Ridge Regression:

Regulating overfitting when using many features

Emily Fox & Carlos Guestrin

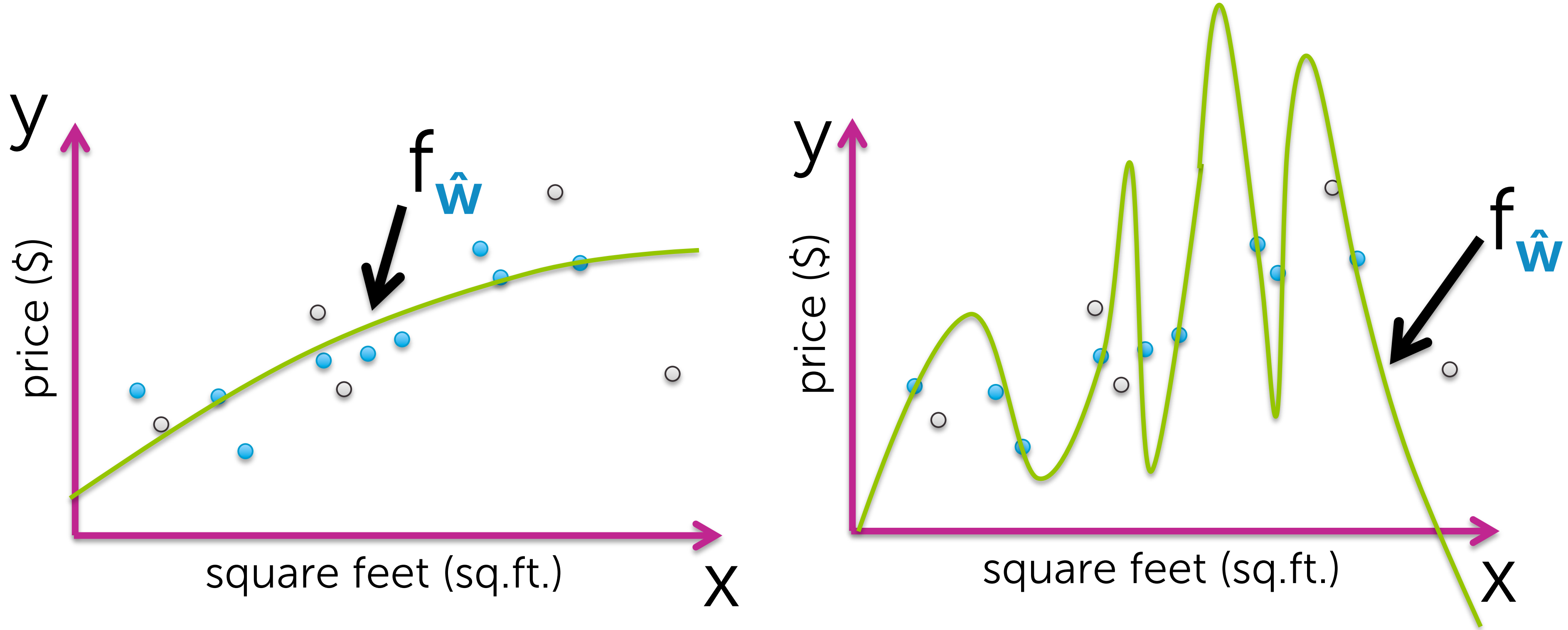
Machine Learning Specialization

University of Washington

Overfitting of polynomial regression

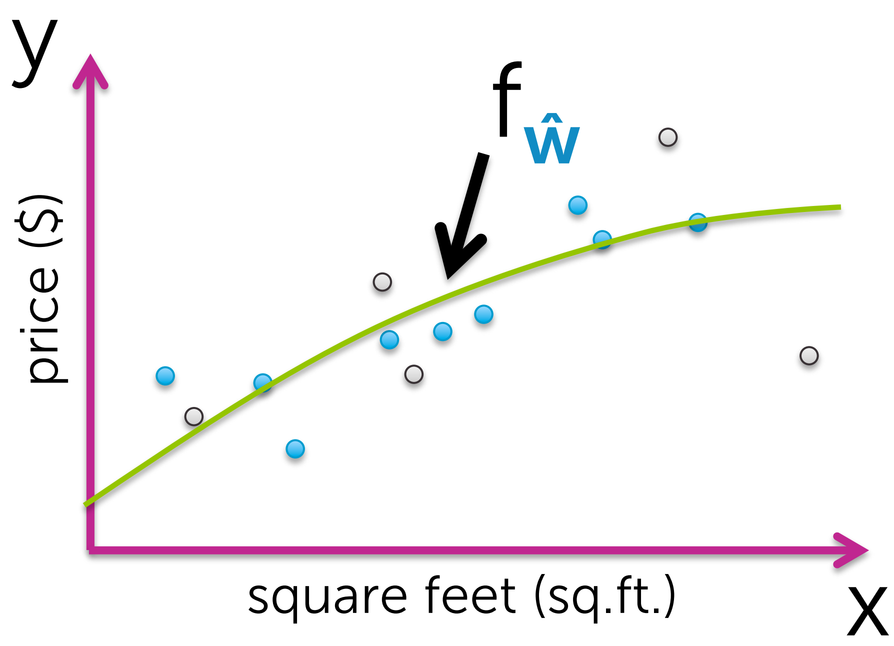
Flexibility of high-order polynomials

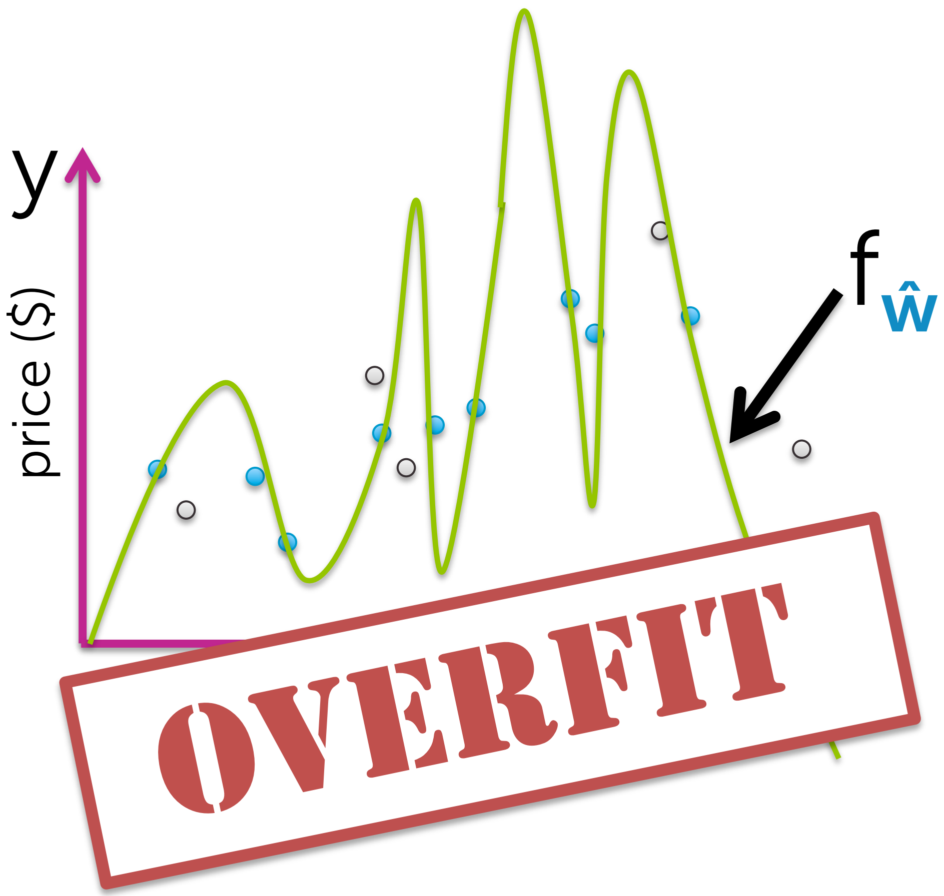
$$y_i = w_0 + w_1 x_i + w_2 x_i^2 + \ldots + w_p x_i^p + \varepsilon_i$$



Flexibility of high-order polynomials

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





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)

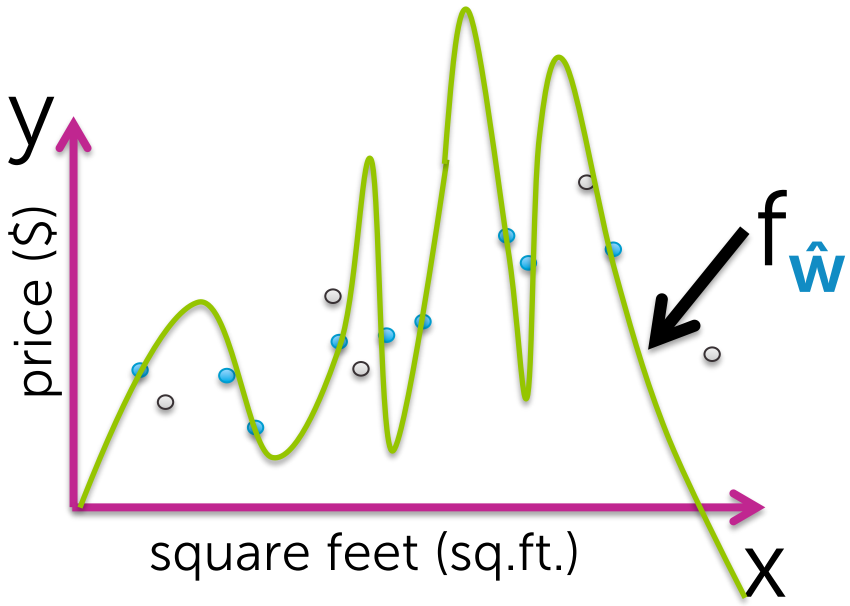
$$y_i = \sum_{j=0}^{D} w_j h_j(\mathbf{x}_i) + \varepsilon_i$$

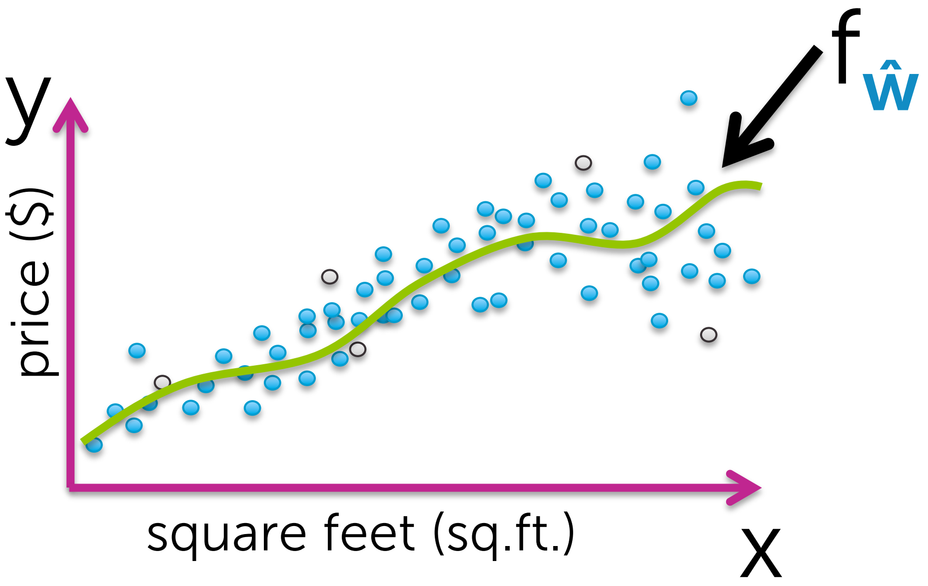
- Square feet
- # bathrooms
- # bedrooms
- Lot size
- Year built
- …

How does # of observations influence overfitting?

Few observations (N small)
→ rapidly overfit as model complexity increases

Many observations (N very large)
→ harder to overfit

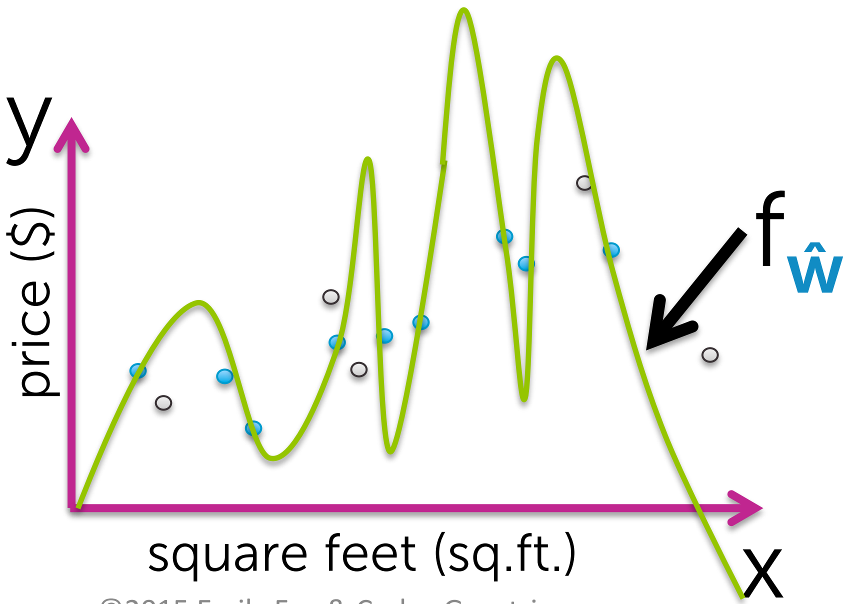




How does # of inputs influence overfitting?

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

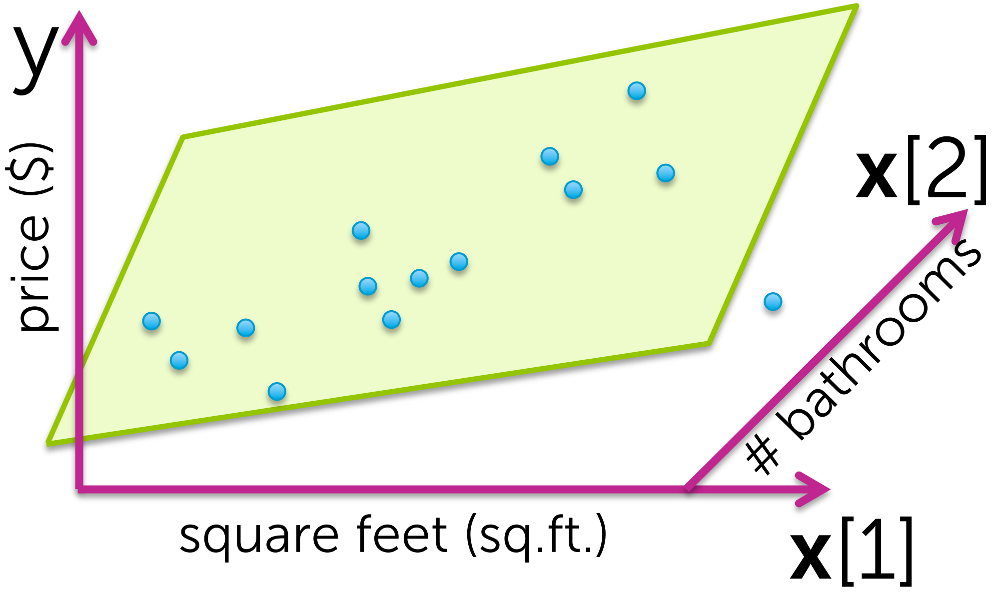
Data must include representative examples of all possible (sq.ft., $) pairs to avoid overfitting



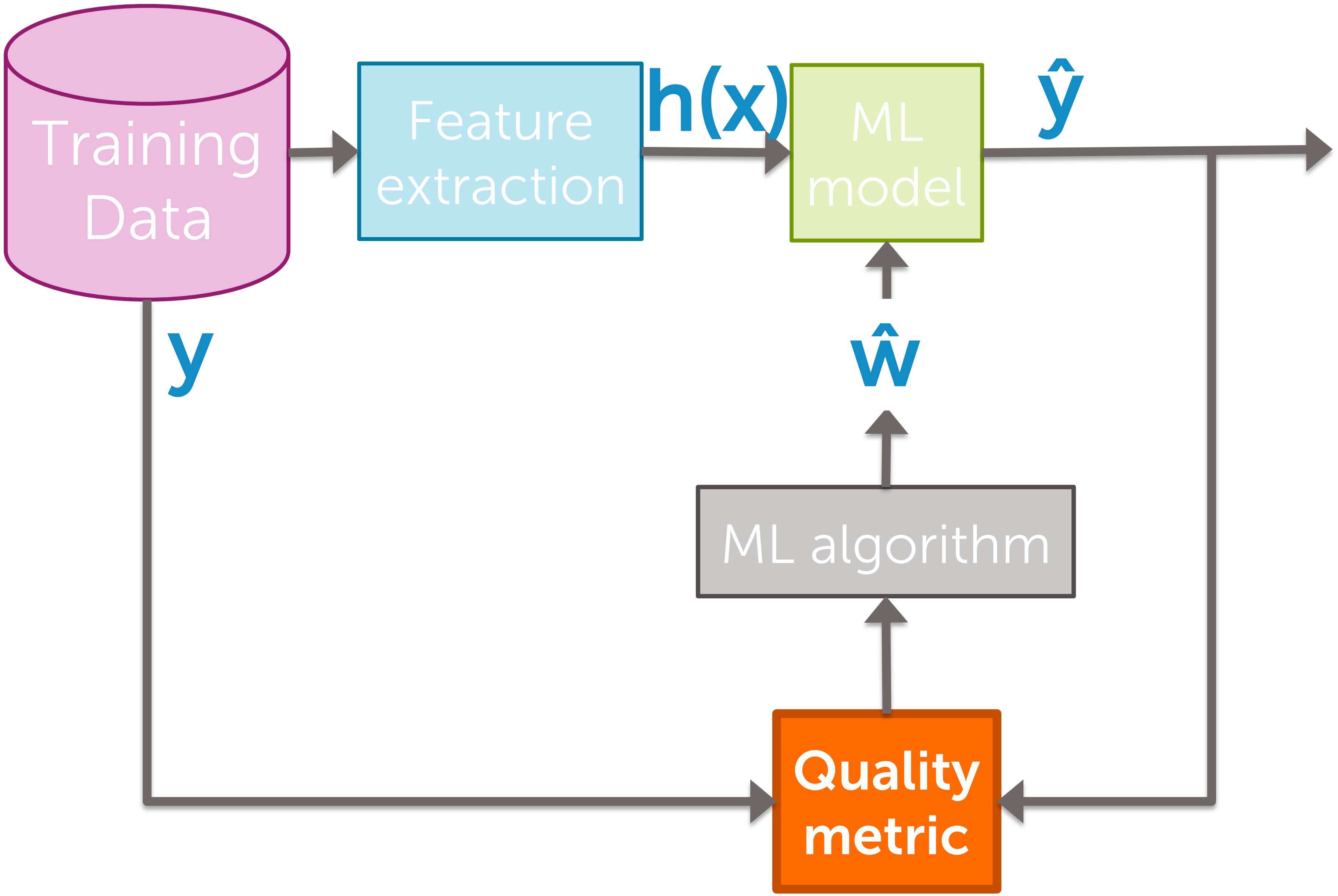
How does # of inputs influence overfitting?

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



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


Training Data
Feature extraction
h(x)
ML model
ŷ
y
ŵ
ML algorithm
Quality metric

Desired total cost format

Want to balance:

i. How well function fits data

ii. Magnitude of coefficients

want to balance

measure quality of fit

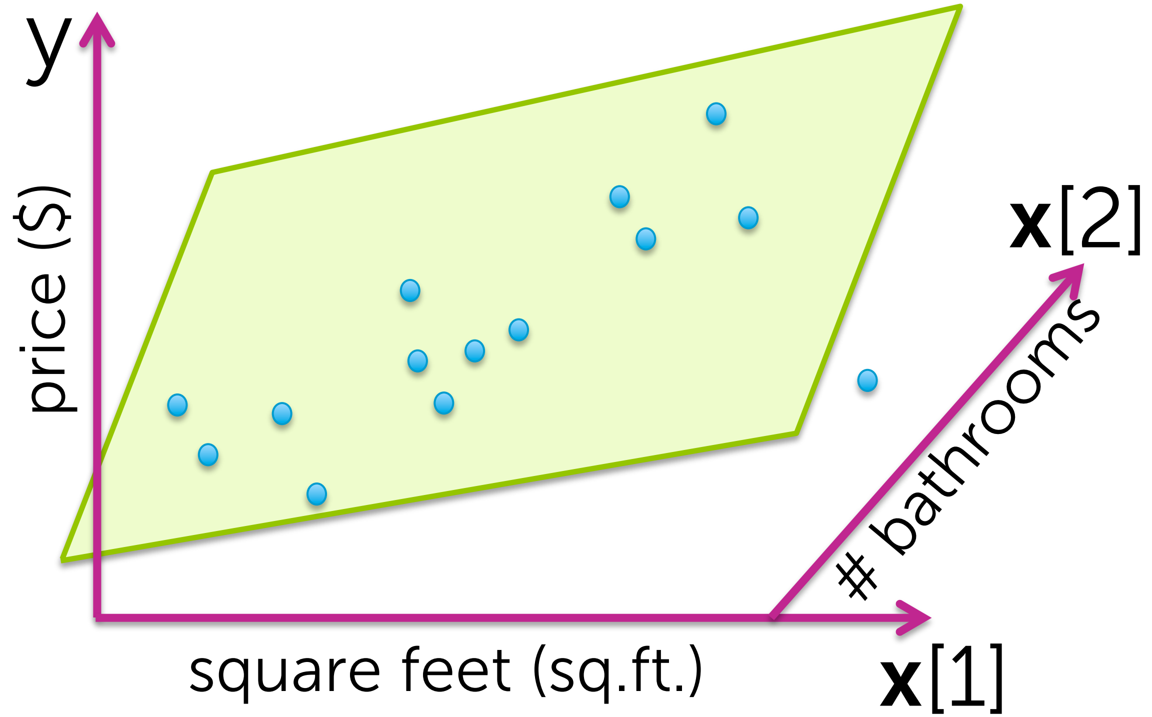
Total cost =

measure of fit + measure of magnitude of coefficients

small # = good fit to training data

small # = not overfit

Measure of fit to training data



$$\text{RSS}(\mathbf{w}) = \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i)^T \mathbf{w})^2$$

pred. value using w

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

small RSS → model fitting training data well

Measure of magnitude of regression coefficient

What summary # is indicative of size of regression coefficients?

- Sum? $w_0 = 1{,}527{,}301 \quad w_1 = -1{,}605{,}253$

  $w_0 + w_1 = \text{small } \#$

- Sum of absolute value?

  $|w_0| + |w_1| + \ldots + |w_D| = \sum_{j=0}^{D} |w_j| \triangleq \|w\|_1 \quad L_1 \text{ norm} \ \ldots$ discuss more in next module

- Sum of squares (L_2 norm)

  $w_0^2 + w_1^2 + \ldots + w_D^2 = \sum_{j=0}^{D} w_j^2 \triangleq \|w\|_2^2 \quad L_2 \text{ norm} \ \ldots$ focus of this module

Consider specific total cost

Total cost =

measure of fit + measure of magnitude of coefficients

Consider specific total cost

Total cost =

$$\underbrace{\text{measure of fit}}_{RSS(\mathbf{w})} + \underbrace{\text{measure of magnitude of coefficients}}_{||\mathbf{w}||_2^2}$$

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 λ=0:

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

If $\lambda=\infty$:

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

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

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

Consider resulting objective

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

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

tuning parameter = balance of fit and magnitude

Ridge regression
(a.k.a L_2 regularization)

Bias-variance tradeoff

Large λ:

high bias, low variance

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

In essence, λ controls model complexity

Small λ:

low bias, high variance

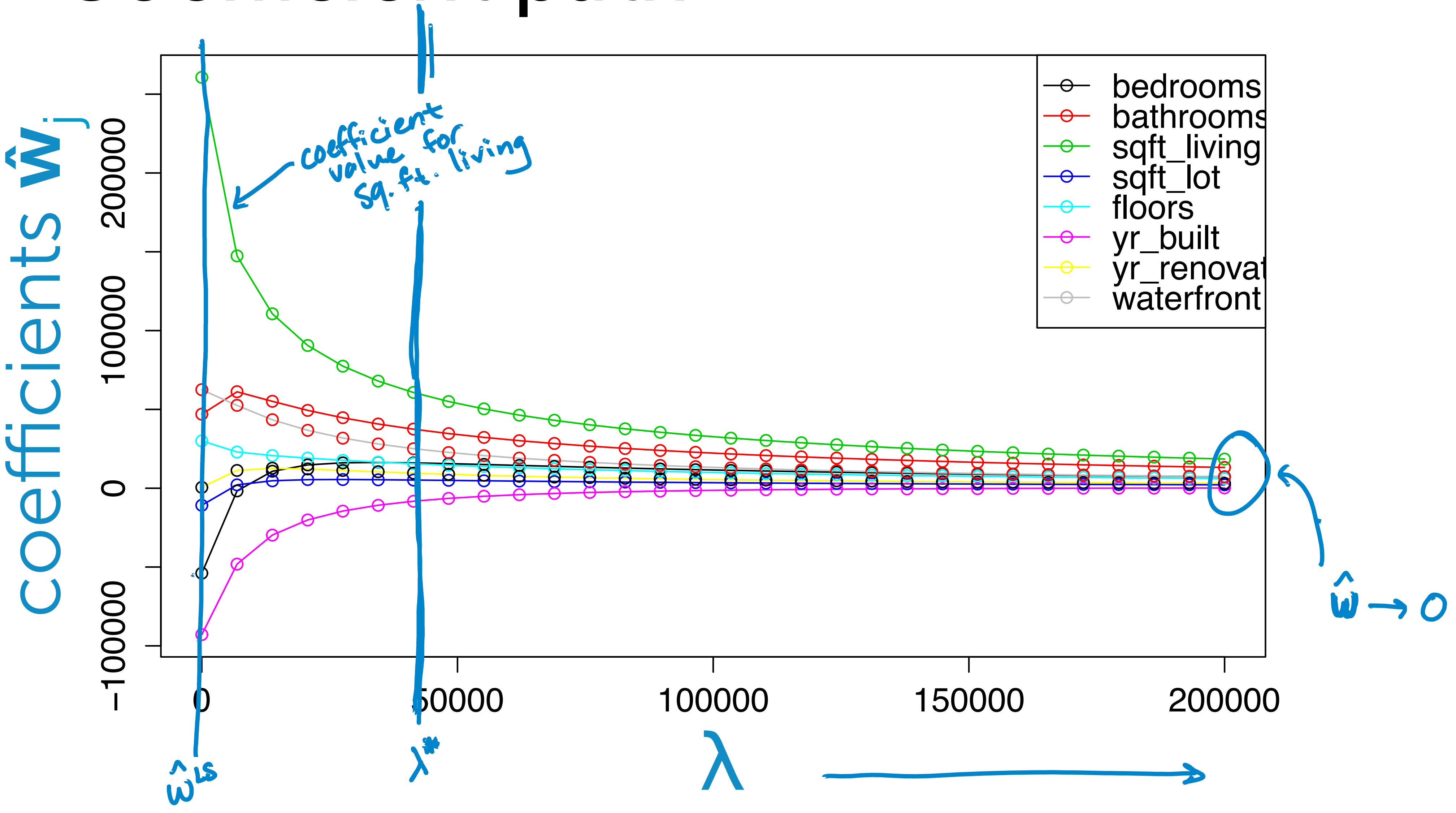
(e.g., standard least squares (RSS) fit of high-order polynomial for $\lambda = 0$)

Revisit polynomial fit demo

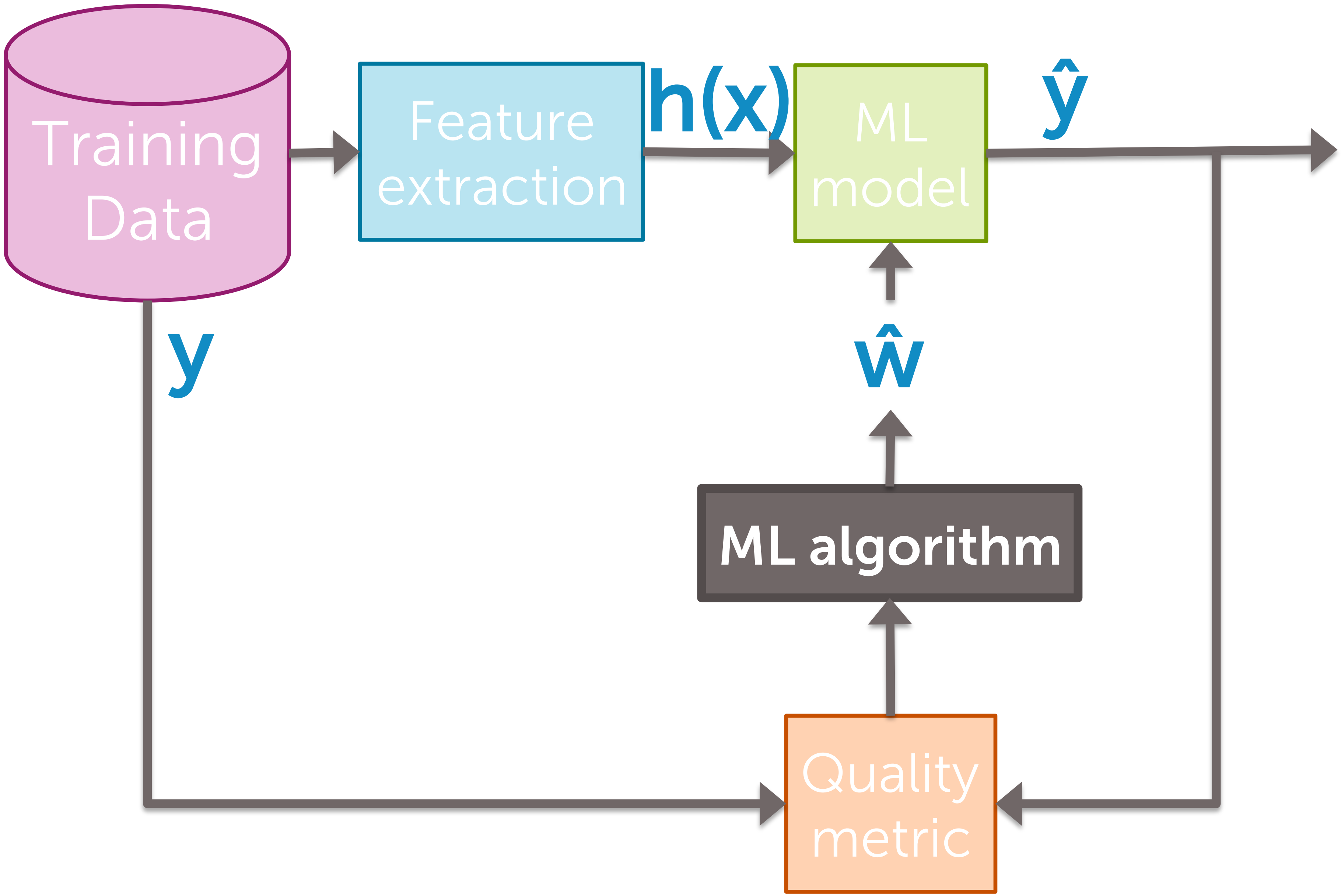
What happens if we refit our high-order polynomial, but now using **ridge regression**?

Will consider a few settings of λ …

Coefficient path



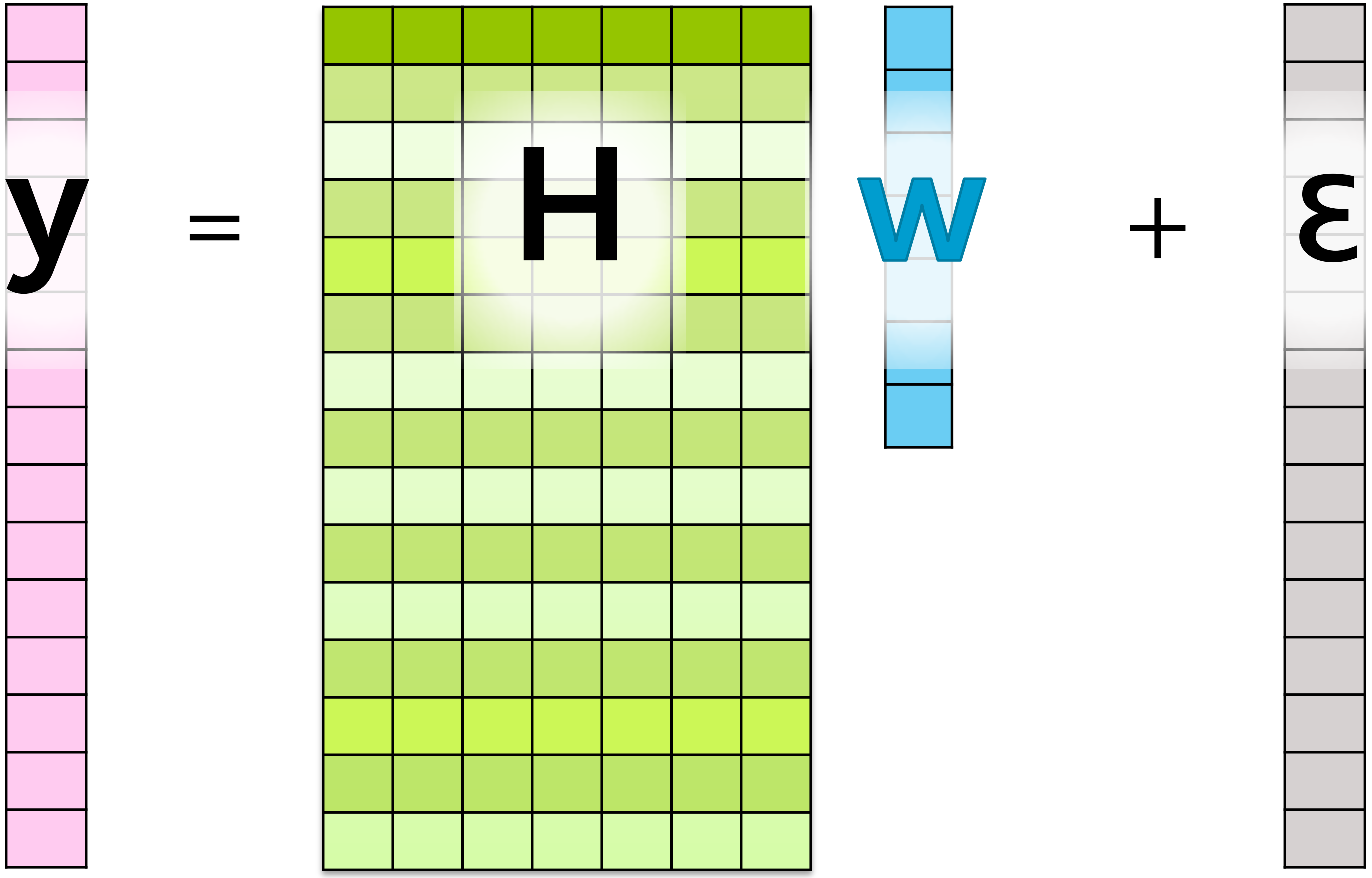
Fitting the ridge regression model (for given λ value)


Training Data
Feature extraction
h(x)
ML model
ŷ
y
ŵ
ML algorithm
Quality metric

Step 1:
Rewrite total cost in matrix notation

Recall matrix form of RSS

Model for all N observations together



Recall matrix form of RSS

$$\text{RSS}(\mathbf{w}) = \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i)^T \mathbf{w})^2$$

$$= (\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w})$$

Rewrite magnitude of coefficients in vector notation

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

$$= \begin{bmatrix} w_0 & w_1 & w_2 & \cdots & & & w_D \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ \\ \\ w_D \end{bmatrix}$$

$$= w^T w$$

Putting it all together

In matrix form, ridge regression cost is:

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

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

Step 2:
Compute the gradient

Gradient of ridge regression cost

$$\nabla[\mathrm{RSS}(\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2] = \nabla[(\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w}) + \lambda\mathbf{w}^T\mathbf{w}]$$

$$= \underbrace{\nabla[(\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w})]}_{-2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w})} + \lambda \underbrace{\nabla[\mathbf{w}^T\mathbf{w}]}_{2\mathbf{w}}$$

Why? By analogy to 1d case…

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

Step 3, Approach 1:

Set the gradient = 0

Aside:
Refresher on identity matrics

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

Fun facts:

$$\underset{\text{vector}}{\overset{n\times n\ \ n\times 1}{\mathbf{Iv}}} = \mathbf{v} \qquad \overset{n\times n\ \ n\times m}{\mathbf{IA}} = \mathbf{A} \qquad \overset{n\times n\ \ \ n\times n}{\mathbf{A}^{-1}\mathbf{A}} = \mathbf{I} \qquad \overset{n\times n\ \ n\times n}{\underbrace{\mathbf{AA}^{-1}}_{I}} = \mathbf{I}$$

matrix

AI = A

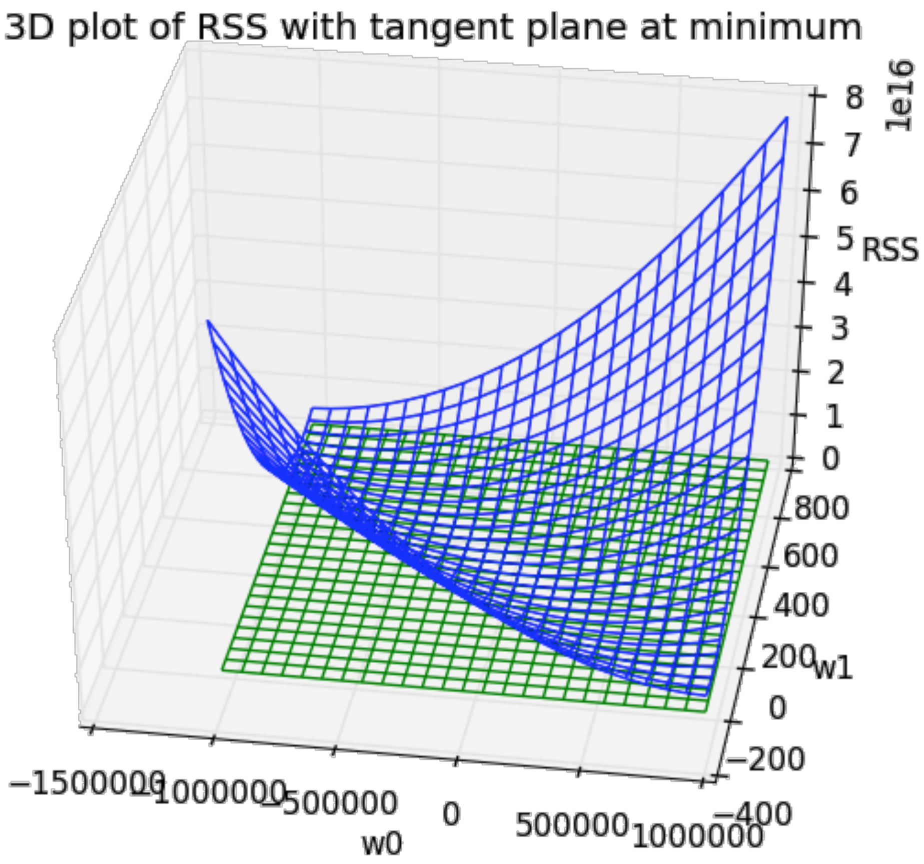
by definition of matrix inverse

A = A ✓

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

equivalent

Ridge closed-form solution



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

Solve for $\mathbf{w}$:

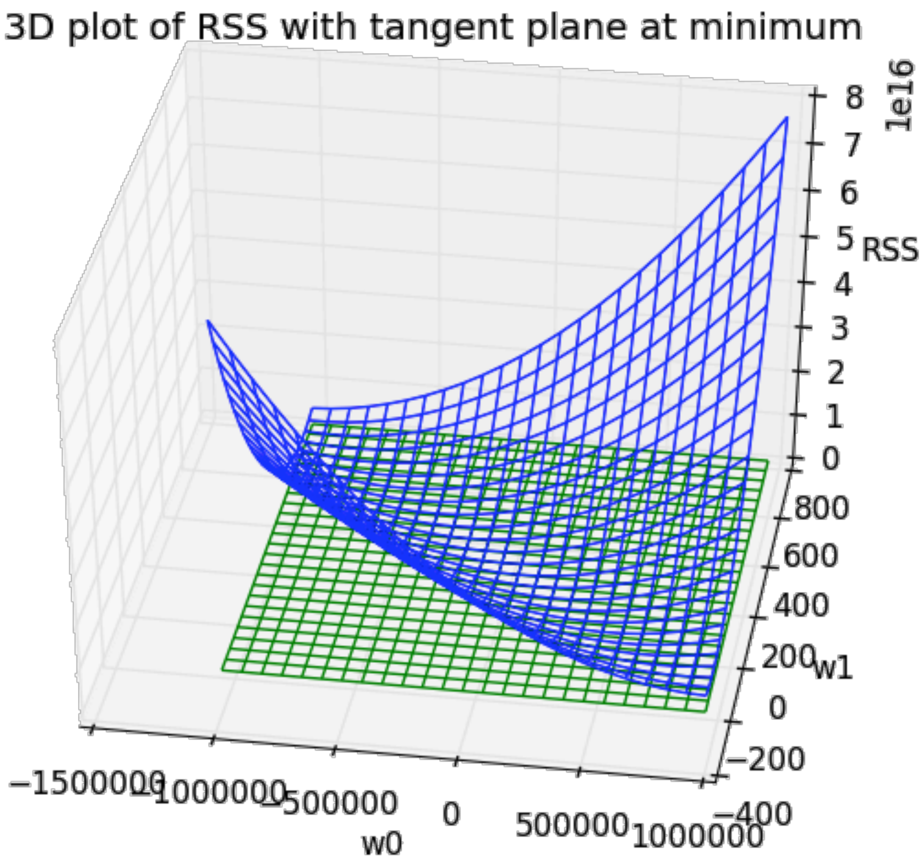
$$-H^T y + H^T H \hat{w} + \lambda I \hat{w} = 0$$

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

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

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

Interpreting ridge closed-form solution



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

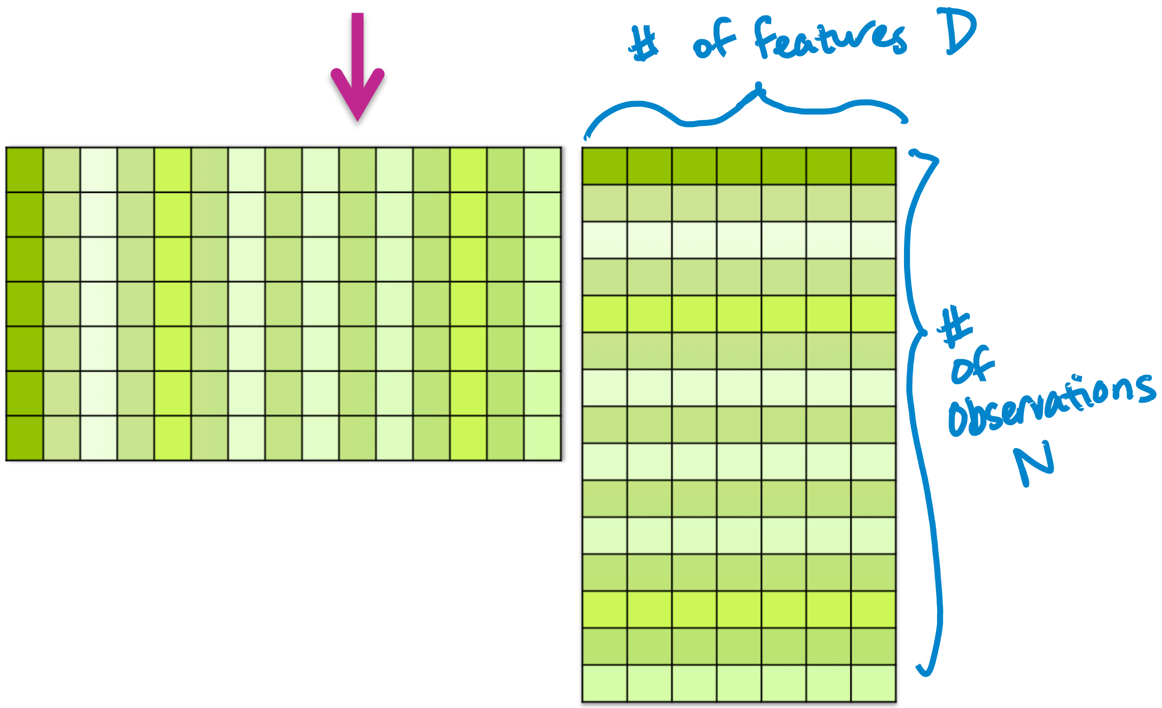
If $\lambda=0$: $\hat{w}^{ridge} = (H^TH)^{-1}H^Ty = \hat{w}^{LS}$ ← old solution!

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

✓

Recall discussion on previous closed-form solution

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



Invertible if:

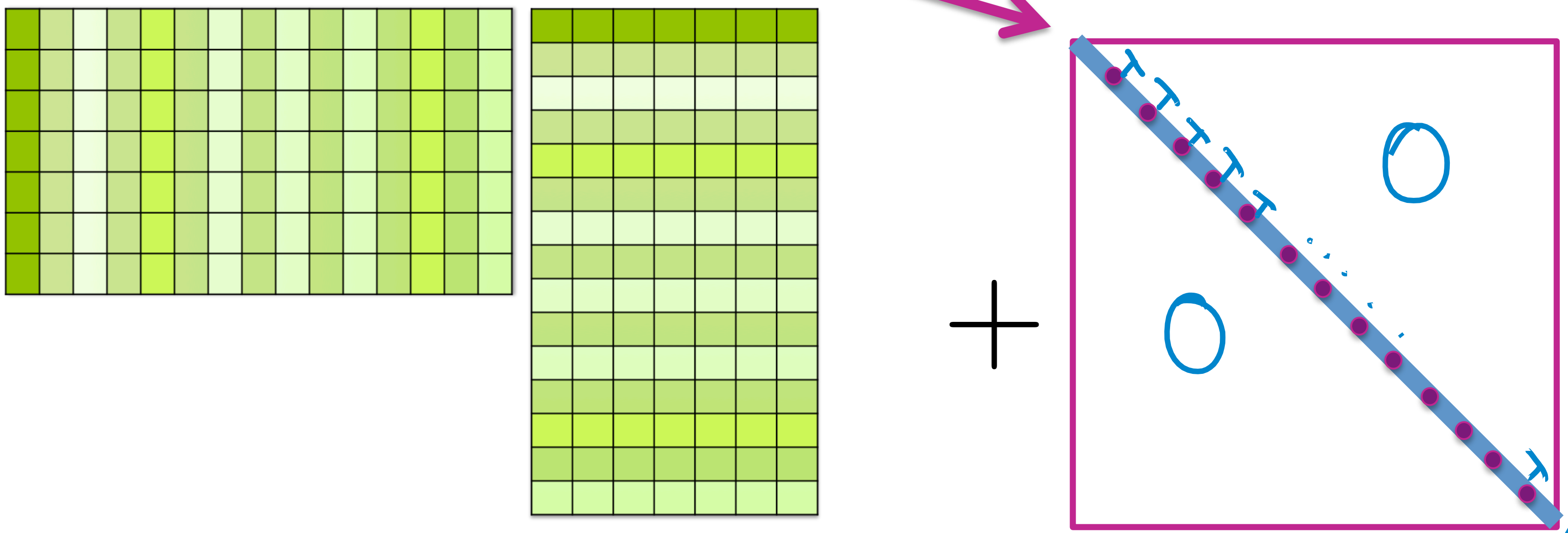
In general, (# linearly independent obs) N > D

Complexity of inverse: O(D^3)

Discussion of ridge closed-form solution

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

really important for large D
(lots of features)



Invertible if:

Always if λ>0, even if N < D

Complexity of inverse:

O(D³)…

big for large D!

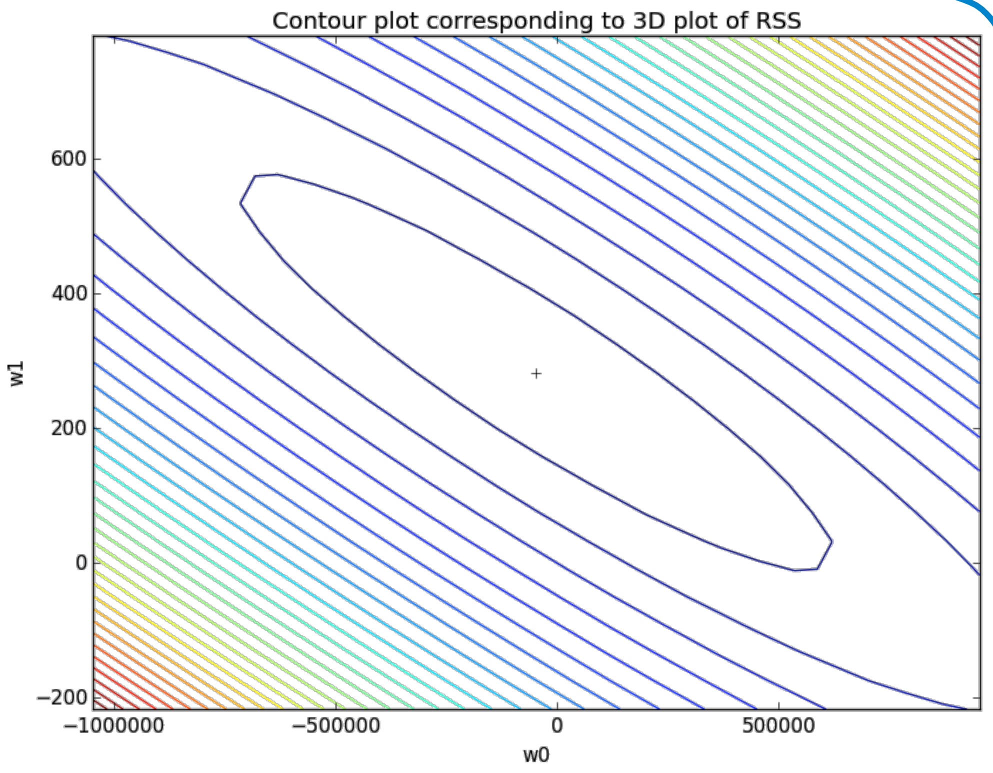
λI is making $H^TH+\lambda I$ more "regular"

→ "regularized"

Step 3, Approach 2:
Gradient descent

Elementwise ridge regression gradient descent algorithm

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



Update to j^{th} feature weight:

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

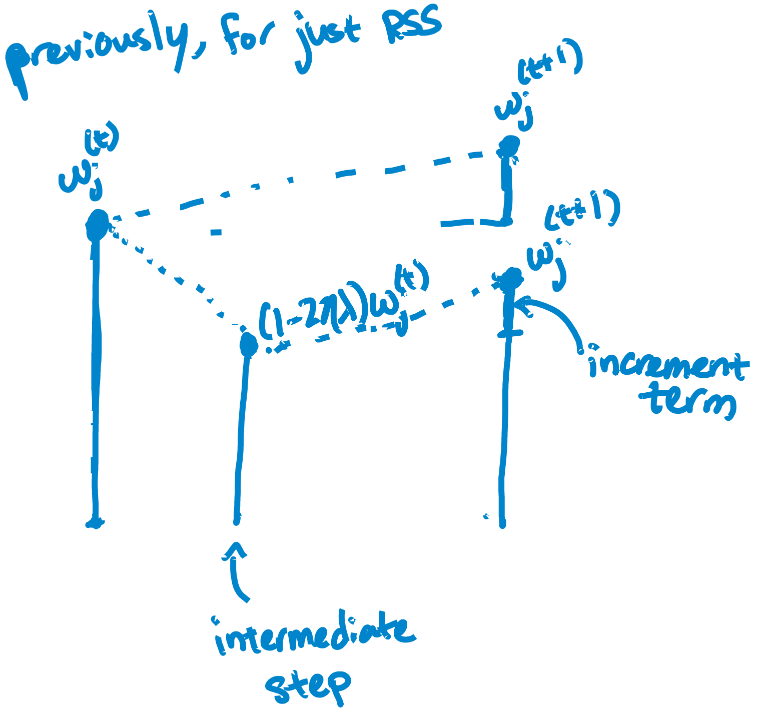
Same as before (from RSS term)

$$\left[-2 \sum_{i=1}^{N} h_j(\mathbf{x}_i)(y_i - \hat{y}_i(\mathbf{w}^{(t)})) + 2\lambda w_j^{(t)} \right]$$

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

Elementwise ridge regression gradient descent algorithm

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



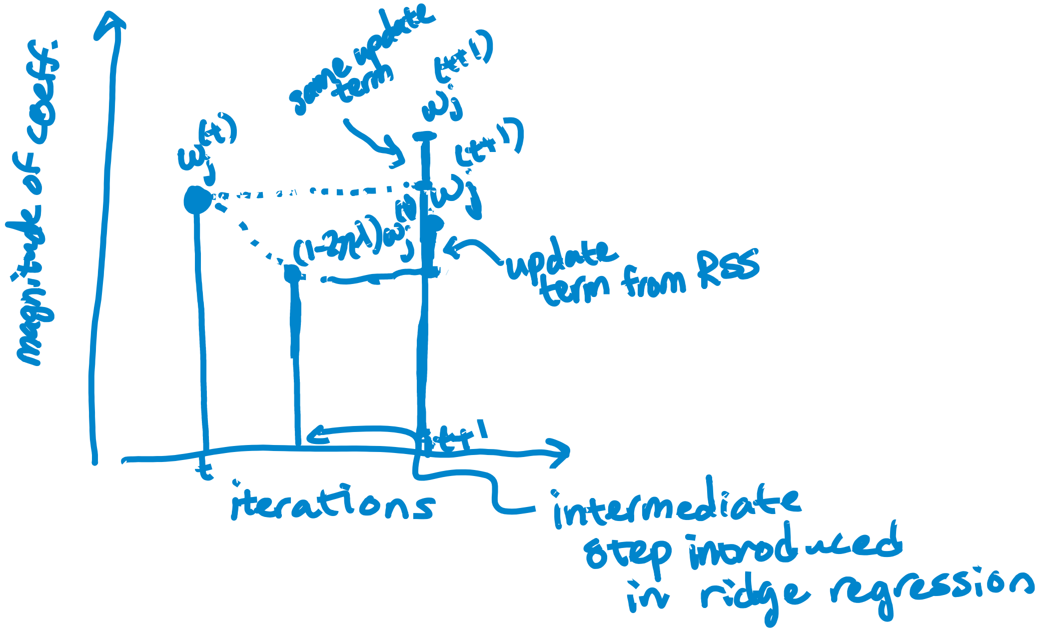
Equivalently:

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

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, just RSS (LS)



Equivalently:

$$w_j^{(t+1)} \leftarrow \Big(\underbrace{1-\overbrace{2\overset{\ge 0}{\eta}\overset{\ge 0}{\lambda}}^{2\eta\lambda<1}}_{\le 1}\Big) w_j^{(t)} + \underbrace{2\eta \sum_{i=1}^{N} h_j(\mathbf{x}_i)(y_i-\hat{y}_i(\mathbf{w}^{(t)}))}_{\text{update term from RSS}}$$

Recall previous algorithm

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

while $||\nabla RSS(\mathbf{w}^{(t)})|| > \varepsilon$

for $j=0,\ldots,D$

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

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

$t \leftarrow t + 1$

Summary of ridge regression algorithm

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

while $||\nabla RSS(\mathbf{w}^{(t)})|| > \varepsilon$

for $j=0,\ldots,D$

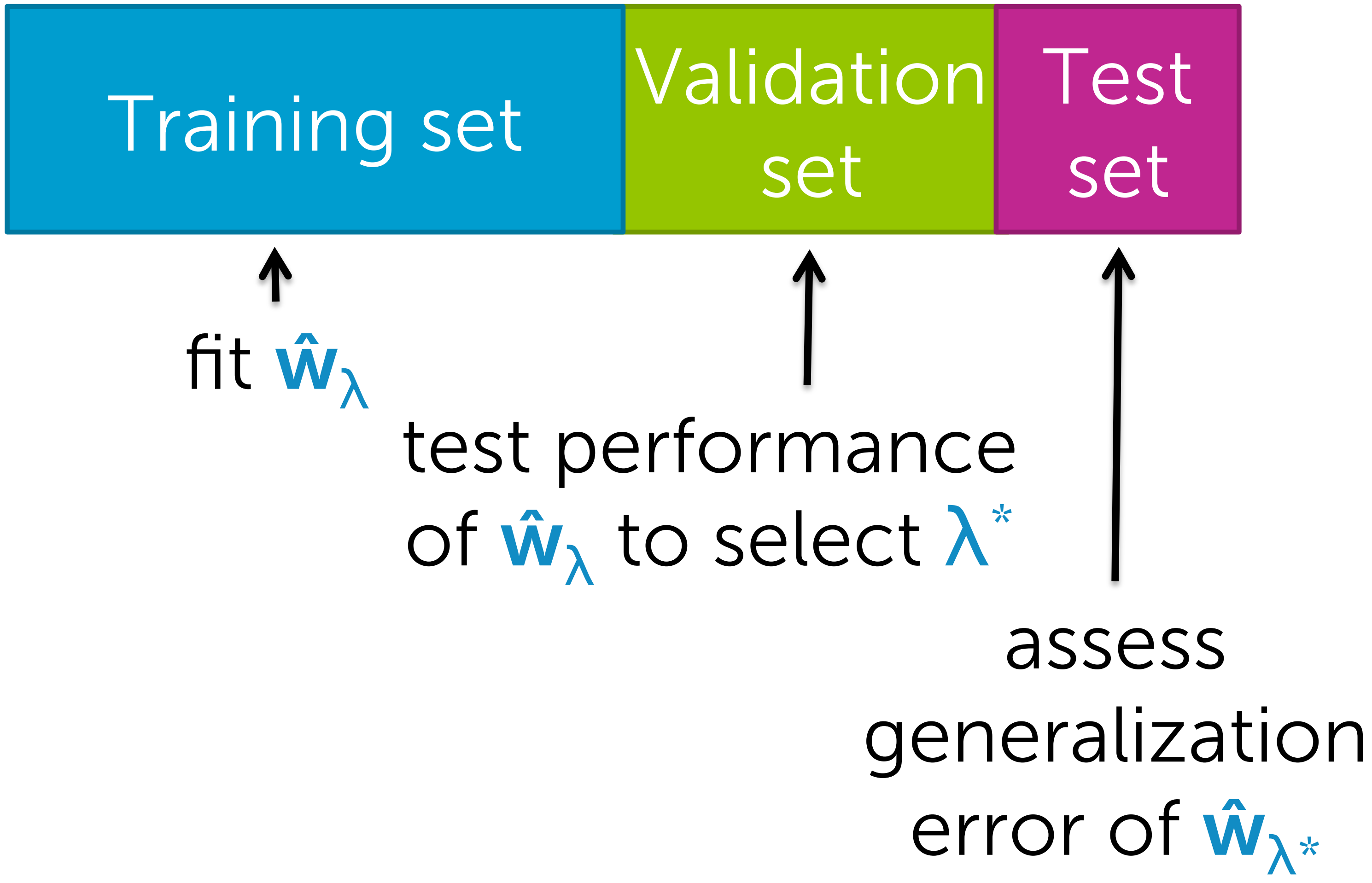
$$\text{partial}[j] = -2 \sum_{i=1}^{N} h_j(\mathbf{x}_i)(y_i-\hat{y}_i(\mathbf{w}^{(t)}))$$

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

$t \leftarrow t + 1$

How to choose λ

If sufficient amount of data…



Start with smallish dataset

All data

Still form test set and hold out

Rest of data	Test set

How do we use the other data?

Rest of data

use for both training and
validation, but not so naively

Recall naïve approach

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

Rest of data

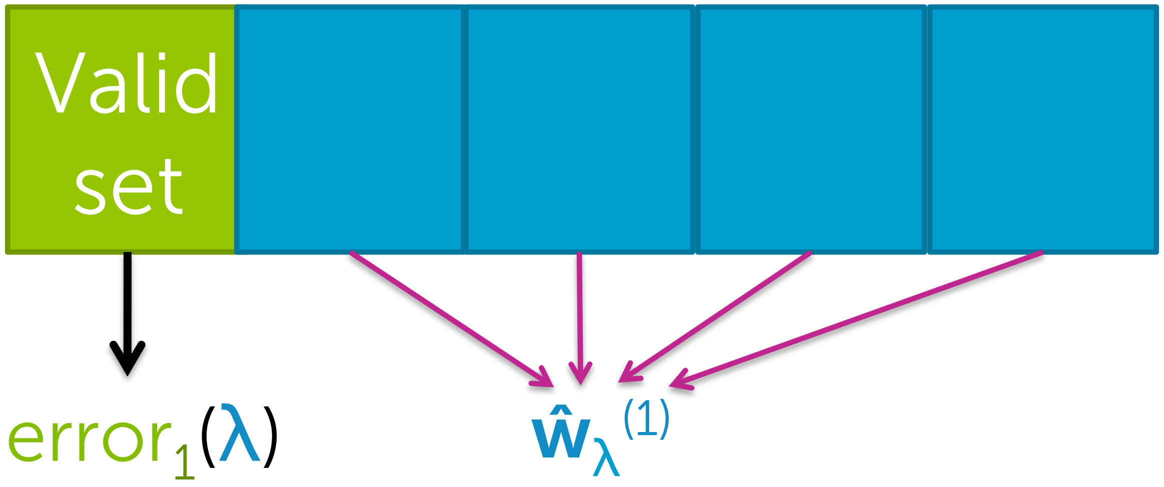
$\frac{N}{K}$ $\frac{N}{K}$ $\frac{N}{K}$ $\frac{N}{K}$ $\frac{N}{K}$

Preprocessing:

Randomly assign data to K groups

(use same split of data for all other steps)

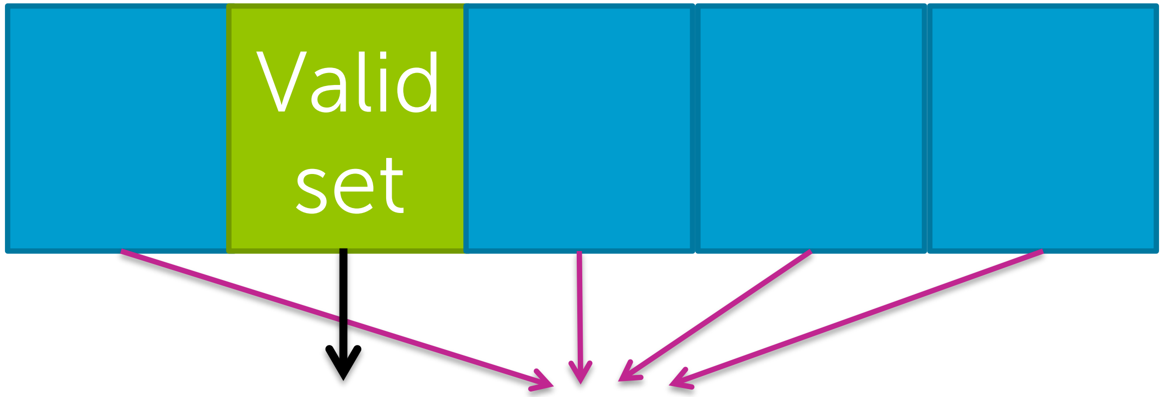
K-fold cross validation



For k=1,…,K

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

K-fold cross validation

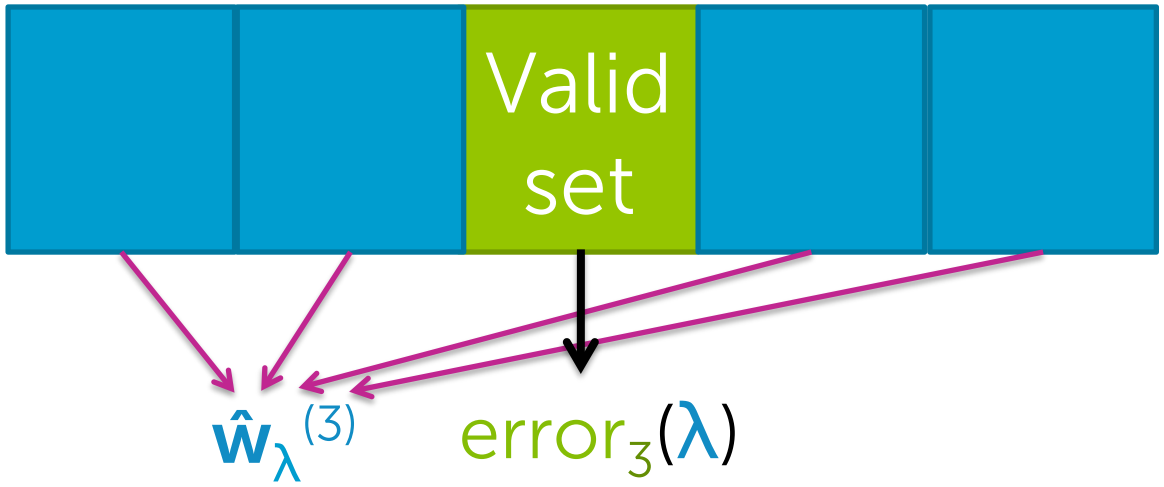


error$_2(\lambda)$ $\hat{\mathbf{w}}_\lambda^{(2)}$

For k=1,…,K

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

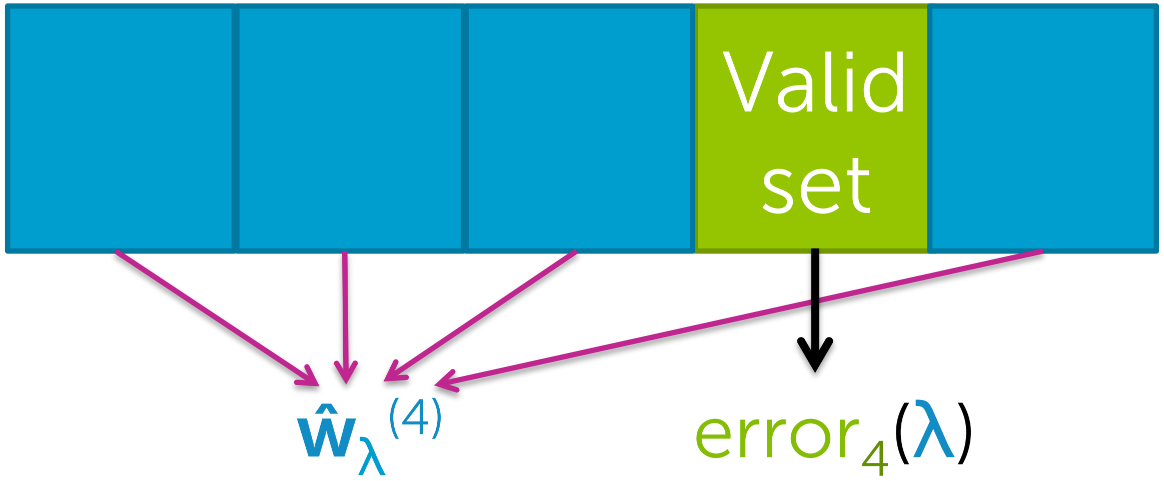
K-fold cross validation



For k=1,…,K

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

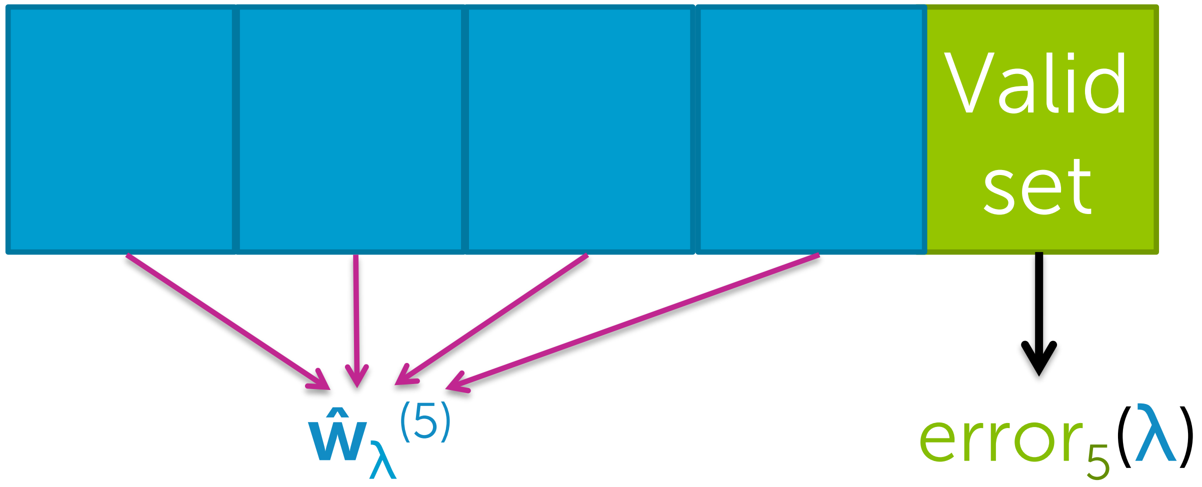
K-fold cross validation



For k=1,…,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

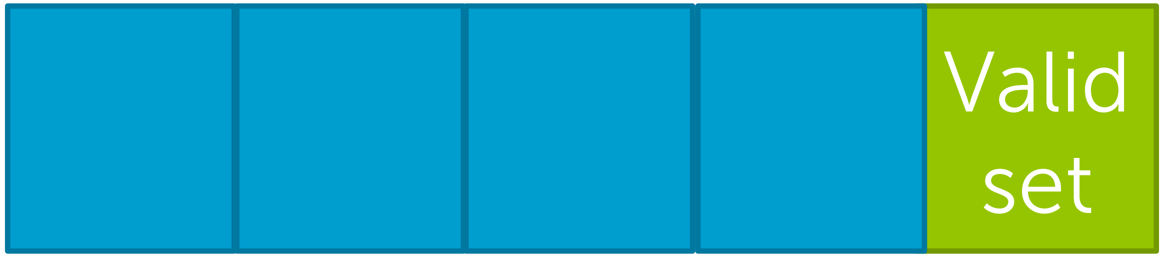


For k=1,…,K

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

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

K-fold cross validation



Repeat procedure for each choice of λ

$\boxed{\text{Choose } \lambda^* \text{ to minimize } CV(\lambda)}$

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 λ

Typically, K=5 or 10

5-fold CV

10-fold CV

How to handle the intercept

Recall multiple regression model

Model:

$$y_i = w_0 h_0(\mathbf{x}_i) + w_1 h_1(\mathbf{x}_i) + \ldots + w_D h_D(\mathbf{x}_i) + \varepsilon_i$$

$$= \sum_{j=0}^{D} w_j h_j(\mathbf{x}_i) + \varepsilon_i$$

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., $\mathbf{x}[2]$

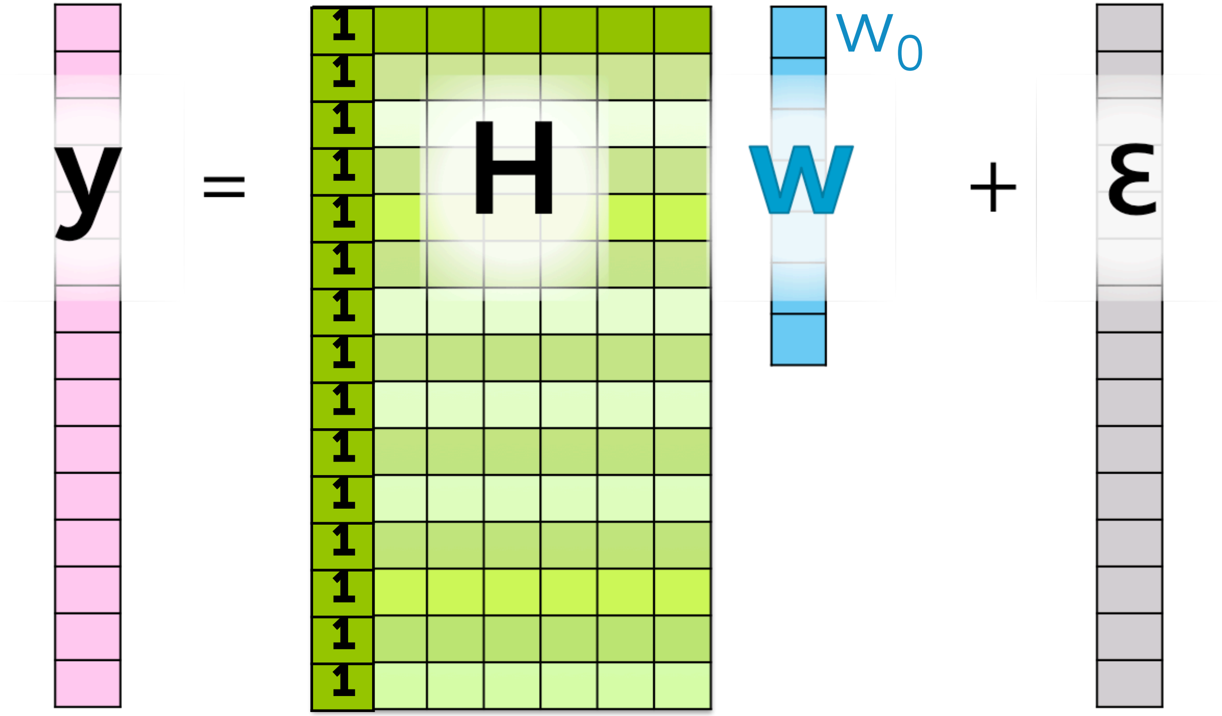
…

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

If constant feature…

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

In matrix notation for N observations:



Do we penalize intercept?

Standard ridge regression cost:

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

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

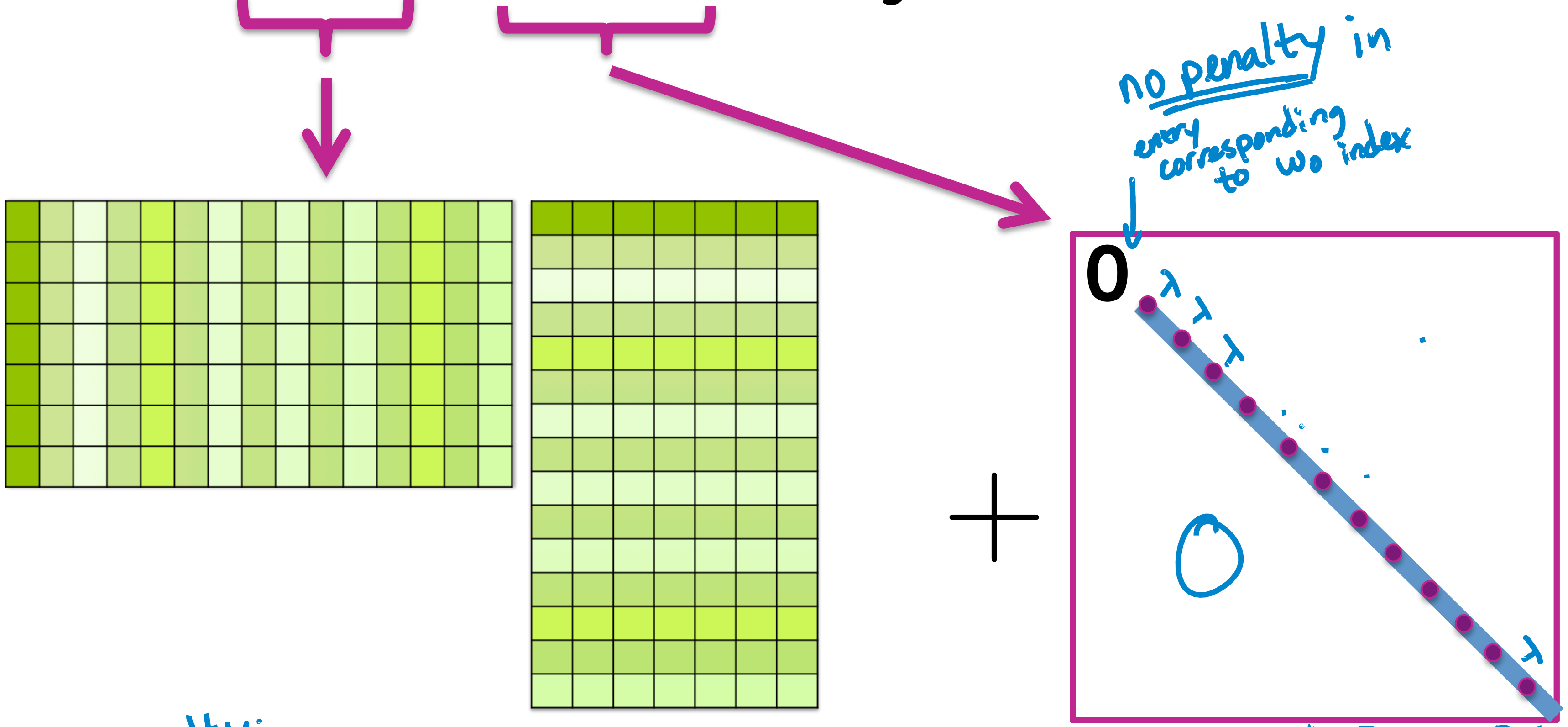
Modified ridge regression cost:

$$\text{RSS}(w_{0,} \mathbf{w}_{rest}) + \lambda ||\mathbf{w}_{rest}||_2^2$$

How to implement this in practice?

Option 1: Don't penalize intercept – Closed-form solution –

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



new penalty:

$\lambda \, w_{rest}^T w_{rest}$

gradient:

$$2\lambda w_{rest} = 2\lambda \underset{(D-1)(D-1)}{I} \underset{D-1 \times 1}{w_{rest}} \rightarrow 2\lambda \underset{D\times D}{\begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}} \underset{D\times 1}{\begin{bmatrix} w_0 \\ w_{rest} \end{bmatrix}}$$

Option 1: Don't penalize intercept – Gradient descent algorithm –

while $||\nabla \mathrm{RSS}(\mathbf{w}^{(t)})|| > \varepsilon$

for j=0,…,D

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

if j==0

$w_0^{(t+1)} \leftarrow w_0^{(t)} - \eta\,$partial[j] ← old LS update (no shrinkage to w_0)

else ← for all other features

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

$t \leftarrow t + 1$

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 λ 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 λ