ITCS 6156/8156 Fall 2024
Machine Learning

Linear Regression

Instructor: Hongfei Xue
Email: hongfei.xue@charlotte.edu
Class Meeting: Tue & Thu, 4:00 PM – 5:15 PM, WWH 130

UNIVERSITY OF NORTH CAROLINA CHARLOTTE

Some content in the slides is based on Dr. Razvan's lecture

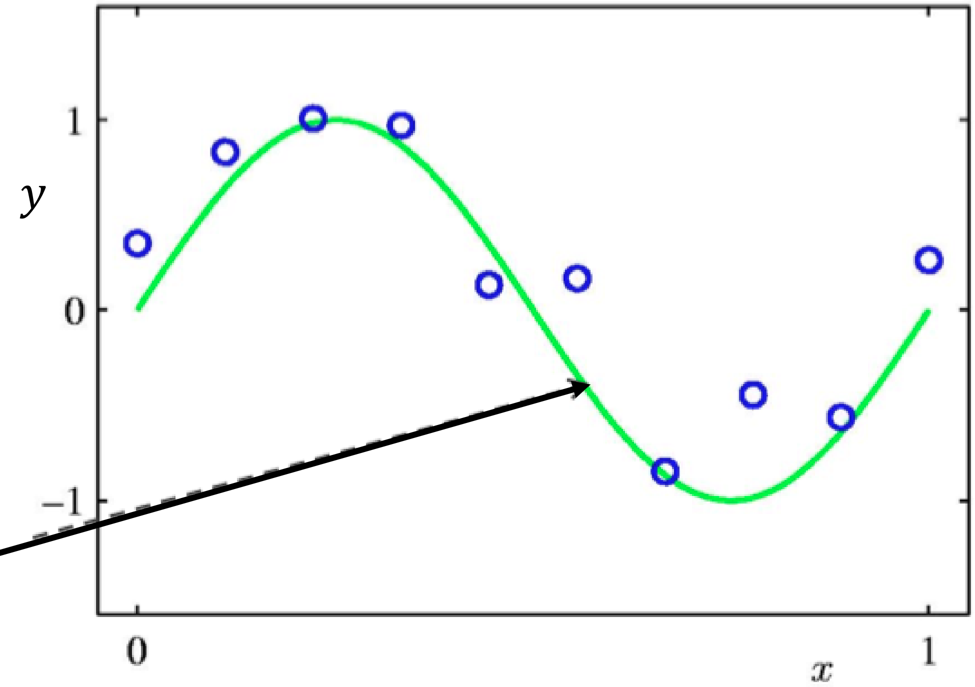
Polynomial Basis Functions

- Q: What if the raw feature is insufficient for good performance?
  - Example: non-linear dependency between label and raw feature.

- A: Engineer / Learning higher-level features, as functions of the raw feature.

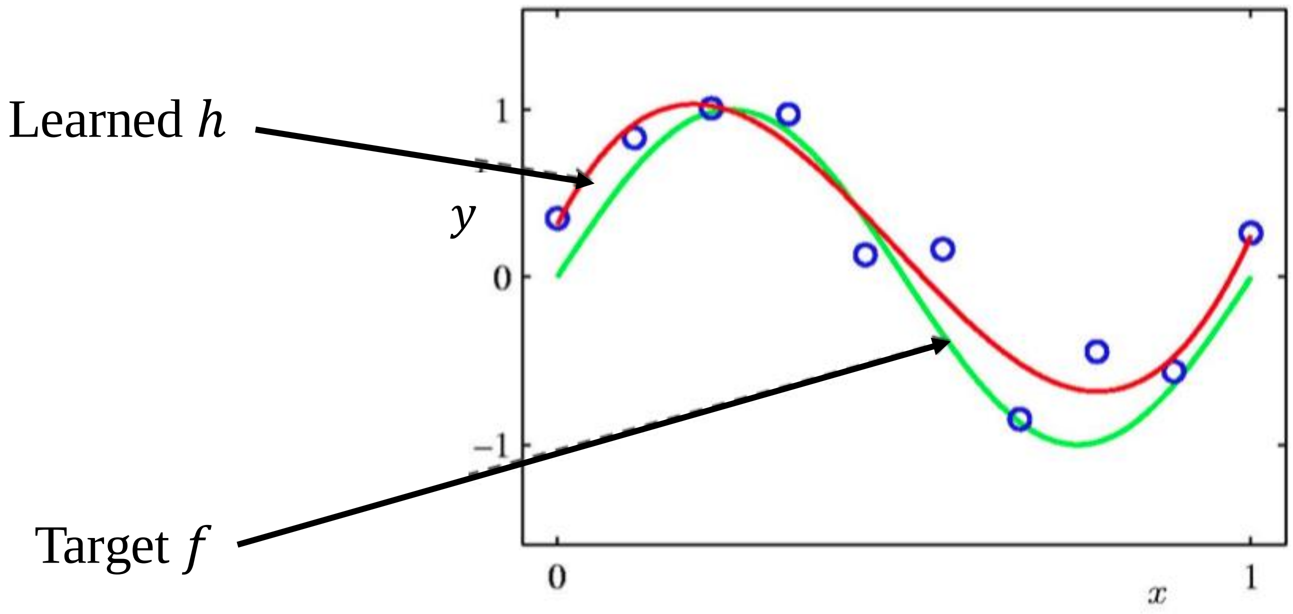
- Polynomial curve fitting:
  - Add new features, as polynomials of the original feature.

Regression: Curve Fitting



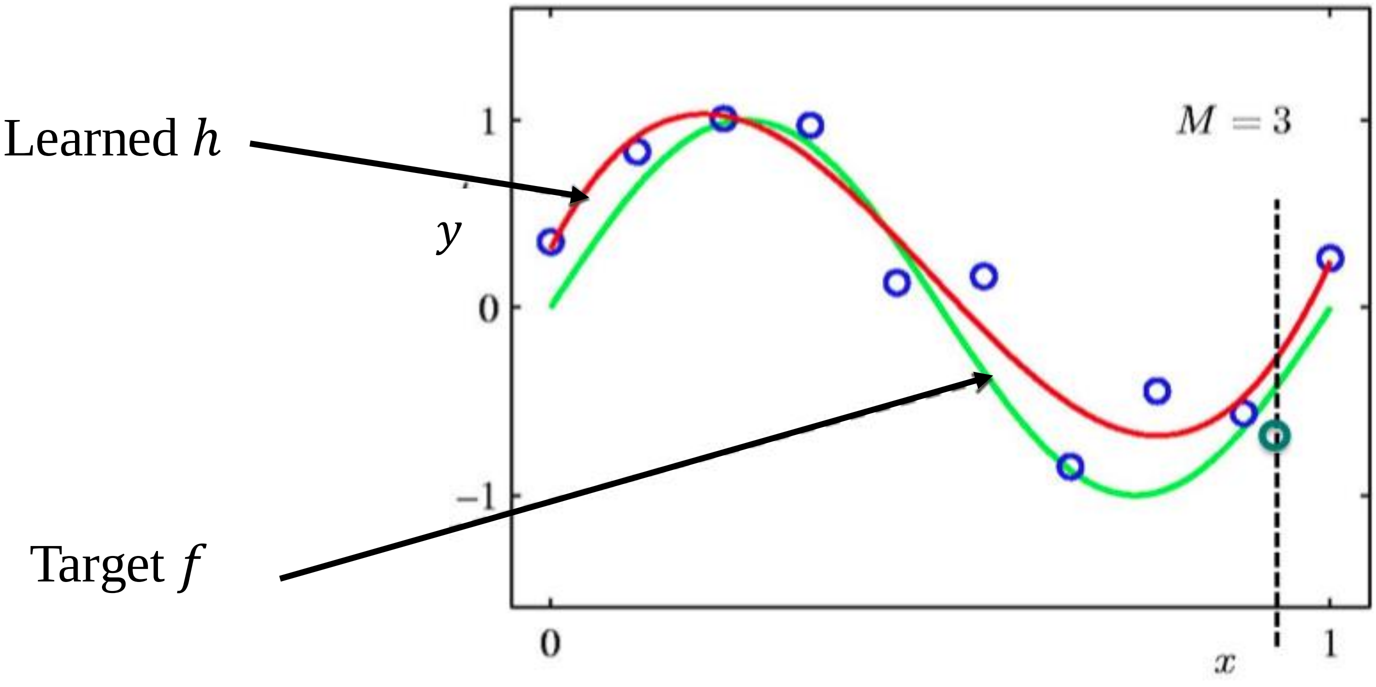
Target f

Regression: Curve Fitting



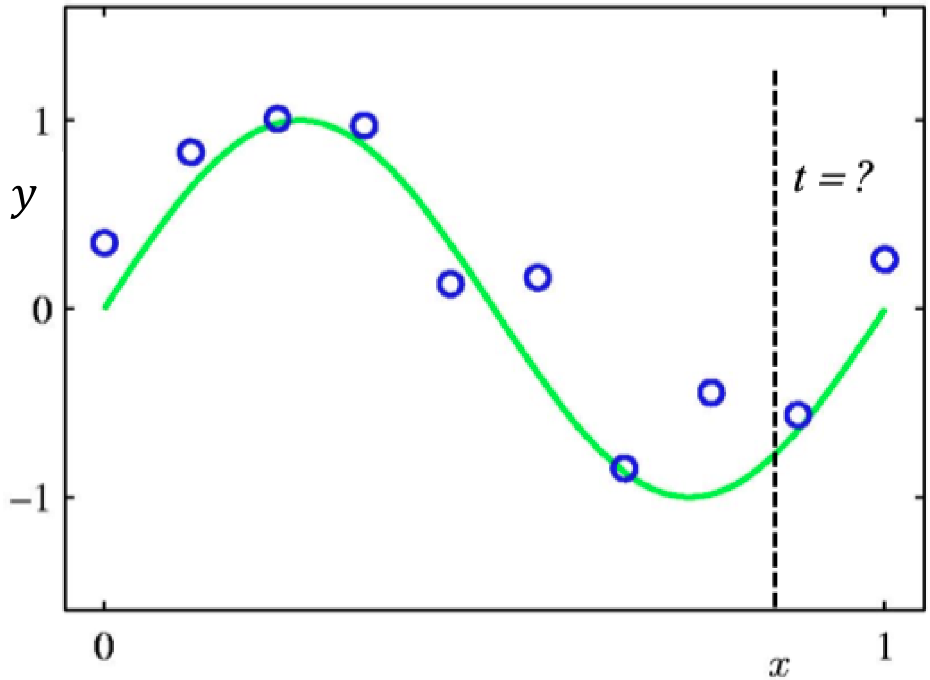
- Training: Build a function $h(x)$, based on (noisy) training examples $(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)$.

Regression: Curve Fitting



- Testing: for arbitrary (unseen) instance $x \epsilon$ **X**, compute target output $h(x)$; want it to be close to f(x).

Regression: Polynomial Curve Fitting



$$h(x) = h(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

parameters

features

Polynomial Curve Fitting

- Parametric model:

$$h(x) = h(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

- Polynomial curve fitting is (Multiple) Linear Regression:

$$\mathbf{x} = [1, x, x^2, \cdots, x^M]^T$$
$$h(x) = h(\mathbf{x}, \mathbf{w}) = h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^{\boldsymbol{T}}\mathbf{x}$$

- **Learning =** minimize the Sum-of-Squares error function:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\text{argmin}} J(\mathbf{w}) \qquad J(\mathbf{w}) = \frac{1}{2N} \sum_{n=1}^{N} (h_{\mathbf{w}}(\mathbf{x}_n) - y_n)^2$$

- Least Square Estimate:

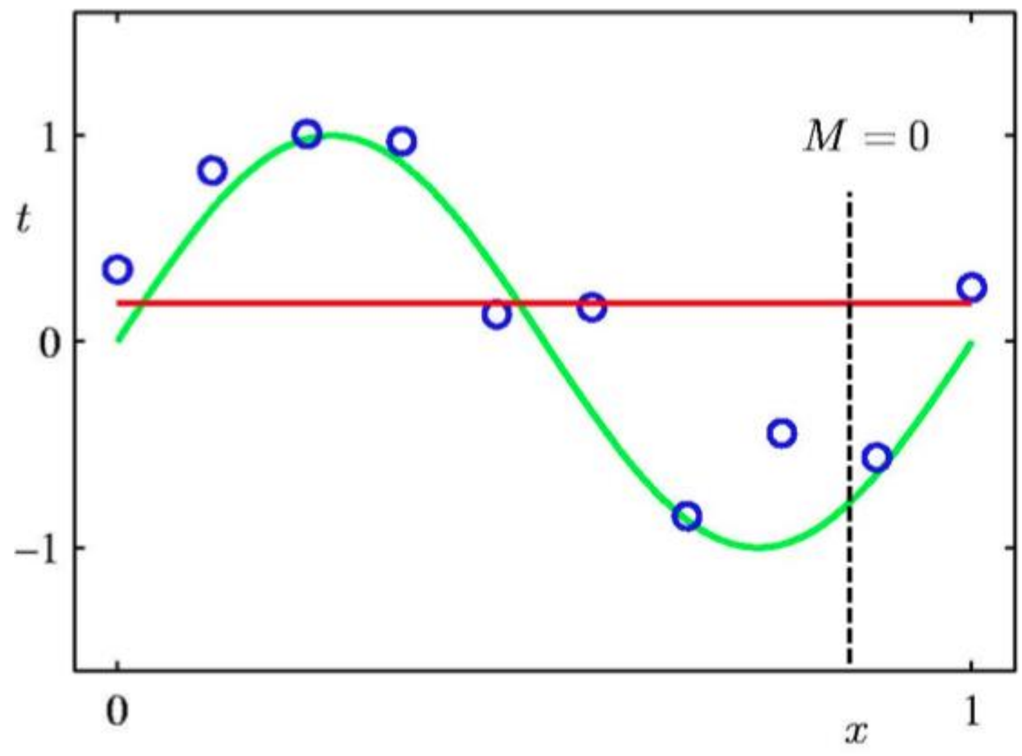
$$\hat{\mathbf{w}} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-\mathbf{1}}\mathbf{X}^{\mathbf{T}}\mathbf{y}$$

Polynomial Curve Fitting

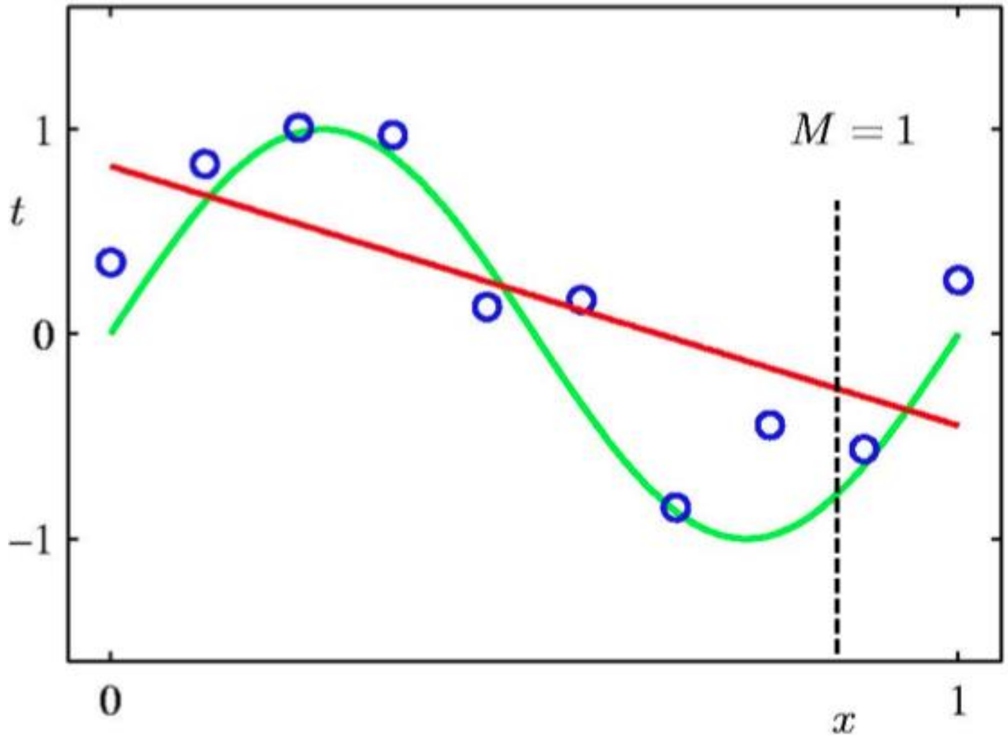
- Generalization = how well the parameterized $h(x, \mathbf{w})$ performs on arbitrary (unseen) test instances $x \epsilon X$.

- Generalization performance depends on the value of M

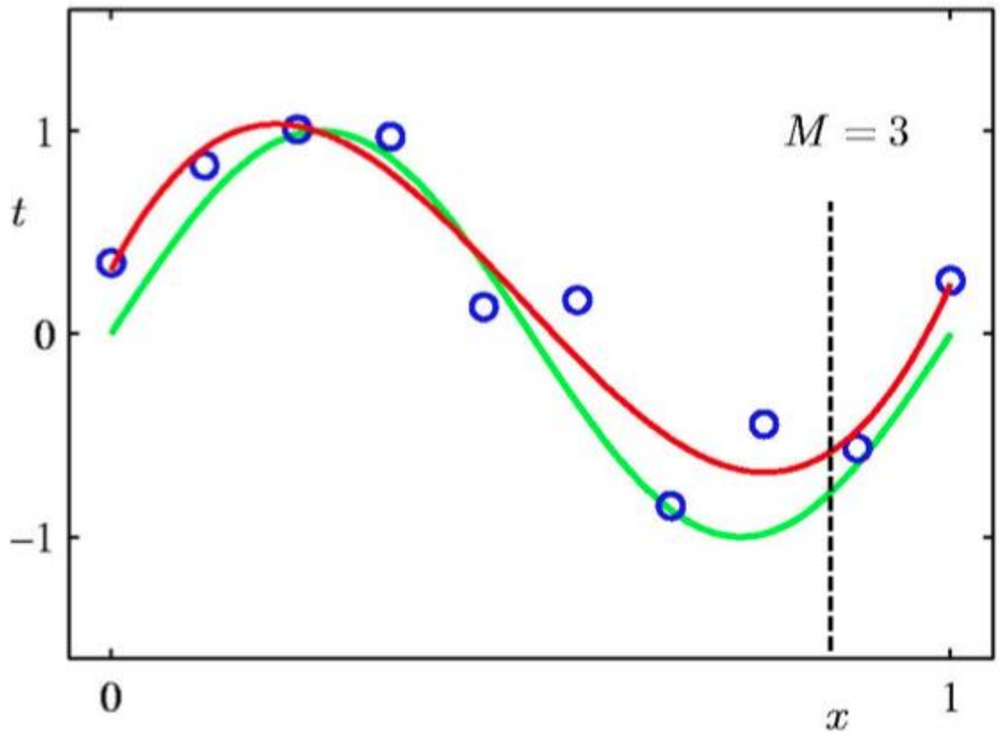
0^{th} Order Polynomial



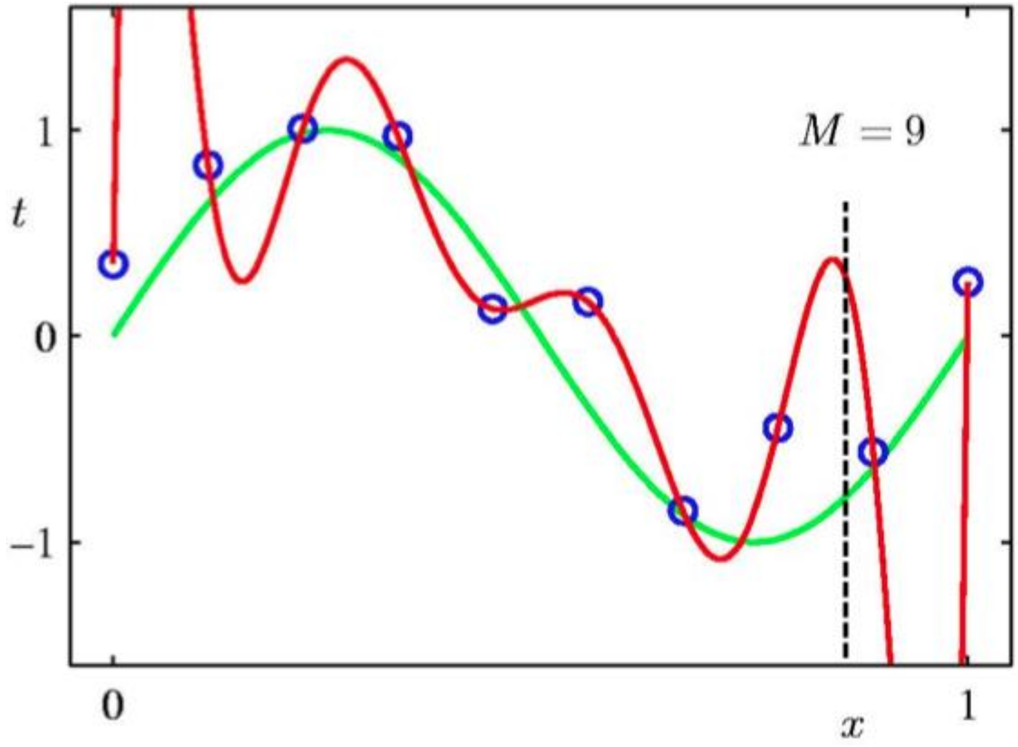
1st Order Polynomial



3^{rd} Order Polynomial



9th Order Polynomial



- Which M to pick? Why?
- Follow the wisdom of a philosopher.

Occam's Razor

William of Occam (1288 – 1348)

English Franciscan friar, theologian and philosopher.

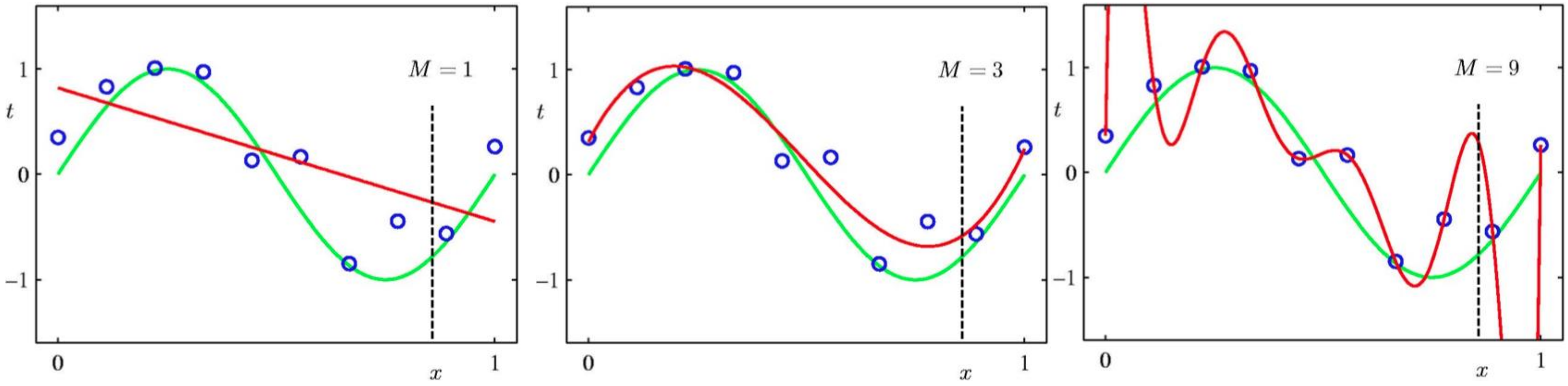
"*Entia non sunt multiplicanda praeter necessitatem*"

- Entities must not be multiplied beyond necessity.

i.e. Do not make things needlessly complicated.

i.e. Prefer the simplest hypothesis that fits the data.

Polynomial Curve Fitting



- **Model Selection**: choosing the order M of the polynomial.
  - Best generalization obtained with M=3.
  - M = 9 obtains poor generalization, even though it fits training examples perfectly:
    - But M = 9 polynomials subsume M = 3 polynomials!

- **Overfitting** ≡ good performance on training examples, poor performance on test examples.

Over-fitting and Parameter Values

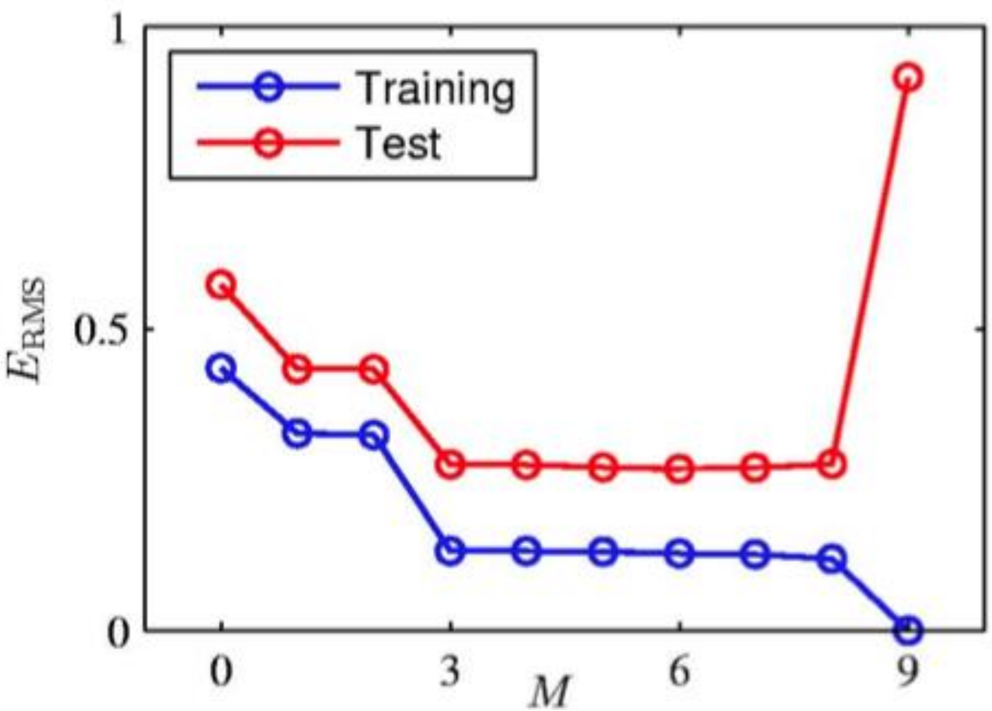
	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Overfitting

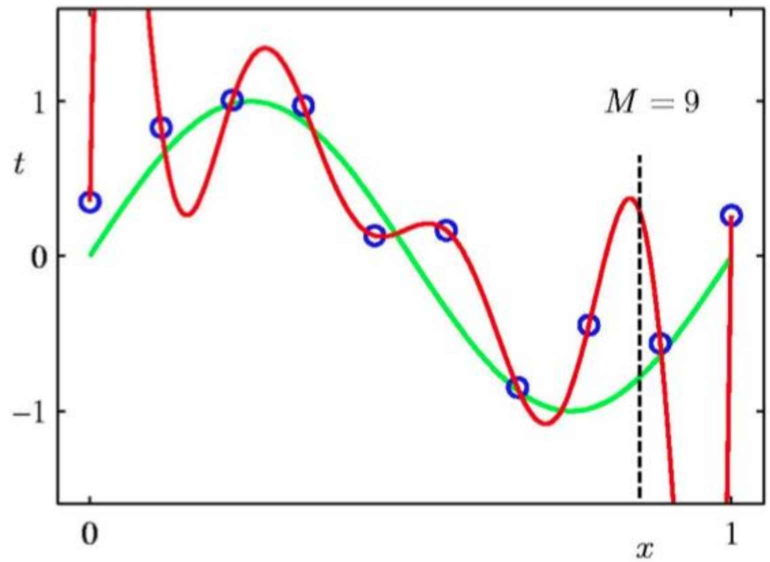
- Measure fit using the Root-Mean-Square (RMS) error (RMSE):

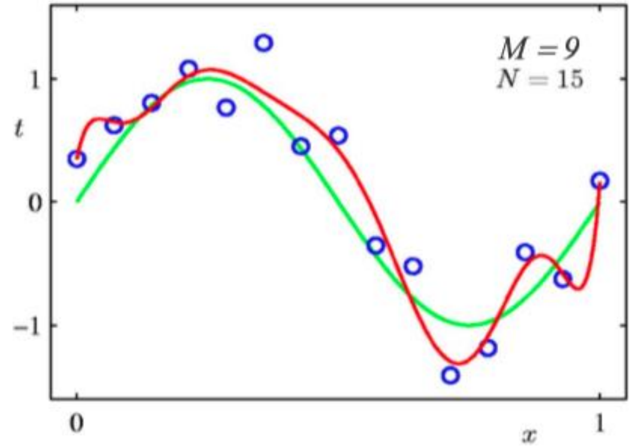
$$E_{RMS(\mathbf{w})} = \sqrt{\frac{\sum_n (\mathbf{w}^T \mathbf{x}_n - \boldsymbol{t}_n)^2}{N}}$$

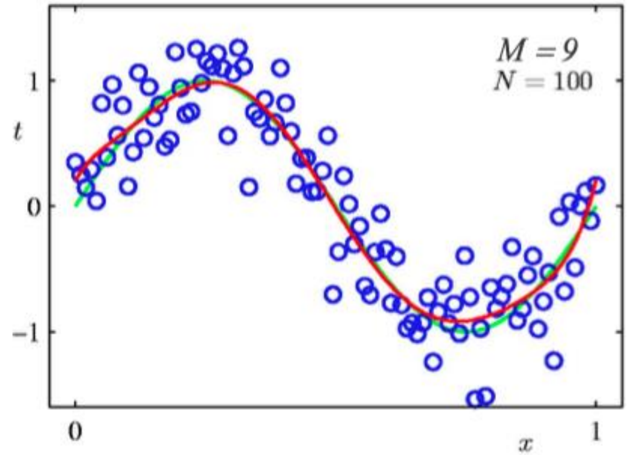
- Use 100 random test examples, generated in the same way:



Overfitting vs. Data Set Size







- More training data ⟹ less overfitting

- What if we do not have more training data?
  - Use **regularization**

Regularization

- Penalize large parameter values:

$$E(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(h_{\mathbf{w}}(\mathbf{x}_n) - t_n)^2 + \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|^2}_{\textit{Regularizer}}$$

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} E(\mathbf{w})$$

Ridge Regression

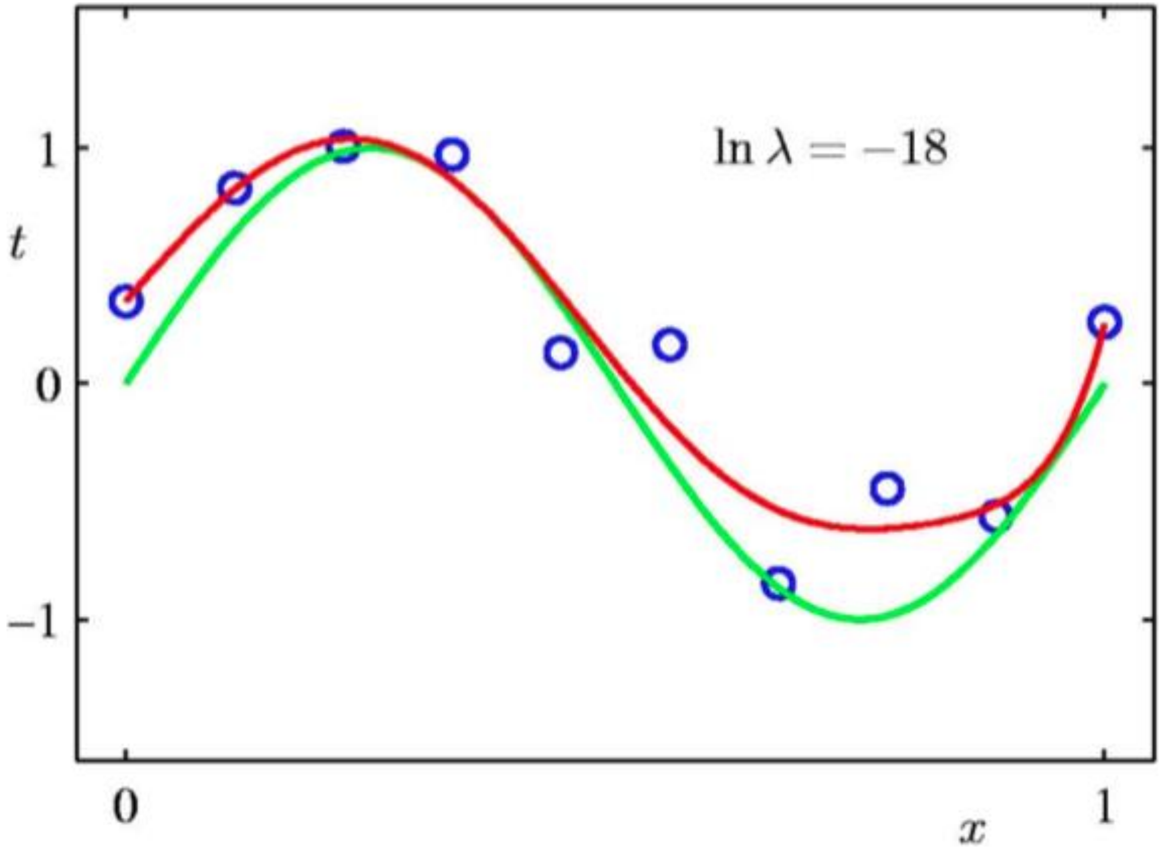
- Multiple linear regression with L2 regularization:

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(h_{\mathbf{w}}(\mathbf{x}_n) - t_n)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

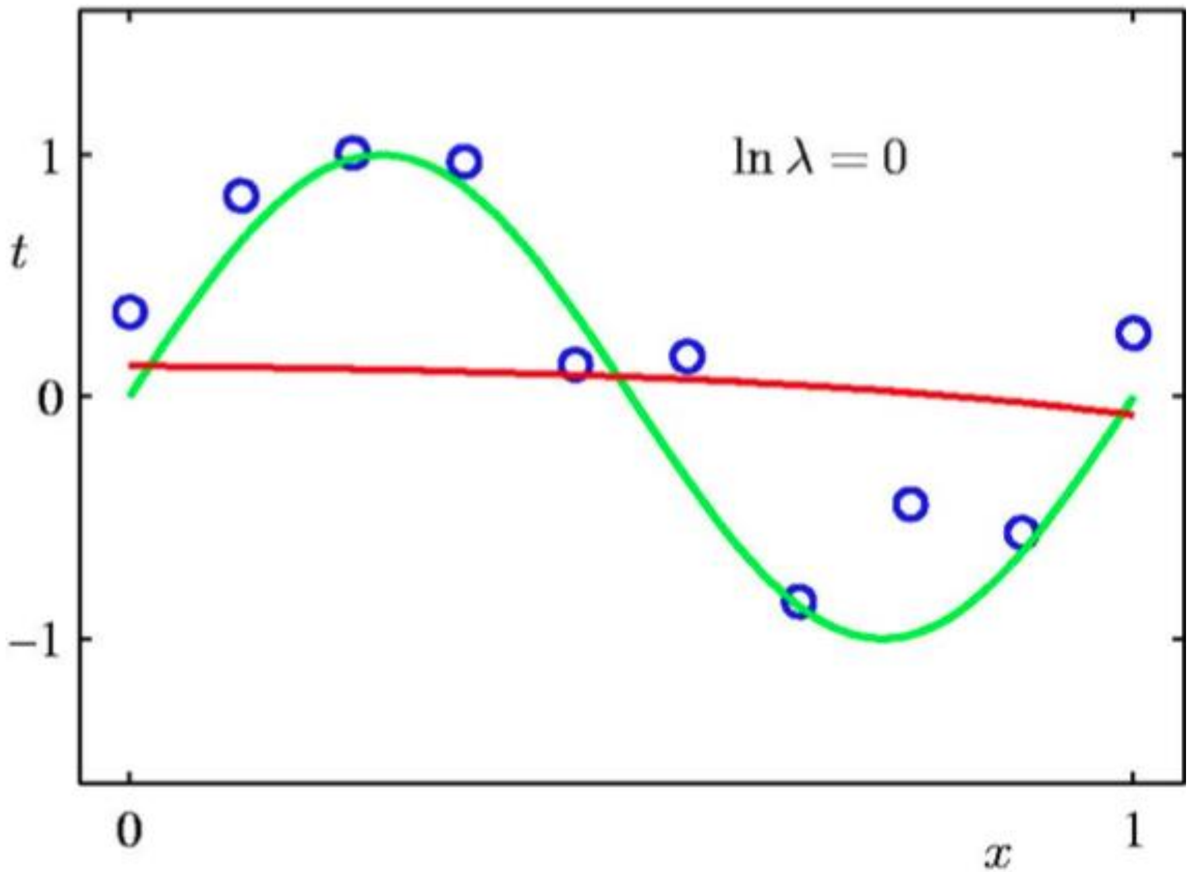
$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} J(\mathbf{w})$$

- Solution is $\mathbf{w} = (\boldsymbol{\lambda} \boldsymbol{N}\mathbf{I} + \mathbf{X}^{\boldsymbol{T}}\mathbf{X})^{-\mathbf{1}}\mathbf{X}^{\boldsymbol{T}}\mathbf{t}$
  - Prove it.

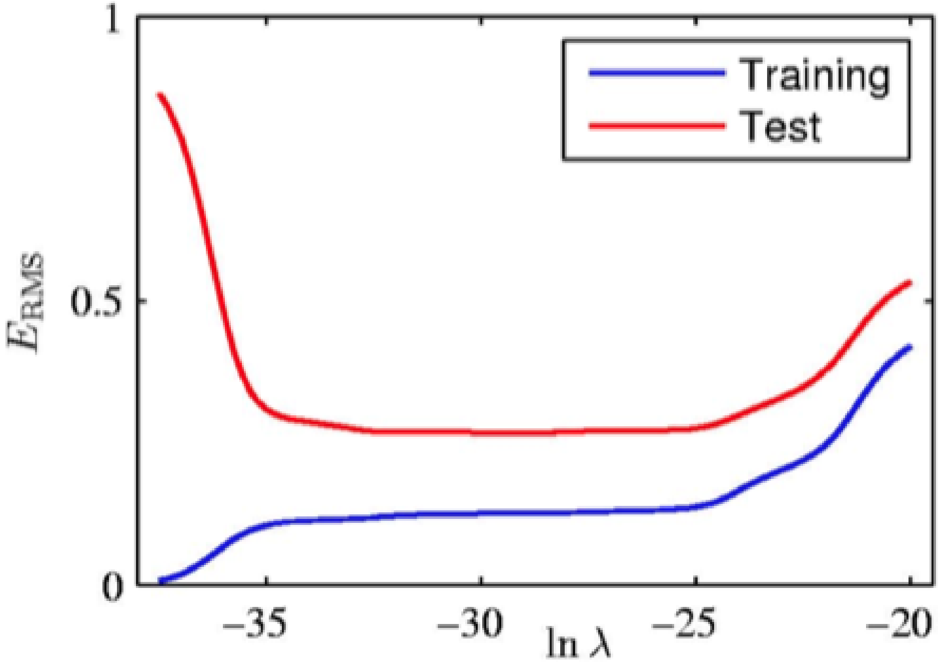
9th Order Polynomial with Regularization



9^{th} Order Polynomial with Regularization



Training & Test error vs. ln λ

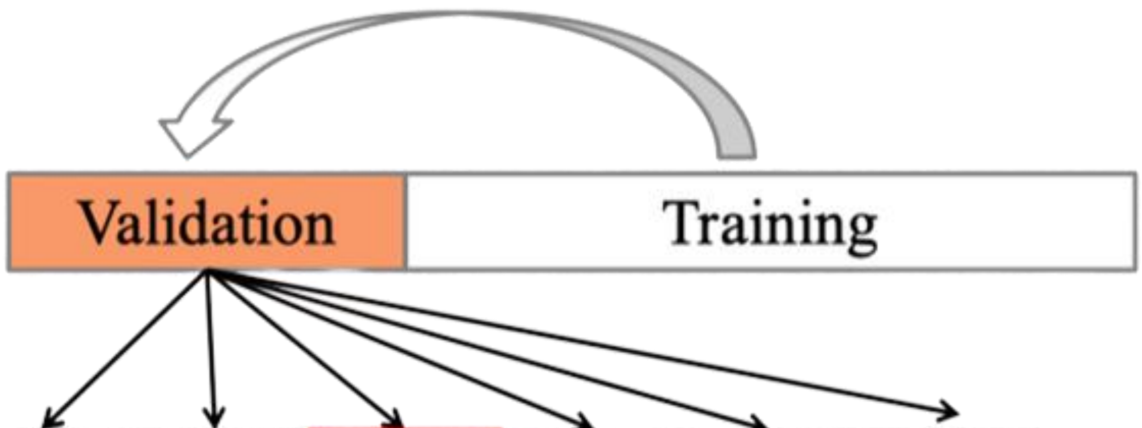


How do we find the optimal value of λ?

Model Selection

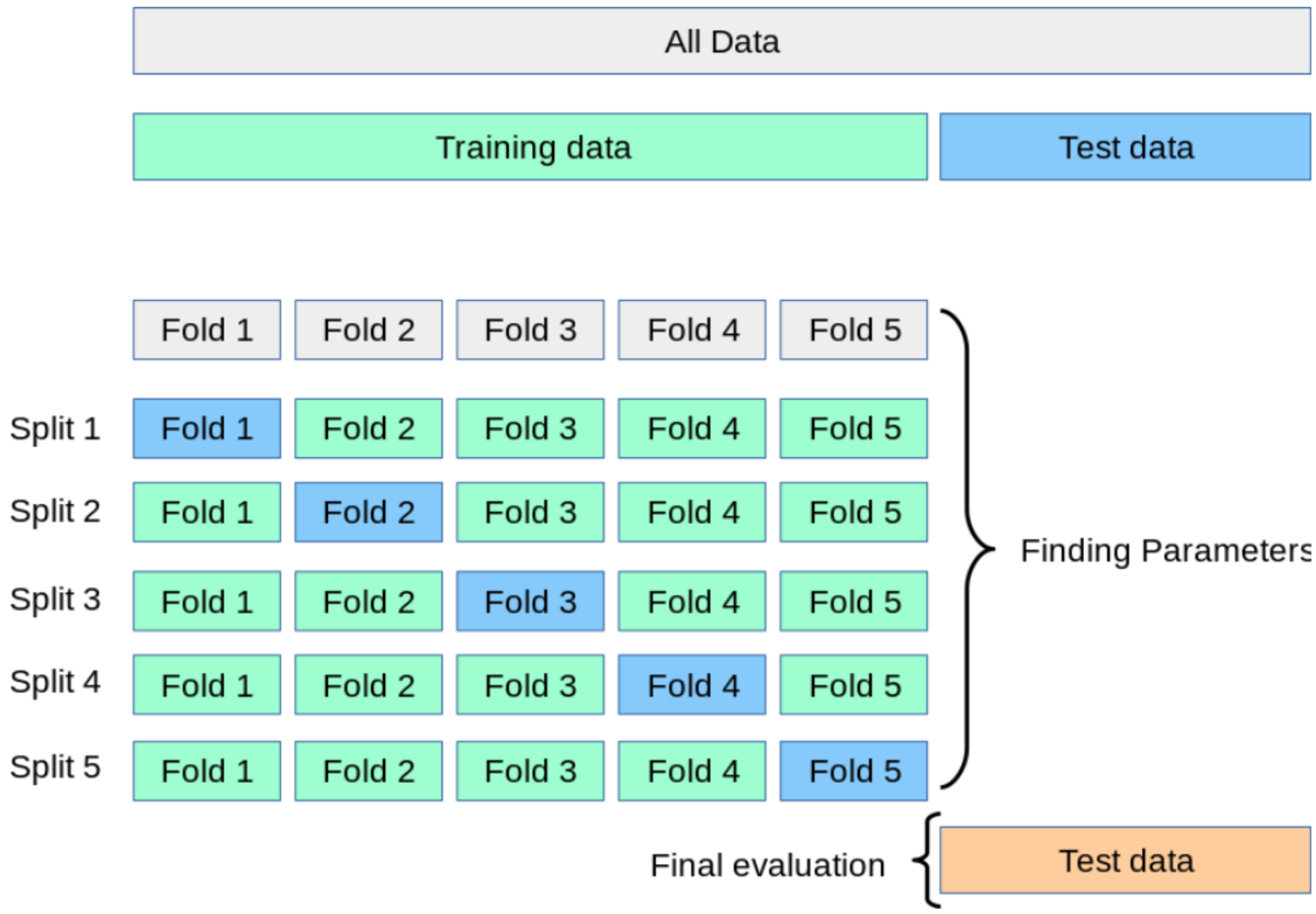
- Put aside an independent validation set.
- Select parameters giving best performance on validation set.

$$\ln \lambda \, \epsilon \{-40, -35, -30, -25, -20, -15\}$$



$\ln \lambda$	-40	-35	-30	-25	-20	-15
E_{RMS}	1.05	0.30	0.25	0.27	0.52	0.55

K-fold Cross-Validation



Source: https://scikit-learn.org/stable/modules/cross_validation.html

K-fold Cross-Validation

- Split the training data into K folds and try a wide range of tunning parameter values:
  - split the data into K folds of roughly equal size
  - iterate over a set of values for λ
    - iterate over $\mathrm{k} = 1,2, \cdots, \mathrm{K}$
      - use all folds except k for training
      - validate (calculate test error) in the k-th fold
    - error[λ] = average error over the K folds
  - choose the value of λ that gives the smallest error.

Regularization: Ridge vs. Lasso

- Ridge regression:

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(h_{\mathbf{w}}(\mathbf{x}_n) - t_n)^2 + \frac{\lambda}{2}\sum_{j=1}^{M} w_j^2$$

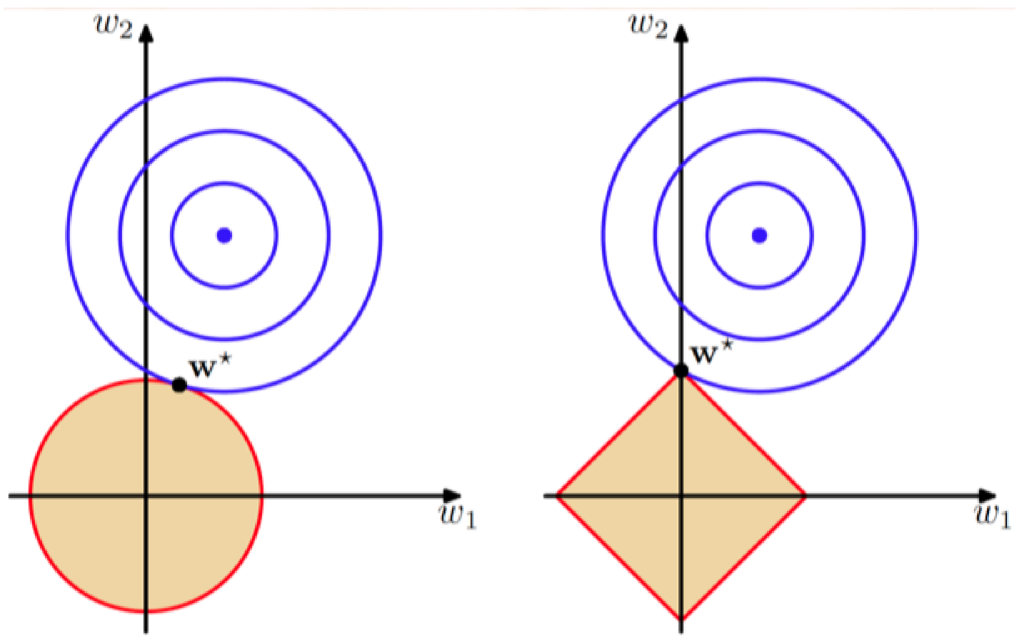
- Lasso:

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(h_{\mathbf{w}}(\mathbf{x}_n) - t_n)^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|$$

- if λ is sufficiently large, some of the coefficients w_j are driven to 0 $\Longrightarrow$ sparse model

Regularization: Ridge vs. Lasso

Plot of the contours of the unregularized error function (blue) along with the constraint region (3.30) for the quadratic regularizer $q = 2$ on the left and the lasso regularizer $q = 1$ on the right, in which the optimum value for the parameter vector $\mathbf{w}$ is denoted by $\mathbf{w}^*$. The lasso gives a sparse solution in which $\mathbf{w}^* = \mathbf{0}$.



Regularization

- **Parameter norm penalties (term in the objective).**
- Limit parameter norm (constraint).
- Dataset augmentation.
- Dropout.
- Ensembles.
- Semi-supervised learning.
- Early stopping
- Noise robustness.
- Sparse representations.
- Adversarial training.

Questions?

UNIVERSITY OF NORTH CAROLINA
CHARLOTTE