Lihan Yao ML, Spring 2017
Homework 2: Lasso

1 Introduction

1.1 Dataset Construction

1.1.1

```
X = numpy.random.rand(150,75)
```

1.1.2

```
t = [-1,1]
theta_1 = numpy.random.choice(t,10)*10
theta_2 = numpy.zeros(65)
theta = numpy.concatenate((theta_1,theta_2), axis=0)
```

1.1.3

```
epsilon = 0.1*numpy.random.randn(150)
y = numpy.dot(X,theta)+epsilon
```

1.1.4

```
X_training = X[0:80,:]
y_training = y[0:80]
X_validation = X[80:100,:]
y_validation = y[80:100]
X_testing = X[100:150,:]
y_testing = y[100:150]
```

The above code is taken from the data generating file. Since we were instructed to use the given data, the following answers uses the .txt files provided in the zip. Below is feature normalization based on max and min values of the training set:

```
def feature_normalization(train, valid, test):
    for col in range(train.shape[1]):
        minimum = np.nanmin(train[:,col])
        train[:, col] -= minimum
        test[:, col] -= minimum
        valid[:, col] -= minimum

        max_ = np.nanmax(train[:,col])
        train[:,col] /= max_
        test[:,col] /= max_
        valid[:, col] /= max_
    return train, valid, test
```

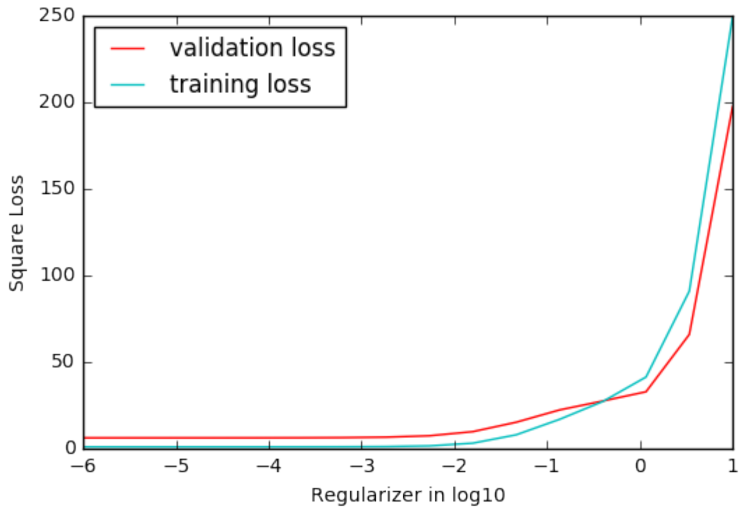
2 Ridge Regression

2.1 Ridge Regression

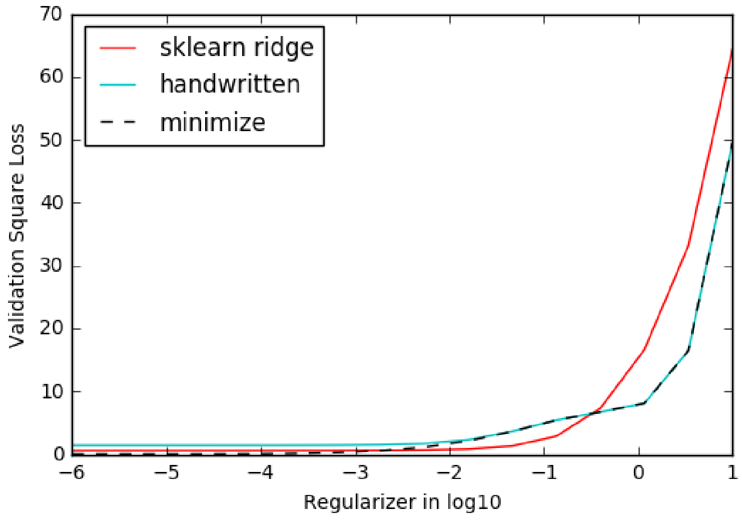
Below is my ridge regression from homework 1.

```
def regularized_grad_descent(
    X, y,  lambda_reg, alpha=5, num_iter=1000, l_search = True):

    (num_instances, num_features) = X.shape
    theta = np.append(np.ones(num_features-1), 1) #Initialize theta w/ bias
    theta_hist = np.zeros((num_iter, num_features))  #Initialize theta_hist
    loss_hist = np.zeros(num_iter) #initialize loss_hist

    for i in range(num_iter):

        loss = r_sq_loss(theta, X, y, lambda_reg)
        loss_hist[i] = loss

        gradient = compute_regularized_square_loss_gradient(theta, X, y, lambda_reg)

        if l_search:
            alpha= backtrack_line_search(loss, gradient, X, y, theta, alpha)
            a_hist.append(round(alpha,4))

        theta -= alpha*gradient
        theta_hist[i,:] = theta

    return loss_hist, theta_hist
```

$\lambda = 10e - 5$ minimizes validation set's square loss.

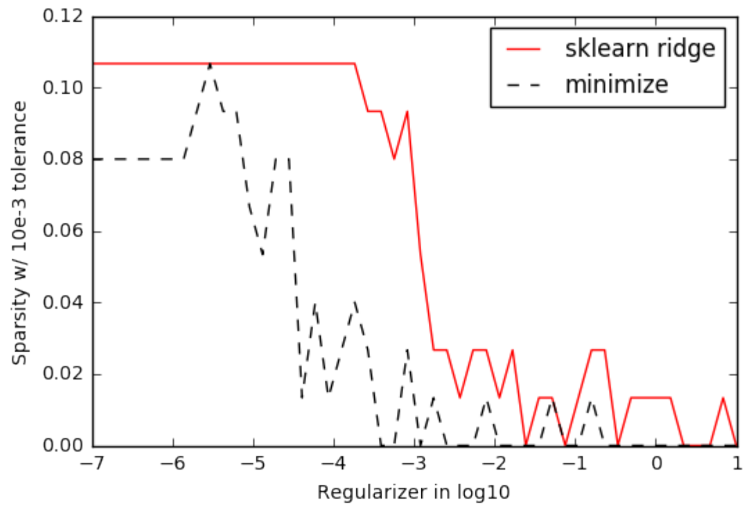


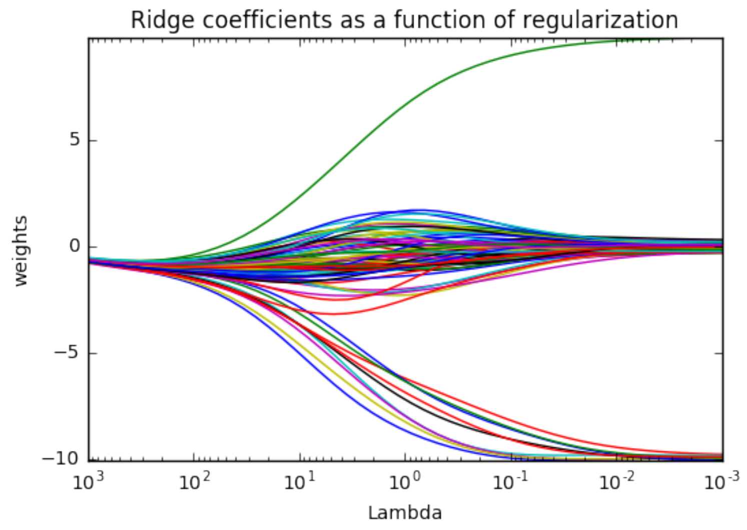
I have decided to go with this value of λ and the above implementation from HW1 to study ridge regression sparsity, though my validation loss are comparable with the other two package implementations (as some rudimentary form of model selection).



For $\lambda = 10e-5$, vector θ_{10e-5} contains no components that are strictly 0. We know the true weight vector θ to contain non-zero values only for the first ten components, so the remainder 65 true value 0 components have been estimated to be non-zero (86 % of the components) . At a tolerance of $10e-3$, 5.2% of θ_{10e-5} components are zero. This indicates 80% of true value 0 components remain wrongly estimated to be non-zero even with this tolerance.

Because of our knowledge about true θ and the added noise, observe that at this regularization strength, ridge regression has not effectively shrunk the low variance principal components in the feature space.





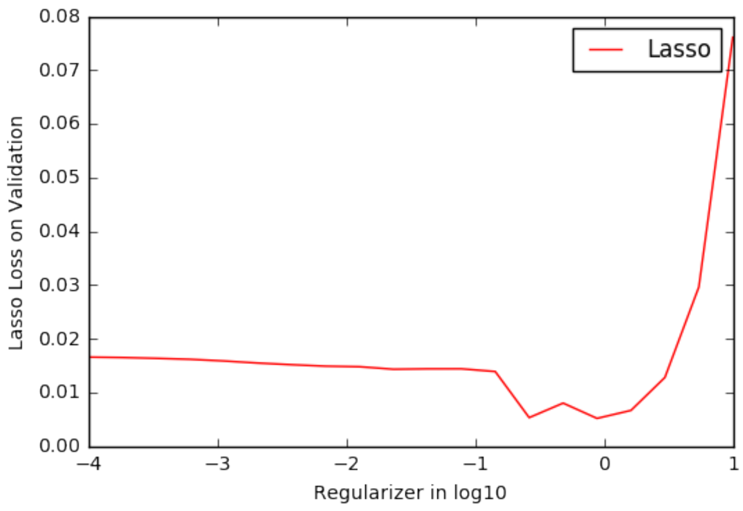
3 Coordinate Descent for Lasso

3.1 Experiments with Shooting Algorithm

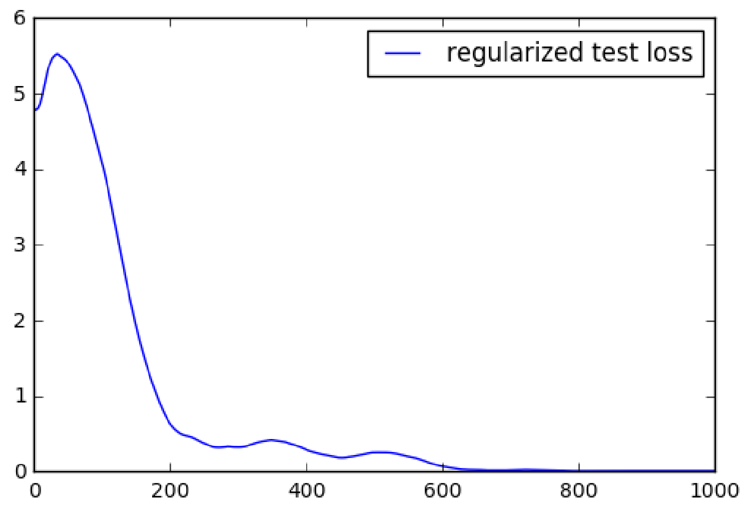
3.1.1

```
def lasso(X, y, w, L, num_iter = 100):
    (num_samp, num_fea) = X.shape

    for run in range(num_iter):
        for j in range(num_fea):
            a_j = 0
            c_j = 0
            for i in range(num_samp):
                a_j += 2*(X[i,j]**2)
                c_j += 2* X[i,j]*(y[i] - np.dot(w.T, X[i,:]) + w[j]*X[i,j])
            a = c_j/a_j
            delta = L/a_j
            w[j] = np.sign(a)* max((abs(a) - delta), 0)
    return w
```

$\lambda = 1$ minimizes square error on the validation set (plot below).

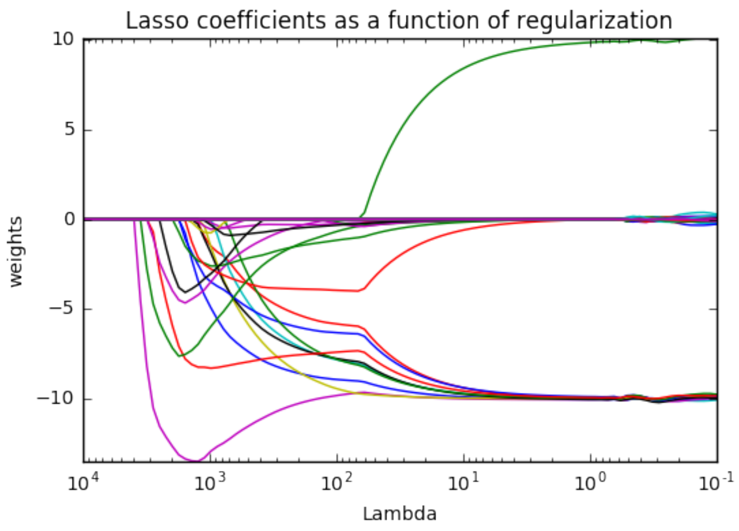


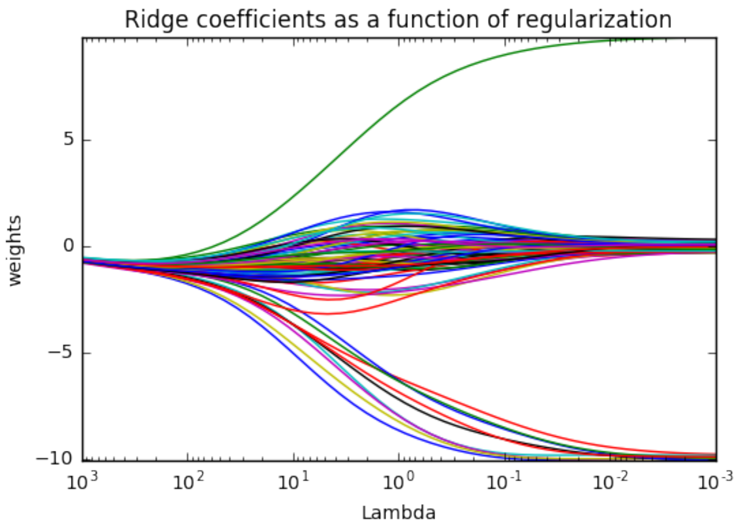
For the same λ the test error (with regularization term) is 0.009 after 1000 iterations.



3.1.2

At $\lambda = 1$, we have 33 true value 0 components that have been estimated correctly (0.51%). All non-zero components in the true theta have been estimated correctly. The two plots show that, while at a regularizer strength of 10, ridge regression has almost always wrongly estimated true value 0 components, while lasso regression has shrunk them. At $\lambda = 1$ where validation error is lowest, almost all feature components have been estimated correctly.





3.1.3

$\lambda_{max} = 3823$.

```
def lasso_homotopy(X, y, L):
    (num_samp, num_fea) = X.shape
    # ridge regression to init
    w = np.zeros(X.shape[1])

    while L >= 0:
        L -= 0.01
        w = lasso(X, y, w, L, num_iter = 1)

    return w
```

I have the regularization path for question 1 as 'np.logspace(-4, 1, num=20).' The homotopy method took 139 seconds. While at 1000 iterations per function call, basic lasso took 1603 seconds.

Given that basic lasso is dictated by number of iterations and the homotopy method by initial lambda and lambda decrement constant, these are the run times for 100 iterations, with λ step-sizes of 0.01 (λ_i is decremented that much after computing the current θ_i):

λ	Basic Lasso	Homotopy
0	8.1	0.012
0.1	9.8	0.084
1	8.9	0.65
10	7.63	6.35
100	7.56	65.83

We see that the basic lasso has performed at similar speed, while the homotopy method has increased dramatically with size of λ.

3.1.4

Let X_j denote the jth column vector of X.
$a_j = 2||X_j||_2^2$
$c_j = 2\dot{X}_j(y - Xw + w_jX_j)$

```
def vec_lasso(X, y, w, L, num_iter = 100):
    (num_samp, num_fea) = X.shape

    for run in range(num_iter):
        for j in range(num_fea):
            a_j = 2*np.dot(X[:,j],X[:,j])
            vec = y- np.matmul(w.T, X.T) + (w*X)[:,j]
            c_j = 2*np.dot(X[:,j], vec)
            a = c_j/a_j
            delta = L/a_j
            w[j] = np.sign(a)* max((abs(a) - delta), 0)
    return w
```

With the same regularization path as before, at 1000 iterations, vectorized lasso took 126.8 seconds, which is vastly faster than basic lasso for the same number of iterations (1603 seconds).

By listing the elapsed time (in seconds) between the basic and vectorized version, we see that the vectorized version is more than 10 times as fast.

Iterations	Basic Lasso	Vectorized Lasso
1	0.087	0.007
100	7.56	0.622
1000	75.4	6.27
5000	378.3	35.46

3.2 Deriving the Coordinate Minimizer for Lasso

3.2.1

The expression to minimize is

$$f(w_j) = \sum_{i=1}^{n} \left[w_j 0 + \sum_{k \neq j} w_k x_{ik} - y_i \right]^2 + \lambda |w_j| + \lambda \sum_{k \neq j} |w_k|$$

By setting $w_j = 0$, the second term is eliminated, so 0 is the coordinate minimizer.

3.2.2

Since $w_j \neq 0$ and f is differentiable everywhere except at $w_j = 0$:

$$\begin{aligned} \frac{\partial f(w_j)}{\partial w_j} &= 2 \sum_{i=1}^{n} \left[w_j x_{ij} + \sum_{k \neq j} w_k x_{ik} - y_i \right] x_{ij} + \lambda \text{sgn}(w_j) \\ &= a_j w_j + 2 \sum_{i=1}^{n} \left[\sum_{k \neq j} w_k x_{ik} - y_i \right] x_{ij} + \lambda \text{sgn}(w_j) \\ &= a_j w_j - c_j + \lambda \text{sgn}(w_j) \end{aligned}$$

3.2.3

Since f is strictly convex in w_j it has an unique minimum. Set $\frac{\partial f(w_j)}{\partial w_j} = 0$ to find its unique minimum:

$$\begin{aligned} a_j w_j - c_j + \lambda \text{sgn}(w_j) = 0 \\ w_j = \frac{1}{a_j}(c_j - \lambda \text{sgn}(w_j)) \end{aligned}$$

Now for the case $w_j > 0$, we know its sign is one, and its sign is negative one if $w_j < 0$. By plugging these into the expression above we have shown both claims.

To give conditions on c_j which implies whether w_j is positive or negative, I will make arguments on $\text{sgn}(c_j - \lambda)$. First notice that a_j is strictly positive in the denominator, and has no effect on the sign of the right handside. We have:

$$w_j a_j = c_j - \lambda \text{sgn}(w_j)$$

If $c_j > \lambda$, regardless of the sign of w_j, right and left sides of the equation are positive. So $w_j > 0$. If $c_j < -\lambda$, regardless of $\text{sgn}(w_j)$, $w_j < 0$.

3.2.4

First compute the numerator.

$$f(\epsilon) - f(0) = \sum_i \left[\epsilon x_{ij} + \sum_{k \neq j} w_k x_{ik} - y_i \right]^2 + \lambda |\epsilon| + \lambda \sum_{k \neq j} |w_k| - \sum_i \left[\sum w_k x_{ik} - y_i \right]^2 - \lambda \sum_{k \neq j} |w_k|$$

In the limits below, the large numerators cancel out once we take ϵ to 0:

$$\begin{aligned}
\lim_{\epsilon \to 0} \frac{f(\epsilon) - f(0)}{\epsilon} &= \lim_{\epsilon \to 0} \frac{\sum_i \left[\left(\epsilon x_{ij} + \sum_{k \neq j} w_k x_{ik} - y_i \right)^2 - \left(\sum_{k \neq j} w_k x_{ik} - y_i \right)^2 \right] + \lambda \epsilon}{\epsilon} \\
&= \lambda \geq 0 \\
\lim_{\epsilon \to 0} \frac{f(-\epsilon) - f(0)}{\epsilon} &= \lim_{\epsilon \to 0} \frac{\sum \left[(-\epsilon x_{ij} + \sum w_k x_{ik} - y_i)^2 - (\sum w_k x_{ik} - y_i)^2 \right] + -\lambda \epsilon}{\epsilon} \\
&= -\lambda \geq 0
\end{aligned}$$

If $c_j \in [-\lambda, \lambda]$, the only possibility is if $c_j = 0$. So in order for w_j to be the minimizer in this case, $w_j = 0$.

3.2.5

Like in an earlier sub-question, we noticed that a_j is strictly positive, it has no effect on the sign of the terms it resides in. The expression implemented is:

$$w_j = \operatorname{sgn}\left(\frac{c_j}{a_j} \right) \max\left(\left| \frac{c_j}{a_j} \right| - \frac{\lambda}{a_j}, 0 \right)$$

If $c_j > \lambda$,

$$w_j = \operatorname{sgn}\left(\frac{c_j}{a_j} \right) \max\left(\frac{|c_j| - \lambda}{a_j}, 0 \right) = \frac{c_j - \lambda \operatorname{sgn}(c_j)}{a_j}$$

Where the last equality follows from the fact that the fraction is strictly positive. If $c_j < -\lambda$, the max function returns its first argument and we have

$$w_j = \frac{c_j - \lambda \operatorname{sgn}(c_j)}{a_j} = \frac{c_j + \lambda}{a_j}$$

Lastly, if $c_j \in [-\lambda, \lambda]$, the fraction argument of the max function is negative, and so 0 is chosen and $w_j = 0$.

4 Lasso Properties

4.1 Deriving λ_{max}

4.1.1

$$\begin{aligned}
L'(0;v) &= \lim_{h\to 0} \frac{L(0+hv)-L(0)}{h} \\
&= \lim_{h\to 0} \frac{||hXv-y||_2^2+\lambda||hv||_1-||-y||_2^2}{h} \\
&= \lim_{h\to 0} \frac{(hXv-y)\cdot(hXv-y)-y\cdot y}{h} + \lambda||v||_1
\end{aligned}$$

Now compute the numerator $(hXv-y)\cdot(hXv-y)-y\cdot y$:

$$\begin{aligned}
(hXv-y)\cdot(hXv-y)-y\cdot y &= \sum_i (hX_iv-y_i)^2 - \sum_i y_i^2 \\
&= \sum_i (h^2(X_iv)^2-2h(X_iv)y_i+y_i^2) - \sum_i y_i^2 \\
&= h^2(Xv\cdot Xv) - 2h(Xv)^Ty
\end{aligned}$$

Finally we have

$$\begin{aligned}
L'(0;v) &= \lim_{h\to 0} \frac{h^2(Xv\cdot Xv)-2h(Xv)^Ty}{h} + \lambda||v||_1 \\
&= \lim_{h\to 0} h(Xv\cdot Xv) - 2(Xv)^Ty + \lambda||v||_1 \\
&= -2(Xv)^Ty + \lambda||v||_1
\end{aligned}$$

4.1.2

Setting $-2(Xv)^Ty+\lambda||v||_1$ to be greater than zero, solve for λ:

$$\begin{aligned}
-2(Xv)^Ty+\lambda||v||_1 &\geq 0 \\
\lambda||v||_1 &\geq 2(Xv)^Ty \\
\lambda &\geq \frac{2(Xv)^Ty}{||v||_1}
\end{aligned}$$

4.1.3

$$\frac{2(Xv)^Ty}{||v||_1} = \frac{2v^TX^Ty}{||v||_1} = 2\sum_i \frac{v_i}{||v||_1}(X^Ty)_i$$

We see that certain choices of v are suboptimal in maximizing the right hand side. We would like to orient v in the direction of the ith component where X^Ty has its largest absolute component of size $||X^Ty||_\infty$. So to maximize the expression over all possible v, orient it as a unit vector in that direction. The desired expression of λ_{max} is achieved.

4.1.4

4.2 Feature Correlation

4.2.1

With $x_1 = x_2$, the lasso objective function is

$$L(\theta) = ||x_1(\theta_1 + \theta_2) + X_r\theta_r - y||_2^2 + \lambda|\theta_1| + \lambda|\theta_2| + \lambda||\theta_r||_1$$

First I show a and b must have the same sign. WLOG suppose $a < 0$ and $b > 0$. For sake of contradiction decrease their difference by $2\epsilon > 0$. Set new variables $a' = a + \epsilon$ and $b' = b - \epsilon$. In the objective function this is

$$\begin{aligned} L(a', b', r) &= ||x_1(a + \epsilon + b - \epsilon) + X_r r - y||_2^2 + \lambda|a + \epsilon| + \lambda|b - \epsilon| + \lambda||r||_1 \\ &= ||x_1(a + b) + X_r r - y||_2^2 + \lambda|a| - \lambda|\epsilon| + \lambda|b| - \lambda|\epsilon| + \lambda||r||_1 \\ &= L(a, b, r) - 2\lambda\epsilon \end{aligned}$$

Where in the second equality, $-\lambda\epsilon$ is a result of $a < 0$. This contradicts $L(a, b, r)$'s properties as a minimizer. We have shown a and b to have the same sign, or one is zero.

Now to perform as well as $\hat{\theta}$, that is, for c and d to incur the same loss, $a + b \geq c + d$, where only equality is possible due to $\hat{\theta}$ being a minimizer. Moreover, c and d must share the same sign.

4.2.2

The ridge objective function is

$$L(\hat{\theta}) = ||x_1(a + b) + X_r r - y||_2^2 + \lambda(a^2 + b^2 + ||r||_2^2)$$

I claim that a and b must be equal. To see this, suppose they differ by a positive distance 2ϵ, with $a + \epsilon = b - \epsilon$.

$$L(\hat{\theta}) = ||x_1(a-\epsilon+b+\epsilon)+X_r r-y||_2^2+\lambda((a-\epsilon)^2+(b+\epsilon)^2+||r||_2^2) = ||x_1(a+b)+X_r r-y||_2^2+\lambda(a^2+b^2-2a\epsilon+2b\epsilon+2\epsilon^2+||r||_2^2)$$

In the second term, $a\epsilon < b\epsilon$. The right hand side increases as ϵ increases implies that a distance of zero between a and b minimizes the objective function.

4.2.3

5 Ellipsoids in the l_1/l_2 regularization picture

5.0.4

5.0.5

5.0.6

5.0.7

6 Projected SGD via Variable Splitting

6.0.8

6.0.9

6.0.10

6.0.11