Regularization

Sewoong Oh

CSE446
University of Washington

Sensitivity: how to detect overfitting in order to prevent it

- consider a linear predictor

$$f(x) = w_0 + w_1 x[1] + w_2 x[2] + \cdots + w_d x[d]$$

- if $|w_i|$ is large then the predictor is very **sensitive** to small changes in x_i lead to large changes in the prediction

- large sensitivity can lead to overfitting and poor generalization or models that overfit tend to have large sensitivity

- for $x[0] = 1$ there is no sensitivity, as it is a constant

- This suggests that we would like w or $(w_{1:d}$ if $x[0] = 1)$ not to be large

Regularizer

- we measure the size of w using a **regularizer** function $r : \mathbf{R}^d \rightarrow \mathbf{R}$

- $r(w)$ is the measure of the size of w (or $w_{1:d}$)

- **quadratic regularizer** (a.k.a L2 or sum-of-squares)

$$r(w) = \|w\|^2 = w_1^2 + w_2^2 + \cdots + w_d^2$$

- **absolute value regularizer** (a.k.a. L1)

$$r(w) = \|w\|_1 = |w_1| + |w_2| + \cdots + |w_d|$$

- What is wrong with

$$r(w) = w_1 + w_2 + \cdots + w_d$$

Adding a regularizer to the loss

- we want small empirical risk (without normalization by $\frac{1}{n}$)

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

- we want small sensitivity

$$r(w)$$

- these two objectives are traded off via regularized loss

$$\sum_{i=1}^{n} (w^T x_i - y_i)^2 \; + \; \lambda \, r(w)$$

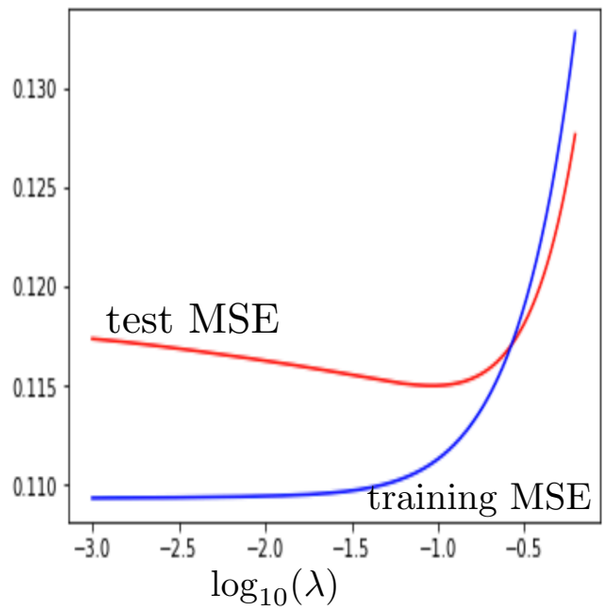
- $\lambda \geq 0$ is the **regularization parameter** (or **hyper parameter**) and is one of the most relevant hyper parameter to tune in training

- solve the optimization problem for a choice of $r(w)$ to choose w that minimizes the regularized loss

- $$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2 \; + \; \lambda \, r(w)$$
- when $\lambda = 0$ this reduces to the standard quadratic loss

- this defines a **family** of predictors, each (hyper)-parametrized by λ
- in practice, we try out tens of values of λ in a wide range

- we use validation to choose the right λ
- we choose the largest λ that gives near minimum test error, that is least sensitive predictor that generalizes well



to be precise, this process is flawed and we should use a more principled way using cross-validation (which is at the end of this chapter)

Ridge regression

- **ridge regression**: quadratic loss and quadratic regularizer

- also called **Tykhonov regularized least squares**

$$\mathscr{L}(w) + \lambda r(w) \;=\; \underbrace{\sum_{i=1}^{n} (w^T x_i - y_i)^2}_{\|\mathbf{X}w - \mathbf{y}\|_2^2} + \;\lambda\; \underbrace{\sum_{j=1}^{d} w_j^2}_{\|w\|_2^2}$$

$$\hat{w}_{\text{ridge}} \;=\; \arg\min_{w} \mathscr{L}(w) + \lambda\, r(w)$$

- the optimal solution is also analytical (or closed-form)

$$\hat{w}_{\text{ridge}} \;=\; \left(\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I}_{d\times d}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

where $\mathbf{I_{d\times d}}$ is the d-dimensional identity matrix

- this follows from the fact that

$$\mathscr{L}(w) + \lambda r(w) \;=\; \|\mathbf{X}w - \mathbf{y}\|_2^2 + \lambda \|w\|_2^2$$

$$= \left\| \begin{bmatrix} \mathbf{X} \\ \lambda^{\mathbf{1/2}} \mathbf{I_{d\times d}} \end{bmatrix} w - \begin{bmatrix} \mathbf{y} \\ \mathbf{0_d} \end{bmatrix} \right\|_2^2$$

  where $\mathbf{I_{d\times d}}$ is the $d \times d$-dimensional identity matrix, and $\mathbf{0_d}$ is the d-dimensional zero vector

- the gradient with respect to w is

$$2 \begin{bmatrix} \mathbf{X}^T & \lambda^{1/2}\mathbf{I_{d\times d}} \end{bmatrix} \left(\begin{bmatrix} \mathbf{X} \\ \lambda^{1/2}\mathbf{I_{d\times d}} \end{bmatrix} w - \begin{bmatrix} \mathbf{y} \\ \mathbf{0_d} \end{bmatrix} \right)$$

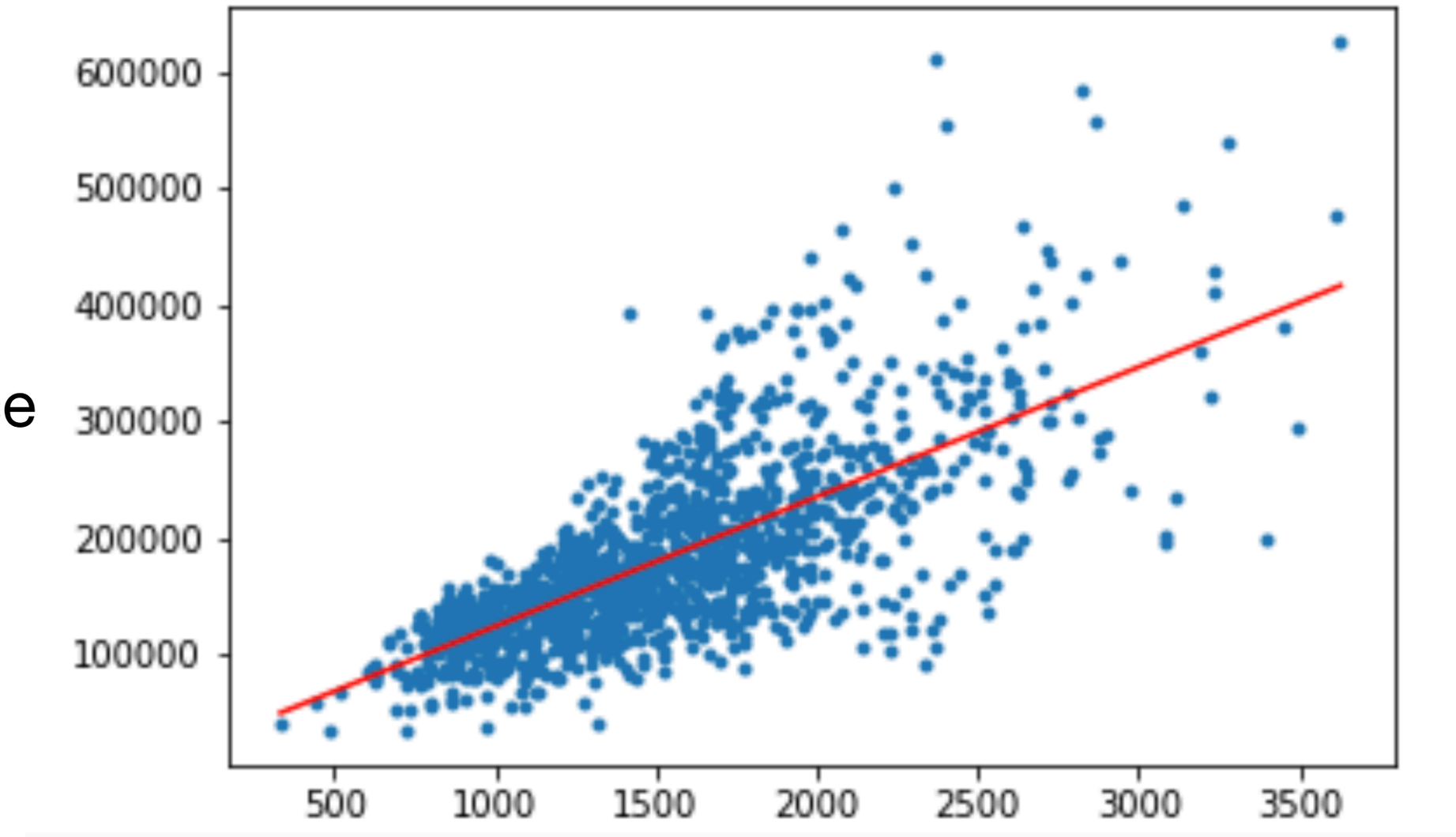
$$= 2\left(\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I_{d\times d}}\right) w - 2 \begin{bmatrix} \mathbf{X}^T & \lambda^{1/2}\mathbf{I_{d\times d}} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{0_d} \end{bmatrix}$$

$$= 2\left(\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I_{d\times d}}\right) w - 2\mathbf{X}^T\mathbf{y}$$

- Setting this gradient to zero, we get

$$\hat{w}_{\text{ridge}} \;=\; (\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I_{d\times d}})^{-1}\mathbf{X}^T\mathbf{y}$$

Example: housing price (data from kaggle)

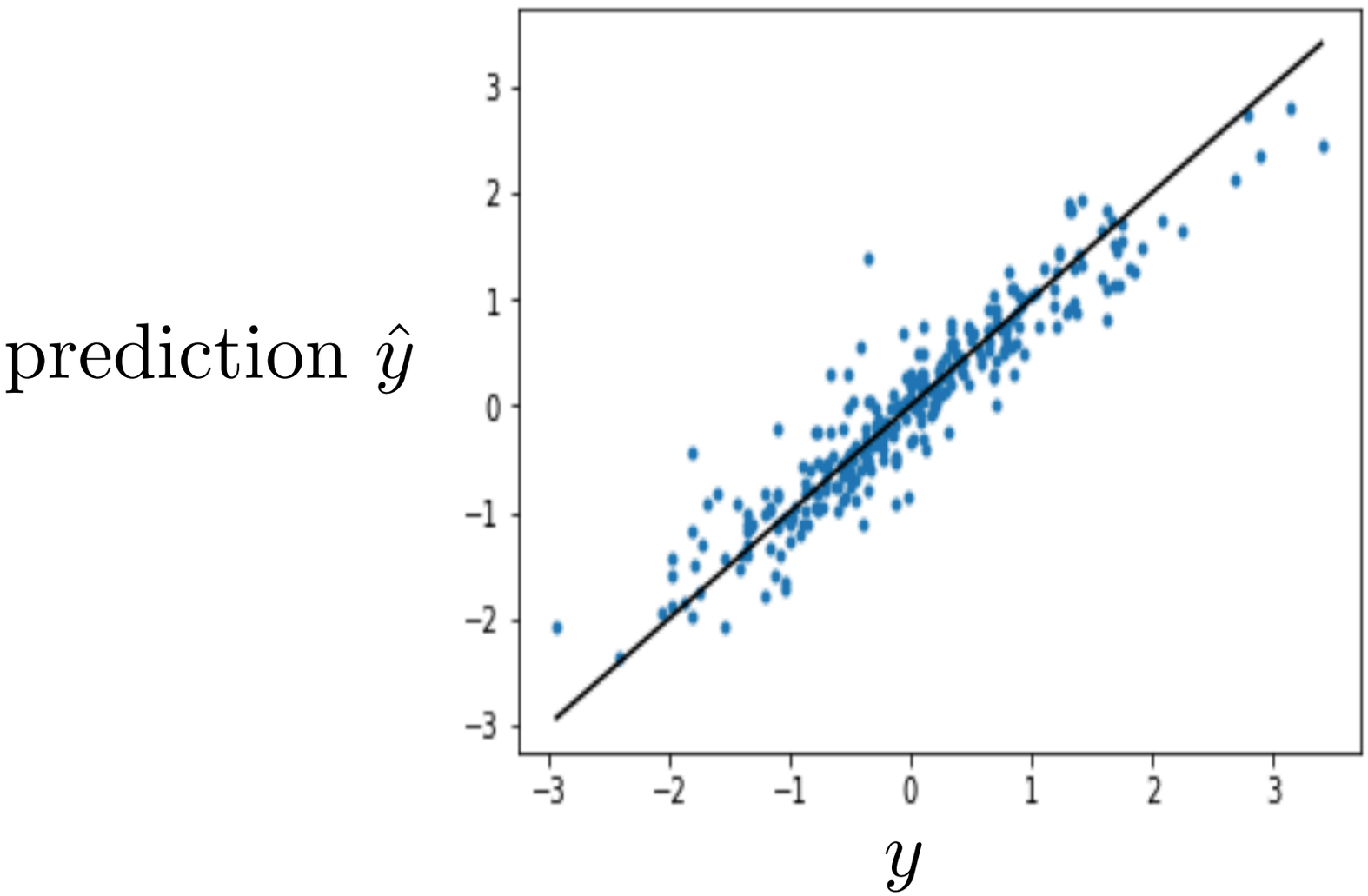


- sale prices of 1459 homes in Ames, Iowa from 2006 to 2010
- out of 80 features, we use 16
- we manually remove 4 outliers with are>4000 sq.ft.
  we will learn outlier detection later

Input features

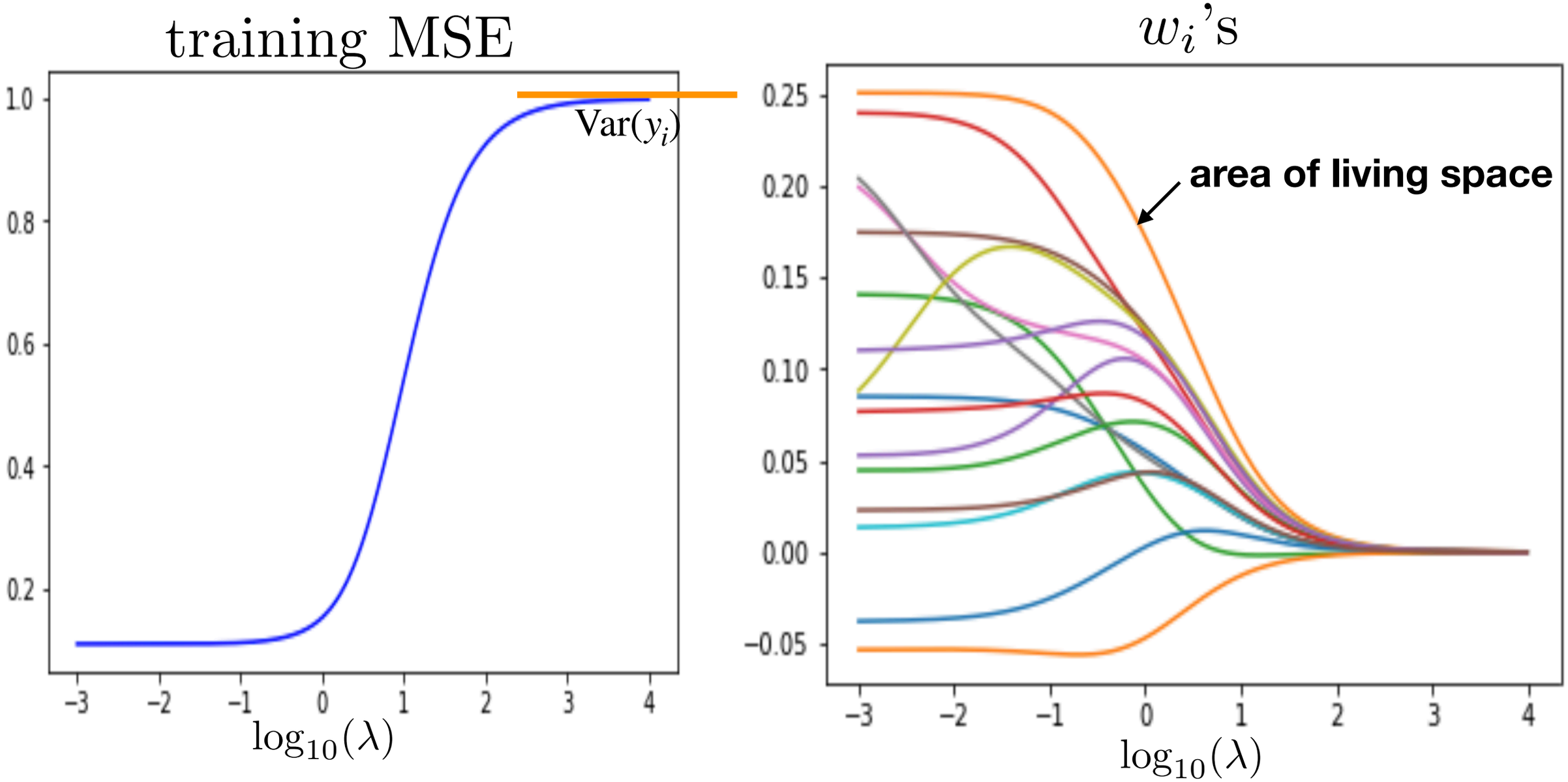
- house price input data:
  area of living space
  garage (no:0, yes:1)
  year built
  area of lot
  year of last remodel
  area of basement
  area of first floor
  area of second floor
  number of bedrooms (above ground)
  number of kitchens (above ground)
  number of fireplaces
  area of garage
  area of wooden deck
  number of half bathrooms
  overall condition (1-10)
  overall quality of materials and finish (1-10)
  number of rooms (above ground)

Example: regression (with no regularization)



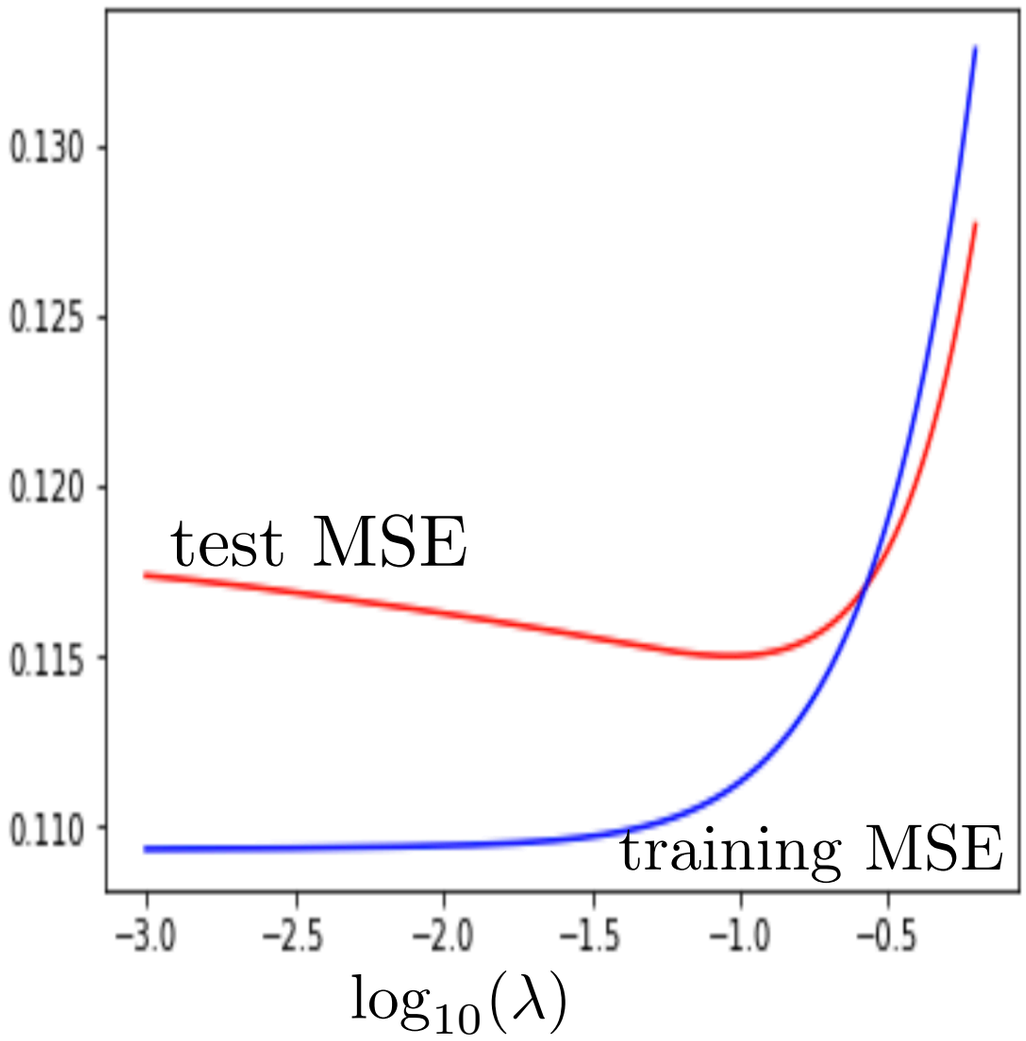
- split data randomly into 1164 training and 291 test
- target is log(price)
- **standardize** all features (and log(price)): shift and scale each feature (and the outcome log(price)) such that they are zero mean and variance one
- training error = 0.1093
- test error = 0.1175
- plot shows all 291 test points

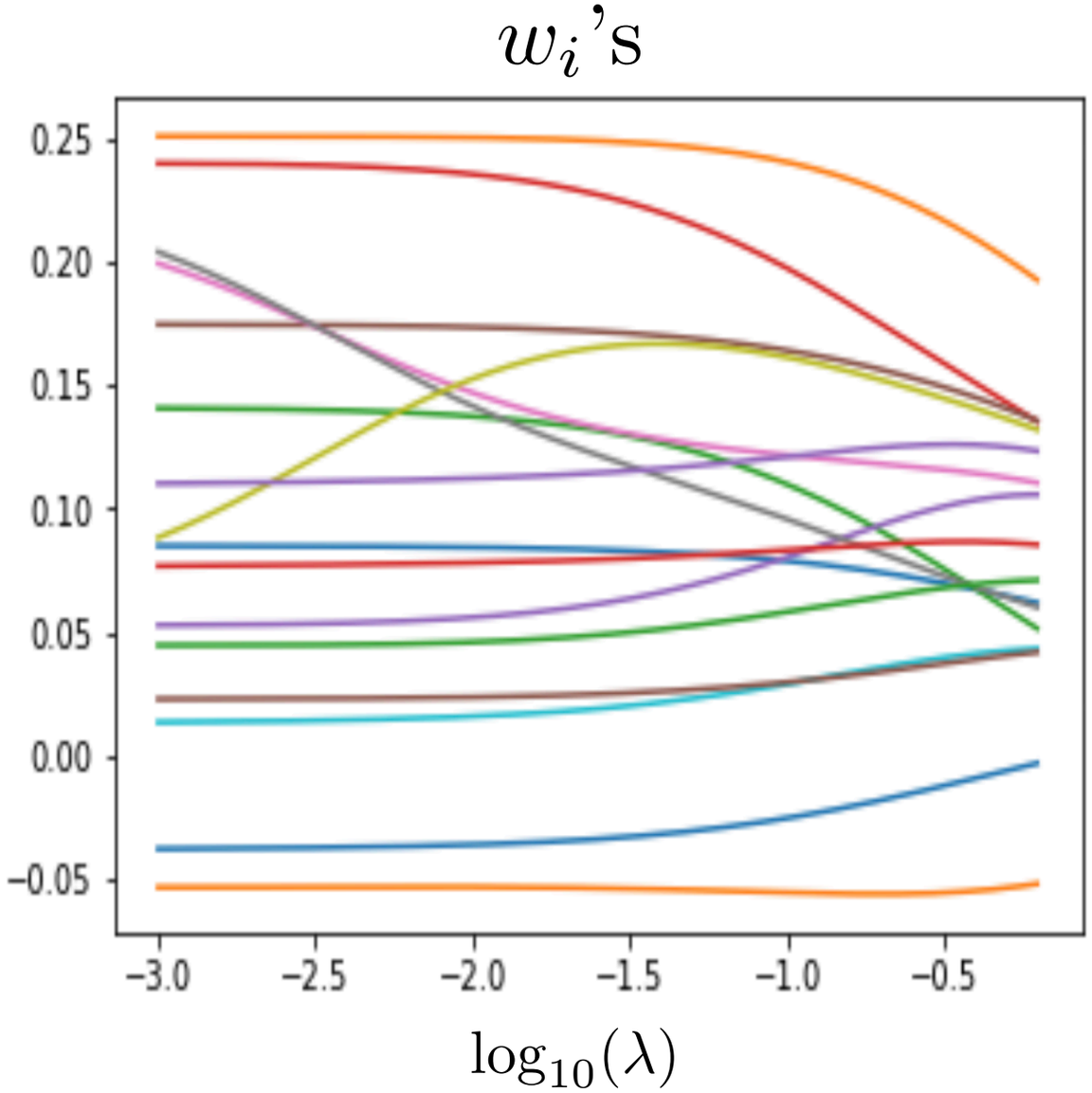
Example: Ridge regression $\text{minimize} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda \|w\|_2^2$



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

Example: Ridge regression





- optimal regularizer lambda= 0.1412
- slightly improves the test performance
- from test MSE = 0.1175 to test MSE = 0.1147
- this gain comes from shrinking w's to get a less sensitive predictor

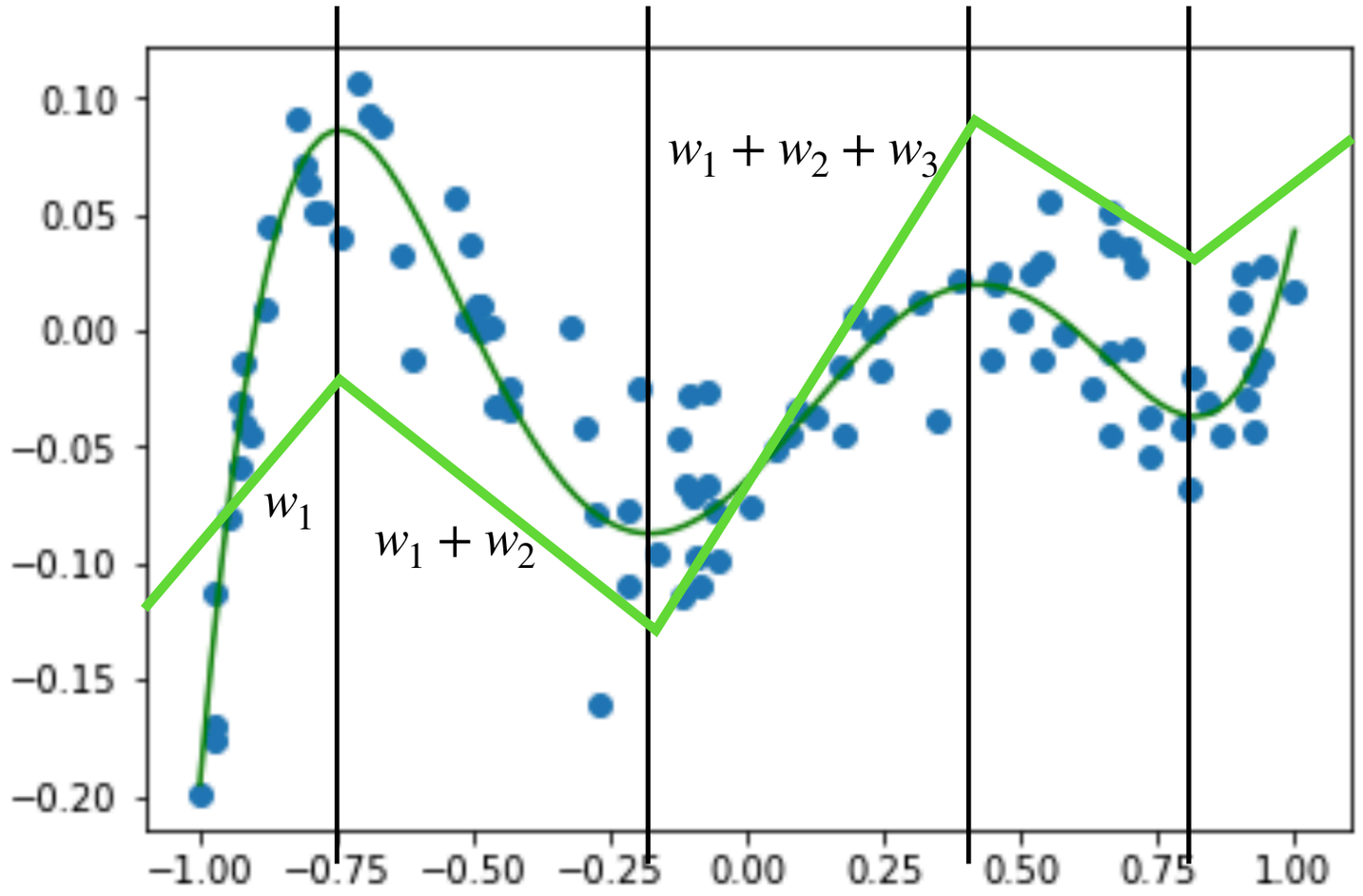
Example: piecewise linear fit

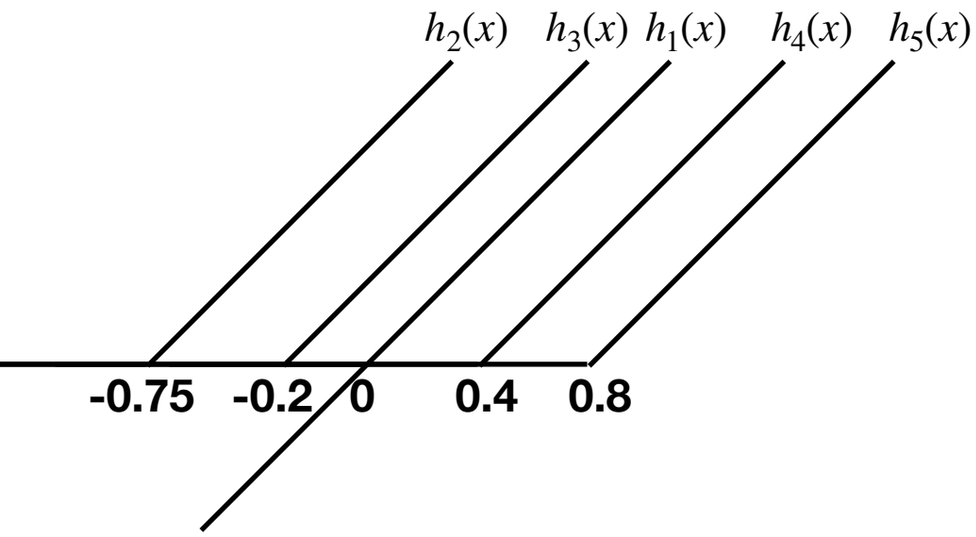
we fit a linear model: $f(x) = w_0 + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$

- with a specific choice of features using piecewise linear functions

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

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

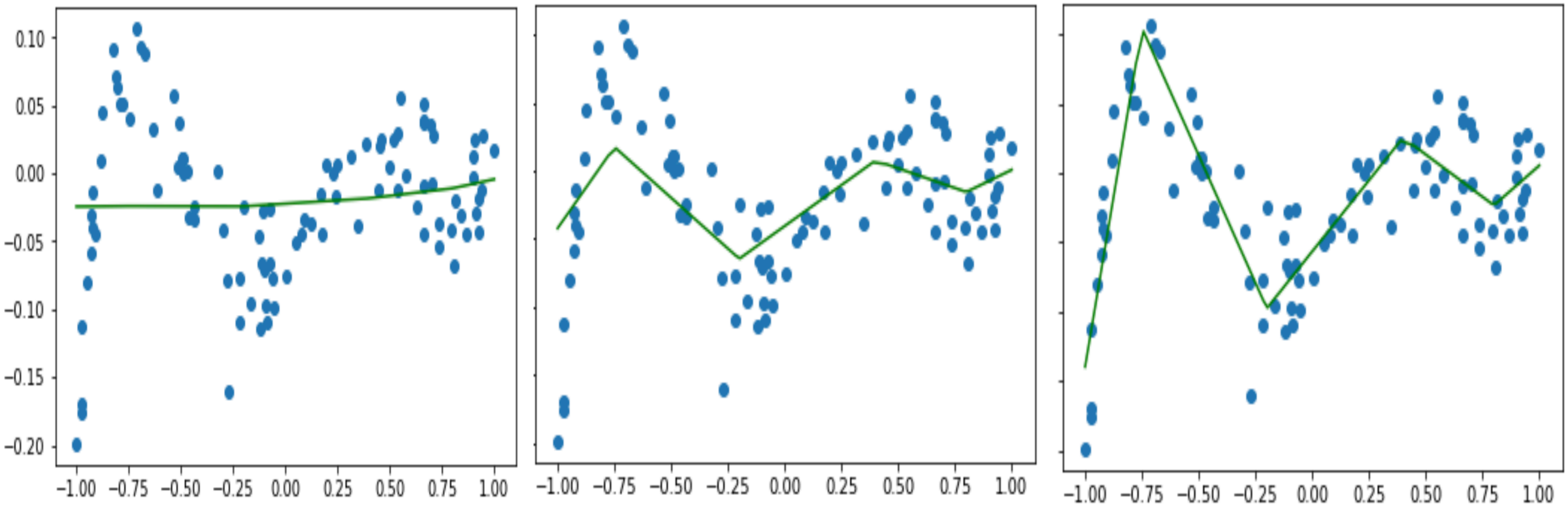




the weights capture the change in the slopes

demo5_piece

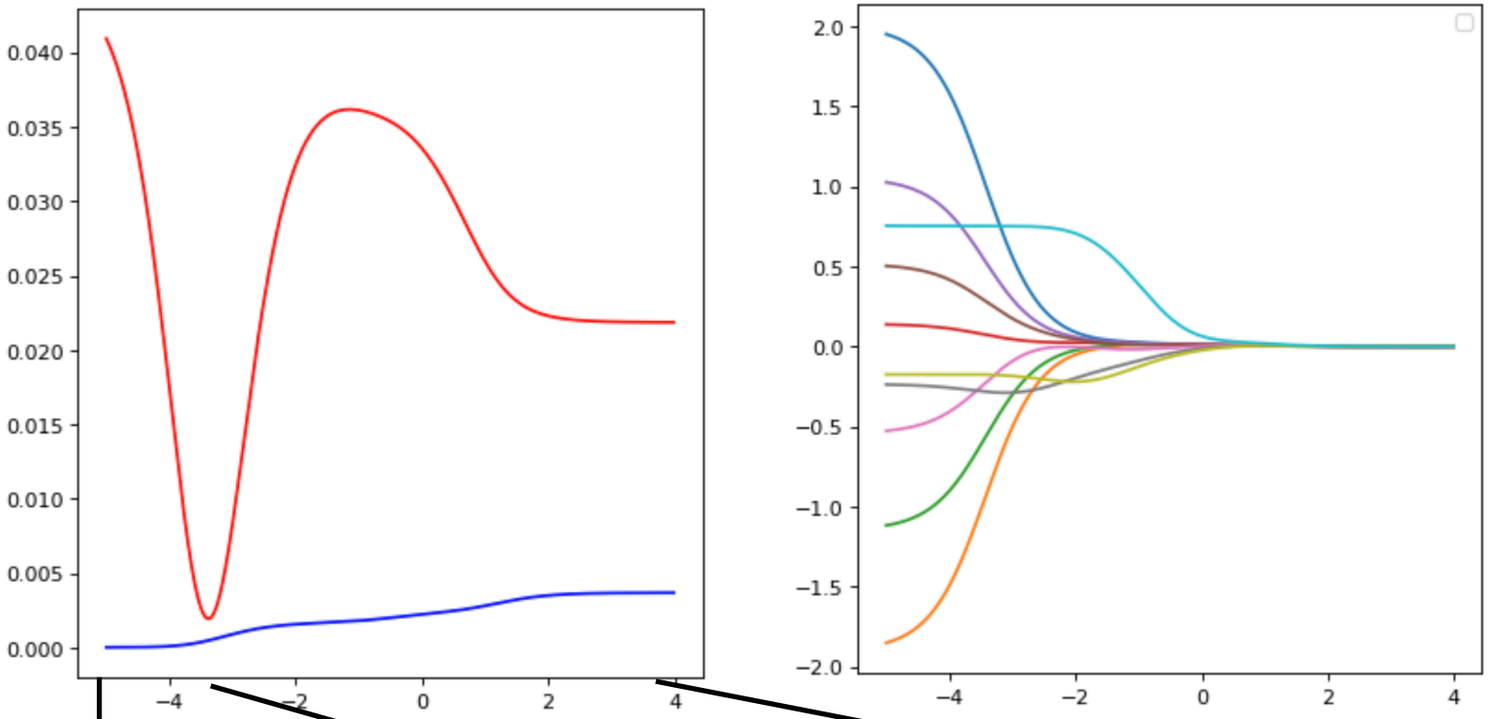
Example: piecewise linear fit

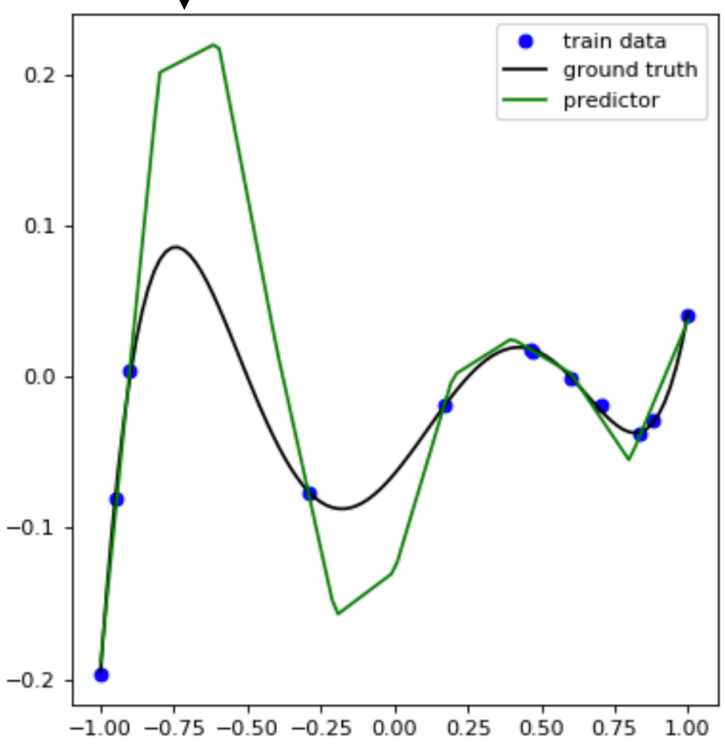


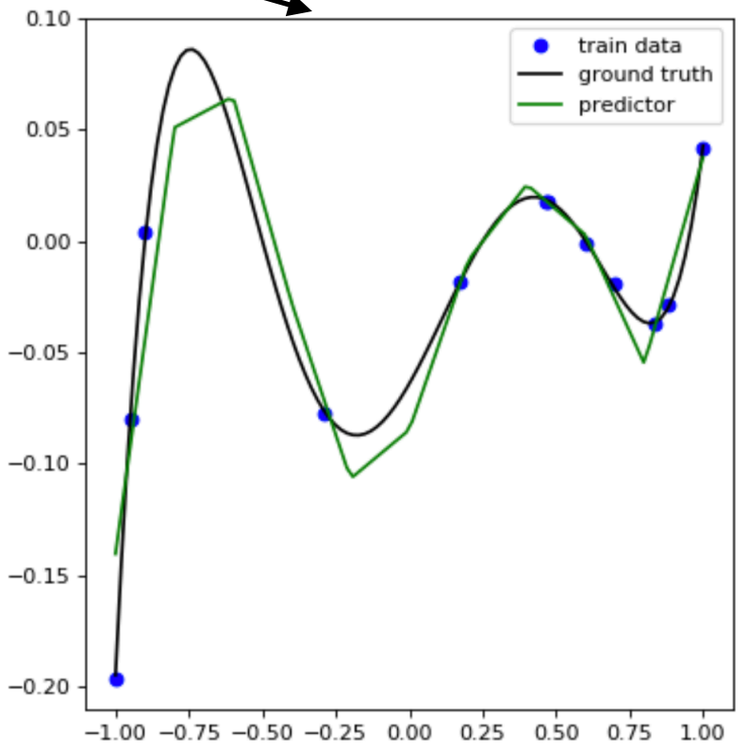
$\lambda = 1$ $\lambda = 0.005$ $\lambda = 0.000001$

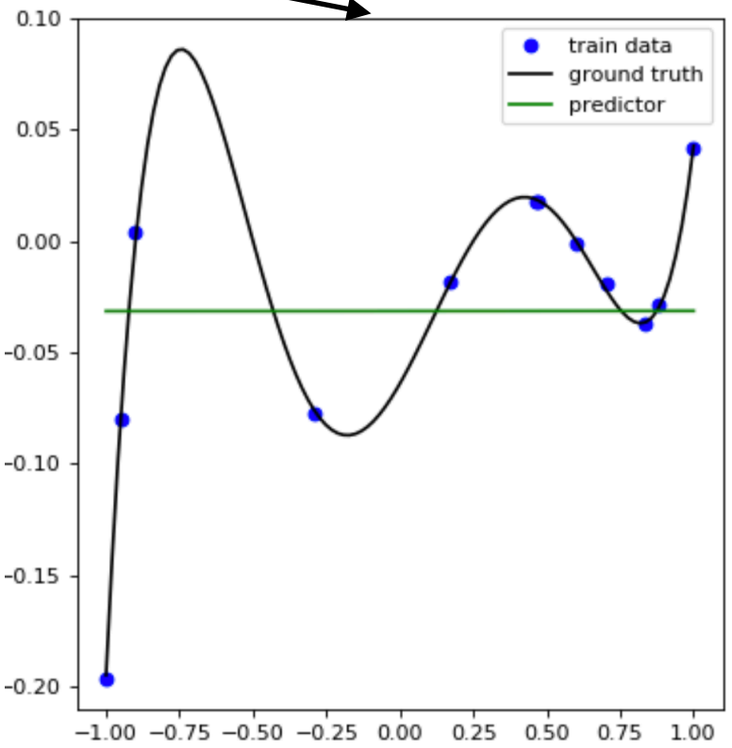
- features: $h(x) = (1, x, [x + 0.75]^+, [x + 0.2]^+, [x - 0.4]^+, [x - 0.8]^+)$
- lambda=1 gives
  w= [-0.0377, 0.00140, -0.00177, 0.01014, 0.00875, 0.01482]
- lambda=1e-6 gives
  w=[-0.1382, 0.97846, -1.3467, 0.57375, -0.32763, 0.2658]

Piecewise linear with 10 parameters

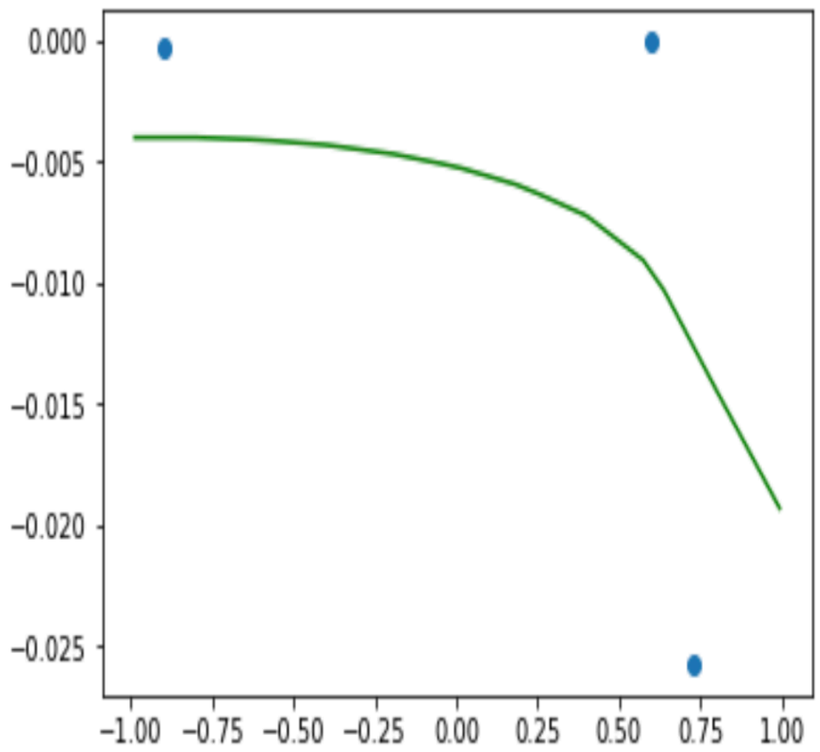




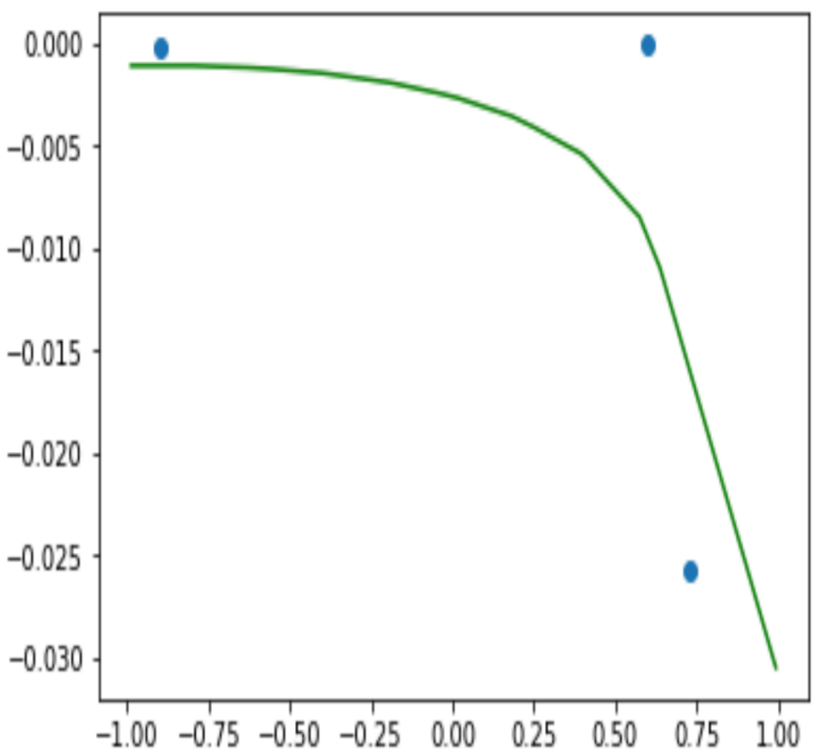




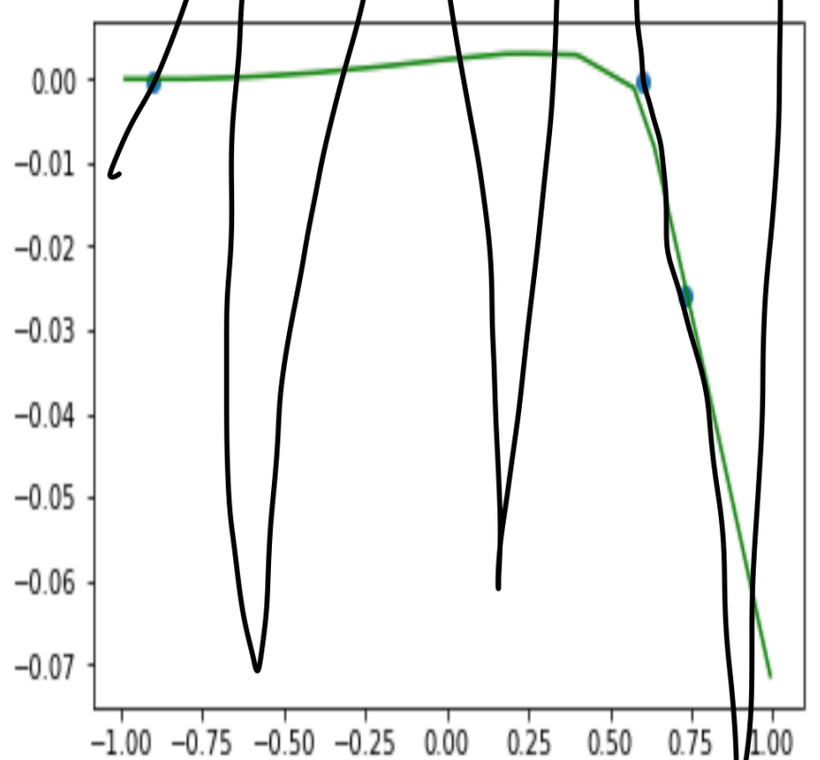
Fitting predictors with more parameters than data points



λ=10



λ=3

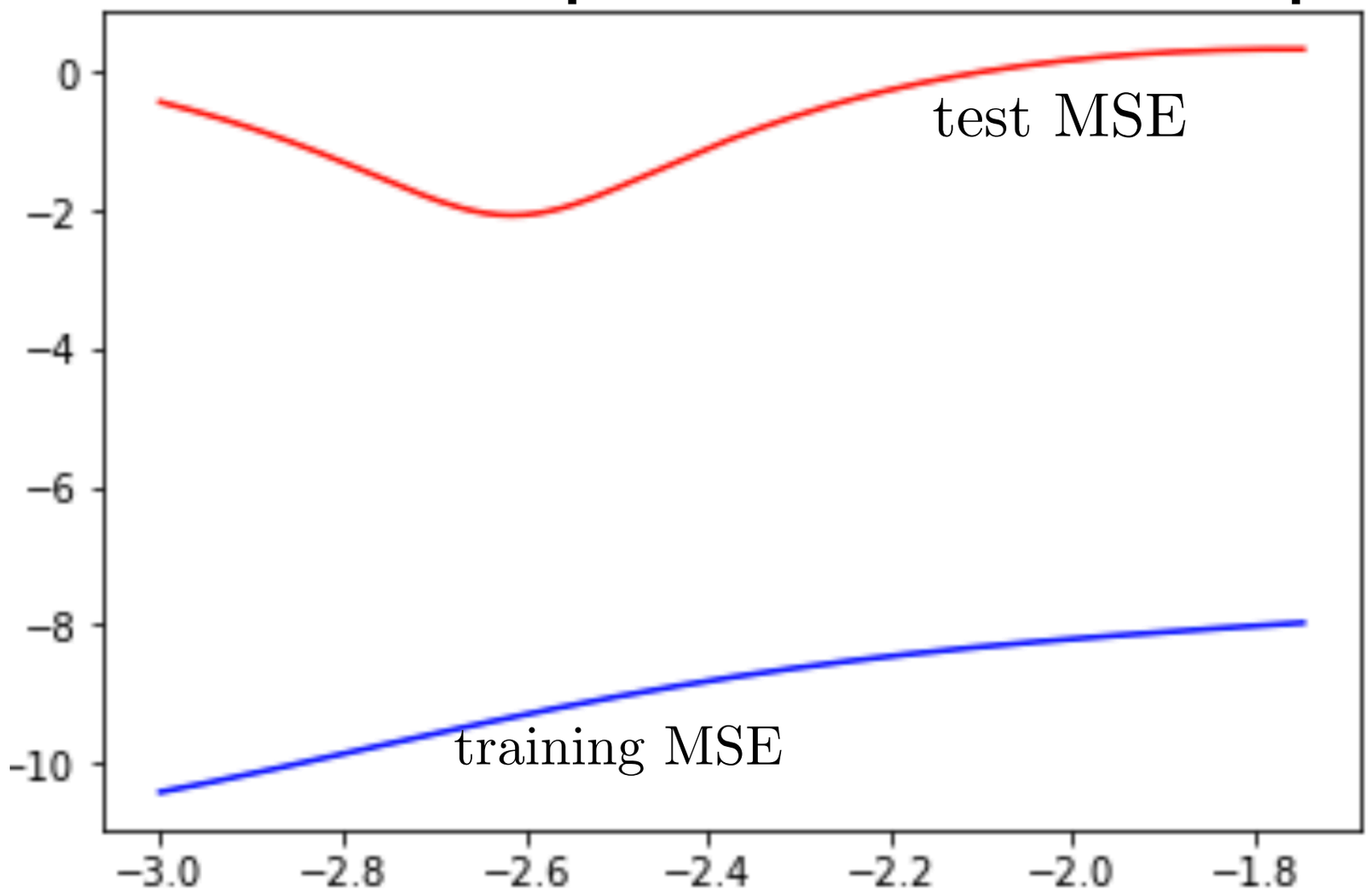


λ=1

- in general, fitting a model with more parameters than data points does not make sense
- but one can fit such **over-parametrized** models with regularization
- 10 piece linear model with 10 parameters

Model complexity and lambda

Fitting predictors with more parameters than data points



- Having large regularization limits what type of models we can choose from, hence enforces simpler models

Theoretical analysis

- note that theoretical analysis is for the purpose of understanding the performance of a proposed approach (in this case Ridge Regression)

- for this purpose, we assume a specific model and analyze the performance

- in particular, such theoretical analysis cannot be done in real problems, as you do not know the underlying model

- however, it tells you how performance depends on the problem parameters (like dimension, noise variance, true model parameters, number of samples, and regularization parameter)

Exercise: simple example

- we analyze the resulting true error for a simple model,
  to illustrate how error depends on the parameters of the problem
  (training sample size n, number of features d, noise variance σ^2,
  ground truth model parameter w) and the choice of regularization
  parameter λ
- model: $y_i = w^T x_i + \varepsilon_i$
  where $x_i \in \mathbb{R}^d$, $y_i, \varepsilon_i \in \mathbb{R}$
  - we further assume that $\varepsilon_i \sim N(0,\sigma^2)$ is zero mean Gaussian with variance σ^2
  - each feature is also independently a zero mean Gaussian with unit variance, i.e. $x_i[j] \sim N(0,1)$ for all $j \in [d]$
  - $[d] = \{1,\cdots,d\}$ denotes the set of first d positive integers
  - $w \in \mathbb{R}^d$ is the ground truth model parameter, which is a fixed deterministic vector

Exercise: simple example

- the **linear least squares predictor** is given by

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

where we used the fact that $\mathbf{y} = \mathbf{X}w + \varepsilon$

$X^T X = n \cdot \mathbb{I}_{d\times d}$

$= n \cdot \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}_{d}$

- again using the fact that for any $j \in [d]$,

$$(\mathbf{X}^T\mathbf{X})_{jj} = \sum_{i=1}^{n} x_i[j]^2 \simeq \sum_{i=1}^{n} \mathbb{E}[x_i[j]^2] = n$$

which follows from strong law of large numbers, and the fact that $x_i[j] \sim N(0,1)$ has a variance one, and for any $j \neq \ell \in [d]$

$$(\mathbf{X}^T\mathbf{X})_{j\ell} = \sum_{i=1}^{n} x_i[j]x_i[\ell] \simeq \sum_{i=1}^{n} \mathbb{E}[x_i[j]x_i[\ell]] = 0$$

we will substitute (for simplicity of the analysis)

$$(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}) = (n + \lambda)\mathbf{I}$$

- the resulting **predictor** is

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

Exercise: simple example

$$\hat{w} = \frac{n}{n+\lambda} w + \frac{n}{n+\lambda} X^T \varepsilon$$

- and the **expected predictor** is

$$\mathbb{E}[f_{\hat{w}_{\text{ridge}}}(x) \,|\, x] \;=\; \mathbb{E}[\hat{w}_{\text{ridge}}]^T x = \frac{n}{n+\lambda} w^T x$$

- we are ready to compute **the (conditional) bias**:

$$(\mathbb{E}[f_{\hat{w}_{\text{ridge}}}(x) \,|\, x] - f_0(x))^2 = \left((\frac{n}{n+\lambda} w^T - w^T) x \right)^2$$

$$= \frac{\lambda^2}{(n+\lambda)^2} (w^T x)^2$$

- the **expected bias** is:

$$\mathbb{E}_{x \sim p_x}[(\mathbb{E}[f_{\hat{w}_{\text{ridge}}}(x) \,|\, x] - f_0(x))^2] \;=\; \frac{\lambda^2}{(n+\lambda)^2} \mathbb{E}[w^T x x^T w]$$

$$= \frac{\lambda^2}{(n+\lambda)^2} w^T \mathbb{E}[x x^T] w$$

$$= \frac{\lambda^2}{(n+\lambda)^2} w^T \mathbf{I}_{d \times d} w$$

$$= \frac{\lambda^2 \|w\|_2^2}{(n+\lambda)^2}$$

where $\mathbb{E}[x x^T] = \mathbf{I}_{d \times d}$ follows from the fact that $x \sim N(0, \mathbf{I}_{d \times d})$

Exercise: simple example

- in a similar way, we can compute the **(conditional) variance**

$$\mathbb{E}\Big[\,(f_{\hat{w}_{\text{ridge}}}(x) - \mathbb{E}[f_{\hat{w}_{\text{ridge}}}(x)\,|\,x])^2\,|\,x\Big] = \mathbb{E}\left[\left((\hat{w}^T_{\text{ridge}} - \frac{n}{n+\lambda}w^T)x\right)^2\,|\,x\right]$$

$$= \mathbb{E}\left[(\frac{1}{n+\lambda}\varepsilon^T\mathbf{X}x)^2\,|\,x\right]$$

$$= \frac{1}{(n+\lambda)^2}x^T\mathbb{E}\left[\mathbf{X}^T\varepsilon\varepsilon^T\mathbf{X}\,|\,x\right]x$$

$$= \frac{\sigma^2}{(n+\lambda)^2}x^T(n\mathbf{I})x$$

$$= \frac{\sigma^2\, n}{(n+\lambda)^2}\|x\|_2^2$$

where we used the fact that $\hat{w}_{\text{ridge}} = \frac{n}{n+\lambda}w + \frac{1}{n+\lambda}\mathbf{X}^T\varepsilon,$

and $\mathbb{E}_{\mathbf{X},\varepsilon}[\mathbf{X}^T\varepsilon\varepsilon^T\mathbf{X}] = \mathbb{E}_{\mathbf{X}}[\mathbf{X}^T\mathbb{E}_{\varepsilon}[\varepsilon\varepsilon^T]\mathbf{X}] = \sigma^2\mathbb{E}_{\mathbf{X}}[\mathbf{X}^T\mathbf{I}\mathbf{X}] = \sigma^2\mathbb{E}_{\mathbf{X}}[\mathbf{X}^T\mathbf{X}] = \sigma^2 n\mathbf{I}$

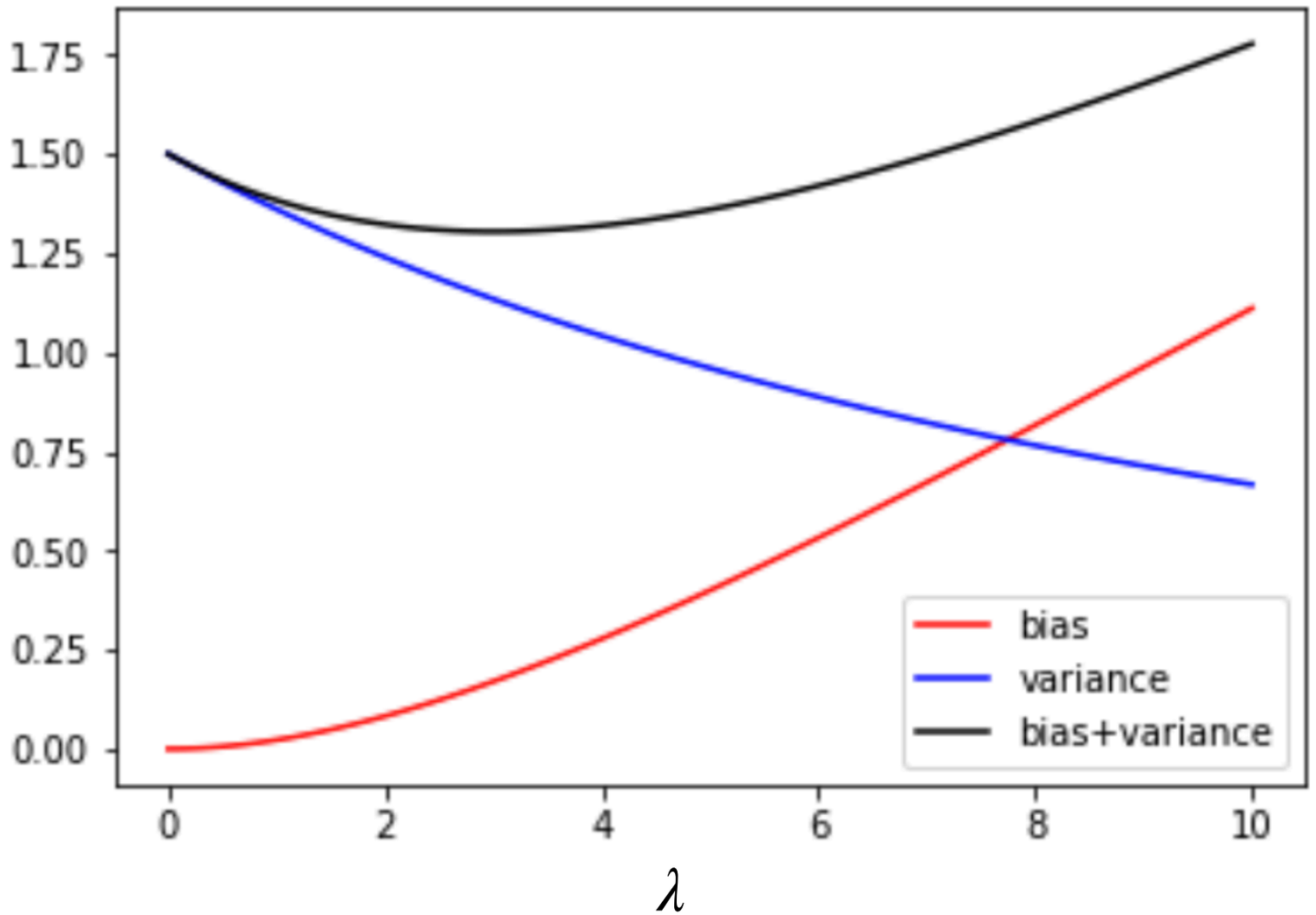
and $\mathbb{E}[\mathbf{X}^T\mathbf{X}] = n\mathbf{I}$ was computed 2 slides ago.

- taking expectation w.r.t. (with respect to) $x \sim N(0,\mathbf{I}_{d\times d})$, we get

$$\mathbb{E}\Big[\,(f_{\hat{w}_{\text{ridge}}}(x) - \mathbb{E}[f_{\hat{w}_{\text{ridge}}}(x)\,|\,x])^2\Big] = \frac{\sigma^2\, n\, d}{(n+\lambda)^2}$$

Bias-variance tradeoff w.r.t λ

- $\text{bias}^2 = \dfrac{\lambda^2 \|w\|_2^2}{(n+\lambda)^2}$
- $\text{variance} = \dfrac{\sigma^2 n d}{(n+\lambda)^2}$



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

Cross-validation:

how to choose regularization parameter λ, or the degree of polynomial features to use

Rule #1: Never use test set in training!

- but, does choosing λ based on test error count as using test data in training?
- first wrong approach:
  - train 10 predictors with 10 values of λ, each using all train data S_{train}
  - compute test error on test data S_{test} for all 10 models
  - pick λ^* that reported the smallest test error
  - deploy predictor f_{λ^*}

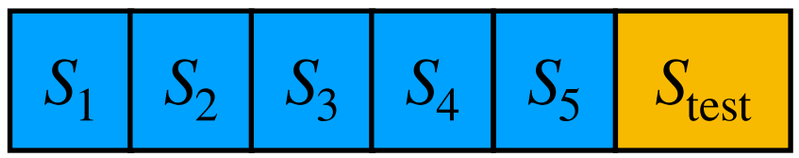
- why is it wrong?
  - because we used S_{test} in picking λ, we chose a model that works well on S_{test}
  - precisely, $\mathbb{E}_{\text{new data } (x,y)}[(f_{\lambda^*}(x) - y)^2] \neq \mathbb{E}_{S_{\text{test}}}\left[\frac{1}{|S_{\text{test}}|}\sum_{i \in S_{\text{test}}}\left\{(f_{\lambda^*}(x_i) - y_i)^2\right\}\right]$

    we sometimes use $\mathbb{E}_{S_{\text{test}}}[\,\cdot\,]$ interchangeably with $\mathbb{E}_{\text{test}}[\,\cdot\,]$ (e.g. in Assignment 1)
  - this commonly happens in machine learning competitions, and the competition organizers enforce several rules to prevent it
    - for example, each team can evaluate their test data performance only once per week

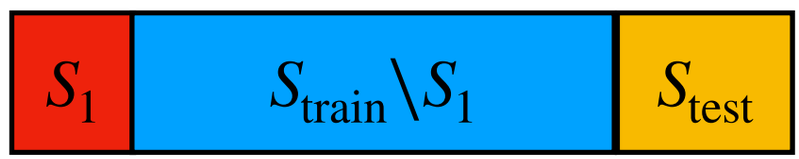
k-fold cross validation

- input
  - S_{train} and S_{test}

- procedure
  1. randomly divide the S_{train} into k equal sized partitions:
     $$\{S_1, \ldots, S_k\}$$
  2. define
     $$S_{\text{train}} \backslash S_j \triangleq \{i : i \in S_{\text{train}} \text{ and } i \notin S_j\}$$
     this operation $\backslash$ is called “set minus”, as it is taking a set away from another set

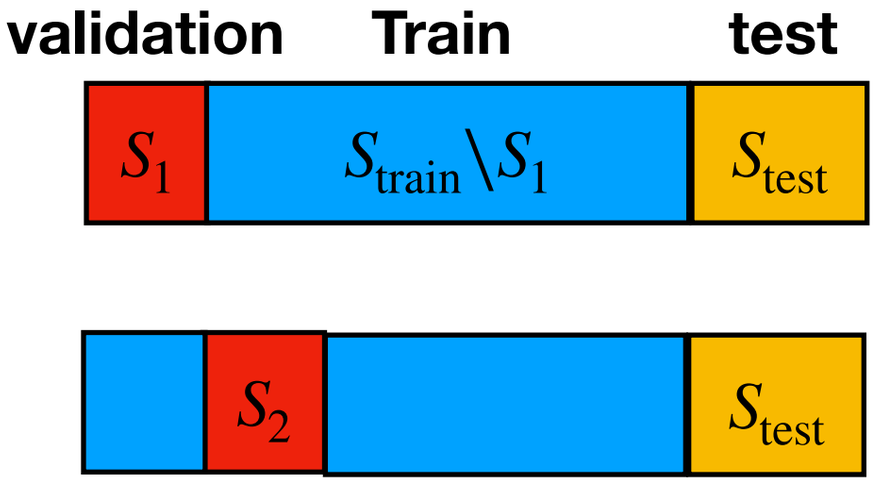


  3. train k predictors, such that the first predictor is trained on $S_{\text{train}} \backslash S_1$ and validated on S_1
     - $f_{S_{\text{train}} \backslash S_1}(x)$ minimizes $\sum_{i \in S_{\text{train}} \backslash S_1} (f_{S_{\text{train}} \backslash S_1}(x_i) - y_i)^2$
     - we keep track of **error on the validation set:**
       $$\text{error}_1 = \frac{1}{|S_1|} \sum_{i \in S_1} (f_{S_{\text{train}} \backslash S_1}(x_i) - y_i)^2$$
     - repeat for each partition $S_j, j \in \{1, \ldots, k\}$



k-fold cross validation

- First predictor $f_{S_{\text{train}} \setminus S_1}(\cdot)$ is trained on $S_{\text{train}} \setminus S_1$ and error_1 is evaluated on S_1
- j-th predictor $f_{S_{\text{train}} \setminus S_j}(\cdot)$ is trained on $S_{\text{train}} \setminus S_j$ and error_j is evaluated on S_j for all $j \in \{1, \ldots, k\}$



- finally, **k-fold cross validation error** is computed

$$\text{error}_{k-\text{fold}} = \frac{1}{k} \sum_{j=1}^{k} \text{error}_j$$

- k = 5 to 10 seems to work well in practice
- small k like two leads to **overestimating** the true error
  - because we are training on much smaller data size
- large k leads to many computations
- if k=N it is called Leave-one-out (LOO) cross validation

(LOO) leave-one-out cross validation

- slower but more accurate estimation of the error
- LOO cross validation is an extreme case of k-fold cross validation with k=n the total number of training samples

$$\text{error}_{\text{LOO}} = \frac{1}{n}\sum_{i=1}^{n}(f_{S_{\text{train}}\setminus\{i\}}(x_i) - y_i)^2$$

- we leave one data out and train a model, hence the name leave-one-out
- as each model is using $n-1$ training samples, this LOO validation error provides a close approximation of the true error of a model trained on all n training samples
- however, if n=100,000 (which is common size of modern dataset), it takes 100,000 times longer run-time to finish LOO cross-validation

example:

- Given 10,000-dimensional training data with n samples,
  - First, we pick 50 features that have highest correlation with the y_i's such that have largest

$$\text{pick 50 j's that have largest } \frac{\sum_{i=1}^{n} x_i[j] y_i}{\sqrt{\sum_{i=1}^{n} x_i[j]^2}}$$

  - We then use **k-fold cross validation** to train a ridge regressor on those 50 features, and choose λ using the cross validation error
  - What is wrong with this?
    For example, did we use any of the validation data S_1 in training, for example, a model $f_{S_{\text{train}} \setminus S_1}(x)$?