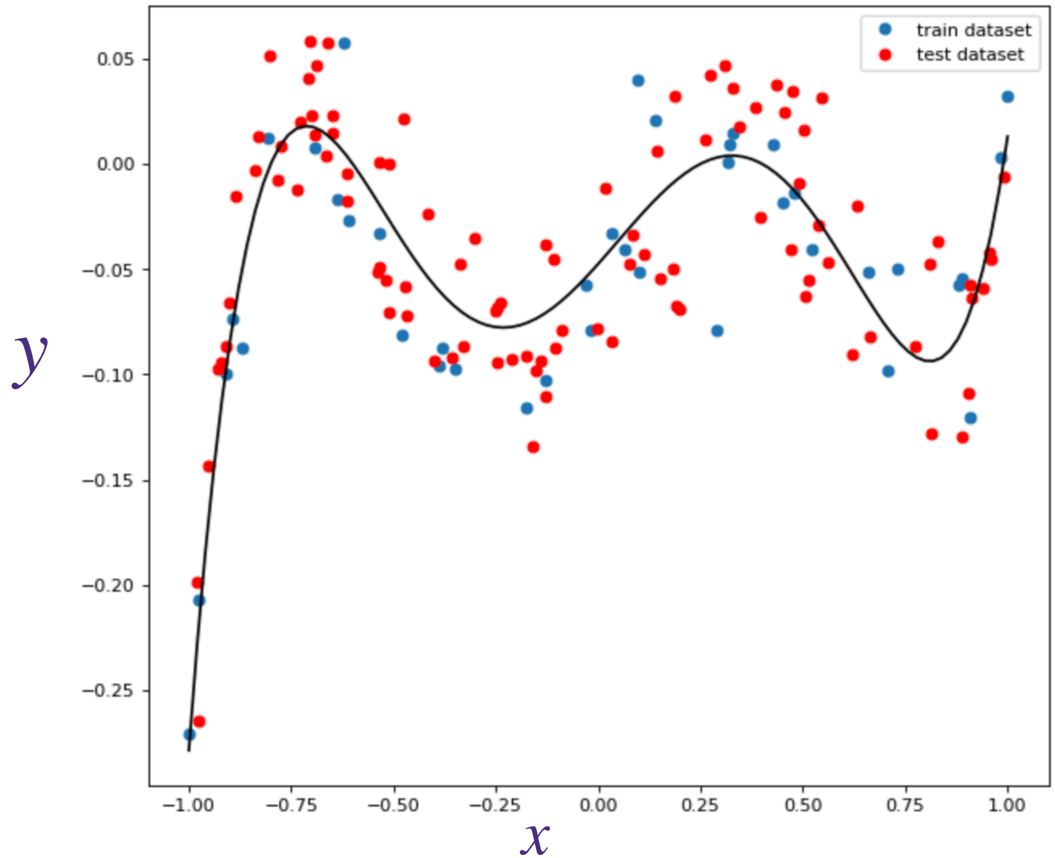
Regularization

Recap: bias-variance tradeoff

- Consider 100 training examples and 100 test examples i.i.d.drawn from degree-5 polynomial features
  $x_i \sim \text{Uniform}[-1,1]$, $y_i \sim f_{w*}(x_i) + \epsilon_i$, $\epsilon_i \sim \mathcal{N}(0,\sigma^2)$

$$f_w(x_i) = b^* + w_1^* x_i + w_2^*(x_i)^2 + w_3^*(x_i)^3 + w_4^*(x_i)^4 + w_5^*(x_i)^5$$

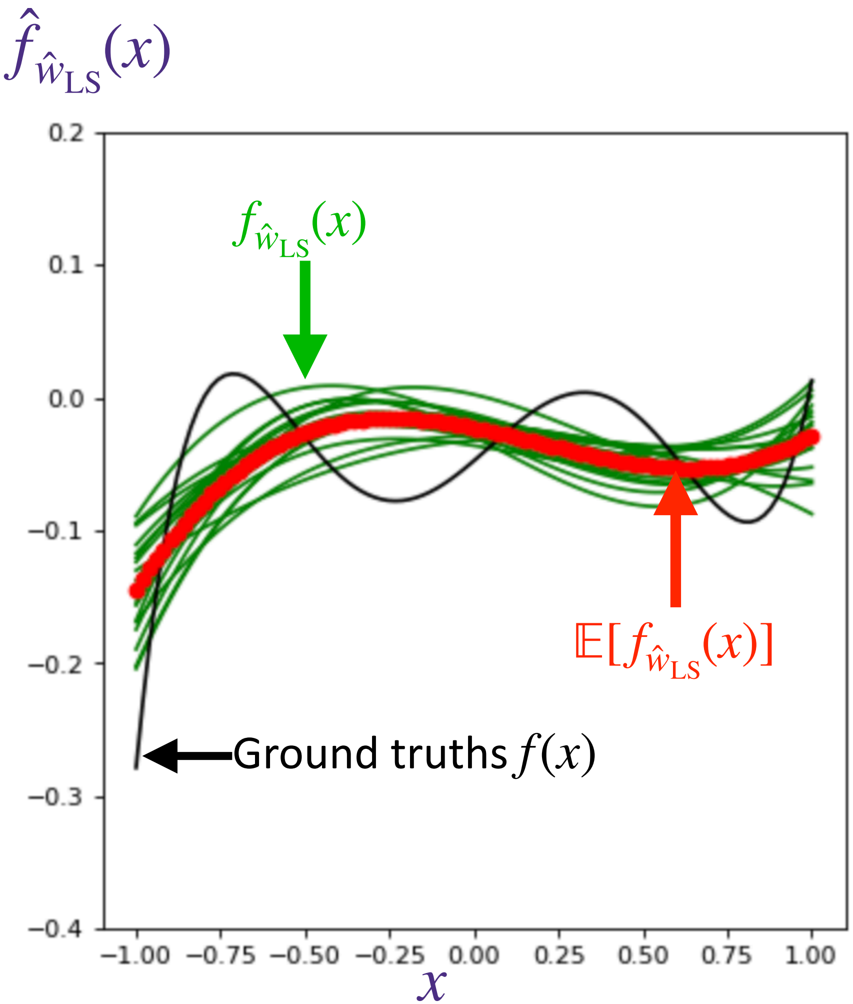


This is a linear model with features

$$h(x_i) = (x_i, (x_i)^2, (x_i)^3, (x_i)^4, (x_i)^5)$$

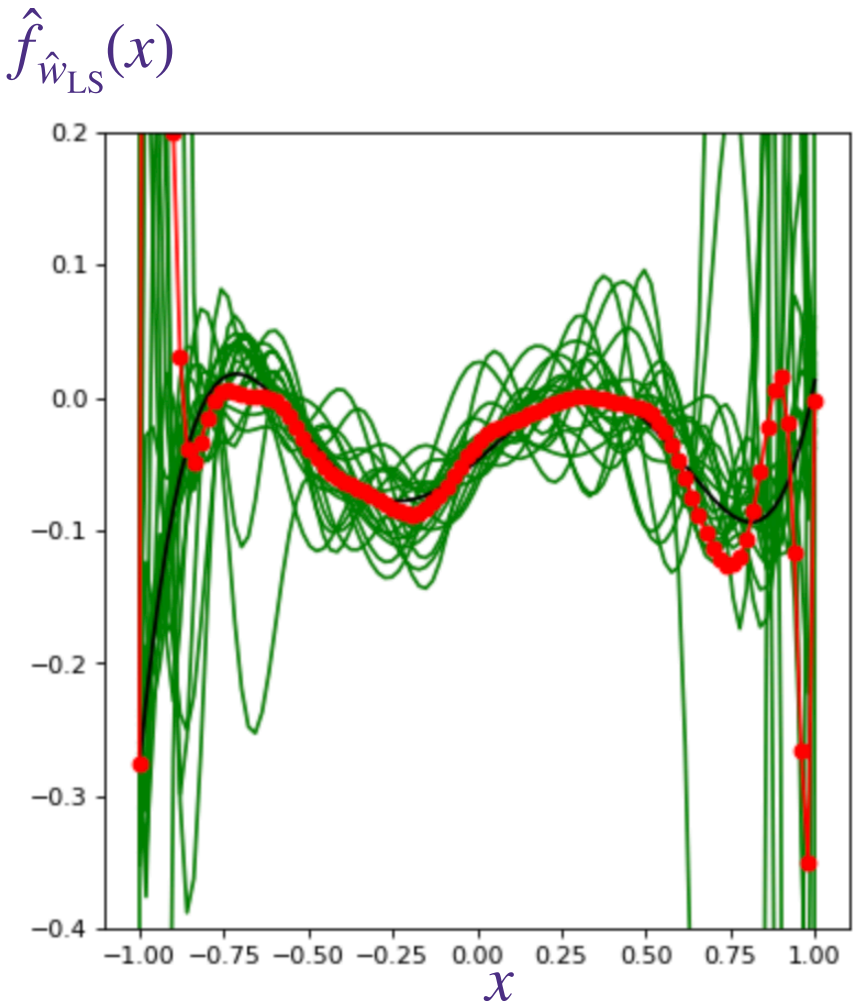
Recap: bias-variance tradeoff

With degree-3 polynomials, we underfit




```
current train error = 0.0036791644380554187
current test error  = 0.0037962529988410953
```

With degree-20 polynomials, we overfit




```
0.0005421686349568773
0.14210029429557927
```

Sensitivity: how to detect overfitting

- For a linear model,
  $y \simeq b + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$
  if $|w_j|$ is large then the prediction is sensitive to small changes in x_j
- Large sensitivity leads to overfitting and poor generalization, and equivalently models that overfit tend to have large weights
- Note that b is a constant and hence there is no sensitivity for the offset b

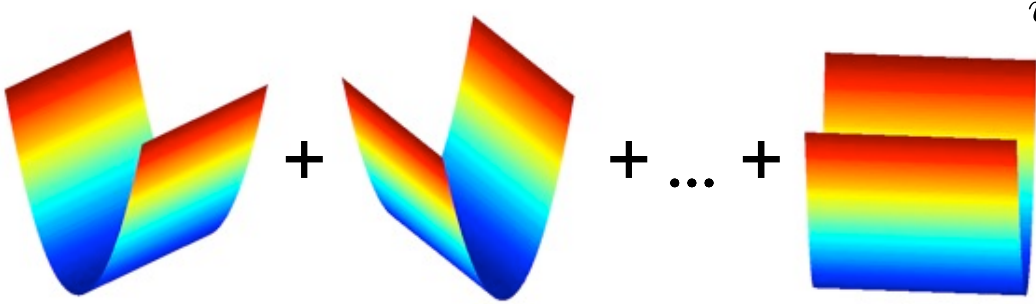
- In **Ridge Regression,** we use a regularizer $||w||_2^2$ to measure and control the sensitivity of the predictor
- And optimize for small loss and small sensitivity, by adding a **regularizer** in the objective (assume no offset for now)

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

Ridge Regression

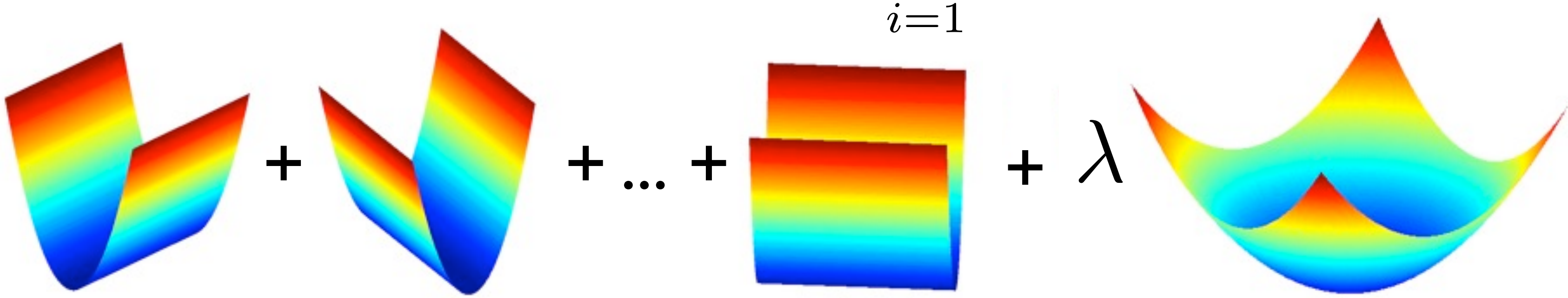
- (Original) Least squares objective:

$$\widehat{w}_{LS} = \arg\min_w \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$$



- Ridge Regression objective:

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



Minimizing the Ridge Regression Objective

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

Shrinkage Properties

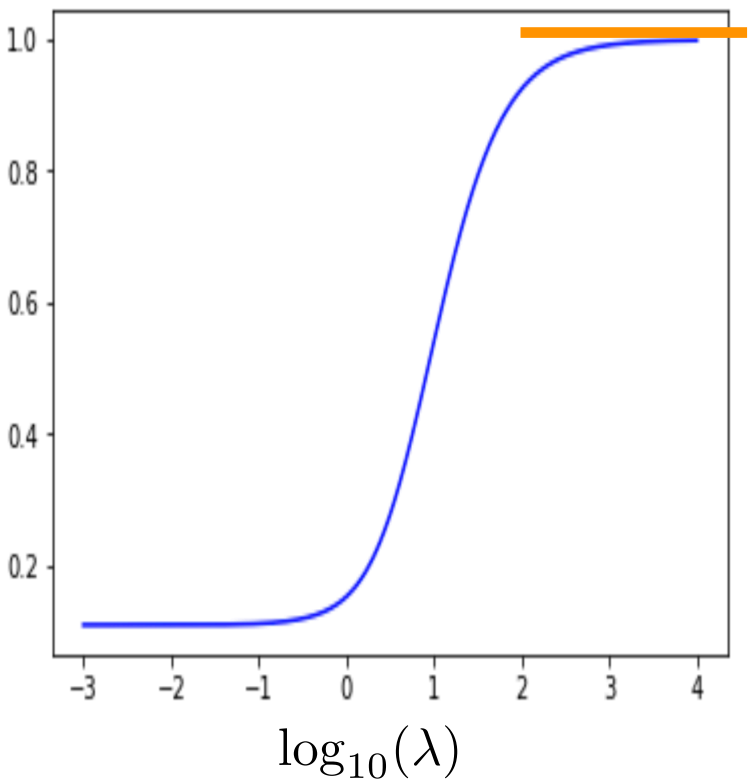
$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$

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

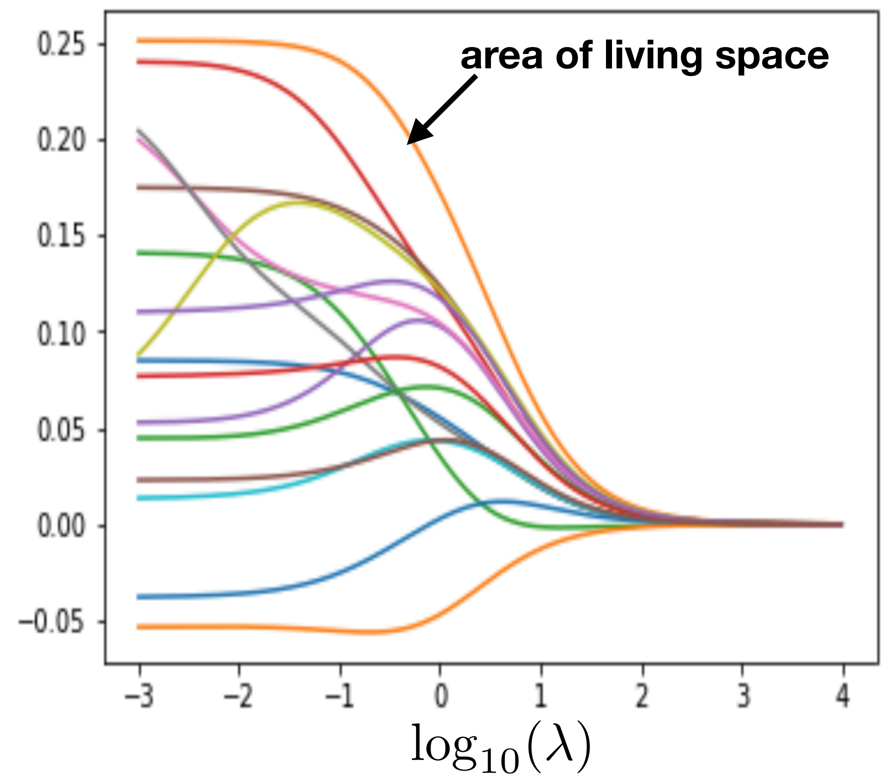
- When $\lambda = 0$, this gives the least squares model
- This defines a family of models hyper-parametrized by λ
- Large λ means more regularization and simpler model
- Small λ means less regularization and more complex model

Ridge regression: minimize $\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \lambda\|w\|_2^2$

training MSE $\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T \hat{w}_{\text{ridge}}^{(\lambda)})^2$

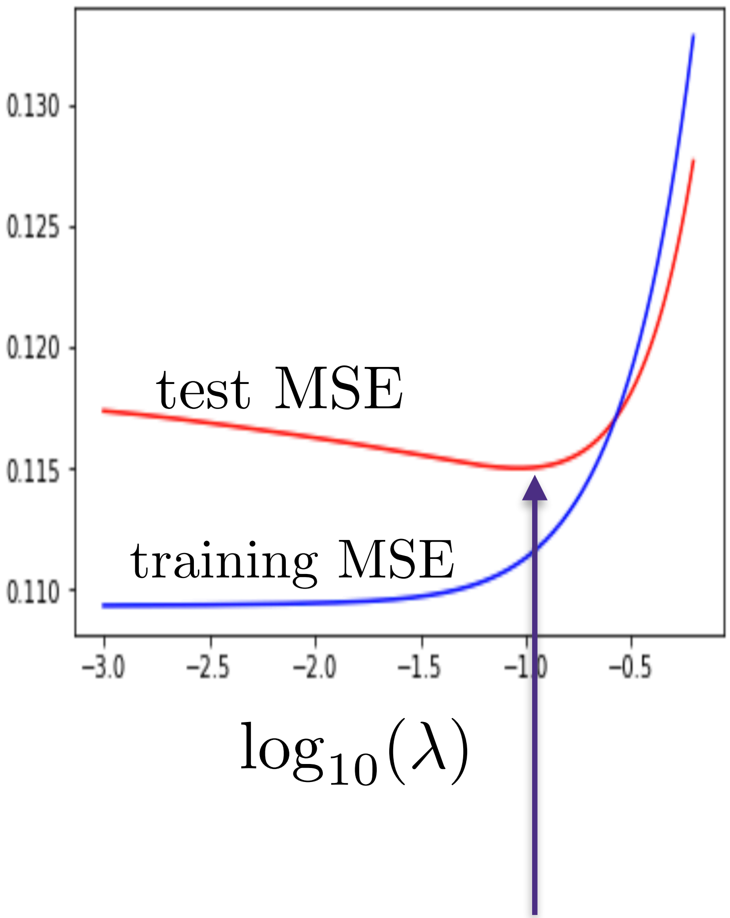


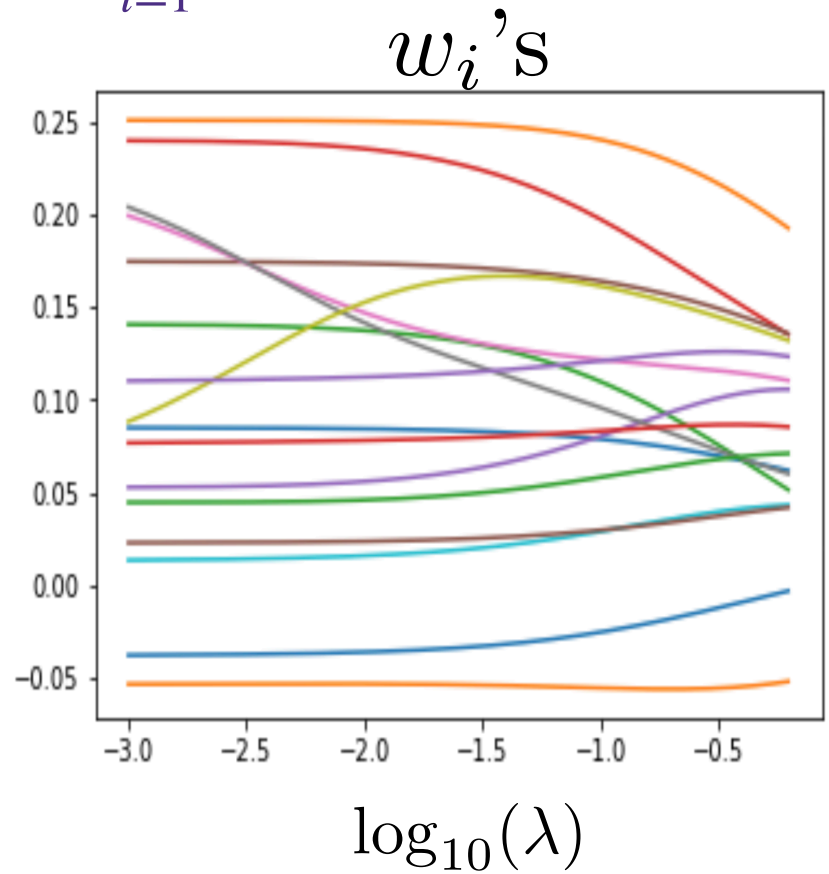
w_i's



- Left plot: leftmost training error is with no regularization: 0.1093
- Left plot: rightmost training error is variance of the training data: 0.9991
- Right plot: called **regularization path**

Ridge regression: minimize $\sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda \|w\|_2^2$





- this gain in test MSE comes from shrinking w's to get a less sensitive predictor
  (which in turn reduces the variance)

Bias-Variance Properties

- Recall: $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
- To analyze bias-variance tradeoff, we need to assume probabilistic generative model: $x_i \sim P_X, \ \mathbf{y} = \mathbf{X}w + \epsilon, \ \epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$
- The true error at a sample with feature x is $\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$

Bias-Variance Properties

- Recall: $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
- To analyze bias-variance tradeoff, we need to assume probabilistic generative model: $x_i \sim P_X, \;\; \mathbf{y} = \mathbf{X}w + \epsilon, \;\; \epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$
- The true error at a sample with feature x is

$$\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$
$$= \underbrace{\mathbb{E}_{y|x}[(y - \mathbb{E}[y\,|\,x])^2 \,|\, x]}_{\text{Irreducible Error}} + \underbrace{\mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}[y\,|\,x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]}_{\text{Learning Error}}$$

Bias-Variance Properties

- Recall: $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
- To analyze bias-variance tradeoff, we need to assume probabilistic generative model: $x_i \sim P_X,\ \ \mathbf{y} = \mathbf{X}w + \epsilon,\ \ \epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$
- The true error at a sample with feature x is

$$\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$
$$= \mathbb{E}_{y|x}[(y - \mathbb{E}[y\,|\,x])^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}[y\,|\,x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$
$$= \mathbb{E}_{y|x}[(y - x^Tw)^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(x^Tw - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

Bias-Variance Properties

- Recall: $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
- To analyze bias-variance tradeoff, we need to assume probabilistic generative model: $x_i \sim P_X,\ \ \mathbf{y} = \mathbf{X}w + \epsilon,\ \ \epsilon \sim \mathcal{N}(0,\sigma^2\mathbf{I})$
- The true error at a sample with feature x is

$$\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \mathbb{E}_{y|x}[(y - \mathbb{E}[y\,|\,x])^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}[y\,|\,x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \mathbb{E}_{y|x}[(y - x^Tw)^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(x^Tw - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \underbrace{\sigma^2}_{\text{Irreduc. Error}} + \underbrace{(x^Tw - \mathbb{E}_{\mathcal{D}_{\text{train}}}[x^T\hat{w}_{\text{ridge}} \,|\, x])^2}_{\text{Bias-squared}} + \underbrace{\mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}_{\tilde{\mathcal{D}}_{\text{train}}}[x^T\hat{w}_{\text{ridge}} \,|\, x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]}_{\text{Variance}}$$

Bias-Variance Properties

- Recall: $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
- To analyze bias-variance tradeoff, we need to assume probabilistic generative model: $x_i \sim P_X,\ \ \mathbf{y} = \mathbf{X}w + \epsilon,\ \ \epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$
- The true error at a sample with feature x is

$$\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \mathbb{E}_{y|x}[(y - \mathbb{E}[y\,|\,x])^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}[y\,|\,x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \mathbb{E}_{y|x}[(y - x^Tw)^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(x^Tw - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \sigma^2 + (x^Tw - \mathbb{E}_{\mathcal{D}_{\text{train}}}[x^T\hat{w}_{\text{ridge}} \,|\, x])^2 + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}_{\tilde{\mathcal{D}}_{\text{train}}}[x^T\hat{w}_{\text{ridge}} \,|\, x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

Irreduc. Error — Bias-squared — Variance

Suppose $\mathbf{X}^T\mathbf{X} = n\mathbf{I}$, then $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T(\mathbf{X}w + \epsilon)$

$$= \frac{n}{n+\lambda}w + \frac{1}{n+\lambda}\mathbf{X}^T\epsilon$$

Bias-Variance Properties

Suppose $\mathbf{X}^T\mathbf{X} = n\mathbf{I}$, then

$$\hat{w}_{\text{ridge}} = \frac{n}{n+\lambda} w + \frac{1}{n+\lambda}\mathbf{X}^T\epsilon$$

- Recall: $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
- To analyze bias-variance tradeoff, we need to assume probabilistic generative model: $x_i \sim P_X, \;\; \mathbf{y} = \mathbf{X}w + \epsilon, \;\; \epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$
- The true error at a sample with feature x is

$$\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \mathbb{E}_{y|x}[(y - \mathbb{E}[y|x])^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}[y|x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \mathbb{E}_{y|x}[(y - x^Tw)^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(x^Tw - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \sigma^2 + (x^Tw - \mathbb{E}_{\mathcal{D}_{\text{train}}}[x^T\hat{w}_{\text{ridge}} \,|\, x])^2 + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}_{\tilde{\mathcal{D}}_{\text{train}}}[x^T\hat{w}_{\text{ridge}} \,|\, x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

(verify at home)

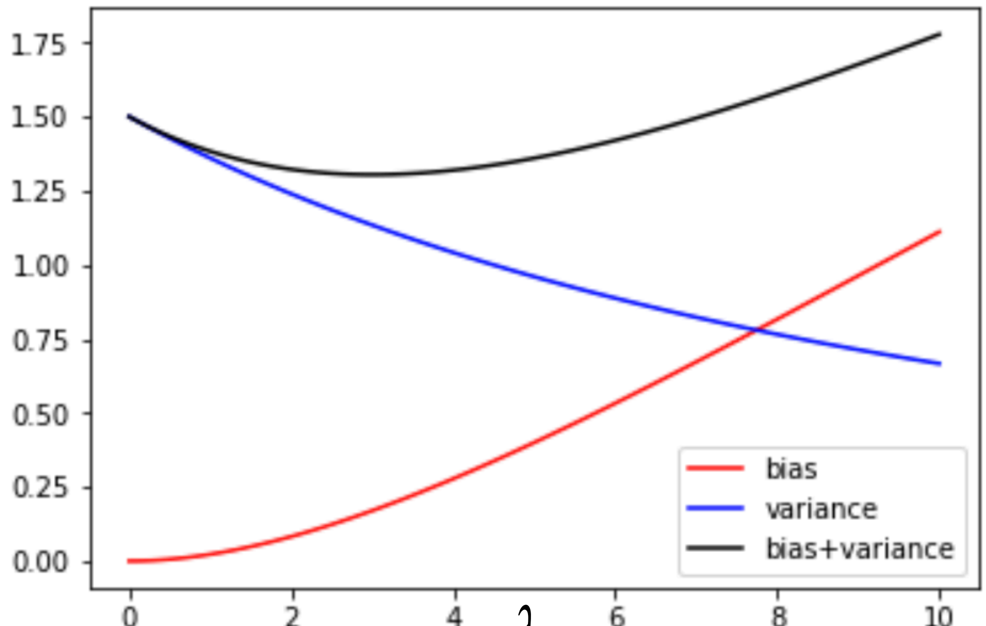
$$= \underbrace{\sigma^2}_{\text{Irreduc. Error}} + \underbrace{\frac{\lambda^2}{(n+\lambda)^2}(w^Tx)^2}_{\text{Bias-squared}} + \underbrace{\frac{\sigma^2 n}{(n+\lambda)^2}\|x\|_2^2}_{\text{Variance}}$$

Bias-Variance Properties

- Ridge regressor: $\widehat{w}_{ridge} = \arg\min_w \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$
- True error

$$\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T \hat{w}_{\text{ridge}})^2 \,|\, x] = \sigma^2 + \underbrace{\frac{\lambda^2}{(n+\lambda)^2}(w^T x)^2}_{\text{Bias-squared}} + \underbrace{\frac{\sigma^2 n}{(n+\lambda)^2}\|x\|_2^2}_{\text{Variance}}$$

d=10, n=20, $\sigma^2 = 3.0, \|w\|_2^2 = 10$



λ

as $\lambda \to 0$, $\hat{w}_{\text{ridge}} \to \hat{w}_{\text{LS}}$

as $\lambda \to \infty$, $\hat{w}_{\text{ridge}} \to 0$

What you need to know…

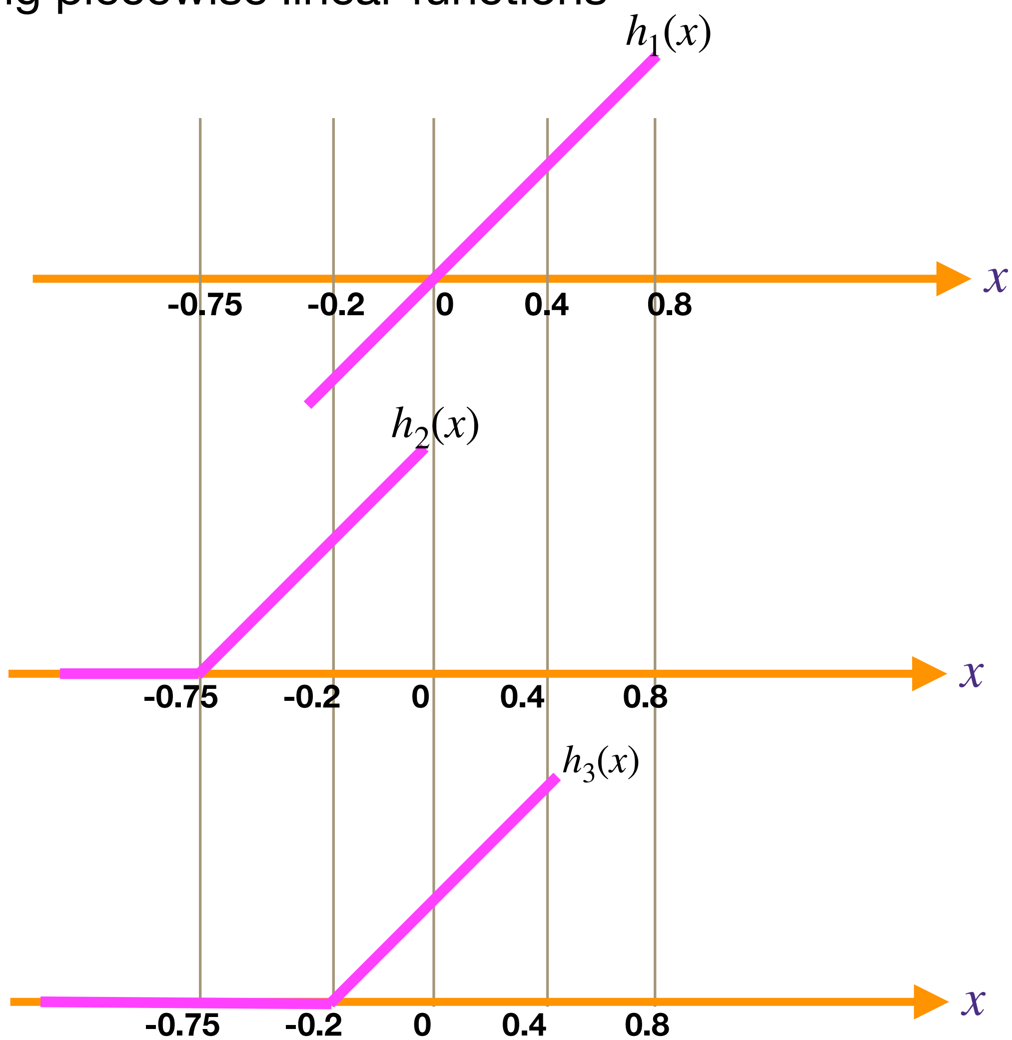
- Regularization
  - Penalizes complex models towards preferred, simpler models
- Ridge regression
  - L_2 penalized least-squares regression
  - Regularization parameter trades off model complexity with training error
  - Never regularize the offset!

Example: piecewise linear fit

- we fit a linear model:
  $f(x) = b + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$
- with a specific choice of features using piecewise linear functions

$$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ h_3(x) \\ h_4(x) \\ h_5(x) \end{bmatrix} = \begin{bmatrix} x \\ [x+0.75]^+ \\ [x+0.2]^+ \\ [x-0.4]^+ \\ [x-0.8]^+ \end{bmatrix}$$

$$[a]^+ \triangleq \max\{a, 0\}$$

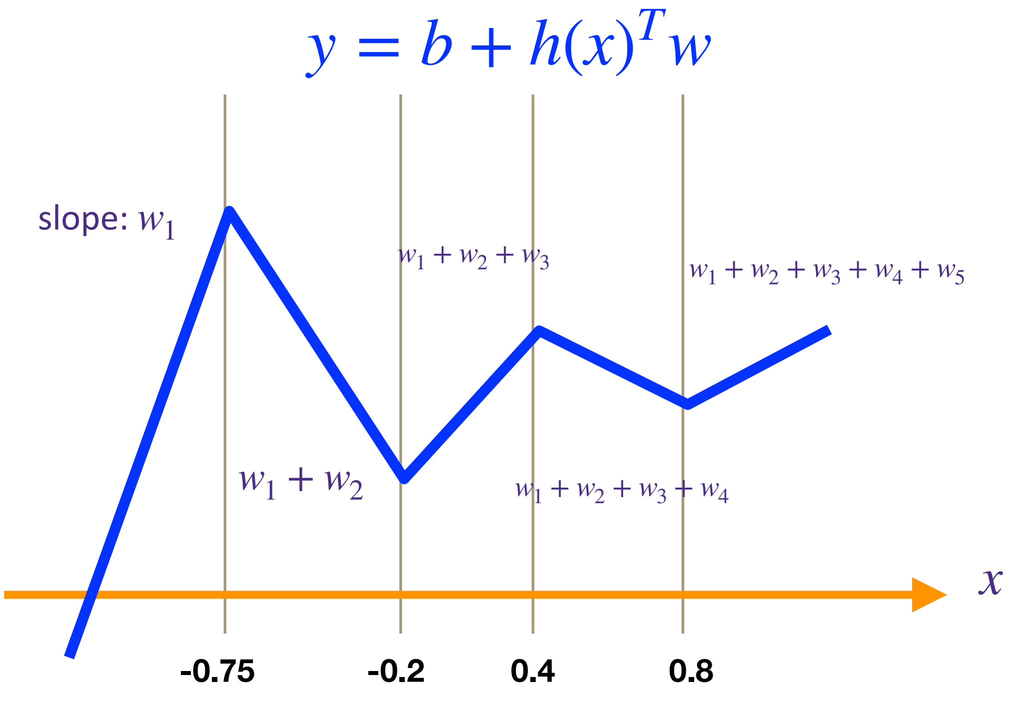


Example: piecewise linear fit

- we fit a linear model:
  $f(x) = b + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$
- with a specific choice of features using piecewise linear functions

$$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ h_3(x) \\ h_4(x) \\ h_5(x) \end{bmatrix} = \begin{bmatrix} x \\ [x+0.75]^+ \\ [x+0.2]^+ \\ [x-0.4]^+ \\ [x-0.8]^+ \end{bmatrix}$$

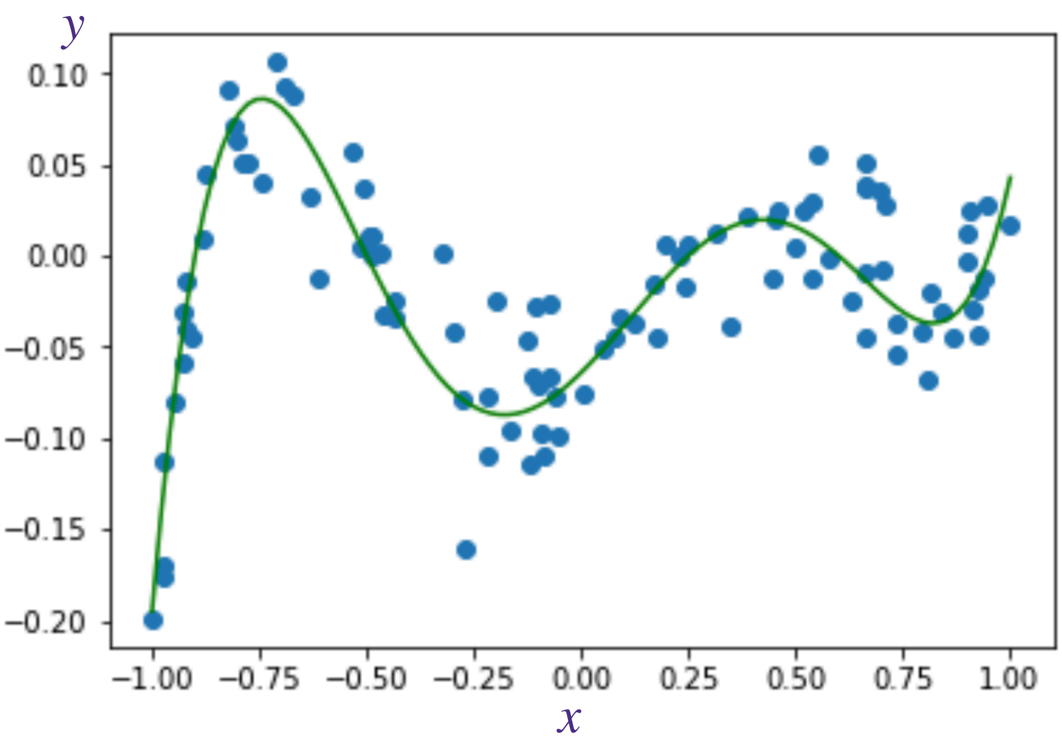
$$[a]^+ \triangleq \max\{a, 0\}$$



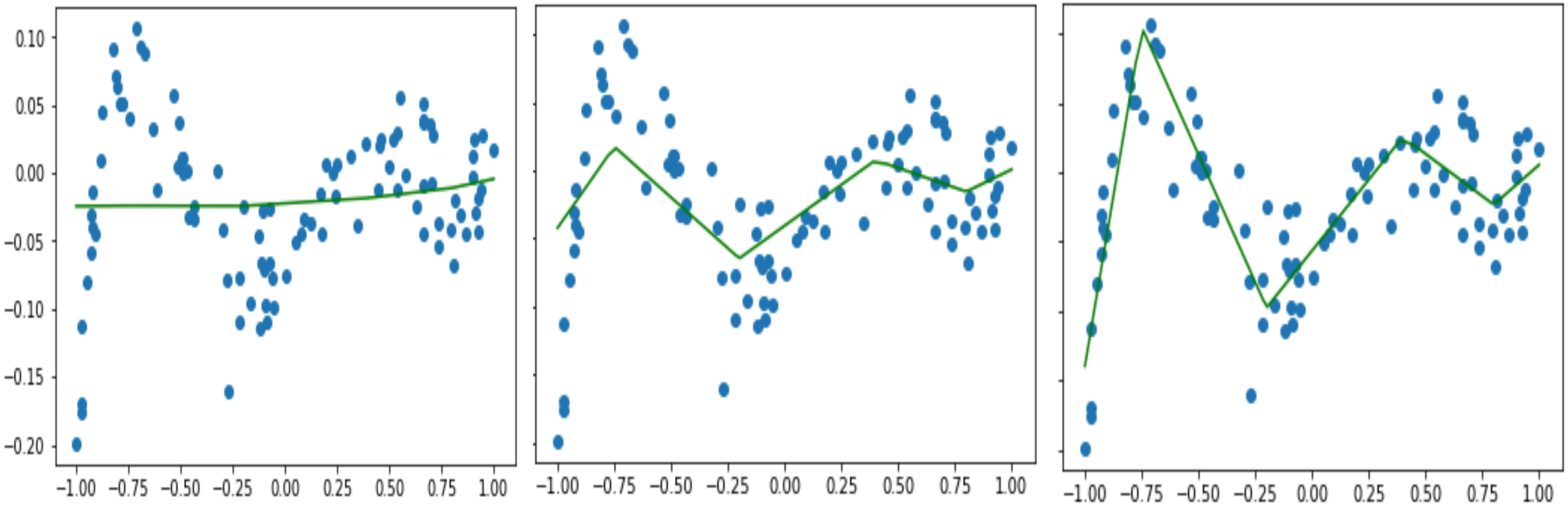
the weights capture the change in the slopes

Example: piecewise linear fit

- we fit a linear model:
  $f(x) = b + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$
- with a specific choice of features using piecewise linear functions



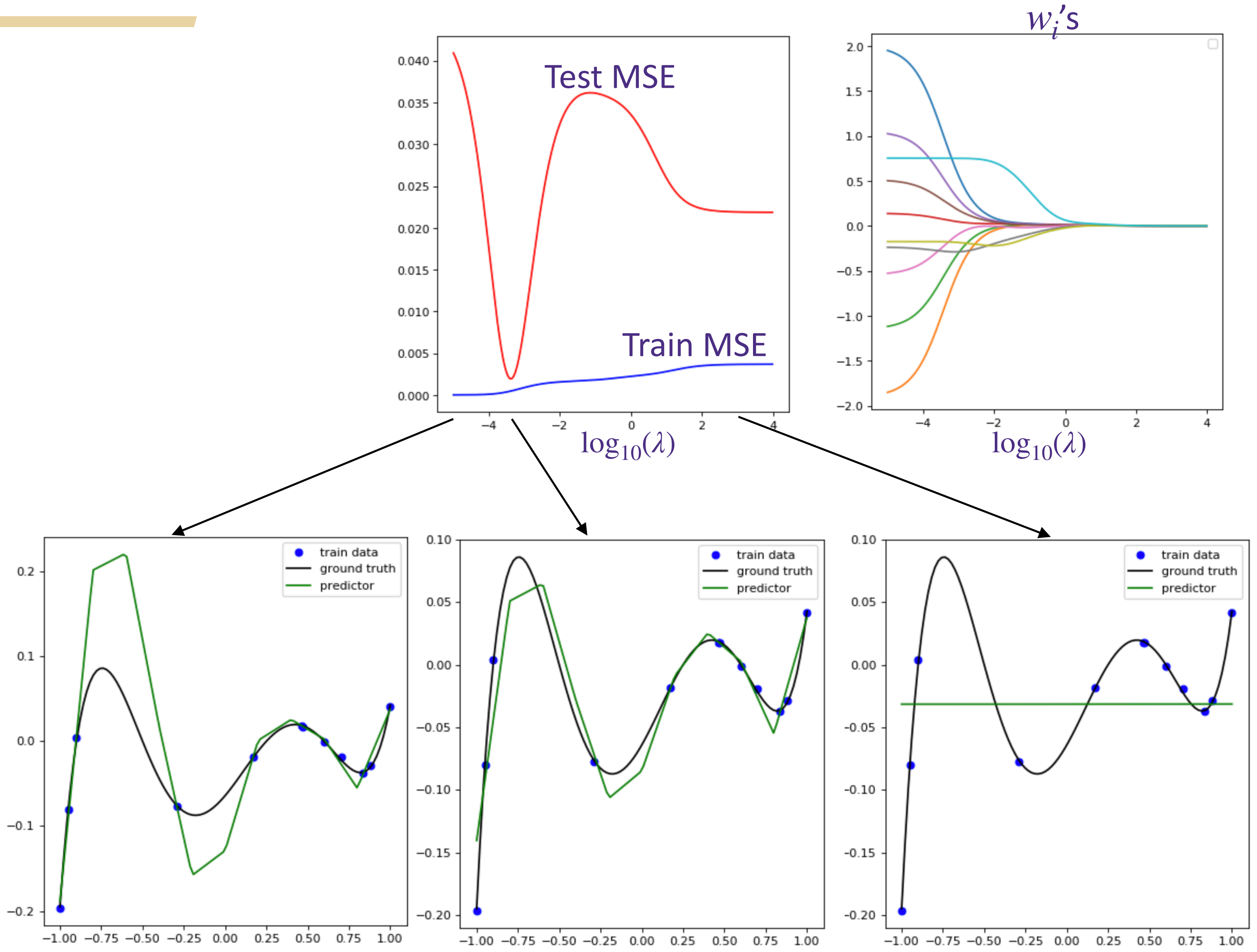
Example: piecewise linear fit (ridge regression)



$\lambda = 1$ $\lambda = 0.005$ $\lambda = 0.000001$

We do not observe overfitting, as d=5 and n=100

Piecewise linear with $w \in \mathbb{R}^{10}$ and n=11 samples

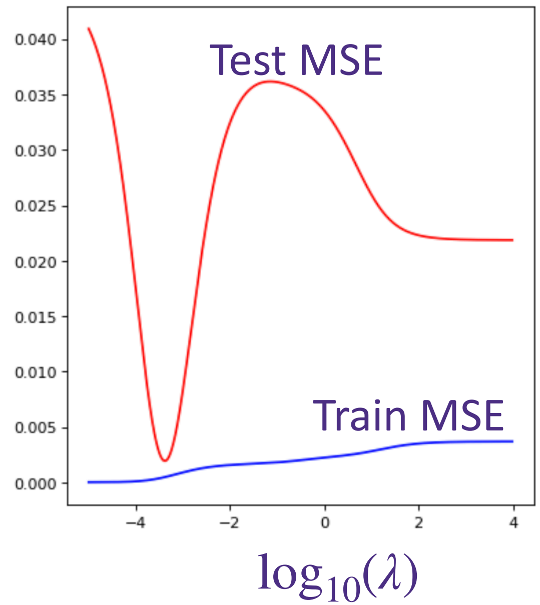


Model selection using Cross-validation

How… How… How????????

- Ridge regression:
  How do we pick the regularization constant λ…
- Polynomial features:
  How do we pick the number of basis functions…

- We could use the test data, but…



How… How… How???????

- Ridge regression:
  How do we pick the regularization constant λ…
- Polynomial features:
  How do we pick the number of basis functions…

- We could use the test data, but…
  - Never ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever **train on the test data**

  - Use test data only for reporting the test error
    (once in the end)

(LOO) Leave-one-out cross validation

- Consider a validation set with 1 example:
  - $\mathcal{D}$: training data
  - $\mathcal{D} \backslash j$: training data with j-th data point (x_j, y_j) moved to validation set
- Learn model $f_{\mathcal{D} \backslash j}$ with $\mathcal{D} \backslash j$ dataset
- The squared error on predicting y_j: $(y_j - f_{\mathcal{D} \backslash j}(x_j))^2$

  is an unbiased estimate of the **true error**

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

  but, variance of $(y_j - f_{\mathcal{D} \backslash j}(x_j))^2$ is too large

(LOO) Leave-one-out cross validation

- Consider a validation set with 1 example:
  - $\mathscr{D}$: training data
  - $\mathscr{D}\backslash j$: training data with j-th data point (x_j, y_j) moved to validation set
- Learn model $f_{\mathscr{D}\backslash j}$ with $\mathscr{D}\backslash j$ dataset
- The squared error on predicting y_j: $(y_j - f_{\mathscr{D}\backslash j}(x_j))^2$

  is an unbiased estimate of the **true error**

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

  but variance of $(y_j - f_{\mathscr{D}\backslash j}(x_j))^2$ is too large, so instead

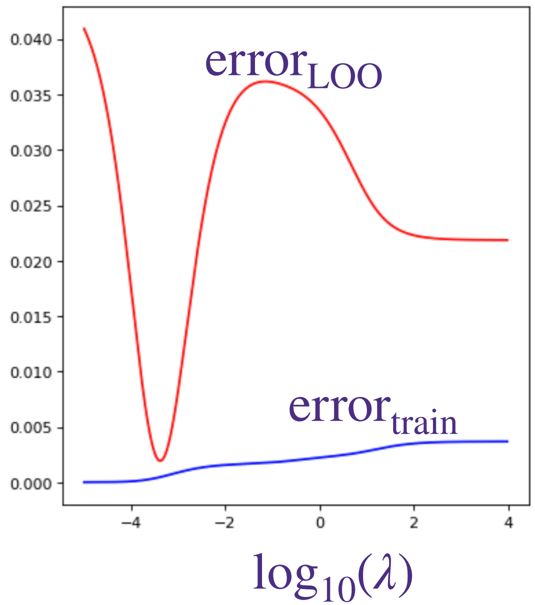
- **LOO cross validation**: Average over all data points j:
  - Train n times:
    for each data point you leave out, learn a new classifier $f_{\mathscr{D}\backslash j}$
  - **Estimate the true error** as:

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

LOO cross validation is (almost) unbiased estimate!

- When computing LOOCV error, we only use $n - 1$data points to train
  - So it's not estimate of true error of learning with n data points
  - Usually pessimistic – learning with less data typically gives worse answer. (Leads to an over estimation of the error)

- LOO is almost unbiased! Use LOO error for model selection!!!
  - **E.g., picking λ**



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!!

Use *k*-fold cross validation

- Randomly divide training data into *k* equal parts
  - $D_1,\ldots,D_k$
- For each *i*
  - Learn model $f_{\mathscr{D}\backslash\mathscr{D}_i}$ using data point not in $\mathscr{D}_i$
  - Estimate error of $f_{\mathscr{D}\backslash\mathscr{D}_i}$ on validation set $\mathscr{D}_i$:

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

$\mathscr{D} =$

	$\mathscr{D}_1$	$\mathscr{D}_2$	$\mathscr{D}_3$	$\mathscr{D}_4$	$\mathscr{D}_5$
$f_{\mathscr{D}\backslash\mathscr{D}_3}$	Train	Train	Validation	Train	Train

Use *k*-fold cross validation

> Randomly divide training data into *k* equal parts
  - $D_1, \ldots, D_k$

> For each *i*
  - Learn model $f_{\mathscr{D} \setminus \mathscr{D}_i}$ using data point not in $\mathscr{D}_i$
  - Estimate error of $f_{\mathscr{D} \setminus \mathscr{D}_i}$ on validation set $\mathscr{D}_i$:

$\mathscr{D} =$	$\mathscr{D}_1$	$\mathscr{D}_2$	$\mathscr{D}_3$	$\mathscr{D}_4$	$\mathscr{D}_5$
$f_{\mathscr{D} \setminus \mathscr{D}_3}$	Train	Train	Validation	Train	Train

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

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

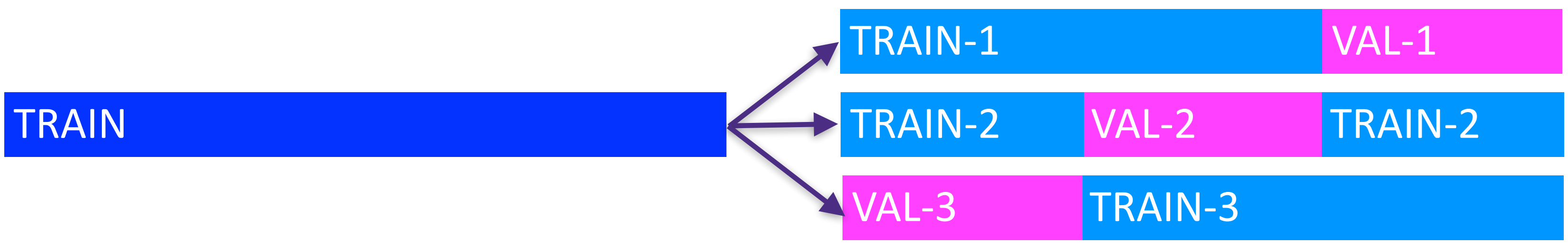
$$\text{error}_{k-\text{fold}} = \frac{1}{k} \sum_{i=1}^{k} \text{error}_{\mathscr{D}_i}$$

> k-fold cross validation properties:
  - Much faster to compute than LOO as $k \ll n$
  - More (pessimistically) biased – using much less data, only $n - \frac{n}{k}$
  - Usually, k = 10

Recap

> Given a dataset, begin by splitting into

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



> Model assessment: Use TEST to assess the accuracy of the model you output

- Never ever ever ever ever train or choose parameters based on the test data

Model selection using cross validation

- For $\lambda \in \{0.001, 0.01, 0.1, 1, 10\}$
  - For $j \in \{1, \ldots, k\}$
    - $$\hat{w}_{\lambda,\text{Train}-j} \leftarrow \arg\min_{w} \sum_{i \in \text{Train}-j} (y_i - w^T x_i)^2 + \lambda \|w\|_2^2$$
- $$\hat{\lambda} \leftarrow \arg\min_{\lambda} \frac{1}{k} \sum_{j=1}^{k} \sum_{i \in \text{Val}-j} (y_i - \hat{w}_{\lambda,\text{Train}-j}^T x_i)^2$$

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?

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

> After picking our 50 features, we then use CV with the training set to train ridge regression with regularization λ

> What's wrong with this procedure?

Recap

- Learning is…
  - Collect some data
    - E.g., housing info and sale price
  - Randomly split dataset into TRAIN, VAL, and TEST
    - E.g., 80%, 10%, and 10%, respectively
  - Choose a hypothesis class or model
    - E.g., linear with non-linear 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

Simple variable selection: LASSO for sparse regression

Sparsity

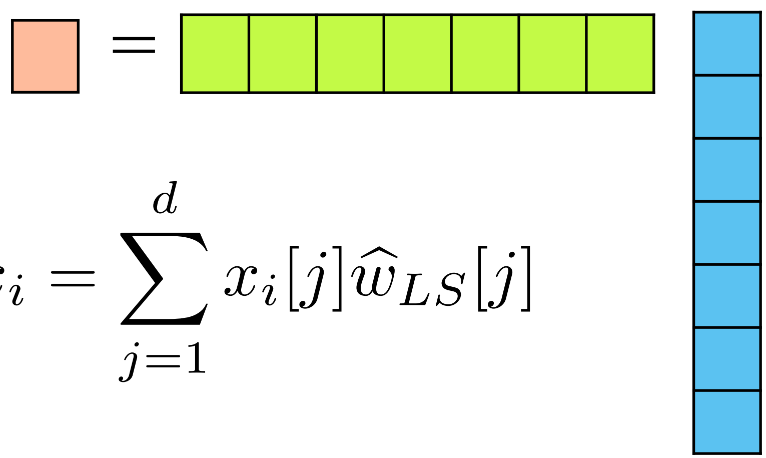
$$\widehat{w}_{LS} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$$

- Vector w is **sparse**, if many entries are zero

Sparsity

$$\widehat{w}_{LS} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$$

- Vector w is **sparse**, if many entries are zero
  - **Efficiency**: If size(w) = 100 Billion, each prediction $w^T x$ is expensive:
    - If w is sparse, prediction computation only depends on number of non-zeros in w



$$\widehat{y}_i = \widehat{w}_{LS}^{\top} x_i = \sum_{j=1}^{d} x_i[j]\widehat{w}_{LS}[j]$$

Sparsity

$$\widehat{w}_{LS} = \arg\min_w \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$$

- Vector w is **sparse**, if many entries are zero
  - **Interpretability**: What are the relevant features to make a prediction?

Lot size
Single Family
Year built
Last sold price
Last sale price/sqft
Finished sqft
Unfinished sqft
Finished basement sqft
floors
Flooring types
Parking type
Parking amount
Cooling
Heating
Exterior materials
Roof type
Structure style

Dishwasher
Garbage disposal
Microwave
Range / Oven
Refrigerator
Washer
Dryer
Laundry location
Heating type
Jetted Tub
Deck
Fenced Yard
Lawn
Garden
Sprinkler System

- How do we find "best" subset of features useful in predicting the price among all possible combinations?

Finding best subset: Exhaustive

- Try all subsets of size 1, 2, 3, … and one that minimizes validation error
- Problem?

Finding best subset: Greedy

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

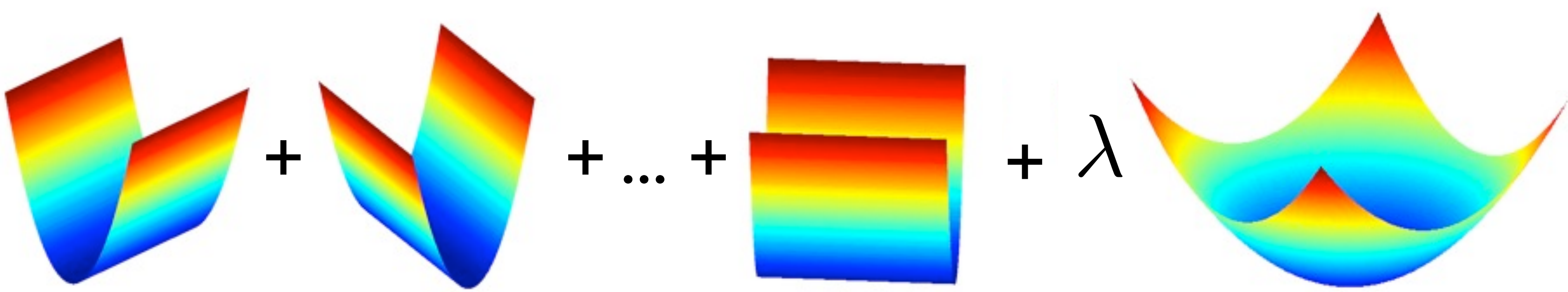
Combining forward and backward steps:
In forward algorithm, insert steps to remove features no longer as important

Lots of other variants, too.

Finding best subset: Regularize

Ridge regression makes coefficients small

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

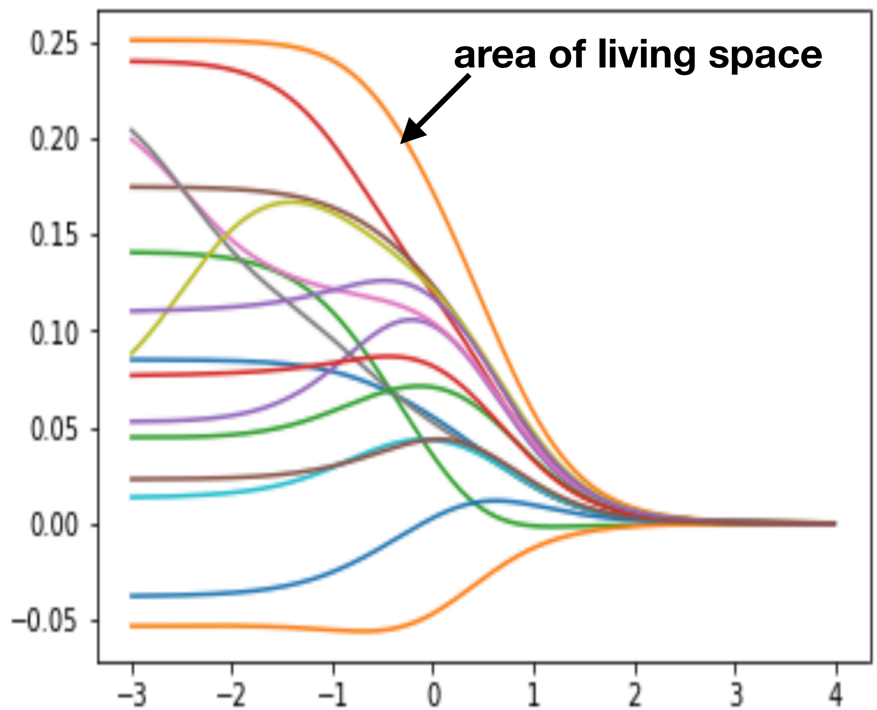


Finding best subset: Regularize

Ridge regression makes coefficients small

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

w_i's

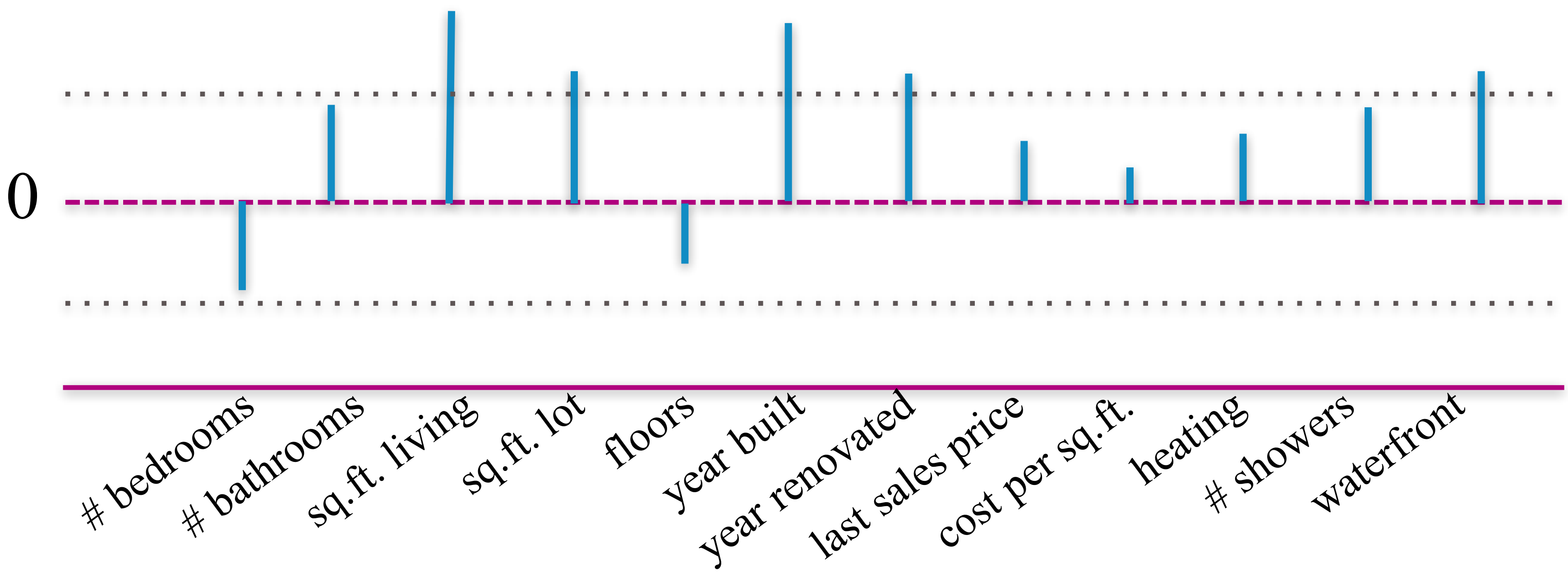


$\log_{10}(\lambda)$

Thresholded Ridge Regression

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

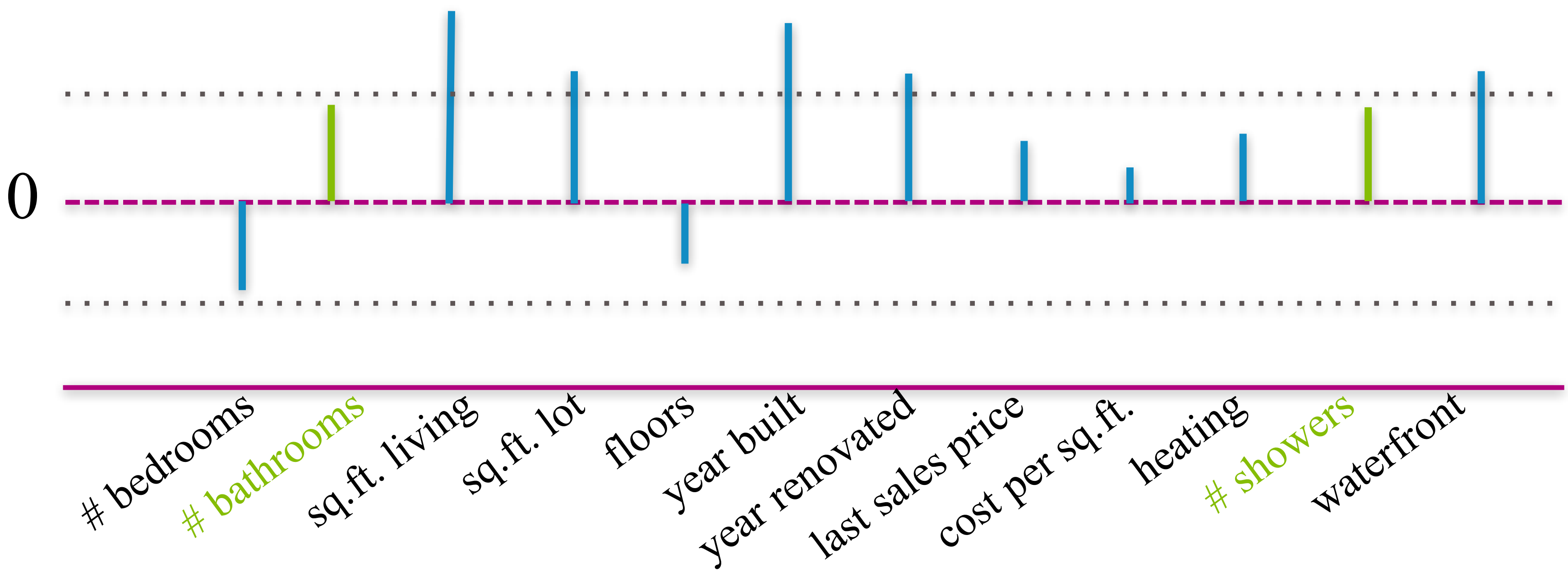
Why don't we just set **small** ridge coefficients to 0?



Thresholded Ridge Regression

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

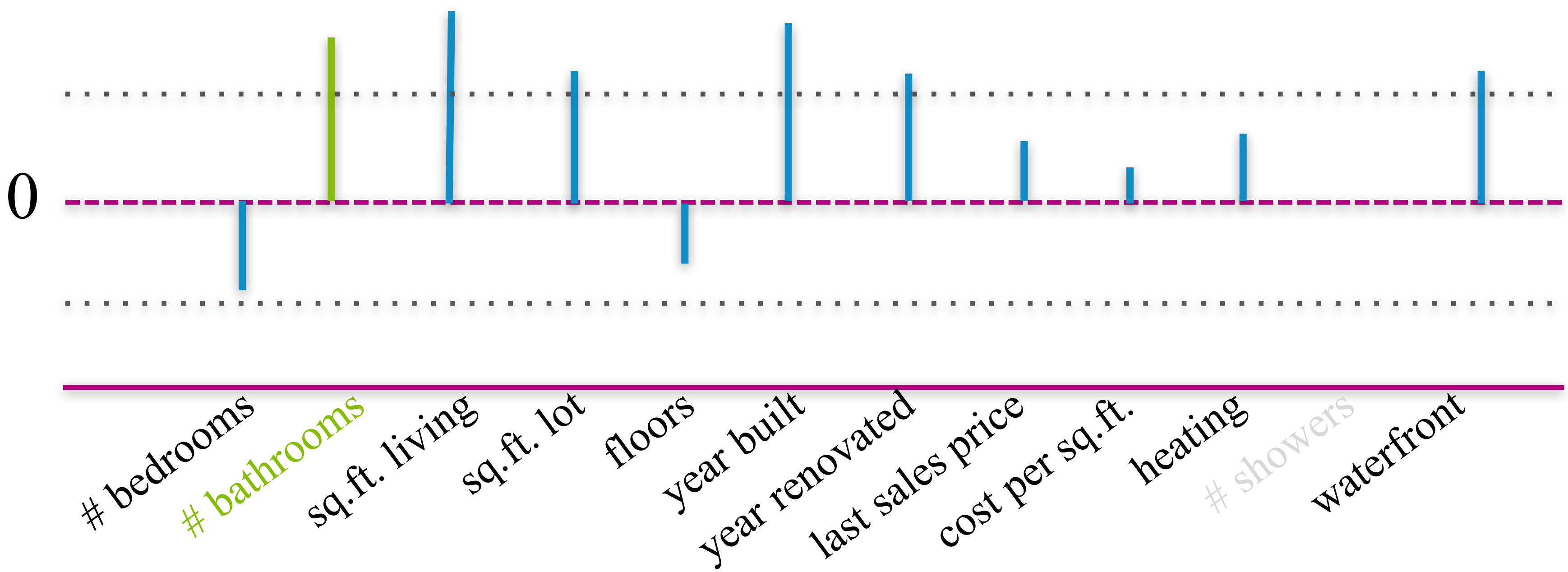
Consider two related features (bathrooms, showers)



Thresholded Ridge Regression

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

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

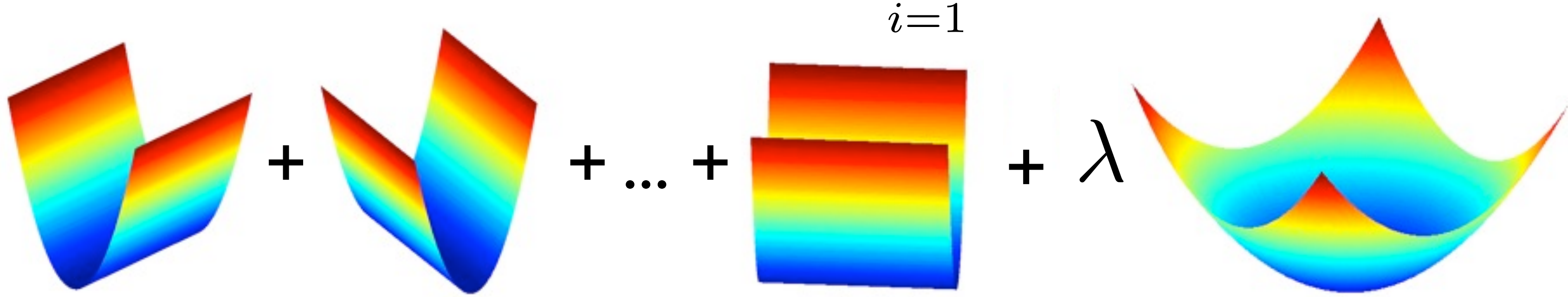


Can another regularizer perform selection automatically?

Recall Ridge Regression

- Ridge Regression objective:

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

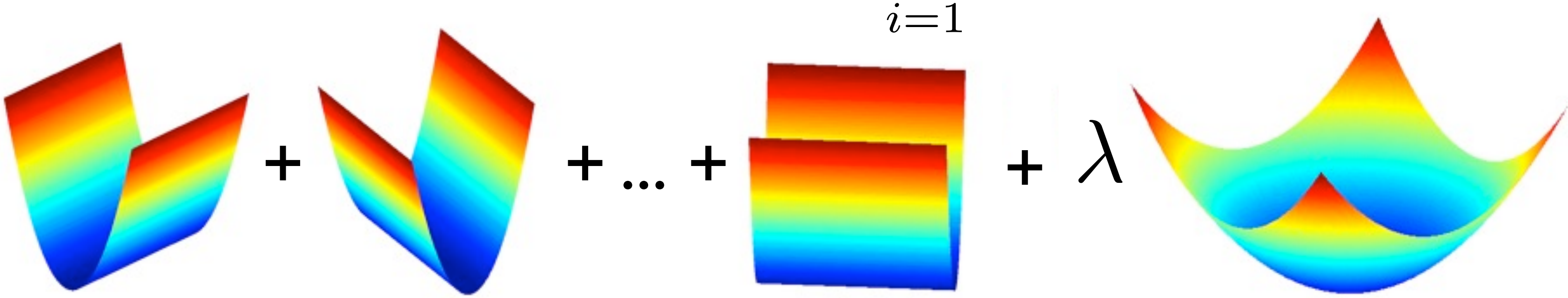


$$\|w\|_p = \left(\sum_{i=1}^{d} |w|^p \right)^{1/p}$$

Ridge vs. Lasso Regression

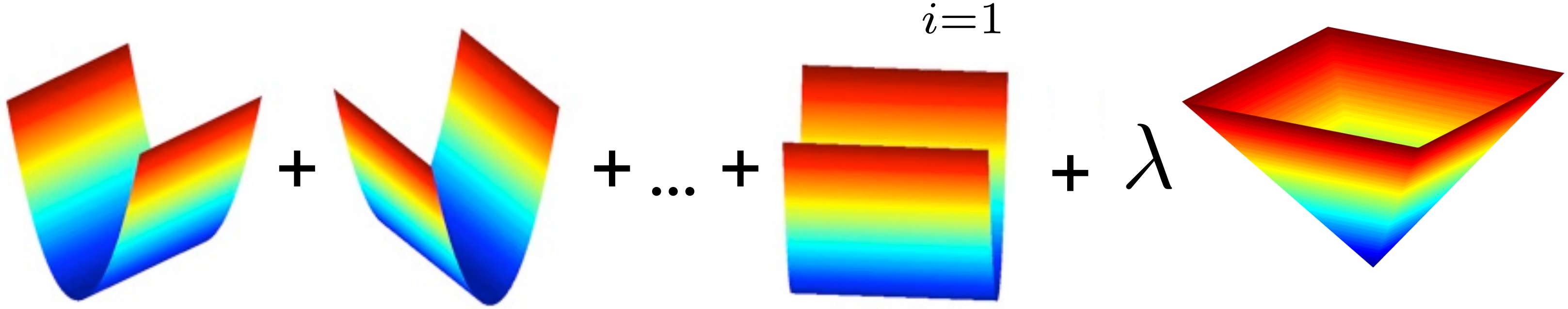
- Ridge Regression objective:

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



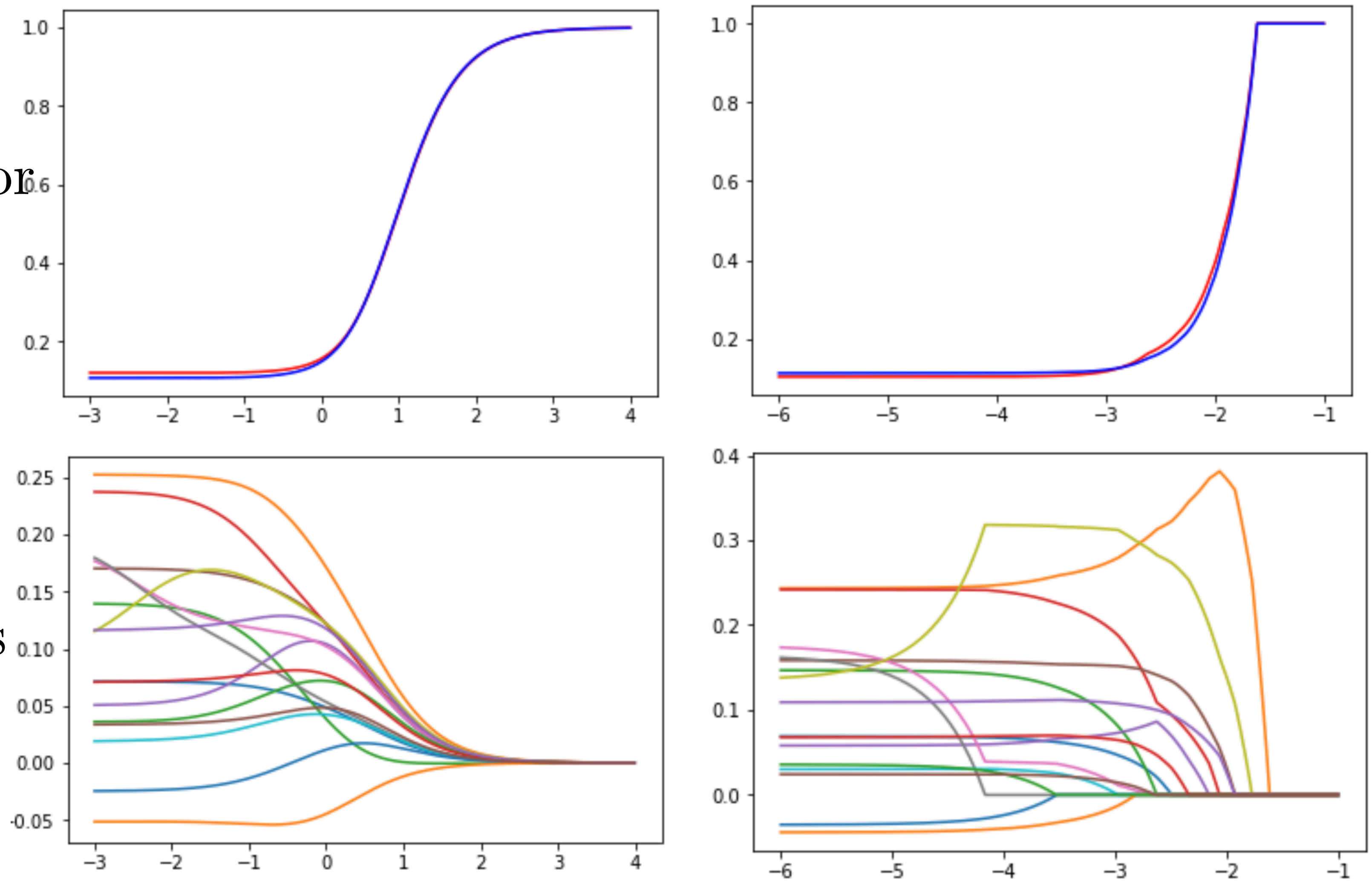
- Lasso objective:

$$\widehat{w}_{lasso} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_1$$



Example: house price with 16 features

test error is red and train error is blue



$\log_{10} \lambda$

Ridge regression

$\log_{10} \lambda$

Lasso regression

Lasso regression naturally gives sparse features

- **feature selection** with Lasso regression
    1. choose λ based on cross validation error
    2. keep only those features with non-zero (or not-too-small) parameters in w at optimal λ
    3. **retrain** with the sparse model and $\lambda = 0$

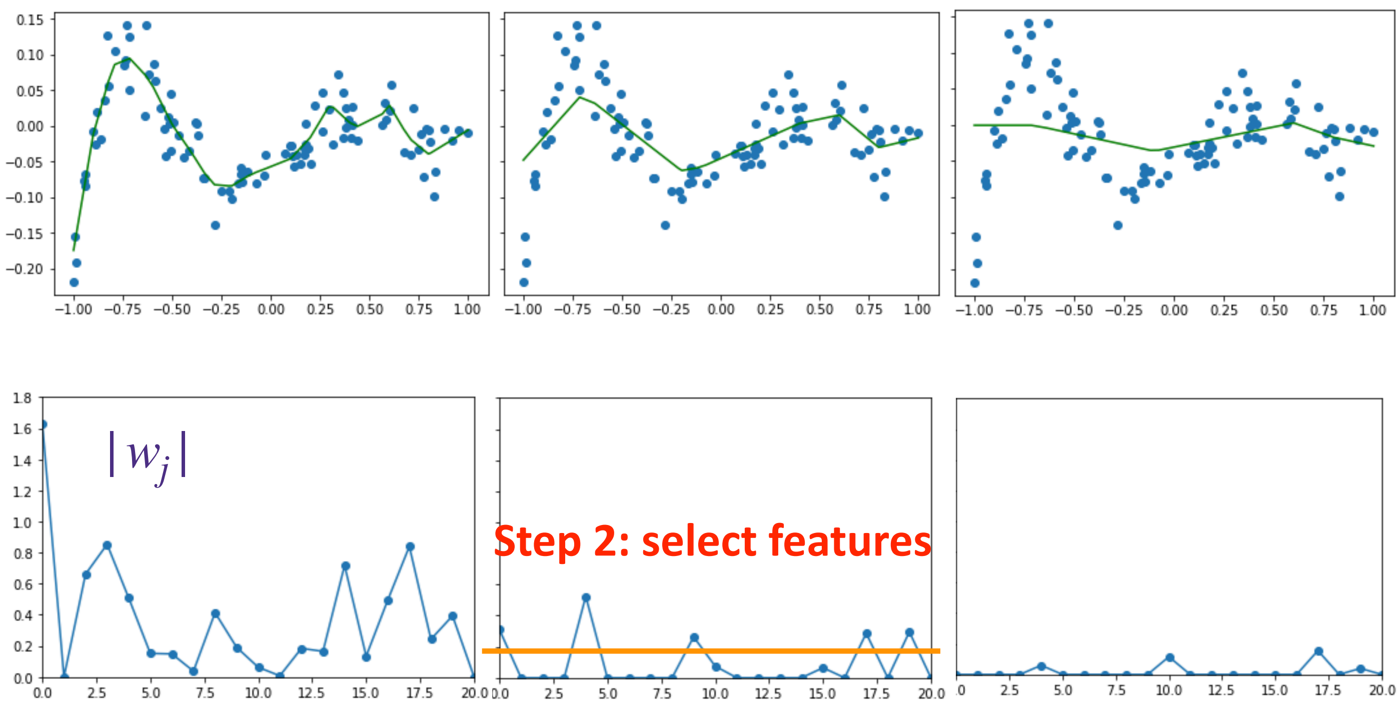
Example: piecewise-linear fit

- We use Lasso on the piece-wise linear example

$$h_0(x) = 1$$
$$h_i(x) = [x + 1.1 - 0.1i]^+$$

Step 1: find optimal $\lambda*$

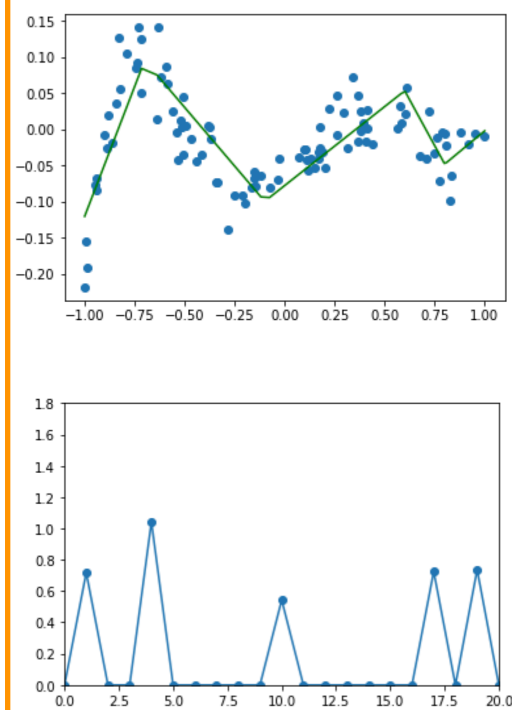
$$\text{minimize}_w \;\; \mathscr{L}(w) + \lambda\|w\|_1$$



$\lambda = 10^{-8}$ $\quad$ $\lambda = 10^{-4}$ $\quad$ $\lambda = 2 \times 10^{-4}$

Step 3: retrain

$$\text{minimize}_w \;\; \mathscr{L}(w)$$



$\lambda = 0$

but only use selected features

- de-biasing (via re-training) is critical!

Penalized Least Squares

$$\text{Ridge} : r(w) = ||w||_2^2 \qquad \text{Lasso} : r(w) = ||w||_1$$

$$\widehat{w}_r = \arg\min_w \sum_{i=1}^{n} \left(y_i - x_i^T w \right)^2 + \lambda r(w)$$

Penalized Least Squares

$$\text{Ridge} : r(w) = ||w||_2^2 \qquad \text{Lasso} : r(w) = ||w||_1$$

$$\widehat{w}_r = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2 + \lambda r(w)$$

For any $\lambda \geq 0$ for which $\hat{w}_r$ achieves the minimum, there exists a $\mu \geq 0$ such that

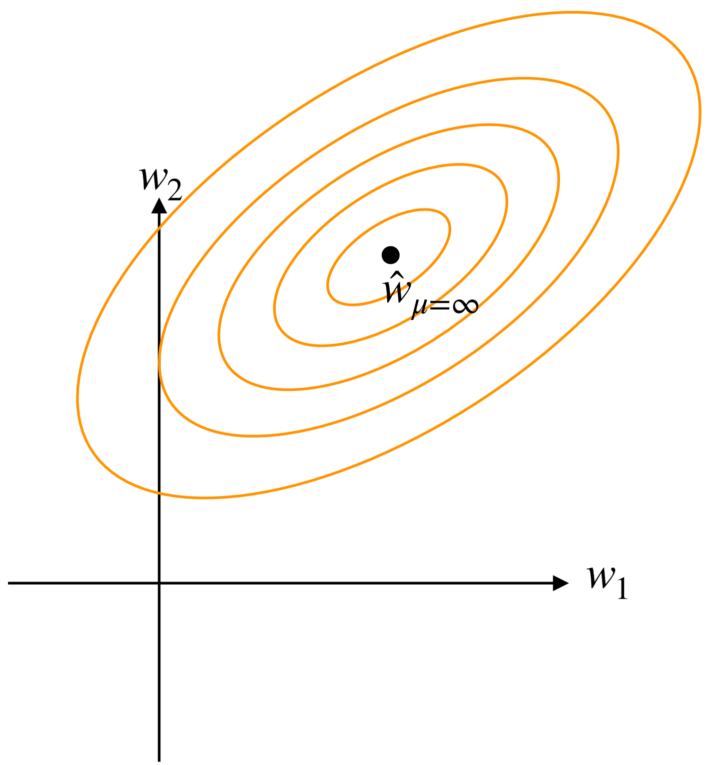
$$\widehat{w}_r = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2 \qquad \text{subject to } r(w) \leq \mu$$

Why does Lasso give sparse solutions?

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_1 \ \leq \ \mu$$

- the **level set** of a function $\mathscr{L}(w_1, w_2)$ is defined as the set of points (w_1, w_2) that have the same function value
- the level set of a quadratic function is an oval
- the center of the oval is the least squares solution $\hat{w}_{\mu=\infty} = \hat{w}_{\text{LS}}$

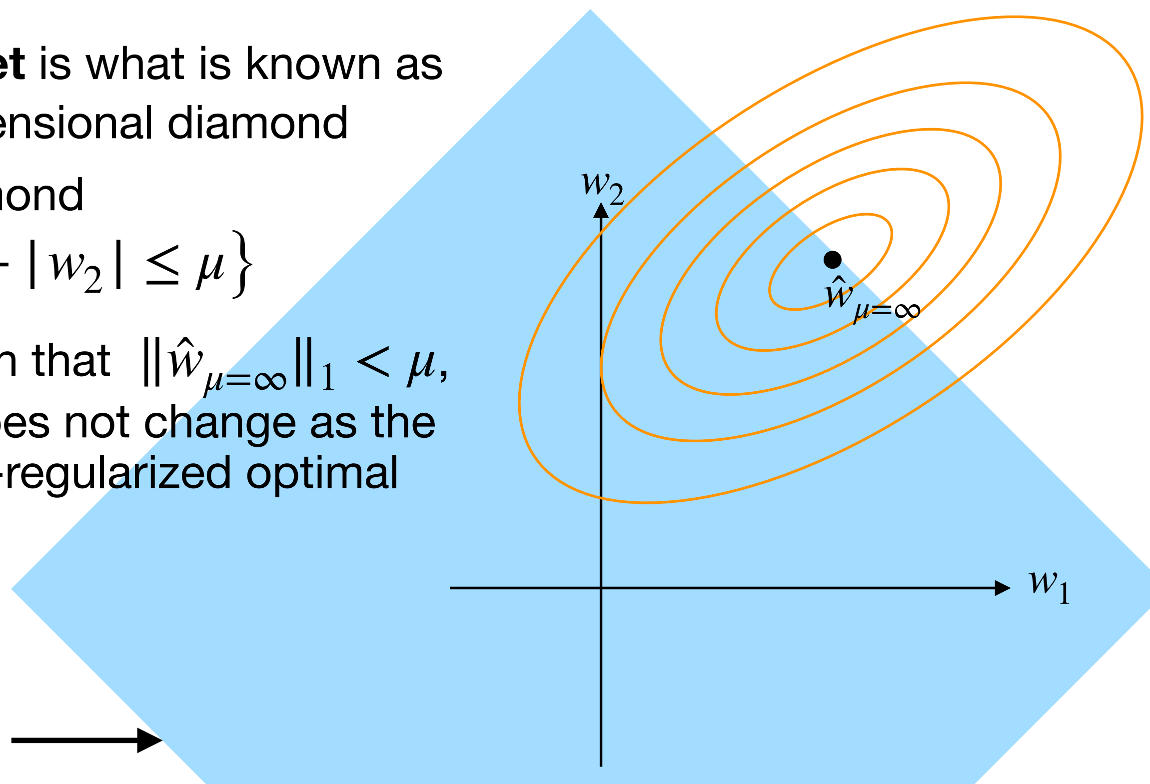


Why does Lasso give sparse solutions?

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_1 \leq \mu$$

- as we decrease μ from infinity, the feasible set becomes smaller
- the shape of the **feasible set** is what is known as L_1 ball, which is a high dimensional diamond
- In 2-dimensions, it is a diamond
$$\left\{(w_1, w_2) \,\middle|\, |w_1| + |w_2| \leq \mu \right\}$$
- when μ is large enough such that $\|\hat{w}_{\mu=\infty}\|_1 < \mu$, then the optimal solution does not change as the feasible set includes the un-regularized optimal solution



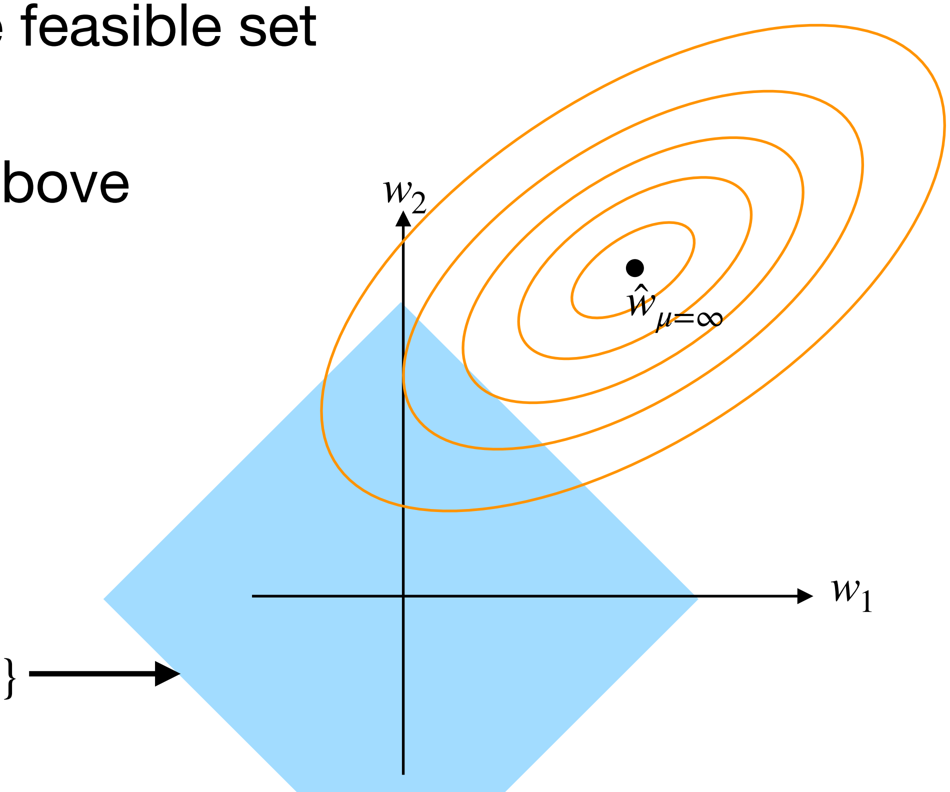
feasible set: $\{w \in \mathbb{R}^2 \mid \|w\|_1 \leq \mu\}$ →

Why does Lasso give sparse solutions?

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_1 \leq \mu$$

- As μ decreases (which is equivalent to increasing regularization) the feasible set (blue diamond) shrinks
- The optimal solution of the above optimization is



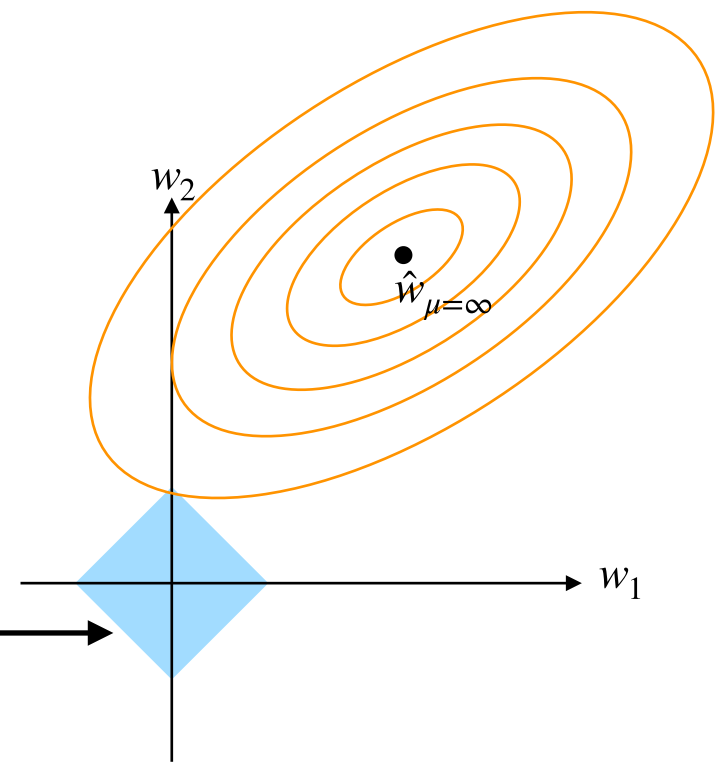
feasible set: $\{w \in \mathbb{R}^2 \mid \|w\|_1 \leq \mu\}$

Why does Lasso give sparse solutions?

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_1 \ \leq \ \mu$$

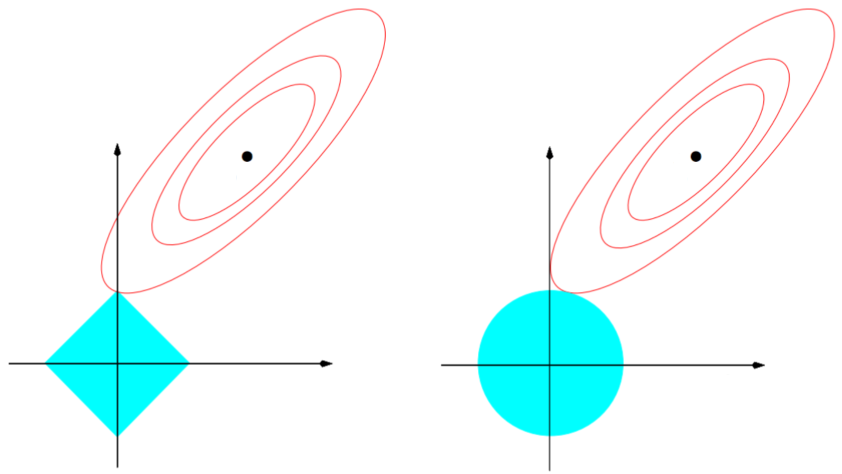
- For small enough μ, the optimal solution becomes **sparse**
- This is because the L_1-ball is “pointy”,i.e., has sharp edges aligned with the axes



feasible set: $\{w \in \mathbb{R}^2 \mid \|w\|_1 \leq \mu\}$

Penalized Least Squares

- Lasso regression finds sparse solutions, as L_1-ball is "pointy"
- Ridge regression finds dense solutions, as L_2-ball is "smooth"



$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2 \qquad\qquad \text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_1 \le \mu \qquad\qquad \text{subject to} \quad \|w\|_2^2 \le \mu$$

Questions?