CS 4340 - Project 5

Austin Hester

December 05, 2017

Introduction

The goal of this project is to implement linear regression using validation and ridge regularization.

Our training data is:

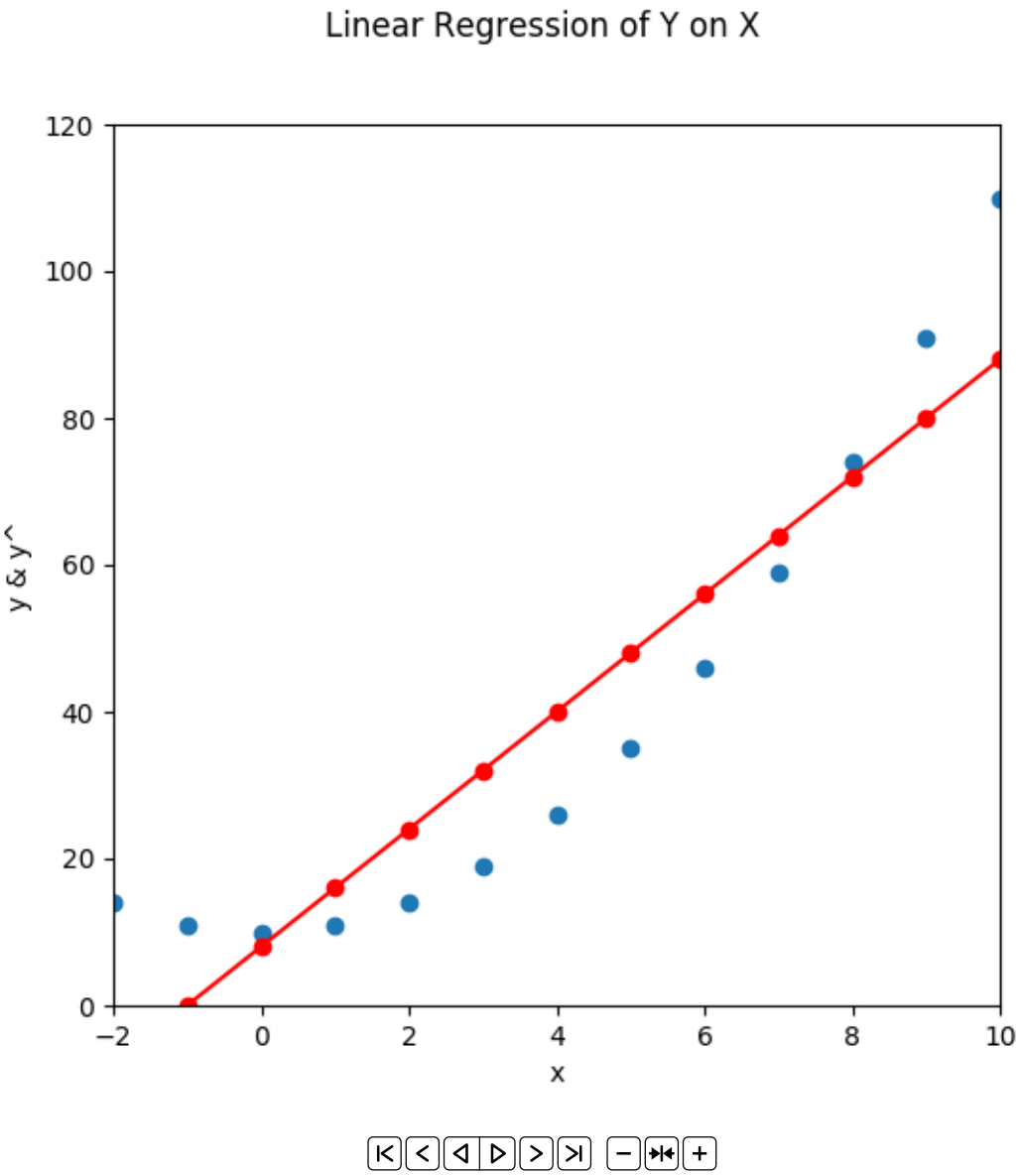
Training Data	
X	**y**
-2	14
-1	11
0	10
1	11
2	14
3	19
4	26
5	35
6	46
7	59
8	74
9	91
10	110

Our y is given by

$$y = f(x) = x^2 + 10$$

.

Animated GIF of 3-Fold Validation



The Code

```
# Austin Hester
# Linear Regression with Regularization
# CS 4340 - Intro to Machine Learning
# 12.04.17

import numpy as np
import matplotlib.pyplot as plt

def make_x ():
    X = [ [ 1, i-2 ] for i in range ( 13 ) ]
    return np.array ( X )

def get_y ( X ):
    y = [ x**2 + 10. for x in X ]
    return np.array ( y )

def get_estimated_y ( X, w_lin ):
    return np.array ( [ round ( yi ) for yi in np.dot ( X, w_lin) ] )

def estimate_single ( N, w_lin ):
    return np.dot ( np.array ( [1, N] ), w_lin )

def print_results ( X, y, y_hat, w_lin ):
    print ( "X =\t", X )
    print ( "Wlin =\t", w_lin )
    print ( "y  =\t", y )
    print ( "y^ =\t", y_hat.T )
    slope, intercept = np.polyfit ( X, y_hat, 1 )
    print ( "Equation of linear regression line: ", "y = ",
            slope, "*x + ", intercept )

def plot ( X, y, y_hat, w_lin , turn=0 ):
    fig = plt.figure ( figsize=(6,6) )
    fig.suptitle ( "Linear Regression of Y on X" )
    # Plot training points
    ax = fig.add_subplot ( 1,1,1 )
    ax.scatter ( X, y, cmap='prism' )
    # Plot line learned from linear regression
    slope, intercept = np.polyfit ( X, y_hat, 1 )
    ablines = [ slope * i + intercept for i in X ]
    cx = fig.add_subplot ( 1,1,1 )
    cx.plot ( X, ablines, 'or-' )
    plt.ylabel ( 'y & y^' )
```

```
    plt.xlabel ( 'x' )
    plt.xlim ( -2, 10 )
    plt.ylim ( 0, 120 )
    #fig.savefig ( "run%d.png" % turn )

def get_MSE ( y, y_hat, w_norm, lam=0.1 ):
    total = 0
    for i in range ( y.size ):
        total += ( y[i] - y_hat[i] )**2
    return total / y.size + ( lam * w_norm )

def get_Eval ( y, y_hat, w_norm, lam=0.1 ):
    Eval = get_MSE ( y, y_hat, w_norm, lam )
    return Eval

def validate ( X, y, turn=0 , lam=1 ):
    if ( turn == 0 ):
        Xtrain = X[:8]
        ytrain = y[:8]
        Xval =   X[8:]
        yval =   y[8:]
    elif ( turn == 1):
        Xtrain = X[:4]  +  X[8:12]
        ytrain = y[:4]  +  y[8:12]
        Xval =   X[4:8]
        yval =   y[4:8]
    else:
        Xtrain = X[4:]
        ytrain = y[4:]
        Xval =   X[:4]
        yval =   y[:4]
    Xt, w_lin = regularize_ridge ( Xtrain, ytrain, lam)
    y_hat = get_estimated_y (X, w_lin)
    w_norm = np.dot ( w_lin.T, w_lin )
    print ( "lambda = ", lam )
    Eval = get_Eval ( y , y_hat , w_norm, lam )
    print ( "Eval\t=", Eval )
    X1d = X.T [1] [:]
    print_results ( X1d, y, y_hat, w_lin )

    return Xt, w_lin, Eval



def regularize_ridge ( X, y, lam=1 ):
    XxXT = np.dot ( X.T, X )              # Compute X.T * X <- A
```

```
    Z = XxXT + ( lam * np.identity ( 2 ) )
    Zinv = np.linalg.inv ( Z )
    Xt = np.dot ( Zinv, X.T )
    w_lin = np.dot ( Xt, y.T )          # Compute weight vector w_lin = C
        * y
    return Xt, w_lin.T

def linear_regression ( X, y ):
    XxXT = np.dot ( X.T, X )            # Compute X.T * X <- A
    XxXT_inv = np.linalg.inv ( XxXT ) # Compute inverse of A <- B
    Xt = np.dot ( XxXT_inv, X.T )  # Psuedo-inverse of X = B * X.T <- C
    w_lin = np.dot ( Xt, y.T )     # Compute weight vector w_lin = C * y
    return Xt, w_lin.T

def run ():
    # Get training points
    X = make_x ()
    X1d = X.T [1] [:]
    y = get_y ( X1d )
    # Run linear regression on those points
    Xt, w_lin = linear_regression ( X, y )
    y_hat = get_estimated_y ( X, w_lin )
    plot ( X1d, y, y_hat , w_lin )
    # Regularize and Validate
    w_norm = np.dot ( w_lin.T, w_lin )
    print ( "Non-regularized Linear Regression" )
    print ( "MSE = ", get_MSE(y, y_hat, w_norm, 0.1 ))
    print_results ( X1d, y, y_hat, w_lin )
    ls = [ 0.1, 1, 10, 100 ]
    Ecvs = []
    for j in ls:
        Es = []
        for i in range(3):
            print ( "-------------------------------------------------------------"
                )
            print ( "Validation", i+1 )
            Xt, w_lin, Eval = validate ( X, y, i, j )
            Es.append ( Eval )
            print( w_lin )
            y_hat = get_estimated_y ( X, w_lin )
            #plot ( X1d, y, y_hat , w_lin, i+1 )
        w_norm = np.dot ( w_lin.T, w_lin )
        avgMSE = (Es[0] + Es[1] + Es[2]) / 3
        Ecvs.append ( avgMSE )
        print ( "-------------------------------------------------------------" )
        print ( "lambda  = ", j )
```

```
        print ( "E_c.v.= ", avgMSE )
        print ( "------------------------------------------------------------" )
    low = 9999999999
    for i, ecv in enumerate(Ecvs):
        print ( "lambda  = ", ls[i] )
        print ( "E_c.v.= ", ecv )
        if ( ecv < low ):
            low = ls[i]
    print ( "------------------------------------------------------------" )
    print ( "We chose lambda = ", low )
    Xt, w_reg = regularize_ridge ( X, y, low )
    y_hat = get_estimated_y ( X, w_reg )
    print_results ( X1d, y, y_hat, w_reg )
    Eval = get_Eval ( y , y_hat , np.dot ( w_reg.T, w_reg ), low )
    print ( "Eval\t=", Eval )
    plot ( X1d, y, y_hat , w_lin, i+1 )


run ()
plt.show ()
```

Notes

a) Our training data.

Training Data	
X	**yl**
-2	14
-1	11
0	10
1	11
2	14
3	19
4	26
5	35
6	46
7	59
8	74
9	91
10	110

b) The equation of the line obtained in part (1) is:

$$y = 8x + 8$$

c) For the following data: turn 1 is the first 8 points, turn 2 is all but the middle 4 points, and turn 3 is the last 8 points. I know there are 13 points total, but 13 is prime and we can ignore 1 here or there.

$\lambda = 0.1$
MSE turn 1 = 746.262661375
MSE turn 2 = 197.356271565
MSE turn 3 = 523.109196851
$E_{c.v.}$ = 488.909376597

$\lambda = 1$
MSE turn 1 = 842.139472172
MSE turn 2 = 328.670923052
MSE turn 3 = 546.035469641
$E_{c.v.}$ = 572.281954955

$\lambda = 10$
MSE turn 1 = 1182.93732293
MSE turn 2 = 1232.34933626
MSE turn 3 = 1058.17326210
$E_{c.v.}$ = 1157.81997376

$\lambda = 100$
MSE turn 1 = 2141.77756418
MSE turn 2 = 5683.71920214
MSE turn 3 = 5620.29574743
$E_{c.v.}$ = 4481.93083792

d) I chose $\lambda = 0.1$ because it gives the lowest MSE (besides zero). Running ridge regularization on X over Y using $\lambda = 0.1$ gives:

e) Our final equation of the line is:

$$y = 8x + 8$$

From $w = [7.88759553, 8.01293266]$

Example Run

Here is example output of a run of 3-fold validation

```
Non-regularized Linear Regression
MSE =  166.8
X =  [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =  [ 8.  8.]
y  =  [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =  [ -8.  -0.   8.  16.  24.  32.  40.  48.  56.  64.  72.  80.  88.]
Equation of linear regression line:  y =  8.0 *x +  8.0
```

```
Validation 1
lambda =  0.1
Eval   = 746.262661375
X =  [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =  [ 12.78259094   3.03841778]
y  =  [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =  [  7.  10.  13.  16.  19.  22.  25.  28.  31.  34.  37.  40.  43.]
Equation of linear regression line:  y =  3.0 *x +  13.0
[ 12.78259094   3.03841778]
```

```
Validation 2
lambda =  0.1
Eval   = 197.356271565
X =  [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =  [ 14.1697503    7.21191107]
y  =  [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =  [ -0.   7.  14.  21.  29.  36.  43.  50.  57.  65.  72.  79.  86.]
Equation of linear regression line:  y =  7.1978021978 *x +  14.2087912088
[ 14.1697503    7.21191107]
```

```
Validation 3
lambda =  0.1
Eval   = 523.109196851
X =  [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =  [-17.93961782  11.8009356 ]
y  =  [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =  [ -42.  -30.  -18.   -6.    6.   17.   29.   41.   53.   65.   76.   88.
  100.]
Equation of linear regression line:  y =  11.8076923077 *x +  -18.0769230769
[-17.93961782  11.8009356 ]
```

```
lambda  =  0.1
E_c.v.=  488.909376597
```

```
Validation 1
lambda =  1
Eval    = 842.139472172
X =  [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =   [ 11.1308642    3.31851852]
y  =     [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =     [  4.   8.  11.  14.  18.  21.  24.  28.  31.  34.  38.  41.  44.]
Equation of linear regression line:  y =  3.32967032967 *x +  10.989010989
[ 11.1308642    3.31851852]
```

```
Validation 2
lambda =  1
Eval    = 328.670923052
X =  [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =   [ 9.82278481  8.30379747]
y  =     [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =     [ -7.   2.  10.  18.  26.  35.  43.  51.  60.  68.  76.  85.  93.]
Equation of linear regression line:  y =  8.31318681319 *x +  9.82417582418
[ 9.82278481  8.30379747]
```

```
Validation 3
lambda =  1
Eval    = 546.035469641
X =  [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =   [-10.67023555  10.75374732]
y  =     [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =     [-32. -21. -11.   0.  11.  22.  32.  43.  54.  65.  75.  86.  97.]
Equation of linear regression line:  y =  10.7417582418 *x +  -10.5824175824
[-10.67023555  10.75374732]
```

```
lambda  =  1
E _ c.v.=  572.281954955
```

```
Validation 1
lambda =  10
Eval    = 1182.93732293
X =  [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =   [ 5.16845878  3.91397849]
y  =     [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =     [ -3.   1.   5.   9.  13.  17.  21.  25.  29.  33.  36.  40.  44.]
Equation of linear regression line:  y =  3.91758241758 *x +  5.0989010989
[ 5.16845878  3.91397849]
```

```
Validation 2
```

```
lambda =  10
Eval     = 1232.34933626
X =  [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =    [ 4.05839416  9.40145985]
y  =      [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =      [-15.  -5.   4.  13.  23.  32.  42.  51.  60.  70.  79.  89.  98.]
Equation of linear regression line:  y =  9.40659340659 *x +  3.98901098901
[ 4.05839416  9.40145985]
```

```
Validation 3
lambda =  10
Eval     = 1058.1732621
X =  [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =    [-1.24726477  9.2166302 ]
y  =      [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =      [-20. -10.  -1.   8.  17.  26.  36.  45.  54.  63.  72.  82.  91.]
Equation of linear regression line:  y =  9.20879120879 *x +  -1.21978021978
[-1.24726477  9.2166302 ]
```

```
lambda  =  10
E _ c.v.=  1157.81997376
```

```
Validation 1
lambda =  100
Eval     = 2141.77756418
X =  [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =    [ 1.07189542  2.01960784]
y  =      [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =      [ -3.  -1.   1.   3.   5.   7.   9.  11.  13.  15.  17.  19.  21.]
Equation of linear regression line:  y =  2.0 *x +  1.0
[ 1.07189542  2.01960784]
```

```
Validation 2
lambda =  100
Eval     = 5683.71920214
X =  [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =    [ 2.02863962  7.08353222]
y  =      [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =      [-12.  -5.   2.   9.  16.  23.  30.  37.  45.  52.  59.  66.  73.]
Equation of linear regression line:  y =  7.10989010989 *x +  1.94505494505
[ 2.02863962  7.08353222]
```

```
Validation 3
lambda =  100
Eval     = 5620.29574743
```

```
X = [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =  [ 0.74157303  7.28089888]
y =  [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =  [-14.  -7.   1.   8.  15.  23.  30.  37.  44.  52.  59.  66.  74.]
Equation of linear regression line:  y =  7.3021978022 *x +  0.637362637363
[ 0.74157303  7.28089888]
```

```
lambda  =  100
E_c.v.=  4481.93083792
```

```
lambda  =  0.1
E_c.v.=  488.909376597
lambda  =  1
E_c.v.=  572.281954955
lambda  =  10
E_c.v.=  1157.81997376
lambda  =  100
E_c.v.=  4481.93083792
```

```
We chose lambda =  0.1
X = [-2 -1  0  1  2  3  4  5  6  7  8  9 10]
Wlin =  [ 7.88759553  8.01293266]
y =  [  14.   11.   10.   11.   14.   19.   26.   35.   46.   59.   74.   91.
  110.]
y^ =  [ -8.  -0.   8.  16.  24.  32.  40.  48.  56.  64.  72.  80.  88.]
Equation of linear regression line:  y =  8.0 *x +  8.0
Eval  = 166.642125313
```

---

p.s. Both graphs looked the same. They differ very minutely. MSE of original differs from our last one by only 0.2. The ridge regression actually beats standard linear regression on MSE by a whopping 0.2.