Homework 5

Lingfeng Zhao

LZ1973

1

(a)

```
from itertools import combinations
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

X = np.random.rand(30, 5)
y = X[:, [3]] * 2
Xtr, Xts, ytr,  yts = train_test_split(X, y)


scores = []
for i in range(Xtr.shape[1]):
    model = LinearRegression()  # Create a linear regression model object
    model.fit(Xtr[:, [i]], ytr)  # Fits the model
    yhat = model.predict(Xts[:, [i]])
    scores.append(np.mean((yhat - yts)**2))

print(f"Best model is order {np.argmin(scores)}, rss is {np.min(scores)}")

```

(b)

```
from itertools import combinations
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

X = np.random.rand(30, 5)
y = X[:, [4]] * 2 + X[:, [2]] * 6
Xtr, Xts, ytr,  yts = train_test_split(X, y)


scores = []
columns = []
for i in combinations(range(Xtr.shape[1]), 2):
    model = LinearRegression()  # Create a linear regression model object
    model.fit(Xtr[:, i], ytr)  # Fits the model
    yhat = model.predict(Xts[:, i])
```

```
    scores.append(np.mean((yhat - yts)**2))
    columns.append(i)

print(f"Best model is order {columns[np.argmin(scores)]}, rss is {np.min(scores)}")
```

(c)

As is shown in previous code, training times is $\binom{p}{k}$. For $k = 10$ and $p = 1000$, total training times is $\binom{1000}{10} \approx 2.634 \times 10^{23}$.

2

(a)

$$\phi(\mathbf{w}) = 0$$

(b)

$$\phi(\mathbf{w}) = \sum_{i=1}^{N} e^{-aw_i}$$

(c)

$$\phi(\mathbf{w}) = \sum_{i=2}^{N} |w_i - w_{i-1}|$$

(d)

$$\phi(\mathbf{w}) = \sum_{i=2}^{N} |sgn(w_i - w_{i-1})|$$

4

```
def normalize(x):
    return (x - np.mean(x, axis=0))/np.std(x, axis=0)

Xtr = normalize(Xtr)
ytr = normalize(ytr)
Xts = normalize(Xts)
yts = normalize(yts)

model = SomeModel() # Creates a model
model.fit(Xtr,ytr) # Fits the model, expecting normalized features
yhat = model.predict(Xts) # Predicts targets given features
Rss = np.mean((yhat - yts)**2)
```

5

```
alphas = np.random.uniform(a, b, p)

Ztr = np.exp(-Xtr*alphas)
Zts = np.exp(-Xts*alphas)

model = Lasso(lam=lam)
beta = model.fit(Ztr, ytr)
yhat = model.predict(Zts)
rss = np.mean((yhat - yts)**2)

beta_tide = model.coef_
opt_alpha = alphas[np.argsort(-beta_tide) < p] # select biggest p in beta_tide
opt_beta = beta_tide[np.argsort(-beta_tide) < p]
```