Regularization

Sewoong Oh

CSE/STAT 416
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 $r : \mathbf{R}^d \to \mathbf{R}$

- $r(w)$ 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

10001 -10000

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

Adding a regularizer to the loss

- we want small measure of fit

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

- we want small sensitivity $r(w)$

- these two objectives are traded off via regularized loss

$$\underset{w}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N}(w^T x_i - y_i)^2 + \lambda r(w)$$

coefficient

- $\lambda \geq 0$ is the **regularization parameter** (or **hyper parameter**)

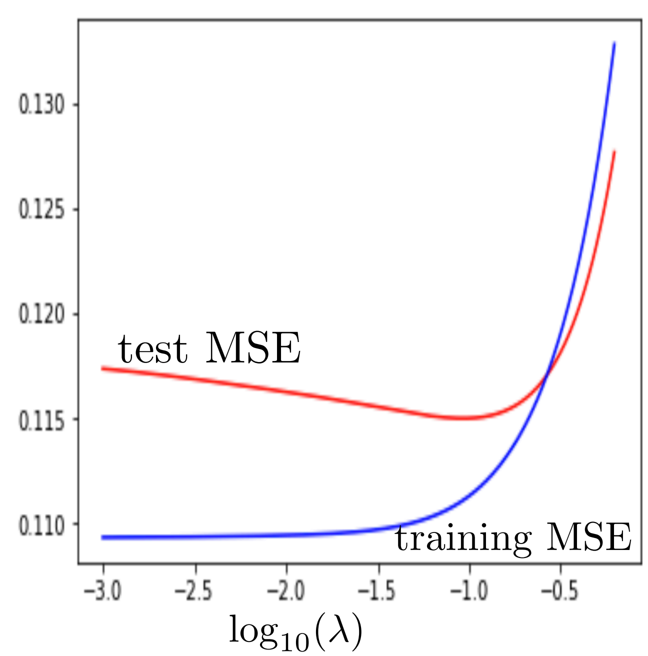
- solve the optimization problem for a choice of $r(w)$ to choose w that minimizes the regularized loss

$$\underset{w}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N}(w^T x_i - y_i)^2 \; + \; \lambda\, r(w)$$

$$= \sum_{j=1}^{d} w_j^2$$

- 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



Ridge regression

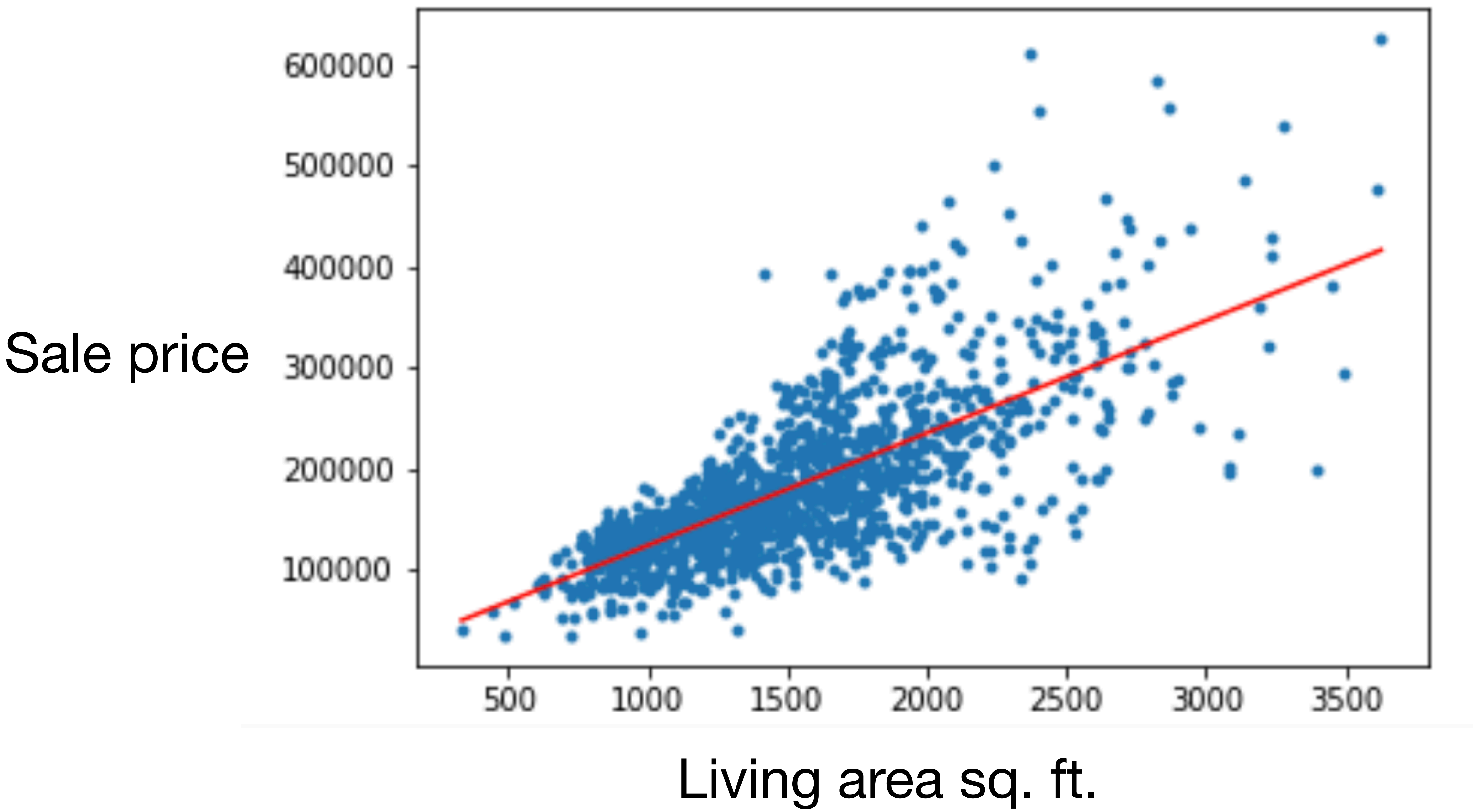
- **ridge regression**: quadratic loss and quadratic regularizer

- also called **Tykhonov regularized least squares**

$$\mathrm{MSE}(w) + \lambda\, r(w) \;=\; \underbrace{\frac{1}{N}\sum_{i=1}^{N}(w^T x_i - y_i)^2}_{\frac{1}{N}\|Xw-y\|^2} + \lambda \underbrace{\sum_{j=0}^{d} w_j^2}_{\|w\|^2}$$

- or $r(w) = \|w_{1:d}\|^2$ if $x_0 = 1$

Example: housing price (data from kaggle)

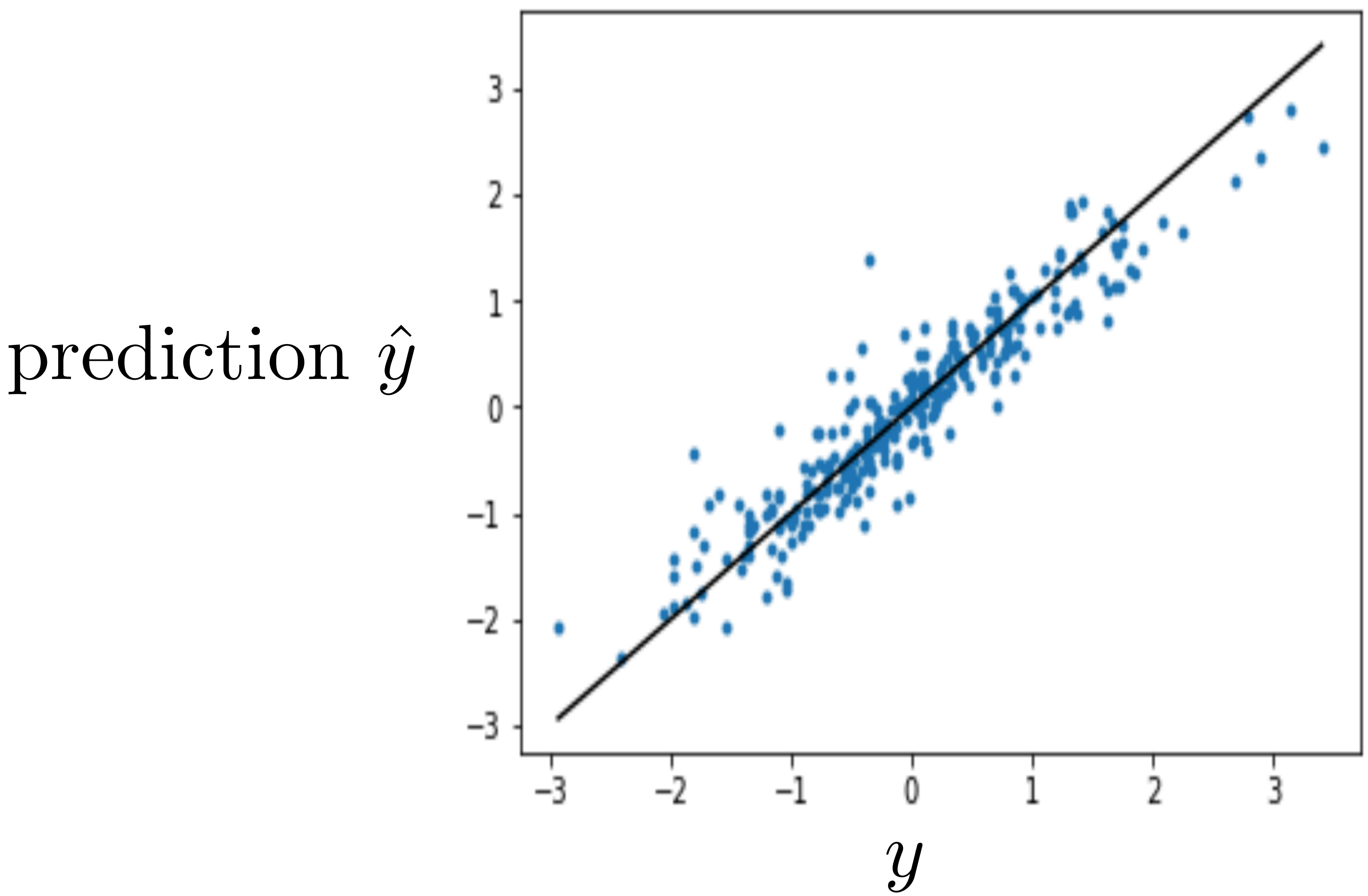


- sale prices of 1459 homes in Ames, Iowa from 2006 to 2010
- out of 80 features, we use 16
- we manually remove 4 outliers with are>4000 sq.ft.
  we will learn outlier detection later

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)

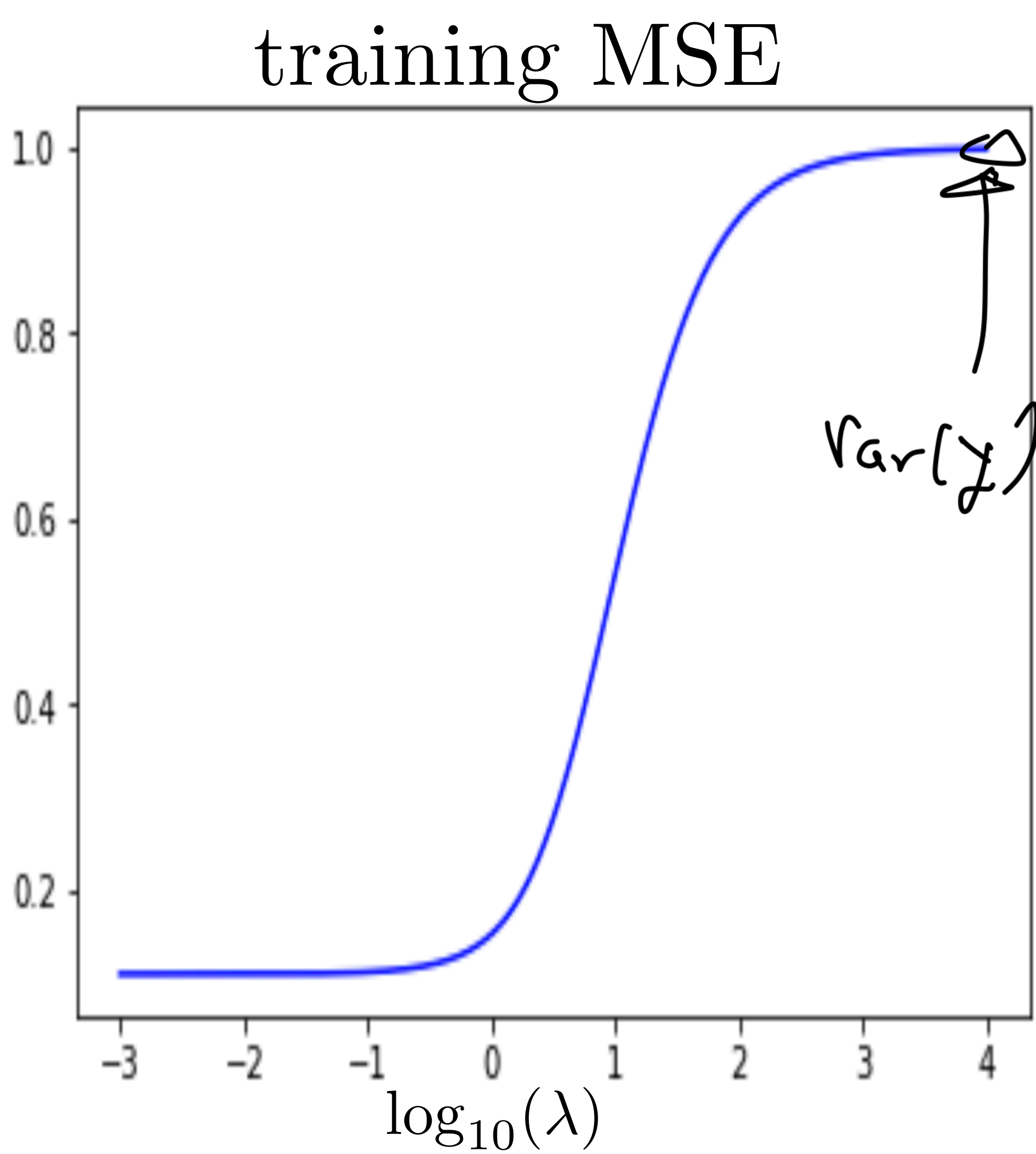
Example: regression (with no regularization)

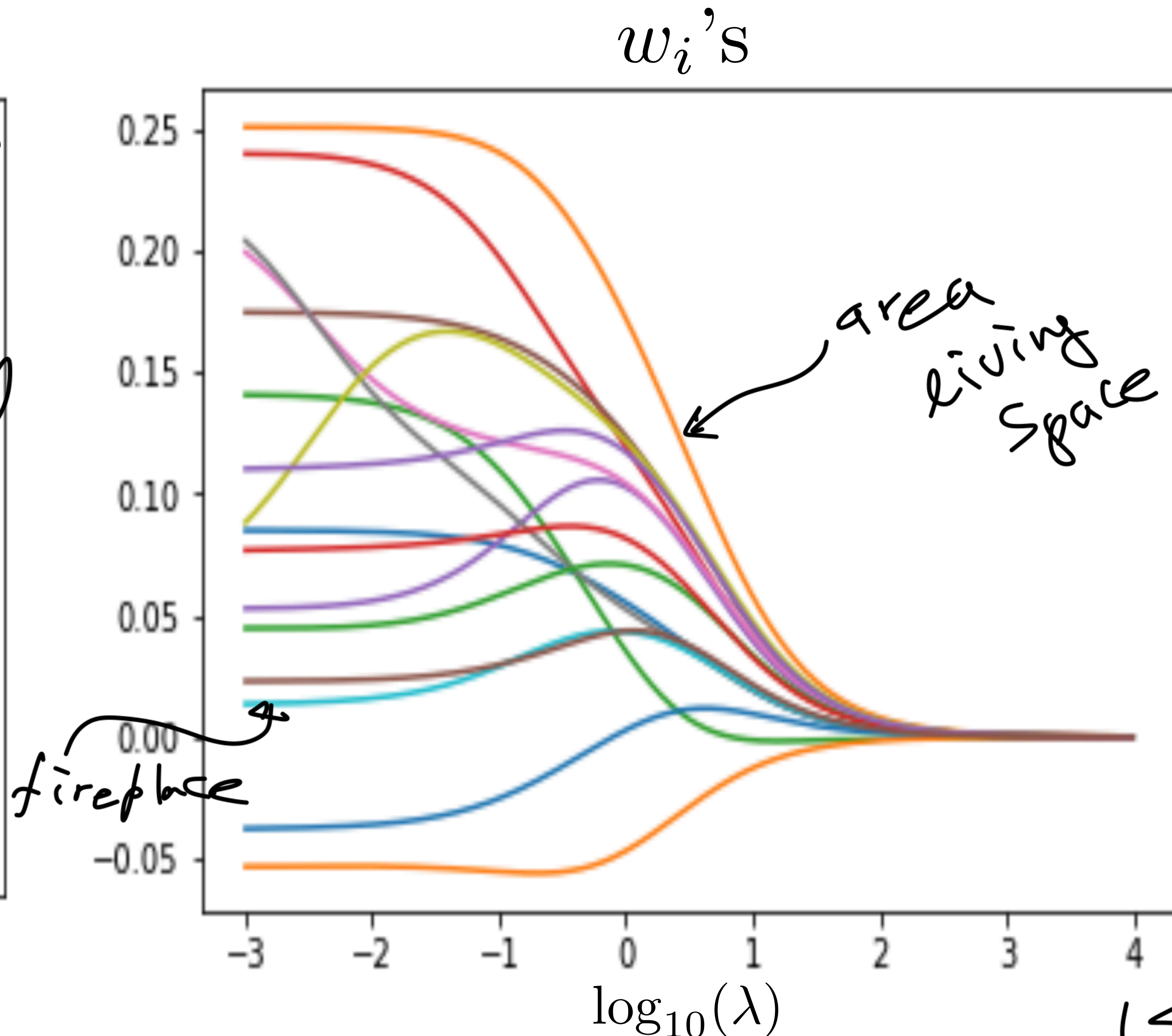


- split data randomly into 1164 training and 291 test
- target is log(price)
- standardize all features (and log(price))
- training error = 0.1093
- test error = 0.1175
- plot shows all 291 test points

Example: Ridge regression

$$\underset{w}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N}(w^T x_i - y_i)^2 \; + \; \lambda\, r(w)$$

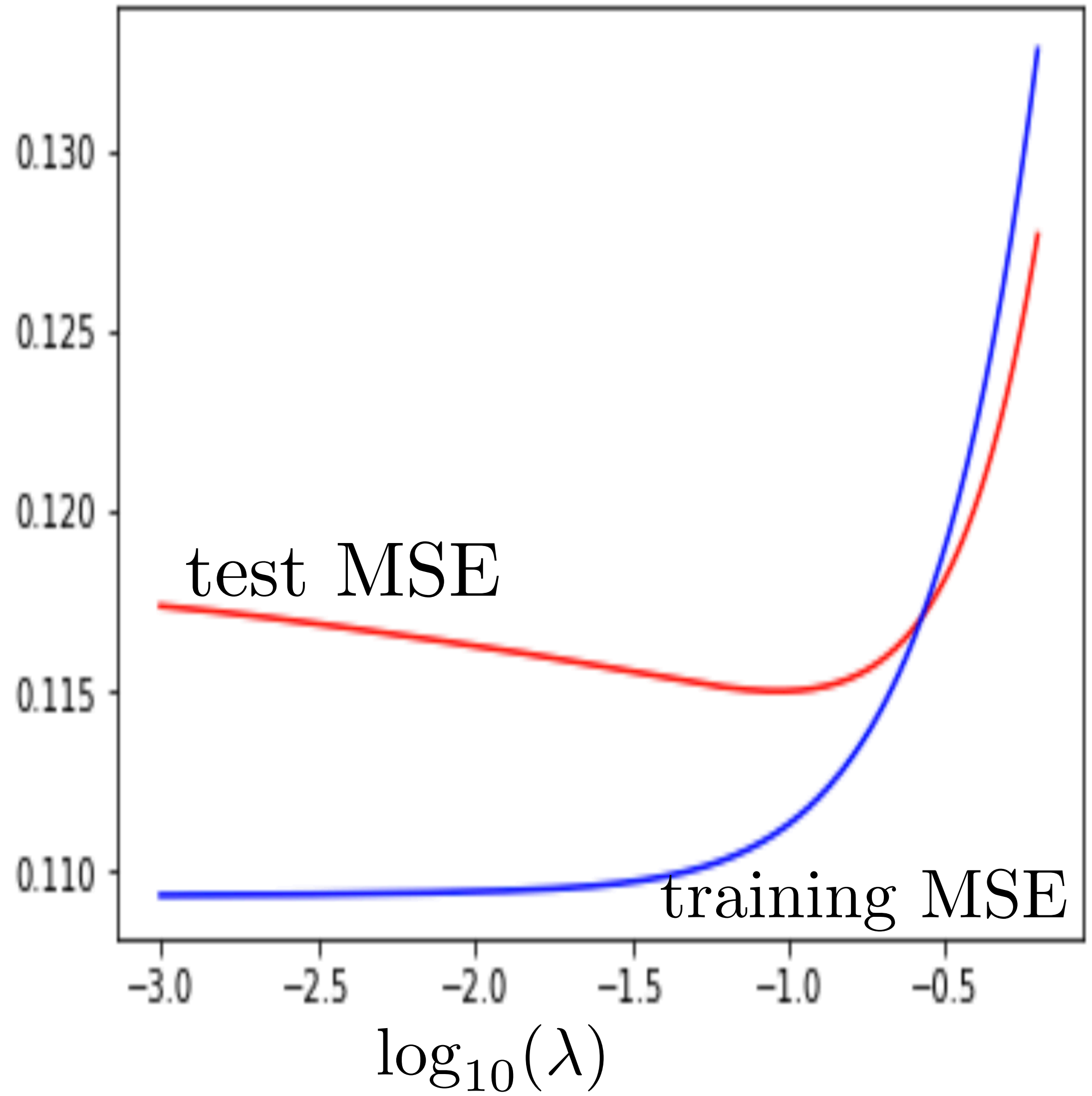


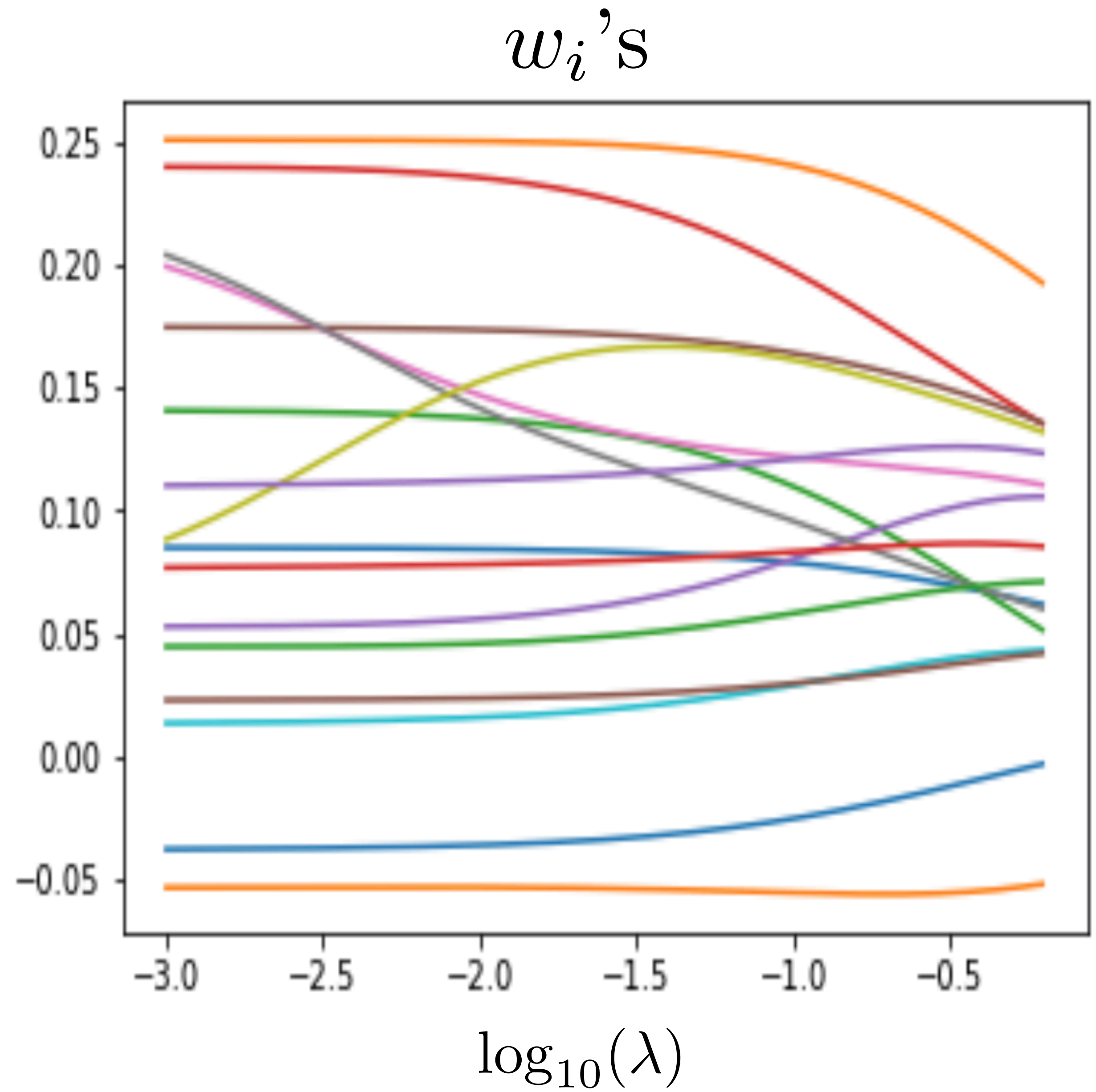


- leftmost training error is with no regularization: 0.1093
- rightmost training error is variance of the training data: 0.9991
- the right plot is called **regularization path**

Example: Ridge regression

$$\underset{w}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N}(w^T x_i - y_i)^2 \; + \; \lambda\, r(w)$$

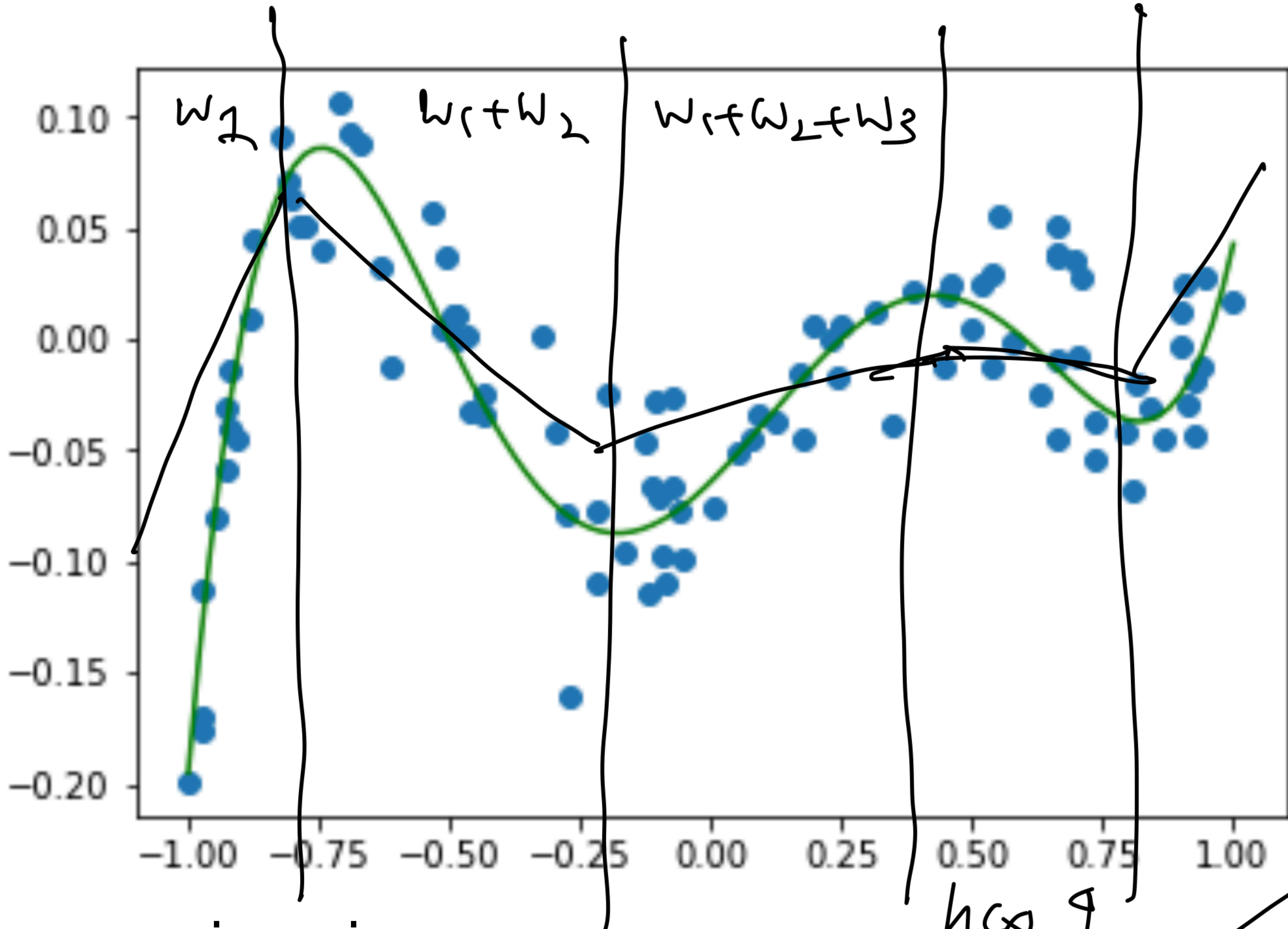




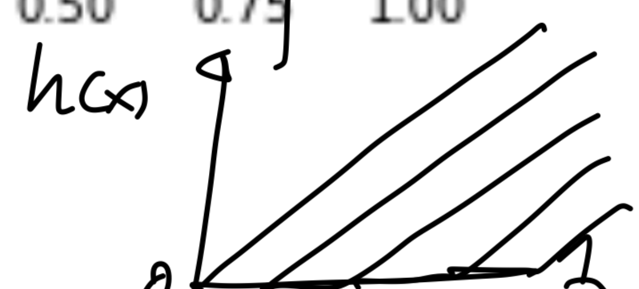
- optimal regularizer lambda= 0.1412
- slightly improves the test performance
- from test MSE = 0.1175 to tes MSE = 0.1147
- this gain comes from shrinking w's to get a less sensitive predictor

Example: piecewise linear fit

$f(x) = w_0 + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$



- features engineering:
  use piecewise linear functions

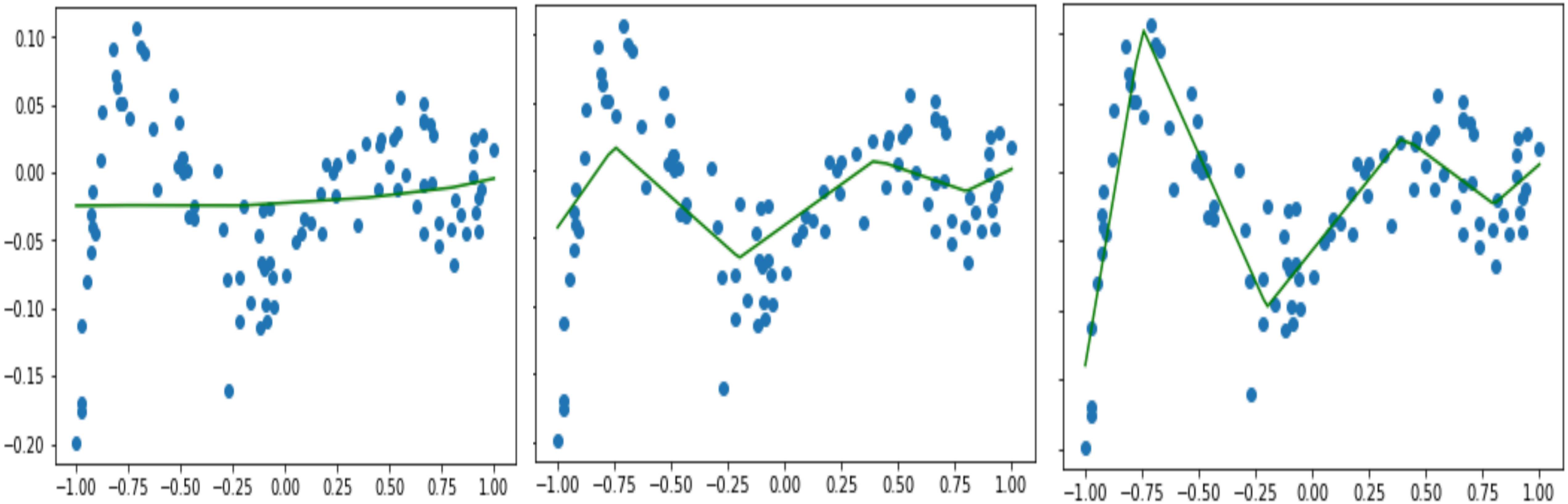


$$h(x) = (1, x, [x + 0.75]^+, [x + 0.2]^+, [x - 0.4]^+, [x - 0.8]^+)$$

$h_1(x)$ $h_2(x)$

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

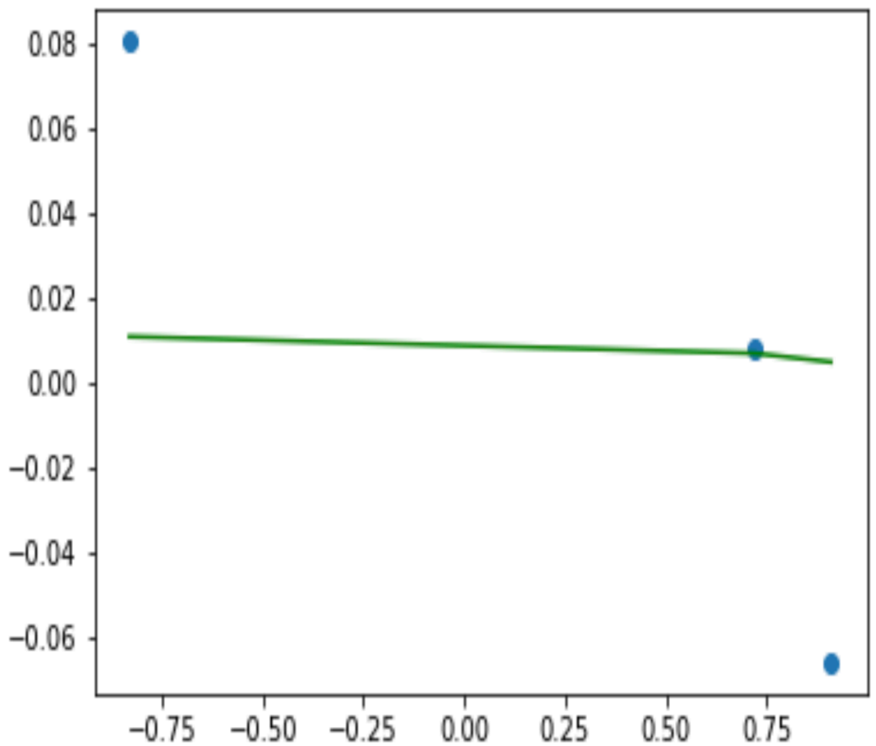
Example: piecewise linear fit



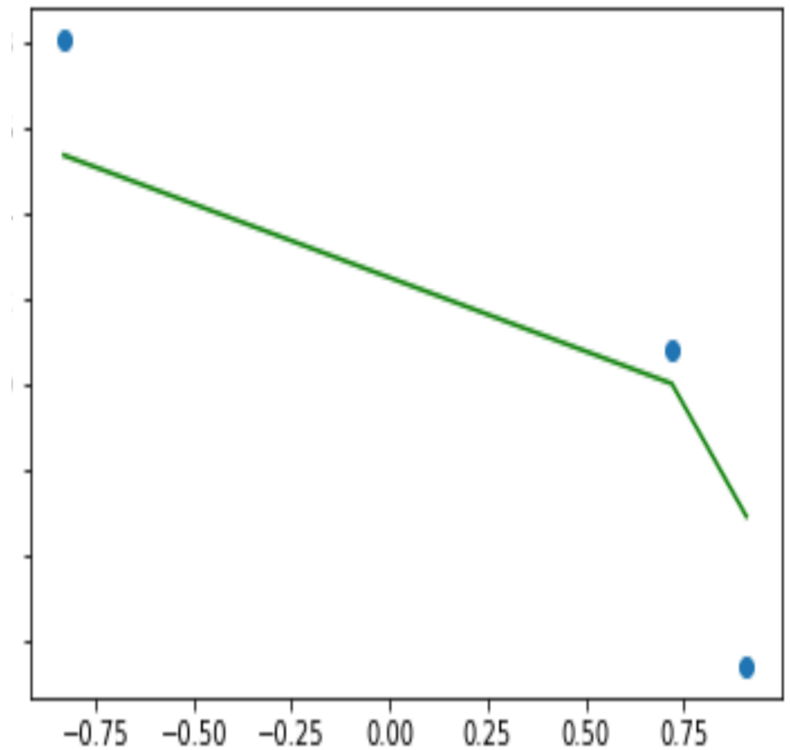
$\lambda = 1$ $\lambda = 0.005$ $\lambda = 0.000001$

- 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]

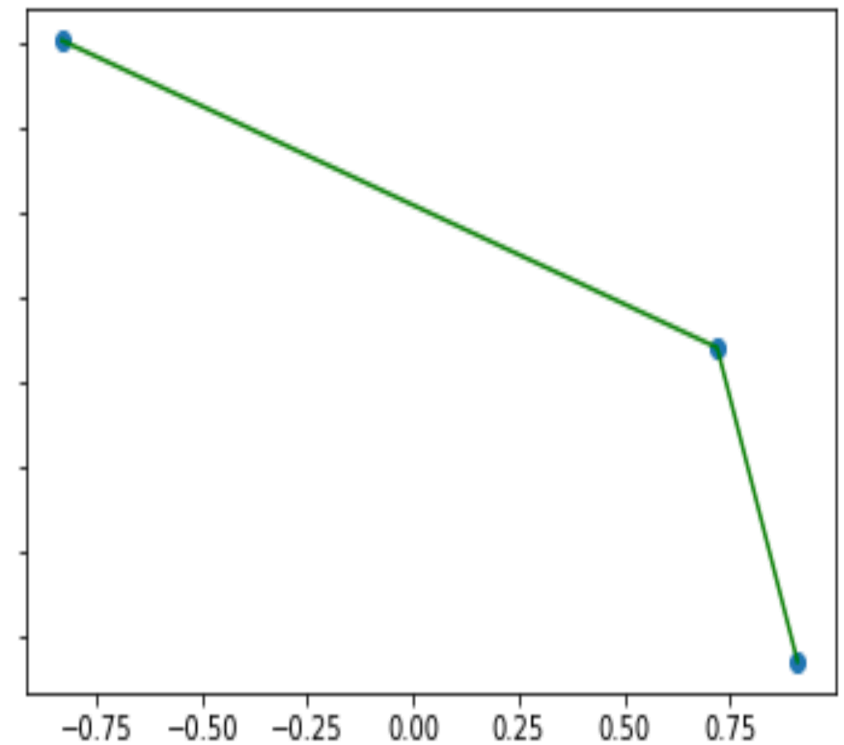
Fitting predictors with more parameters than data points



$\lambda = 100$



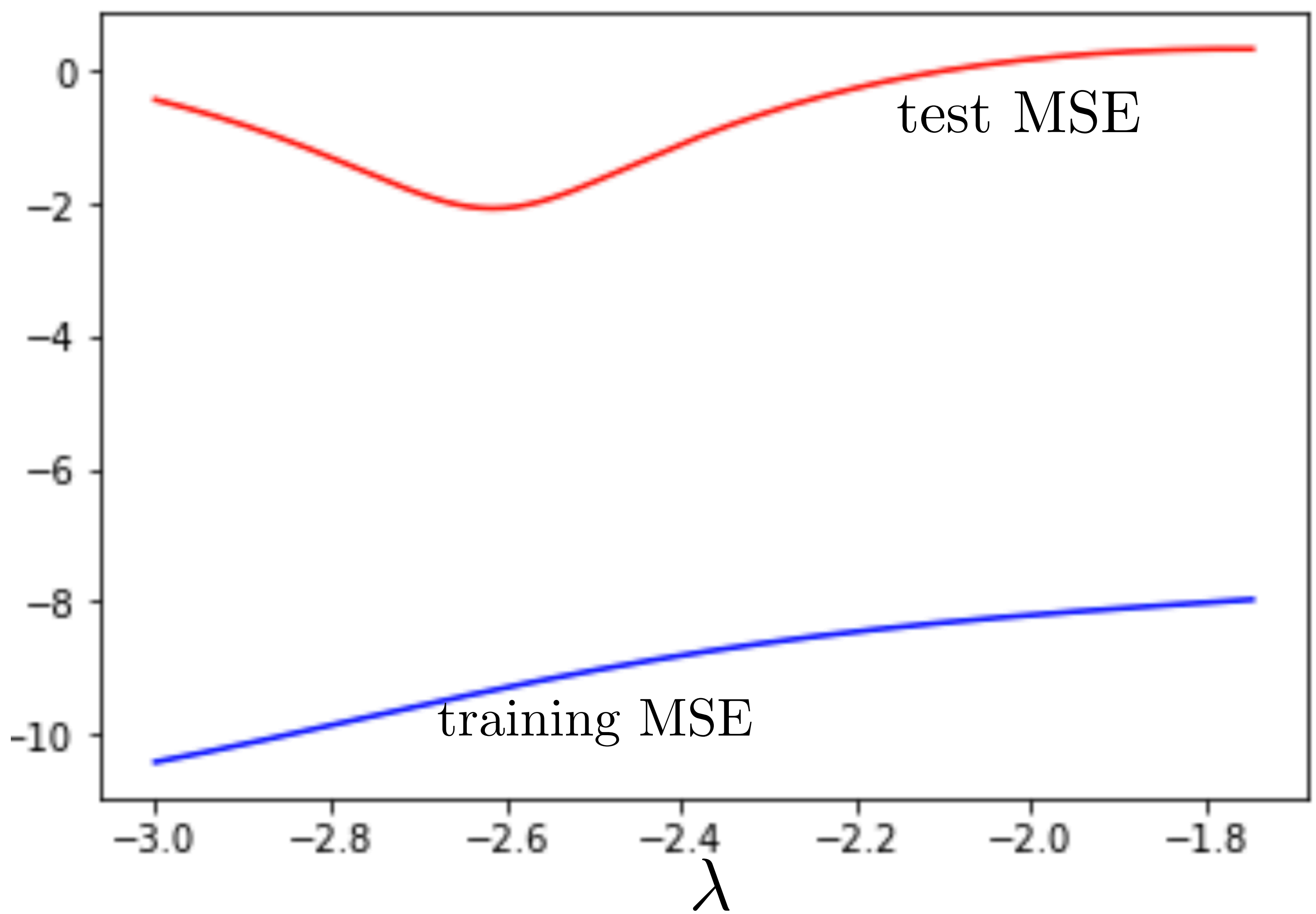
$\lambda = 3$



$\lambda = 0.0001$

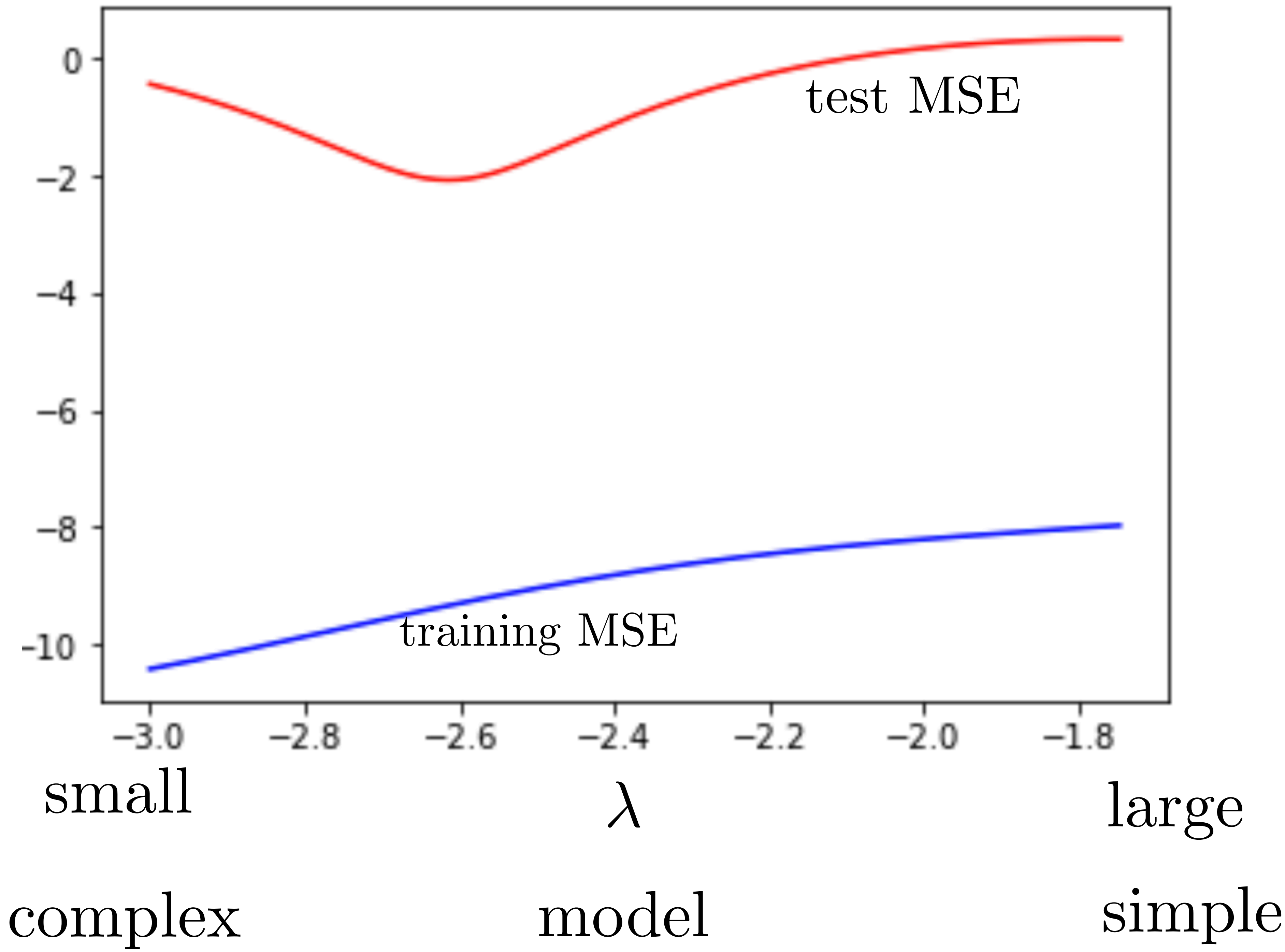
- in general, fitting a model with more parameters than data points does not make sense
- but one can fit such overparametrized models with regularization
- lambda=100 gives [0.01827, -0.00066429, -0.00069, -0.00109, -0.00268, -0.00962]
- lambda=0.0001 gives [0.471, -0.01027461, -0.0109, -0.0196, -0.0691, -0.4807]

Fitting predictors with more parameters than data points



- appropriate lambda balances fitting training data vs. sensitivity

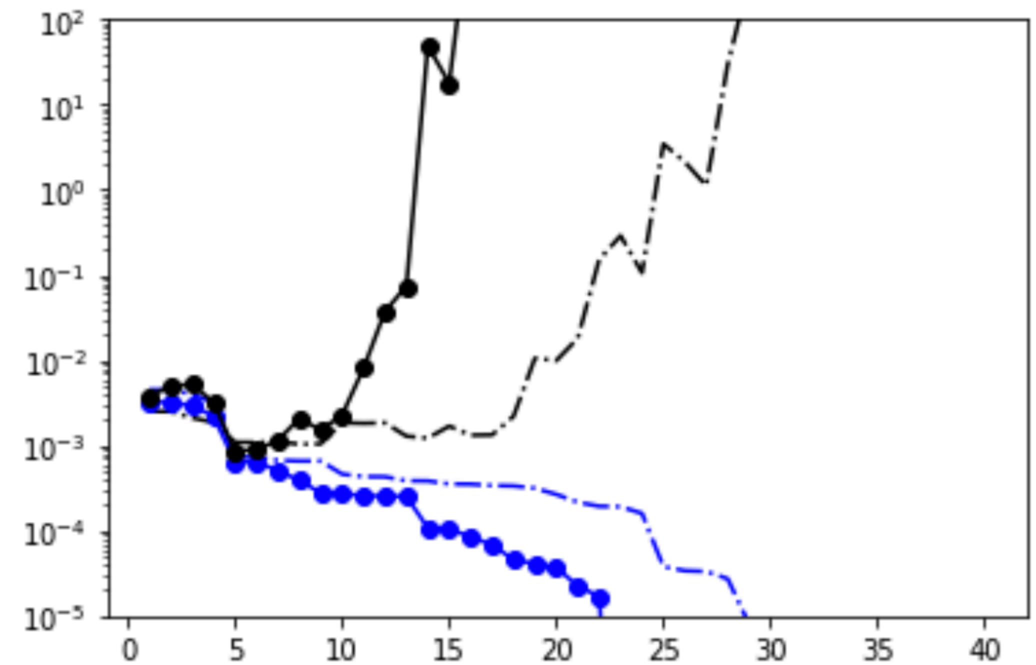
Model complexity and lambda



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

How to choose lambda with validation

- also called model selection
- Naive model selection can underestimate the test error



- 1. Train models for multiple lambda and compute test MSE
- 2. Pick lambda that minimizes the test error
- 3. Report that minimum test error

How to choose lambda with validation

- Split into train/test/validation sets

Training set	Validation set	Test set
80%	10%	10%
50%	25%	25%

- 1. Train models for multiple lambda by minimizing **training error**
- 2. Compute test error for each
- 3. Pick the lambda with smallest **test error**
- 4. compute **validation error** for that lambda

- Key idea is not to use the same data for “choosing lambda” and “evaluating error”