

## Logistics

- Question from a student: can we get access to the recording from the past offerings of CSE441?
  - For FERPA reasons, we are not able to release them.
- We put other public video lectures in the course website that covers a free online textbook
- Office hours are there to help you not just with homework for more generally.

# Lecture 8: Model selection using Cross-validation

---

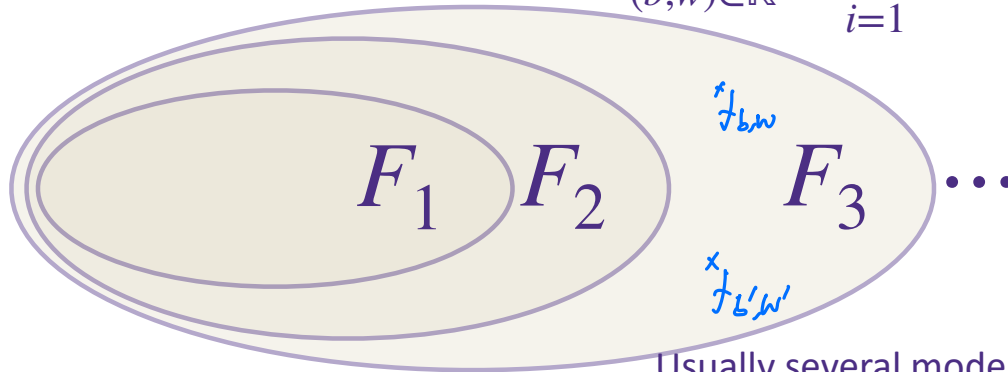
- How do we do Hyper-parameter tuning?



# Parameter and hyper-parameter

- A **model class** is set of functions, each function is indexed by its **parameters** representing the function
  - e.g., a model class  $F_p = \{\text{all degree-}p \text{ polynomials in } \mathbb{R}\}$   
each function in that class is represented (or indexed) by parameter  $(b, w) \in \mathbb{R}^{p+1}$ , which can be also written more explicitly as  $\mathbb{R} \times \mathbb{R}^p$
- $F_p = \{f_{b,w} : \mathbb{R} \rightarrow \mathbb{R} \mid f_{b,w}(x) = b + w_1x + \dots + w_px^p, \text{ for some } (b, w) \in \mathbb{R}^{1+p}\}$ 
  - **Parameter** is what is optimized when training a model

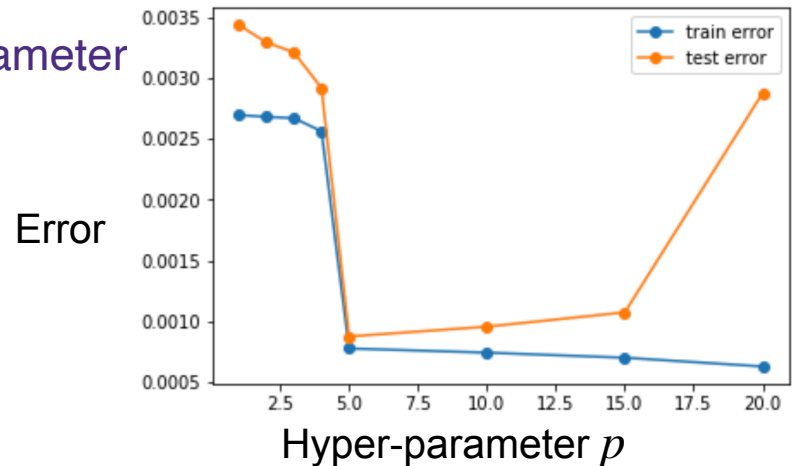
e.g.,  $(\hat{b}_p, \hat{w}_p) = \arg \min_{(b,w) \in \mathbb{R}^{1+p}} \sum_{i=1}^n (y_i - (h(x_i)^T w + b))^2$



Usually several model classes form a nested hierarchy of classes with increasing complexities

# Parameter and hyper-parameter

- **A family of model classes** is a set of model classes, each class indexed by its **hyper-parameter** representing a model class
    - e.g., a family  $\mathcal{F} = \{F_p \mid p \in \{1, 2, \dots\}\}$  is a set of model classes with hyper-parameter  $p \in \{1, 2, \dots\}$ , where  $F_p$  is a class of degree- $p$  polynomials
    - Hyper-parameter is usually fixed during training
- e.g.,  $\hat{f}_p = \arg \min_{f \in F_p} \sum_{i=1}^n (y_i - f(x_i))^2$
- And we run multiple training for multiple choices of the hyper-parameter



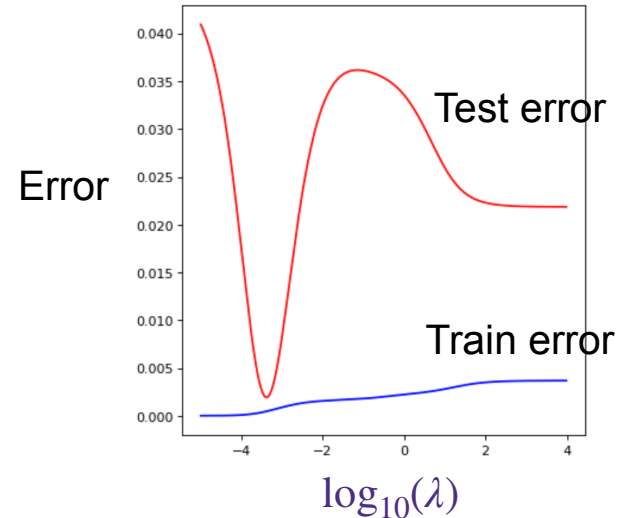
# Hyper-parameter for ridge regression

$$w_{\text{ridge}} = \arg \min_w \sum_{i=1}^n (y_i - w^T x_i)^2 + \lambda \|w\|^2$$

- Hyper parameter does not have to represent a model class
- It can represent the algorithm being used also
- **hyper-parameter**  $\lambda$  for ridge regression
  - e.g., a linear model class  $F_1 = \{f(x) = b + x^T w \mid (b, w) \in \mathbb{R}^{d+1}\}$  trained by minimizing a regularized loss

$$(\hat{b}_\lambda, \hat{w}_\lambda) = \arg \min_{(b, w) \in \mathbb{R}^{d+1}} \sum_{i=1}^n (y_i - (b + x_i^T w))^2 + \lambda \|w\|_2^2$$

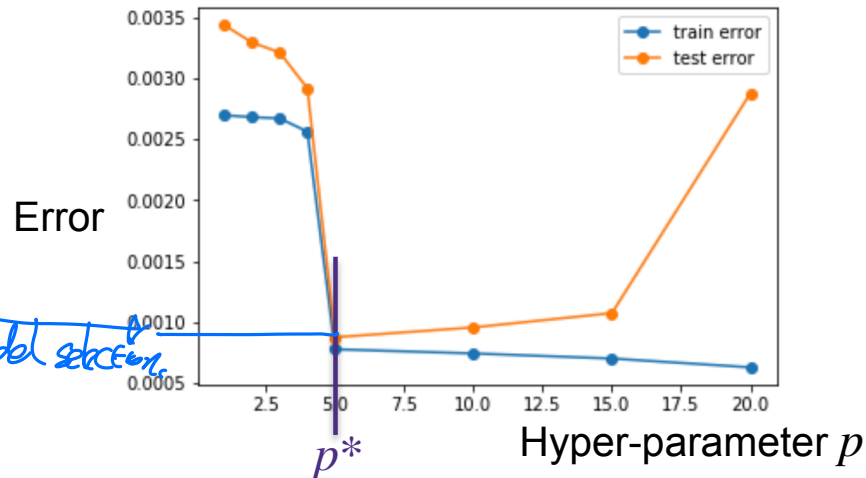
- $\lambda$  is a hyper-parameter, because it is fixed during training
- And we run multiple training for multiple choices of the hyper-parameter



# Model selection $\hat{=}$ hyper-parameter tuning.

- **Model selection** asks the following question: among all the models we got for different hyper-parameters, how do we choose the “best” one to deploy?
- Wrong approach 1:
  - Randomly split the dataset into Train Set and Test Set with 80/20 split
  - Train models for various hyper-parameters and report the Train Error and Test Error
  - Deploy the model  $\hat{w}_{p^*}$  achieving minimum Test Error
  - Report its Test Error as an approximation of the True Error
- Issue:

- ① Underestimates True Error because Test Error is used in model selection.
- ② 20% of data for Test Error.



# Why using test error for model selection gives under estimation of the true error

- Consider a simple experiment where we have two coins

- $x \sim \text{Bern}(p)$  and  $y \sim \text{Bern}(q)$  such that

Test Error  $\rightarrow$   $x = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1 - p \end{cases}$       True Error  $\rightarrow$   $y = \begin{cases} 1 & \text{with probability } q \\ 0 & \text{with probability } 1 - q \end{cases}$

- I want to find out  $\min\{p, q\}$  given  ~~$n$  samples from each~~
- Using test set to both select the model and report the test error is same as
  - Reporting the smaller one, i.e.,  $\min\{x, y\}$  *this underestimates  $\min\{p, q\}$*
  - We can show that this reported value is strictly smaller than what we wanted in expectation:  $\mathbb{E}[\min\{x, y\}] < \min\{p, q\}$
  - For example,  ~~$\mathbb{E}[\min\{\hat{p}, \hat{q}\}] = \mathbb{E}[\min\{x, y\}] = p\hat{q} < \min\{p, q\}$~~

$$\min\{x, y\} = \begin{cases} 0 & 1-p\hat{q} \\ 1 & p\hat{q} \end{cases}$$

# To avoid underestimating test error

---

- Never use the **test set** for
  - training any model, or
  - tuning hyper-parameter = model selection
- **Test set** should only be used once to report test error (as an approximation of the true error) in the end
- Idea:
  - use part of training data (called **Validation Set**) to estimated the error and for model selection
  - For example:



- **Train set**: train (multiple) models for different hyper-parameters
- **Validation set**: compute validation error, to be used in model selection
- **Test set**: use it once in the end to report test error for the selected model

# (LOO) Leave-one-out cross validation

- Consider a validation set with 1 example:

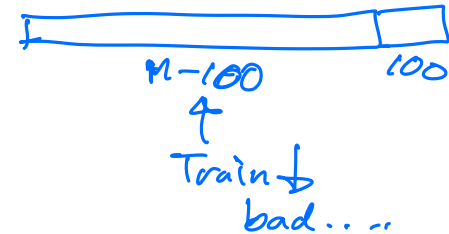
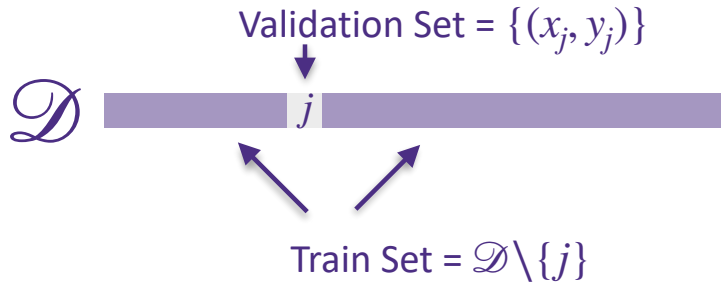
Notation:

$A \setminus B = A \cap B^C$  denotes setminus

- $\mathcal{D}$  : dataset

- $\mathcal{D} \setminus \{j\}$  : train set with  $j$ -th data point  $(x_j, y_j)$  moved to validation set

- Learn model  $f_{\mathcal{D} \setminus \{j\}}$  with  $\mathcal{D} \setminus \{j\}$  dataset:  $f_{\mathcal{D} \setminus \{j\}} = \arg \min_f \sum_{i \in \mathcal{D} \setminus \{j\}} (y_i - f(x_i))^2$



- Validation error:**  $\text{error}_j \triangleq (y_j - f_{\mathcal{D} \setminus \{j\}}(x_j))^2$

- It is an unbiased estimate of the error

$$\text{error}_{\text{true}}(f_{\mathcal{D} \setminus \{j\}}) \triangleq \mathbb{E}_{(x,y) \sim P_{x,y}} [(y - f_{\mathcal{D} \setminus \{j\}}(x))^2]$$

- but, variance of  $\text{error}_j$  is too large. Why?  $\leftarrow$  Valid set #1 entry

- Any ideas?



# (LOO) Leave-one-out cross validation

- To reduce the variance of the validation error, use instead
- **LOO cross validation**: Average over all possible single sample validation set  $\{j\}$  for  $j \in \{1, \dots, n\}$ :



- Train  $n$  times:

for each data point you leave out, learn a new classifier  $f_{\mathcal{D} \setminus \{j\}}$

*train  $n$  models*

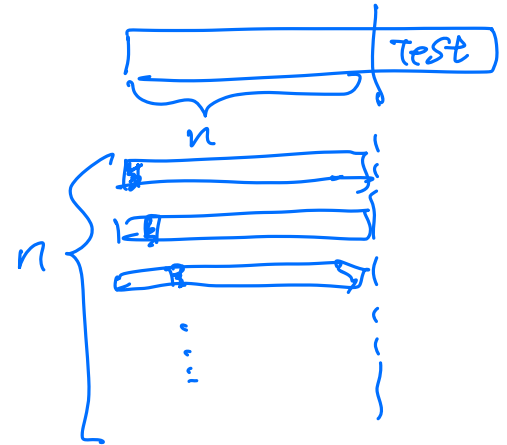
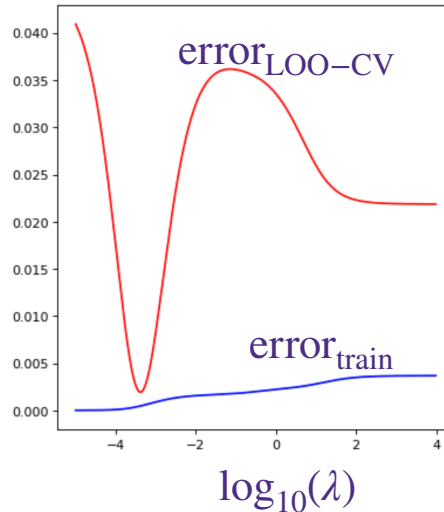
- Validation error is now averaged over all different splits:

$$\text{error}_{\text{LOO-CV}} = \frac{1}{n} \sum_{j=1}^n \text{error}_j = \frac{1}{n} \sum_{j=1}^n (y_j - f_{\mathcal{D} \setminus \{j\}}(x_j))^2$$

# LOO cross validation is (almost) unbiased estimate!

*≈ error of using (n-1) samples*

- When computing LOO-CV error, we only use  $n - 1$  data points to train
  - So it's not an estimate of true error of learning with  $n$  data points
- **true error** =  $\mathbb{E}_{X,Y}[(Y - f_{\mathcal{D}}(X))^2]$
- Usually (slightly) pessimistic – learning with less data typically gives worse answer.
- Leads to a (slight) over estimation of the error compared to true error
- LOO-CV is almost unbiased! Use LOO-CV error for model selection!!!
  - **E.g., picking  $\lambda$**



# Computational cost of LOO

---

- Suppose you have 100,000 data points
- say, you implemented a fast version of your learning algorithm
  - Learns in only 1 second
- Computing LOO will take about 1 day
- In general, LOO takes  $n$  times longer than training one model
- Any ideas?

# Use $k$ -fold cross validation

- Randomly divide data into  $k$  equal parts

- $\mathcal{D}_1, \dots, \mathcal{D}_k$

- For each  $i$

- Learn model  $f_{\mathcal{D} \setminus \mathcal{D}_i}$  using data point not in  $\mathcal{D}_i$
  - Estimate error of  $f_{\mathcal{D} \setminus \mathcal{D}_i}$  on validation set  $\mathcal{D}_i$ :

$$\text{error}_{\mathcal{D}_i} = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j, y_j) \in \mathcal{D}_i} (y_j - f_{\mathcal{D} \setminus \mathcal{D}_i}(x_j))^2$$

- $k$ -fold cross validation error is average over data splits:

$$\text{error}_{k\text{-fold}} = \frac{1}{k} \sum_{i=1}^k \text{error}_{\mathcal{D}_i}$$

- $k$ -fold cross validation properties:

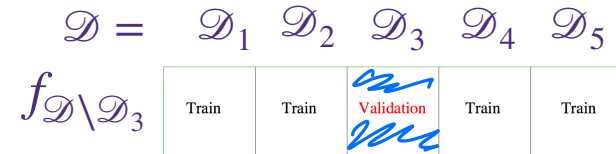
- Much faster to compute than LOO-CV as  $k \ll n$

- More (pessimistically) biased – using much less data, only

$$\boxed{n - \frac{n}{k}} \text{ vs. } n-1$$

LOO-CV

- Usually,  $k = 10$

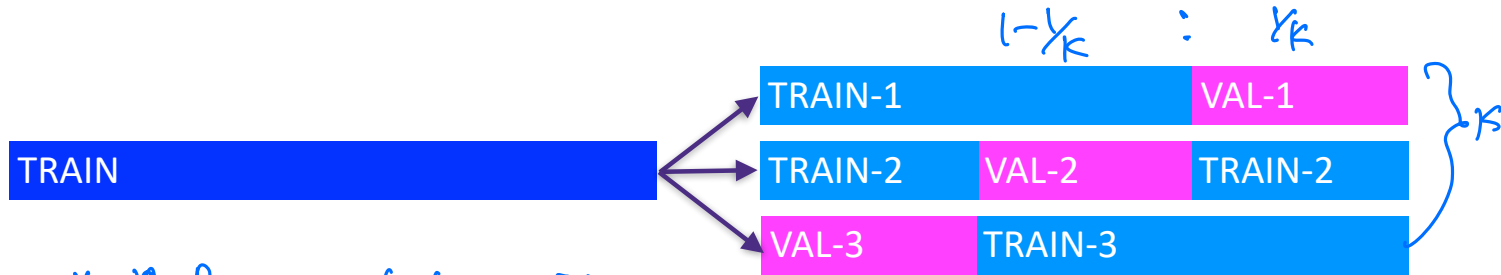


# Recap

- > Given a dataset, begin by splitting into RANDOMLY



- > Model selection: Use k-fold cross-validation on **TRAIN** to train predictor and choose hyper-parameters such as  $\lambda$



\* Process not too sensitive to K.  $2^k 10,000 ?$  ← Determined by Time

- > Model assessment: Use **TEST** to assess the accuracy of the model you output
  - Never train or choose parameters based on the test data

# Example 1

---

- You wish to predict the stock price of zoom.us given historical stock price data  $y_i$ 's (for each  $i$ -th day) and the historical news articles  $x_i$ 's
- You use all daily stock price up to Jan 1, 2020 as **TRAIN** and Jan 2, 2020 - April 13, 2020 as **TEST**
- What's wrong with this procedure?

*Not Randomly split*

*COVID*

# Example 2

- Given 10,000-dimensional data and  $n$  examples, we pick a subset of 50 dimensions that have the highest correlation with labels in the training set:

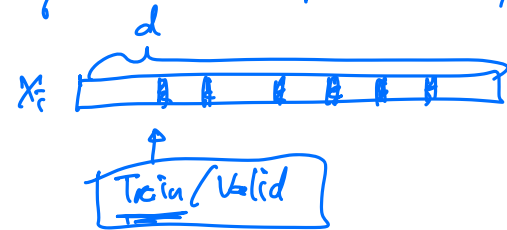
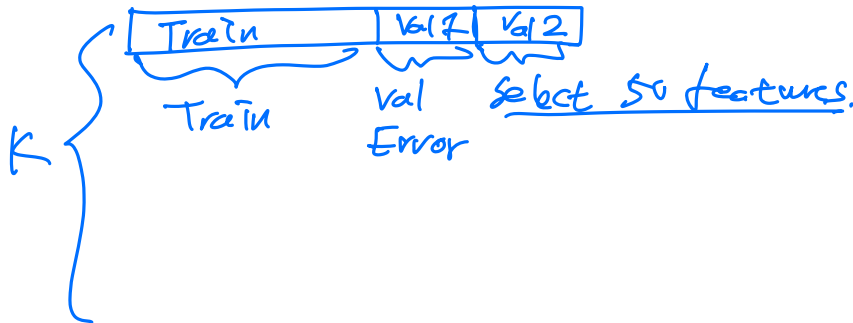
50 indices  $j$  that have largest

$$\frac{\left| \sum_{i=1}^n x_{i,j} y_i \right|}{\sqrt{\sum_{i=1}^n x_{i,j}^2}}$$

How good is  $j$ -th feature in predicting  $y$ ?

$j$ -th feature

- After picking our 50 features, we then use CV with the training set to train ridge regression with regularization  $\lambda$
- What's wrong with this procedure?



# Recap

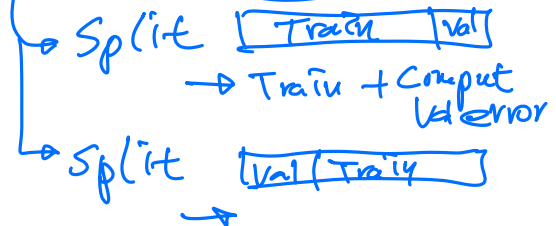
## > Learning is...

- Collect some data
  - > E.g., housing info and sale price
- Randomly select **TEST set** and split the remaining dataset into **TRAIN**, and **VAL** (multiple splits are needed if doing cross validation)
  - > E.g., **80%**, **10%**, and **10%**, respectively
- Choose a hypothesis class or model
  - > E.g., **linear with non-linear features** (also called transformations)
- Choose a loss function
  - > E.g., least squares **with ridge regression penalty on TRAIN**

## Choose an optimization procedure

- > E.g., set derivative to zero to obtain estimator, **cross-validation on VAL to pick num. features and amount of regularization**

- Justifying the accuracy of the estimate
  - > E.g., report **TEST error**





# Questions?

---

|       |     |      |
|-------|-----|------|
| Trait | Val | Test |
|-------|-----|------|

# Questions?

---

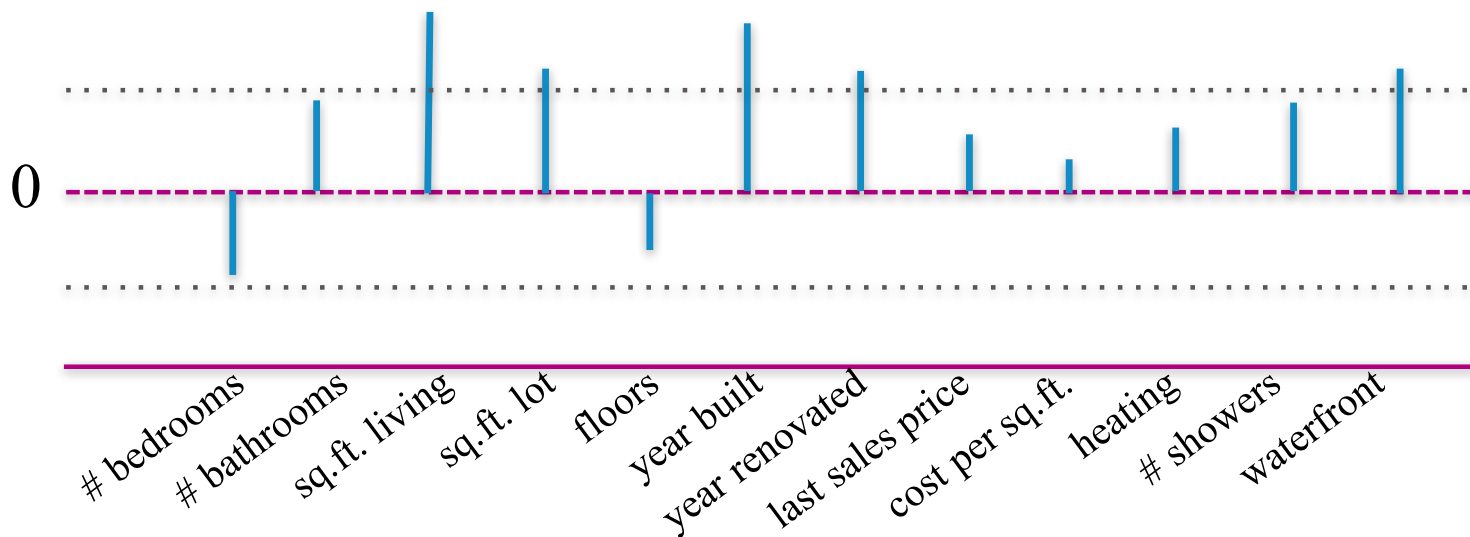
# Questions?

---

# Thresholded Ridge Regression

$$\hat{w}_{ridge} = \arg \min_w \sum_{i=1}^n (y_i - x_i^T w)^2 + \lambda ||w||_2^2$$

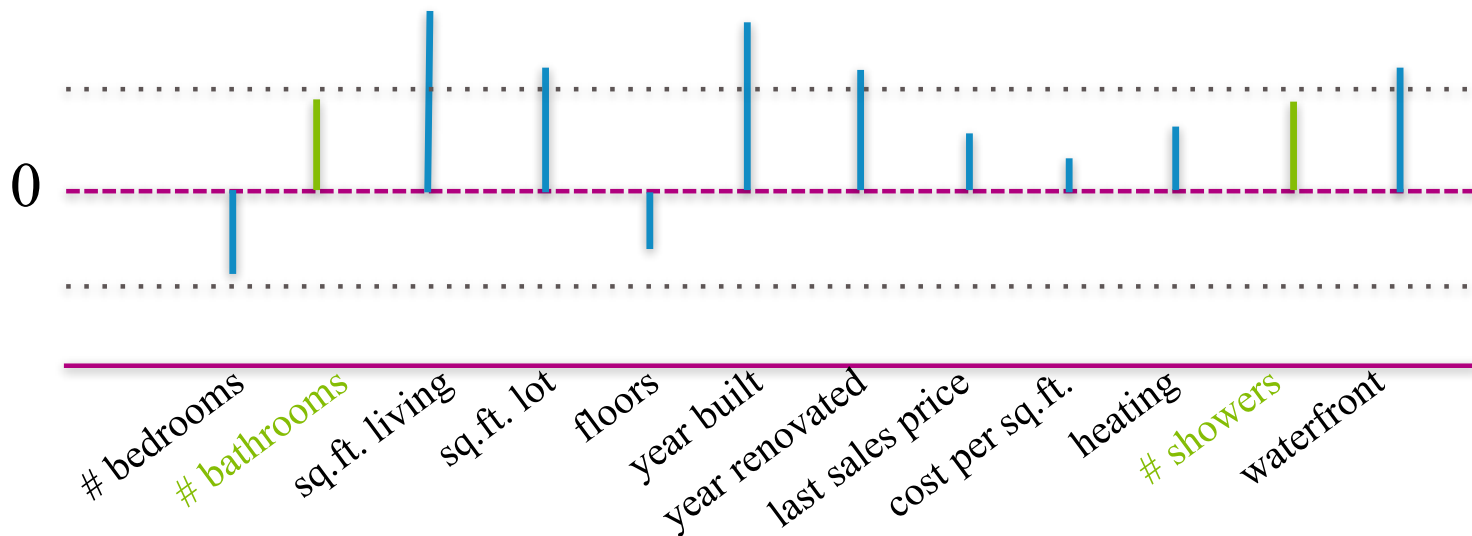
Why don't we just set **small** ridge coefficients to 0?



# Thresholded Ridge Regression

$$\hat{w}_{ridge} = \arg \min_w \sum_{i=1}^n (y_i - x_i^T w)^2 + \lambda ||w||_2^2$$

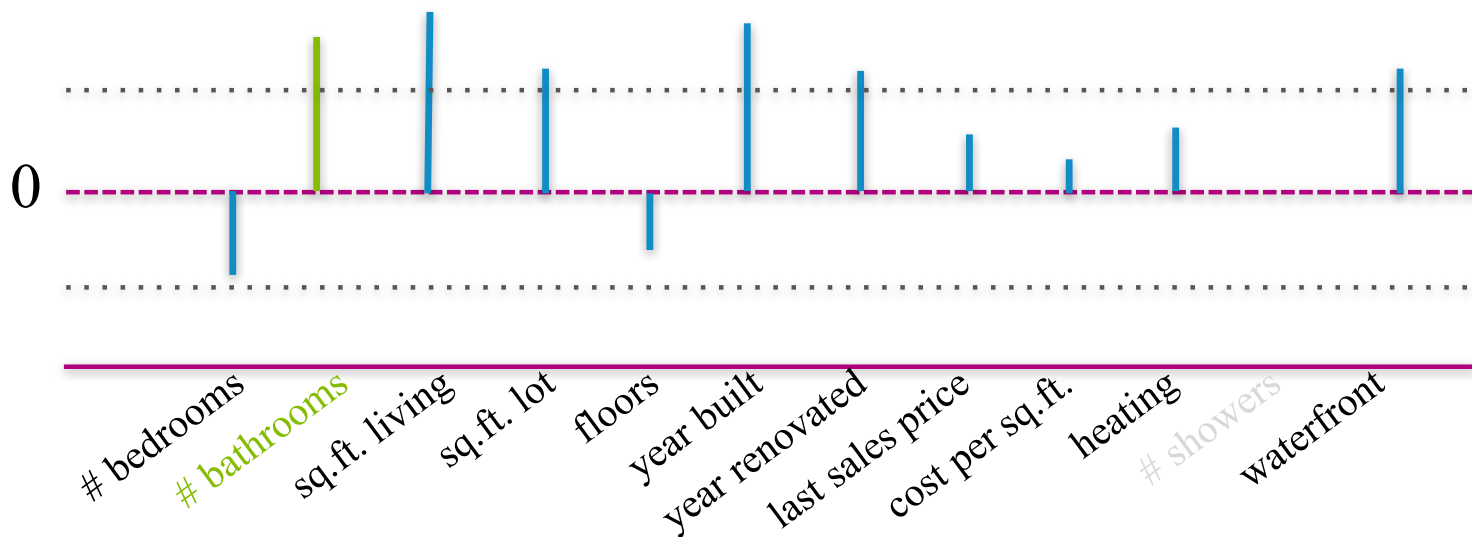
Consider two **related** features (bathrooms, showers)



# Thresholded Ridge Regression

$$\hat{w}_{ridge} = \arg \min_w \sum_{i=1}^n (y_i - x_i^T w)^2 + \lambda ||w||_2^2$$

What if we **didn't** include showers? Weight on bathrooms increases!



Can another regularizer perform selection automatically?