Brenden Hein

HW 3

CSE 847

Linear Algebra 3

1. Let $A \in R^{m\,x\,n}$ with rank = n; prove $\left\|A(A^TA)^{-1}A^T\right\|_2 = 1$

   $Since\ A\ is\ m\ x\ n\ a\ rectangular\ matrix\ with\ a\ rank\ n, it\ is\ full\ column\ rank$
   $This\ implies\ that\ m \geq n$
   $Since\ A\ is\ full\ column\ rank, this\ implies\ there\ exists\ a\ pseudoinverse\ of\ A$

   $A^\dagger = (A^TA)^{-1}A^T$
   $\left\|AA^\dagger\right\|_2 = 1$
   $\left\|A(A^TA)^{-1}A^T\right\|_2 = 1$
   $\left\|AA^{-1}A^{T^{-1}}A^T\right\|_2 = 1$
   $\left\|IA^{T^{-1}}A^T\right\|_2 = 1$
   $\left\|II\right\|_2 = 1 \rightarrow \left\|I\right\|_2 = 1$
   $I = \begin{bmatrix} 1 & \ldots & 0 \\ \ldots & \ldots & \ldots \\ 0 & \ldots & 1 \end{bmatrix}$

   $\boldsymbol{Since\ you're\ talking\ the\ 2-norm\ of\ the\ identity\ matrix, the\ max\ will\ always\ be\ 1}$

2. Let A and B be 2 semi-definite positive matrices in $R^{n\,x\,n}$; Prove or disprove the following
   a. $A + B$ is positive semi-definite

      $If\ A + B\ is\ positive\ semidefinite$:
      $\forall_x(x \in R^n \rightarrow x^T(A+B)x \geq 0)$
      $Basically, for\ all\ x\ vectors\ in\ R^n, x^T(A+B)x\ will\ always\ be\ greater\ than\ 0$

      $x^T(A+B)x = x^TAx + x^TBx$
      $Since\ A\ and\ B\ are\ positive\ semidefinite$:
      $x^TAx \geq 0\ \ and\ \ x^TBx \geq 0$
      $Thus, x^T(A+B)x = x^TAx + x^TBx \geq 0$

      TRUE

b. AB is positive semi-definite

$If\ AB\ is\ positive\ semidefinite$:
$\forall_x(x\ \in R^n \rightarrow x^T(AB)x \geq 0)$
$Basically, for\ all\ x\ vectors\ in\ R^n, x^T(AB)x\ will\ always\ be\ greater\ than\ 0$
$While\ we\ know\ x^TAx \geq 0\ \ and\ \ x^TBx \geq 0$
$We\ don't\ know\ that\ x^TABx \geq 0$

FALSE

c. B^T is positive semi-definite

$If\ B^T\ is\ positive\ semidefinite$:
$\forall_x(x\ \in R^n \rightarrow x^T(B^T)x \geq 0)$
$Basically, for\ all\ x\ vectors\ in\ R^n, x^T(B^T)x\ will\ always\ be\ greater\ than\ 0$
$Since\ B\ is\ positive\ semidefinite, B\ is\ symmetric$
$Thus, B^T = B$
$x^T(B^T)x = \ x^T(B)x$
$Since\ we\ know\ x^TAx \geq 0\ \ and\ \ x^TBx \geq 0,\ B^T\ is\ postive\ semidefinite$

TRUE

Linear Classification

1. Given a set of data points, $\{x^n\}$, we can define the convex hull to be the set of points given by $x = \sum_n \alpha_n x_n$, where $\alpha_n \geq 0$ and $\sum_n \alpha_n = 1$. Consider a second set of points, $\{y^n\}$, together, with their corresponding convex hull. By definition, the 2 sets of points will be linearly separable if there exists a vector, $\hat{w}$, and scalar, w_0, such that $\hat{w}x_n + w_0 > 0$ for all x_n, and $\hat{w}y_n + w_0 < 0$ for all y_n. Show that if the 2 convex hulls intersect, their points are not linearly separable. Also show that if the 2 convex hulls don't intersect, they are linearly separable.

$$\hat{w}x_n + w_0 = \ \hat{w}(\sum_n \alpha_n x_n) + w_0$$

$$\hat{w}y_n + w_0 = \ \hat{w}(\sum_n \alpha_n y_n) + w_0$$

$If\ the\ two\ functions\ intersect\ at\ some\ point$:

$\hat{w}^Tx_n + w_0 \leq 0$ and $\hat{w}^Ty_m + w_0 \geq 0\ at\ some\ point(s)$

$\hat{w}^Tx_n + w_0 \leq \hat{w}^Ty_m + w_0 \ \ \rightarrow \ \ \hat{w}^Tx_n \leq \hat{w}^Ty_m$

$which\ means\ there\ is\ an\ x\ value\ that\ exists\ inside\ the\ convex\ hull\ of\ y,$

$thus, they\ are\ not\ linearly\ seperable; conversly, if\ no\ such\ points\ exist\ where\ this$

$is\ true, it\ means\ that\ that\ there\ is\ a\ clear\ seperation\ where\ all\ points, x_n, follow$

$\widehat{w}x_n + w_0 > 0\ and\ all\ points, y_n, follow\ \widehat{w}y_n + w_0 < 0$

2. By making use of $y = w^T x,\ m_k = w^T m_k,$ and $s_k^2 = \sum_{n\in c_k}(y_n - m_k)^2$, show that the Fischer Criterion: $J(w) = \frac{(m_2-m_1)^2}{s_1^2+s_2^2}$ can be written in the form: $J(w) = \frac{w^T S_B w}{W^T S_w w}$
   $S_B = (m_2 - m_1)(m_2 - m_1)^T$
   $S_w = \sum_{n\in c_1}(x_n - m_1)(x_n - m_1)^T + \sum_{n\in c_2}(x_n - m_2)(x_n - m_2)^T$

$$J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

$$J(w) = \frac{(w^T m_2 - w^T m_1)^2}{s_1^2 + s_2^2}$$

$$J(w) = \frac{(w^T m_2 - w^T m_1)^2}{\sum_{n\in c_1}(y_n - m_1)^2 + \sum_{n\in c_2}(y_n - m_2)^2}$$

$$J(w) = \frac{(w^T m_2 - w^T m_1)^2}{\sum_{n\in c_1}(w^T x_n - w^T m_1)^2 + \sum_{n\in c_2}(w^T x_n - w^T m_2)^2}$$

$$J(w) = \frac{(w^T m_2 - w^T m_1)(w^T m_2 - w^T m_1)}{\sum_{n\in c_1}(w^T x_n - w^T m_1)(w^T x_n - w^T m_1) + \sum_{n\in c_2}(w^T x_n - w^T m_2)(w^T x_n - w^T m_2)}$$

$Since\ (w^T m_2 - w^T m_1)\ and\ (w^T x_n - w^T m_1)\ are\ scalars, aa^T == a^2$

$$J(w) = \frac{(w^T m_2 - w^T m_1)(w^T m_2 - w^T m_1)^T}{\sum_{n\in c_1}(w^T x_n - w^T m_1)(w^T x_n - w^T m_1)^T + \sum_{n\in c_2}(w^T x_n - w^T m_2)(w^T x_n - w^T m_2)^T}$$

$$J(w) = \frac{(w^T m_2 - w^T m_1)(m_2^T w - m_1^T w)}{\sum_{n\in c_1}(w^T x_n - w^T m_1)(x_n^T w - m_1^T w) + \sum_{n\in c_2}(w^T x_n - w^T m_2)(x_n^T w - m_1^T w)}$$

$$J(w) = \frac{w^T(m_2 - m_1)(m_2^T - m_1^T)w}{\sum_{n\in c_1} w^T(x_n - m_1)(x_n^T - m_1^T)w + \sum_{n\in c_2} w^T(x_n - m_2)(x_n^T - m_1^T)\,w}$$

$$J(w) = \frac{w^T(m_2 - m_1)(m_2 - m_1)^T w}{\sum_{n\in c_1} w^T(x_n - m_1)(x_n - m_1)^T w + \sum_{n\in c_2} w^T(x_n - m_2)(x_n - m_2)^T\,w}$$

$$J(w) = \frac{w^T S_B w}{W^T S_w w}$$

3. Using the definitions of the between-class and within-class covariance matrices given by $S_B = (m_2 - m_1)(m_2 - m_1)^T$ and $S_w = \sum_{n\in c_1}(x_n - m_1)(x_n - m_1)^T + \sum_{n\in c_2}(x_n - m_2)(x_n - m_2)^T$ respectively, together with $w_0 = -w^T m$ and $m = \frac{1}{N}(N_1 m_1 + N_2 m_2)$ and the choice of target values described in Section 4.1.5, show that the expression

$\sum_{n=1}^{N}(w^T x_n + w_0 - t_n)x_n = 0$ that minimizes the sum-of-squares error function can be written in the form $(S_W + \frac{N_1 N_2}{N} S_B)w = N(m_1 - m_2)$

$$\sum_{n=1}^{N}(w^T x_n + w_0 - t_n)x_n = 0$$

$$\sum_{n=1}^{N}(w^T x_n - w^T m \; - t_n)x_n = 0$$

$$\sum_{n=1}^{N}(w^T x_n - w^T \frac{1}{N}(N_1 m_1 + N_2 m_2))x_n = 0 \;\; since \sum_{n=1}^{N} t_n = 0$$

$$\sum_{n=1}^{N}(w^T x_n - w^T \left(\frac{N_1 m_1}{N} + \frac{N_2 m_2}{N}\right))x_n = 0$$

$$\sum_{n=1}^{N}(w^T x_n - w^T \frac{N_1 m_1}{N} - w^T \frac{N_2 m_2}{N})x_n = 0$$

$$\sum_{n=1}^{N}(w^T x_n x_n - w^T \frac{N_1 m_1}{N} x_n - w^T \frac{N_2 m_2}{N} x_n) = 0$$

$$\sum_{n=1}^{N} w^T x_n x_n - \sum_{n=1}^{N} w^T \frac{N_1 m_1}{N} x_n - \sum_{n=1}^{N} w^T \frac{N_2 m_2}{N} x_n = 0$$

$$\sum_{n=1}^{N} w^T x_n x_n = \sum_{n=1}^{N} w^T \frac{N_1 m_1}{N} x_n + \sum_{n=1}^{N} w^T \frac{N_2 m_2}{N} x_n$$

$$w^T \sum_{n=1}^{N} x_n x_n = w^T \frac{N_1 m_1}{N} \sum_{n=1}^{N} x_n + w^T \frac{N_2 m_2}{N} \sum_{n=1}^{N} x_n$$

$$w^T \sum_{n=1}^{N} x_n x_n = w^T N_1 m_1 \left(\frac{N_1 m_1}{N} + \frac{N_2 m_2}{N}\right) + w^T N_2 m_2 \left(\frac{N_1 m_1}{N} + \frac{N_2 m_2}{N}\right)$$

$$w^T \sum_{n=1}^{N} x_n x_n = w^T N_1 m_1 \left(\frac{N_1 m_1}{N} + \frac{N_2 m_2}{N}\right) + w^T N_2 m_2 \left(\frac{N_1 m_1}{N} + \frac{N_2 m_2}{N}\right)$$

4. Show that the Hessian matrix H for the logistic regression model, given by $H = \phi^T R \phi$, is positive definite. Here R is a diagonal matrix with elements $y_n(1 - y_n)$, and y_n is the output of

the logistic regression model for input vector x_n. Hence show that the error function is a concave function of w and that it has a unique minimum.

$$H = \Phi^T R \Phi$$

$$H = \begin{bmatrix} \phi_{11} & \dots & \phi_{n1} \\ \dots & \dots & \dots \\ \phi_{1d} & \dots & \phi_{nd} \end{bmatrix} \begin{bmatrix} y_n(1-y_n) & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & y_n(1-y_n) \end{bmatrix} \begin{bmatrix} \phi_{11} & \dots & \phi_{1d} \\ \dots & \dots & \dots \\ \phi_{n1} & \dots & \phi_{nd} \end{bmatrix}$$

$$Since\ y_n\ is\ always\ a\ postive\ value\ between\ 0\ and\ 1, y_n(1-y_n)\ will\ always\ pe\ positive$$

$$H = \begin{bmatrix} \phi_{11} y_n(1-y_n) & \dots & \phi_{n1} y_n(1-y_n) \\ \dots & \dots & \dots \\ \phi_{1d} y_n(1-y_n) & \dots & \phi_{nd} y_n(1-y_n) \end{bmatrix} \begin{bmatrix} \phi_{11} & \dots & \phi_{1d} \\ \dots & \dots & \dots \\ \phi_{n1} & \dots & \phi_{nd} \end{bmatrix}$$

$$H = \begin{bmatrix} \phi_{11} y_n(1-y_n)\phi_{11} + \cdots\ \phi_{n1} y_n(1-y_n)\phi_{n1} & \dots & \phi_{11} y_n(1-y_n)\phi_{1d} + \cdots\ \phi_{n1} y_n(1-y_n)\phi_{n1} \\ \dots & \dots & \dots \\ \phi_{11} y_n(1-y_n)\phi_{11} + \cdots\ \phi_{n1} y_n(1-y_n)\phi_{n1} & \dots & \phi_{11} y_n(1-y_n)\phi_{11} + \cdots\ \phi_{n1} y_n(1-y_n)\phi_{n1} \end{bmatrix}$$

$$Since\ we\ know\ that\ X^T X = X^T I X\ is\ always\ positive\ definite\ and\ we\ have\ \phi^T R \phi$$

$$where\ R\ is\ a\ diagonal\ matrix\ where\ all\ diagonal\ entries\ are\ between\ 0\ and\ 1, we\ know$$

$$R\ will\ act\ similar\ to\ I\ in\ that\ it\ won't\ change\ the\ positive\ definite\ nature\ of\ \phi^T \phi$$

Linear Regression: Experiment

This program was implemented in Python 3.8, using numpy to handle linear algrebra operations, matplotlib.pyplot to handle plotting, scipy.io to open and read from the diabetes.mat file, and math to apply natural logarithms to lambda.

The program starts by extracting data from the diabetes.mat file. It then estimates to optimal weight vector by applying ridge regression. However, to apply ridge regression, you need a lambda value (regularizization term) to control the effect of the regularization term. If lambda is too big, you end up underfitting your data; if lambda is too small, you end up overfitting your data. Varying lambdas were tested to estimate the weight vector, from .00001, .0001, … 1, 10. With these weight vectors, the predicted y training and testing values were generated. Finally, they were compared with the ground truth through using the Mean Squared Error formula.

After generating these different MSEs for the training and testing error, 5-fold-cross-validation was performed on the training data matrix. Through this, an optimal lamda value was generated. Once more, the ridge regression was called to estimate the weight vector, this time using the optimal lamda. These results were added to previously created lists of generated MSEs for the training and testing datasets. Finally, a plot was created to show the MSE varying with the log of lamba. The code can be seen below and is also attached to the submission.

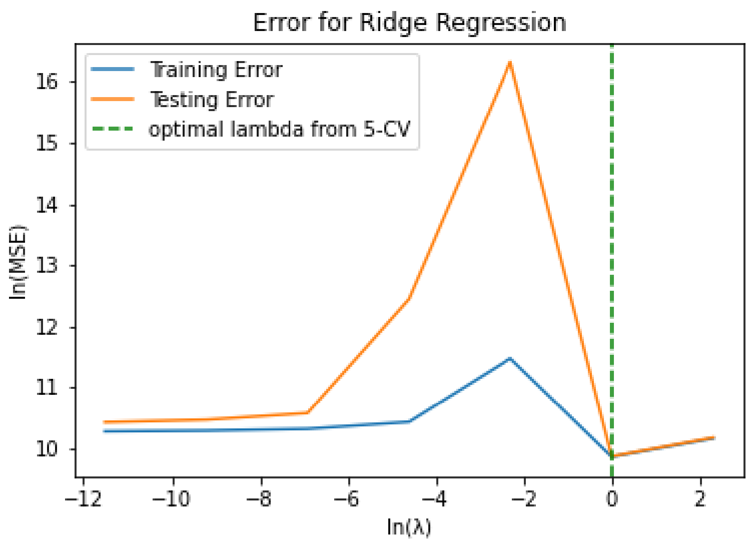

Fig. [1]. A chart showing how to natural log of the mean squared error varies with the natural log of different lambdas, along with the optimal lambda discovered through 5-fold cross validation

Above, a chart can be seen, generated from the program below. This chart shows the varying mean squared error for varying lambdas, again ranging from .00001, .0001, … 1, 10. The natural logarithm was applied to the mean squared error to increase the distinguishability of the error between different lambdas, as the natural log is a monoatomic increasing function. Based on this experiment, it is discovered that a lambda of 1 (ln(lambda) = 0) is the optimal lambda, as based on the cross-validation error from using 5 folds. Using a lambda of 0 in ridge regression for the regularization parameter should yield the most accurate results for this diabetes dataset.

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


def plot_err(x, y_train, y_test, lam_opt):
    plt.ylabel("ln(MSE)")
    plt.xlabel("ln(λ)")
    plt.title("Error for Ridge Regression")
    plt.plot(x, y_train, label="Training Error")
    plt.plot(x, y_test, label="Testing Error")
    plt.axvline(x=lam_opt, color='g', linestyle='--',
                label="optimal lambda from 5-CV")
    plt.legend(loc="upper left")
    plt.savefig("lamda_err.png")
    plt.show()


def MSE(y_pred, y_truth):
    mse = 0
    for i in range(len(y_pred)):
        mse += ((y_truth[i] - y_pred[i]) ** 2)
    return math.log(mse / len(y_pred))


def ridge_regression(lam, X, y):
    # Finds the optimal weights using ridge regression i.e. least squares
    # with a regularization parameter defined by the l2-norm
    xtx = np.matmul(np.transpose(X), X)
    xtx_dim = np.shape(xtx)[0]
    w = (xtx + lam * np.identity(xtx_dim))
    w = np.linalg.inv(w)
    w = np.matmul(w, np.transpose(X))
    w = np.dot(w, y)
    return w


def error(k, pred, truth):
    total = 0
    for i in range(len(pred)):
        total += ((pred[i] - truth[i]) ** 2)
    return total


def cross_val(k, x_train, y, lams):
    fold_len = len(x_train) // k
    # Partition data into k-folds
    D, Y = [], []
    for i in range(0, len(x_train), fold_len):
        D.append(x_train[i: i + fold_len])
        Y.append(y[i: i + fold_len])
    D = D[:-1]  # Got ride of the array of size 2 because bugs
    Y = Y[:-1]

    # Goes through every lambda and every subset
    all_cv = []
    for lam in lams:
        all_fitted = []
        for i in range(len(D)):
            test = D.pop(i)
            y_test = Y.pop(i)
            train = np.array([j for i in D for j in i])
            y_train = np.array([j for i in Y for j in i])
            D.insert(i, test)
            Y.insert(i, y_test)

            # Train the model with subset
            w = ridge_regression(lam, train, y_train)
            y_pred = np.dot(train, w)
            err = error(k, y_pred, y)
            all_fitted.append(err[0])

        # Compute average error for a specific lambda
        cv_err = (1 / k) * sum(all_fitted)
        all_cv.append((cv_err, lam))

    # return minimum cv error's lamda
    return min(all_cv, key=lambda t: t[0])


mat = scipy.io.loadmat('diabetes.mat')

# Get the data
x_train = np.array(mat["x_train"])
x_test = np.array(mat["x_test"])
```

```
y_train = np.array(mat["y_train"])
y_test = np.array(mat["y_test"])

# Test different lambdas to see which one is best ;)
lams = [.00001, .0001, .001, .01, .1, 1, 10]
lams = [math.log(l) for l in lams]
y_train_err, y_test_err = [], []
for lam in lams:
    # Finds training error
    w_train = ridge_regression(lam, x_train, y_train)
    y_pred_train = np.dot(x_train, w_train)
    err_train = MSE(y_pred_train, y_train)
    y_train_err.append(err_train)

    # Finds testing error
    w_test = ridge_regression(lam, x_test, y_test)
    y_pred_test = np.dot(x_test, w_test)
    err_test = MSE(y_pred_test, y_test)
    y_test_err.append(err_test)

# Compute 5-cross fold validation and add the optimal lambda to results
lam = cross_val(5, x_train, y_train, lams)
plot_err(lams, y_train_err, y_test_err, lam[1])
```