group means but there is an additional term "t". I was hoping this would cancel out.

An expression for B0 would be:

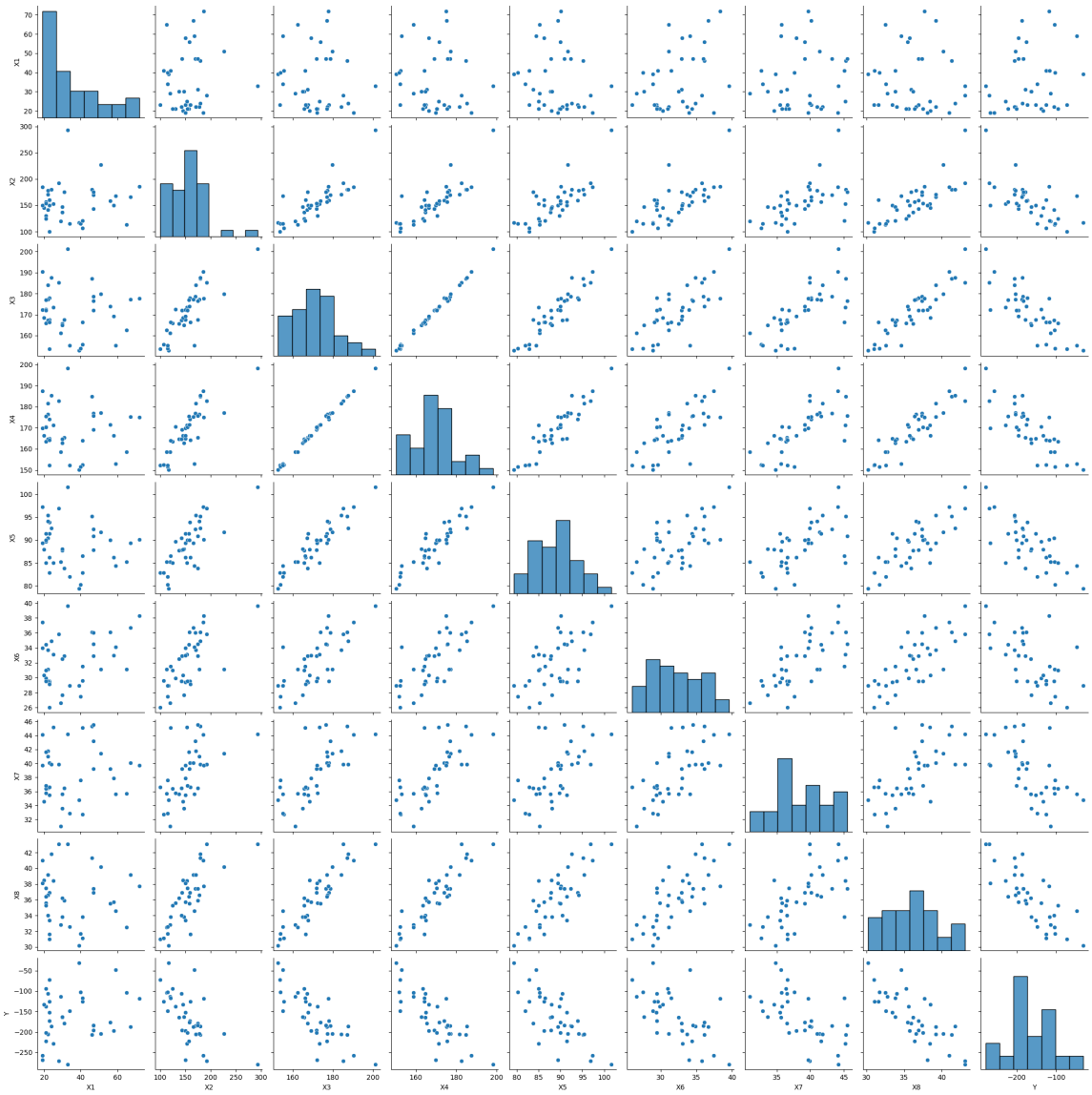
$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1\bar{X}$$

Question 2

See Github repository here for all of the python code used in this question.

a)

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv("./data/data.csv")
sns.pairplot(df)
plt.savefig("./outputs/q2a_pairplots.png")
```



The pairs plots shows a scatter plots between variables in the dataset, while the histogram on the diagonal shows the distribution of each variable. For example, the plot on the fourth row in the second column is the scatter plot of X3 on the y-axis and X2 on the x-axis. Such a matrix is useful for seeing correlations between variables. This is important in linear regression to prevent multicollinearity between the variables, which can skew the coefficients of the regression model and lead to unreliable statistical inferences.

b)

The code in **q2b.py** will produce:

```
Sum of squares for each transformed feature:
X1: 38.000000000000014
X2: 38.0
X3: 38.000000000000014
X4: 37.999999999999986
X5: 37.99999999999998
X6: 38.0
X7: 38.00000000000001
X8: 37.999999999999986
```

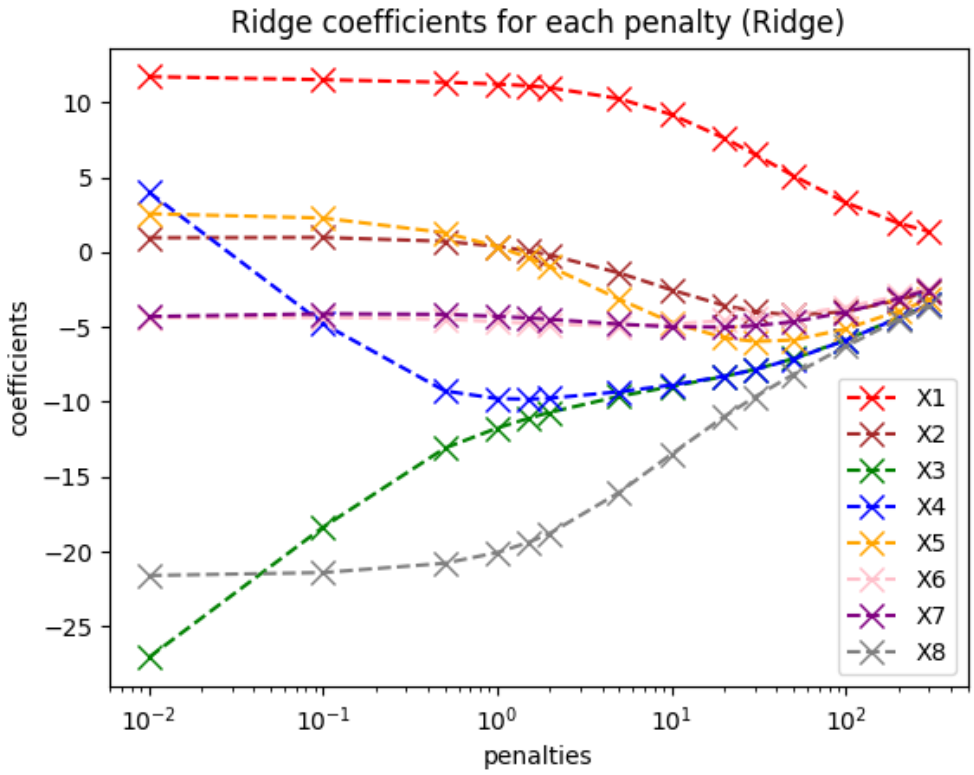
These should all be 38 exactly however they are not due to floating point errors.

c)

Code used (also see in repository here):

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge

penalties = [0.01, 0.1, 0.5, 1, 1.5, 2, 5, 10, 20, 30, 50, 100, 200, 300]

# Import
df = pd.read_csv("./data/transformed_data.csv")
X_columns = df.drop(labels="Y", axis="columns").columns

# Create a model for each penalty and store coeffs
coeffs = []
for penalty in penalties:
    model = Ridge(alpha=penalty)
    model.fit(df.drop(labels="Y", axis="columns"), df["Y"])
    coeffs.append(model.coef_)

# Convert list into np array so that we can transpose
np_coeffs = np.array(coeffs).T

# Plot the coef for each feature
colours = ["red", "brown", "green", "blue", "orange", "pink", "purple", "grey"]

print(coeffs)
ax = plt.gca()
ax.set_prop_cycle(color=colours)
ax.set_xscale("log")

for i, weights in enumerate(np_coeffs):
    ax.plot(penalties, weights, label=X_columns[i])

ax.legend()

plt.xlabel("penalties")
plt.ylabel("coefficients")
plt.title("Ridge coefficients for each penalty")
plt.savefig("./outputs/q2c.png")
```

The following graph is yielded:



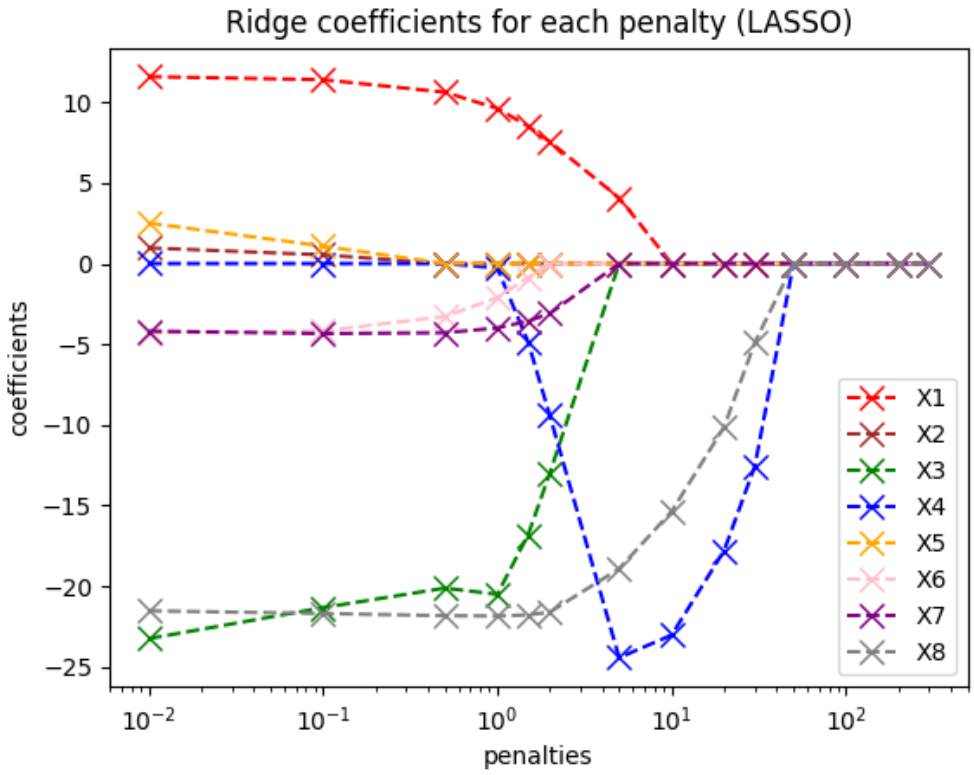
As a general trend, the variance in the coefficients decreases as the penalty for the Ridge regression is increased. Also, by penalty=300, the coefficients are clustered around 0, but do not equal 0. In particular for X3, X4 and X5, these were positively correlated variables and their magnitudes in coefficients were quick to reduce as the penalty increased.

e)

Code used (also see in repository here):

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import Lasso

penalties = [0.01, 0.1, 0.5, 1, 1.5, 2, 5, 10, 20, 30, 50, 100, 200, 300]

# Import
df = pd.read_csv("./data/transformed_data.csv")
X_columns = df.drop(labels="Y", axis="columns").columns

# Create a model for each penalty and store coeffs
coeffs = []
for penalty in penalties:
    model = Lasso(alpha=penalty)
    model.fit(df.drop(labels="Y", axis="columns"), df["Y"])
    coeffs.append(model.coef_)

# Convert list into np array so that we can transpose
np_coeffs = np.array(coeffs).T

# Plot the coef for each feature
colours = ["red", "brown", "green", "blue", "orange", "pink", "purple", "grey"]

print(coeffs)
ax = plt.gca()
ax.set_prop_cycle(color=colours)
ax.set_xscale("log")

for i, weights in enumerate(np_coeffs):
    ax.plot(penalties, weights, label=X_columns[i])

ax.legend()

plt.xlabel("penalties")
plt.ylabel("coefficients")
plt.title("Ridge coefficients for each penalty (LASSO)")
plt.savefig("./outputs/q2e.png")
```

The following graph is yielded:



The first observation made is that coefficients are often set to 0. It's kind of funny to note that all of the coefficients are set to 0 when the penalty reaches a high enough value. For features X3, X4 and X5, when one of them has a non-zero value, the other 2 are often set to 0.

f)

The main difference between LASSO and Ridge was that some of the coefficients found in LASSO were 0. What this means is that a particular feature has no effect on a predicted output. This is why I prefer LASSO because often in a dataset with many features, some of them will not correlate whatsover to the output and the model needs the freedom to recognise this, otherwise then models runs a risk of overfitting to the sample data and is more likely to make false predictions when presented with other data.