# STA208: Homework 1

Prof. James Sharpnack

Due 4/4 in class

In the following, show all your work. Feel free to do all the analytical questions first and then include the code and output second, but the different parts and which question that you are answering should be clearly marked. Code should be as modular as possible, points will be deducted for code that is not reusable (i.e. not broken into general purpose functions), and in the case of gratuitous hard coding.

1. **(Learning paradyms)**

   Decribe the issues involved in the following learning problems using the terminology that we learned in the first lecture. Provide a sentence or two for each problem.

   (a) A 'smart farm' has distributed sensors that detect moisture levels, and the farmers know what are the ideal moisture levels for each plant. They have many controls that adjust the irrigation system and they would like to know what settings produce the most ideal moisture levels.

   (b) Astronomers are trying to map the structure of the universe in terms of how galaxies cluster and form topological structures that they call filaments.

   (c) An online ad company wants to determine which of many ads to show each user based on their browser cookies.

   (d) NASA is mapping the strength of the gravitational field on the surface of Mars. They want you to help with determining its values in a grid of locations on the surface from remote sensing measurements.

   **Solution:**

   (a) This problem is supervised by the moisture levels, so it is a regression setting, and there is an online aspect because losses are incurred daily and learning is continuous. It could be made active by selectively choosing the control variables. Although we have not talked about it, this could be considered a bandit problem.

   (b) This is a modern unsupervised learning problem, where the geometry of the universe needs to be examined. Typically, astro-stats researchers attempt to recover manifolds that the galaxies concentrate around.

   (c) This is a classical online learning problem, since the ad placement has to be chosen when the user is active on the site. Loss is incurred when the user does not click on the problem, and the problem is classification since the response is binary.

   (d) Since we probably will recieve the data after the measurements are made, this is a regression problem. It is complicated by the fact that we are attempting to measure a field.

2. **(Linear Regression)**

   Suppose that we are in the regression setting, $\mathbf{x}_i \in \mathbb{R}^p, y_i \in \mathbb{R}$ are $n$ pairs drawn iid, let $\mathbf{y} = (y_1, \ldots, y_n)^\top$ and $\mathbf{X}^\top = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, and consider the linear regression estimator

   $$\hat{\boldsymbol{\beta}} := \operatorname*{argmin}_{\boldsymbol{\beta} \in \mathbb{R}^p} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2. \tag{1}$$

(a) When is the solution to this program unique? In this case, what is the unique minimizer $\hat{\boldsymbol{\beta}}$?

(b) Give an equation that the minimizers satisfy regardless of uniqueness?

(c) Given a solution to (1), give a reasonable prediction rule $\hat{y} : \mathbb{R}^p \to \mathbb{R}$.

(d) Assume that OLS has a unique solution with probability 1. Suppose that $\mathbb{E}[y_i|\mathbf{x}_i] = \mathbf{x}_i^\top \boldsymbol{\beta}$ for $i = 1 \ldots, n+1$. For a new random draw $(\mathbf{x}_{n+1}, y_{n+1})$, then what is the bias of $\hat{y}(\mathbf{x}_{n+1})$, i.e. $\mathbb{E}[\hat{y}(\mathbf{x}_{n+1}) - y_{n+1}]$?

**Solution:**

(a) When $\mathbf{X}^\top\mathbf{X}$ has a trivial null space, i.e. the only solution to the equation $\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} = \mathbf{0}$ is $\boldsymbol{\beta} = \mathbf{0}$, then the solution is unique. This is because at any solution $\hat{\boldsymbol{\beta}}$ satisfies the zero-gradient condition,

$$2(\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{X}^\top\mathbf{y}) = \mathbf{0}. \tag{2}$$

We can see that this has a unique solution if and only if there is a trivial null space.

(b) Equation (2).

(c) Solving the zero-gradient equation (also known as the normal equations) is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^-\mathbf{X}^\top\mathbf{y} \tag{3}$$

where $.^-$ indicated a pseudoinverse. (It is fine if you assume that it is invertible for this part and do not use the pseudoinverse.)

(d) Let us write out the predictor,

$$\hat{y}(\mathbf{x}_{n+1}) = \mathbf{x}_{n+1}^\top\hat{\boldsymbol{\beta}} = \mathbf{x}_{n+1}^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

Let us condition on the covariates $\mathbf{X}, \mathbf{x}_{n+1}$,

$$\mathbb{E}[\hat{y}(\mathbf{x}_{n+1})|\mathbf{X}, \mathbf{x}_{n+1}] = \mathbf{x}_{n+1}^\top\mathbb{E}[\hat{\boldsymbol{\beta}}|\mathbf{X}, \mathbf{x}_{n+1}] = \mathbf{x}_{n+1}^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbb{E}[\mathbf{y}|\mathbf{X}]$$
$$= \mathbf{x}_{n+1}^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} = \mathbf{x}_{n+1}^\top\boldsymbol{\beta}.$$

This is precisely $\mathbb{E}[y_{n+1}|\mathbf{x}_{n+1}]$ so the bias is $\mathbf{0}$.

3. **(Simulation and ridge regression.)**

(a) Simulate $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^p$ with $p = 12$ and $n = 20$, iid normal with mean $\mathbf{0}$ and variance $\boldsymbol{\Sigma}$ such that

$$\Sigma_{j,k} = \rho^{|j-k|}, \quad j, k = 1, \ldots, p. \tag{4}$$

for $0 < \rho < 1$. Draw $\boldsymbol{\beta} \in \mathbb{R}^p$ such that $\beta_j$ are iid normal with mean 0 and variance 1, and $y_i$ independently normal with mean $\mathbf{x}_i^\top\boldsymbol{\beta}$ and variance 25. Print out your code (not the output), which should consist of functions for generating these objects.

(b) Derive an analytical expression for the solution to ridge regression,

$$\hat{\boldsymbol{\beta}} := \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2, \tag{5}$$

as a function of $\mathbf{X}, \mathbf{y}, \lambda$.

(c) Provide code for solving ridge regression. Use any linear solver you like.

(d) Set $\rho = 0.5$. Simulate the mean square error, $\mathbb{E}\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|_2^2$, for many values of $\lambda$. Be sure that you see instances of overfitting and underfitting and can clearly see the point where $\lambda$ is optimal. Plot this curve as functions of lambda and include your code.

**Solution:**

(a)
```
import numpy as np
from matplotlib import pyplot as plt

def gaussian_geom(n,p,rho):
    if rho == 0.:
        Sigma = np.eye(p)
    else:
        Sigma = rho**np.abs(np.outer(np.ones(p),np.arange(p))
                - np.outer(np.arange(p),np.ones(p)))
    mu = np.zeros(p)
    return np.random.multivariate_normal(mu,Sigma,n)

def lin_model_geom(n,p,rho):
    X = gaussian_geom(n,p,rho)
    beta = gaussian_geom(1,p,0.).T
    ybar = X.dot(beta)
    y = ybar + 5.*gaussian_geom(1,n,0.).T
    return y, beta, X
```

(b) The gradient of the objective is

$$2(\mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} - \mathbf{X}^\top \mathbf{y}) + 2\lambda\boldsymbol{\beta}$$

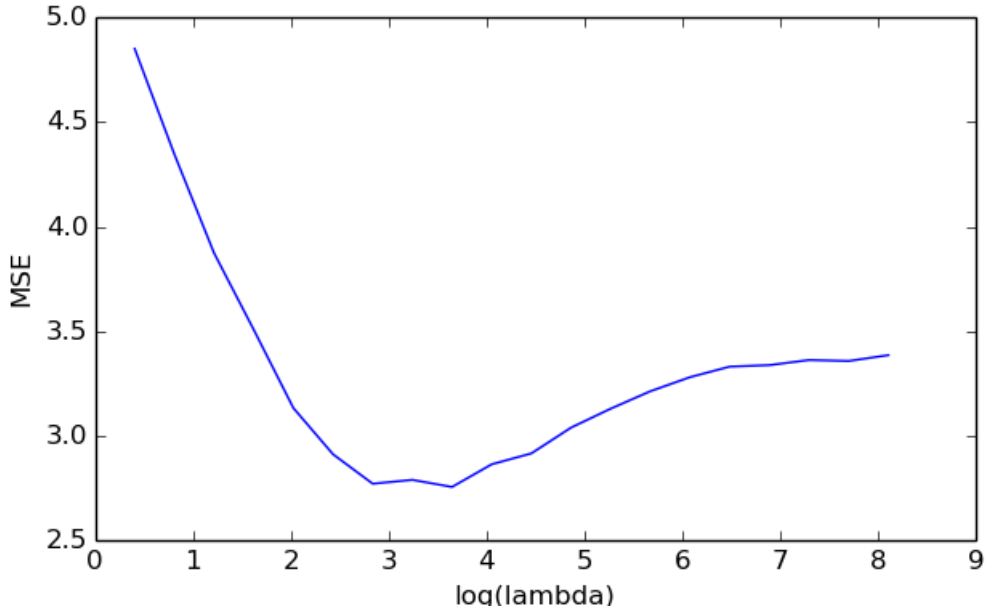
and setting this equal zero to gives us

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^\top \mathbf{y}.$$

(c)
```
def ridge_sol(X,y,lamb):
    n,p = X.shape
    G = X.T.dot(X) + lamb*np.eye(p)
    xy = X.T.dot(y)
    betahat = np.linalg.solve(G,xy)
    return betahat
```

(d)
```
def sim_error(n,p,rho,lamb):
    y, beta, X = lin_model_geom(n,p,rho)
    betahat = ridge_sol(X,y,lamb)
    return np.sum((betahat - beta)**2.)

n = 20
p = 12
rho = 0.5
T = 1000
L = 20
lamb_seq = 1.5**(np.arange(L) + 1.)
err_seq = np.zeros(L)
for i in range(L):
    lamb = lamb_seq[i]
    err_seq[i] = np.mean([sim_error(n,p,rho,lamb) for j in range(T)])

plt.plot(np.log(lamb_seq),err_seq)
```

```
plt.xlabel("log(lambda)")
plt.ylabel("MSE")
plt.show()
```




4. **(Airfoil)**

Download the airfoil dataset, which is linked in the homework section of the course site. We will focus on predicting the scaled sound pressure, which is the 6th row.

(a) Set aside a test set at random.

(b) Form the coefficients for ordinary least squares with the training set (use any built in function/linear solver you like to fit the coefficents). Write a function with a new $\mathbf{x}$ and $\hat{\boldsymbol{\beta}}$ as arguments and returns the prediction.

(c) Write a function, or use a package of your choice, that takes a new $\mathbf{x}$, $k$, and the training data, and outputs the k-nearest neighbor prediction with Euclidean distance.

(d) Write a function, or use a package of your choice, that takes a new $\mathbf{x}$, a bandwidth parameter, and the training data, and outputs the kernel prediction with boxcar kernel and Euclidean distance.

(e) Evaluate your methods on the test set, calculating the test error (empirical risk on the test set). Vary the tuning parameters and plot the test error as a function of the tuning parameters. Plot the OLS test error as a horizonal line (since there is no tuning parameter).

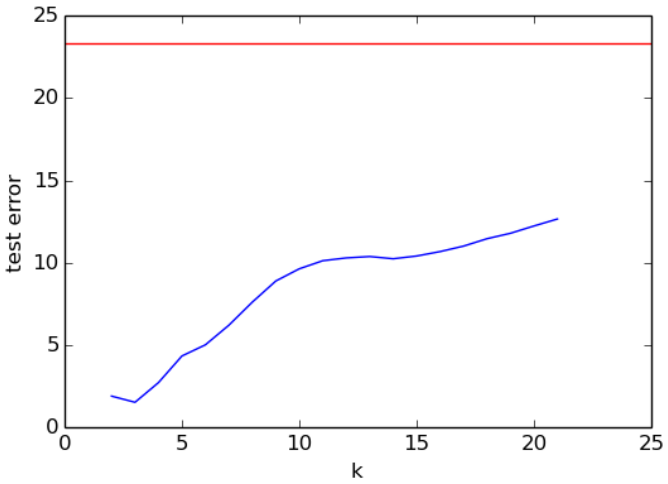
(f) Is the best test error a good estimate of the true risk for these methods? Why/why not? What can be done to estimate the true risk?

**Solution:**

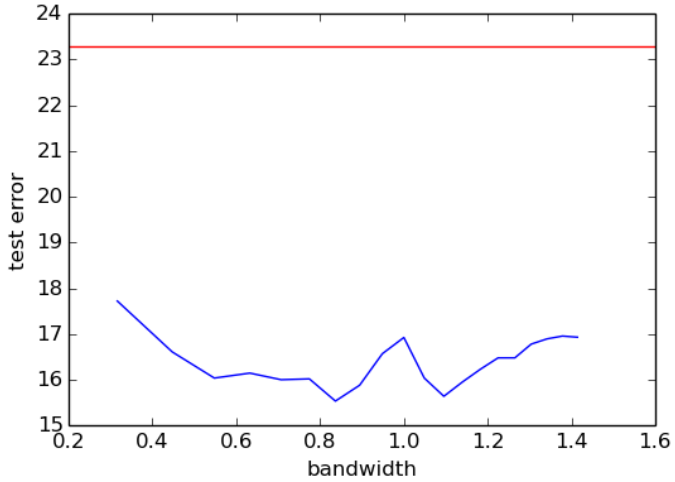
```
import numpy as np
import scipy as sci
from sklearn import linear_model, neighbors, preprocessing, cross_validation
from matplotlib import pyplot as plt
```

```
### The answers to (b),(c),(d) are in the following functions, I use the regression
### and knn class from sklearn, and make my own kernel smoother object
### I went out on a limb and when a point in the test set had no training point
###  within the bandwidth
class BoxKernelSmoother:
    def __init__(self,Xtr,ytr,bandwidth):
        self.Xtr = Xtr
        self.ytr = ytr
        self.bandwidth = bandwidth
    def predict(self,Xte):
        Xtr = self.Xtr
        ytr = self.ytr
        ntr, p = Xtr.shape
        nte = Xte.shape[0]
        inner_prod = Xtr.dot(Xte.T)
        tr_norms = np.sum(np.abs(Xtr)**2,axis=-1)
        te_norms = np.sum(np.abs(Xte)**2,axis=-1)
        tr_nmat = np.outer(tr_norms,np.ones(nte))
        te_nmat = np.outer(np.ones(ntr),te_norms)
        D_sq = tr_nmat + te_nmat - 2.* inner_prod
        W = 1.*(D_sq.T < self.bandwidth**2.)
        W_norms = W.dot(np.ones(ntr))
        W_norms.shape = (nte,1)
        ypred = W.dot(ytr) / W_norms
        ypred[W_norms == 0.] = np.mean(ytr)
        return ypred

def knn_pred(knn,X,k=5):
    knn.n_neighbors = k
    return knn.predict(X)

def ols_pred(ols,X):
    return ols.predict(X)

def bks_pred(ks,X,bw=1.):
    ks.bandwidth = bw
    return ks.predict(X)

def test_error(yhat,y):
    n = len(y)
    return np.sum((y - yhat)**2.)/n

M = np.matrix(np.loadtxt("airfoil_self_noise.dat"))
y = M[:,5]
X = M[:,0:5]

## Answer (a), I also scale the design
X = preprocessing.scale(X)
Xtr, Xte, ytr, yte = cross_validation.train_test_split(X, y, test_size=0.4)

k = 5
knn = neighbors.KNeighborsRegressor(k)
```

```
knn.fit(X,y)

ols = linear_model.LinearRegression()
ols.fit(Xtr,ytr)
yhat_ols = ols_pred(ols,Xte)
ols_err = test_error(yhat_ols,yte)

bandwidth = .2
ks = BoxKernelSmoother(Xtr,ytr,bandwidth)

L = 20
bw_seq = (np.arange(L)/10. + .1)**0.5
k_seq = range(2,L+2)
knn_err = []
ks_err = []
for i in range(L):
    k = k_seq[i]
    bw = bw_seq[i]
    yhat_knn = knn_pred(knn,Xte,k)
    yhat_ks = bks_pred(ks,Xte,bw)
    knn_err.append(test_error(yhat_knn,yte))
    ks_err.append(test_error(yhat_ks,yte))

plt.plot(k_seq,knn_err)
plt.xlabel("k")
plt.ylabel("test error")
plt.axhline(ols_err,color='r')
plt.show()

plt.plot(bw_seq,ks_err)
plt.xlabel("bandwidth")
plt.ylabel("test error")
plt.axhline(ols_err,color='r')
plt.show()
```




The red line is the ols error and the blue line is the knn test error as a function of k.




The red line is the ols error and the blue line is the kernel smoother test error as a function of bandwidth.

(e) If the methods are tuned using the test error then it will not be a good estimate of the true risk, since it would be used for both tuning and testing. You need to further split the data into training, validation, and test sets.