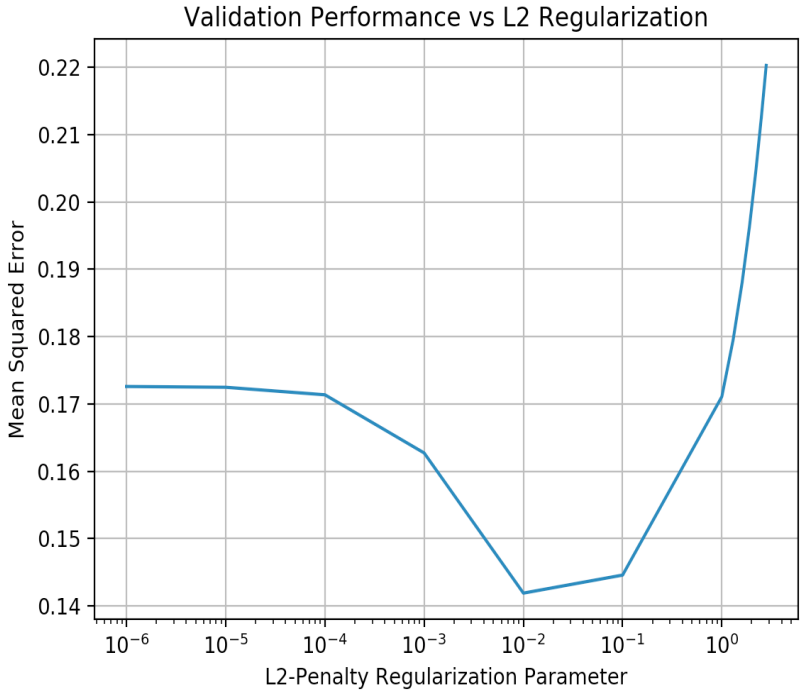
1 Ridge Regression(2.1)

Upon running the ridge regression on the provided data set, we get the best performance(low cost) on $\lambda = 0.01$ as can be seen both in the table below and in the graph from various values of λ.

	param_l2reg	mean_test_score	mean_train_score
0	0.000001	0.172579	0.006752
1	0.000010	0.172464	0.006752
2	0.000100	0.171345	0.006774
3	0.001000	0.162705	0.008285
4	0.010000	0.141887	0.032767
5	0.100000	0.144566	0.094953
6	1.000000	0.171068	0.197694
7	1.300000	0.179521	0.216591
8	1.600000	0.187993	0.233450
9	1.900000	0.196361	0.248803
10	2.200000	0.204553	0.262958
11	2.500000	0.212530	0.276116
12	2.800000	0.220271	0.288422

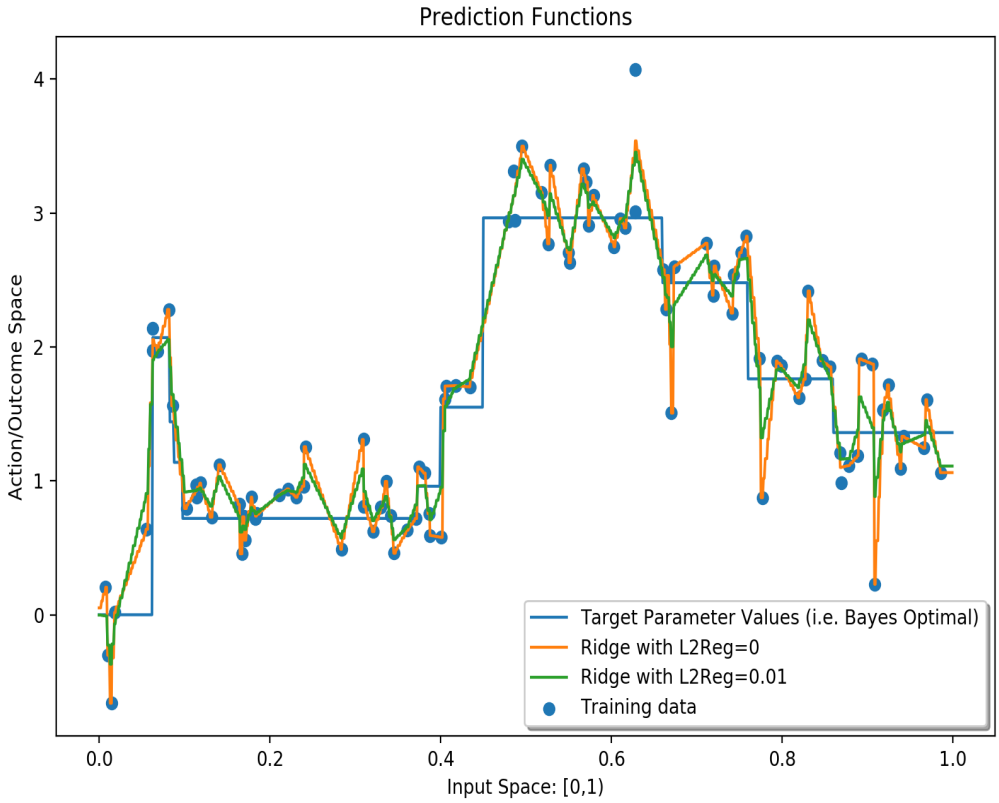
$\lambda - table$



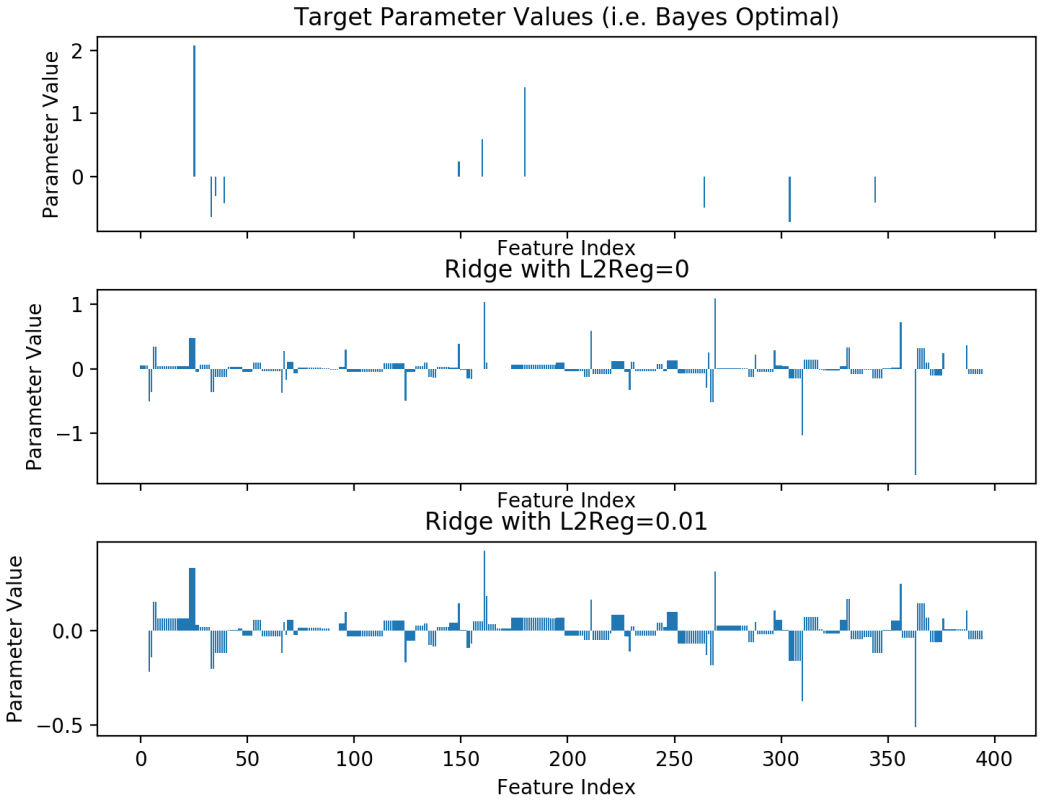
$\lambda - figure$

2 Ridge Regression(2.2)

The following figure plots the result from ridge regression, the target function, unregularized ridge regression, and the scatter of the data points we had.



$prediction - table$


Target Parameter Values (i.e. Bayes Optimal)
Parameter Value
Feature Index
Ridge with L2Reg=0
Parameter Value
Feature Index
Ridge with L2Reg=0.01
Parameter Value
Feature Index

$coef - figure$

3 Ridge Regression(2.3)

The confusion matrix follows:

Confusion matrix for the threshold of $10^{-}6$ is:

[5 385]
[0 10]

Confusion matrix for the threshold of $10^{-}3$ is:

[8 382]
[0 10]

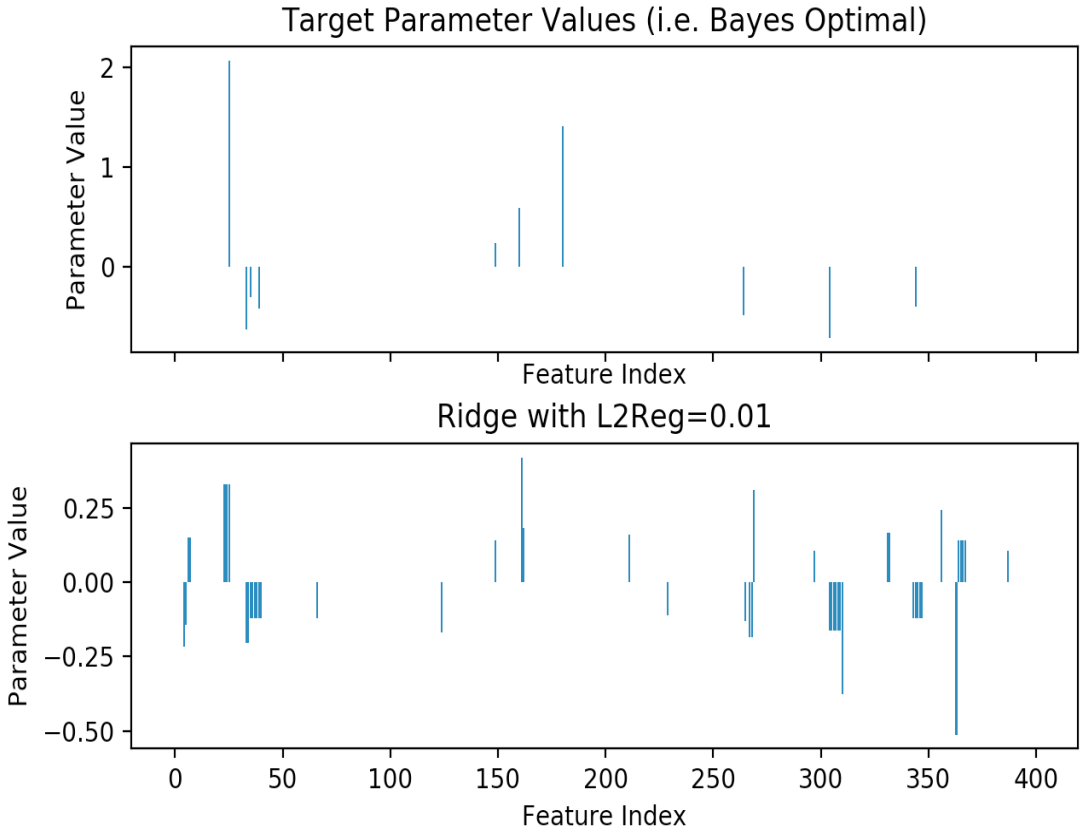
Confusion matrix for the threshold of 0.01 is:

[53 337]
[0 10]

Confusion matrix for the threshold of 0.1 is:

[349 41]
[3 7]

And upon plotting the coefficient with a threshold $\epsilon = 0.1$, we get the following figure:



$trimmed - coefs$

4 Shooting Algorithm(3.1.1)

In the matrix notation, the terms from the algorithm look as follows:

$$a_j = 2\sum_{i=1}^{n} x_{ij}^2$$

$$\Rightarrow a_j = 2X_{.j}^T X_{.j}$$

Similarly,

$$c_j = 2\sum_{i=1}^{n} x_{ij}(y_i - w^T x_i + w_j x_{ij})$$

$$\Rightarrow c_j = 2X_{.j}^T(y - Xw + w_j X_{.j})$$

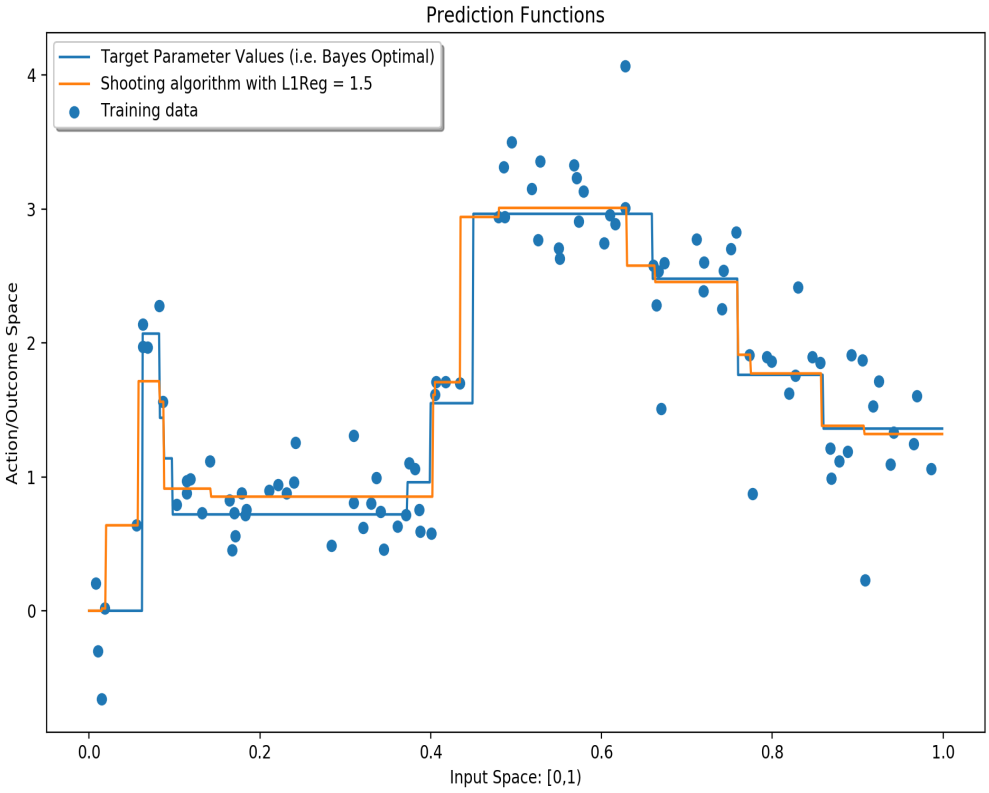
5 Shooting Algorithm(3.1.2)

The code for lshooting algorithm follows:

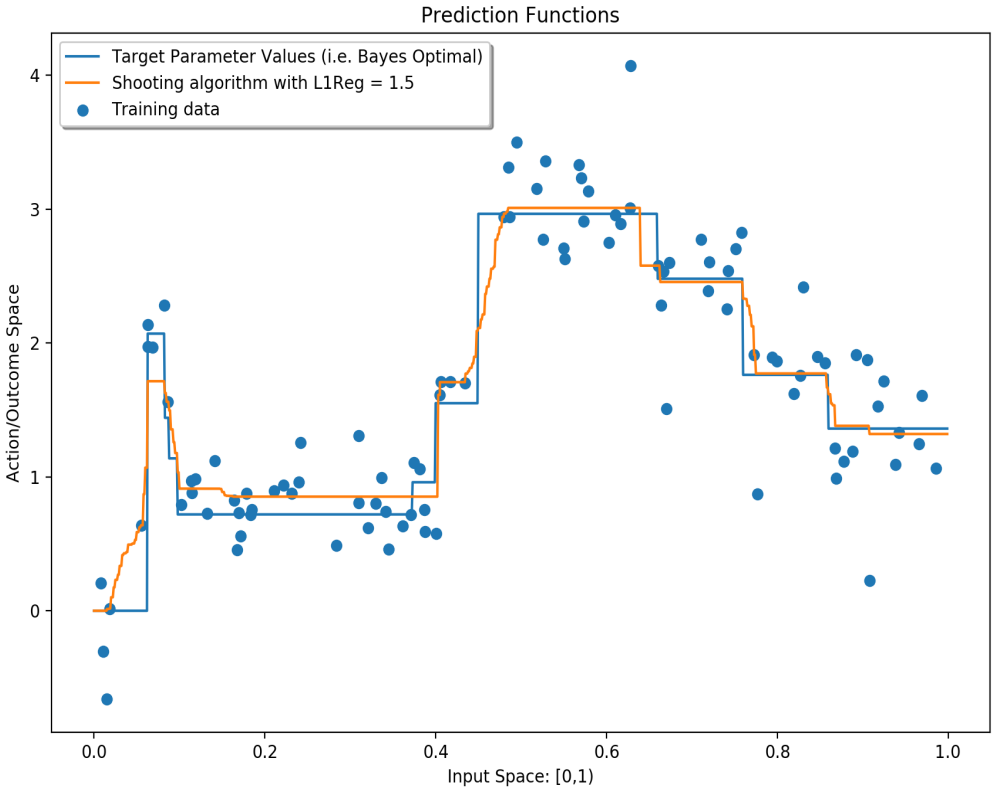
```
### Coordinate Descent:

def soft(a, c, l1reg):
    if not a == 0:
        sgn = np.sign(c/a)
        remain = max(0, abs(c/a) - (l1reg/a))
        final = sgn*remain
    elif a == 0:
        final = 0
    return final

def coordinate_descent(X, y, l1reg):
    # warm starting
    n, num_ftrs = X.shape
    one = np.identity(num_ftrs)
    w0 = ( (X.transpose()).dot(X) + l1reg*one)
    inverse = np.linalg.inv(w0)
    w1 = inverse.dot(X.transpose())
    w = w1.dot(y)

    for i in range(1000):
        for j in range(num_ftrs):
            Xj = X[:,j]
            a = 2*np.dot(Xj, Xj.transpose())
            in_val = y - X.dot(w) + w[j]*Xj
            c = 2*Xj.dot(in_val)
            w[j] = soft(a,c,l1reg)
    return w

def random_coordinate_descent(X, y, l1reg):
    n, num_ftrs = X.shape
    w0 = np.zeros(num_ftrs)
    for i in range(1000):
        rand_lst = np.arange(num_ftrs)
        np.random.shuffle(rand_list)
        for j in rand_lst:
            Xj = X[:,j]
            a = 2*np.dot(Xj, Xj.transpose())
            in_val = y - X.dot(w) + w[j]*Xj
            c = 2*Xj.dot(in_val)
            w[j] = soft(a,c,l1reg)
    return w
```

The figures from the output from shooting algorithm are given below in the order of zero start, random start, and warm start.

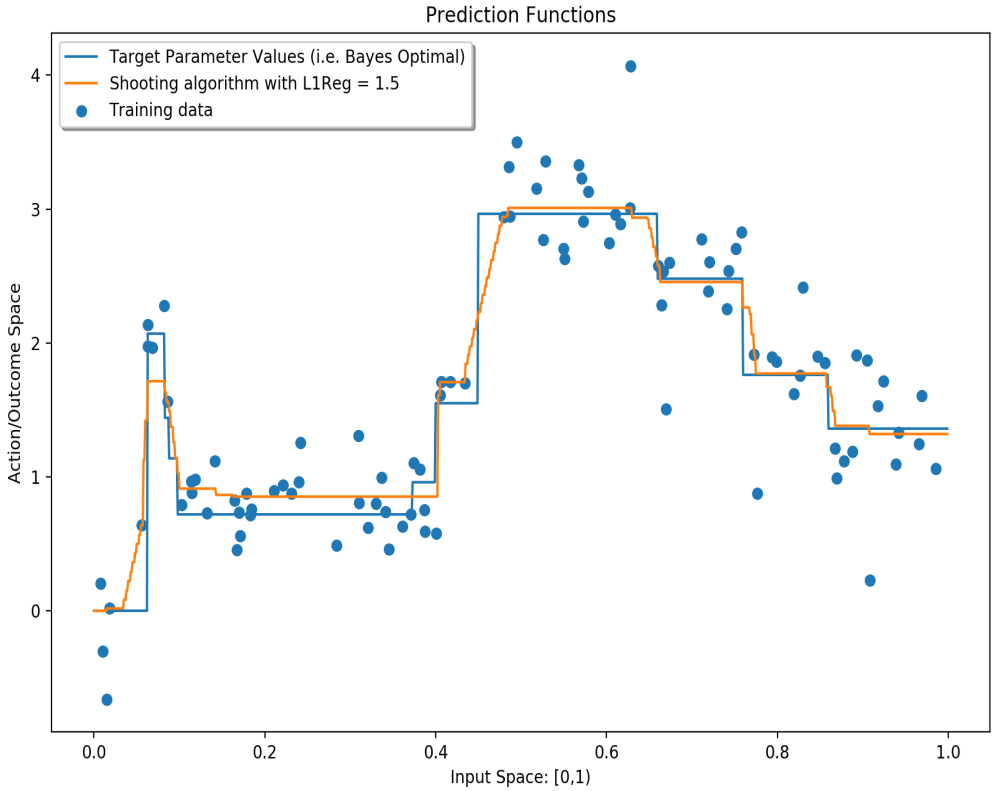
The zero start seem to give a smooth prediction function with a lot of vertical and horizontal lines. Meanwhile, the randomized function return a much less smoother function that isn't always vertical or horizontal. And finally, warm start returns a relatively smooth function that isn't always vertical or horizontal.



$zero - start$



$random - start$


Prediction Functions
Target Parameter Values (i.e. Bayes Optimal)
Shooting algorithm with L1Reg = 1.5
Training data
Action/Outcome Space
Input Space: [0,1)
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0

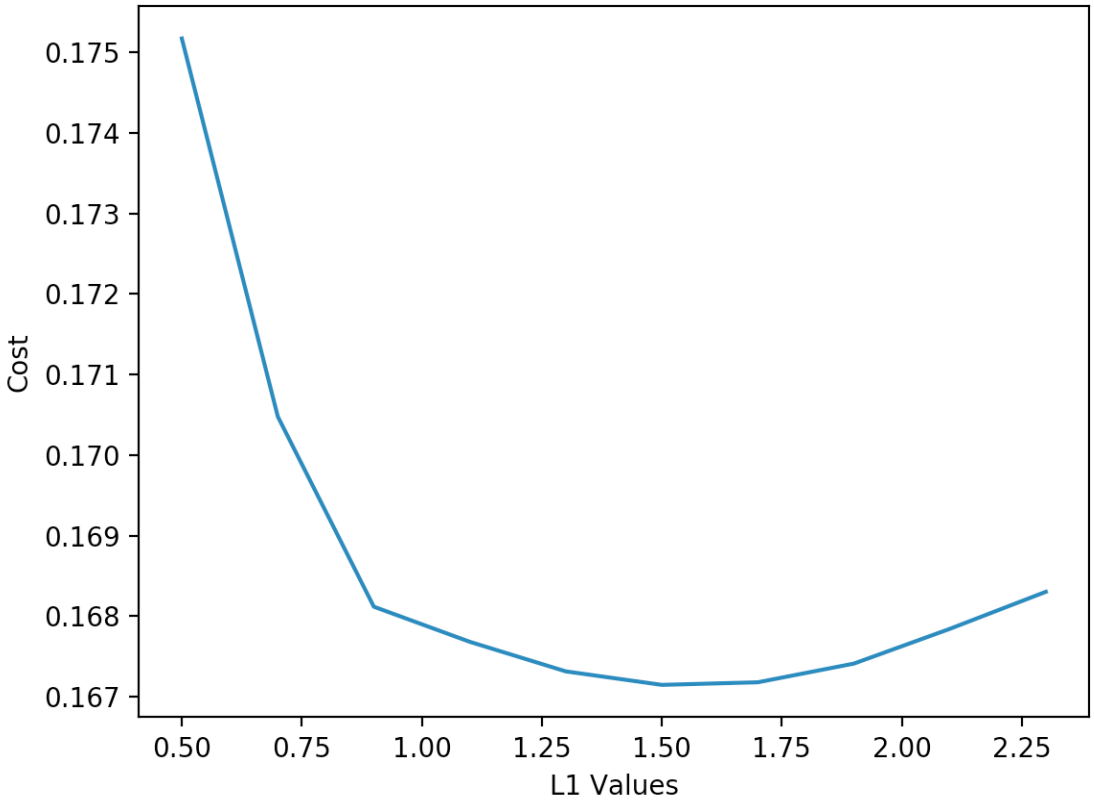
$$warm - start$$

6 Shooting Algorithm(3.1.3)

The table and graph for different values of λ for the lasso follows:

l1_val	cost
0.5	0.1752
0.7	0.1705
0.9	0.1681
1.1	0.1677
1.3	0.1673
1.5	0.1671
1.7	0.1672
1.9	0.1674
2.1	0.1678
2.3	0.1683

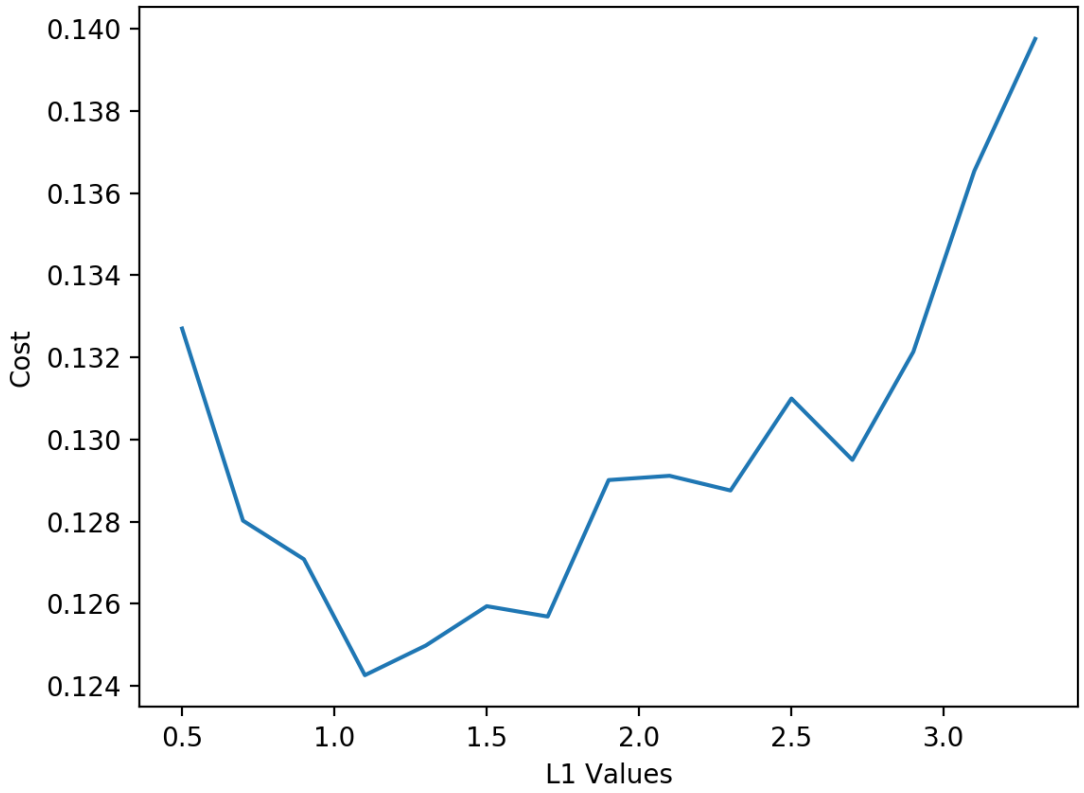
$\lambda - table$



$\lambda - plot$

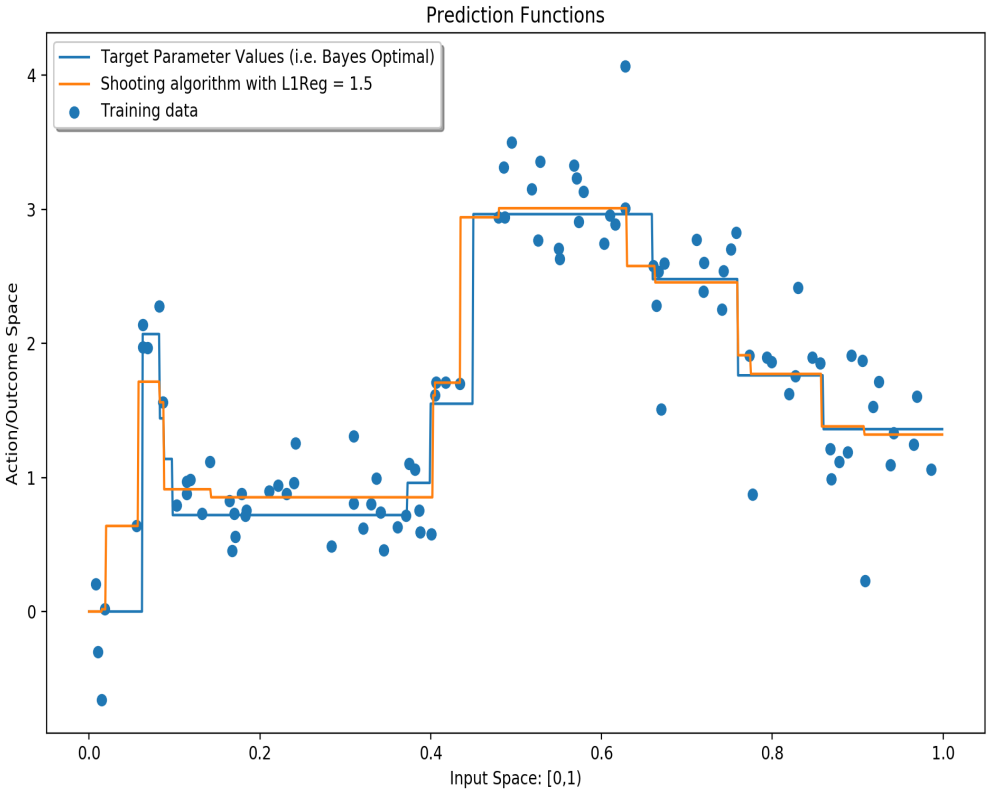
The table and the figure indicates that $\lambda = 1.5$ is the optimum value.

Remark: The randomized shooting algorithm, as expected, randomly performs better than cyclical descent. Following is a *lambda* and cost graph for a randomized descent, which upon multiple run seems to give lowest cost for λ in range $(1, 2)$



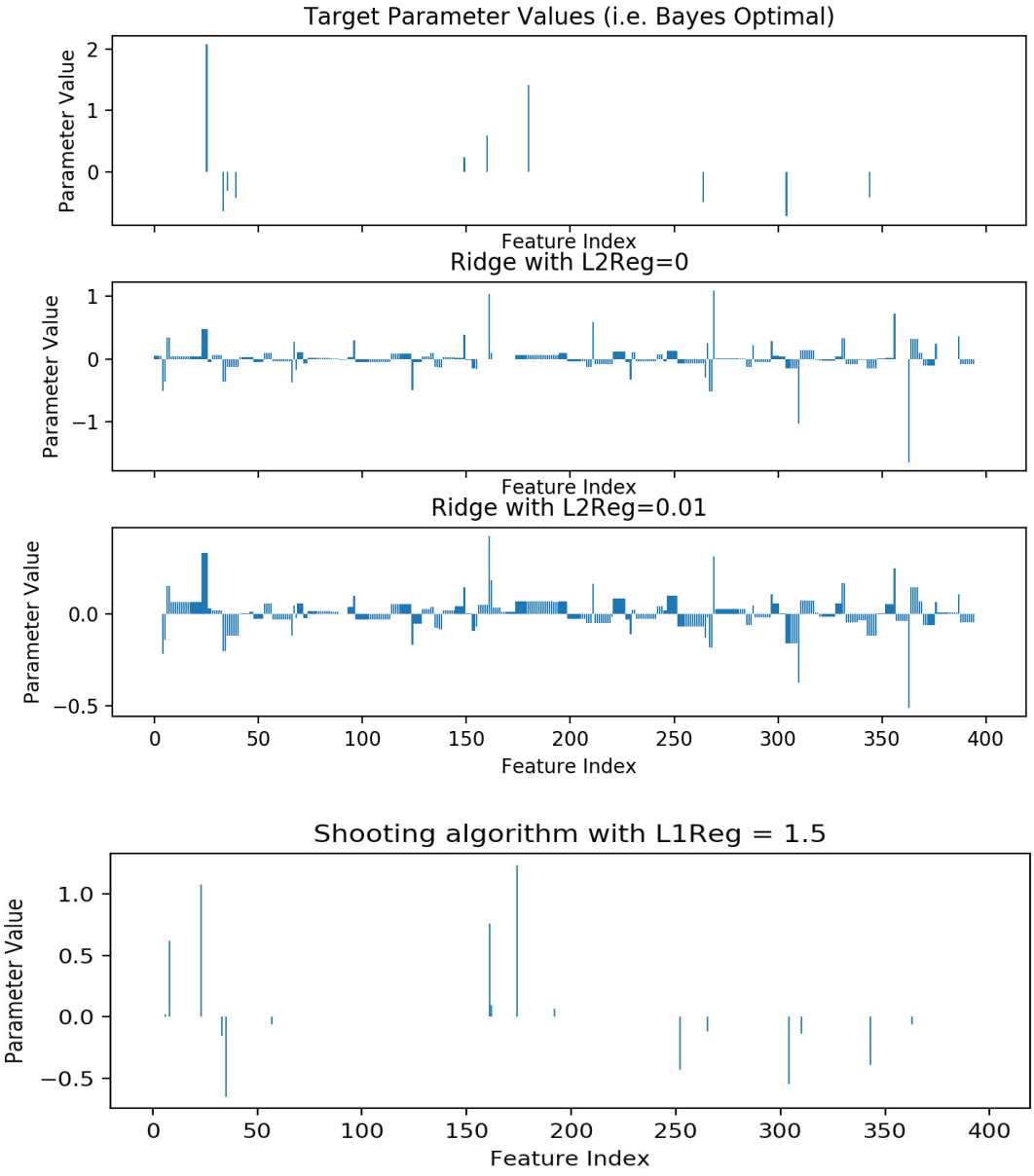
$random - tuning$

The best performing Lasso prediction function with optimized λ follows:



$$zero - start$$

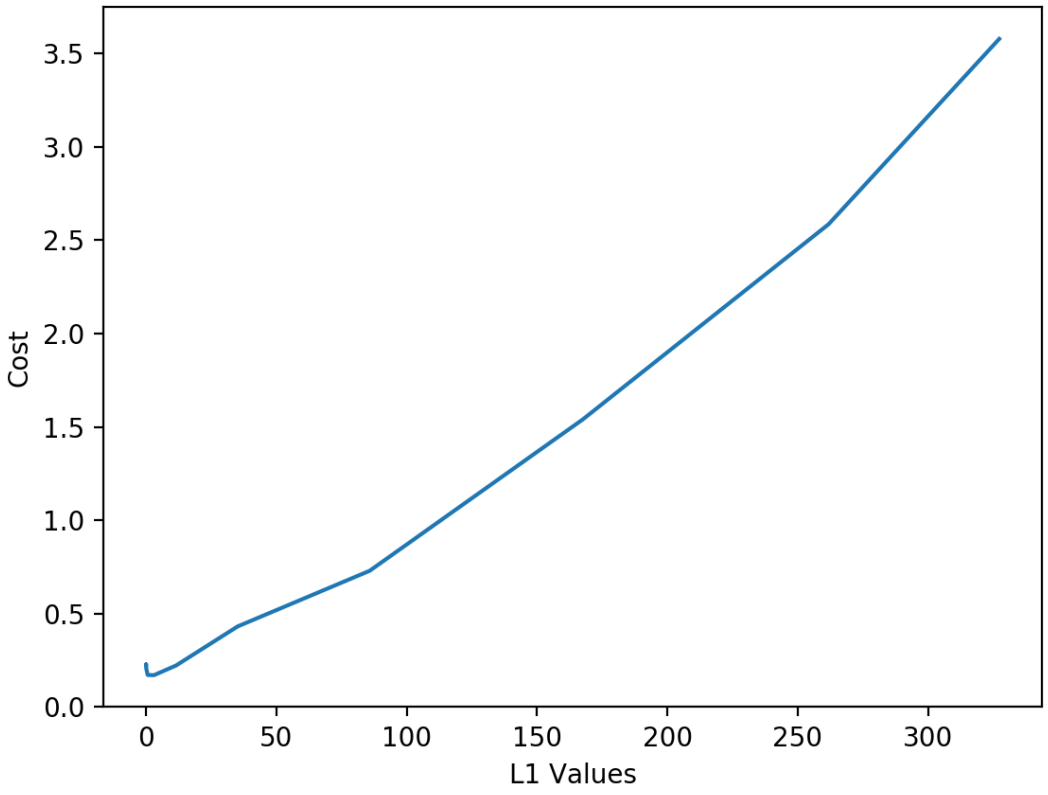
Furthermore you can compare the coefficient values in the following figure:


Target Parameter Values (i.e. Bayes Optimal)
Parameter Value
Feature Index
Ridge with L2Reg=0
Parameter Value
Feature Index
Ridge with L2Reg=0.01
Parameter Value
Feature Index
Shooting algorithm with L1Reg = 1.5
Parameter Value
Feature Index

$coefs - val$

7 Shooting Algorithm(3.1.4)

The validation loss against the decreasing λ value from homotopy is in the graph below:



$\lambda - vals$

As exepected the validation loss decreases as we go down from a large(330) value for λ and reaches minimum at around $\lambda = 1.5$

8 Lasso properties(4.1.1)

To find,

$$f'(x;v) = \lim_{h \downarrow 0} \frac{f(x+hv) - f(x)}{h}$$

Note that, the first part of: $J(w) = \|Xw - y\|_2^2 + \lambda \|w\|_1$ is differentiable and thus has same one-sided derivative as the real derivative.

And we have also calculated the gradient of $\|Xw - y\|_2^2$ previously (in the first HW), which is:

$$\nabla J(w) = 2(X^T X w - X^T y)$$

So, the directional derivative w.r.t 'v' is just the dot productwith 'v':

$$\Rightarrow \nabla J(w).v = 2(X^T X w - X^T y).v$$

$$\Rightarrow \nabla J(0).v = -2(X^T y).v$$

And when $w = 0$ as the question asks,

As for the remaining non-differentiable part,

$$f' = \lim_{h \downarrow 0} \frac{\lambda \|w + hv\|_1 - \lambda \|w\|_1}{h}$$

And, when $w = 0$

$$\Rightarrow f' = \lim_{h \downarrow 0} \frac{\lambda \|hv\|_1}{h}$$

$$\Rightarrow f' = \lambda \|v\|_1$$

Combining the derivative,

$$J'(0;v) = -2(X^T y).v + \lambda \|v\|_1$$

9 Lasso properties(4.1.2)

To find: C s.t $\lambda \geq C$

We have,

$$J'(0; v) \geq 0$$

$$\Rightarrow -2(X^T y).v + \lambda \|v\|_1 \geq 0$$

$$\Rightarrow \lambda \geq \frac{2(X^T y).v}{\|v\|_1}$$

$$\Rightarrow C = \frac{2(X^T y).v}{\|v\|_1}$$

10 Lasso properties(4.1.3)

Consider,

$$\lambda \geq \frac{2(X^T y).v}{\|v\|_1}$$

Here $\frac{v}{\|v\|_1}$ is a unit vector w.r.t l_1 norm. It is easy to see that a vector v that maximizes the lower bound for the λ for $w = 0$ to be a optimum is the one that has zero everywhere but in the position with $Sup(2(X^T y)$. We can argue by contradiction that, if not, the weight 1 of vector v will be distributed to other components smaller than $Sup(2(X^T y)$, which is not the maximum possible value. So, if we consider such a v,

$$Sup[\frac{2(X^T y).v}{\|v\|_1}] = 2Sup(X^T y)$$

, where sup is the l_∞

$$\Rightarrow \lambda_{max} = 2|X^T y|_\infty$$

This can also be seen using the Holder's inequality too, but that would be a bot of an overkill.

So, $\lambda \geq 2\|X^T y\|_\infty$ implies, $w = 0$ is the local minimum since $J'(0; v) \geq 0$ in all direction v.

11 Lasso properties(4.2.1)

Here,

$$\hat{\theta} = \begin{pmatrix} a \\ b \\ r \end{pmatrix}$$

is the optimal solution, then,

$$J(\hat{\theta}) = \left\| x_1 a + x_2 b + X_r \Sigma y \right\|_2^2 + \lambda|a| + \lambda|b| + \lambda \left\| r \right\|_1$$

If a, b are part of the optimum, and they have different sign, we can say $k = a + b$ and since a, b have different signs, $k < |a| + |b|$,

Then,

$$\hat{\theta'} = \begin{pmatrix} k \\ 0 \\ r \end{pmatrix}$$

is a better solution, which we can check by the fact that

$$\lambda|a| + \lambda|b| > \lambda|k|$$

This implies $\hat{\theta'}$ is the optimal. A contradiction!

So, sign of a and b is the same.

Furthermore, it is clear that for $\left(c, d, r^T\right)^T$ to also be a solution, $|a + b| = |c + d|$

12 Lasso properties(4.2.2)

$$\hat{\theta} = \begin{pmatrix} a \\ b \\ r \end{pmatrix}$$

minimizes the ridge regression, meaning it minimizes

$$J(\hat{\theta}) = \left\| x_1 a + x_2 b + X_r \Sigma y \right\|_2^2 + \lambda a^2 + \lambda b^2 + \lambda \left\| r \right\|_2$$

,

As long as $a + b = k$, for some constant k, the first term is unaffected. the under $a + b = k$ we minimize,

$$\lambda a^2 + \lambda b^2$$

$$or, \lambda a^2 + \lambda (k - a)^2$$

Differentiating w.r.t 'a' and setting to zero to find minima,

$$\frac{d}{da}(\lambda a^2 + \lambda (k - a)^2) = 0$$

$$\Rightarrow (\lambda 2a - \lambda 2(k - a)) = 0$$
$$\Rightarrow \lambda (2a - 2k + 2a) = 0$$

Since $\lambda > 0$:

$$\Rightarrow a = \frac{k}{2} = b$$