Introduction to Machine Learning

Lecture 10:

Regularized Least Squares

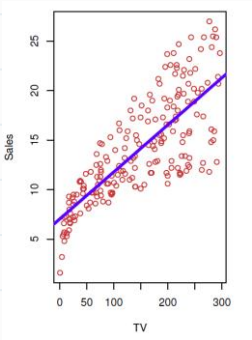
Prof. Subrahmanya Swamy Peruru
Department of Electrical Engineering
IIT Kanpur

Recap: Linear Regression

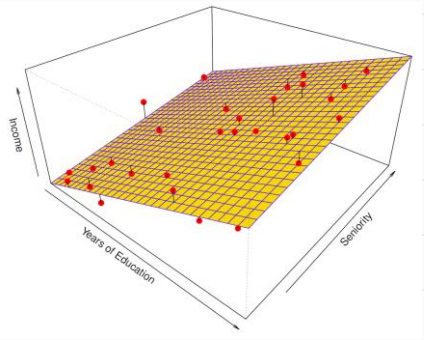
Single input: $f(\mathbf{x}) = w_1 x_1 + w_2$

Two inputs: $f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + w_3$

Line



Plane



Generally: $f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots w_{d-1} x_{d-1} + w_d$ **(Hyperplane)**

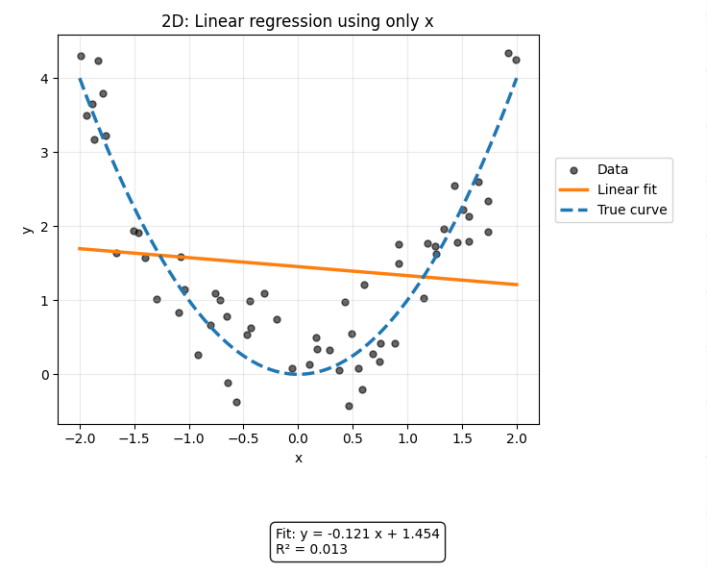
Task:

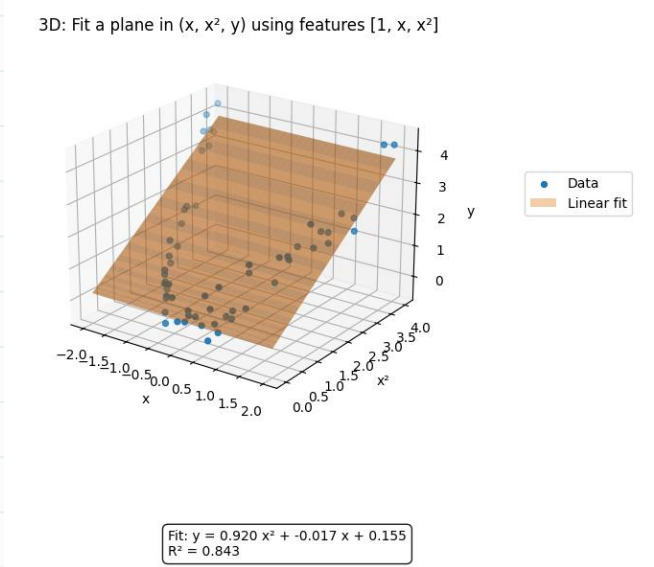
Given training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$

Find the unknown parameters $\mathbf{w} = (w_1, w_2, w_3, \ldots, w_d)$ that best fits the data

Figure Source: ISLP Book

What if the true relationship is not linear?





The model is still linear in parameters
Expanded features

but the function can now be highly nonlinear in input.

A linear model: $f(x) = w_1 x + w_2$

- Simple
- Easy to interpret
- Efficient to compute

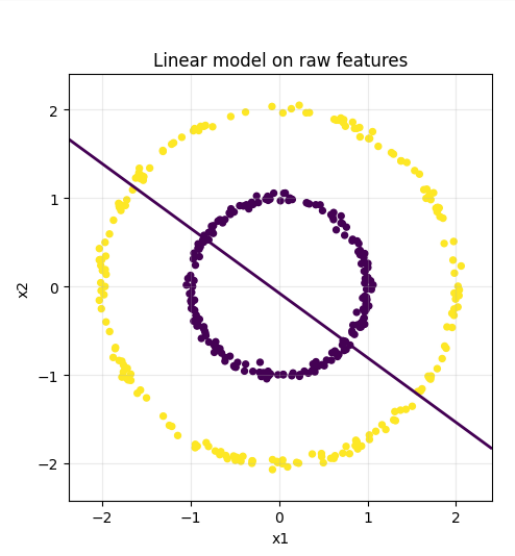
Feature expansion:

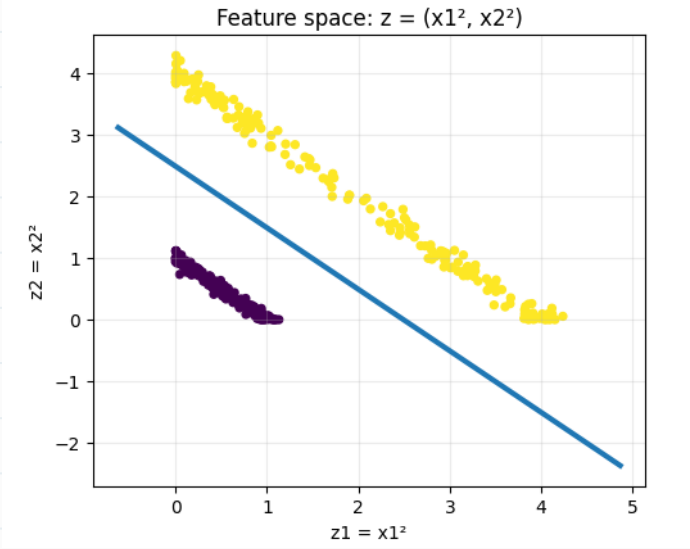
$$\boldsymbol{\phi}(x) = (x, x^2, 1)$$

Model:

$$f(\boldsymbol{\phi}(x)) = w_1 \boldsymbol{\phi}_1(x) + w_2 \boldsymbol{\phi}_2(x) + w_3$$

A Classification example





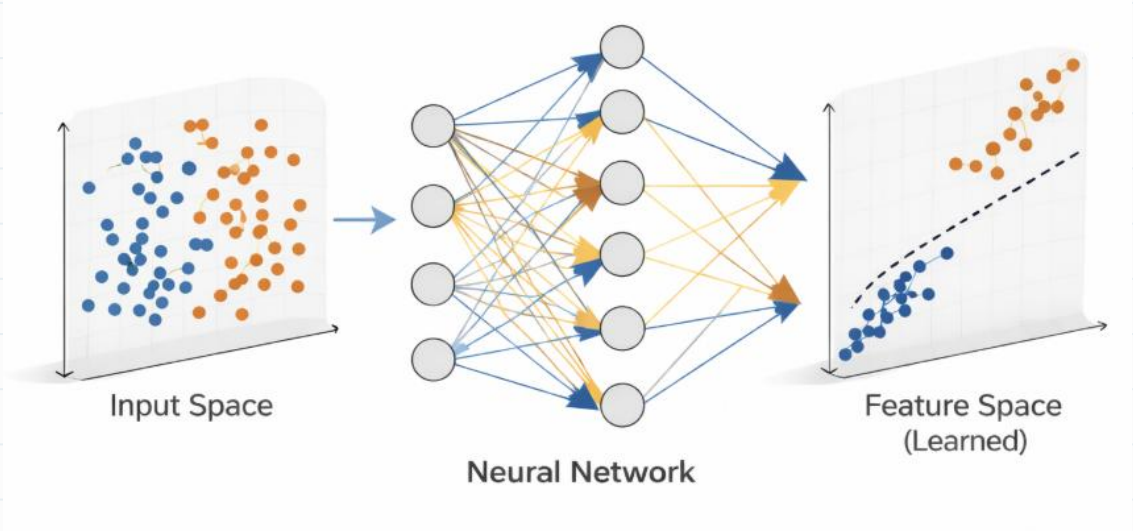
How do we know the right feature representation for our data?

A linear model is unable to classify the points using the inputs (x_1, x_2)

While it works on the new feature representation $(z_1, z_2) = (x_1^2, x_2^2)$

Neural networks: Learn the representation from data

Key Idea: **Learn a feature representation in which the target is approximately linear**



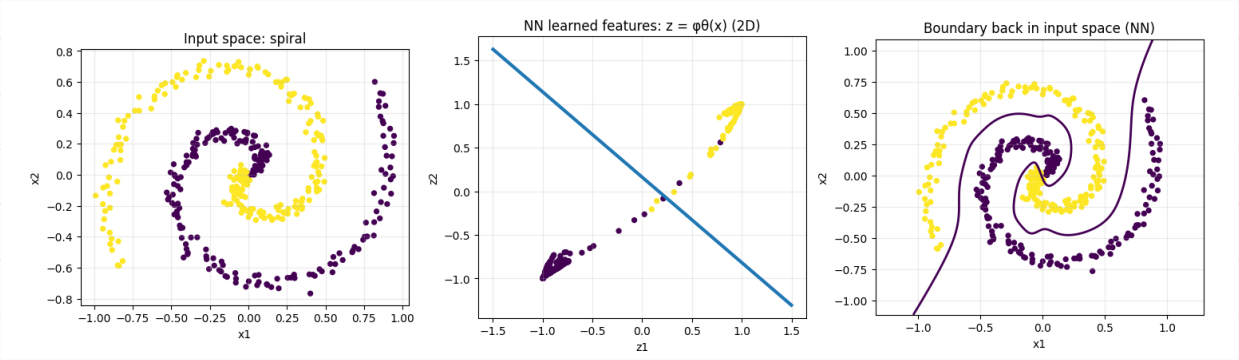
A neural network can be viewed as:

$$\boldsymbol{\phi}_\theta(\mathbf{x}) \quad \text{(learned feature map)}$$

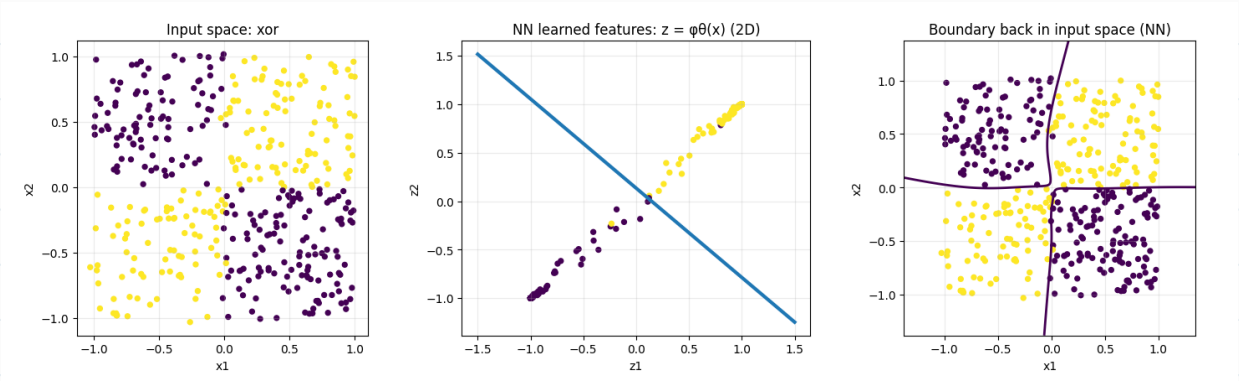
Final prediction:

$$f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}_\theta(\mathbf{x})$$

Learned feature representations change with data



Scan this QR for a demo



Original inputs were not linearly separable

Learned features became linearly separable

From the data, neural networks automatically learn appropriate feature representations

Many rich features ⇒ Even linear models can be highly flexible

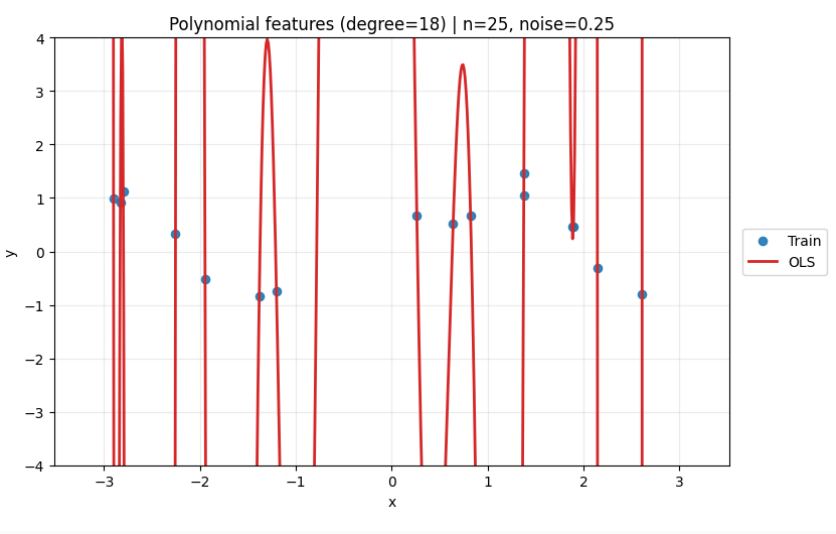
Even linear models can be highly flexible,
depending on the feature representation
and the amount of training data.

Too much flexibility leads to overfitting!

Example:

$$\boldsymbol{\phi}(x) = (x, x^2, \cdots, x^{18}, 1)$$

$$f(x) = \sum_i w_i \boldsymbol{\phi}_i(x)$$



How do we control this flexibility in a principled way?

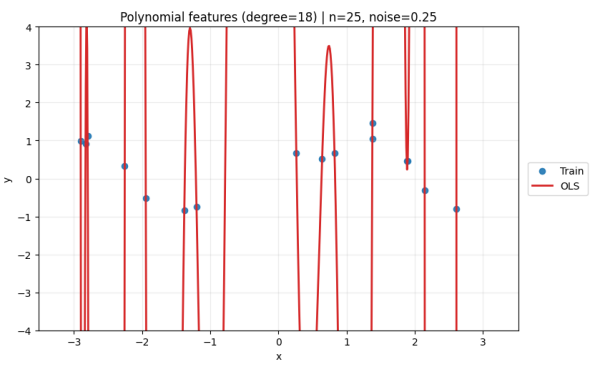
This naturally leads to the idea of **regularization**.

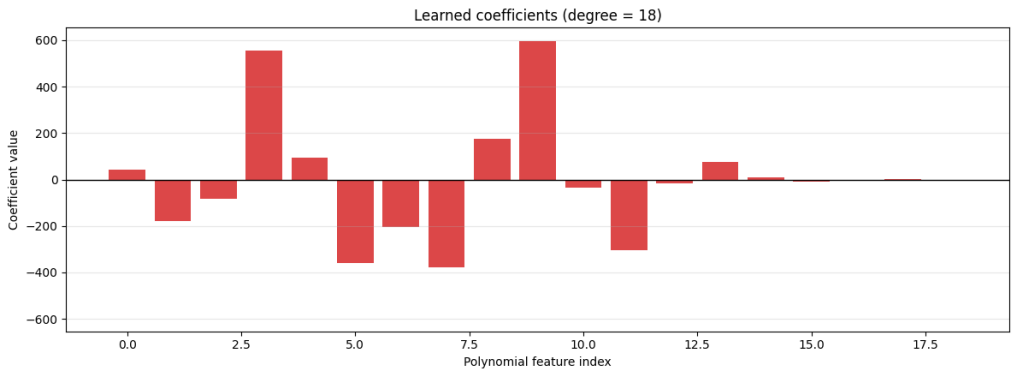
Regularization in Linear Regression

Controlling Overfitting via Norm Constraints

Regularization

Overfitting often corresponds to solutions with large weights.





The weight vector has a large norm $\|\mathbf{w}\|_2 \approx 1074$

Large weights ⇒ large sensitivity ⇒ overfitting
small changes in input lead to large changes in output

Regularization: Add a penalty to discourage unnecessarily large weights

L2 Regularization (Ridge Regression)

Ordinary least squares (OLS):

$$\min_{\mathbf{w}} \ \| \mathbf{y} - \mathbf{X}\mathbf{w} \|_2^2$$

L2 Regularization (Ridge Regression):

$$\min_{\mathbf{w}} \ \| \mathbf{y} - \mathbf{X}\mathbf{w} \|_2^2 \ + \lambda \| \mathbf{w} \|_2^2$$

Penalizes unnecessarily large weights
Suppresses overly sensitive solutions

$\lambda > 0$ is a **hyperparameter** that controls strength of regularization

Large λ: stronger regularization → smoother models
Small λ: weaker regularization → lower training error

Regularization introduces a trade-off
between training error and model complexity.

Closed-Form Solution for Ridge Regression

Convex and differentiable loss function.
Gradient zero point gives the global minimum.

$$\min_{\mathbf{w}} \ \| \mathbf{y} - \mathbf{X}\mathbf{w} \|_2^2 \ + \lambda \| \mathbf{w} \|_2^2$$

Step 1: Expand the objective

$$(\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \, \mathbf{w}^\top \mathbf{w}$$

Step 2: Take gradient w.r.t. $\mathbf{w}$

$$\nabla_{\mathbf{w}} = -2\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) + 2\lambda \mathbf{w}$$

Step 3: Set gradient to zero

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} + \lambda \mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

Step 4: Solve for $\mathbf{w}$

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

$\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}$ *is always invertible even when* $X^\top X$ *is not invertible.*
Additional $\lambda \mathbf{I}$ improves numerical stability by shifting all eigenvalues away from zero

LASSO (L1 Regularization)

LASSO (Least **Absolute** Shrinkage and **Selection** Operator) objective:

$$\min_{\mathbf{w}} \ \| \mathbf{y} - \mathbf{X}\mathbf{w} \|_2^2 \ + \lambda \| \mathbf{w} \|_1$$

where:

$$\| \mathbf{w} \|_1 = \sum_j | w_j |$$

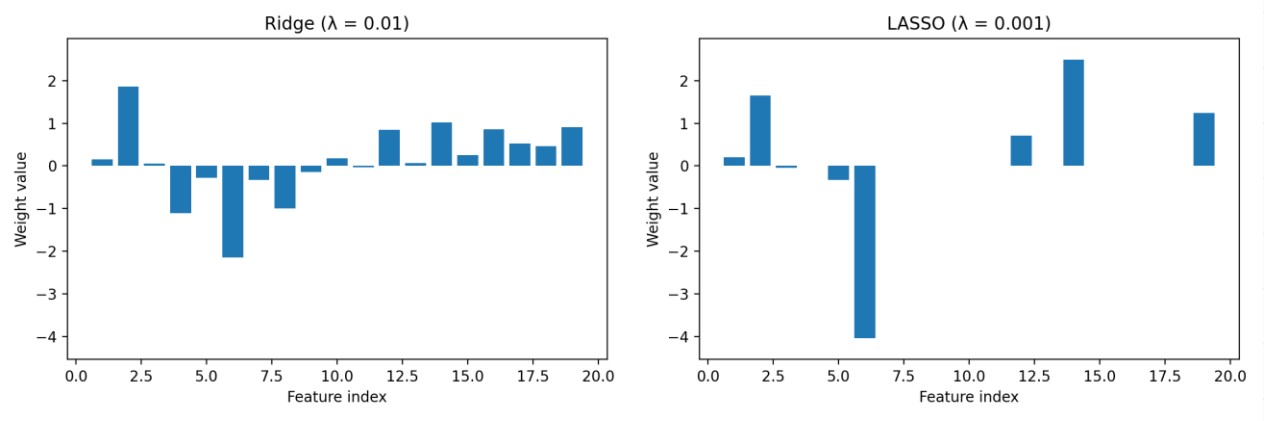
No closed-form solution:
Objective is convex but non-differentiable
Solved using iterative optimization methods

Encourages sparse solutions:
Many coefficients become exactly zero
Performs feature selection

Ridge vs LASSO

$$\boldsymbol{\phi}(x) = (x, x^2, \cdots, x^{18}, 1)$$

$$f(x) = \sum_i w_i \boldsymbol{\phi_i}(x)$$



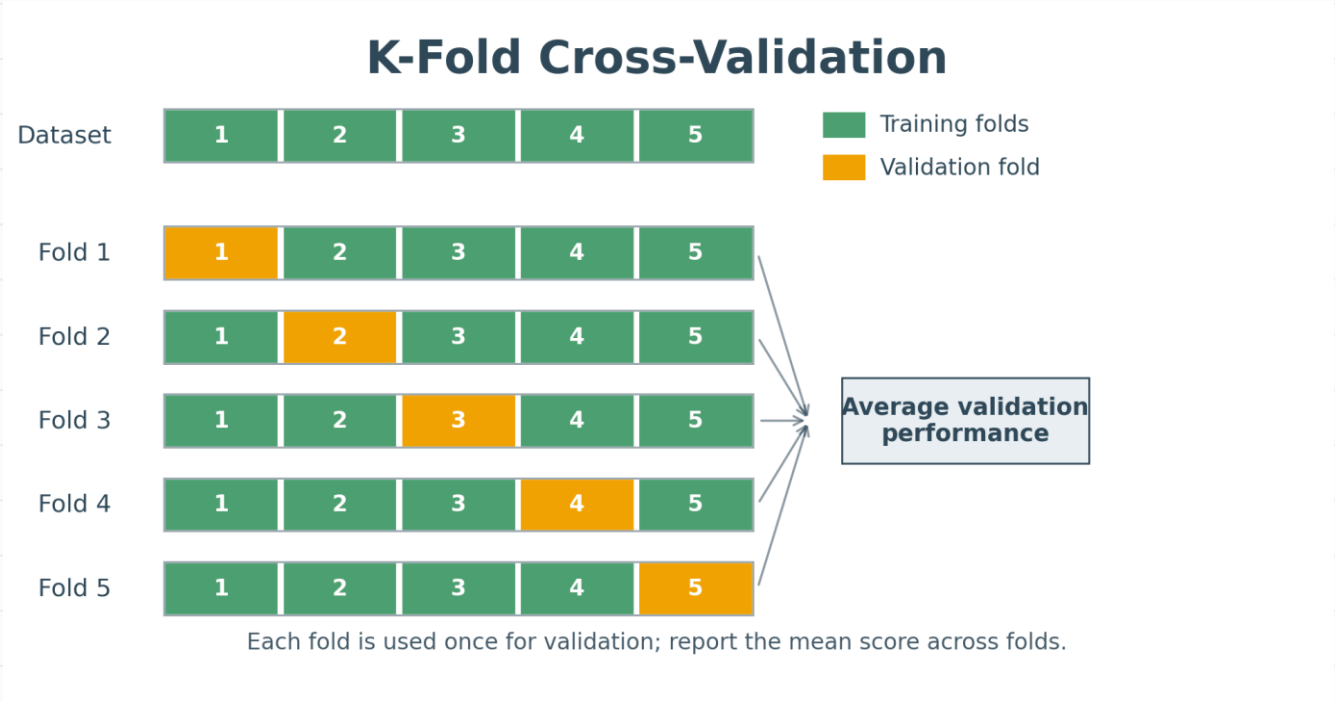
Demo

Ridge: all weights are shrunk toward zero (but typically remain non-zero)

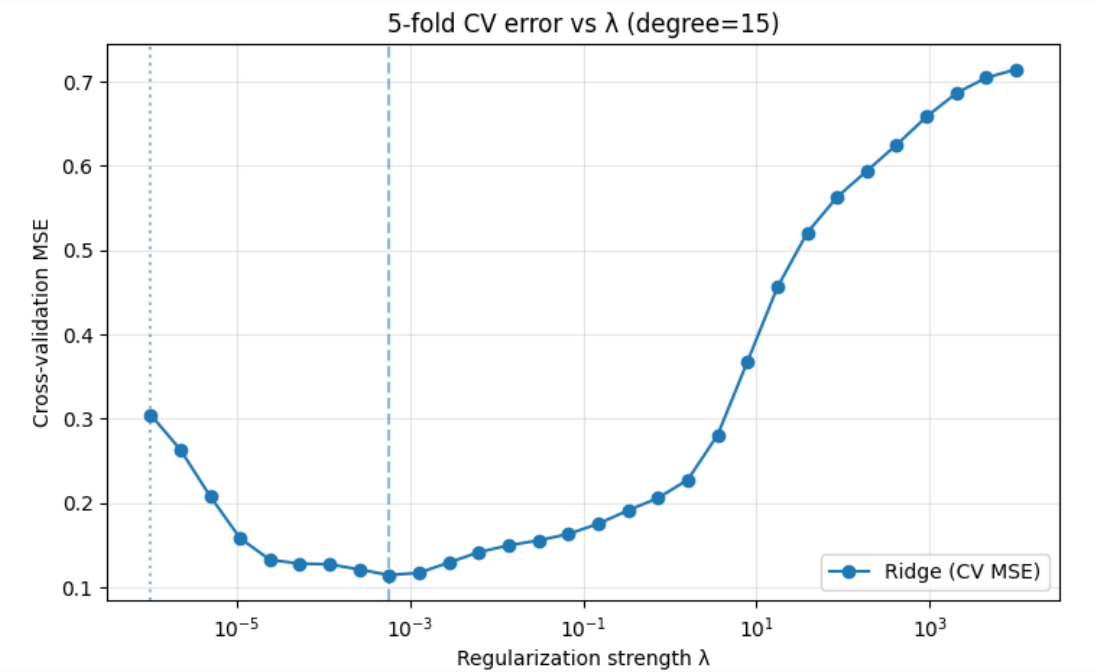
LASSO: some weights are driven exactly to zero (feature selection)

Regularization choice shapes which functions are preferred.

What Regularization parameter to use?



What Regularization parameter to use?



for each choice of λ:
  for each fold i in $range(1, k)$:
    compute validation $MSE_i(\lambda)$
  compute average MSE across folds

$$\mathrm{CV}(\lambda) = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i(\lambda)$$

Small λ: low bias, high variance
Large λ: high bias, low variance

λ is typically selected by cross-validation over a logarithmically spaced grid (e.g., powers of 10)**.**

Thank You