COMP755-Lect18

October 30, 2018

1 COMP 755

Plan for today

1. Review Coordinate Descent for Ridge and Lasso
2. Fix-point analysis -- exam question practice
3. Full regularization path solution
4. Least Angle Regression solver for Lasso
5. Issues with Lasso

```
In [1]:  def draw_contours(b1s,b2s,objective,n=40):
             B1s, B2s = np.meshgrid(b1s,b2s)
             O = np.zeros((len(b1s),len(b2s)))
             for (i,b1) in enumerate(b1s):
                 for (j,b2) in enumerate(b2s):
                     beta = np.asarray([[b1],[b2]])
                     O[i,j] = objective(beta)
             plt.contour(B1s,B2s,O,n)
```

2 Sparsity in parameters

Optimization of ridge penalized linear regression objective

$$\overbrace{\underbrace{\frac{1}{2}\|\mathbf{y}-\mathbf{Xfi}\|_2^2}_{\text{Negative Log Likelihood}} \quad + \quad \underbrace{\frac{\alpha}{2}\sum_j \beta_j^2}_{\text{ridge penalty}}}^{\text{Ridge Regression}}$$

does not produce sparse **fi**.

We can consider other functions in place of ridge penalty

$$\overbrace{\underbrace{\frac{1}{2}\|\mathbf{y}-\mathbf{Xfi}\|_2^2}_{\text{Negative Log Likelihood}} \quad + \lambda \underbrace{\sum_j |\beta_j|}_{\ell_1 \text{ penalty}}}^{\text{LASSO regression}}.$$

The name of the penalty stems from ℓ_1 norm

$$\|\mathbf{z}\|_1 = \sum_j |z_j|\,.$$

and **LASSO** stands for Least absolute shrinkage and selection operator.

3 Coordinate descent

However, you could also use a simpler approach of updating single β_i at a time
For example,

$$\begin{aligned}
\beta_1^{\text{new}} &= \underset{\beta_1}{\operatorname{argmin}}\, f(\beta_1, \beta_2^{\text{old}}, \beta_3^{\text{old}}) \\
\beta_2^{\text{new}} &= \underset{\beta_2}{\operatorname{argmin}}\, f(\beta_1^{\text{new}}, \beta_2, \beta_3^{\text{old}}) \\
\beta_3^{\text{new}} &= \underset{\beta_3}{\operatorname{argmin}}\, f(\beta_1^{\text{new}}, \beta_2^{\text{new}}, \beta_3)
\end{aligned}$$

and cycling these updates until the changes become small $\sum_j \left|\beta_j^{\text{new}} - \beta_j^{\text{old}}\right| < \epsilon$
At each step, we update a variable to **optimal** value given the rest.

4 Coordinate descent -- derivation procedure

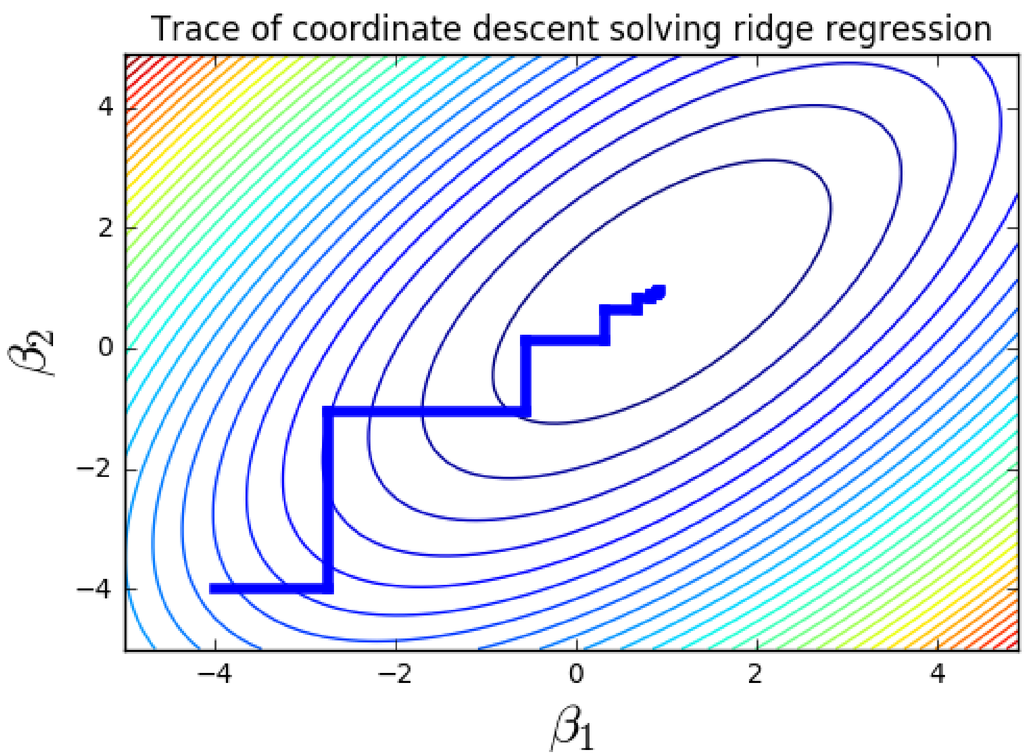
1. Express objective in terms of a single variable (β_k) while keeping rest fixed
2. Compute partial derivative with respect to the variable
3. Equate the partial derivative zero and solve to obtain the update

In [370]:

```
def objective(X,y,beta,alpha):
    return ( 0.5*np.sum((y - np.dot(X,beta)) **2.0)
               + alpha/2.0*np.sum(beta**2.0) )

def update(X,y,beta,alpha,k):
    beta[k] = 0
    y_k = y - np.dot(X,beta) # residual since beta[k]=0
    xk = X[:,[k]]
    beta[k] = np.dot(y_k.T,xk)/(np.dot(xk.T,xk) +  alpha)
    return beta

n = 10
p = 2
np.random.seed(1)
X = np.random.randn(n,p)

y = 1.0*X[:,[0]]+1.0*X[:,[1]] + 0.1*np.random.randn(n,1)
bs = np.arange(-5,5,0.1)
```

```
draw_contours(bs,bs,lambda b: objective(X,y,b,0.1))

beta = np.asarray([[-4.],[-4.]])
for it in range(10):
    for k in range(2):
        beta_old = np.copy(beta)
        beta = update(X,y,beta,0.1,k)
        plt.plot([beta_old[0],beta[0]],
                 [beta_old[1],beta[1]],'b',linewidth=4)

plt.xlabel('$\\beta_1$',fontsize=20)
plt.ylabel('$\\beta_2$',fontsize=20)
plt.title('Trace of coordinate descent solving ridge regression')
```

Out[370]: <matplotlib.text.Text at 0x2f5ff9e8>



5 Coordinate descent for penalized linear regression

Updates for β_k variable for Ridge and Lasso

$$\beta_k^{\mathrm{new}} = \frac{\mathbf{x}_k^T \mathbf{y}^{[-k]}}{\mathbf{x}_k^T \mathbf{x}_k + \alpha} \tag{Ridge}$$

$$\beta_k^{\mathrm{new}} = S\left(\frac{\mathbf{x}_k^T \mathbf{y}^{[-k]}}{\mathbf{x}_k^T \mathbf{x}_k}, \lambda\right) \tag{Lasso}$$

where

$$y_t^{[-k]} = y_t - \sum_{j \neq k} \beta_j x_{tj}$$

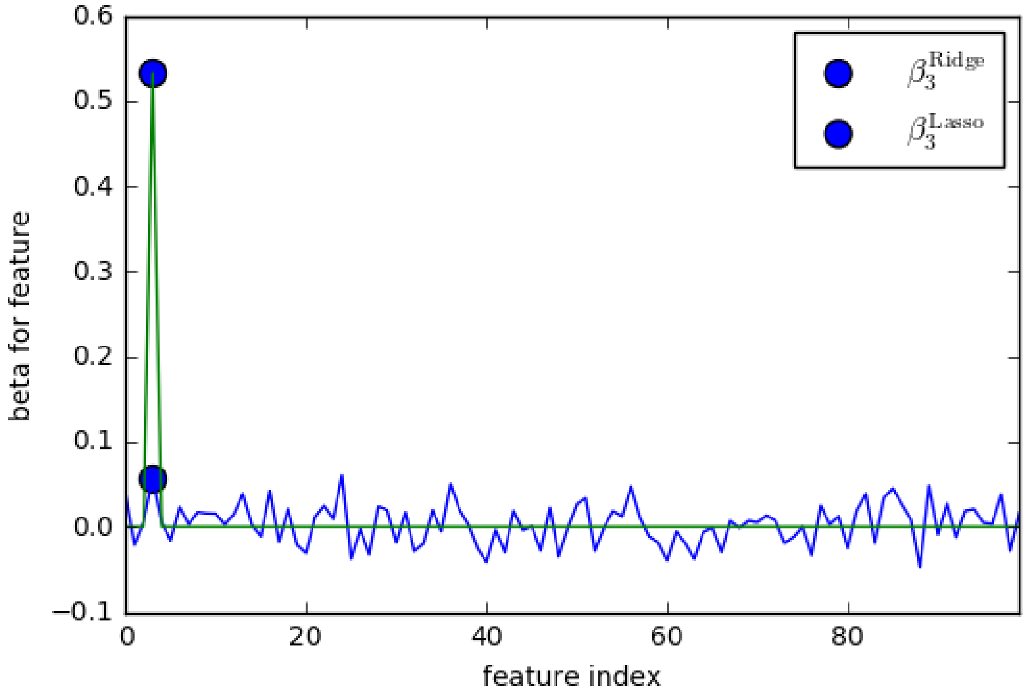
and

$$S(y, \lambda) = \mathrm{sign}(y) \max(|y| - \lambda, 0)$$

In [108]:
```
# a toy example
from sklearn.linear_model import Ridge, Lasso
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
np.random.seed(0)
n = 10
p = 100
X = np.random.randn(n,p)
# use fourth feature
y = 1.0*X[:,[3]] + 0.2*np.random.randn(n,1)
print y.shape


# objective is 1/2*||y - X*beta||^2 + alpha||beta||^2
model = Ridge(alpha=1)
model.fit(X,y)
betas = model.coef_[0]
plt.plot(betas)
plt.scatter(3,betas[3],s=100,label=' $\\beta^{\mathrm{Ridge}}_3$ ')

model2 = Lasso(alpha=0.3)
model2.fit(X,y)
betas2 = model2.coef_
plt.plot(betas2)
plt.scatter(3,betas2[3],s=100,label=' $\\beta^{\mathrm{Lasso}}_3$ ')

plt.xlabel('feature index')
plt.ylabel('beta for feature');
plt.xlim([0,p-1])
plt.legend(scatterpoints = 1)
```

```
(10L, 1L)
```

Out[108]: <matplotlib.legend.Legend at 0x18658da0>



6 Fix-point analysis for iterative algorithms

For smooth objectives $\mathcal{LL}(\mathbf{fi})$ we sought $\mathbf{fi}^*$ such that $\nabla_{\mathbf{fi}}\mathcal{LL}(\mathbf{fi}^*) = 0$.
In linear regression $\nabla_{\mathbf{fi}}\mathcal{LL}(\mathbf{fi}^*) = 0$ becomes a system of linear equations.
There is another, more general way, way to analyze convergence points of algorithms.

7 Fix-point analysis for iterative algorithms

Given an update rule f, for example $\beta^{\mathrm{new}} = f(\beta^{\mathrm{old}})$, an algorithm iterating this update converges when

$$\beta^* = f(\beta^*).$$

A point x for which $f(x) = x$ is called **fix-point** of mapping f.
We will perform fix-point analysis of coordinate descent for ridge for a simple case.

8 Fix-point analysis of coordinate descent for Ridge -- toy example

Assume we are given data Data $= \{(y_t, [x_{t1}, x_{t2}]) \mid t = 1, .., n\}$ where $x_{t1} = \mathbf{x}_{t2}$ -- two features are exactly the same. Further assume that $\mathbf{x}_1$ an d $\mathbf{x}_2$ are normalized (mean is 0.0, and sum of squares is 1.0). The optimization problem of this ridge regression problem is given by

$$\underset{\beta_1,\beta_2}{\text{minimize}} \frac{1}{2} \|\mathbf{y} - \mathbf{x}_1\beta_1 - \mathbf{x}_2\beta_2\|^2 + \frac{\alpha}{2}(\beta_1^2 + \beta_2^2)$$

Coordinate descent mapping is given by:

$$\begin{aligned}
\beta_1^{\text{new}} &= \frac{(\mathbf{y} - \mathbf{x}_2\beta_2^{\text{old}})^T\mathbf{x}_1}{1+\alpha} \\
\beta_2^{\text{new}} &= \frac{(\mathbf{y} - \mathbf{x}_1\beta_1^{\text{new}})^T\mathbf{x}_2}{1+\alpha}
\end{aligned}$$

In [1]:
```
import numpy as np
def normalize(x):
    x = x - np.mean(x)
    x = x/np.linalg.norm(x)
    return x

n = 100
np.random.seed(1)
y = np.random.randn(n)
x1 = np.random.randn(n)
x1 = normalize(x1)
x2 = x1

def update_beta1(y,x1,x2,beta1,beta2,alpha):
    return 1./(1. + alpha)*(np.dot(y - beta1*x1,x2))

def update_beta2(y,x1,x2,beta1,beta2,alpha):
    return 1./(1. + alpha)*(np.dot(y - beta2*x2,x1))
```

9 Fix-point analysis of coordinate descent for Ridge

We will express fixpoints for β_1 and β_2 by dropping *new* and *old* superscripts. Also we get rid off fractions.

$$(1+\alpha)\beta_1 = (\mathbf{y} - \mathbf{x}_2\beta_2)^T\mathbf{x}_1 \tag{1}$$
$$(1+\alpha)\beta_2 = (\mathbf{y} - \mathbf{x}_1\beta_1)^T\mathbf{x}_2 \tag{2}$$
$$\tag{3}$$

10 Fix-point analysis of coordinate descent for Ridge

We simplify fixpoint for β_1 using the fact that $\mathbf{x}_1 = \mathbf{x}_2$ and that $\|\mathbf{x}_1\| = \sqrt{\mathbf{x}_1^T\mathbf{x}_1} = 1.0$

$$
\begin{aligned}
(1+\alpha)\beta_1 &= (\mathbf{y}-\mathbf{x}_2\beta_2)^T\mathbf{x}_1 && (4)\\
(1+\alpha)\beta_1 &= \mathbf{y}^T\mathbf{x}_1 - \beta_2 \overbrace{\mathbf{x}_2^T\mathbf{x}_1}^{\mathbf{x}_1^T\mathbf{x}_1=1} && (5)\\
(1+\alpha)\beta_1 &= \mathbf{y}^T\mathbf{x}_1 - \beta_2 && (1)
\end{aligned}
$$

We simplify fixpoint for β_2 analogously to what we did for β_1

$$
\begin{aligned}
(1+\alpha)\beta_2 &= \mathbf{y}^T\mathbf{x}_2 - \beta_1 && (2)\\
& && (6)
\end{aligned}
$$

11 Fix-point analysis of coordinate descent for Ridge

Express β_2 in the in terms of β_1 using Eq.1

$$\beta_2 = \mathbf{y}^T\mathbf{x}_1 - (1+\alpha)\beta_1 \tag{3}$$

Use Eq.3 to rewrite Eq.2 in terms of β_1 and simplify to obtain closed-form solution for β_1:

$$
\begin{aligned}
(1+\alpha)(\mathbf{y}^T\mathbf{x}_1 - (1+\alpha)\beta_1) &= \mathbf{y}^T\mathbf{x}_1 - \beta_1\\
(1+\alpha)(\mathbf{y}^T\mathbf{x}_1) - (1+\alpha)^2\beta_1 &= \mathbf{y}^T\mathbf{x}_1 - \beta_1\\
(1-(1+\alpha)^2)\beta_1 &= (1-(1+\alpha))\mathbf{y}^T\mathbf{x}_1\\
\beta_1 &= \frac{(1-(1+\alpha))}{(1-(1+\alpha)^2)}\mathbf{y}^T\mathbf{x}_1\\
\beta_1 &= \frac{1}{1+(1+\alpha)}\mathbf{y}^T\mathbf{x}_1\\
\beta_1 &= \frac{\mathbf{y}^T\mathbf{x}_1}{2+\alpha}
\end{aligned}
$$

Analgously solve for β_2

$$\beta_2 = \frac{\mathbf{y}^T\mathbf{x}_2}{2+\alpha}$$

12 Fix-point analysis of coordinate descent for Ridge

Note, that we could get the same solution by equating gradient of

$$\frac{1}{2}\|\mathbf{y}-\mathbf{x}_1\beta_1-\mathbf{x}_2\beta_2\|^2 + \frac{\alpha}{2}(\beta_1^2+\beta_2^2)$$

to zero.

Recap: 1. Write out the coordinate descent updates either as $\mathbf{fi}^{\text{new}} = f(\mathbf{fi}^{\text{old}})$ 2. Drop new and old superscript 3. Solve the resulting system

```
In [ ]: def solve(y,x1,x2,alpha):
            beta1 = np.dot(y.T,x1)/(2.0 + alpha)
            beta2 = np.dot(y.T,x2)/(2.0 + alpha)
            return beta1, beta2

        alpha = 0.5
        beta1,beta2 = solve(y,x1,x2,alpha)
        # are these fix-points?
        assert(np.abs(beta1 - update_beta1(y,x1,x2,beta1,beta2,alpha))<1e-7)
        assert(np.abs(beta2 - update_beta2(y,x1,x2,beta1,beta2,alpha))<1e-7)
```

13 Fix-point analysis of coordinate descent for Lasso -- toy example

Assume we are given data Data $= \{(y_t, [x_{t1}, x_{t2}]) \mid t = 1, .., n\}$ such that $\mathbf{x}_1^T\mathbf{x}_2 = 0, \mathbf{x}_1^T\mathbf{x}_1 = 1, \mathbf{x}_2^T\mathbf{x}_2$. Let $\mathbf{y}^T\mathbf{x}_1 = c_1$ and $\mathbf{y}^T\mathbf{x}_2 = c_2$, and $c_1 > c_2 > 0$.

For optimization problem

$$\underset{\beta_1,\beta_2}{\text{minimize}} \frac{1}{2} \|\mathbf{y} - \mathbf{x}_1\beta_1 - \mathbf{x}_2\beta_2\|^2 + \lambda(|\beta_1| + |\beta_2|),$$

figure out which values of λ lead to solutions 1. $\beta_1 = \beta_2 = 0$ 2. $\beta_1 > 0, \beta_2 = 0$ 3. $\beta_1 > 0, \beta_2 > 0$ 4. $\beta_1 = 0, \beta_2 > 0$

Take a breath.

14 Reading between the lines

Assume we are given data Data $= \{(y_t, [x_{t1}, x_{t2}]) \mid t = 1, .., n\}$ such that $\overbrace{\mathbf{x}_1^T\mathbf{x}_2 = 0, \mathbf{x}_1^T\mathbf{x}_1 = 1, \mathbf{x}_2^T\mathbf{x}_2}^{\text{features are orthonormal}}$. Let $\mathbf{y}^T\mathbf{x}_1 = c_1$ and $\mathbf{y}^T\mathbf{x}_2 = c_2$, and $c_1 > c_2 > 0$. $\mathbf{x}_1^T\mathbf{x}_2 = 0$ means something is going to disappear; $\mathbf{x}_1^T\mathbf{x}_1 = 1$ means denominators might be simpler; $c_1 > c_2 > 0$ some sort of asymmetry between $\mathbf{x}_1$ and $\mathbf{x}_2$

For optimization problem

$$\underset{\beta_1,\beta_2}{\text{minimize}} \frac{1}{2} \|\mathbf{y} - \mathbf{x}_1\beta_1 - \mathbf{x}_2\beta_2\|^2 + \lambda(|\beta_1| + |\beta_2|),$$

looks like lasso; can be solved by coordinate descent using shrinkage and thresholding operator

figure out which values of λ lead to solutions 1. $\beta_1 = \beta_2 = 0$ fix point 2. $\beta_1 > 0, \beta_2 = 0$ β_2 of fix point given 3. $\beta_1 > 0, \beta_2 > 0$ fix point characterized 4. $\beta_1 = 0, \beta_2 > 0$ β_1 of fix point characterized;

solution for 4. probably not the same as 2. because $c_1 > c_2$.

15 Fix point analysis of coordinate descent for Lasso

Update

$$\beta_k^{\text{new}} = S\left(\frac{\mathbf{x}_k^T\mathbf{y}^{[-k]}}{\mathbf{x}_k^T\mathbf{x}_k}, \lambda\right) \tag{Lasso}$$

where

$$y_t^{[-k]} = y_t - \sum_{j \neq k} \beta_j x_{tj} \qquad \text{(residual without } k\text{th predictor)}$$

and

$$S(y, \lambda) = \text{sign}(y) \max(|y| - \lambda, 0). \qquad \text{(shrinkage and thresholding operator)}$$

Using orthonormality we can simplify to:

$$\beta_k^{\text{new}} = S\left(\mathbf{x}_k^T \mathbf{y}, \lambda\right).$$

Will work this out on board.

16 Fix point analysis of coordinate descent for Lasso

Q: In terms of y and λ, when is

$$S(y, \lambda) = \text{sign}(y) \max(|y| - \lambda, 0) = 0?$$

Q: In terms of y and λ, when is

$$S(y, \lambda) = \text{sign}(y) \max(|y| - \lambda, 0) > 0?$$

17 Fix point analysis of coordinate descent for Lasso

Using orthonormality

$$\beta_k^{\text{new}} = S\left(\mathbf{x}_k^T \mathbf{y}, \lambda\right).$$

we know that $\mathbf{y}^T \mathbf{x}_1 = c_1, \mathbf{y}^T \mathbf{x}_2 = c_2, c_1 > c_2 > 0$ so

$$\beta_1^* = S\left(c_1, \lambda\right)$$

and

$$\beta_2^* = S\left(c_2, \lambda\right).$$

Q: In terms of c_1, c_2, and λ when is optimal solution 1. $\beta_1^* = \beta_2^* = 0$? 2. $\beta_1^* > \beta_2^* = 0$? 3. $\beta_1^* > \beta_2^* > 0$? 4. $\beta_1^* = 0, \beta_2^* > 0$?

18 Fix point analysis of Lasso -- toy example 2

For different values of λ in Lasso regression we obtain solutions with different levels of sparsity.

The smallest λ for which optimal solution is all zeros is equal to

$$\lambda^{\max} = \max_i \frac{|\mathbf{y}^T \mathbf{x}_i|}{\mathbf{x}_i^T \mathbf{x}_i}$$

To show this, let $c_i = \frac{|\mathbf{y}^T \mathbf{x}_i|}{\mathbf{x}_i^T \mathbf{x}_i}$, consider starting coordinate descent with $\beta_1 = \cdots = \beta_p = 0$.
Since all βs are zero $\mathbf{y}^{[-l]} = \mathbf{y} - \sum_{i \neq l} \beta_i \mathbf{x}_i = \mathbf{y}$
Consider update for β_k

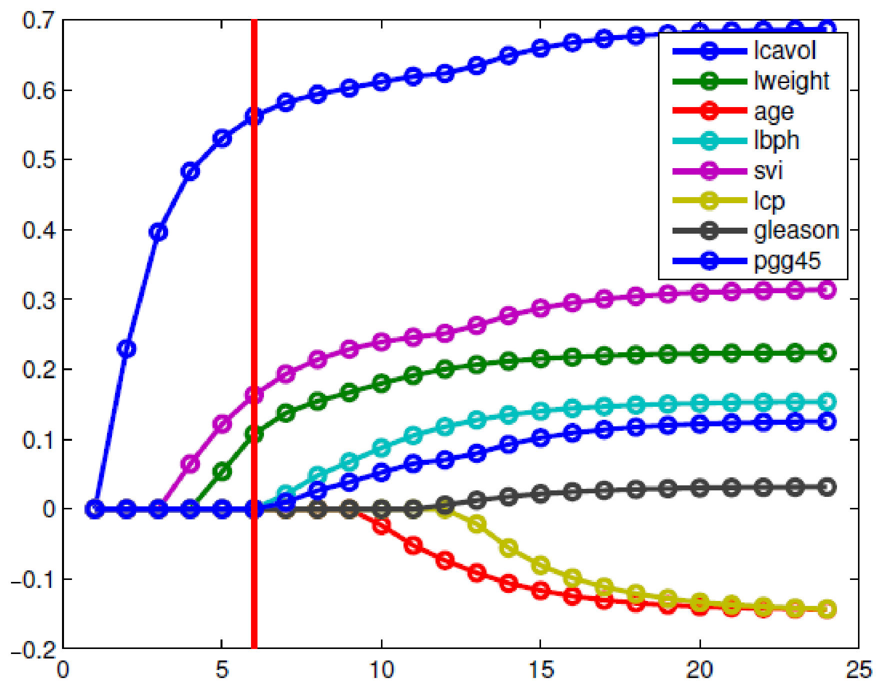


fig 13.7 b)

$$\beta_k^{\text{new}} = S\left(\frac{\mathbf{x}_k^T \mathbf{y}^{[-k]}}{\mathbf{x}_k^T \mathbf{x}_k}, \lambda^{\max}\right) \tag{7}$$

$$\beta_k^{\text{new}} = S\left(\frac{\mathbf{x}_k^T \mathbf{y}}{\mathbf{x}_k^T \mathbf{x}_k}, \lambda^{\max}\right) \qquad \text{(since all betas are 0)}$$

$$\beta_k^{\text{new}} = S\left(c_k, \lambda\right) \qquad \text{(by def. of } c_k\text{)}$$

$$\beta_k^{\text{new}} = \text{sign}(c_k) \max(|c_k| - \lambda^{\max}, 0) \qquad \text{(by def of } S(\cdot,\cdot)\text{)}$$

$$\beta_k^{\text{new}} = \text{sign}(c_k) 0 \qquad \text{(by def of } \lambda^{\max}\text{)}$$

Hence, all updates leave βs at zero.

19 Regularization path for penalized regression

For different values of λ in Lasso regression we obtain solutions with different levels of sparsity.

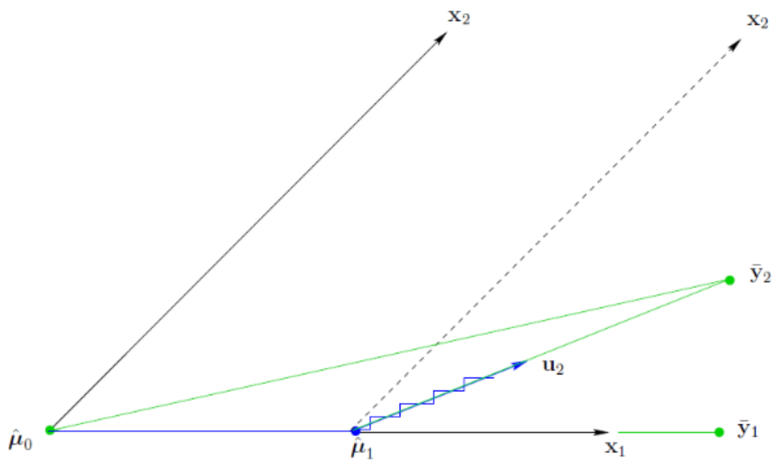
Plot of weights vs. sum of absolute values of weight vector achieved for different λ.

20 Full regularization path

Guessing at the level of sparsity for particular λ is non-trivial.

We would have to construct a list of candidates and fit the lasso model for each of them.

Q: Suppose you know that for $\lambda = 1.0$ number of non-zeros (nnz) βs is 4 and $\lambda = 2.0$ nnz βs is 6. How would you find λ for which nnz βs is 5?



21 Least Angle Regression

It turns out that there is relatively elegant algorithm for obtaining the full regularization path without having to guess at λ schedule.

Assume predictors $\mathbf{x}_k$ are normalized (mean 0, norm 1) and $\mathbf{y}$ is centered (mean 0).

1. Set $\mathbf{r} = \mathbf{y}$
2. $j = \text{argmax}_i \left|\mathbf{x}_i^T \mathbf{y}\right|$
3. Increase β_j in direction of $\mathbf{x}_j^T \mathbf{r}$ and update $\mathbf{r} = \mathbf{y} - \beta_j \mathbf{x}_j$ until

$$\left|\text{corr}(\mathbf{x}_j, \mathbf{r})\right| = \left|\text{corr}(\mathbf{x}_l, \mathbf{r})\right|$$

   for $l \neq j$.
4. Regress $\mathbf{r}$ onto $\mathbf{x}_i, \mathbf{x}_l$ to obtain b_i, b_l
5. Increase β_j and β_k in direction b_i, b_l and update $\mathbf{r} = \mathbf{y} - \beta_j \mathbf{x}_j - \beta_k \mathbf{x}_k$ until

$$\left|\text{corr}(\mathbf{x}_j^T b_i + \mathbf{x}_k b_j, \mathbf{r})\right| = \left|\text{corr}(\mathbf{x}_l^T, \mathbf{r})\right|$$

22 Least Angle Regression

23 Least Angle Regression (LARS)

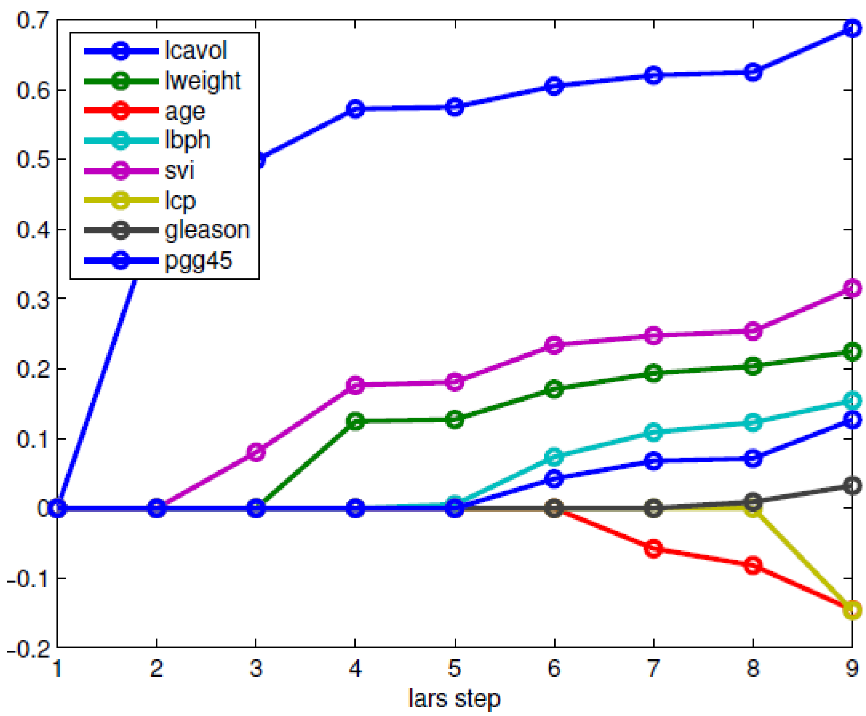
LARS provides solutions with increasingly many nnz entries.

24 Issues with Lasso

Lasso objective does not spread weights around on correlated predictors.

For example, given two equal predictors $\mathbf{x}_1 = \mathbf{x}_2$, Lasso objective

$$\frac{1}{2} \left\|\mathbf{y} - \mathbf{x}_1 \beta_1 - \mathbf{x}_2 \beta_2\right\|^2 + \lambda(|\beta_1| + |\beta_2|)$$



does not have any preference among solutions $(\beta, 0)$ $(\beta/2, \beta/2)$ $(0, \beta)$.

Hence, we can not interpret 0 weight as indication of the predictor being uninformative.

25 Today

1. Review Coordinate Descent for Ridge and Lasso
2. Fix-point analysis -- exam question practice
3. Full regularization path solution
4. Least Angle Regression solver for Lasso
5. Issues with Lasso

More details on full regularization path methods and coordinate descent: here