MATH-GA 2840 HW#4

Yifei(Fahy) Gao yg1753

1. (Proximal operator) The proximal operator of a function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\text{prox}_f(y) := \arg\min_x f(x) + \frac{1}{2}\|x - y\|_2^2$$

(a) Derive the proximal operator of the squared ℓ_2 norm weighted by a constant $\alpha > 0$, i.e. $f(x) = \alpha\|x\|_2^2$.

since we can set $f(x) = \alpha\|x\|_2^2$, then:

$$\begin{aligned}\nabla_x prox_f(y) &= \nabla_x \left(\alpha\|x\|_2^2 + \frac{1}{2}\|x - y\|_2^2 \right) \\ &= \nabla_x \left(\alpha x^T x + \frac{1}{2}(x^T x - 2x^T y + y^T y) \right) \\ &= 2\alpha x + x - y\end{aligned}$$

Since $\nabla_x prox_f(y) = 0$, then:

$$\begin{aligned}0 &= 2\alpha x + x - y \\ x &= \frac{y}{2\alpha + 1}\end{aligned}$$

(b) Prove that the proximal operator of the ℓ_1 norm weighted by a constant $\alpha > 0$ is a soft-thresholding operator,

$$\text{prox}_{\alpha\|\cdot\|_1}(y) = \mathcal{S}_\alpha(y)$$

where

$$\mathcal{S}_\alpha(y)[i] := \begin{cases} y[i] - \text{sign}(y[i])\alpha & \text{if } |y[i]| \geq \alpha \\ 0 & \text{otherwise} \end{cases}$$

We can rewrite the proximal operator with more details:

$$\text{prox}_{\alpha\|\cdot\|_1}(y) = \arg\min \|x\|_1 + \frac{1}{2}\|x - y\|_2^2$$

And we can see, we can define this operator respect to $x[i]$, where $i \in (-\infty, \infty)$ and it be seperated into three situations:

```
1.x[i]=0;
2.x[i]>0;
3.x[i]<0;
```

Then we can first differentiate the proximal operation:

$$\begin{aligned}\nabla_x \operatorname{prox}_{\alpha\|\cdot\|_1}(y) &= \nabla_x \left(\alpha\|x\|_1 + \frac{1}{2}\|x-y\|_2^2 \right) \\ &= \nabla_x \left(\alpha|x[i]| + \frac{1}{2}(x[i]^T x[i] - 2x[i]^T y[i] + y[i]^T y[i]) \right) \\ &= \nabla_x \alpha|x[i]| + \nabla_x \frac{1}{2}(2x[i]^T x[i] - 2x[i]^T y[i] + y[i]^T y[i]))\end{aligned}$$

For situation (1),

$$\nabla_x \operatorname{prox}_{\alpha\|\cdot\|_1}(y) = \nabla_x \alpha|0| + \nabla_x \frac{1}{2}(2 * 0^T 0 - 2 * 0^T y[i] + y[i]^T y[i])) = 0$$

which we can tell it is not differentiable due to the $l_1\, norm$.

For situation (2),

$$\begin{aligned}\nabla_x \operatorname{prox}_{\alpha\|\cdot\|_1}(y) = \alpha + \frac{1}{2}(2x[i] - 2y[i]) &= 0 \\ \alpha + x[i] - y[i] &= 0 \\ x[i] &= y[i] - \alpha\end{aligned}$$

For situation (3),

$$\begin{aligned}\nabla_x \operatorname{prox}_{\alpha\|\cdot\|_1}(y) = \alpha + \frac{1}{2}(2y[i] - 2x[i]) &= 0 \\ \alpha + y[i] - x[i] &= 0 \\ x[i] &= y[i] + \alpha\end{aligned}$$

And we have when $x[i] < 0, x[i] = y[i] + \alpha$ is the minimum when $y[i] < -a$

when $x[i] > 0, x[i] = y[i] - \alpha$is the minimum when $y[i] > a$

when $x[i] = 0$, there only exists zero.

Therefore, we can conclude that:

$$\mathcal{S}_\alpha(y)[i] := \begin{cases} y[i] - \operatorname{sign}(y[i])\alpha & \text{if } |y[i]| \geq \alpha \\ 0 & \text{otherwise} \end{cases}$$

(c) Prove that if $X \in \mathbb{R}^{p\times n}$ has orthonormal rows ($p \leq n$) and $y \in \mathbb{R}^n$, then for any function f

$$\arg\min_\beta \frac{1}{2}\left\|y - X^T\beta\right\|_2^2 + f(\beta) = \arg\min_\beta \frac{1}{2}\|Xy - \beta\|_2^2 + f(\beta)$$

For the equation, we actually only need to prove:
$\arg\min_\beta \frac{1}{2}\left\|y - X^T\beta\right\|_2^2 = \arg\min_\beta \frac{1}{2}\|Xy - \beta\|_2^2 + f(\beta)$,

so for the left side:

$$\begin{aligned}\arg\min_\beta \frac{1}{2}\left\|y - X^T\beta\right\|_2^2 &= \arg\min_\beta \frac{1}{2}(y - X^T\beta)^T(y - X^T\beta) \\ &= \arg\min_\beta \frac{1}{2}(y^T y - y^T X^T\beta - y^T X^T\beta + \beta^T XX^T\beta)\end{aligned}$$

Then let us go find the derivative of the proximal operator ∇_β, and make it equal to 0:

$$\nabla_\beta \frac{1}{2}(y^T y - y^T X^T \beta - y^T X^T \beta + \beta^T X X^T \beta) = \frac{1}{2}(-2y^T X^T + 2\beta^T X X^T \beta)$$
$$0 = -y^T X^T - \beta^T X X^T \beta$$
$$y^T X^T = X X^T \beta$$
$$\beta = (X X^T)^{-1} y^T X^T$$

For the right side:

$$\arg\min_\beta \frac{1}{2}\|Xy - \beta\|_2^2 = \arg\min_\beta \frac{1}{2}(Xy - \beta)^T (Xy - \beta)$$
$$= \arg\min_\beta \frac{1}{2}(y^T X^T X y - 2y^T X^T \beta + \beta^T \beta)$$

Then do the same thing as what we've done for the left side:

$$\nabla_\beta \frac{1}{2}(y^T X^T X y - 2y^T X^T \beta + \beta^T \beta) = \frac{1}{2}(-2y^T X^T + 2\beta)$$
$$0 = -y^T X^T + \beta$$
$$\beta = y^T X^T$$

And since X is orthnormal, then $X^T X = X X^T = I$, therefore:

$$\arg\min_\beta \frac{1}{2}\left\|y - X^T\beta\right\|_2^2 + f(\beta) = (X X^T)^{-1} y^T X^T = I y^T X^T = y^T X^T = \arg\min_\beta \frac{1}{2}\|Xy - \beta\|_2^2$$

(d) Use the answers to the previous questions to compare the ridge-regression and lasso estimators for a regression problem where the features are orthonormal.

For ridge-regression:

$$\arg\min_\beta \frac{1}{2}\left\|y - X^T\beta\right\|_2^2 + \lambda\|\beta\|_2^2$$

From what we have proved in part c, we know

$$\arg\min_\beta \frac{1}{2}\left\|y - X^T\beta\right\|_2^2 + \lambda\|\beta\|_2^2 = \arg\min_\beta \frac{1}{2}\|Xy - \beta\|_2^2 + \lambda\|\beta\|_2^2 = \text{prox}_{\lambda\|\cdot\|_2}(Xy))$$

Thus based on the part a,

$$\beta_{RR} = \frac{Xy}{2\lambda + 1}$$

Similarly, for lasso estimators:

$$\arg\min_\beta \frac{1}{2}\left\|y - X^T\beta\right\|_2^2 + \lambda\|\beta\|_1 = \arg\min_\beta \frac{1}{2}\|Xy - \beta\|_2^2 + \lambda\|\beta\|_1 = \text{prox}_{\lambda\|\cdot\|_1}(Xy)$$

Then based on part b, we get:

$$\beta_{lasso} = S_\lambda(Xy)$$

2. (Proximal gradient method)

(a) The first-order approximation to a function $f : \mathbb{R}^p \to \mathbb{R}$ at $x \in \mathbb{R}^p$ equals $f(x) + \nabla f(x)^T(y - x)$ We want to minimize this first-order approximation locally. To this end we fix a real constant $\alpha > 0$ and augment the approximation with an ℓ_2 -norm term that keeps us close to x

$$f_x(y) := f(x) + \nabla f(x)^T(y - x) + \frac{1}{2\alpha}\|y - x\|_2^2$$

Prove that the minimizer of f_x is the gradient descent update $x - \alpha\nabla f(x)$.

Define $\hat{y}$ as the minimizer of the function $f_x(y)$, then:

$$\begin{aligned}\hat{y} = \arg\min_y f_x(y) &= \arg\min_y \{ f(x) + f(x)^T(y - x) + \frac{1}{2\alpha}\|y - x\|_2^2\} \\ &= \arg\min_y \{ f(x) + f(x)^T(y - x) + \frac{1}{2\alpha}(y^T y - 2y^T x + x^T x)\end{aligned}$$

Then set the result to 0,

$$\begin{aligned}\nabla_y f_x(y) &= \nabla f(x)^T + \frac{1}{2\alpha}(2y - 2x) \\ 0 &= \nabla f(x)^T + \frac{1}{\alpha}(y - x) \\ y &= x - \alpha\nabla f(x)^T \ \blacksquare\end{aligned}$$

(b) Inspired by the previous question, how would you modify gradient descent to minimize a function of the form

$$h(x) = f_1(x) + f_2(x)$$

where f_1 is differentiable, and f_2 is nondifferentiable but has a proximal operator that is easy to compute?

Based part a), the gradient descent update step from $x^k -> x^{k+1}$ as:

$$x^{k+1} = x^k - \alpha\nabla f(x^k) = \arg\min_y \{ f(x^k) + \nabla f(x^k)^T(y - x^k) + \frac{1}{2\alpha}\|y - x^k\|_2^2\}$$

since we know $h(x) = f_1(x) + f_2(x)$ and the part a gives us f_1 then,

$$x^* = \arg\min_y \{ f(x^k) + \nabla f(x^k)^T(y - x^k) + \frac{1}{2\alpha}\|y - x^k\|_2^2 + f_2(x)\}$$

And since $\arg\min_y \{ f(x^k) + \nabla f(x^k)^T(y - x^k) + \frac{1}{2\alpha}\|y - x^k\|_2^2\}$ is the gradient descent update rule, so $y = x - \alpha\nabla f(x)$ and:

$$\hat{x} = \arg\min_y \{ \frac{1}{2\alpha}\|x - (x^k - \alpha\nabla f(x^k)\|_2^2 + f_2(x)\}$$

And this is exactly the proximal operator of $\text{prox}_{af_2}(x^k - \alpha\nabla f(x^k))$, so the proximal gradient descent is:

$$\hat{x} = x^{k+1} = \text{prox}_{\alpha f_2}(x^k - \alpha\nabla f_1(x^k)) \ \blacksquare$$

(c) Show that a vector x^* is a solution to

$$\text{minimize} \quad f_1(x) + f_2(x),$$

where f_1 is differentiable, f_2 is nondifferentiable and both functions are convex, if and only if it is a fixed point of the iteration you proposed in the previous question for any $\alpha > 0$

Since f_1, f_2 are both convex, then let $h(x) = f_1 + f_2$. $h(x)$ is convex as well. And based on the Optimality Condition, A convex function attains its minimum value at a vector x if and only if the zero vector is a subgradient of f at x. Thus:

First, we can let a_{min} to be the minimum, and $a_{min} = \arg\min_x h(x) = \arg\min_x \{\frac{1}{2\alpha}\|x - (x^k - \alpha$

And let $a_{min} = x^* = \text{prox}_{\alpha f_2}(x^k - \alpha \nabla f_1(x^k))$,

$$\begin{aligned}
\nabla_{a_{min}} h(a_{min}) &= \nabla_{a_{min}} h\left(x^k - \alpha \nabla f_1\left(x^k\right)\right) \\
&= \nabla_{a_{min}} \left[\frac{1}{2}\left\|\left(x^k - \alpha \nabla f_1\left(x^k\right)\right) - \left(x^k - \alpha \nabla f_1\left(x^k\right)\right)\right\|_2^2 + \alpha f_2\left(x^k - \alpha \nabla f_1\left(x^k\right)\right)\right. \\
&= \nabla_{a_{min}} \alpha f_2\left(x^k - \alpha \nabla f_1\left(x^k\right)\right) \\
&= \nabla_{a_{min}} \alpha f_2\left(x^*\right) = \nabla_{a_{min}} h\left(x^*\right)
\end{aligned}$$

Since $\nabla_{a_{min}} h(x^*)$ is the subgradient of the function $h(x)$ at x^* where $0 \in \nabla_{a_{min}} h(x^*)$ by definition, so the function attains its minimum optimization at the fixed point x^* with positive α. ∎

3. (Iterative shrinkage-thresholding algorithm)

(a) What is the proximal gradient update corresponding to the lasso problem defined below? Your answer will involve a hyperparameter which we will call α.

$$\frac{1}{2}\|y - X\beta\|_2^2 + \lambda|\beta|_1$$

Since we can define the lasso problem as the sum of two convex functions which one is differentiable and the other is not as what we showed from the past question:

$$\begin{aligned}
f_1 &= \frac{1}{2}\|y - X\beta\|_2^2 \\
f_2 &= \lambda|\beta|_1
\end{aligned}$$

Then from part b in question 2, we get:

$$\beta_{lasso}^{k+1} = \text{prox}_{\alpha f_2}(\beta_{lasso}^k - \alpha \nabla f_1(\beta_{lasso}^k)) = \arg\min_\beta \{\frac{1}{2}\|\beta - (\beta_{lasso}^k - \alpha \nabla f_1(\beta_{lasso}^k))\|_2^2 + \alpha\lambda|\beta|_1\}$$

From part b in question 1,

$$\text{prox}_{\alpha\|\cdot\|_1}(y) = \arg\min \|x\|_1 + \frac{1}{2}\|x - y\|_2^2 = S_\alpha(y) = y - sign(y)\alpha \text{ when } |y| \geq \alpha$$

So

$$\beta_{lasso}^{k+1} = \text{prox}_{\alpha\lambda\|\cdot\|_1}(\beta_{lasso}^k) = S_{\alpha\lambda}(\beta_{lasso}^k - \alpha \nabla f_1(\beta_{lasso}^k)) = S_{\alpha\lambda}(\beta_{lasso}^k - \alpha(X^T X \beta_{lasso}^k - X^T y))$$

(b) How would you check whether you have reached an optimum? How would you modify this to take into account possible numerical inaccuracies?

take into account possible numerical inaccuracies?

First we use the β we find in each iteration to set up this equation:

$$\beta^{k+1} = \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|$$

,

Then we can create a tolerance (tol) for the difference in loss and if the difference is smaller or equal to the tol, then we can say it reach the optimum. However, the numerical inaccrancies might exist because of the X^TX term so we should revise the equation as the part b in question 1:

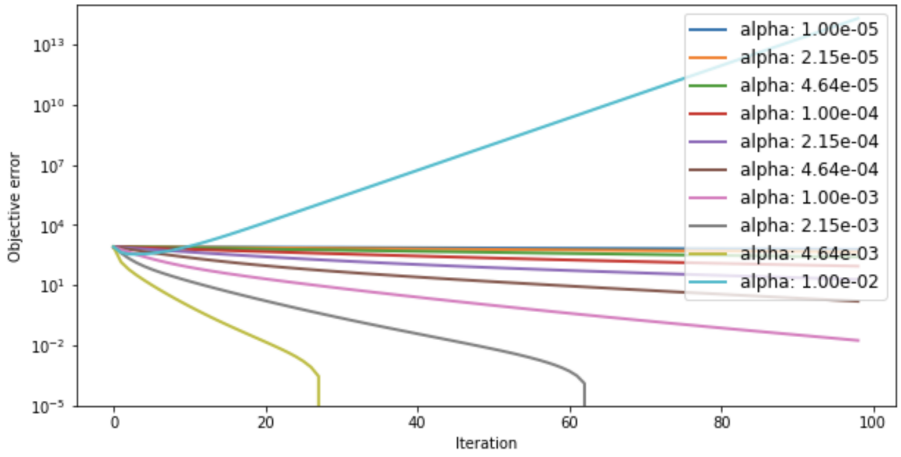
$$\beta^{k+1} = S_{\alpha\lambda}(\beta^k - \alpha X^T(X\beta^k_{lasso} - y))$$

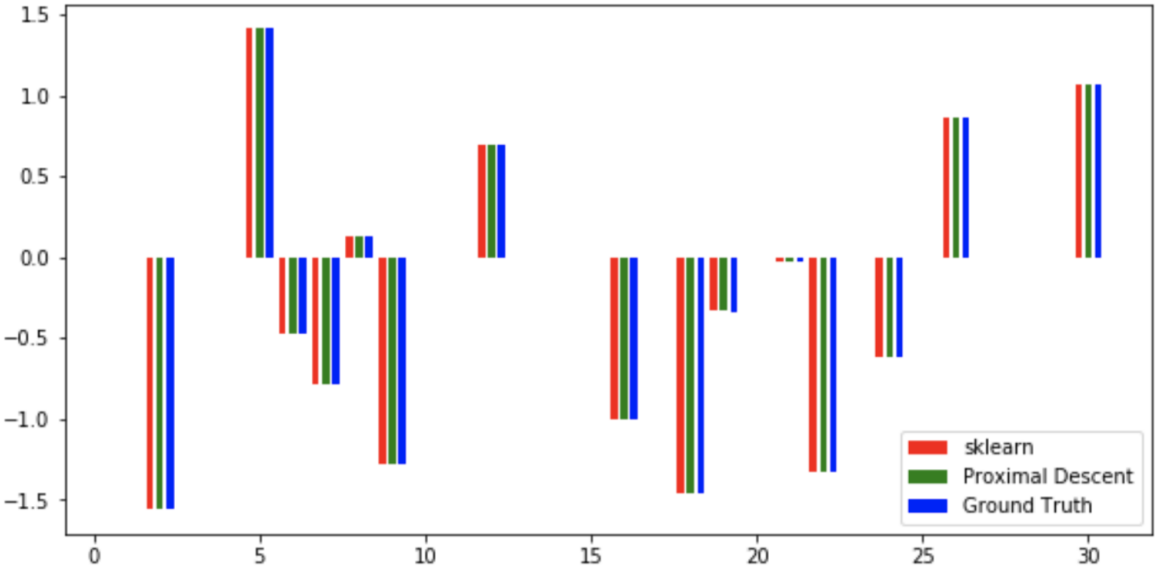
So we can calculate the $X\beta^k_{lasso} - y$ first to avoid the X^TX then do the rest vector multiplications.

(c) Implement the method and apply it to the problem in pgd_lasso-question. ipynb. You have to fill in blocks of code corresponds to the proximal gradient update step and termination condition. Report all the generated plots.

Here is my code:

```
## Proximal Gadient

obj_pg = {} #stores obj function value as a function of iteration for each alpha
w_pg = {} #stores the final weight vector learned for each alpha
tol=1e-5

for alpha in alpha_array:
    print('Alpha: ', alpha)

    w_pg[alpha] = np.matrix([0.0]*dim).T
    obj_pg[alpha] = []


    for t in range(1, max_iter):
        obj_val = obj(w_pg[alpha])
        obj_pg[alpha].append(obj_val.item())

        w_pg[alpha] = w_pg[alpha]-alpha*X.T@(X*w_pg[alpha]-y)
        w_pg[alpha] = np.multiply(np.sign(w_pg[alpha]),np.maximum(np.abs(w_pg[alpha])-lamda*alpha,0))


        new_val=obj(w_pg[alpha])

        if (t%5==0):
            print('iter= {},\tobjective= {:3f}'.format(t, obj_val.item()))

        subgradient_norm=np.abs(new_val-obj_val)

        if subgradient_norm < tol:
            print("At iteration"+str(t)+"="+str(subgradient_norm)+" < "+ str(epsilon)+" it reaches the convergence. ")

            break
```





4. (Forward selection) A very simple way to fit a sparse linear model is to build it gradually by greedily selecting features that are correlated with the residual. This is called forward selection in statistics. Let $y_{\text{train}} \in \mathbb{R}^n$ be a vector containing the response, and $X \in \mathbb{R}^{p \times n}$ the corresponding feature matrix. We initialize the set $\mathcal{S}$ of selected features to be empty and the residual to equal $r := y$. Then we update the model until $\mathcal{S}$ contains a predetermined number of features k (which can be chosen by cross validation). The updates consist of incorporating the feature that is most correlated with the residual, and recomputing the residual,

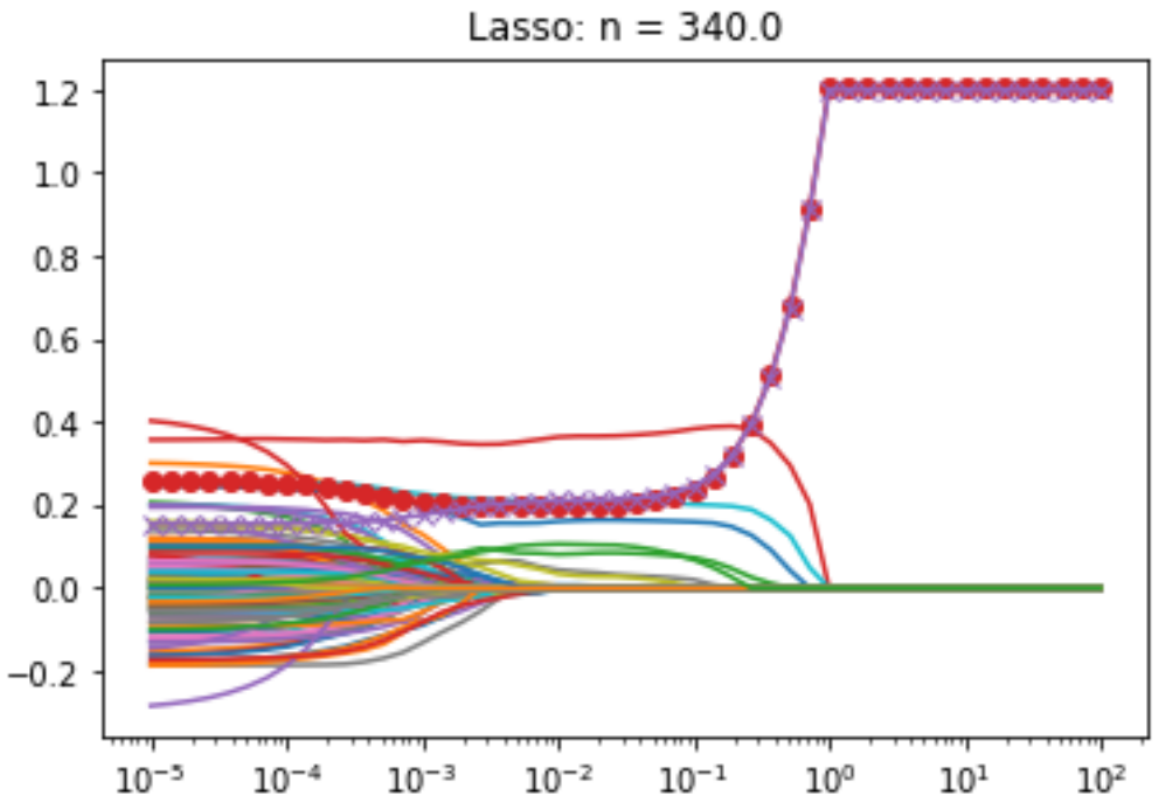
$$
\begin{aligned}
i^* &:= \arg\max_i \left| x_i^T r \right| \\
\mathcal{S} &:= \mathcal{S} \cup \{i^*\} \\
r &:= y - X_{\mathcal{S}}^T \beta_{\mathcal{S}}, \quad \beta_{\mathcal{S}} := \left(X_{\mathcal{S}}^T X_{\mathcal{S}} \right)^{-1} X_{\mathcal{S}} y
\end{aligned}
$$

Implement this approach and apply it to the temperature example from the sparse regression notes by filling in the FSR function in the notebook FSR . ipynb Submit and describe all the plots generated by the notebook.

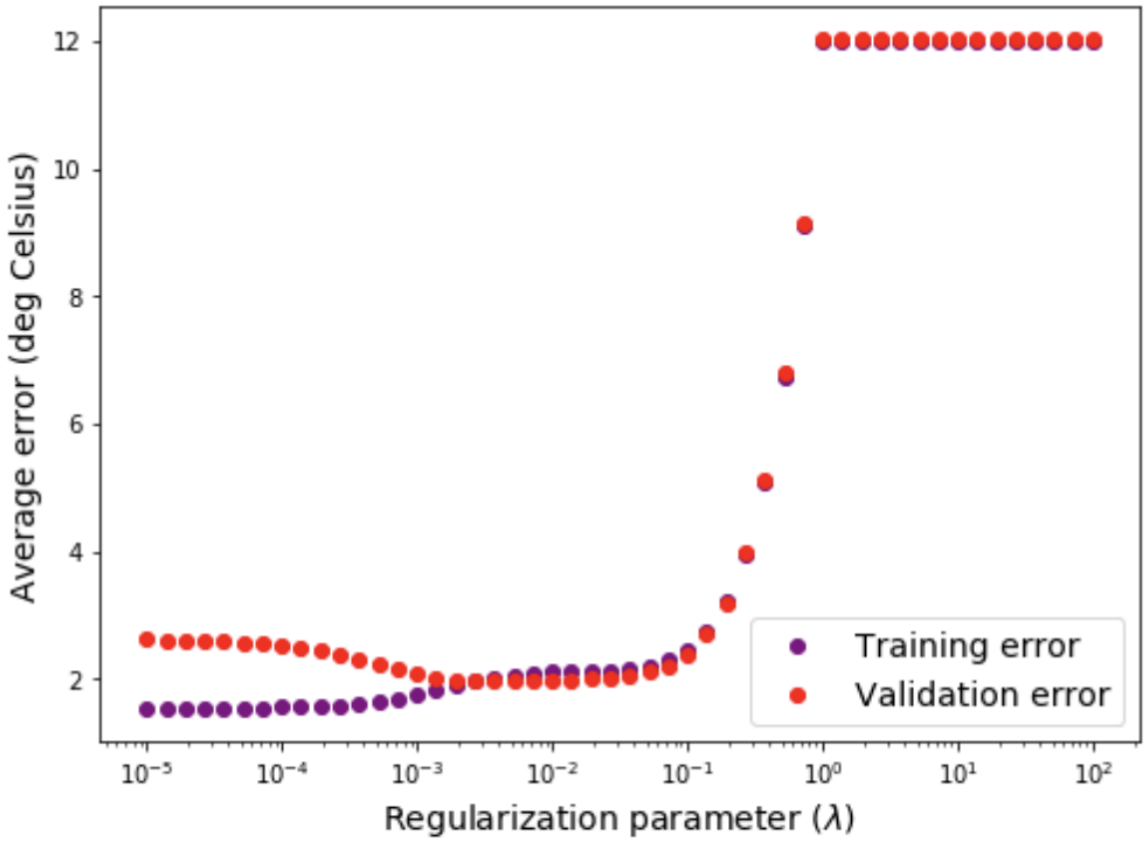
Here is my code of FSR:

```
def FSR(X,y,num_features=20):
    '''
    X: feature matrix
    y: response
    return
        coeff: coefficient betta
    '''
    SF=[]
    residual=y
    beta = np.zeros(X.shape[1])
    while len(SF)<num_features:
        i = np.argmax(np.abs(X.T@residual))
        SF.append(i)
        beta[S]=np.linalg.pinv(X[:,SF].T@X[:,SF])@X[:,SF].T@y
        residual=y -X@beta

    coeff= beta

    return coeff
```

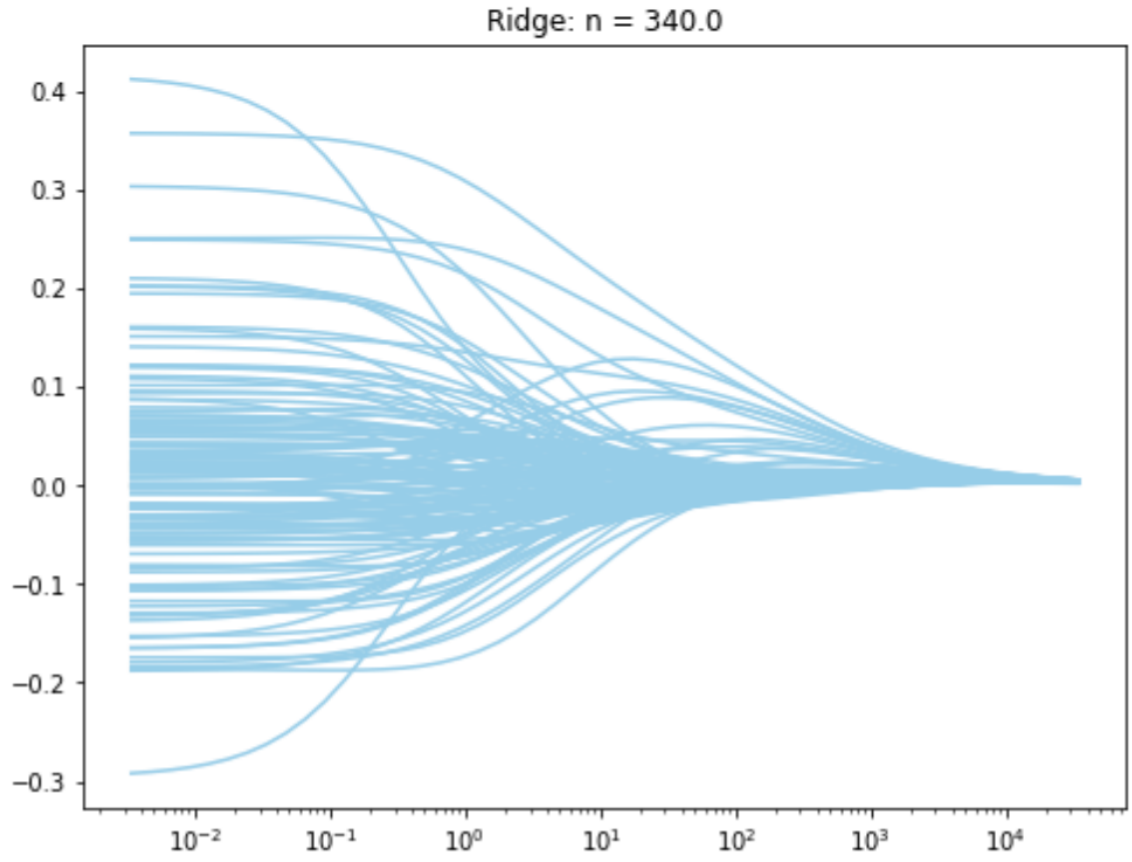
After pluging the code, I got these 11 graphs and the following explanations:



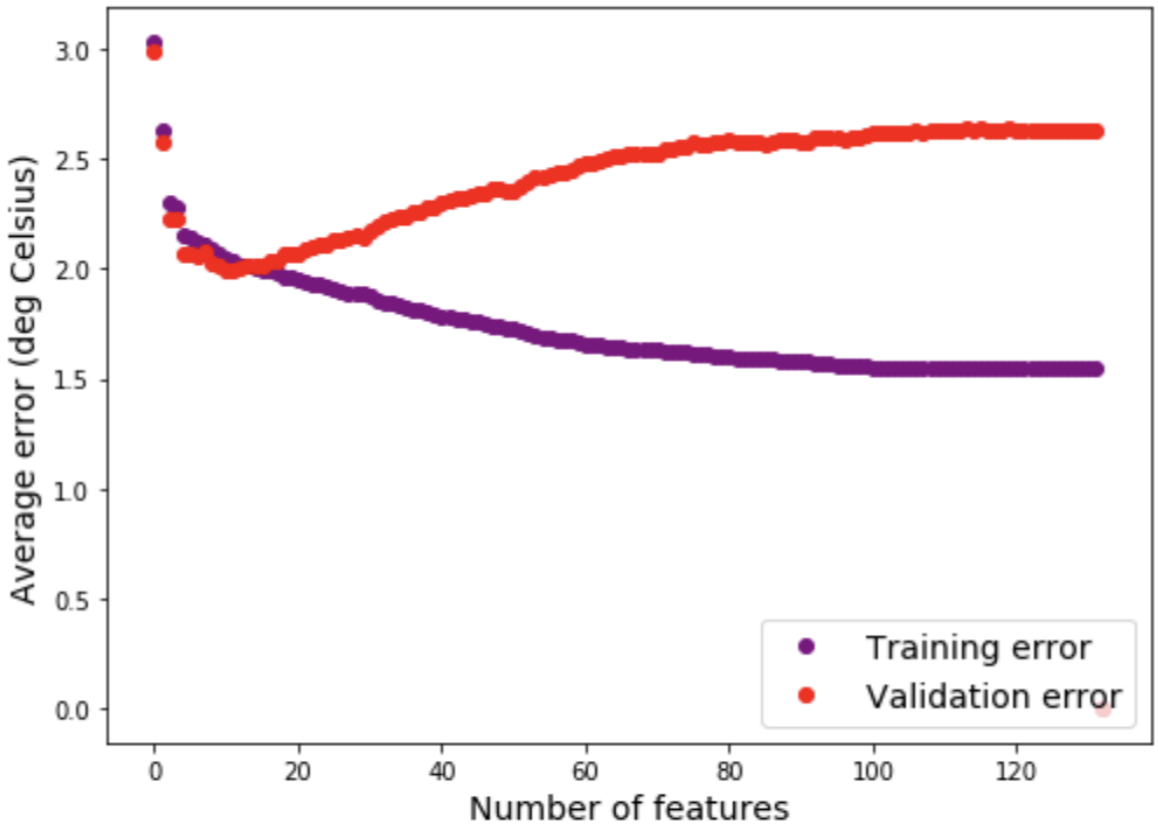
This is Lasso at n = 340. The coefficients are sparse as regularized parameters (λ). The validation and training error both increase to 1.2 when the regularization parameters increase till the point that every parameters turn to be zero.



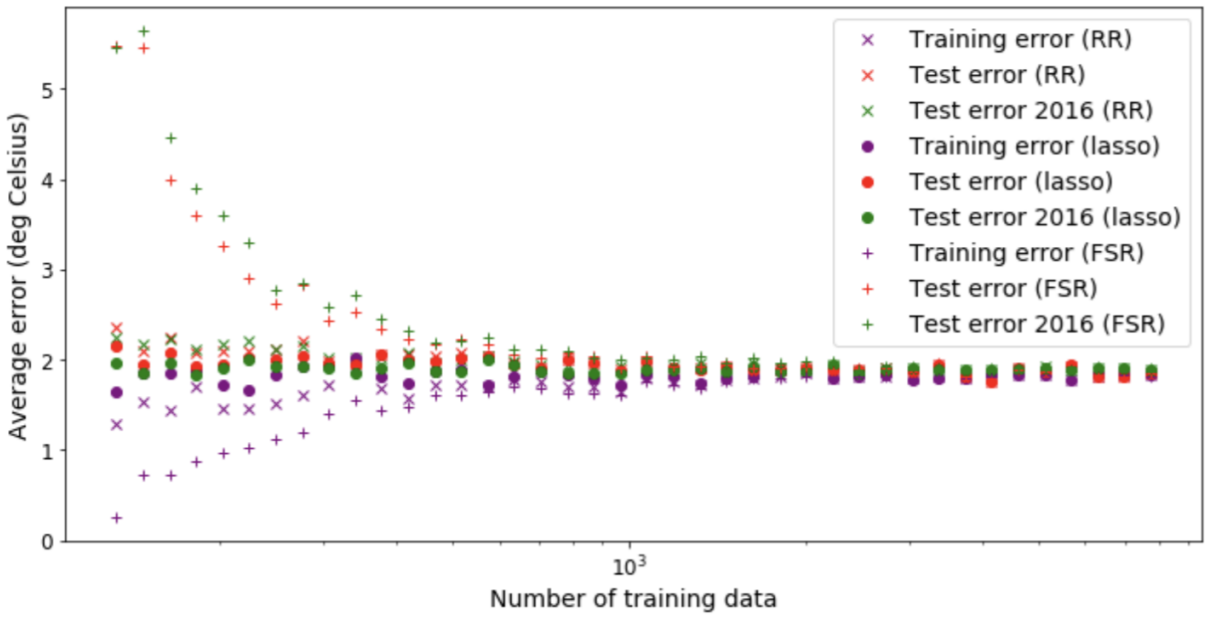
This is just a more clearr plot to explain the change of Validation error and Training error that they reach 12 when regularization parameters increase.



This is the plot of Ridge coefficients with n=340 of regularization parameters.

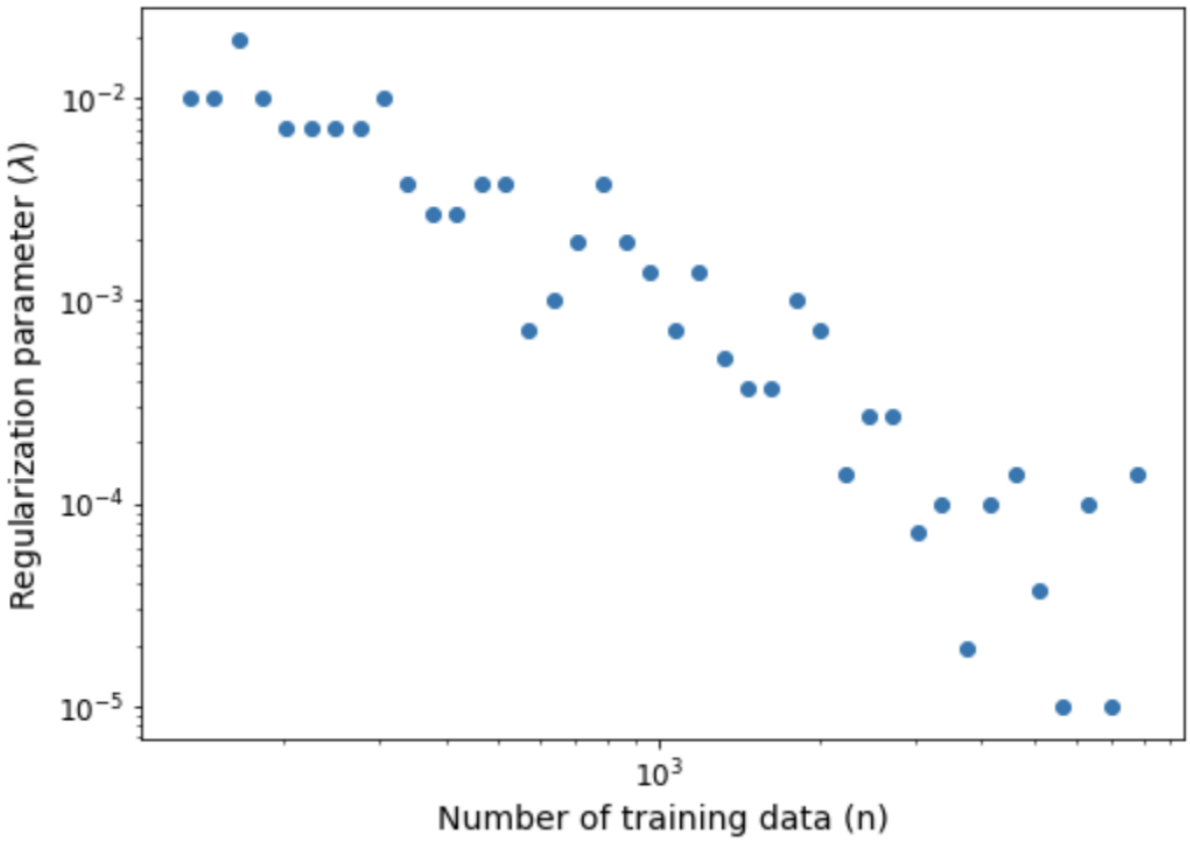


This is Forward Selection Regression. Training error decreases with number of features increase, while the validation error increases.

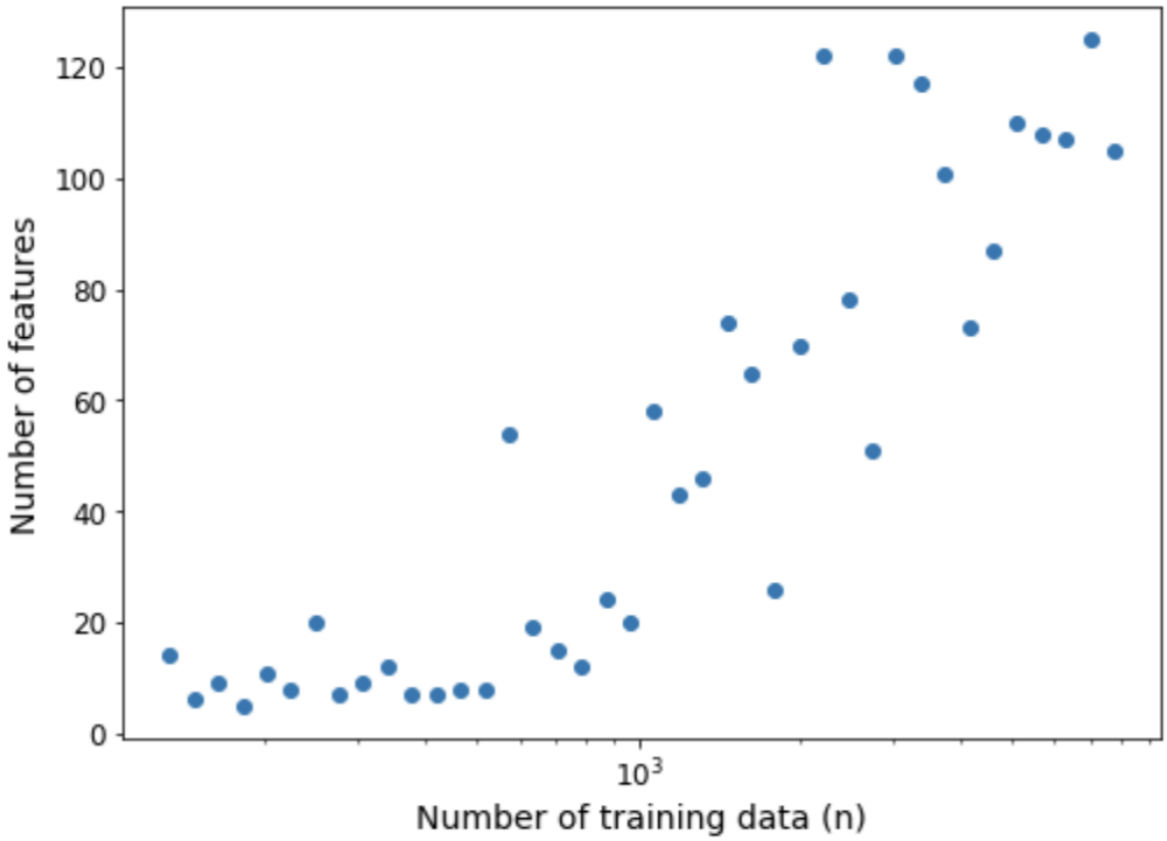


This is a collection of the errors for different methods with small number of training data. We can see from the scatter plot that FSR has largest average error on Test error but lowest training error.

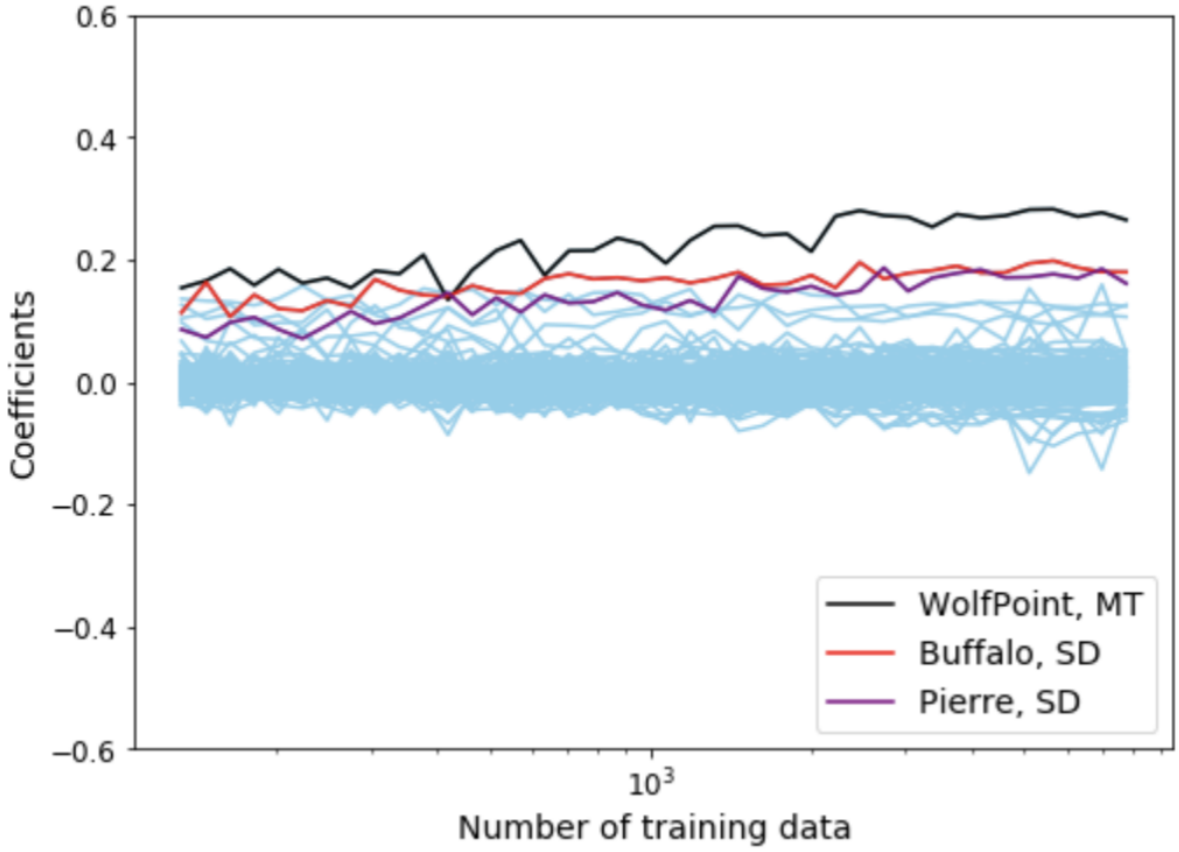
However, as the number of training data increase, FSR eventually converge to the similar error of ridge and lasso estimator.



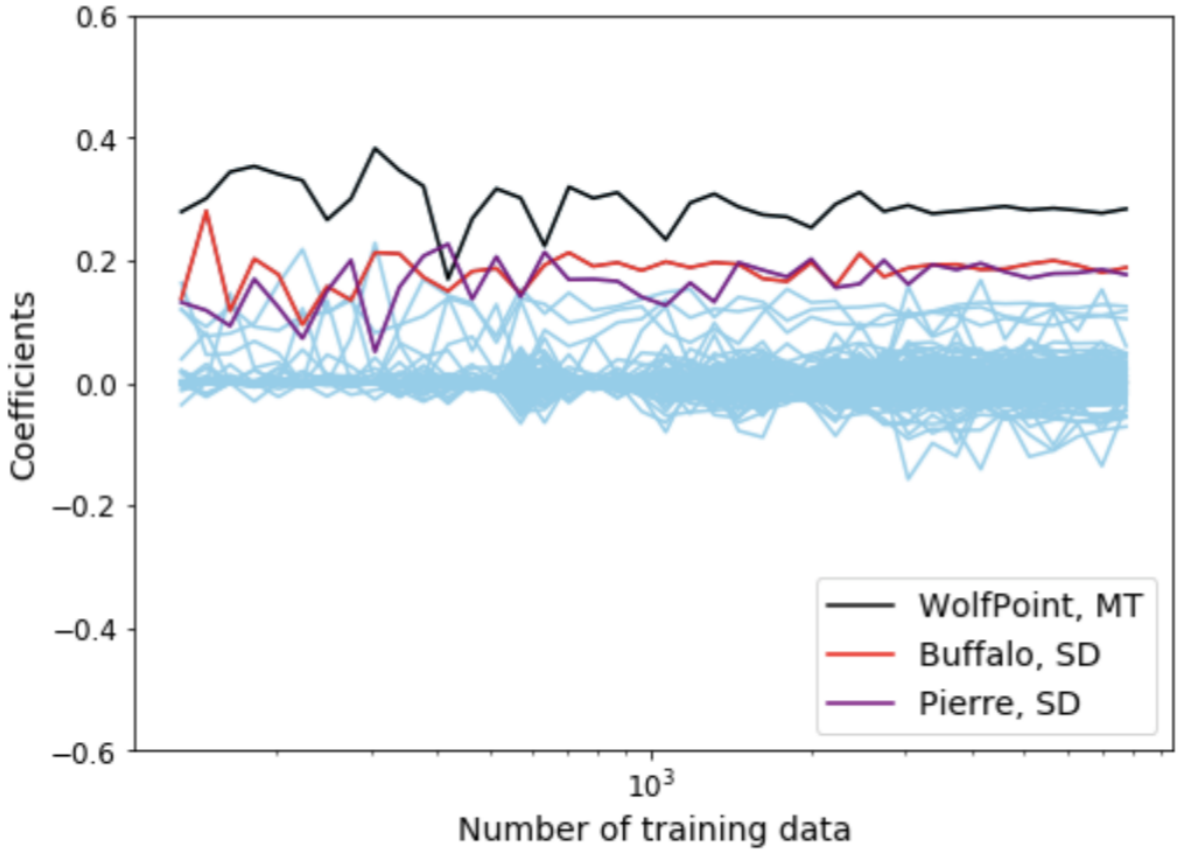
This is a scatter plot to show the relationship of number of training data (n) and the regularization parameter (λ). As # of training data increase, λ decreases in general.

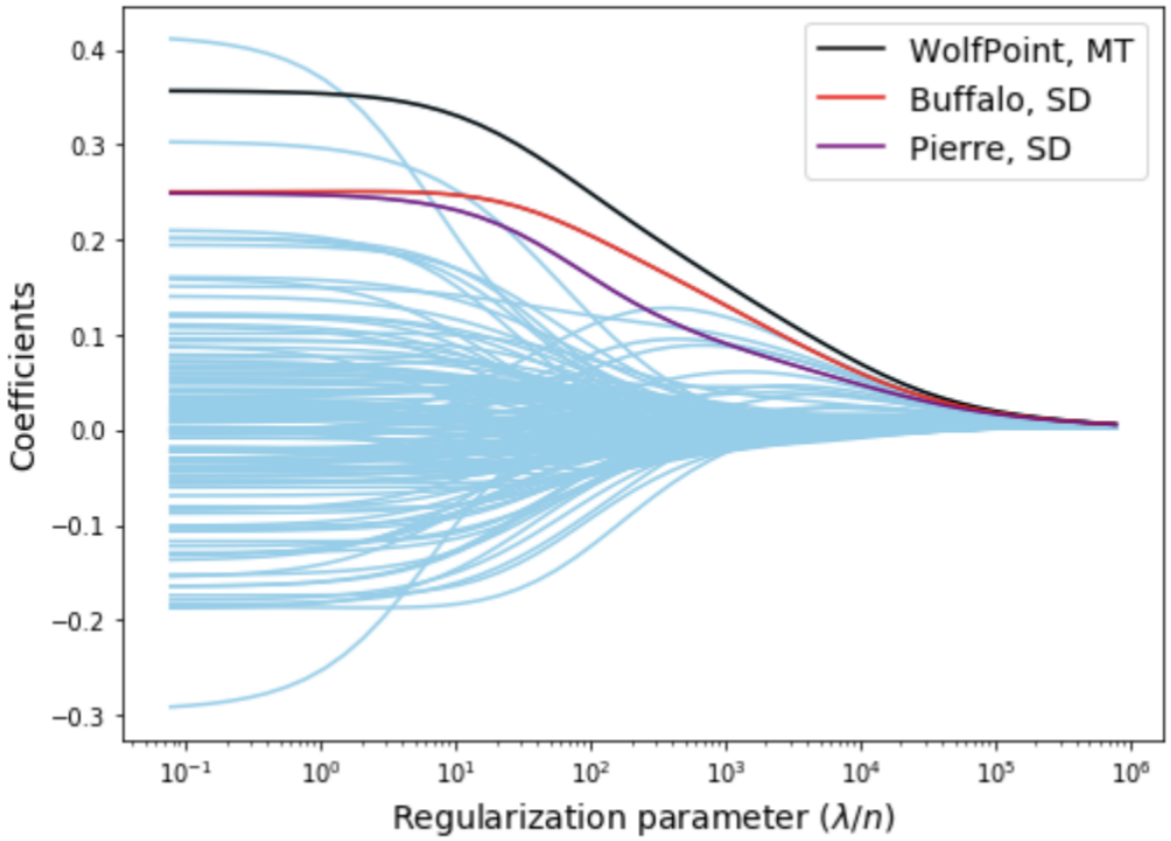


This is a scatter plot to show the relationship of number of training data (n) and the number of features. As # of training data increase, features increases.

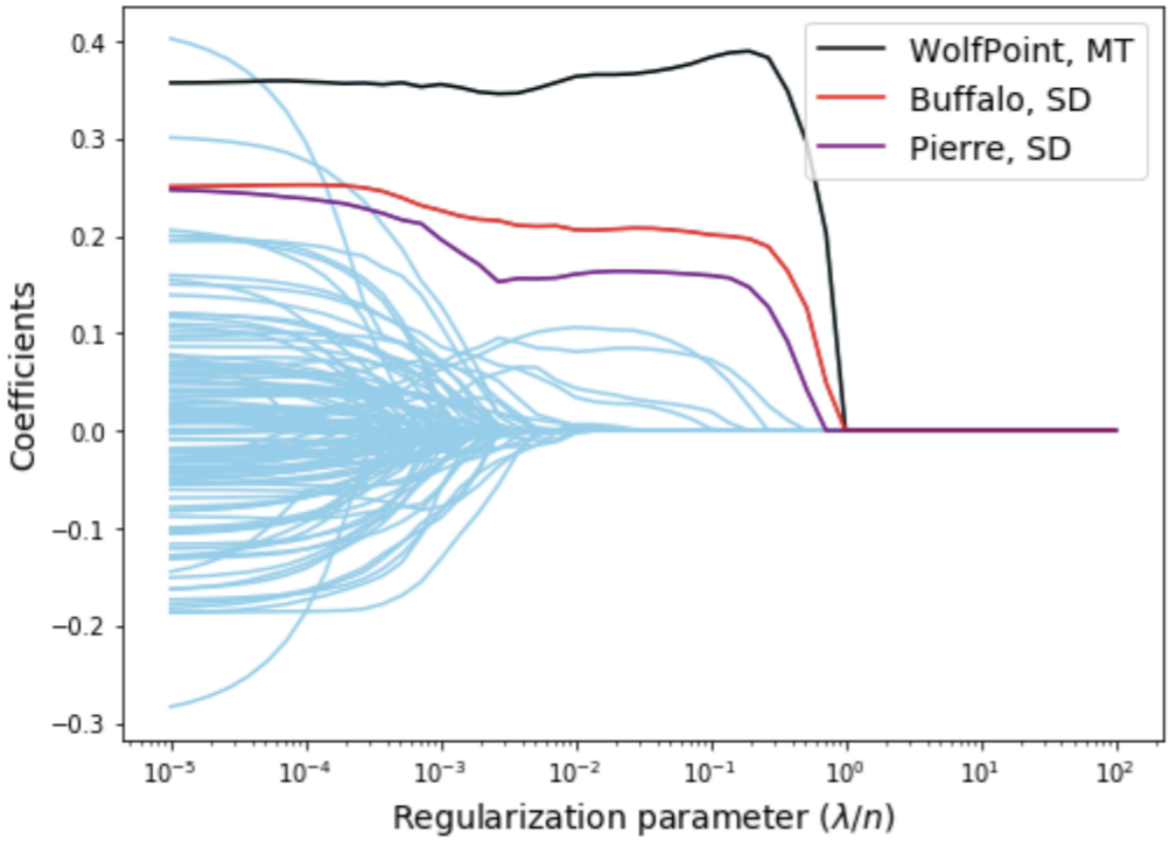


This is a plot to show the lasso sparse-regression method.





These two plots above are plots to show the ridge regression method.



This is a plot to show the FAS method.

Overall, FSR has large average error. With the number of training examples increasing, FSR converge to ridge and lasso estimator as zero.