ECE-GY 6143 Machine Learning HW 05

Tongda Xu

February 15, 2019

1. Question 1:

   a.

```
Rsq = []
for i in range (X.shape[1]):
    Xtri = Xtr[:,i]
    Xtsi = Xts[:,i]
    model = LinearRegression()
    model.fit(Xtri,y)
    yhat = model.predict(Xtsi)
    Rsq[i] = r2_score(yts, yhat)
best_i = np.argmin(Rsq)
```

   b.

```
Rsq = []
for i in range (X.shape[1]-1):
    for j in range(i+1, X.shape[i]):
        Xtri = np.hshack(Xtr[:,i], Xtr[:,j])
        Xtsi = np.hstack(Xts[:,i], Xts[:,j])
        model = LinearRegression()
        model.fit(Xtri,y)
        yhat = model.predict(Xtsi)
        Rsq[i][j] = r2_score(yts, yhat)
best_i, best_j = np.argmin(Rsq)
```

   c.

   C_p^k

   $\frac{1000^10}{10!} > C_{1000}^{10} > 100^{10}$

2. Question 2: **a.**

   $\sum w^2$

b.
$\sum w^2(1 + bool(w < 0))$
c.
$\sum_{j=1}^{n} |w_j - w_{j-1}|^2$
d.
$\sum_{j=1}^{n} |w_j - w_{j-1}|$

3. Question 4

   ```
   Xmean = np.mean(np.vstack(Xtr, Xts), axis = 0)
   Xstd = np.std(np.vstack(Xtr, Xts), axis = 0)
   ymean = np.mean(np.vstack(ytr, yts), axis = 0)
   ystd = np.std(np.vstack(ytr, yts), axis = 0)

   Xtr, Xts -= Xmean
   Xtr, Xts /= Xstd
   ytr, yts -= Ymean
   ytr, yts /= ystd

   model = SomeModel()
   model.fit(Xtr,ytr)
   yhat = model.predict(yts)
   RRS = np.sum((yhat, yts)**2)
   ```

   d.
   from the Graph, $x = 3$
   But there should be a prove analytically, solving β is to minimize:
   $E\{(f(X) - \hat{f}(X))^2\} = \int_0^1 (1 + 2x - x^2 - \beta_0 - \beta_1 x)^2 Pr(x)dx$
   Since $Pr(X)$ is a constant line, after some complex polynomial calculation, we should reach the same result

4. Question 5

   ```
   # Creating uniform a
   alpha = a + np.arange(p)*(b-a)/p

   # fit the model
   Xtr = np.sum(np.exp(
       -np.dot(Xtr.reshape(-1,1), alpha.reshape(1,-1))
       ), axis=1)
   Xts = np.sum(np.exp(
   ```

```
    -np.dot(Xts.reshape(-1,1), alpha.reshape(1,-1))
     ), axis=1)
model = Lasso(lam=lam)
beta = model.fit(Xtr, ytr)

# Measuring error
yhat = model.predict(Xts)
RSS = np.sum((yhat - yts)**2)

# Find coefficient
idx = np.argsort(beta)[-k:]
max_beta = beta[idx]
max_alpha = alpha[idx]
```