Do not distribute course material

You may not and may not allow others to reproduce or distribute lecture notes and course materials publicly whether or not a fee is charged.

https://www.cs.princeton.edu/courses/archive/spr09/cos513/scribe/lecture10.pdf

http://www.stat.cmu.edu/~ryantibs/advmethods/notes/highdim.pdf

http://timroughgarden.org/s17/l/l6.pdf

Regularization Preventing overfitting

A way to prefer some model/hypothesis over others in our class based on some idea of what is the preferred model

ence…

How can we reduce the out of sample error by preferring some solutions in our hypothesis class

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y})^2 = \frac{1}{N}RSS(\mathbf{w})$$

We will be modifying our linear regression algorithm.

The techniques will discuss will be used in other algorithms later in the semester.

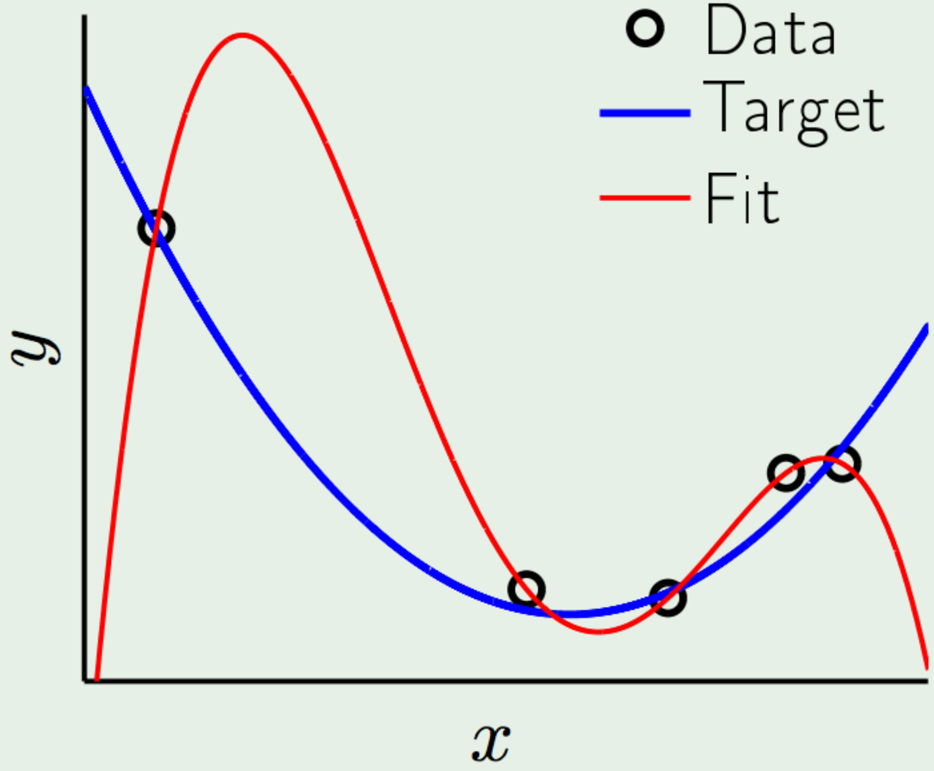
Learning objectives

- Understand regularizer can be used to decrease overfitting
- Understand how to create an objective function that prefers functions with smaller coefficients (simpler models) by adding a L1 or L2 penalty term
- Understand why we don’t regularize the bias term
- How the L1 or L2 penalty term affects bias and variance
- How to use model selection to tune the hyper-parameter λ
- L1 regularization can produce feature selection
- Understand that feature scaling should be performed in most cases

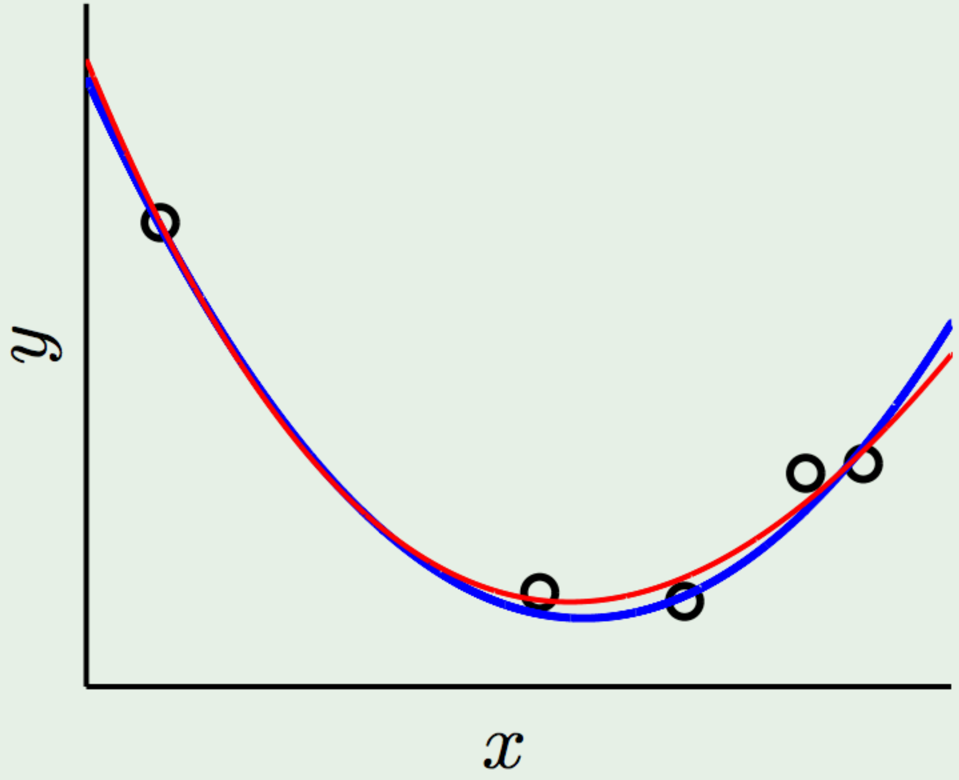
Occam's razor (Latin: *novacula Occami*)

“entities should not be multiplied beyond necessity”, sometimes inaccurately paraphrased as "the simplest explanation is usually the best one.”

Putting the brakes

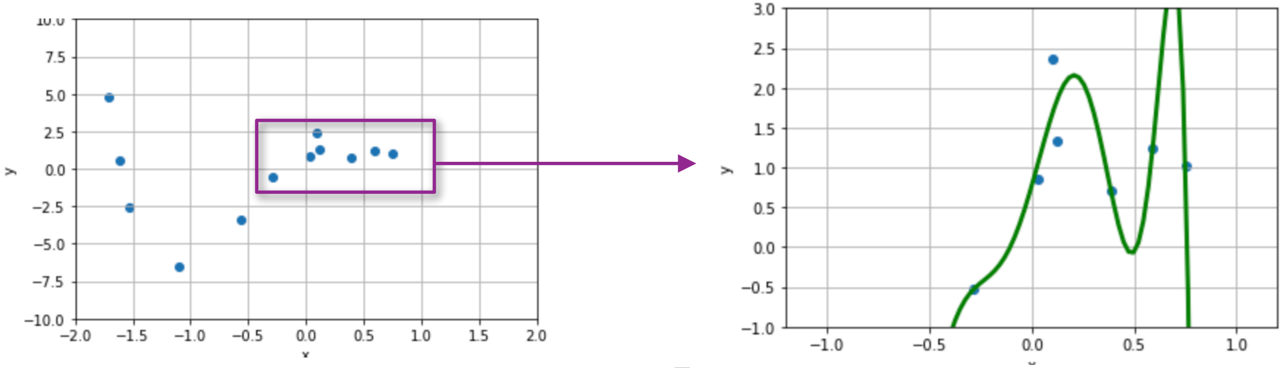


free fit



restrained fit

Example:



Poor Generalization!

$\mathbf{w}^T$=[0.8   9   9   -76 -176 208   551 36 -479 -328 -66]

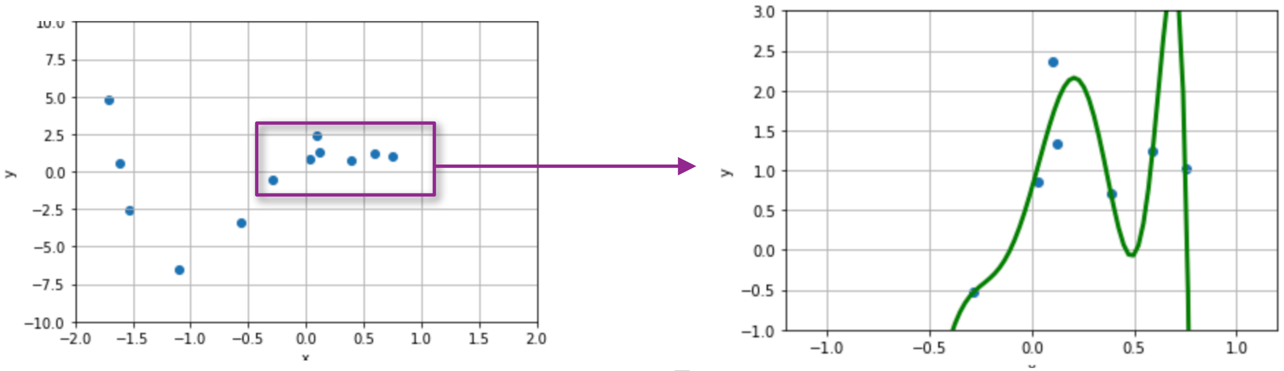
Observations:
Notice that the amount of overfitting depended on the order of the model and how many examples we have.
Our hypothesis that overfit had large coefficients. How could we keep the coefficients small?

We will need to balance between how well we fit the data (the in sample error) and how much we restrict the size of our coefficients (that we are using to prevent overfitting)

$$\underbrace{E_{\text{in}}(\mathbf{w})}_{\text{fit}} + \underbrace{\text{penalty for large } \mathbf{w}}_{\text{restrict the size of our coefficients}}$$

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y})^2 = \frac{1}{N}RSS(\mathbf{w})$$

Example:



Poor Generalization!

$\mathbf{w}^T$=[0.8 9 9 -76 -176 208 551

If $w_j = 551$ then a small change to the value of the jth feature makes a huge change in $\hat{y}$

Observations:
Notice that the amount of overfitting depended on the order of the model and how many examples we have.
Our hypothesis that overfit had large coefficients. How could we keep the coefficients small?

We will need to balance between how well we fit the data (the in sample error) and how much we restrict the size of our coefficients (that we are using to prevent overfitting)

$$\underbrace{E_{\text{in}}(\mathbf{w})}_{\text{fit}} + \underbrace{\text{penalty for large } \mathbf{w}}_{\text{restrict the size of our coefficients}}$$

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y})^2 = \frac{1}{N}RSS(\mathbf{w})$$

Preventing Overfitting

- We prefer to have smaller coefficients, or a smaller number of parameters.
  - How do we choose smaller coefficients?
  - How do we choose which parameters are important?
- Want to reduce variance (and possibly increase bias)
- To do this we can change our *objective function*!

$$E_{aug}(\mathbf{w}) = E_{in}(\mathbf{w}) + \text{penalty for complex models}$$

What function should we use?

$$E_{lasso}(\mathbf{w}) = E_{in}(\mathbf{w}) + \begin{matrix}\text{penalty}\\ \text{on}\end{matrix} \left(\left|\cancel{w_0}\right| + \left|w_1\right| + \cdots + \left|w_d\right|\right)$$

We will explore this penalty - LASSO regression

$$E_{ridge}(\mathbf{w}) = E_{in}(\mathbf{w}) + \begin{matrix}\text{penalty}\\ \text{on}\end{matrix} \left(\cancel{w_0^2} + w_1^2 + \cdots + w_d^2\right)$$

We will explore this penalty - Ridge Regression

sum of coefficients?

$$\cancel{w_0} + w_1 + w_2 + \cdots + w_d$$

sum of absolute value of coefficients?

$$\left|\cancel{w_0}\right| + \left|w_1\right| + \left|w_2\right| + \cdots + \left|w_d\right|$$

Sum of squares of coefficients?

$$\cancel{w_0^2} + w_1^2 + w_2^2 + \cdots + w_d^2$$

- Tuning parameter λ to balance fit and number of parameters

$$E_{lasso}(\mathbf{w}) = E_{in}(\mathbf{w}) + \lambda\left(\left|\cancel{w_0}\right| + \left|w_1\right| + \cdots + \left|w_d\right|\right)$$

$$E_{ridge}(\mathbf{w}) = E_{in}(\mathbf{w}) + \lambda\left(\cancel{w_0^2} + w_1^2 + \cdots + w_d^2\right)$$

λ is called the tuning parameter.

λ determines the amount regularization

Preventing Overfitting

d = # features. Note we don't want to restrict w_0. Many approaches to this issue are possible. We will leave w_0 out of the penalty term. Note: Scaling the features is suggested

sum of coefficients?

$$\cancel{w_0} + w_1 + w_2 + \cdots + w_d$$

sum of absolute value of coefficients?

$$|\cancel{w_0}| + |w_1| + |w_2| + \cdots + |w_d|$$

Sum of squares of coefficients?

$$\cancel{w_0^2} + w_1^2 + w_2^2 + \cdots + w_d^2$$

- ❑ We prefer to have smaller coefficients, or a smaller number of parameters.
  - How do we choose smaller coefficients?
  - How do we choose which parameters are important?
- ❑ Want to reduce variance (and possibly increase bias)
- ❑ To do this we can change our *objective function*!

$$\mathrm{E}_{\text{aug}}(\mathbf{w}) = \mathrm{E}_{\text{in}}(\mathbf{w}) + \text{penalty for complex models}$$

This is a technique that is used in when learning many other models

$$\mathrm{E}_{\text{lasso}}(\mathbf{w}) = \mathrm{E}_{\text{in}}(\mathbf{w}) + \text{penalty on } \left(|\cancel{w_0}| + |w_1| + \cdots + |w_d|\right)$$

We will explore this penalty - LASSO regression

$$\mathrm{E}_{\text{ridge}}(\mathbf{w}) = \mathrm{E}_{\text{in}}(\mathbf{w}) + \text{penalty on } (\cancel{w_0^2} + w_1^2 + \cdots + w_d^2)$$

We will explore this penalty - Ridge Regression

- ❑ Tuning parameter λ to balance fit and number of par

$$\mathrm{E}_{\text{lasso}}(\mathbf{w}) = \mathrm{E}_{\text{in}}(\mathbf{w}) + \lambda\left(|\cancel{w_0}| + |w_1| + \cdots + |w_d|\right)$$

$$\mathrm{E}_{\text{ridge}}(\mathbf{w}) = \mathrm{E}_{\text{in}}(\mathbf{w}) + \lambda(\cancel{w_0^2} + w_1^2 + \cdots + w_d^2)$$

Choosing the right λ is also part of model selection

We will focus on choosing the right tuning parameter for accuracy (not interpretation)

zation

Ridge Regression

L_2 regularization

The size of a vector is referred to as the norm of the vector. What is the "size". It depends

The L2 norm of a vector is the square root of the sum of the squared vector values

$$\mathbf{v}^T = [1,2,3]$$

$$\| \mathbf{v} \|_2 = \sqrt{1^2 + 2^2 + 3^2}$$

$$\mathrm{E}_{\mathrm{ridge}}(\mathbf{w}) = E_{\mathrm{in}}(\mathbf{w}) + \lambda(\cancel{w_0^2} + w_1^2 + w_2^2 + \cdots + w_d^2)$$

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y})^2 = \frac{1}{N}RSS(\mathbf{w})$$

Ridge Regression
L_2 regularization

- Tuning parameter λ to balance fit and number of parameters

$$\mathrm{E}_{\text{ridge}} = E_{\text{in}}(\mathbf{w}) + \text{penalty for complex models}$$

$$\mathrm{E}_{\text{ridge}}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \lambda(w_1^2 + w_2^2 + \cdots + w_d^2)$$

- λ controls the model complexity
  - Large λ
    - high bias, low variance
  - small λ
    - low bias, high variance

If λ=0 then

$$\mathbf{w}_{\text{ridge}} = \mathbf{w}_{\text{lin}}$$

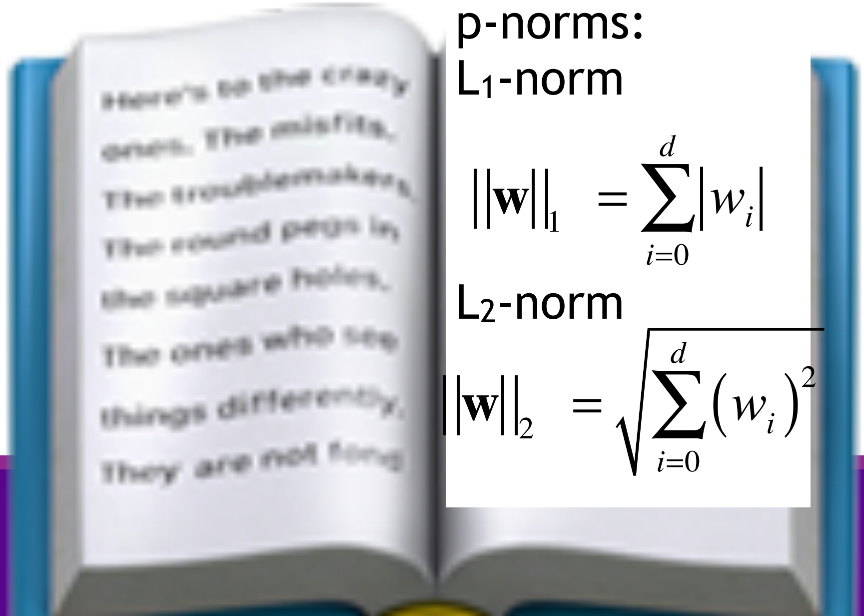
$\mathbf{w}_{\text{lin}}$ = the best parameters for least squares cost

If λ very large then

$$w_i \sim 0 \text{ for all i}$$

If λ is a constant then

$$0 \leq \|\mathbf{w}_{\text{ridge}}\|_2^2 \leq \|\mathbf{w}_{\text{lin}}\|_2^2$$



p-norms:

L_1-norm

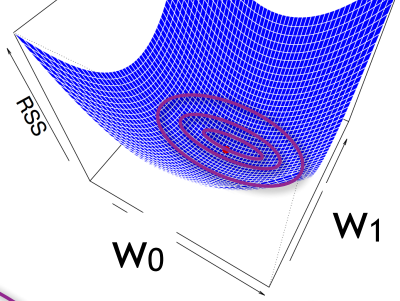
$$\|\mathbf{w}\|_1 = \sum_{i=0}^{d}|w_i|$$

L_2-norm

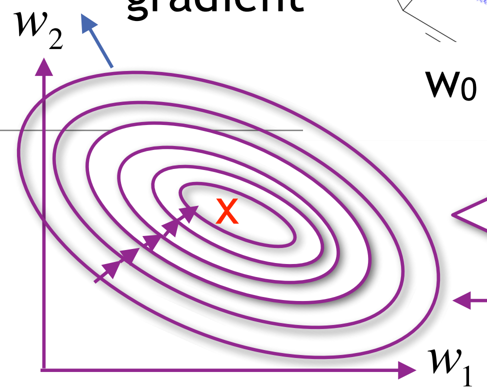
$$\|\mathbf{w}\|_2 = \sqrt{\sum_{i=0}^{d}(w_i)^2}$$

Geometric Intuition

Level curve/contour line/ Isoline of $||X\mathbf{w}-\mathbf{y}||^2 = c$



gradient



contour plot

$E_{in}(\mathbf{w}) = c$

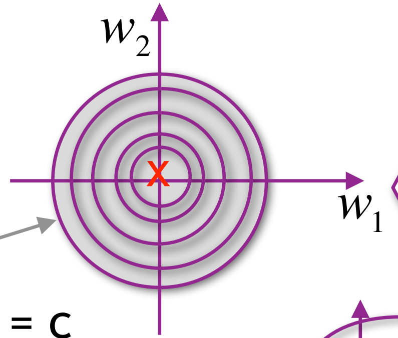
- Looking at the contour plot of RSS

Ellipse

$$\mathrm{E}_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2 = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \mathbf{w}^T\mathbf{x})^2$$

- Looking at the contour plot of the L_2 norm

$$\left\|\mathbf{w}_{1:d}\right\|_2^2 = \sum_{i=1}^{d} w_i^2$$

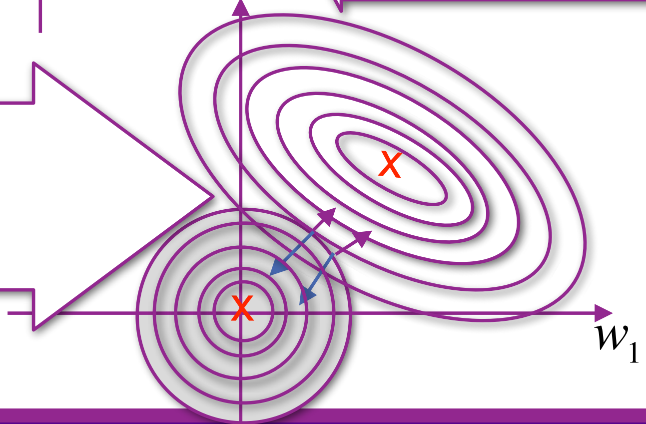


sphere (hypersphere in higher dimensions)

Level curve/contour line/ Isoline of $||\mathbf{w}_{1:d}||^2 = c$

- Looking at $\mathrm{E}_{\text{ridge}}(\mathbf{w}) = \mathrm{E}_{\text{in}}(\mathbf{w}) + \lambda\left\|\mathbf{w}_{1:d}\right\|_2^2$

For each λ there is a point which minimizes $E_{\text{ridge}}(\mathbf{w})$



Remember how in Multiple linear regression we found **w** to minimize E_{in}

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}((\mathbf{w}^T x^{(i)}) - y^{(i)})^2 \quad = \frac{1}{2N} RSS(\mathbf{w})$$

Remember: Multiple linear regression Closed form solution

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}((\mathbf{w}^T x^{(i)}) - y^{(i)})^2 = \frac{1}{2N} RSS(\mathbf{w})$$

An alternative objective function

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y})^2$$
$$= \frac{1}{N} RSS(\mathbf{w})$$

Goal find $\mathbf{w}_{\text{lin}}$ such that $\nabla J(\mathbf{w}) = \mathbf{0}$ (same $\mathbf{w}$ that makes RSS($\mathbf{w}$) $= \mathbf{0}$ and $\nabla E_{\text{in}}(\mathbf{w}) = \mathbf{0}$

$$\nabla J(\mathbf{w}) = \frac{1}{N} X^T (X\mathbf{w} - \mathbf{y}) = \frac{1}{N}(X^T X\mathbf{w} - X^T \mathbf{y})$$

Setting $\nabla J(\mathbf{w}) = \frac{1}{N}(X^T X\mathbf{w} - X^T \mathbf{y}) = \mathbf{0}$

Results in: $X^T X\mathbf{w} = X^T \mathbf{y}$

Thus $\mathbf{w}_{\text{lin}} = \underbrace{\left(X^T X\right)^{-1} X^T}_{\substack{\text{pseudoinverse} \\ \text{left inverse}}} \mathbf{y}$

We added ‘lin’ to $\mathbf{w}$ to specify it was linear regression

$X^T X$ is a $d \times d$ matrix
$(X^T X)^{-1}$ is a $d \times d$ matrix
$(X^T X)^{-1} X^T$ is a $d \times N$ matrix
$(X^T X)^{-1} X^T \mathbf{y}$ is $d \times 1$

$$\nabla J(\mathbf{w}) = \frac{1}{N}(X^T X\mathbf{w} - X^T \mathbf{y})$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{2}{N}(X^T X\mathbf{w} - X^T \mathbf{y})$$

Finding $\mathbf{w}$ to minimize E_{ridge}

$$E_{ridge}(\mathbf{w}) = E_{in}(\mathbf{w}) + \lambda(w_1^2 + w_2^2 + \cdots + w_d^2)$$

Goal find $\mathbf{w}_{ridge}$ such that $\nabla E_{ridge}(\mathbf{w}) = \mathbf{0}$

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y})^2$$

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}(y^{(i)} - \hat{y})^2$$

$$||\mathbf{v}||^2 = \mathbf{v}^T\mathbf{v} = v_1^2 + v_2^2 + \cdots + v_d^2$$

$$\nabla E_{ridge}(\mathbf{w}) = \nabla E_{in}(\mathbf{w}) + \nabla \lambda (\mathbf{w}_{1:d})^T \mathbf{w}_{1:d}$$

$$\nabla E_{in}(\mathbf{w}) = \frac{2}{N}(X^T X\mathbf{w} - X^T\mathbf{y})$$

$$(\mathbf{w}_{1:d})^T\mathbf{w}_{1:d} = w_1^2 + w_2^2 + \cdots + w_j^2 + \cdots + w_d^2$$

$$\frac{\partial(\mathbf{w}_{1:d})^T\mathbf{w}_{1:d}}{\partial w_j} = 2w_j$$

$$\nabla(\mathbf{w}_{1:d})^T\mathbf{w}_{1:d} = 2\mathbf{w}_{1:d}$$

$$\nabla\lambda(\mathbf{w}_{1:d})^T\mathbf{w}_{1:d} = 2\lambda\mathbf{w}_{1:d}$$

Using this form so E_{ridge} to easier to work with

$$2\lambda\mathbf{w}' = 2\lambda\begin{bmatrix} 0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} = 2\lambda\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} = 2\lambda I'\mathbf{w}$$

$$\nabla E_{ridge}(\mathbf{w}) = \frac{2}{N}(X^T X\mathbf{w} - X^T\mathbf{y}) + 2\lambda I'\mathbf{w}$$

Finding **w** to minimize E_{ridge}

$$\mathrm{E}_{\text{ridge}}(\mathbf{w}) = \mathrm{E}_{\text{in}}(\mathbf{w}) + \lambda(w_1^2 + w_2^2 + \cdots + w_d^2)$$

Goal find $\mathbf{w}_{\text{ridge}}$ such that $\nabla E_{\text{ridge}}(\mathbf{w}) = \mathbf{0}$

Setting $\nabla E_{\text{ridge}}(\mathbf{w}) = \frac{2}{N}(X^T X\mathbf{w} - X^T \mathbf{y}) + 2\lambda \mathrm{I}' \mathbf{w} \quad = \mathbf{0}$

Results in:

$$X^T X\mathbf{w} + N\lambda \mathrm{I}' \mathbf{w} = X^T \mathbf{y}$$
$$(X^T X + N\lambda \mathrm{I}')\mathbf{w} = X^T \mathbf{y}$$
$$\mathbf{w} = (X^T X + N\lambda \mathrm{I}')^{-1} X^T \mathbf{y}$$

Thus $\mathbf{w}_{\text{ridge}} = \underbrace{(X^T X + N\lambda \mathrm{I}')^{-1} X^T \mathbf{y}}_{\text{Closed form solution!}}$

Please note that we could have written the regularization as $\lambda/N\ \mathbf{w}^T\mathbf{w}$ since the need for regularization decreases as the number of training examples increases.

In this case we are minimizing Ein(w) + $\lambda/N\ \mathbf{w}^T\mathbf{w}$

and the closed form solution becomes $(X^TX + \lambda I')^{-1}X^T\mathbf{y}$

$$\mathbf{w}_{\text{lin}} = \left(X^T X\right)^{-1} X^T \mathbf{y}$$

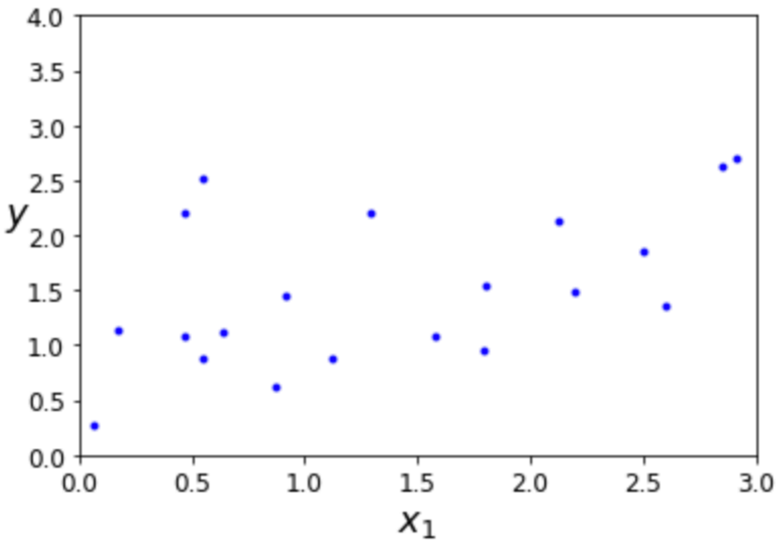
Example

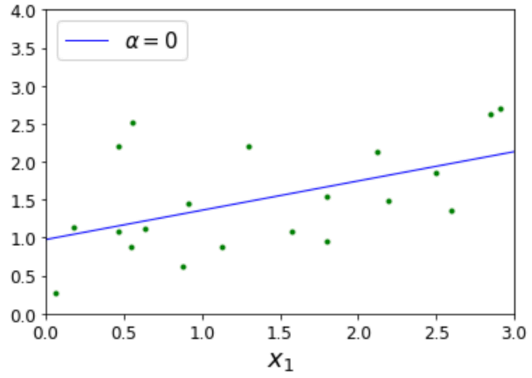
X

```
[[1.    1.12]
 [1.    2.85]
 [1.    2.2 ]
 [1.    1.8 ]
 [1.    0.47]
 [1.    0.47]
 [1.    0.17]
 [1.    2.6 ]
 [1.    1.8 ]
 [1.    2.12]
 [1.    0.06]
 [1.    2.91]
 [1.    2.5 ]
 [1.    0.64]
 [1.    0.55]
 [1.    0.55]
 [1.    0.91]
 [1.    1.57]
 [1.    1.3 ]
 [1.    0.87]]
```

y

```
[[0.89]
 [2.64]
 [1.49]
 [0.96]
 [2.21]
 [1.08]
 [1.13]
 [1.35]
 [1.54]
 [2.14]
 [0.26]
 [2.71]
 [1.85]
 [1.12]
 [0.87]
 [2.51]
 [1.45]
 [1.08]
 [2.2 ]
 [0.62]]
```

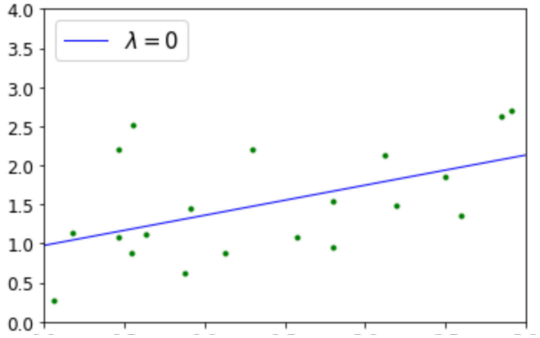




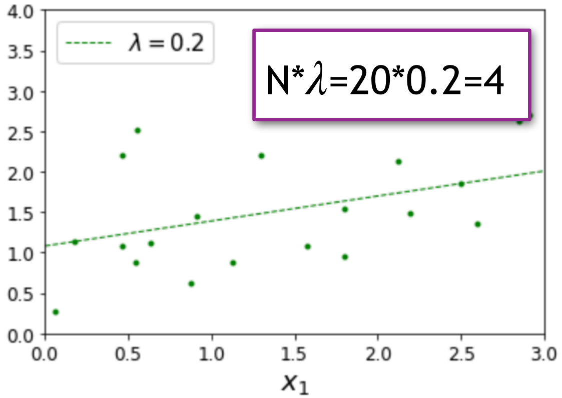
$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \left[\begin{bmatrix} 1 & 1.12 \\ 1 & 2.85 \\ 1 & 2.2 \\ 1 & 1.8 \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1.12 & 2.85 & 2.2 & 1.8 & \cdots \end{bmatrix} \right]^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1.12 & 2.85 & 2.2 & 1.8 & \cdots \end{bmatrix} \begin{bmatrix} 0.89 \\ 2.64 \\ 1.49 \\ 0.96 \\ \vdots \end{bmatrix} = \begin{bmatrix} 0.98 \\ 0.39 \end{bmatrix}$$

$$\mathbf{w}_{\text{ridge}} = (X^T X + N\lambda \mathbf{I}')^{-1} X^T \mathbf{y}$$

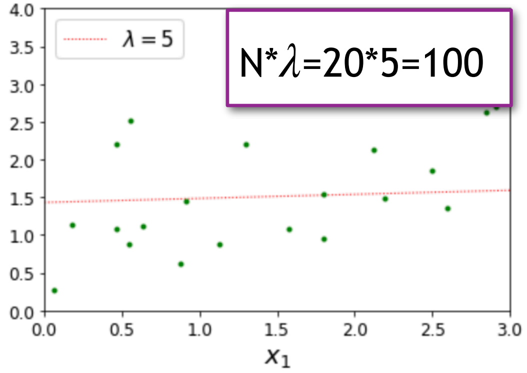
Example



$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \left[\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1.12 & 2.85 & 2.2 & 1.8 & \cdots \end{bmatrix} \begin{bmatrix} 1 & 1.12 \\ 1 & 2.85 \\ 1 & 2.2 \\ 1 & 1.8 \\ \vdots & \vdots \end{bmatrix} \right]^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1.12 & 2.85 & 2.2 & 1.8 & \cdots \end{bmatrix} \begin{bmatrix} 0.89 \\ 2.64 \\ 1.49 \\ 0.96 \\ \vdots \end{bmatrix} = \begin{bmatrix} 0.98 \\ 0.39 \end{bmatrix}$$



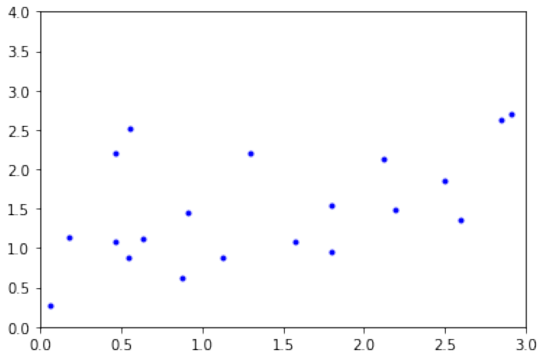
$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \left[\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1.12 & 2.85 & 2.2 & 1.8 & \cdots \end{bmatrix} \begin{bmatrix} 1 & 1.12 \\ 1 & 2.85 \\ 1 & 2.2 \\ 1 & 1.8 \\ \vdots & \vdots \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix} \right]^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1.12 & 2.85 & 2.2 & 1.8 & \cdots \end{bmatrix} \begin{bmatrix} 0.89 \\ 2.64 \\ 1.49 \\ 0.96 \\ \vdots \end{bmatrix} = \begin{bmatrix} 1.08 \\ 0.31 \end{bmatrix}$$

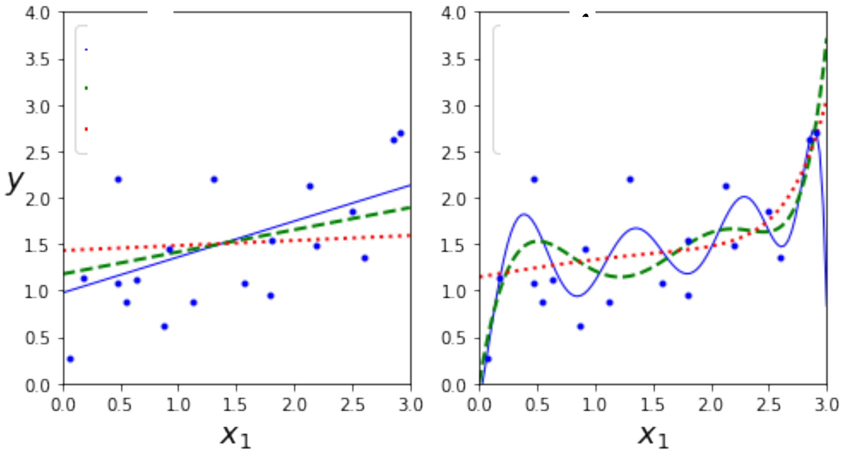


$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \left[\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1.12 & 2.85 & 2.2 & 1.8 & \cdots \end{bmatrix} \begin{bmatrix} 1 & 1.12 \\ 1 & 2.85 \\ 1 & 2.2 \\ 1 & 1.8 \\ \vdots & \vdots \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix} \right]^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1.12 & 2.85 & 2.2 & 1.8 & \cdots \end{bmatrix} \begin{bmatrix} 0.89 \\ 2.64 \\ 1.49 \\ 0.96 \\ \vdots \end{bmatrix} = \begin{bmatrix} 1.43 \\ 0.05 \end{bmatrix}$$

Sklearn Regularization

The true function is linear f(x) = 1 + 0.5 * X + noise





Example from page 129-131 in Machine Learning with Scikit-Learn and TensorFlow

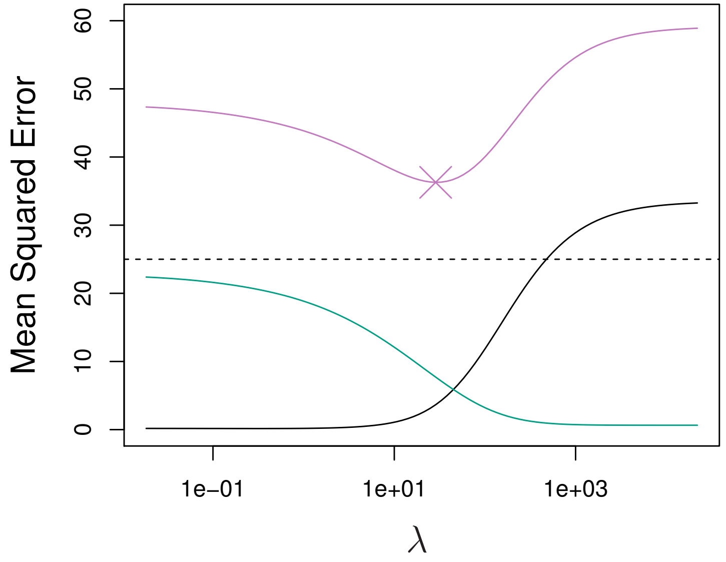


Figure 6.5 from ISLR
The data was synthetic data
Purple - test MSE
Green - variance
Black - bias (aka bias^2)

Lasso - *L*east *A*bsolute *S*election and *S*hrinkage *O*perator

PERFORM VARIABLE SELECTION

MORE EFFICIENT TO HAVE LESS FEATURES

INTERPRETABILITY

Approach taken from https://courses.cs.washington.edu/courses/cse446/17wi/slides/lasso-annotated.pdf

Too many features!

In some datasets, only a subset of the features contribute to the answer

Removing features reduces variance

Removing features makes understanding the coefficients easier

How could we choose which features to use?

- Run the algorithm on all possible subsets of the features
- Remove features one by one are rerun the algorithm - seeing if it get worse
- Start with one feature and slowly add new features if “they help”

We can use LASSO regularization to reduce the number of features d = # features. Note we don't want to restrict w_0. Many approaches to this issue are possible. We will leave w_0 out of the penalty term. Note: Scaling the features is suggested

Lasso Regression

If λ=0 then $\mathbf{w}_{\text{lasso}} = \mathbf{w}_{\text{lin}}$

$\mathbf{w}_{\text{lasso}}$ = the best parameters for LASSO

$\mathbf{w}_{\text{lin}}$ = the best parameters for least squares cost

If $\lambda$is very large then

$w_i \sim 0$ for all $i > 0$

If λ is a constant then

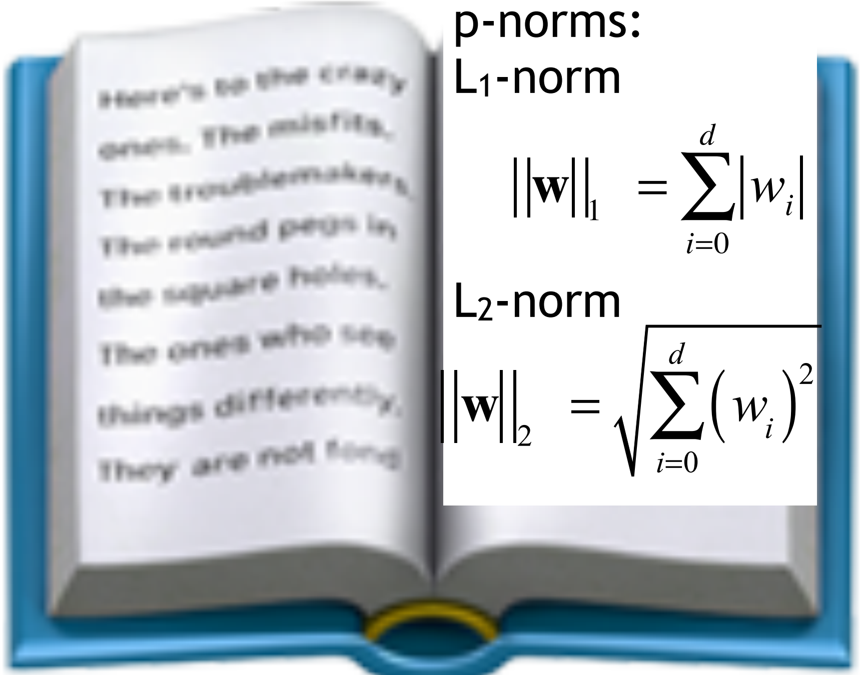
$0 \le \|w_{\text{lasso}}\|_1 \le \|w_{\text{lin}}\|_1$

- Tuning parameter λ to balance fit and number of parameters

$$\mathrm{E}_{\text{aug}}(\mathbf{w}) = \mathrm{E}_{\text{in}}(\mathbf{w}) + \text{penalty for complex models}$$

$$\mathrm{E}_{\text{lasso}}(\mathbf{w}) = \mathrm{E}_{\text{in}}(\mathbf{w}) + \lambda\left(|\cancel{w_0}| + |w_1| + |w_2| + \cdots + |w_d|\right)$$

- λ controls the model complexity
  - Large λ
    - high bias, low variance
  - small λ
    - low bias, high variance



p-norms:

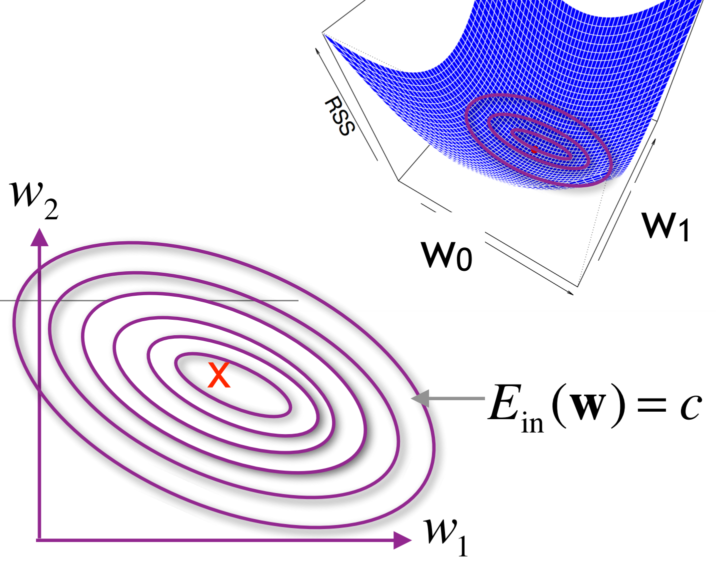
L_1-norm

$$\|\mathbf{w}\|_1 = \sum_{i=0}^{d} |w_i|$$

L_2-norm

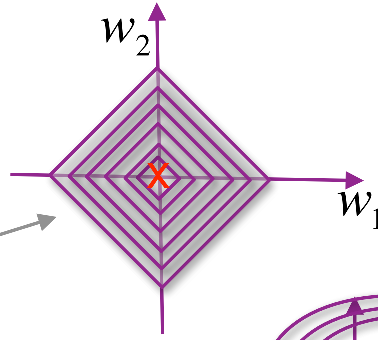
$$\|\mathbf{w}\|_2 = \sqrt{\sum_{i=0}^{d} (w_i)^2}$$

Geometric Intuition



- Looking at the contour plot of RSS

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2 = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \mathbf{w}^T\mathbf{x})^2$$



cross-polytope

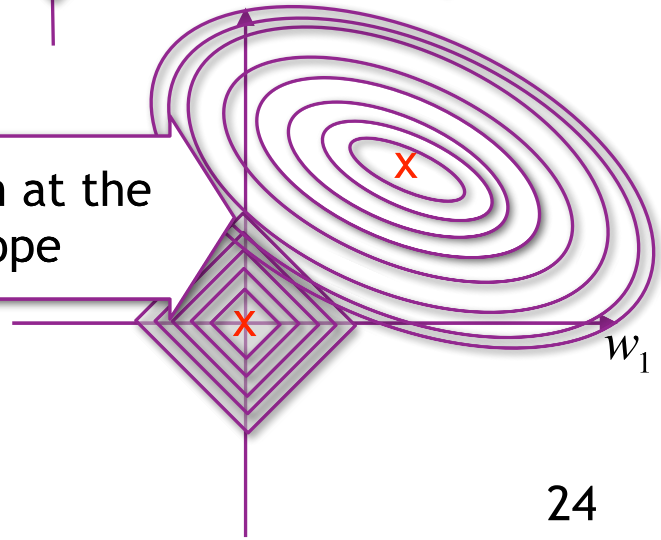
- Looking at the contour plot of the L_1 norm

$$\|\mathbf{w}_{1:d}\|_1 = \sum_{i=1}^{d}|w_i|$$

Level curve/contour line/Isoline

$$\|\mathbf{w}_{1:d}\|_1 = \sum_{i=1}^{d}|w_i| = s$$

- Looking at $\mathrm{E}_{\text{lasso}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \lambda\|\mathbf{w}_{1:d}\|_1$



The min is often at the tip of the polytope

https://en.wikipedia.org/wiki/Cross-polytope

Uh Oh, no closed form solution for minimizing

$$E_{\text{lasso}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \lambda \|\mathbf{w}_{1:d}\|_1$$

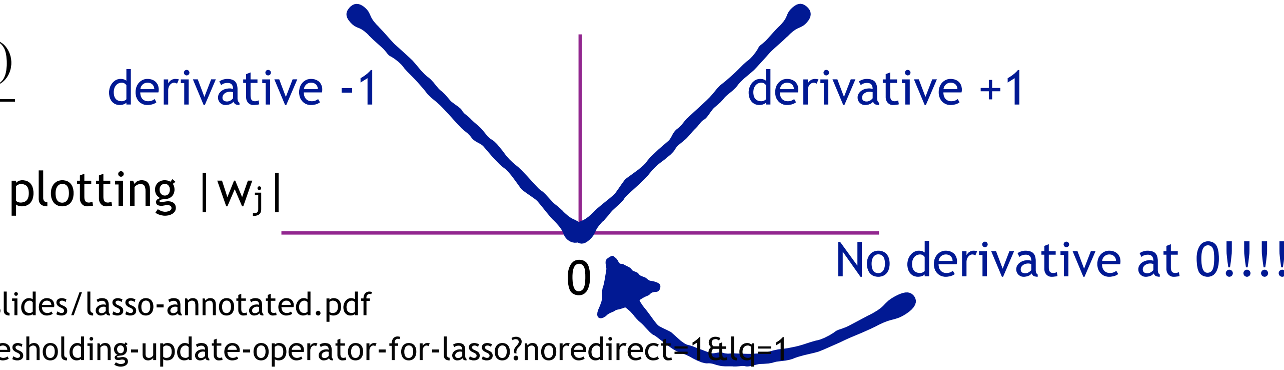
- $E_{\text{lasso}}(\mathbf{w})$ is convex, but we cannot take the derivative and set it to zero to find the optimal **w**
- It is possible to optimize the j[th] coefficient while the others remain fixed

$$w_j^* = \arg\min_z E_{\text{lasso}}(\mathbf{w} + z\mathbf{e_j})$$

$\mathbf{e_j}$ is the jth unit vector

$$\frac{\partial E_{\text{lasso}}(\mathbf{w})}{\partial w_j} = \frac{2}{N}\sum_{i=1}^{N}(y_i - (w_0 x_0^{(i)} + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_j x_j^{(i)} + \cdots + w_d x_d^{(i)}))(-x_j^{(i)}) + \lambda \frac{\partial \|\mathbf{w}_{1:d}\|_1}{\partial w_j}$$

$$\frac{\partial \lambda \|\mathbf{w}_{1:d}\|_1}{\partial w_j} = \lambda \frac{\partial \|\mathbf{w}_{1:d}\|_1}{\partial w_j} = \lambda \frac{\partial(|w_1| + \cdots + |w_d|)}{\partial w_j} = \lambda \frac{\partial(|w_j|)}{\partial w_j}$$



Approach taken from:https://courses.cs.washington.edu/courses/cse546/14au/slides/lasso-annotated.pdf
https://stats.stackexchange.com/questions/123672/coordinate-descent-soft-thresholding-update-operator-for-lasso?noredirect=1&lq=1

https://support.sas.com/resources/papers/proceedings15/3297-2015.pdf
http://www.math.mcgill.ca/yyang/regression/extra/lasso.pdf
https://en.wikipedia.org/wiki/Lasso_(statistics)
http://myweb.uiowa.edu/pbreheny/7600/s16/notes/2-15.pdf

Ridge and LASSO

Overview

Suggested to **scale** features before performing ridge regression or lasso regression.

Ridge regression has a **closed form solution**.
Ridge regression shrinks coefficients towards zero.

Lasso does not have a closed form solution.
Lasso performs **feature selection** as well as **parameter estimation**.

For both lasso and ridge regression, the tuning parameter λ **controls the strength of the regularization**.

Both ridge and lasso **increase bias, but decrease variance**

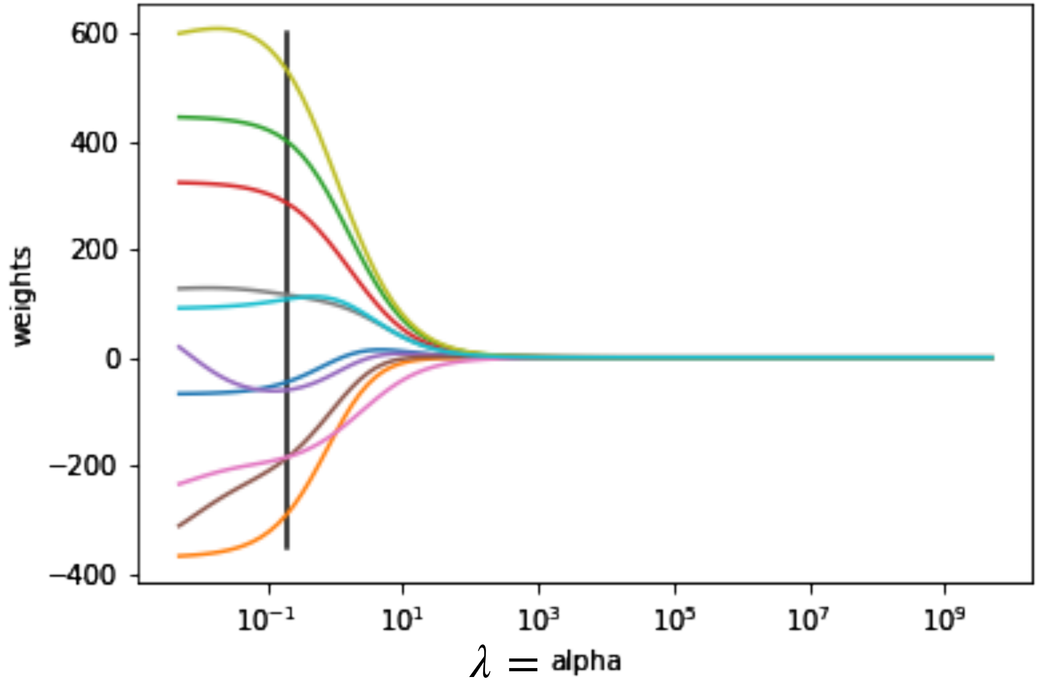
Diabetes dataset

Pair share: which value of λ causes the model to have more bias? $\lambda = \frac{1}{10}$ or $\lambda = \frac{1}{100}$

How changes in λ affect the optimal coefficients

RIDGE

$$E_{\text{ridge}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \lambda(w_1^2 + w_2^2 + w_3^2 + \cdots + w_d^2)$$

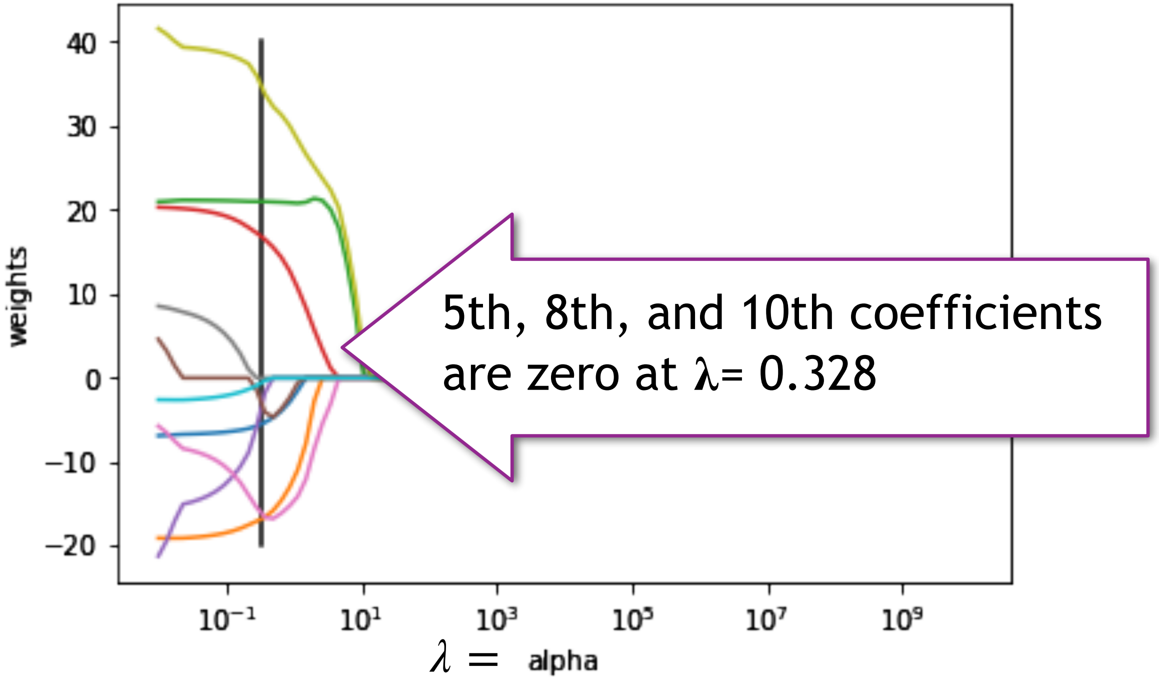



```
The best λ = alpha: 0.188
```

$$E_{\text{ridge}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + 0.188(w_1^2 + w_2^2 + w_3^2 + \cdots + w_d^2)$$

LASSO

$$E_{\text{lasso}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \lambda(|w_1| + |w_2| + |w_3| + \cdots + |w_d|)$$




```
The best λ = alpha:  0.328
```

$$E_{\text{lasso}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + 0.328(|w_1| + |w_2| + |w_3| + \cdots + |w_d|)$$

Slides not covered in class

Pair share

- You are choosing amongst 6 different transformations of the data (one is the identity transformation), and for each, you will try 8 different values for λ.
- How many times do you train a model?

A. 1
B. 6
C. 8
D. 14
E. 36
F. 40
G. 45
H. 48
I. More than 50

Standardization in Scikit-Learn

Many algorithm (such as ridge regression and lasso regression) require the data to be standardized t
work correctly.

We want to apply the same scaling we did on the test data to future examples…

```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train) # zero mean and unit variance
# Then later when want to use our classifier on new data, we scale the new data the same way we scaled the training
X_test_scaled = scaler.transform(X_test)
```

More information from http://scikit-learn.org/stable/modules/preprocessing.html
Many other variations of scaling can be found here

Ridge Regression in Scikit-Learn

auto is default

solver : {*‘auto’, ‘cholesky’, ‘sag”,…*}

closed form solution we proved

stochastic gradient descent (typically faster when when X is large). Needs scaled data!

```
from sklearn.linear_model import Ridge
import numpy as np

clf = Ridge(alpha=1.0, solver = ‘cholesky’)
clf.fit(X, y)
```

```
print(clf.coef_)
print(clf.intercept_)
```

Some the methods

fit(X, y)	Fit Ridge regression model.
predict(X)	Predict class labels for samples in X.
score(X, y[, sample_weight])	Return the coefficient of determination of the prediction

Information from http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

Lasso Regression in Scikit-Learn

```
from sklearn import linear_model
clf = linear_model.Lasso(alpha=0.1)
clf.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])

print(clf.coef_)
print(clf.intercept_)
```

[0.85   0.]

0.150

Some of the methods

fit(X, y)	Fit Lasso regression model.
predict(X)	Predict class labels for samples in X.
score(X, y[, sample_weight])	Return the coefficient of determination of the prediction

Information from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html