MULTICOLLINEARITY USING RIDGE REGRESSION

BY:

M. Ashish Reddy

DSWP -> Batch-5

- Multicollinearity is a phenomenon in which two or more predictors in a multiple regression are highly correlated (R-squared more than 0.7), this can inflate our regression coefficients.
- We can test multicollinearity with the Variance Inflation Factor VIF is the ratio of variance in a model with multiple terms, divided by the variance of a model with one term alone.
- VIF = 1/1-R-squared. A rule of thumb is that if VIF > 10 then multicollinearity is high (a cutoff of 5 is also commonly used).
- To reduce multicollinearity we can use regularization that means to keep all the features but reducing the magnitude of the coefficients of the model.
- **This is a good solution when each predictor contributes to predict the dependent variable**.

- **Ridge Regression** performs a **L2 regularization**, i.e. adds penalty equivalent to square the magnitude of coefficients.
- Minimize the sum of square of coefficients to reduce the impact of correlated predictors.
- **Keeps all predictors in a model**.
- In Ridge Regression, we try to use a trend line that overfit the training data, and so, it has much higher variance then the OLS.
- The main idea of Ridge regression is to fit a new line that doesn’t fit the training data.
- In other words, **we introduce a certain amount on bias into the new trend line**.

- **Ridge Regression Models**
  Following the usual notation, suppose our regression equation is written in matrix form as

$$\underline{\mathbf{Y}} = \mathbf{X}\underline{\mathbf{B}} + \underline{\mathbf{e}}$$

- where **Y** is the dependent variable, **X** represents the independent variables, **B** is the regression coefficients to be estimated, and **e** represents the errors are residuals.

Example of Ridge Regression

```
import mglearn
from sklearn.model_selection import train_test_split
X, y = mglearn.datasets.load_extended_boston()
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=0)
```

Out:

Training set score: 0.89

Test set score: 0.75

- Ridge is a more restricted model, so this model overfit.
- A less complex model means worse performance on the training set, but if you over complex model, this is bad because can overfit.
- How much importance the model places on simplicity versus training set performance can be specified by the user, using the alpha parameter.
- In the previous example, we used the default parameter alpha=1.0 .
- There is no reason why this will give us the best trade-off, though. The optimum setting of alpha depends on the particular dataset we are using.

Increasing alpha forces coefficients to move more toward zero, which decreases training set performance but might help generalization. For example:

```
ridge10 = Ridge(alpha=10).fit(X_train, y_train)

print("Training set score: {:.2f}".format(ridge10.score(X_train,
y_train)))

print("Test set score: {:.2f}".format(ridge10.score(X_test, y_test)))
```

Out:

Training set score: 0.79

Test set score: 0.64

Decreasing alpha allows the coefficients to be less restricted. For very small values of alpha , coefficients are barely restricted at all

```
ridge01 = Ridge(alpha=0.1).fit(X_train, y_train)

print("Training set score: {:.2f}".format(ridge01.score(X_train,
y_train)))

print("Test set score: {:.2f}".format(ridge01.score(X_test, y_test)))
```

Out:

Training set score: 0.93

Test set score: 0.77

THANK YOU