Homework #1 Solutions

1 Probability and calculus

1.1 Variance and covariance

a)
$$\begin{aligned} Cov(X,Y) &= \mathbb{E}(X-\mathbb{E}(X))(Y-\mathbb{E}(Y))^T \\ &= \mathbb{E}(XY^T) - \mathbb{E}(X\mathbb{E}(Y^T)) - \mathbb{E}(\mathbb{E}(X)Y^T) + \mathbb{E}(\mathbb{E}(X)\mathbb{E}(Y^T)) \\ &= \mathbb{E}(X)\mathbb{E}(Y^T) - \mathbb{E}(X)\mathbb{E}(Y^T) - \mathbb{E}(X)\mathbb{E}(Y^T) + \mathbb{E}(X)\mathbb{E}(Y^T) \\ &= 0 \end{aligned} \tag{1}$$

b)
$$\begin{aligned} \mathbb{E}(X+AY) &= \mathbb{E}([X_i] + [(AY)_i]) \\ &= [\mathbb{E}(X_i + (AY)_i)] \\ &= [\mathbb{E}(X_i) + \mathbb{E}((AY)_i)] \\ &= \mathbb{E}(X) + \mathbb{E}(AY) \\ &= \mathbb{E}(X) + \mathbb{E}([\sum_{k=1}^{m} a_{i,k}Y_k]) \\ &= \mathbb{E}(X) + [\sum_{k=1}^{m} a_{i,k}\mathbb{E}(Y_k)] \\ &= \mathbb{E}(X) + A\mathbb{E}(Y) \end{aligned} \tag{2}$$

$$\begin{aligned} Var(X+AY) &= Var([X_i] + [AY_i]) \\ &= Var(X) + Var(AY) \\ &= Var(X) + \mathbb{E}(AY - \mathbb{E}(AY))(AY - \mathbb{E}(AY))^T \\ &= Var(X) + \mathbb{E}(A(Y - \mathbb{E}(Y)))(A(Y - \mathbb{E}(Y)))^T \\ &= Var(X) + A\mathbb{E}(Y - \mathbb{E}(Y))(Y - \mathbb{E}(Y))^T A^T \\ &= Var(X) + AVar(Y)A^T \end{aligned} \tag{3}$$

c)
$$\begin{aligned} \mathbb{E}(AX) &= A\mathbb{E}(X) \\ &= A\mu \end{aligned} \tag{4}$$

$$\begin{aligned} Var(AX) &= AVar(X)A^T \\ &= A\Sigma A^T \end{aligned} \tag{5}$$

Since any linear combination of Gaussians are again a Gaussian, the resulting random vector is also a Gaussian. Its distribution is completely determined by its mean and variance.

1.2 Densities

a) Yes. For example, the probability density function of a random value uniformly distributed between 0 and $\frac{1}{2}$ has value 2 in the interval $[0, \frac{1}{2}]$ such that the volume over the domain integrates to one.

b)
$$\begin{aligned} p(x) &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \\ &= \frac{10}{\sqrt{2\pi}} e^{-50x^2} \end{aligned} \tag{6}$$

c)
$$p(0) = \frac{10}{\sqrt{2\pi}} \tag{7}$$

d) Since X is continuous, $P(X = 0) = 0$.

1.3 Calculus

a)
$$\nabla_x \, x^T y = \Big[\frac{d\sum_{k=1}^{m} x_k y_k}{dx_i}\Big] = [y_i] = y \tag{8}$$

b)
$$\nabla_x \, x^T x = \Big[\frac{d\sum_{k=1}^{m} x_k^2}{dx_i}\Big] = [2x_i] = 2x \tag{9}$$

c)
$$\nabla_x \, x^T A x = \Big[\frac{d\sum_{k=1}^{m}\sum_{j=1}^{m} a_{k,j} x_j x_k}{dx_i}\Big] = \Big[\sum_{j=1}^{m} a_{i,j} x_j + \sum_{k=1}^{m} a_{k,i} x_k\Big] = Ax + A^T x = (A + A^T)x \tag{10}$$

d)
$$\nabla_x \, Ax = \Big[\frac{d\sum_{k=1}^{m} a_{j,k} x_k}{dx_i}\Big] = [a_{j,i}] = A^T \tag{11}$$

2 Regression

2.1 Linear regression

a)
$$\begin{aligned} \mathbb{E}(\hat{\beta}) &= \mathbb{E}((X^T X)^{-1} X^T Y) \\ &= (X^T X)^{-1} X^T \, \mathbb{E}(Y) \\ &= (X^T X)^{-1} X^T X \beta \\ &= \beta \end{aligned} \tag{12}$$

$$\begin{aligned} Var(\hat{\beta}) &= Var((X^T X)^{-1} X^T Y) \\ &= ((X^T X)^{-1} X^T) \sigma^2 I ((X^T X)^{-1} X^T)^T \\ &= \sigma^2 (X^T X)^{-1} X^T X (X^T X)^{-1} \\ &= \sigma^2 (X^T X)^{-1} \end{aligned} \tag{13}$$

$$\hat{\beta} \sim \mathcal{N}(\beta, \sigma^2 (X^T X)^{-1})$$

b)
$$\begin{aligned} log[p(y|X,\beta)] &= log\Big[\frac{1}{(2\pi)^{n/2}|\sigma^2 I|^{1/2}} e^{-\frac{(y-X\beta)^T (y-X\beta)}{2\sigma^2}}\Big] \\ &= -log\Big[(2\pi)^{n/2}|\sigma^2 I|^{1/2}\Big] - \frac{(y-X\beta)^T (y-X\beta)}{2\sigma^2} \end{aligned} \tag{14}$$

$$\frac{dlog[p(y|X,\beta)]}{d\beta} = \frac{X^T (y - X\beta)}{\sigma^2}$$

$$c) \qquad \hat{\beta}_i \sim \mathcal{N}(\beta, \sigma^2 (X^T X)^{-1}_{i,i}) \tag{15}$$

$$\begin{aligned} P(|\hat{\beta}_i - \beta_i| \leq \epsilon) &= P(\hat{\beta}_i \leq \beta + \epsilon) - P(\hat{\beta}_i \leq \beta - \epsilon) \\ &= P\Big(Z \leq \frac{\epsilon}{\sqrt{\sigma^2 (X^T X)^{-1}_{i,i}}}\Big) - P\Big(Z \leq \frac{-\epsilon}{\sqrt{\sigma^2 (X^T X)^{-1}_{i,i}}}\Big) \\ &= 2\Phi\Big(\frac{\epsilon}{\sqrt{\sigma^2 (X^T X)^{-1}_{i,i}}}\Big) - 1 \end{aligned} \tag{16}$$

2.2 Ridge regression

$$\begin{aligned} a) \qquad log[p(\beta|y,X)] &= log\Big[\frac{1}{C} \frac{1}{(2\pi)^{p/2} |\tau^2 I|^{1/2}} e^{-\frac{\beta^T \beta}{2\tau^2}} \frac{1}{(2\pi)^{n/2} |\sigma^2 I|^{1/2}} e^{-\frac{(y - X\beta)^T (y - X\beta)}{2\sigma^2}}\Big] \\ &= -\frac{\beta^T \beta}{2\tau^2} - \frac{(y - X\beta)^T (y - X\beta)}{2\sigma^2} + C \\ \frac{dlog[p(\beta|y,X)]}{d\beta} &= -\frac{\beta}{\tau^2} + \frac{X^T (y - X\beta)}{\sigma^2} \\ 0 &= -\frac{\sigma^2}{\tau^2}\beta + X^T y - X^T X \beta \\ \hat{\beta}_{MAP} &= (X^T X + \frac{\sigma^2}{\tau^2} I)^{-1} X^T y \end{aligned} \tag{17}$$

$$b) \qquad \widetilde{X} = \begin{pmatrix} X \\ \sqrt{\lambda} I_m \end{pmatrix} \; \widetilde{y} = \begin{pmatrix} y \\ 0_{m \times 1} \end{pmatrix} \tag{18}$$

$$\begin{aligned} \widetilde{X}^T \widetilde{X} &= X^T X + \lambda I \\ \widetilde{X}^T \widetilde{y} &= X^T y \\ \hat{\beta} &= (\widetilde{X}^T \widetilde{X})^{-1} \widetilde{X}^T \widetilde{y} \\ &= (X^T X + \lambda I)^{-1} X^T y \end{aligned} \tag{19}$$

3 Cross-validation

Figure 1 shows the training error, test error, 5-fold, and 10-fold cross validation errors for all λ values. The 5-fold cross validation procedure recommends $\lambda = 0.44$ and the 10-fold cross validation procedure recommends $\lambda = 0.32$, where error is lowest. Any value in this range is reasonable.

The training error increases as λ increases. This occurs because a higher value of λ results in a more constrictive constraint on parameter values, and therefore less flexibility to perfectly fit the training data. The testing error first decreases, because overfitting is prevented by the regularization procedure, but then increases because the constraint becomes overly tight. The shapes of the 5-fold and 10-fold cross validation errors are similar to the testing error for the same reasons.

The training error is lower than 5-fold, 10-fold, and testing error since the model is selected to best fit this data. The 5-fold, 10-fold, and testing errors may alternate in which is highest, depending on the properties of the data set and the random seed selected.

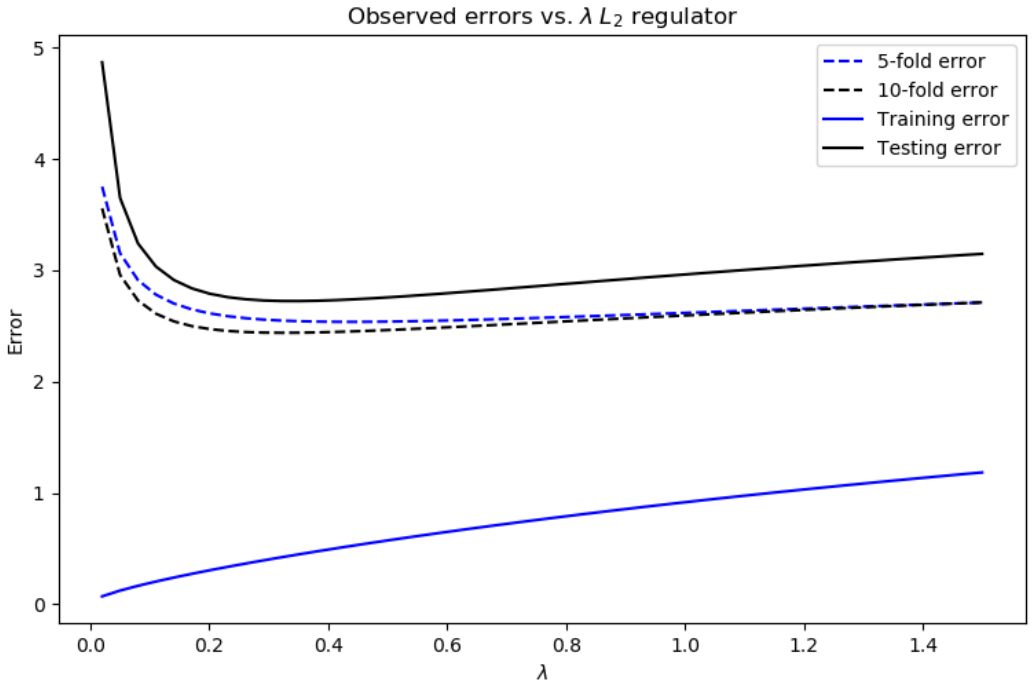

Figure 1: Relationship between average squared-error and ridge regularizing parameter

```
import numpy as np
import scipy.io as sio
import matplotlib.pyplot as plt

def shuffle_data(data):

        return np.random.permutation(data)

def split_data(data, num_folds, fold):

        fold_length = len(data) // num_folds + (len(data) % num_folds > 0)
        data_fold_ids = np.arange((fold - 1)* fold_length, \
                min(int(fold * fold_length), len(data)))
        return data[data_fold_ids,], \
                data[[i for i in range(len(data)) if i not in data_fold_ids],]

def train_model(data, lambd):

        y_train = data[:,0].reshape(-1, 1)
        x_train = data[:,1:]

        return np.linalg.solve(np.dot(np.transpose(x_train), x_train) + \
                lambd * np.identity(len(x_train[0])), \
                np.dot(np.transpose(x_train), y_train))

def predict(data, model):

        return np.dot(data[:,1:], model)

def loss(data, model):
        error_dif = predict(data, model) - data[:,0].reshape(-1,1)
        return np.asscalar(np.dot(np.transpose(error_dif), error_dif) / len(data))
```

```
def cross_validation(data, num_folds, lambd_seq):

        data = shuffle_data(data)
        cv_error = []

        for lambd in lambd_seq:

                cv_loss = 0

                for fold in range(1, num_folds + 1):
                        validation_data, train_data = split_data(data, num_folds, fold)
                        model = train_model(train_data, lambd)
                        cv_loss += loss(validation_data, model)

                cv_error.append(cv_loss / num_folds)

        return cv_error

def train_test_error(data, test_data, lambd_seq):

        train_error = []
        test_error = []

        for lambd in lambd_seq:
                model = train_model(data, lambd)
                train_error.append(loss(data, model))
                test_error.append(loss(test_data, model))

        return train_error, test_error

#import data
dataset = sio.loadmat("C:/Users/Ilan/Desktop/MSc_Statistics/Data_Science/dataset.mat")
data_train_X = dataset['data_train_X']
data_train_y = dataset['data_train_y'][0]
data_test_X = dataset['data_test_X']
data_test_y = dataset['data_test_y'][0]

train_data = np.hstack((data_train_y.reshape(-1,1), data_train_X))
test_data = np.hstack((data_test_y.reshape(-1,1), data_test_X))

#calculate values to plot
np.random.seed(4)

lambdas = np.linspace(0.02, 1.5, num = 50)
cross5_error = cross_validation(train_data, 5, lambdas)
cross10_error = cross_validation(train_data, 5, lambdas)
train_error, test_error = train_test_error(train_data, test_data, lambdas)

#find minimum error
print(lambdas[np.argmin(cross5_error)])
print(lambdas[np.argmin(cross10_error)])

#create plot
plt.plot(lambdas, cross5_error, 'b--', label = '5-fold error')
plt.plot(lambdas, cross10_error, 'k--', label = '10-fold error')
plt.plot(lambdas, train_error, 'b-', label = 'Training error')
plt.plot(lambdas, test_error, 'k-', label = 'Testing error')
plt.legend(loc='upper right')

plt.xlabel(r'$\lambda$')
plt.ylabel('Error')
plt.title(r'Observed errors vs. $\lambda \; L_2$ regulator')
plt.show()
```