Lasso Regression:

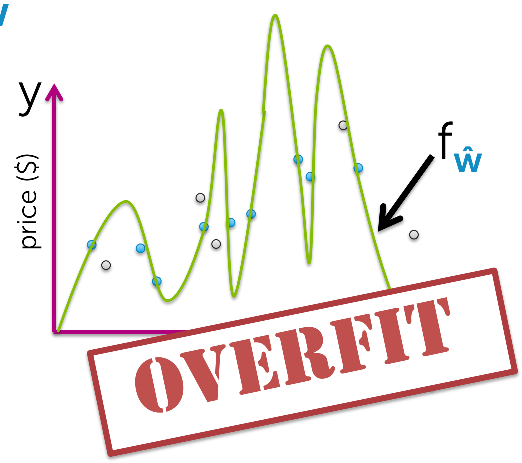
Regularization for feature selection

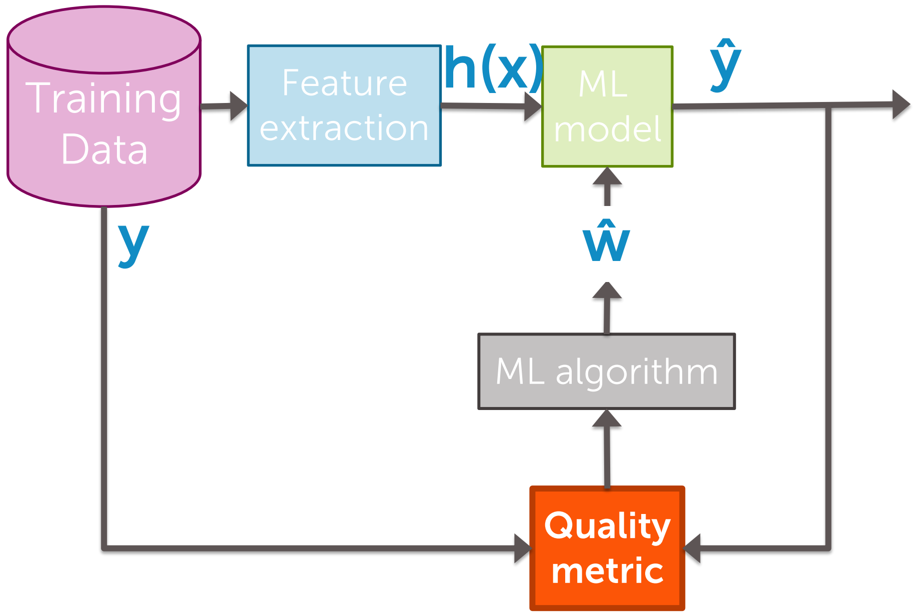
CSE 416: Machine Learning
Emily Fox
University of Washington
April 12, 2018

Symptom of overfitting

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

very large coefficients (+/-)





Consider specific total cost

Want to balance:

i. How well function fits data

ii. Magnitude of coefficients

Total cost =

$$\underbrace{\text{measure of fit}}_{\text{RSS}(\mathbf{w})} + \underbrace{\text{measure of magnitude of coefficients}}_{\|\mathbf{w}\|_2^2 \,=\, \sum_{j=0}^{D} w_j^2}$$

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)

Measure of magnitude of regression coefficient

What summary # is indicative of size of regression coefficients?

- Sum? $w_0 = 1{,}527{,}301$ $w_1 = -1{,}605{,}253$ $w_0 + w_1 =$ small # X
- Sum of absolute value? $\sum_{j=0}^{D} |w_j| \triangleq ||\mathbf{w}||_1$ $\mathbf{w} = [w_0\ w_1 \cdots w_D]$ L_1 norm... discuss ~~next~~ this lecture
- Sum of squares (L_2 norm) ridge $\sum_{j=0}^{D} w_j^2 \triangleq ||\mathbf{w}||_2^2$ focus of ~~this~~ last lecture

Feature selection task

Why might you want to perform feature selection?

Efficiency:

- If size($\mathbf{w}$) = 100B, each prediction is expensive
- If $\hat{\mathbf{w}}$ $\boxed{\text{sparse}}$, computation only depends on # of non-zeros

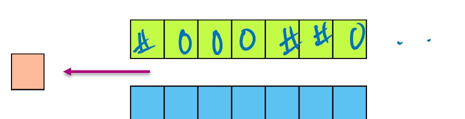
many zeros

est. coeff.

pred. value

$$\hat{y}_i = \sum_{\hat{w}_j \neq 0} \hat{w}_j \, h_j(\mathbf{x}_i)$$

only sum over features w/ non-zero weights



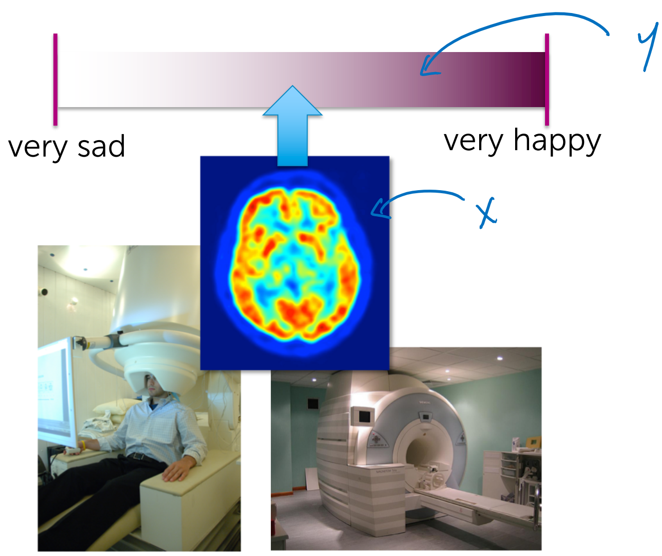
Interpretability:

- Which features are relevant for prediction?

Sparsity: Housing application

Lot size	Dishwasher
Single Family	Garbage disposal
Year built	Microwave
Last sold price	Range / Oven
Last sale price/sqft	Refrigerator
Finished sqft	Washer
Unfinished sqft	Dryer
Finished basement sqft	Laundry location
# floors	Heating type
Flooring types	Jetted Tub
Parking type	Deck
Parking amount	Fenced Yard
Cooling	Lawn
Heating	Garden
Exterior materials	Sprinkler System
Roof type	⋮
Structure style	

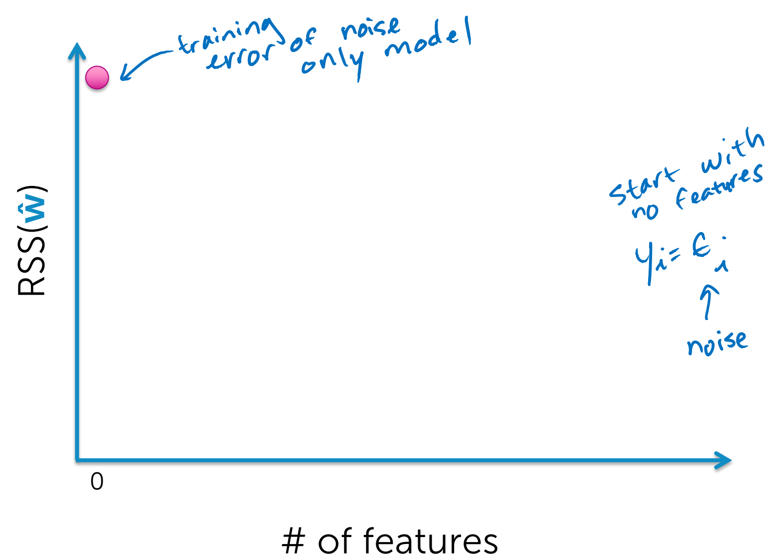
Sparsity: Reading your mind



Activity in which brain regions can predict happiness?

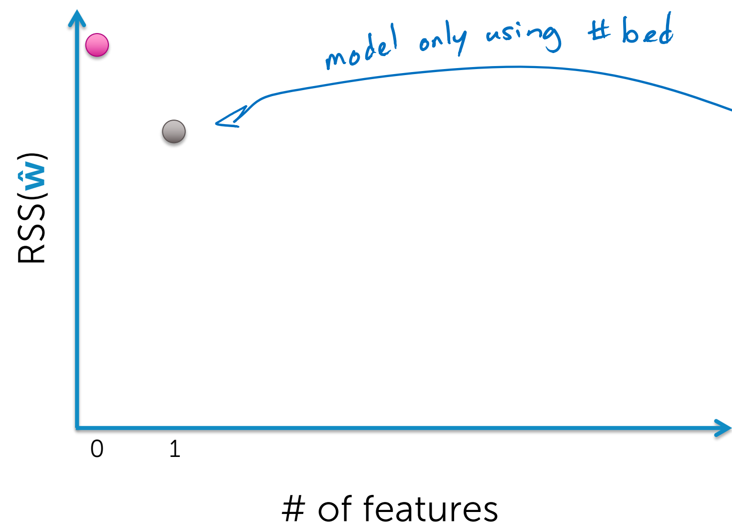
Option 1: All subsets or greedy variants

Find best model of size: 0



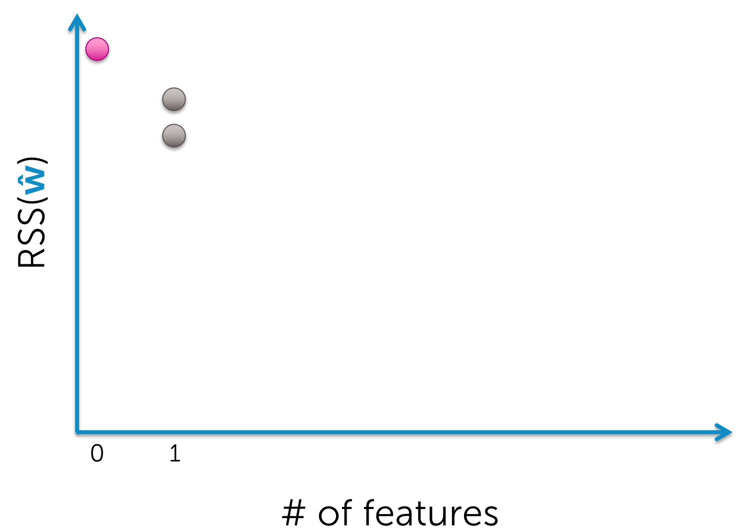
- ~~# bedrooms~~
- ~~# bathrooms~~
- ~~sq.ft. living~~
- ~~sq.ft. lot~~
- ~~floors~~
- ~~year built~~
- ~~year renovated~~
- ~~waterfront~~

Find best model of size: 1



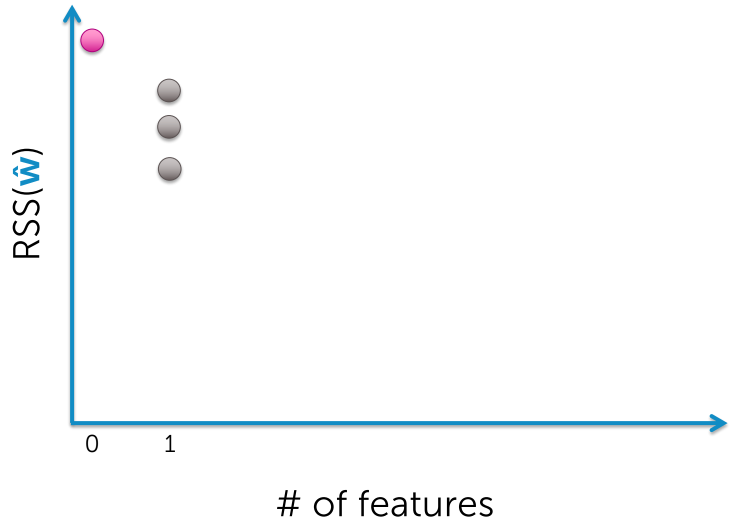
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



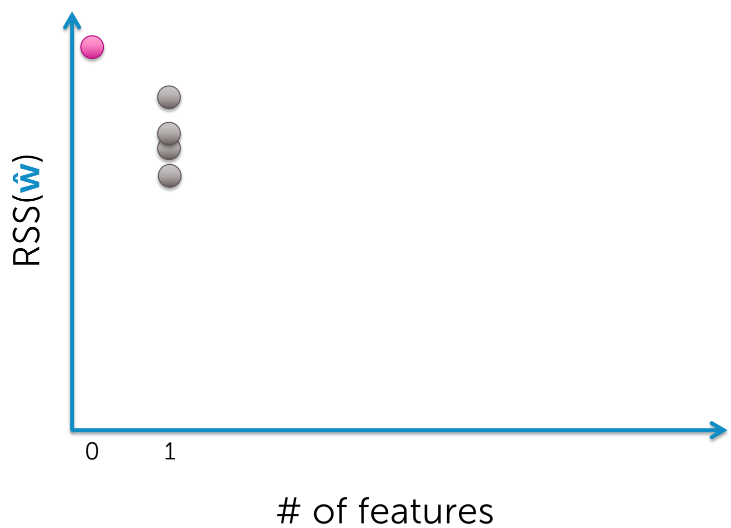
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



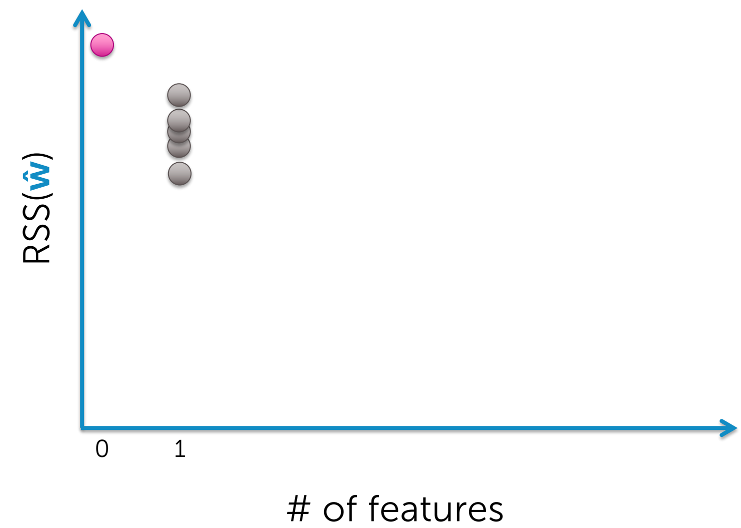
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



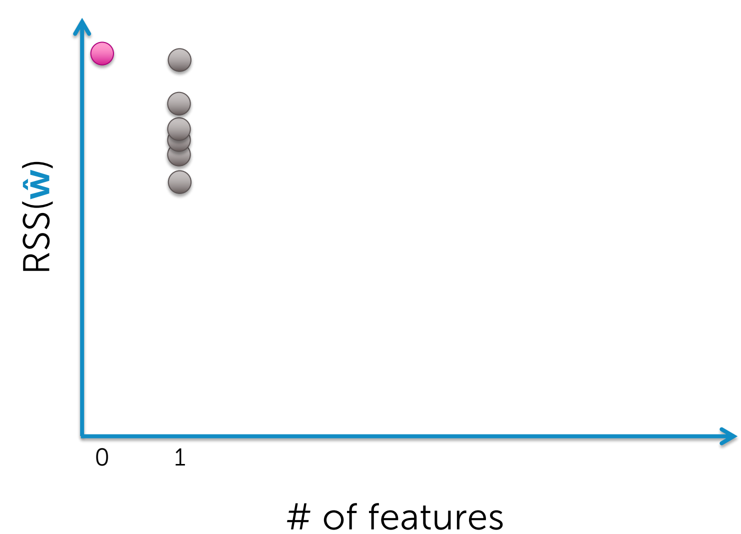
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



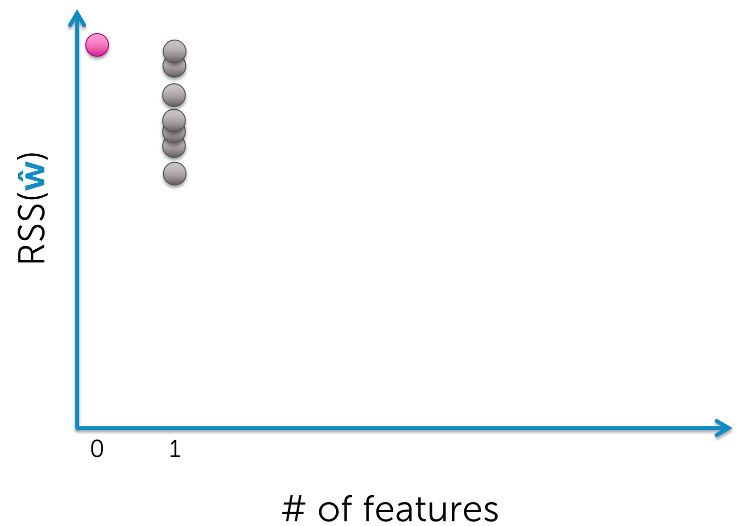
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



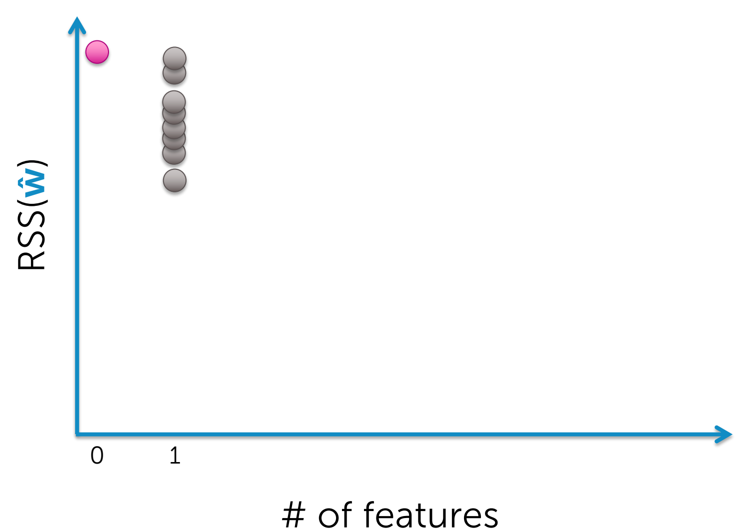
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



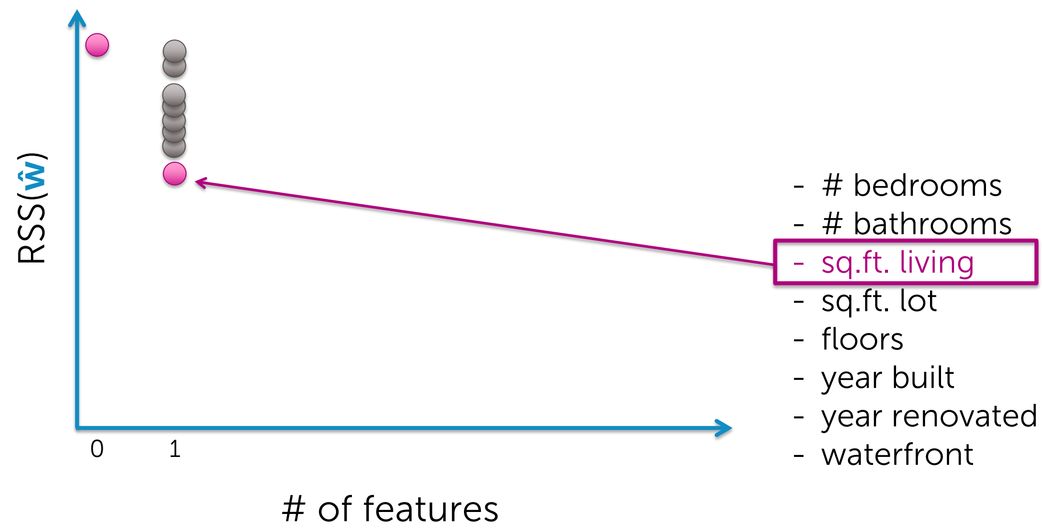
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1

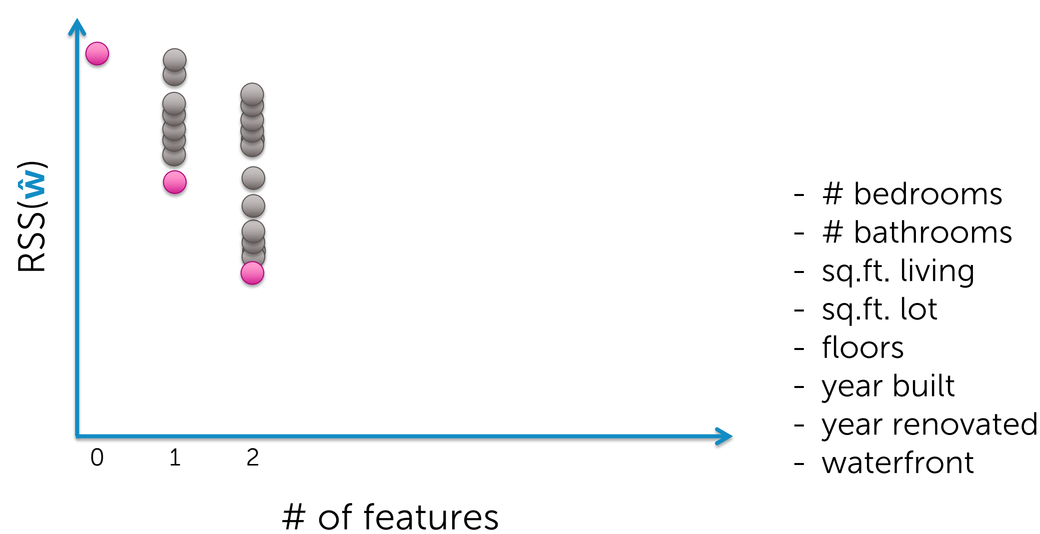


- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

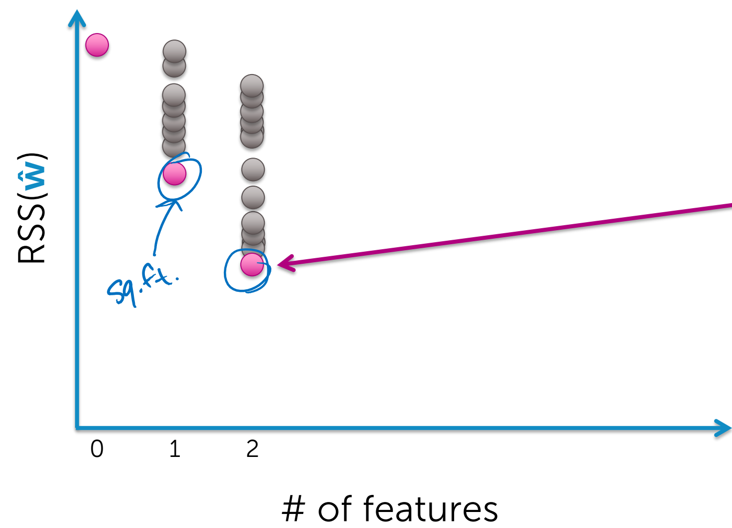
Find best model of size: 1



Find best model of size: 2

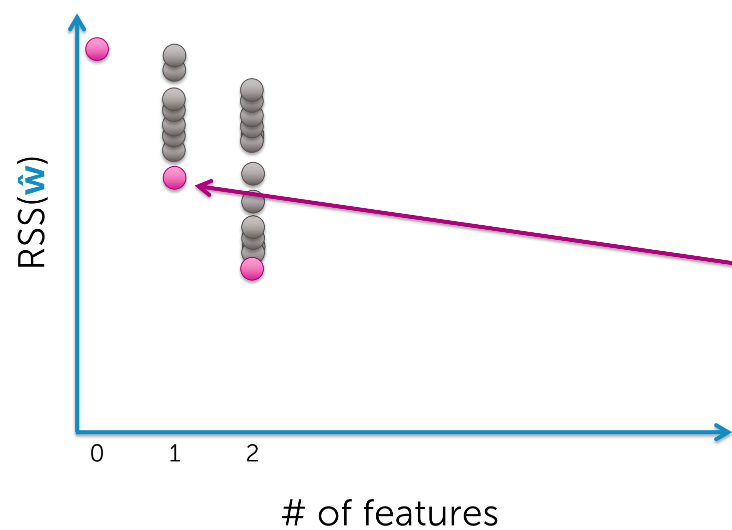


Note: Not necessarily nested!



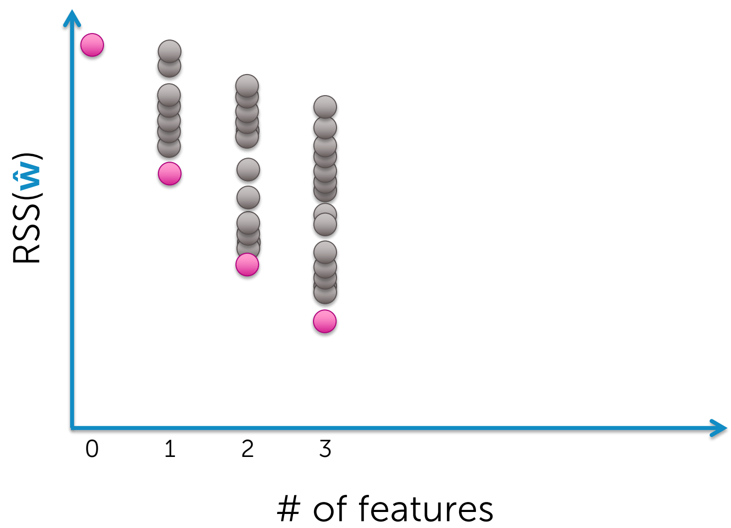
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Note: Not necessarily nested!



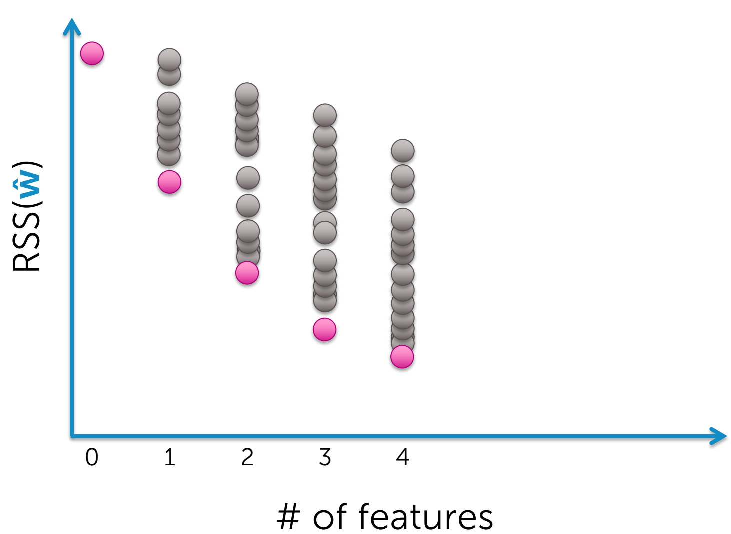
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 3



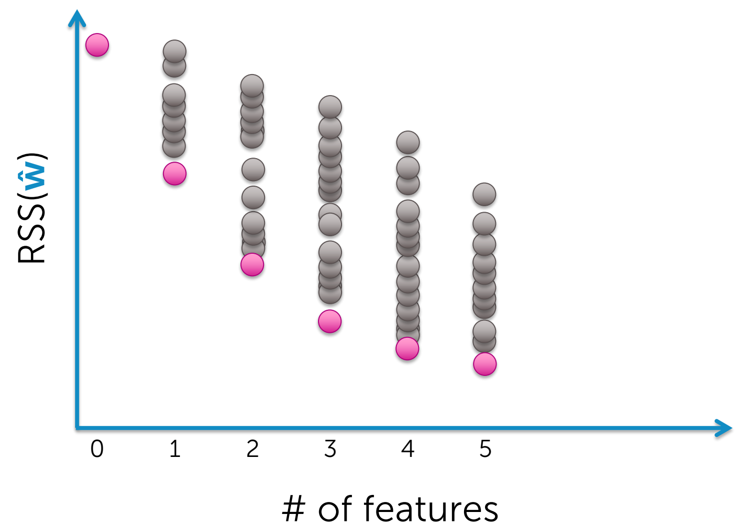
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 4



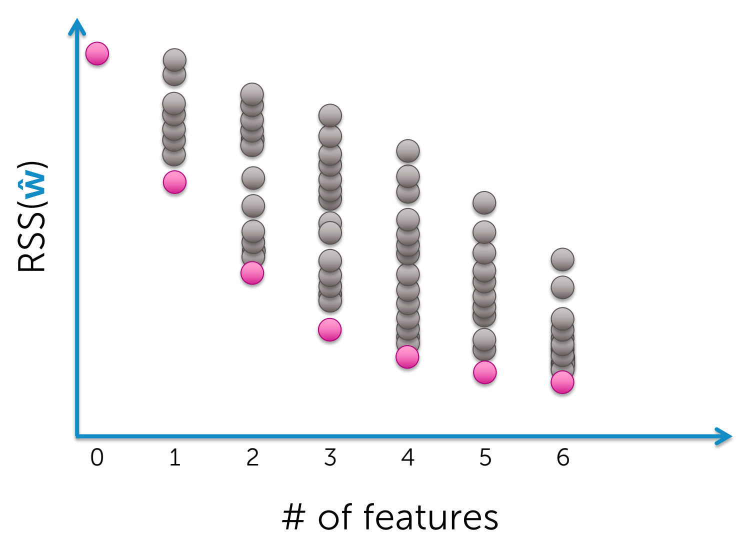
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 5



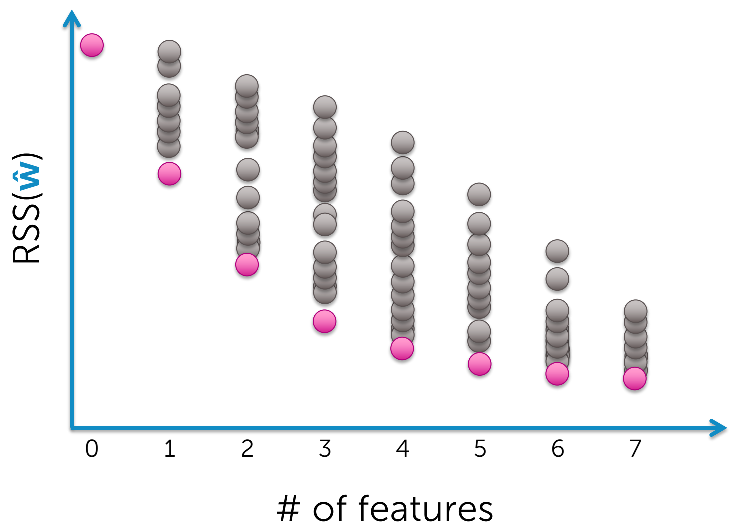
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 6



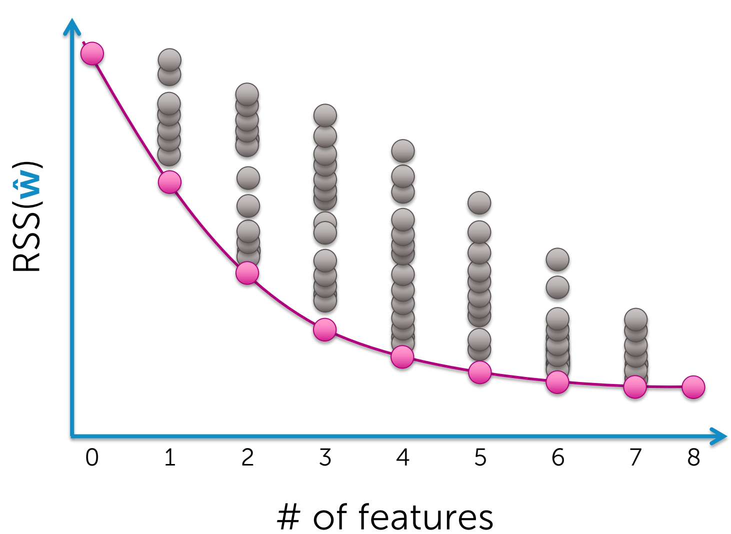
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 7



- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 8



- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Choosing model complexity?

Option 1: Assess on validation set

Option 2: Cross validation

Option 3+: Other metrics for penalizing model complexity like BIC…

Complexity of "all subsets"

How many models were evaluated?

– each indexed by features included

0 if "no"
1 if "yes"

feat 0 feat 1 … feat D

$y_i = \varepsilon_i$	[0 0 0 … 0 0 0]
$y_i = w_0 h_0(\mathbf{x}_i) + \varepsilon_i$	[1 0 0 … 0 0 0]
$y_i = w_1 h_1(\mathbf{x}_i) + \varepsilon_i$	[0 1 0 … 0 0 0]
⋮	⋮
$y_i = w_0 h_0(\mathbf{x}_i) + w_1 h_1(\mathbf{x}_i) + \varepsilon_i$	[1 1 0 … 0 0 0]
⋮	⋮
$y_i = w_0 h_0(\mathbf{x}_i) + w_1 h_1(\mathbf{x}_i) + \ldots + w_D h_D(\mathbf{x}_i) + \varepsilon_i$	[1 1 1 … 1 1 1]
	2 2 2 … 2

2^{D+1}

$2^8 = 256$
$2^{30} = 1{,}073{,}741{,}824$
$2^{1000} = 1.071509 \times 10^{301}$
2^{100B} = HUGE!!!!!!

Typically, computationally infeasible

Greedy algorithms

Forward stepwise:
Starting from simple model and iteratively add features most useful to fit

Backward stepwise:
Start with full model and iteratively remove features least useful to fit

Combining forward and backward steps:
In forward algorithm, insert steps to remove features no longer as important

Lots of other variants, too.

Option 2: Regularize

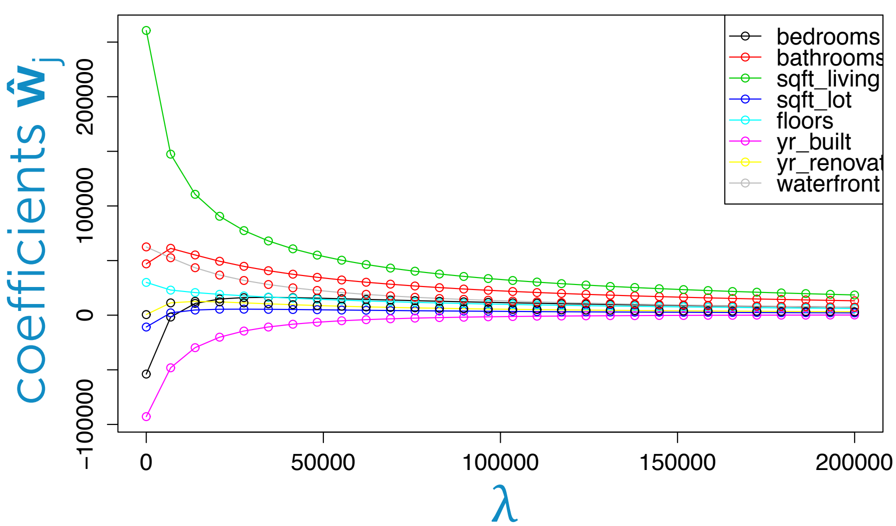
Ridge regression: L_2 regularized regression

Total cost =

$$\underbrace{\text{measure of fit}}_{RSS(\mathbf{w})} + \lambda \underbrace{\text{measure of magnitude of coefficients}}_{||\mathbf{w}||_2^2 = w_0^2 + \ldots + w_D^2}$$

Encourages **small weights**
but not exactly 0

Coefficient path – **ridge**



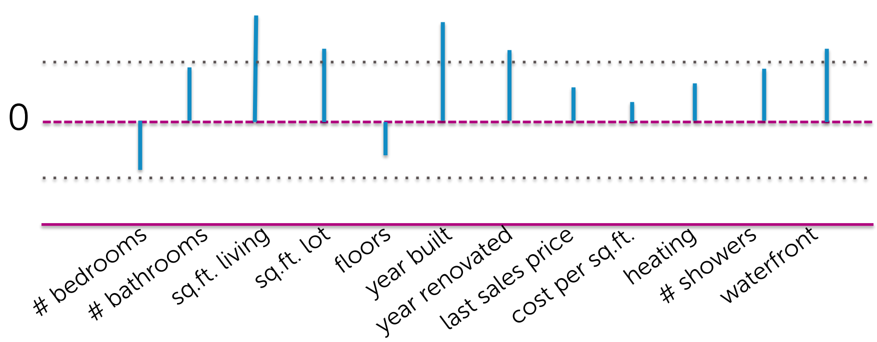
Using regularization for feature selection

Instead of searching over a **discrete** set of solutions, can we use regularization?

- Start with full model (all possible features)
- "Shrink" some coefficients *exactly* to 0
  - i.e., knock out certain features
- Non-zero coefficients indicate "selected" features

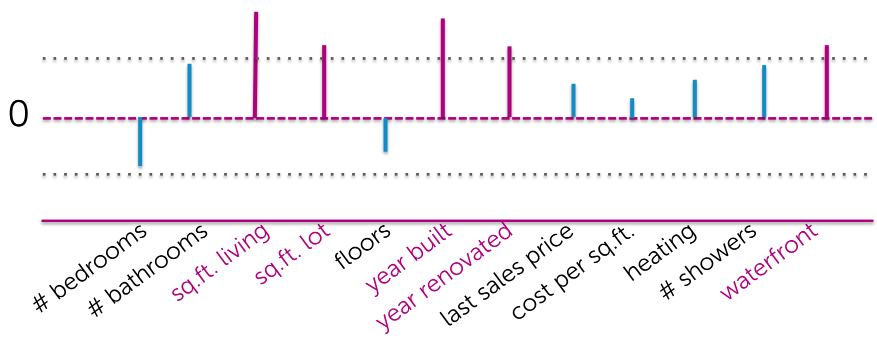
Thresholding ridge coefficients?

Why don't we just set small ridge coefficients to 0?



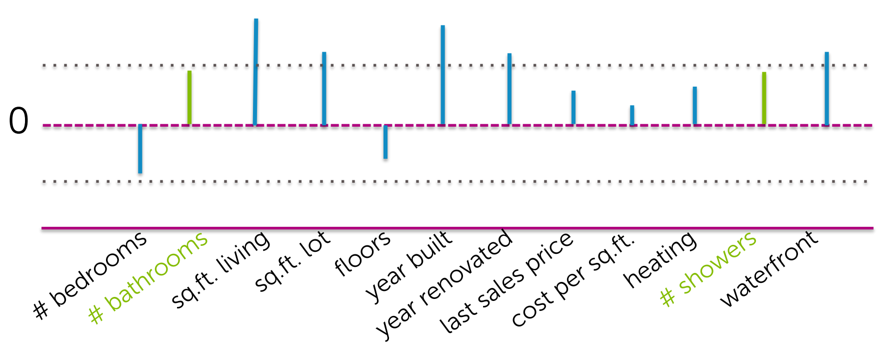
Thresholding ridge coefficients?

Selected features for a given threshold value



Thresholding ridge coefficients?

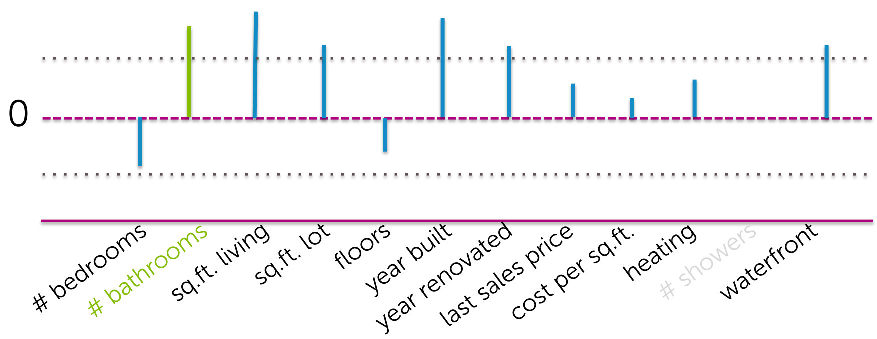
Let's look at two related features…



Nothing measuring bathrooms was included!

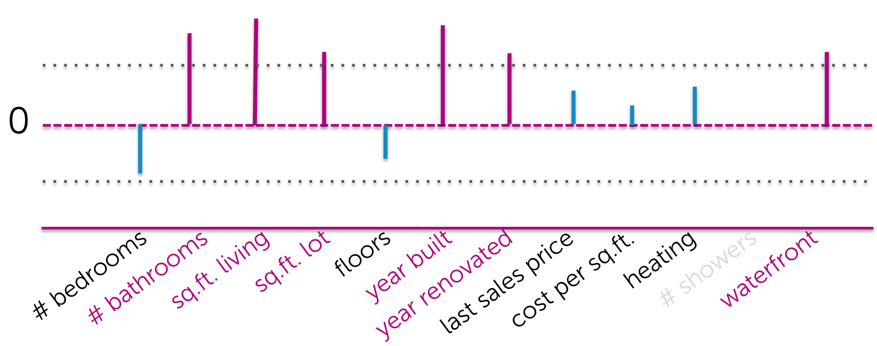
Thresholding ridge coefficients?

If only one of the features had been included…



Thresholding ridge coefficients?

Would have included bathrooms in selected model



Can regularization lead directly to sparsity?

Try this cost instead of ridge…

Total cost =

$$\underbrace{\text{measure of fit}}_{\text{RSS}(\mathbf{w})} + \lambda \underbrace{\text{measure of magnitude of coefficients}}_{||\mathbf{w}||_1 = |w_0| + \ldots + |w_D|}$$

Leads to **sparse** solutions!

Lasso regression
(a.k.a. L_1 regularized regression)

43 ©2018 Emily Fox CSE/STAT 416: Intro to Machine Learning

Lasso regression: L_1 regularized regression

Just like ridge regression, solution is governed by a continuous parameter λ

$$\text{RSS}(\mathbf{w}) + \lambda||\mathbf{w}||_1$$

tuning parameter = balance of fit and **sparsity**

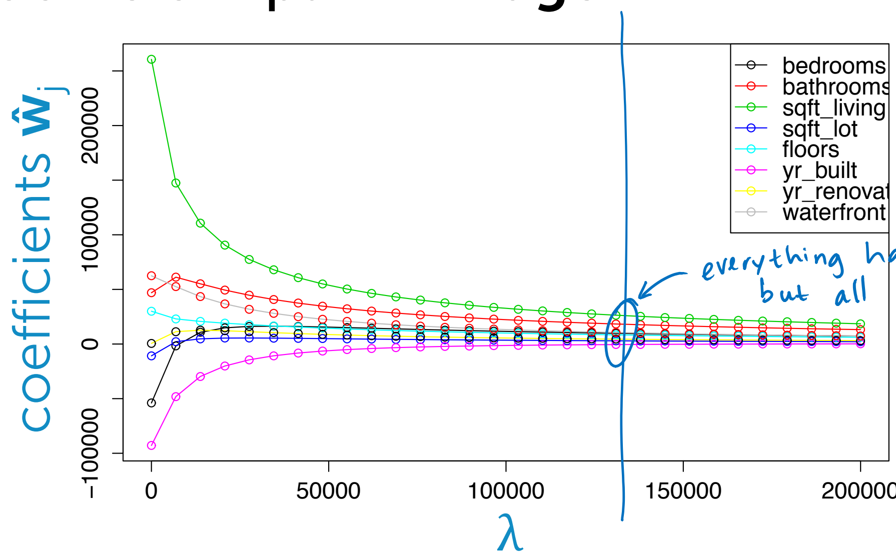
If λ=0: $\hat{w}^{lasso} = \hat{w}^{LS}$ (unreg. soln)

If $\lambda=\infty$: $\hat{w}^{lasso} = 0$

If λ in between: $0 \leq ||\hat{w}^{lasso}||_1 \leq ||\hat{w}^{LS}||_1$

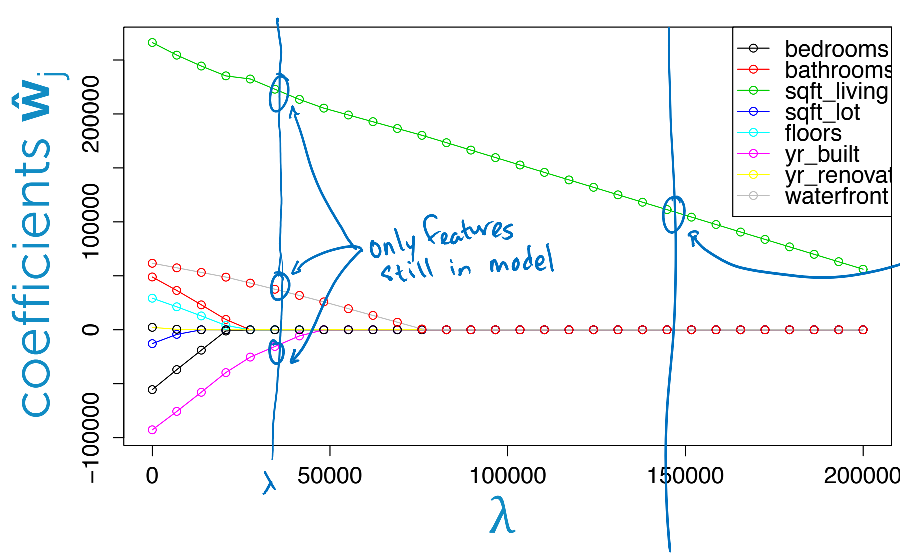
44 ©2018 Emily Fox CSE/STAT 416: Intro to Machine Learning

Coefficient path – **ridge**



everything has $|\hat{w}_j|>0$, but all $\hat{w}_j$ small for large λ

Coefficient path – **lasso**



large weight on sq.ft.living when everything else is out of the model

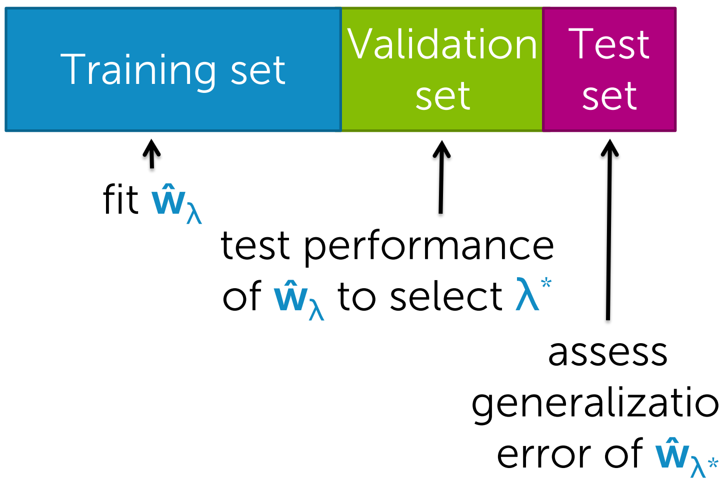
Revisit polynomial fit demo

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

Will consider a few settings of λ …

How to choose λ: Cross validation

If sufficient amount of data…



Start with smallish dataset

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?

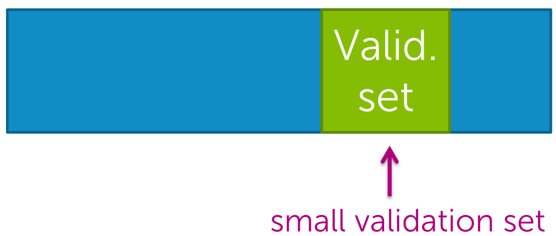
No

Choosing the validation set

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

Can use any data subset

Choosing the validation set

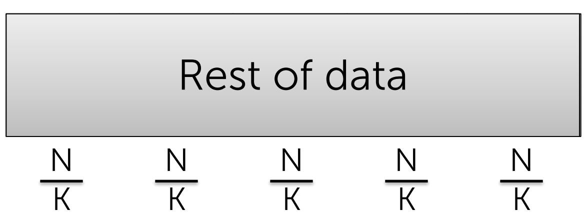


Which subset should I use?

ALL!

average performance
over all choices

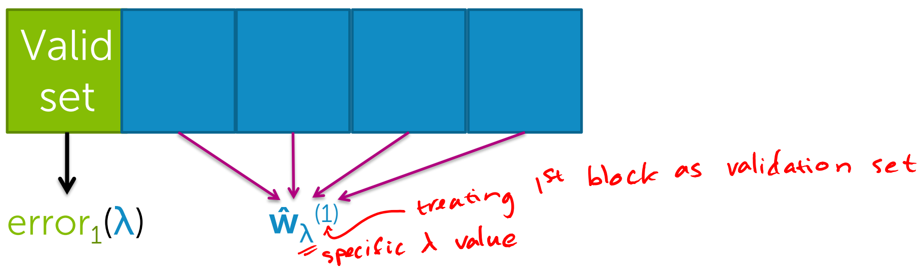
K-fold cross validation



Preprocessing: Randomly assign data to K groups

(use same split of data for all other steps)

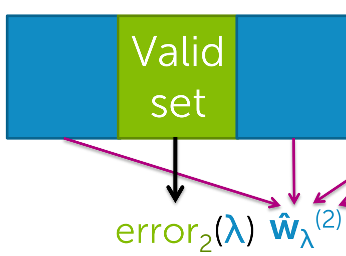
K-fold cross validation



For k=1,…,K

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

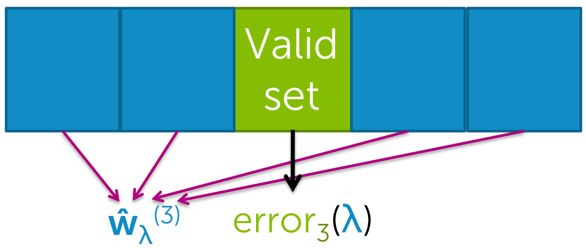
K-fold cross validation



For k=1,…,K

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

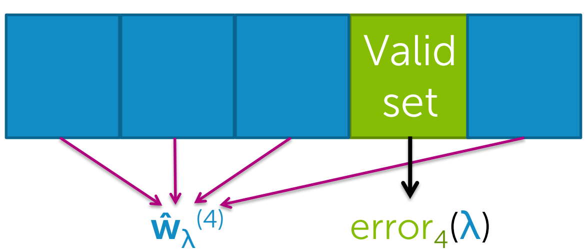
K-fold cross validation



For k=1,…,K

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

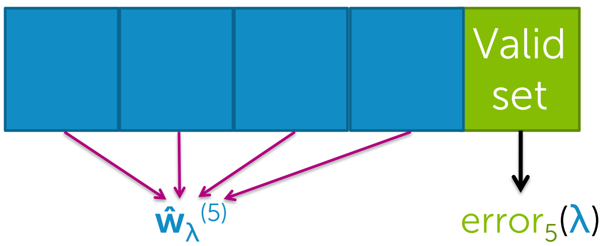
K-fold cross validation



For k=1,…,K

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

K-fold cross validation

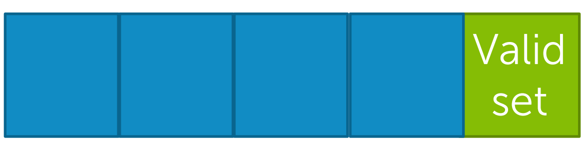


For k=1,…,K

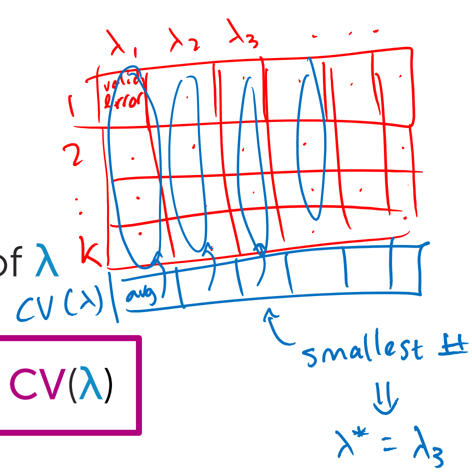
1. Estimate $\hat{\mathbf{w}}_\lambda^{(k)}$ on the training blocks
2. Compute error on validation block: $\text{error}_k(\lambda)$

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

Choosing λ via cross validation for lasso

Cross validation is choosing the λ that provides best predictive accuracy

Tends to favor less sparse solutions, and thus smaller λ, than optimal choice for feature selection

c.f., "Machine Learning: A Probabilistic Perspective", Murphy, 2012 for further discussion

Practical concerns with lasso

Debiasing lasso

Lasso shrinks coefficients relative to LS solution
→ more bias, less variance

Can reduce bias as follows:
1. Run lasso to select features
2. Run least squares regression with only selected features

"Relevant" features no longer shrunk relative to LS fit of same reduced model

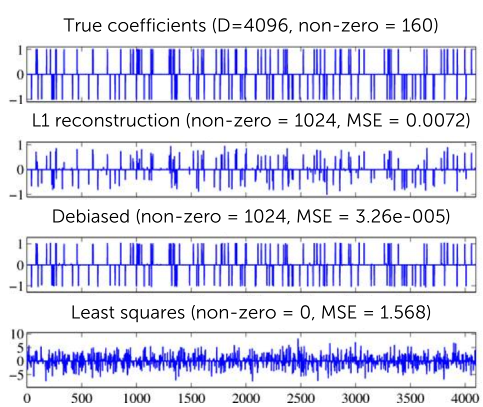


Figure used with permission of Mario Figueiredo
(*captions modified to fit course*)

Issues with standard lasso objective

1. With group of highly correlated features, lasso tends to select amongst them arbitrarily
   – Often prefer to select all together
2. Often, empirically ridge has better predictive performance than lasso, but lasso leads to sparser solution

Elastic net aims to address these issues

– hybrid between lasso and ridge regression
– uses L_1 and L_2 penalties

See Zou & Hastie '05 for further discussion

Summary for feature selection and lasso regression

Impact of feature selection and lasso

Lasso has changed machine learning, statistics, & electrical engineering

But, for feature selection in general, be careful about interpreting selected features

- selection only considers features included
- sensitive to correlations between features
- result depends on algorithm used
- there are theoretical guarantees for lasso under certain conditions

What you can do now…

- Describe "all subsets" and greedy variants for feature selection
- Analyze computational costs of these algorithms
- Formulate lasso objective
- Describe what happens to estimated lasso coefficients as tuning parameter λ is varied
- Interpret lasso coefficient path plot
- Contrast ridge and lasso regression
- Implement K-fold cross validation to select lasso tuning parameter λ