

# Ridge Regression:

Regulating overfitting when  
using many features

Emily Fox & Carlos Guestrin

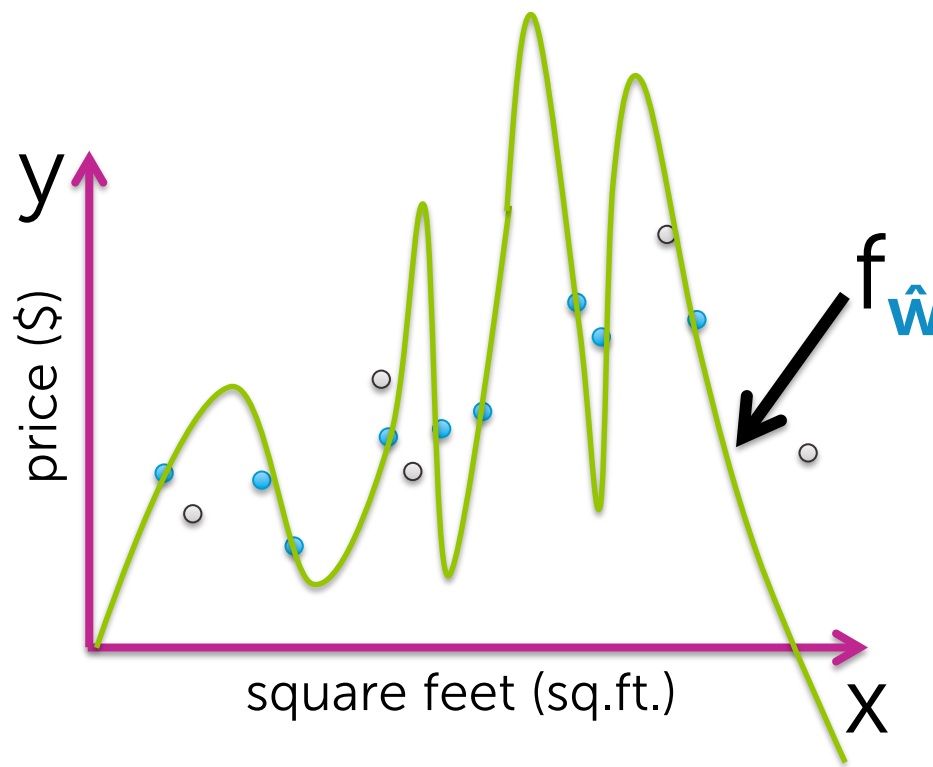
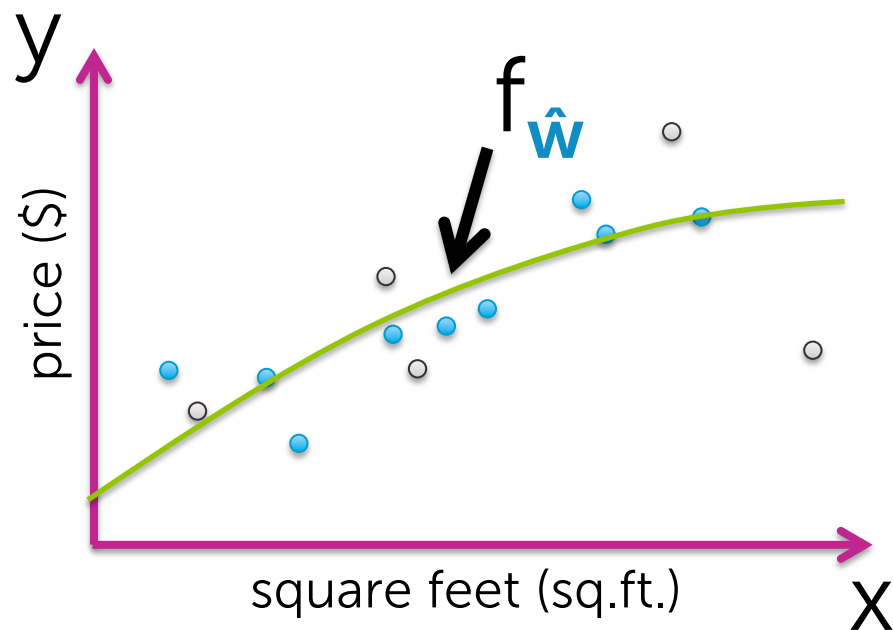
Machine Learning Specialization

University of Washington

# Overfitting of polynomial regression

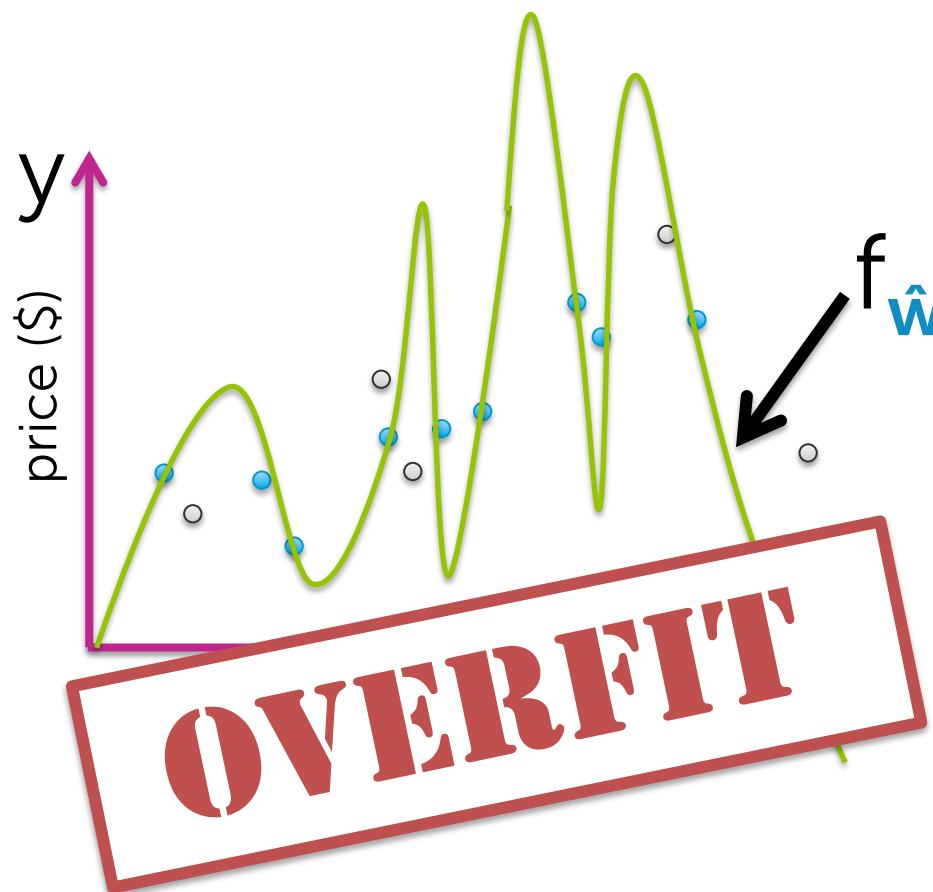
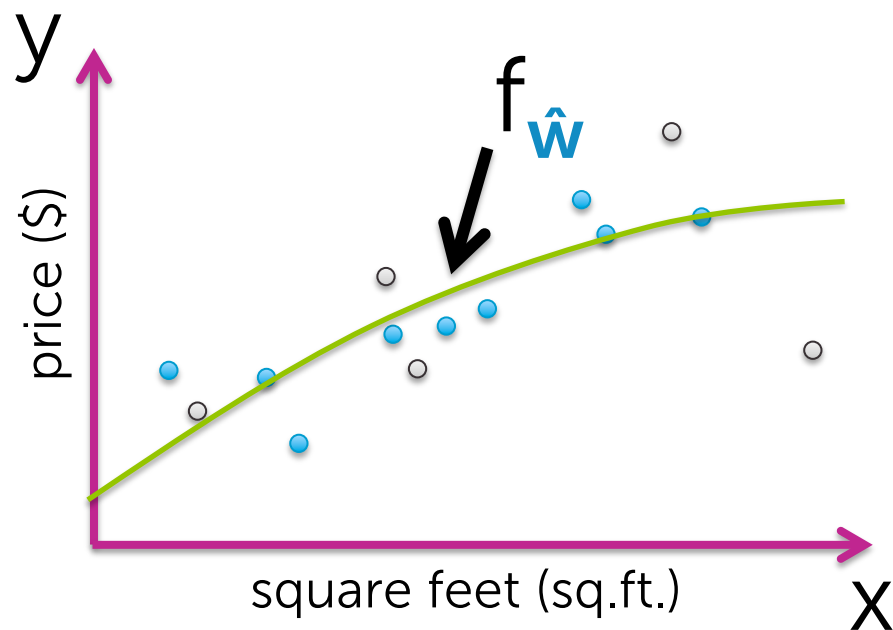
# Flexibility of high-order polynomials

$$y_i = w_0 + w_1 x_i + w_2 x_i^2 + \dots + w_p x_i^p + \varepsilon_i$$



# Flexibility of high-order polynomials

$$y_i = w_0 + w_1 x_i + w_2 x_i^2 + \dots + w_p x_i^p + \varepsilon_i$$



# Symptom of overfitting

Often, overfitting associated with very large estimated parameters  $\hat{\mathbf{w}}$

# Overfitting of linear regression models more generically

# Overfitting with many features

Not unique to polynomial regression,  
but also if **lots of inputs** (**d large**)

Or, generically,  
**lots of features** (**D large**)

$$y_i = \sum_{j=0}^D \mathbf{w}_j h_j(\mathbf{x}_i) + \epsilon_i$$

- Square feet
- # bathrooms
- # bedrooms
- Lot size
- Year built
- ...

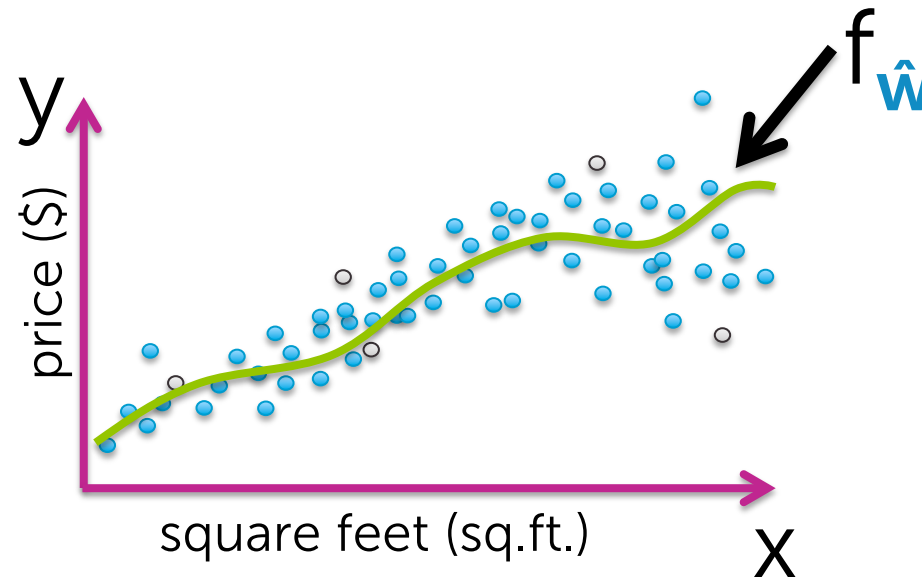
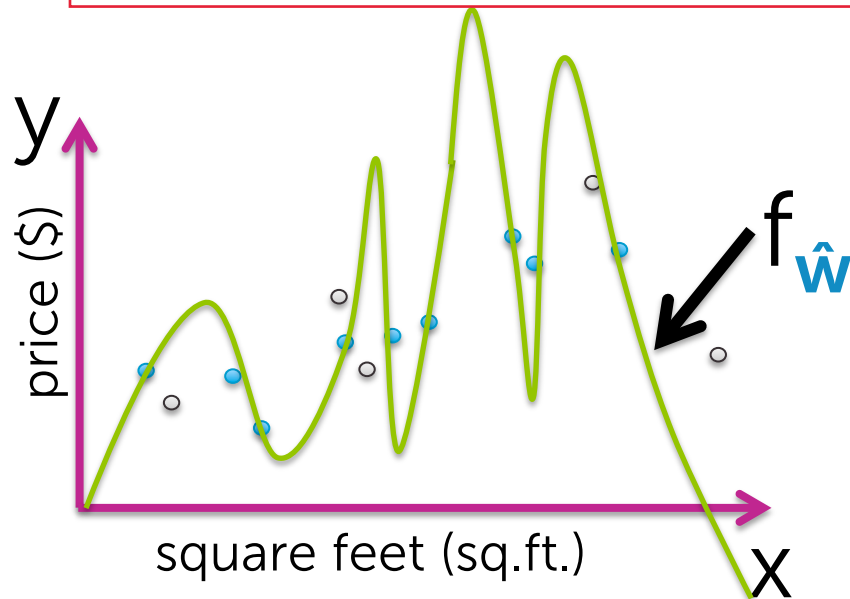
# How does # of observations influence overfitting?

Few observations ( $N$  small)

→ rapidly overfit as model complexity increases

Many observations ( $N$  very large)

→ harder to overfit



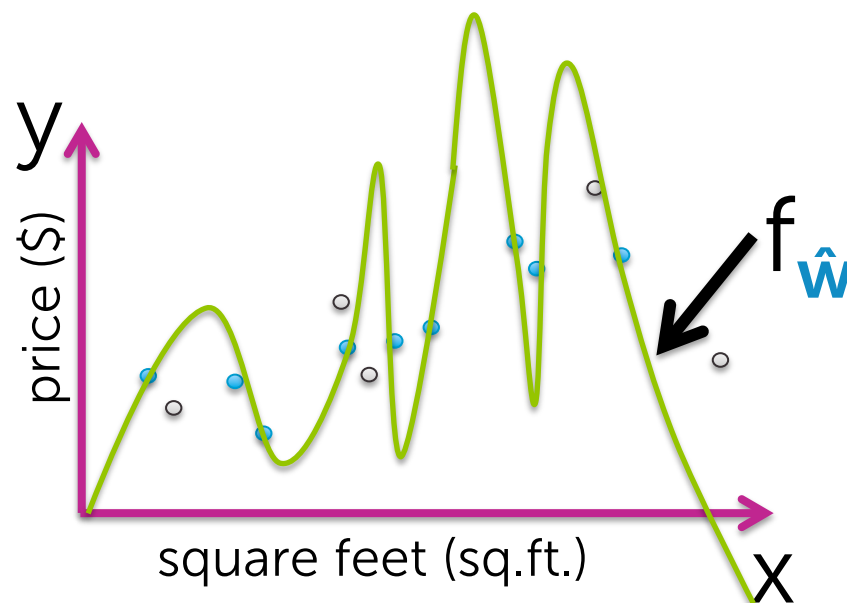


# How does # of inputs influence overfitting?

1 input (e.g., sq.ft.):

Data must include representative examples of all possible (sq.ft., \$) pairs to avoid overfitting

**HARD**

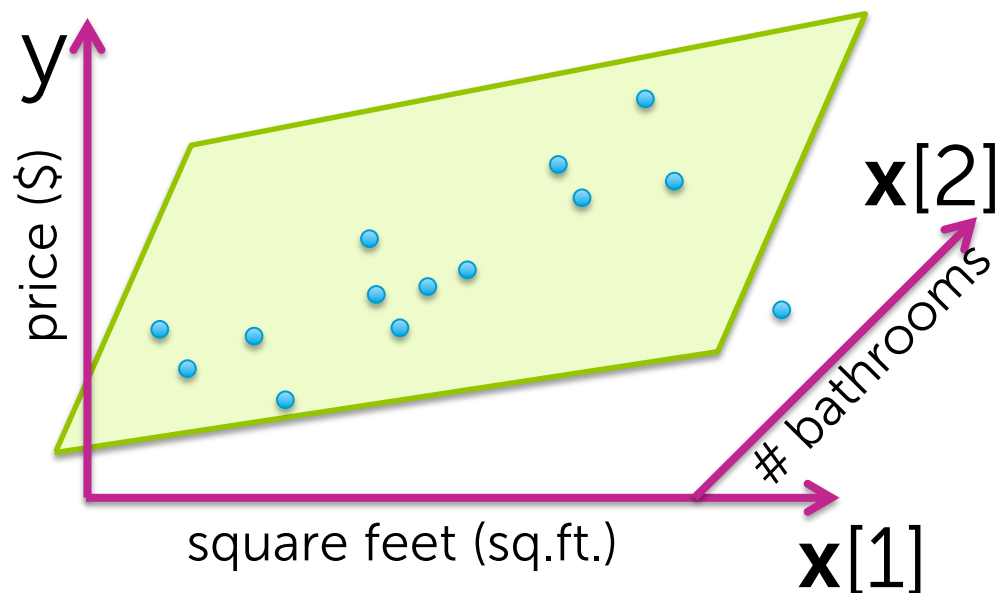


# How does # of inputs influence overfitting?

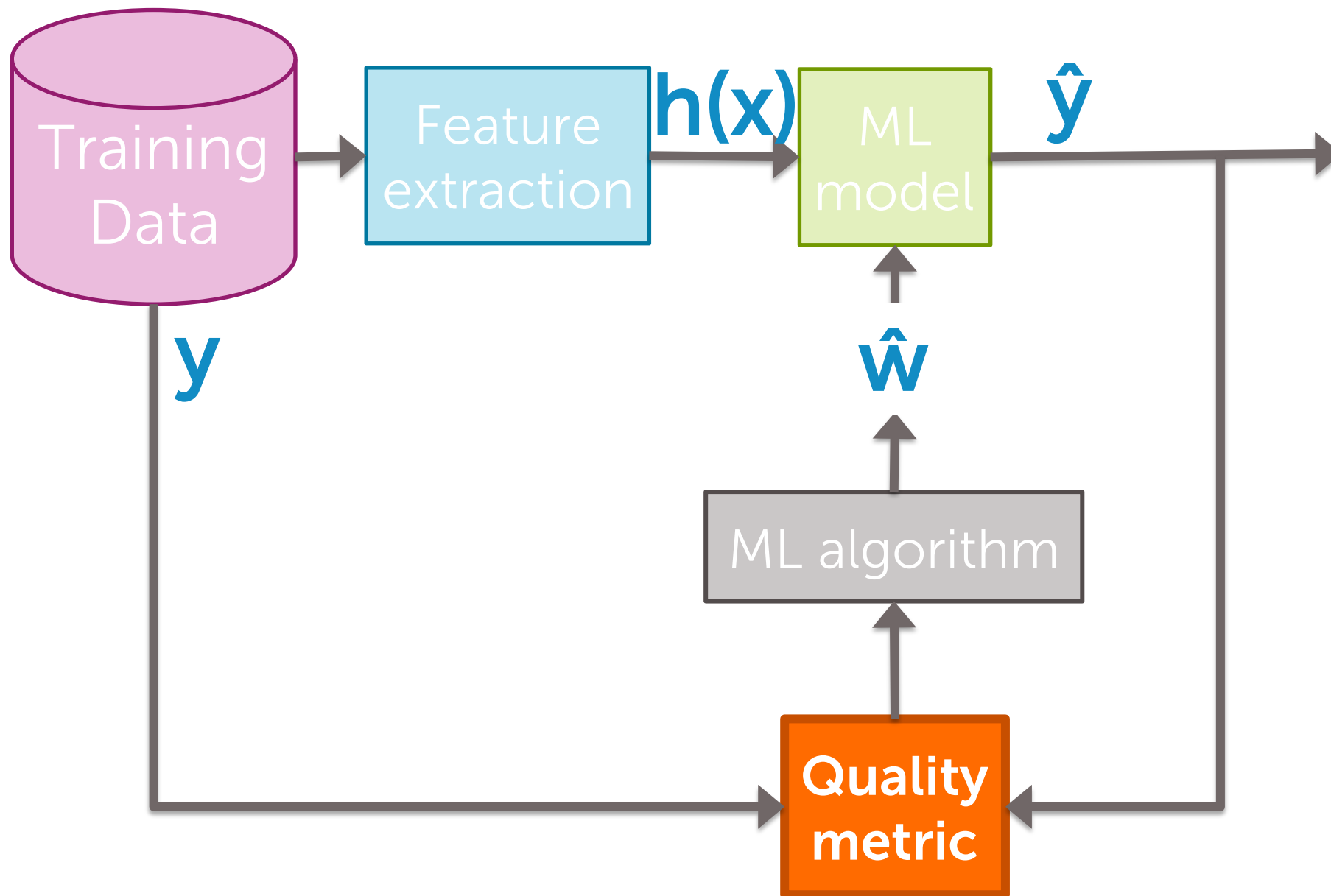
**d inputs** (e.g., sq.ft., #bath, #bed, lot size, year,...):

Data must include examples of all possible (sq.ft., #bath, #bed, lot size, year,..., \$) combos to avoid overfitting

**MUCH!!!  
HARDER**



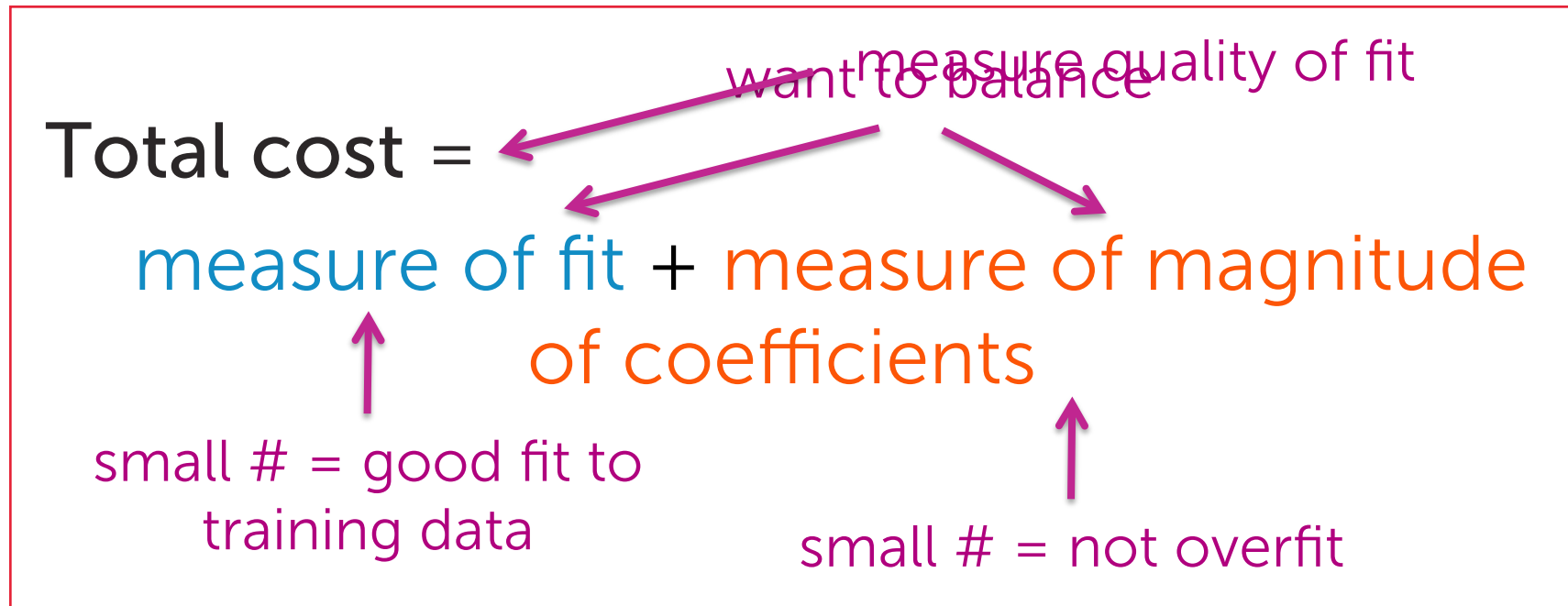
Adding term to cost-of-fit  
to prefer small coefficients



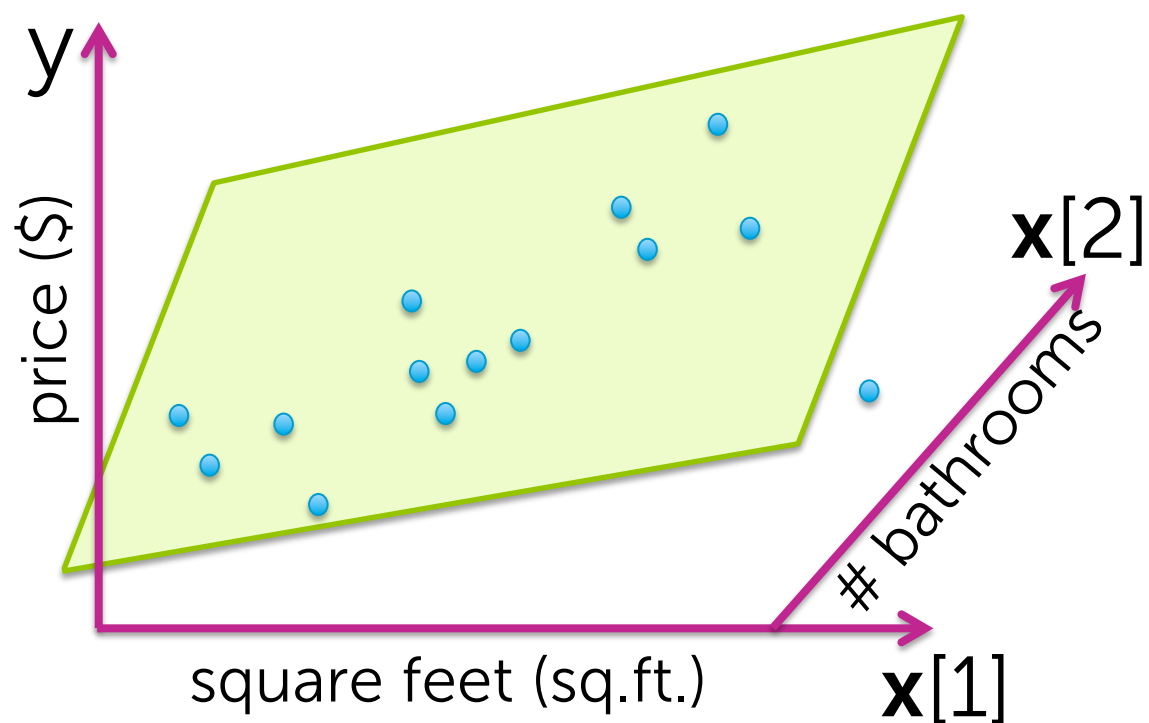
# Desired total cost format

Want to balance:

- i. How well function fits data
- ii. Magnitude of coefficients



# Measure of fit to training data



$$\begin{aligned} \text{RSS}(\mathbf{w}) &= \sum_{i=1}^N (y_i - h(\mathbf{x}_i)^T \mathbf{w})^2 \\ &= \sum_{i=1}^N (y_i - \hat{y}_i(\mathbf{w}))^2 \end{aligned}$$

*pred. value using  $\mathbf{w}$*

small RSS  $\rightarrow$  model fitting training data well

# Measure of magnitude of regression coefficient

What summary # is indicative of size of regression coefficients?

- Sum?  $w_0 = 1,527,301$   $w_1 = -1,605,253$   
 $w_0 + w_1 = \text{small \#}$

- Sum of absolute value?  
 $|w_0| + |w_1| + \dots + |w_D| = \sum_{j=0}^D |w_j| \triangleq \|w\|_1$   $L_1$  norm ... discuss more in next module
- Sum of squares ( $L_2$  norm)  
 $w_0^2 + w_1^2 + \dots + w_D^2 = \sum_{j=0}^D w_j^2 \triangleq \|w\|_2^2$   $L_2$  norm ... focus of this module

# Consider specific total cost

Total cost =

measure of fit + measure of magnitude  
of coefficients



# Consider specific total cost

Total cost =

measure of fit + measure of magnitude  
of coefficients

RSS(**w**)

$\|\mathbf{w}\|_2^2$

# Consider resulting objective

What if  $\hat{\mathbf{w}}$  selected to minimize

$$\text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

tuning parameter = balance of fit and magnitude

If  $\lambda = 0$ :

reduces to minimizing  $\text{RSS}(\mathbf{w})$ , as before (old solution)  $\rightarrow \hat{\mathbf{w}}^{\text{LS}} \leftarrow \text{least squares}$

If  $\lambda = \infty$ :

For solutions where  $\hat{\mathbf{w}} \neq 0$ , then total cost is  $\infty$

If  $\hat{\mathbf{w}} = 0$ , then total cost =  $\text{RSS}(0)$   $\rightarrow$  solution is  $\hat{\mathbf{w}} = 0$

If  $\lambda$  in between: Then  $0 \leq \|\hat{\mathbf{w}}\|_2^2 \leq \|\hat{\mathbf{w}}^{\text{LS}}\|_2^2$

# Consider resulting objective

What if  $\hat{\mathbf{w}}$  selected to minimize

$$\text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

 tuning parameter = balance of fit and magnitude

Ridge regression  
(a.k.a  $L_2$  regularization)

# Bias-variance tradeoff

Large  $\lambda$ :

high bias, low variance

(e.g.,  $\hat{\mathbf{w}} = 0$  for  $\lambda = \infty$ )

In essence,  $\lambda$   
controls model  
complexity

Small  $\lambda$ :

low bias, high variance

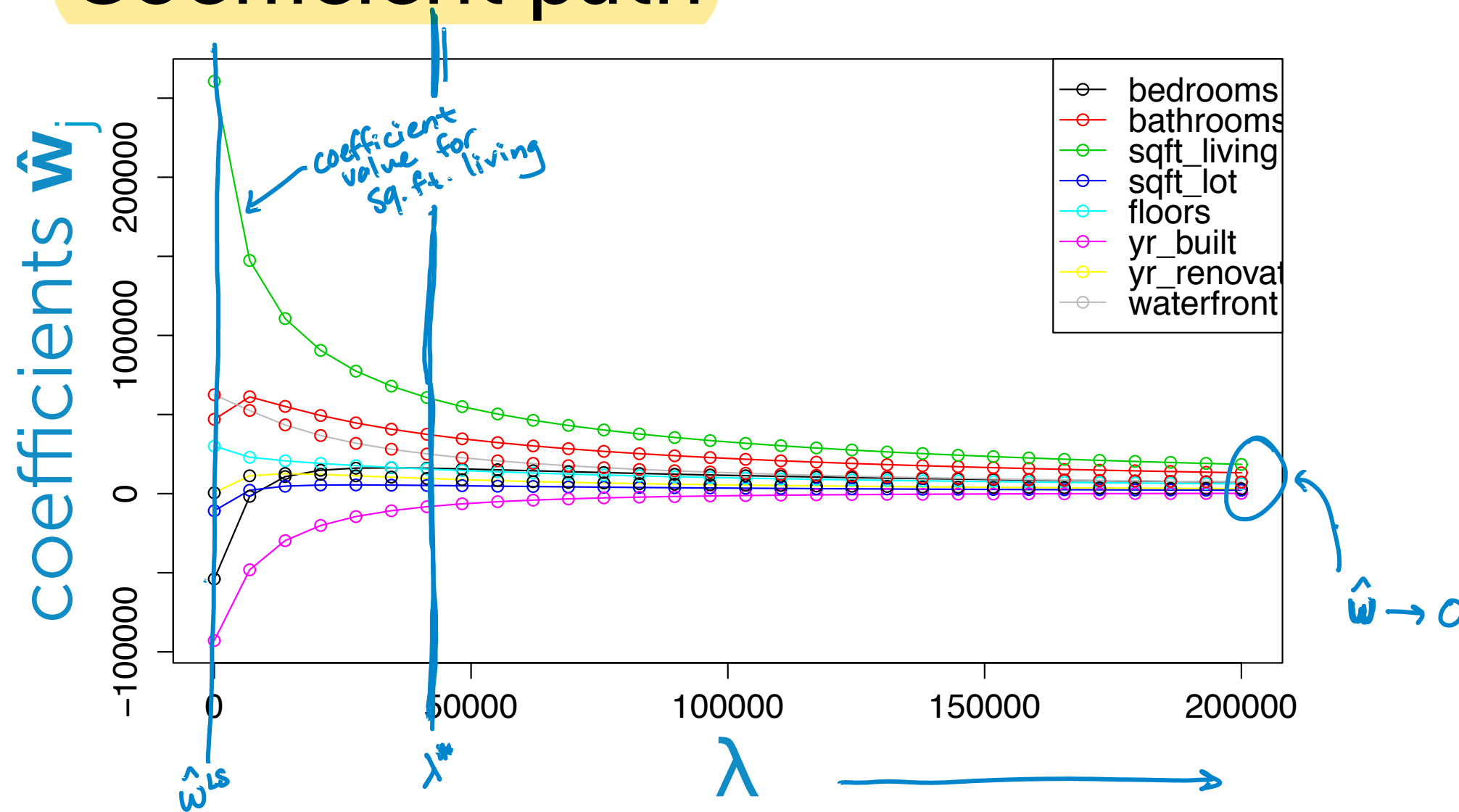
(e.g., standard least squares (RSS) fit of  
high-order polynomial for  $\lambda = 0$ )

# Revisit polynomial fit demo

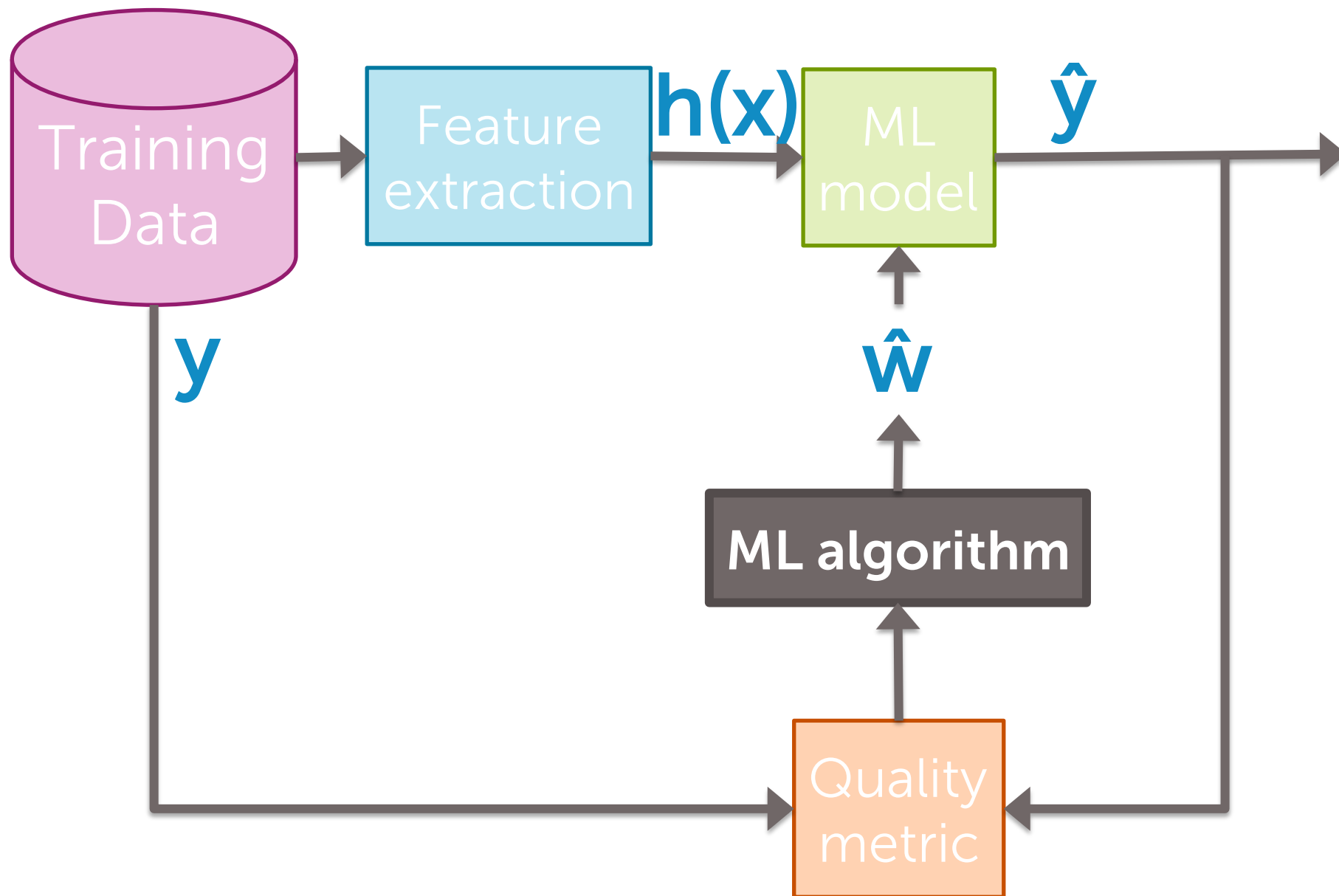
What happens if we refit our high-order polynomial, but now using **ridge regression**?

Will consider a few settings of  $\lambda$  ...

# Coefficient path



# Fitting the ridge regression model (for given $\lambda$ value)





# Step 1:

## Rewrite total cost in matrix notation

# Recall matrix form of RSS

Model for all N observations together

$$\mathbf{y} = \mathbf{H}\mathbf{w} + \boldsymbol{\varepsilon}$$

# Recall matrix form of RSS

$$\begin{aligned}\text{RSS}(\mathbf{w}) &= \sum_{i=1}^N (y_i - \mathbf{h}(\mathbf{x}_i)^\top \mathbf{w})^2 \\ &= (\mathbf{y} - \mathbf{H}\mathbf{w})^\top (\mathbf{y} - \mathbf{H}\mathbf{w})\end{aligned}$$

# Rewrite magnitude of coefficients in vector notation

$$\|\mathbf{w}\|_2^2 = w_0^2 + w_1^2 + w_2^2 + \dots + w_D^2$$

$$= \begin{array}{c} \boxed{\phantom{w_0}} \boxed{\phantom{w_1}} \boxed{\phantom{w_2}} \boxed{\phantom{\dots}} \boxed{\phantom{w_D}} \\ w_0 \quad w_1 \quad w_2 \quad \dots \quad w_D \end{array} \begin{array}{c} \boxed{\phantom{w_0}} \\ \boxed{\phantom{w_1}} \\ \boxed{\phantom{w_2}} \\ \boxed{\phantom{\dots}} \\ \boxed{\phantom{w_D}} \\ w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{array}$$

$$= \underline{\mathbf{w}^T \mathbf{w}}$$

# Putting it all together

In matrix form, ridge regression cost is:

$$\begin{aligned} \text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2 \\ = \underline{(\mathbf{y} - \mathbf{H}\mathbf{w})^\top (\mathbf{y} - \mathbf{H}\mathbf{w}) + \lambda \mathbf{w}^\top \mathbf{w}} \end{aligned}$$

## Step 2:

# Compute the gradient

# Gradient of ridge regression cost

$$\begin{aligned}\nabla [\text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2] &= \nabla [(\mathbf{y} - \mathbf{H}\mathbf{w})^\top (\mathbf{y} - \mathbf{H}\mathbf{w}) + \lambda \mathbf{w}^\top \mathbf{w}] \\ &= \underbrace{\nabla [(\mathbf{y} - \mathbf{H}\mathbf{w})^\top (\mathbf{y} - \mathbf{H}\mathbf{w})]}_{-2\mathbf{H}^\top (\mathbf{y} - \mathbf{H}\mathbf{w})} + \lambda \underbrace{\nabla [\mathbf{w}^\top \mathbf{w}]}_{2\mathbf{w}}\end{aligned}$$

Why? By analogy to 1d case...

$\mathbf{w}^\top \mathbf{w}$  analogous to  $w^2$  and derivative of  $w^2 = 2w$

**Step 3, Approach 1:**  
Set the gradient = 0



Aside:

## Refresher on identity matrices

$$I_1 = [1], \quad I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \dots, \quad I_n = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

Fun facts:

$$\overset{n \times n}{\mathbf{I}} \overset{n \times 1}{\mathbf{v}} = \mathbf{v}$$

↑ vector

$$\overset{n \times n}{\mathbf{I}} \overset{n \times m}{\mathbf{A}} = \mathbf{A}$$

↑ matrix  
 $\mathbf{A} \mathbf{I} = \mathbf{A}$

$$\overset{n \times n}{\mathbf{A}^{-1}} \overset{n \times n}{\mathbf{A}} = \mathbf{I}$$

↑ by definition of matrix inverse

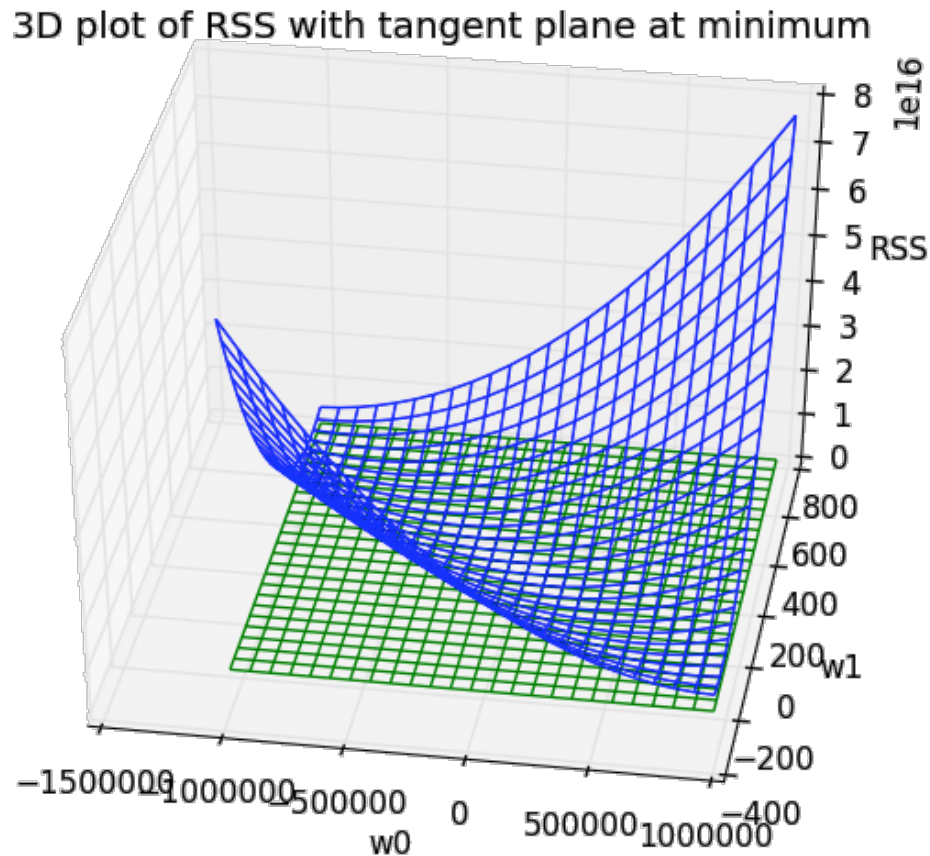
$$\overset{n \times n}{\mathbf{A}} \overset{n \times n}{\mathbf{A}^{-1}} = \mathbf{I}$$

↑  $\mathbf{A} = \mathbf{A} \checkmark$

$$\begin{aligned} \nabla \text{cost}(\mathbf{w}) &= -2\mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{w}) + 2\lambda \mathbf{w} \\ &= -2\mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{w}) + 2\lambda \mathbf{I}\mathbf{w} \end{aligned}$$

equivalent

# Ridge closed-form solution



$$\nabla \text{cost}(\mathbf{w}) = -2\mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{w}) + 2\lambda\mathbf{I}\mathbf{w} = 0$$

Solve for  $\mathbf{w}$ :

$$-\mathbf{H}^T\mathbf{y} + \mathbf{H}^T\mathbf{H}\hat{\mathbf{w}} + \lambda\mathbf{I}\hat{\mathbf{w}} = 0$$

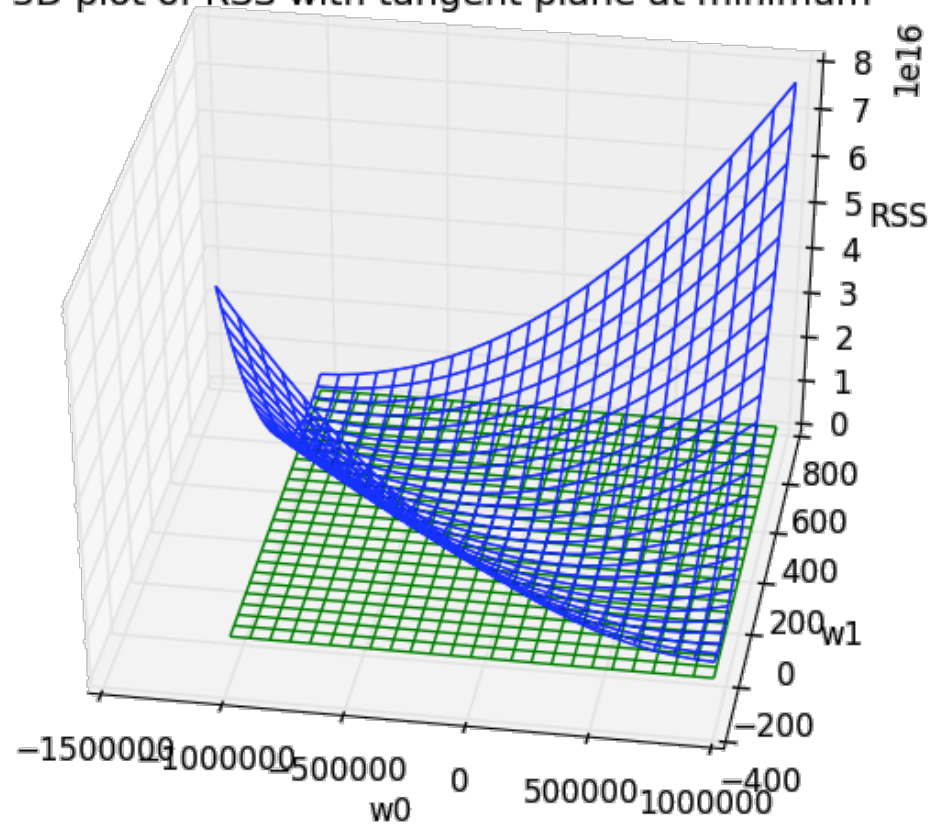
$$\mathbf{H}^T\mathbf{H}\hat{\mathbf{w}} + \lambda\mathbf{I}\hat{\mathbf{w}} = \mathbf{H}^T\mathbf{y}$$

$$(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})\hat{\mathbf{w}} = \mathbf{H}^T\mathbf{y}$$

$$\hat{\mathbf{w}}^{\text{ridge}} = (\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^T\mathbf{y}$$

# Interpreting ridge closed-form solution

3D plot of RSS with tangent plane at minimum



$$\hat{\mathbf{w}}^{\text{ridge}} = (\mathbf{H}^T \mathbf{H} + \lambda \mathbf{I})^{-1} \mathbf{H}^T \mathbf{y}$$

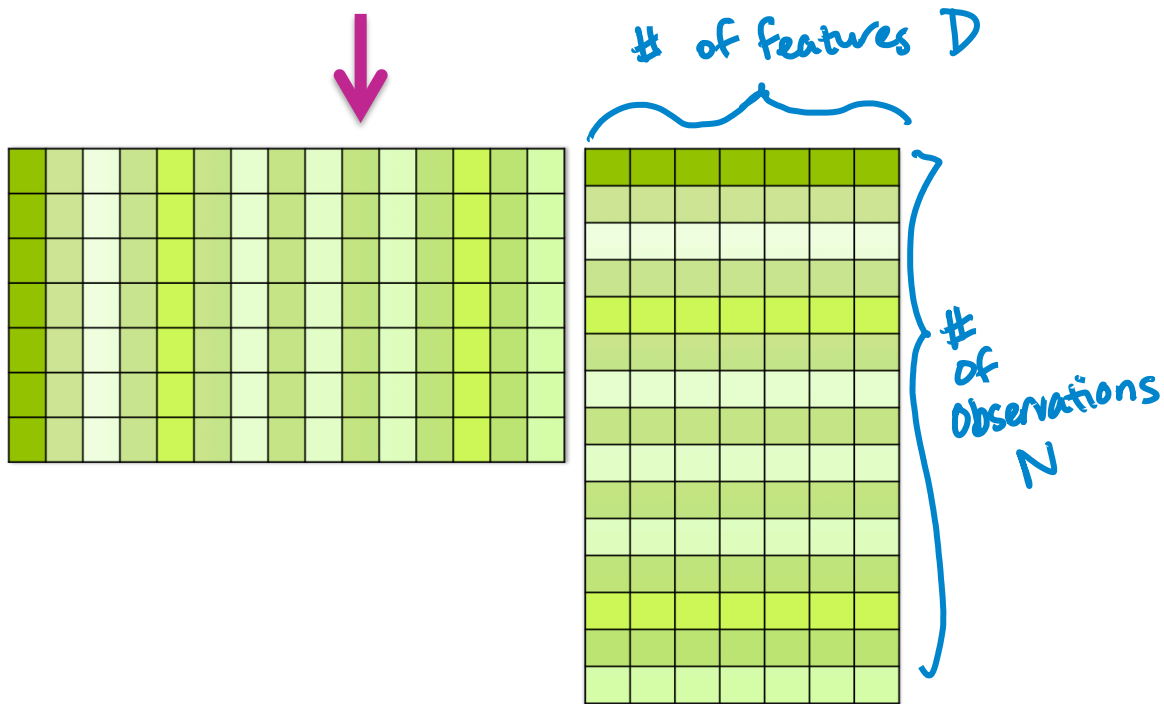
If  $\lambda = 0$ :  $\hat{\mathbf{w}}^{\text{ridge}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{y} = \hat{\mathbf{w}}^{\text{LS}}$  ← old solution!

If  $\lambda = \infty$ :  $\hat{\mathbf{w}}^{\text{ridge}} = \mathbf{0}$  ← because it's like dividing by  $\infty$



# Recall discussion on previous closed-form solution

$$\hat{\mathbf{w}}^{\text{LS}} = (\underbrace{\mathbf{H}^T \mathbf{H}}_{\downarrow})^{-1} \mathbf{H}^T \mathbf{y}$$



Invertible if:

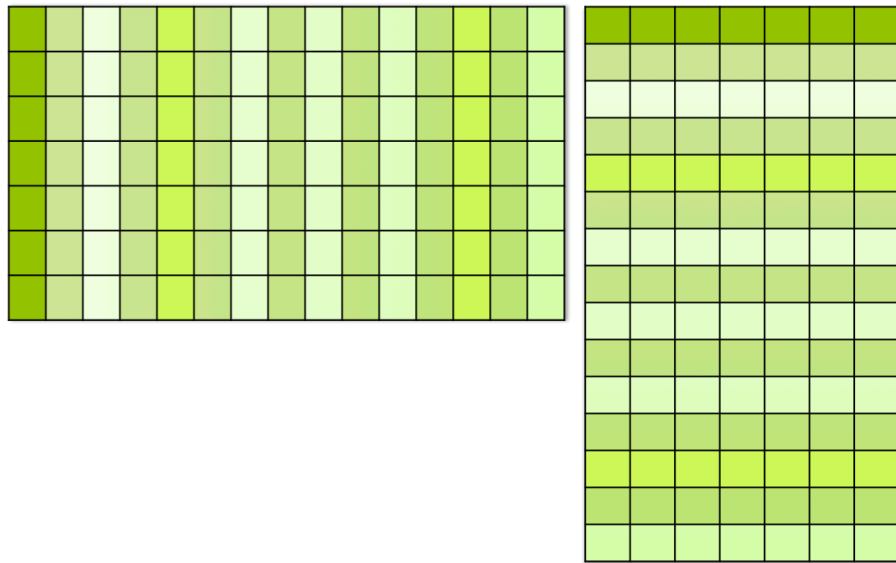
In general,  
(# linearly independent obs)  
 $N > D$

Complexity of inverse:

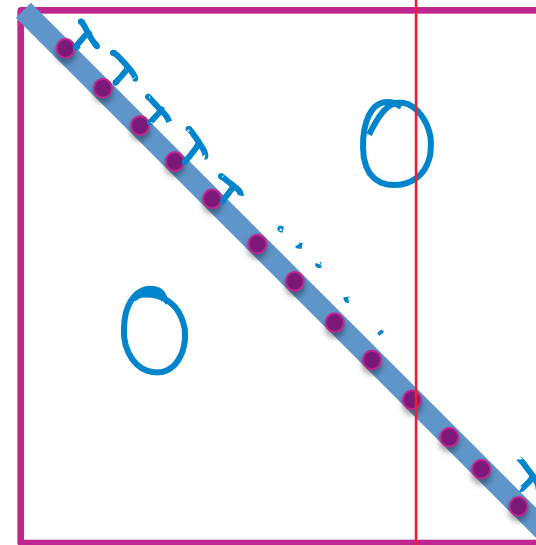
$O(D^3)$

# Discussion of ridge closed-form solution

$$\hat{\mathbf{w}} = (\underbrace{\mathbf{H}^T \mathbf{H}}_{\text{matrix}} + \underbrace{\lambda \mathbf{I}}_{\text{regularization}})^{-1} \mathbf{H}^T \mathbf{y}$$



+



really important for large  $D$   
(lots of features)

Invertible if:

Always if  $\lambda > 0$ ,  
even if  $N < D$

Complexity of  
inverse:

$O(D^3)$ ...

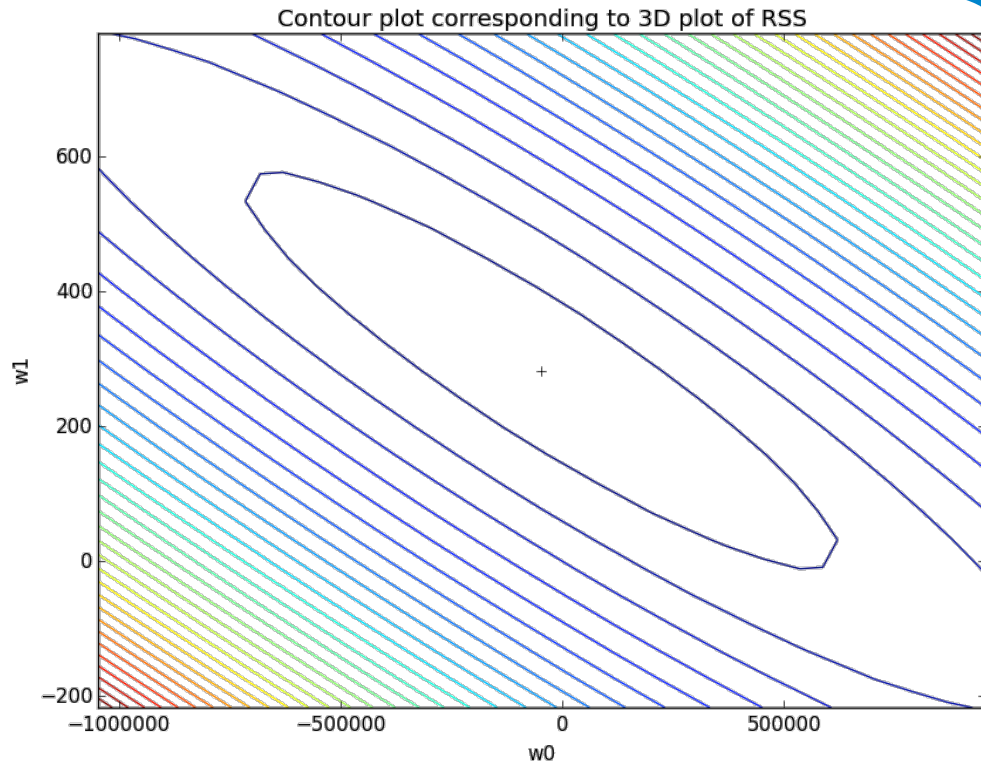
big for large  $D$ !

$\lambda \mathbf{I}$  is making  $\mathbf{H}^T \mathbf{H} + \lambda \mathbf{I}$  more "regular"  
→ "regularized"

## Step 3, Approach 2: Gradient descent

# Elementwise ridge regression gradient descent algorithm

$$\nabla \text{cost}(\mathbf{w}) = -2\mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{w}) + 2\lambda\mathbf{w}$$



Update to  $j^{\text{th}}$  feature weight:

$$w_j^{(t+1)} \leftarrow \underline{w_j^{(t)}} - \eta *$$

Same as before (from RSS term)  $\rightarrow$

$$2 \sum_{i=1}^N \mathbf{x}_i (y_i - \hat{y}_i(\mathbf{w}^{(t)}))$$

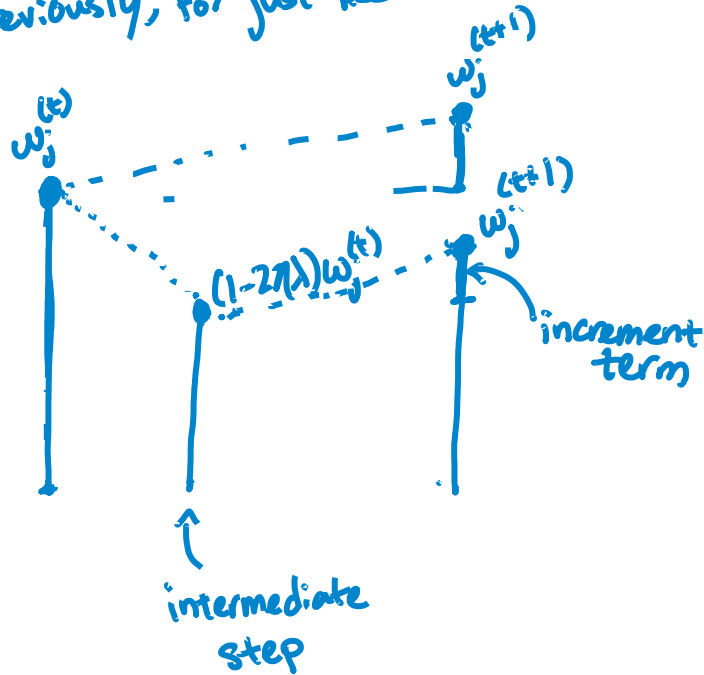
$$+ 2\lambda \underline{w_j^{(t)}}]$$

new term, comes from the  $j^{\text{th}}$  component of  $2\lambda \mathbf{w}$

# Elementwise ridge regression gradient descent algorithm

$$\nabla \text{cost}(\mathbf{w}) = -2\mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{w}) + 2\lambda\mathbf{w}$$

previously, for just RSS



Equivalently:

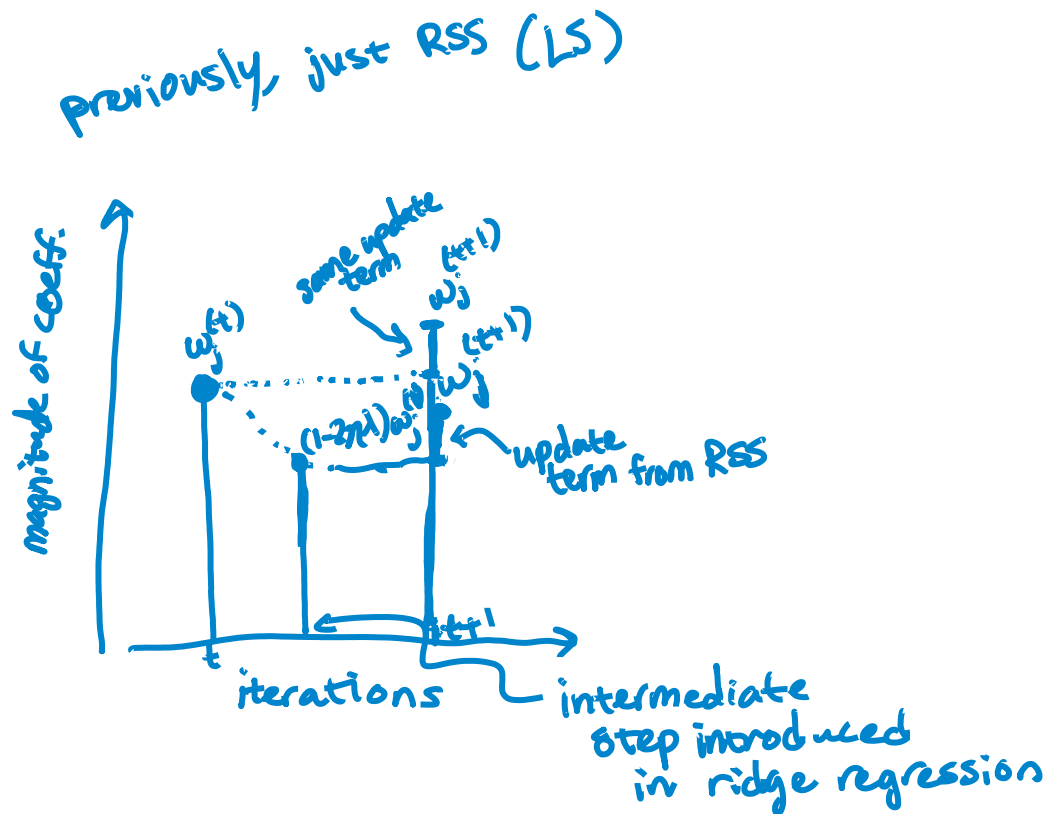
$$w_j^{(t+1)} \leftarrow \underbrace{(1 - 2\eta\lambda)}_{\leq 1} w_j^{(t)} + 2\eta \underbrace{\sum_{i=1}^N \mathbf{x}_i (y_i - \hat{y}_i(\mathbf{w}^{(t)}))}_{\text{increment term}}$$

$2\eta\lambda < 1$



# Elementwise ridge regression gradient descent algorithm

$$\nabla \text{cost}(\mathbf{w}) = -2\mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{w}) + 2\lambda\mathbf{w}$$



Equivalently:

$$w_j^{(t+1)} \leftarrow \underbrace{(1 - 2\eta \lambda)}_{\leq 1} w_j^{(t)} + 2\eta \underbrace{\sum_{i=1}^N \mathbf{x}_i (y_i - \hat{y}_i(\mathbf{w}^{(t)}))}_{\text{update term from RSS}}$$

*Handwritten notes:  $2\eta\lambda < 1$  and  $\lambda^0 \lambda^0$  above the  $\lambda$  in the first term.*

# Recall previous algorithm

init  $\mathbf{w}^{(1)} = 0$  (or randomly, or smartly),  $t = 1$

**while**  $\|\nabla \text{RSS}(\mathbf{w}^{(t)})\| > \epsilon$

**for**  $j = 0, \dots, D$

partial[j] =  $-2 \sum_{i=1}^N (\mathbf{x}_i)_j (y_i - \hat{y}_i(\mathbf{w}^{(t)}))$

$w_j^{(t+1)} \leftarrow w_j^{(t)} - \eta \text{partial}[j]$

$t \leftarrow t + 1$

# Summary of ridge regression algorithm

init  $\mathbf{w}^{(1)} = 0$  (or randomly, or smartly),  $t = 1$

**while**  $|| \nabla \text{RSS}(\mathbf{w}^{(t)}) || > \epsilon$

**for**  $j = 0, \dots, D$

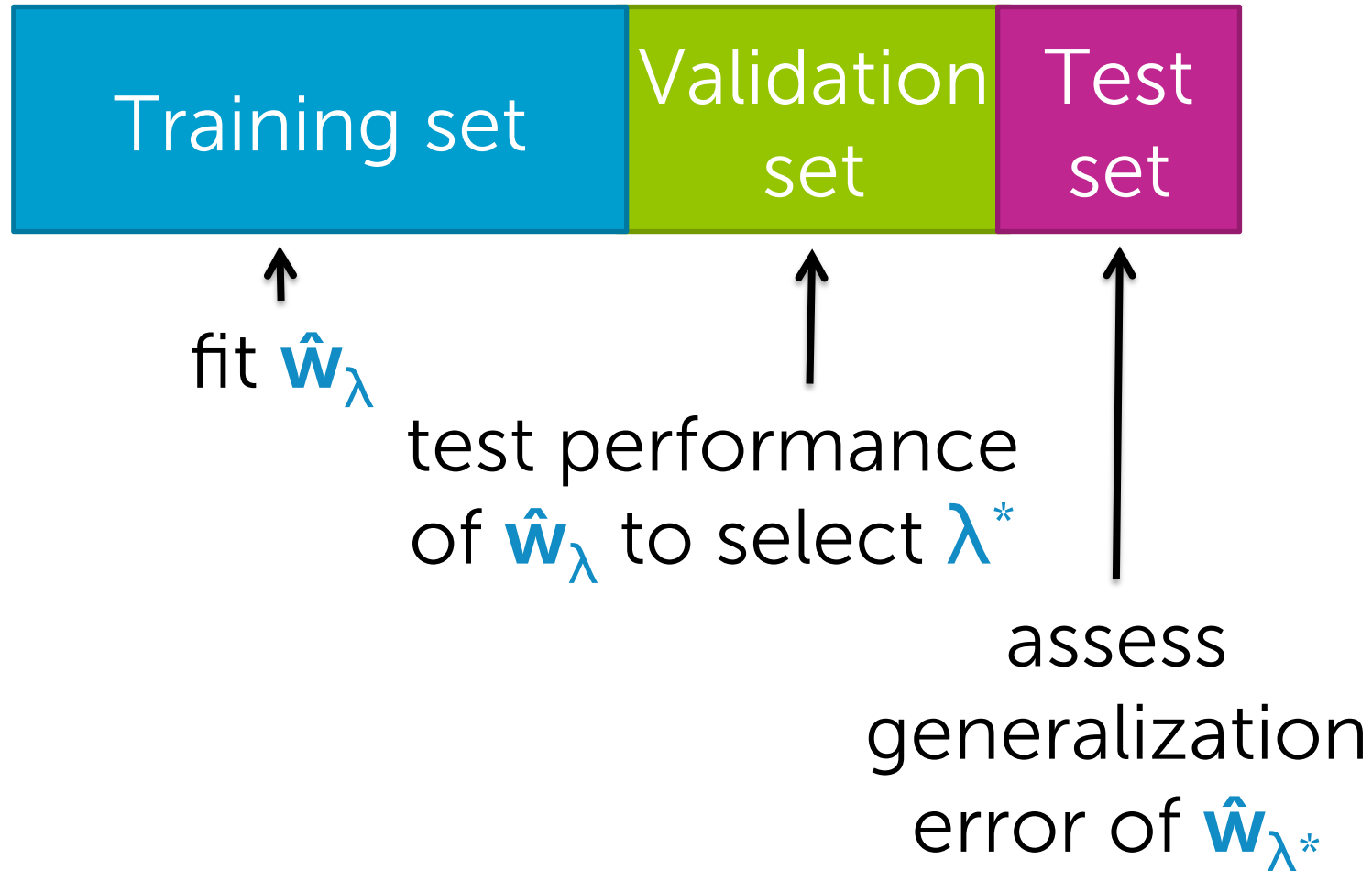
partial[j] =  $-2 \sum_{i=1}^N (\mathbf{x}_i)_j (y_i - \hat{y}_i(\mathbf{w}^{(t)}))$

$w_j^{(t+1)} \leftarrow (1 - 2\eta\lambda) w_j^{(t)} - \eta \text{partial}[j]$

$t \leftarrow t + 1$

# How to choose $\lambda$

# If sufficient amount of data...

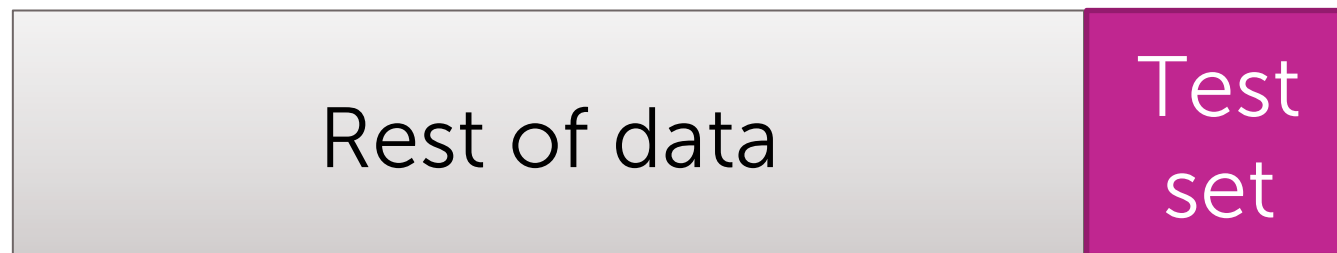


# Start with smallish dataset



All data

# Still form test set and hold out



# How do we use the other data?



use for both training and  
validation, but not so naively



# Recall naïve approach



↑  
small validation set

Is validation set enough to compare performance of  $\hat{w}_\lambda$  across  $\lambda$  values?

No

# Choosing the validation set



small validation set

Didn't have to use the last data points tabulated to form validation set

Can use **any data subset**

# Choosing the validation set



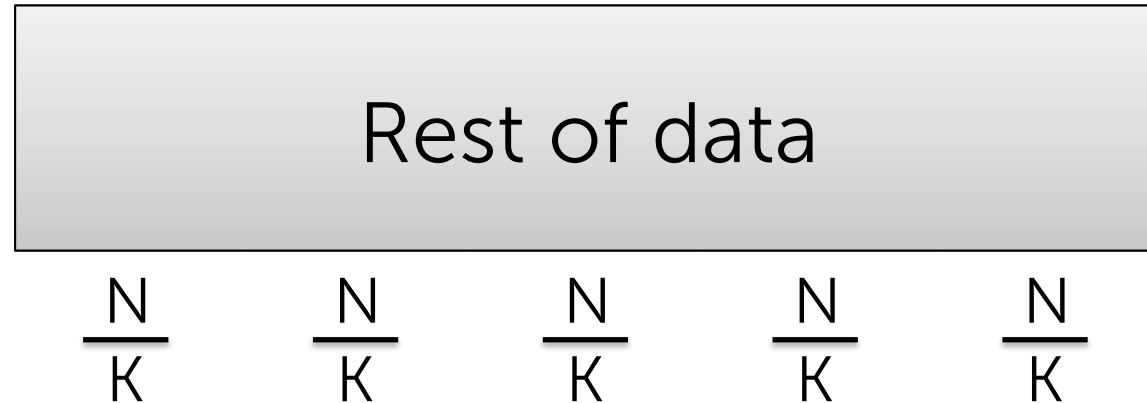
small validation set

Which subset should I use?

**ALL!**

average  
performance  
over all  
choices

# K-fold cross validation

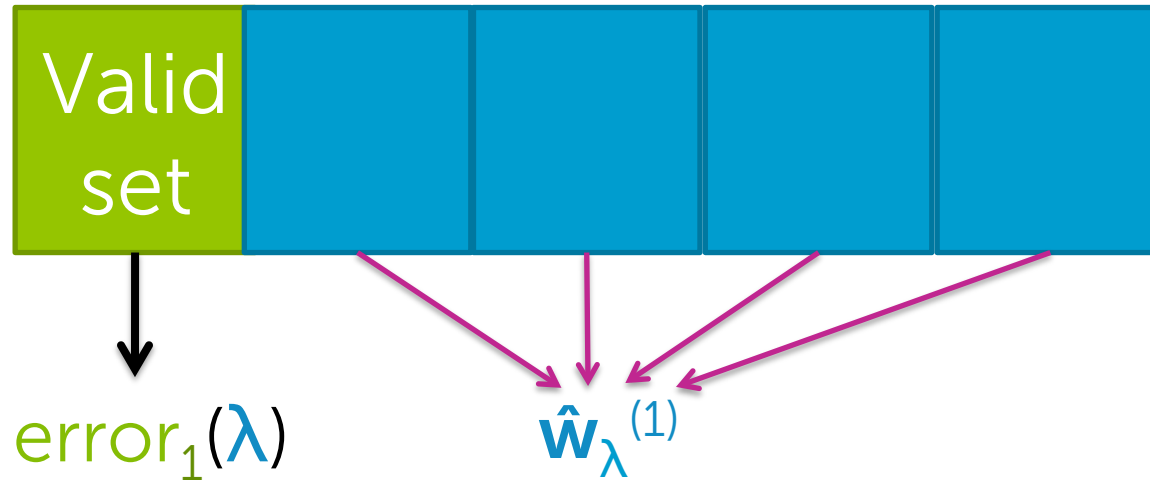


## Preprocessing:

Randomly assign data to K groups

*(use same split of data for all other steps)*

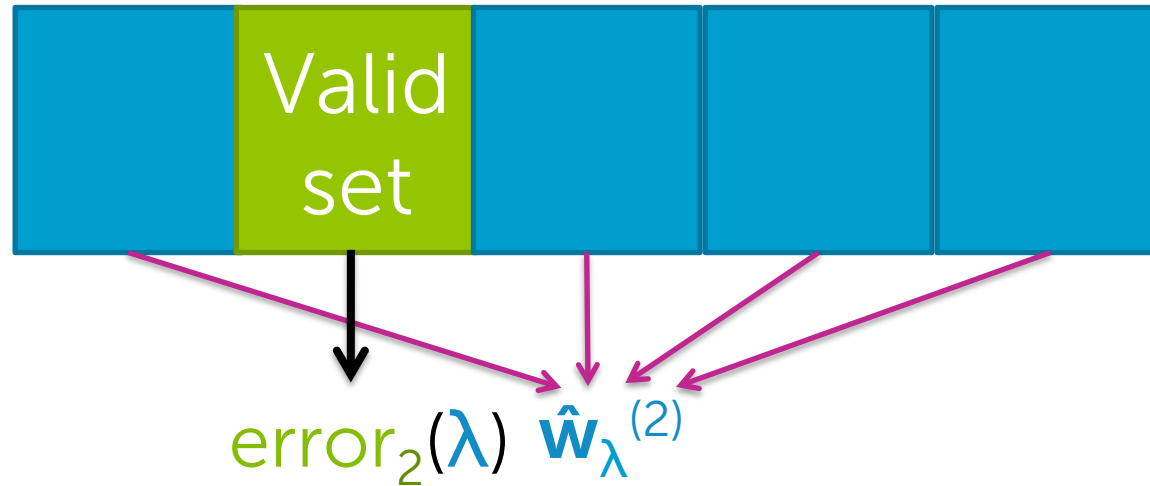
# K-fold cross validation



For  $k=1, \dots, K$

1. Estimate  $\hat{\mathbf{w}}_{\lambda}^{(k)}$  on the training blocks
2. Compute error on validation block:  $\text{error}_k(\lambda)$

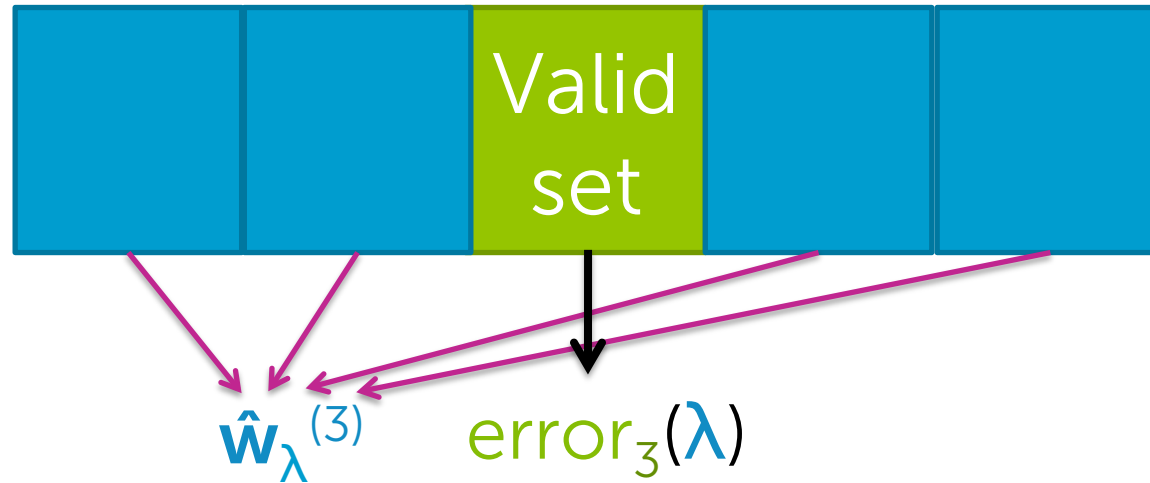
# K-fold cross validation



For  $k=1, \dots, K$

1. Estimate  $\hat{\mathbf{w}}_{\lambda}^{(k)}$  on the training blocks
2. Compute error on validation block:  $\text{error}_k(\lambda)$

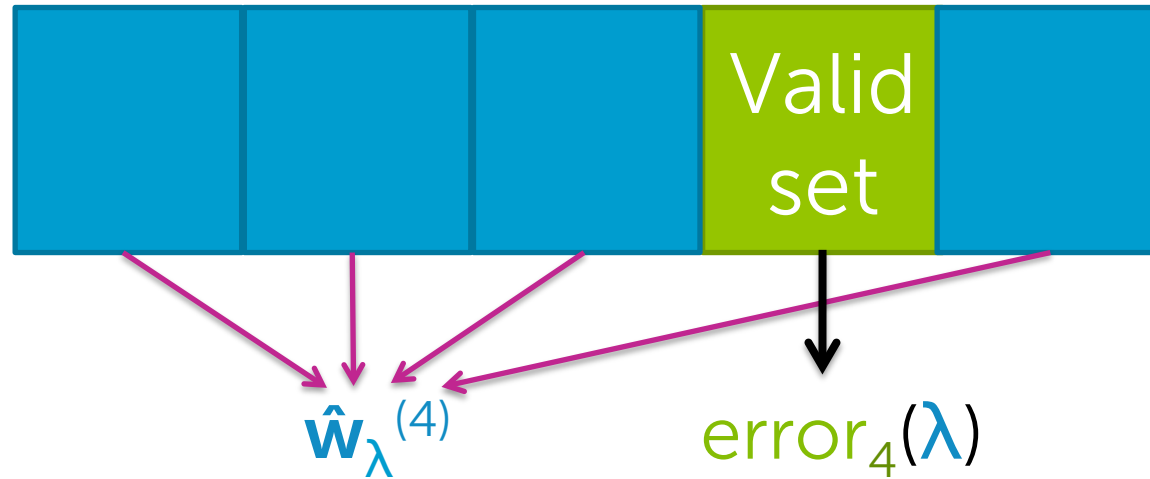
# K-fold cross validation



For  $k=1, \dots, K$

1. Estimate  $\hat{\mathbf{w}}_{\lambda}^{(k)}$  on the training blocks
2. Compute error on validation block:  $\text{error}_k(\lambda)$

# K-fold cross validation

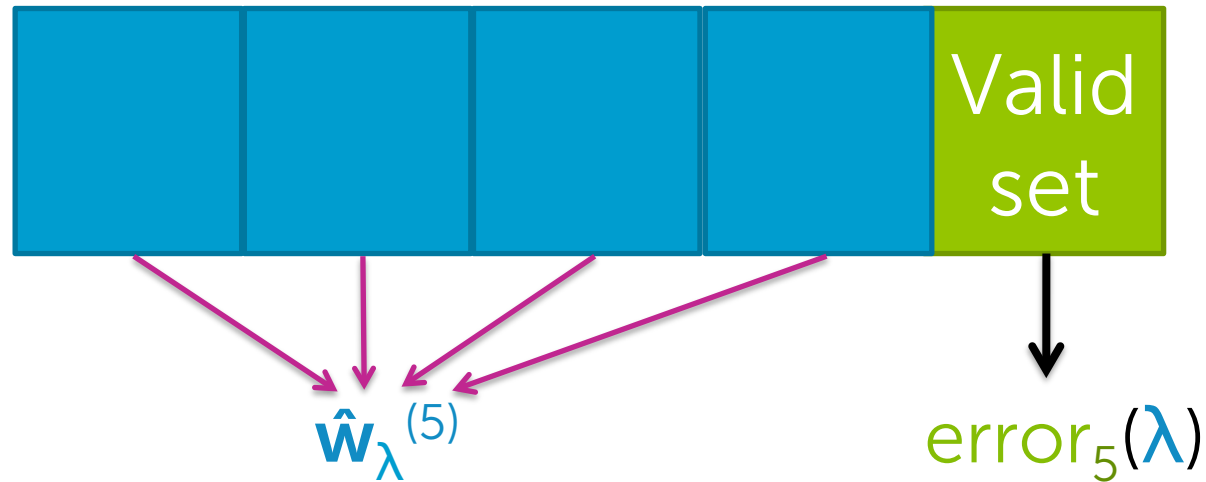


For  $k=1, \dots, K$

1. Estimate  $\hat{\mathbf{w}}_{\lambda}^{(k)}$  on the training blocks
2. Compute error on validation block:  $\text{error}_k(\lambda)$



# K-fold cross validation



For  $k=1, \dots, K$

1. Estimate  $\hat{\mathbf{w}}_{\lambda}^{(k)}$  on the training blocks
2. Compute error on validation block:  $\text{error}_k(\lambda)$

Compute average error:  $\text{CV}(\lambda) = \frac{1}{K} \sum_{k=1}^K \text{error}_k(\lambda)$

# K-fold cross validation



Repeat procedure for each choice of  $\lambda$

Choose  $\lambda^*$  to minimize  $CV(\lambda)$

# What value of K?

Formally, the **best approximation** occurs for validation sets of size 1 ( $K=N$ )

leave-one-out  
cross validation

Computationally intensive

- requires computing  $N$  fits of model per  $\lambda$

Typically,  $K=5$  or  $10$

5-fold CV

10-fold CV

# How to handle the intercept

# Recall multiple regression model

Model:

$$y_i = w_0 h_0(\mathbf{x}_i) + w_1 h_1(\mathbf{x}_i) + \dots + w_D h_D(\mathbf{x}_i) + \varepsilon_i$$
$$= \sum_{j=0}^D w_j h_j(\mathbf{x}_i) + \varepsilon_i$$

*feature 1* =  $h_0(\mathbf{x})$ ...often 1 (constant)

*feature 2* =  $h_1(\mathbf{x})$ ... e.g.,  $\mathbf{x}[1]$

*feature 3* =  $h_2(\mathbf{x})$ ... e.g.,  $\mathbf{x}[2]$

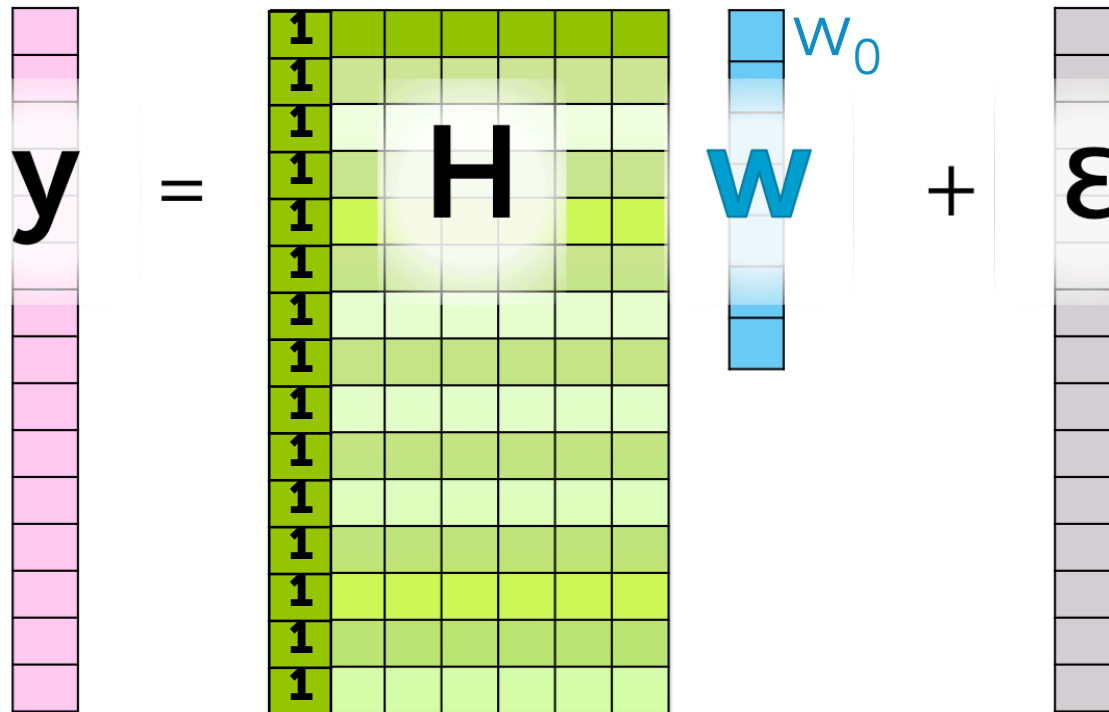
...

*feature D+1* =  $h_D(\mathbf{x})$ ... e.g.,  $\mathbf{x}[d]$

# If constant feature...

$$y_i = w_0 + w_1 h_1(\mathbf{x}_i) + \dots + w_D h_D(\mathbf{x}_i) + \varepsilon_i$$

In matrix notation for N observations:



The diagram illustrates the matrix notation for N observations. It shows a vertical vector  $y$  (pink) equal to the product of a matrix  $H$  (green) and a vector  $w$  (blue), plus a vertical vector  $\varepsilon$  (grey). The matrix  $H$  has 15 rows and 6 columns. The first column of  $H$  contains 15 ones, representing the constant feature. The vector  $w$  has 6 elements, with the first element labeled  $w_0$ . The vector  $\varepsilon$  has 15 elements.

$$\mathbf{y} = \mathbf{H} \mathbf{w} + \boldsymbol{\varepsilon}$$

# Do we penalize intercept?

Standard ridge regression cost:

$$\text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

 strength of penalty

Encourages intercept  $w_0$  to also be small

Do we want a small intercept?

Conceptually, not indicative of overfitting...

# Option 1: Don't penalize intercept

Modified ridge regression cost:

$$\text{RSS}(\mathbf{w}_0, \mathbf{w}_{\text{rest}}) + \lambda \|\mathbf{w}_{\text{rest}}\|_2^2$$

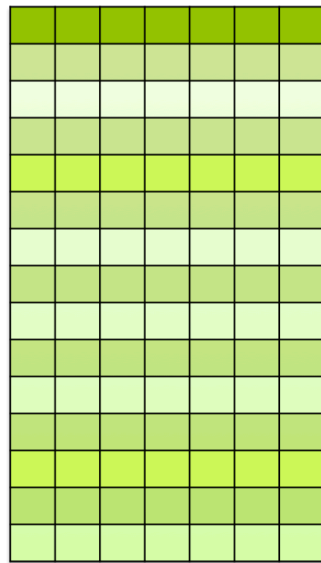
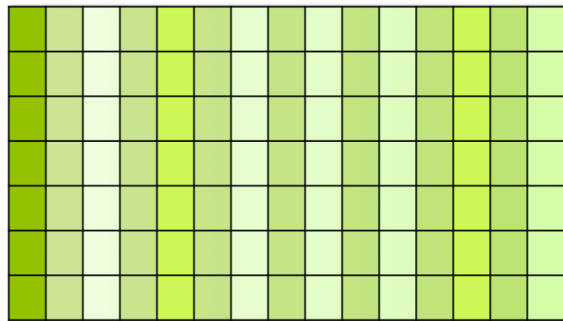
How to implement this in practice?



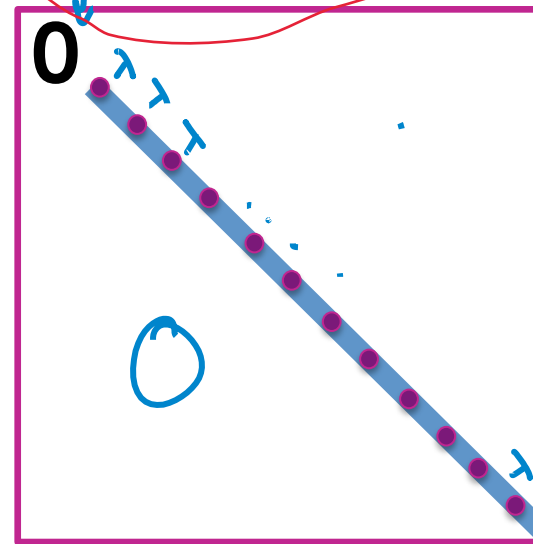
# Option 1: Don't penalize intercept

– Closed-form solution –

$$\hat{\mathbf{w}} = (\underbrace{\mathbf{H}^T \mathbf{H}}_{\text{matrix}} + \underbrace{\lambda \mathbf{I}^{\text{mod}}}_{\text{matrix}})^{-1} \mathbf{H}^T \mathbf{y}$$



+



no penalty in error corresponding to  $w_0$  index

new penalty:  
 $\lambda \mathbf{w}_{\text{rest}}^T \mathbf{w}_{\text{rest}}$

gradient:

$$2\lambda \mathbf{w}_{\text{rest}} = 2\lambda \mathbf{I}^{\text{mod}} \mathbf{w}_{\text{rest}} \rightarrow 2\lambda \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} w_0 \\ \mathbf{w}_{\text{rest}} \end{bmatrix}$$

# Option 1: Don't penalize intercept

- Gradient descent algorithm –

```
while ||  $\nabla$  RSS( $\mathbf{w}^{(t)}$ ) || >  $\epsilon$ 
  for j=0,...,D
    partial[j] = -2  $\sum_{i=1}^N \mathbf{x}_i(j) (y_i - \hat{y}_i(\mathbf{w}^{(t)}))$ 
    if j==0
       $w_0^{(t+1)} \leftarrow w_0^{(t)} - \eta \text{ partial}[j]$ 
    else
       $w_j^{(t+1)} \leftarrow (1 - 2\eta\lambda) w_j^{(t)} - \eta \text{ partial}[j]$ 
    t  $\leftarrow$  t + 1
```

← old LS update (no shrinkage to  $w_0$ )

← for all other features

← ridge update

# Option 2: Center data first

If data are first **centered about 0**, then favoring small intercept not so worrisome

**Step 1:** Transform  $y$  to have 0 mean

**Step 2:** Run ridge regression as normal  
(closed-form or gradient algorithms)

# Summary for ridge regression

# What you can do now...

- Describe what happens to magnitude of estimated coefficients when model is overfit
- Motivate form of ridge regression cost function
- Describe what happens to estimated coefficients of ridge regression as tuning parameter  $\lambda$  is varied
- Interpret coefficient path plot
- Estimate ridge regression parameters:
  - In closed form
  - Using an iterative gradient descent algorithm
- Implement K-fold cross validation to select the ridge regression tuning parameter  $\lambda$