FYS-STK4155 Week 36

Janita Ovidie Sandtrøen Willumsen

(Dated: September 10, 2023)

EXERCISE 1A

1) $MSE = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \frac{1}{n}\sum_{i=0}^{n-1}\left(y_i - \sum_{j=0}^{p-1} X_{ij}\beta_j\right)^2 = C(\beta)$

$X \in \mathbb{R}^{n\times p}$, $X^TX, I \in \mathbb{R}^{p\times p}$

$\bar{\beta} \in \mathbb{R}^{p\times 1}$, $\bar{y} \in \mathbb{R}^{n\times 1}$

a) Will show that the optimal parameters for Ridge Regression is given by

$$\hat{\beta}_{Ridge} = (X^TX + \lambda I)^{-1}X^T\bar{y}$$

A cost function can be defined by

$$\min_{\beta\in\mathbb{R}^p} \frac{1}{n}\|\bar{y} - X\bar{\beta}\|_2^2 + \lambda\|\bar{\beta}\|_2^2$$

$$C(X,\bar{\beta}) = \left\{(\bar{y} - X\bar{\beta})^T(\bar{y} - X\bar{\beta})\right\} + \lambda\bar{\beta}^T\bar{\beta}$$

Want to minimize this by taking derivative with respect to β. First, writing out

$$(\bar{y} - X\bar{\beta}) = y_i - \sum_{j=0}^{p-1} X_{ij}\beta_j$$

$$(\bar{y} - X\bar{\beta})^T = y_j - \sum_{i=0}^{p-1} X_{ji}\beta_i = y_i - \sum_{j=0}^{p-1} X_{ij}\beta_j$$

$$(\bar{y} - X\bar{\beta})^T(\bar{y} - X\bar{\beta}) = \sum_{i=0}^{n-1}\left(y_i - \sum_{j=0}^{p-1} X_{ij}\beta_j\right)^2$$

$$\frac{\partial C_i}{\partial \beta_j} = 2\left(y_i - \sum_{j=0}^{p-1} X_{ij}\beta_j\right)\cdot(-X_{ji}) + 2\lambda\bar{\beta}_i$$

$$= -2X^T(\bar{y} - X\bar{\beta}) + 2\lambda\bar{\beta} = -2X^T\bar{y} + 2X^TX\bar{\beta} + 2\lambda\bar{\beta}$$

$$\frac{\partial^2 C}{\partial\beta_j\partial\beta_j} = 2X_{ji}X_{ij} + 2\lambda$$

$$= 2X^TX + 2\lambda I$$

Setting the derivate lik 0

$$0 = -2X^T(\bar{y} - X\bar{\beta}) + 2\lambda\bar{\beta} = -2X^T\bar{y} + 2X^TX\bar{\beta} + 2\lambda I\bar{\beta}$$

$$2X^TX\beta + 2\lambda I\bar{\beta} = 2X^T\bar{y} \quad \Big| \div 2$$

$$X^TX\bar{\beta} + \lambda I\bar{\beta} = X^T\bar{y}$$

$$(X^TX + \lambda I)\bar{\beta} = X^T\bar{y}$$

$$(X^TX + \lambda I)^{-1}(X^TX + \lambda I)\bar{\beta} = (X^TX + \lambda I)^{-1}X^T\bar{y} = \hat{\beta}_{Ridge}$$

EXERCISE 1B

b) Will show that

$$\hat{y}_{OLS} = X\hat{\beta} = \sum_{j=0}^{p-1} u_j u_j^T y$$

First, the design matrix X can be factorized as

$$X = U\Sigma V^T \quad , \quad U \in \mathbb{R}^{n\times n} \quad , \quad V \in \mathbb{R}^{p\times p} \quad , \quad \Sigma \in \mathbb{R}^{n\times p}$$

OLS result is given by

$$\hat{\beta}_{OLS} = (X^TX)^{-1}X^T\bar{y}$$

$$\hat{y}_{OLS} = X\hat{\beta}_{OLS} = X(X^TX)^{-1}X^T\bar{y}$$

$$= U\Sigma V^T((U\Sigma V^T)^T(U\Sigma V^T))^{-1}(U\Sigma V^T)^T y$$

$$= U\Sigma V^T(V\Sigma^T \underbrace{U^TU}_{\text{ont}}\Sigma V^T)^{-1}V\Sigma^T U^T\bar{y}$$

$$= U\Sigma V^T(V\Sigma^T\Sigma V^T)^{-1}V\Sigma^T U^T\bar{y}$$

$$= U\Sigma V^T(V^T)^{-1}(\Sigma^T\Sigma)^{-1}(V)^{-1}V\Sigma^T U^T y,$$

$$= U\Sigma \underbrace{V^TV}_{I_p}(\Sigma^T\Sigma)^{-1}\underbrace{V^TV}_{I_p}\Sigma^T U^T y$$

$$= U\Sigma(\Sigma^T\Sigma)^{-1}\Sigma^T U^T\bar{y}$$

- kvadratisk matrise $p\times p$, med verdien $\frac{1}{\sigma_{ij}^2}$ på diagonal
- Σ^T er en $p\times n$ matrise med verdien σ_{ji} for $i=j$, tilsvarende for Σ, dvs σ_{ij} for $i=j$

$$\Sigma(\Sigma^T\Sigma)^{-1}\Sigma^T = \Sigma(\Sigma^2)^{-1}\Sigma^T \Rightarrow \sigma_{ij}\cdot\frac{1}{\sigma_{ij}^2}\cdot\sigma_{ji} = 1$$

$$\hat{y}_{OLS} = UU^T\bar{y} = \sum_{j=0}^{p-1}\bar{u}_j\bar{u}_j^T\bar{y}$$

For Ridge

$$\hat{y}_{Ridge} = X\hat{\beta}_{Ridge} = X((X^TX+\lambda I)^{-1}X^T\bar{y})$$

$$= U\Sigma V^T\underbrace{(V\Sigma^T\Sigma V^T + \lambda I)}^{-1}V\Sigma^T U^T\bar{y}$$

invertibel: kvadratisk matrise, determinant > 0

$$= U\Sigma V^T(V(\Sigma^T\Sigma+\lambda I)V^T)^{-1}V\Sigma^T U^T\bar{y}$$

$$= U\Sigma(\Sigma^T\Sigma+\lambda I)^{-1}\Sigma^T U^T\bar{y} = \sum_{i=0}^{p-1}\bar{u}_j\frac{\sigma_i^2}{\sigma_j^2+\lambda}\bar{u}_j^T\bar{y}$$

EXERCISE 2

For smaller values of λ the mean squared error of ridge regression is closer to ordinary least squares. This is expected as λ close to zero would give the same β as for OLS. In addition, by increasing the polynomial order MSE decrease for both Ridge and OLS.

```
import numpy as np
import matplotlib.pyplot as plt
from typing import List, Optional
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score


# Higher order model
def poly_feature_matrix(p: int, x: np.ndarray, scale=False) -> np.ndarray:
    if p < 1:
        print(f"n = {n} is not valid polynomial order")
        return
    X = np.column_stack((np.ones_like(x), x))
    for i in range(2, p+1):
        X = np.column_stack((X, x**i))
    if scale is True:
        X = (X - np.mean(X))
    return X

# General computation of beta
def compute_beta(X: np.ndarray, y: np.ndarray, lmbda: List=None) -> np.ndarray:
    beta = []
    if lmbda is None:
        # beta_ols
        A = np.linalg.inv(X.T @ X) @ X.T
        return A @ y
    else:
        # beta_ridge
        XX = X.T @ X
        p, _ = XX.shape
        for l in lmbda:
            I = np.identity(p) * l
            A = np.linalg.inv(XX + I) @ X.T
            beta.append(A @ y)
        return beta

np.random.seed(35)
n = 100
p = 15
x = np.linspace(-3, 3, n)
y = np.exp(-x**2) + 1.5 * np.exp(-(x-2)**2) + np.random.normal(0, 0.1, x.shape)
X = poly_feature_matrix(p, x)

# Split data and find beta
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
lmbda = [0.0001, 0.001, 0.01, 0.1, 1.0]
beta_ridge = compute_beta(X=X_train, y=y_train, lmbda=lmbda)

# Prediction Ridge
y_tilde = []
y_predict = []
mse_train = []
```

```
r2_train = []
mse_test = []
r2_test = []

for b in beta_ridge:
    y_tilde.append(X_train @ b)
    y_predict.append(X_test @ b)

print("Ridge Regression")
for i in range(len(y_tilde)):
    print(f"Lambda: {lmbda[i]}")
    mse_train.append(mean_squared_error(y_train, y_tilde[i]))
    r2_train.append(r2_score(y_train, y_tilde[i]))
    print(f"Train \nMSE = {mse_train[i]} \n R2 = {r2_train[i]}")

    mse_test.append(mean_squared_error(y_test, y_predict[i]))
    r2_test.append(r2_score(y_test, y_predict[i]))
    print(f"Test \nMSE = {mse_test[i]} \n R2 = {r2_test[i]}\n")

beta_ols = compute_beta(X=X_train, y=y_train)

# Prediction OLS
y_tilde = X_train @ beta_ols
y_predict = X_test @ beta_ols

print("OLS")
mse_train = mean_squared_error(y_train, y_tilde)
r2_train = r2_score(y_train, y_tilde)
print(f"Train \nMSE = {mse_train} \n R2 = {r2_train}")

mse_test = mean_squared_error(y_test, y_predict)
r2_test = r2_score(y_test, y_predict)
print(f"Test \nMSE = {mse_test} \n R2 = {r2_test}\n")
```

TABLE I. Output from running script with polynomial order 5, 10, and 15.

λ		$p = 5$	$p = 10$	$p = 15$
0	Train	0.022483	0.010269	0.009284
0	Test	0.022759	0.012183	0.013650
0.0001	Train	0.022483	0.010269	0.009284
0.0001	Test	0.022759	0.012184	0.013645
0.001	Train	0.022483	0.010269	0.009284
0.001	Test	0.022757	0.012185	0.013598
0.01	Train	0.022483	0.010270	0.009310
0.01	Test	0.022743	0.012205	0.013270
0.1	Train	0.022487	0.010369	0.009562
0.1	Test	0.022609	0.012450	0.012729
1.0	Train	0.022892	0.012649	0.011192
1.0	Test	0.021629	0.014356	0.014077