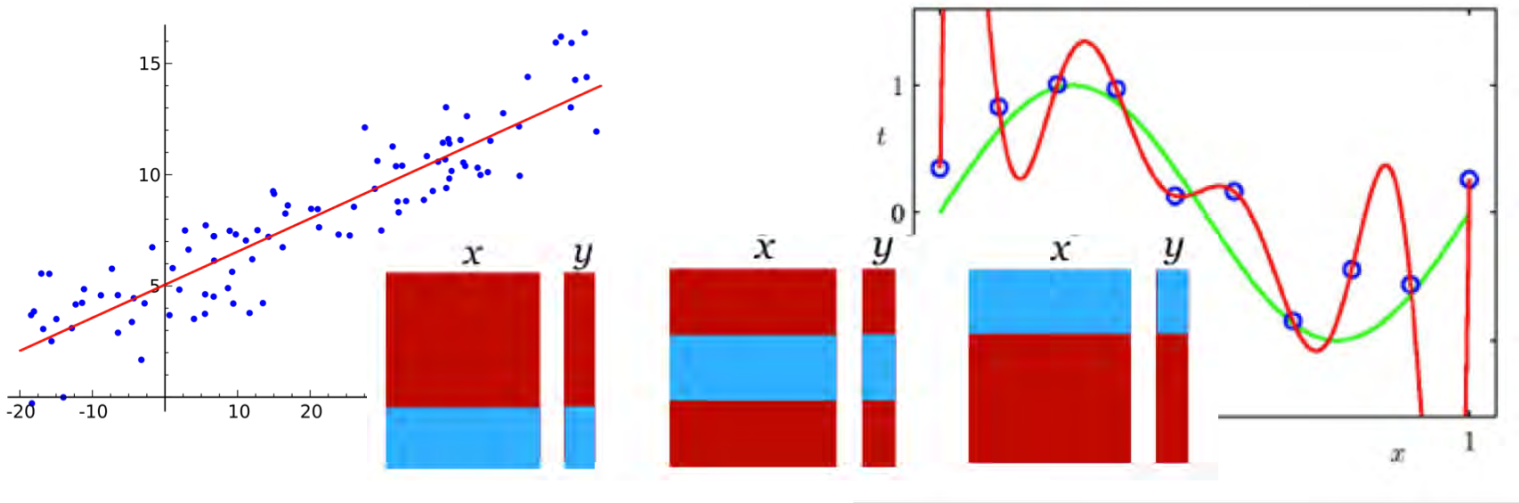
Tufts COMP 135: Introduction to Machine Learning
https://www.cs.tufts.edu/comp/135/2019s/

Cross Validation and Penalized Linear Regression



Prof. Mike Hughes

Many slides attributable to:
Erik Sudderth (UCI)
Finale Doshi-Velez (Harvard)
James, Witten, Hastie, Tibshirani (ISL/ESL books)

CV & Penalized LR Objectives

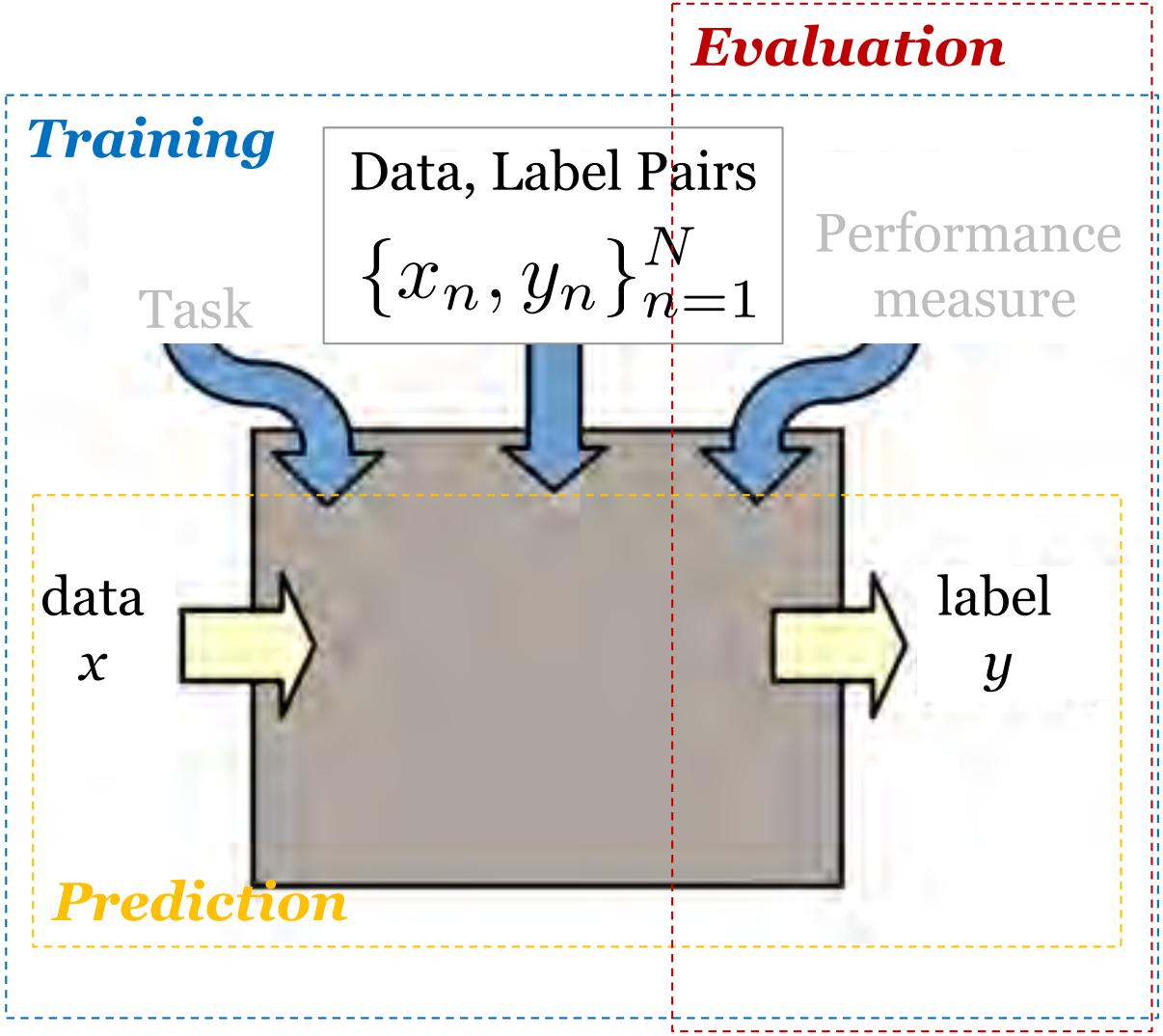
- Regression with transformations of features
- Cross Validation
- L2 penalties
- L1 penalties

What will we learn?

Supervised Learning

Unsupervised Learning

Reinforcement Learning



Task: Regression

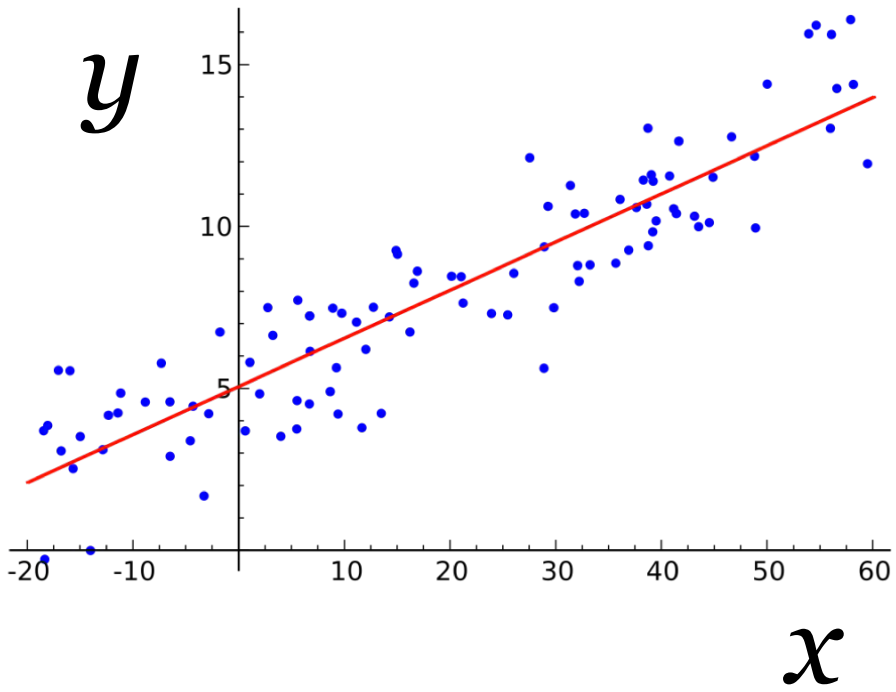
Supervised Learning

regression

Unsupervised Learning

Reinforcement Learning

y is a numeric variable
e.g. sales in $$



Review: Linear Regression

Optimization problem: "Least Squares"

$$\min_{w,b} \sum_{n=1}^{N} \left(y_n - \hat{y}(x_n, w, b) \right)^2$$

Exact formula for optimal values of w, b exist!

$$\tilde{X} = \begin{bmatrix} x_{11} & \dots & x_{1F} & 1 \\ x_{21} & \dots & x_{2F} & 1 \\ & & \dots & \\ x_{N1} & \dots & x_{NF} & 1 \end{bmatrix}$$

$$[w_1 \dots w_F \ b]^T = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T y$$

Math works in 1D and for many dimensions

Recap: solving linear regression

- More examples than features (N > F)

  And if inverse of X^T X exists (needs to be full rank)

  Then an optimal weight vector exists, can use formula

  Likely has non-zero error (overdetermined)

- Same number of examples and features (N=F)

  And if inverse of X^T X exists (needs to be full rank):

  Then an optimal weight vector exists, can use formula

  Will have zero error on training set.

- Fewer examples than features (N < F) or low rank

  Then:

  Infinitely many optimal weight vectors exist with zero error

  Inverse of X^T X does not exist (naïvely, formula will fail)

Recap

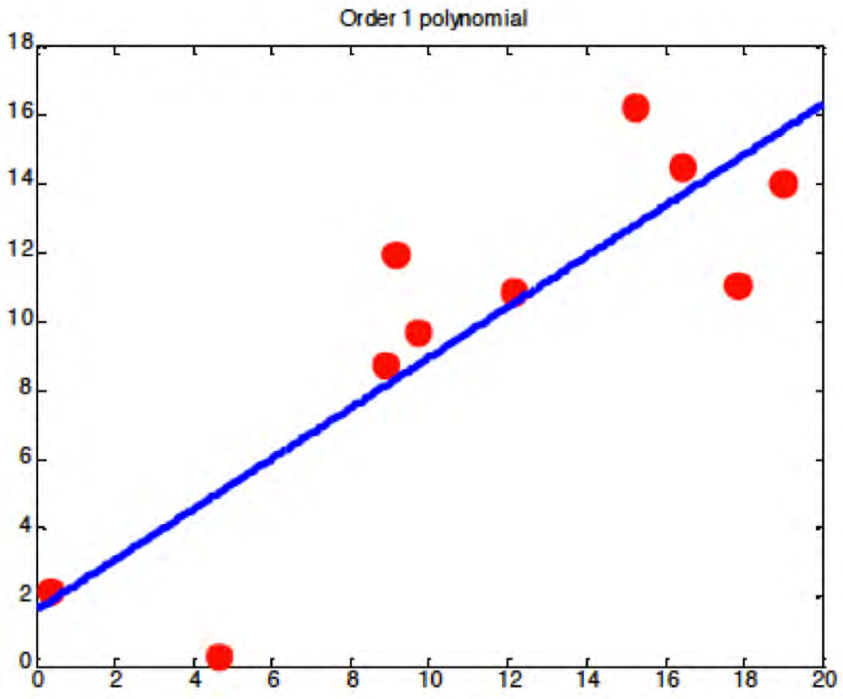
- Squared error is ***special***
  - Exact formulas for estimating parameters

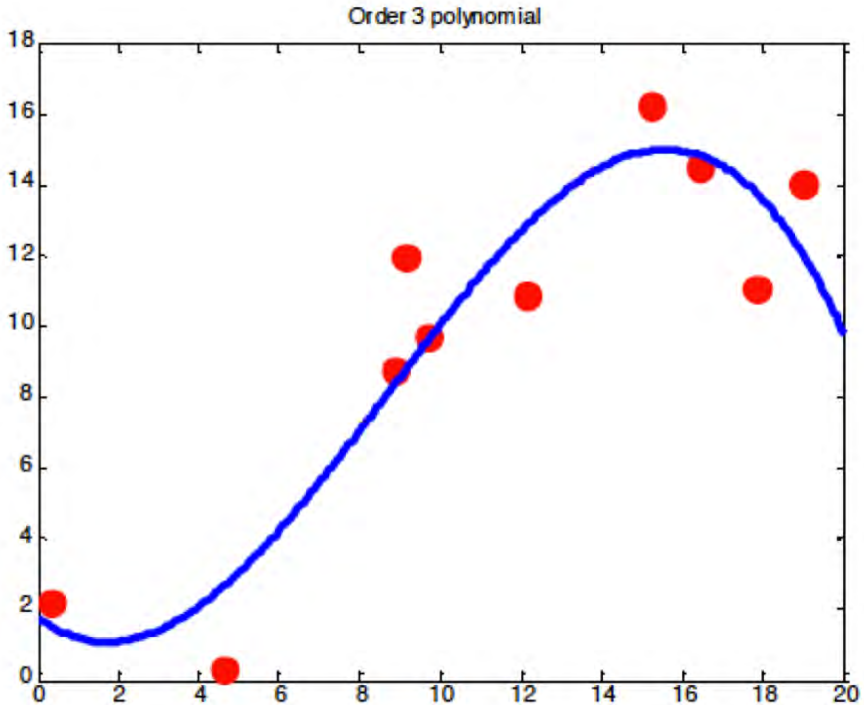
- Most metrics do not have exact formulas
  - Take derivative, set to zero, try to solve, …. **HARD**!
  - Example: absolute error

- General algorithm: Gradient Descent!
  - As long as first derivative exists, we can do iterations to estimate optimal parameters

Transformations of Features

Fitting a line isn't always ideal





(c) Alexander Ihler

Can fit **linear** functions to **nonlinear** features

A nonlinear function of x:

$$\hat{y}(x_i) = \theta_0 + \theta_1 x_i + \theta_2 x_i^2 + \theta_3 x_i^3$$

Can be written as a linear function of $\phi(x_i) = [1 \; x_i \; x_i^2 \; x_i^3]$

$$\hat{y}(x_i) = \sum_{g=1}^{4} \theta_g \phi_g(x_i) = \theta^T \phi(x_i)$$

"Linear regression" means linear in the parameters (weights, biases)

Features can be arbitrary transforms of raw data

What feature transform to use?

- Anything that works for your data!
  - sin / cos for periodic data
  - polynomials for high-order dependencies

$$\phi(x_i) = [1 \; x_i \; x_i^2 \; \ldots]$$

  - interactions between feature dimensions

$$\phi(x_i) = [1 \quad x_{i1}x_{i2} \quad x_{i3}x_{i4} \; \ldots]$$

  - Many other choices possible

Linear Regression with Transformed Features

$$\phi(x_i) = [1 \ \phi_1(x_i) \ \phi_2(x_i) \ldots \phi_{G-1}(x_i)]$$

$$\hat{y}(x_i) = \theta^T \phi(x_i)$$

Optimization problem: "Least Squares"

$$\min_\theta \sum_{n=1}^{N} (y_n - \theta^T \phi(x_i))^2$$

Exact solution:

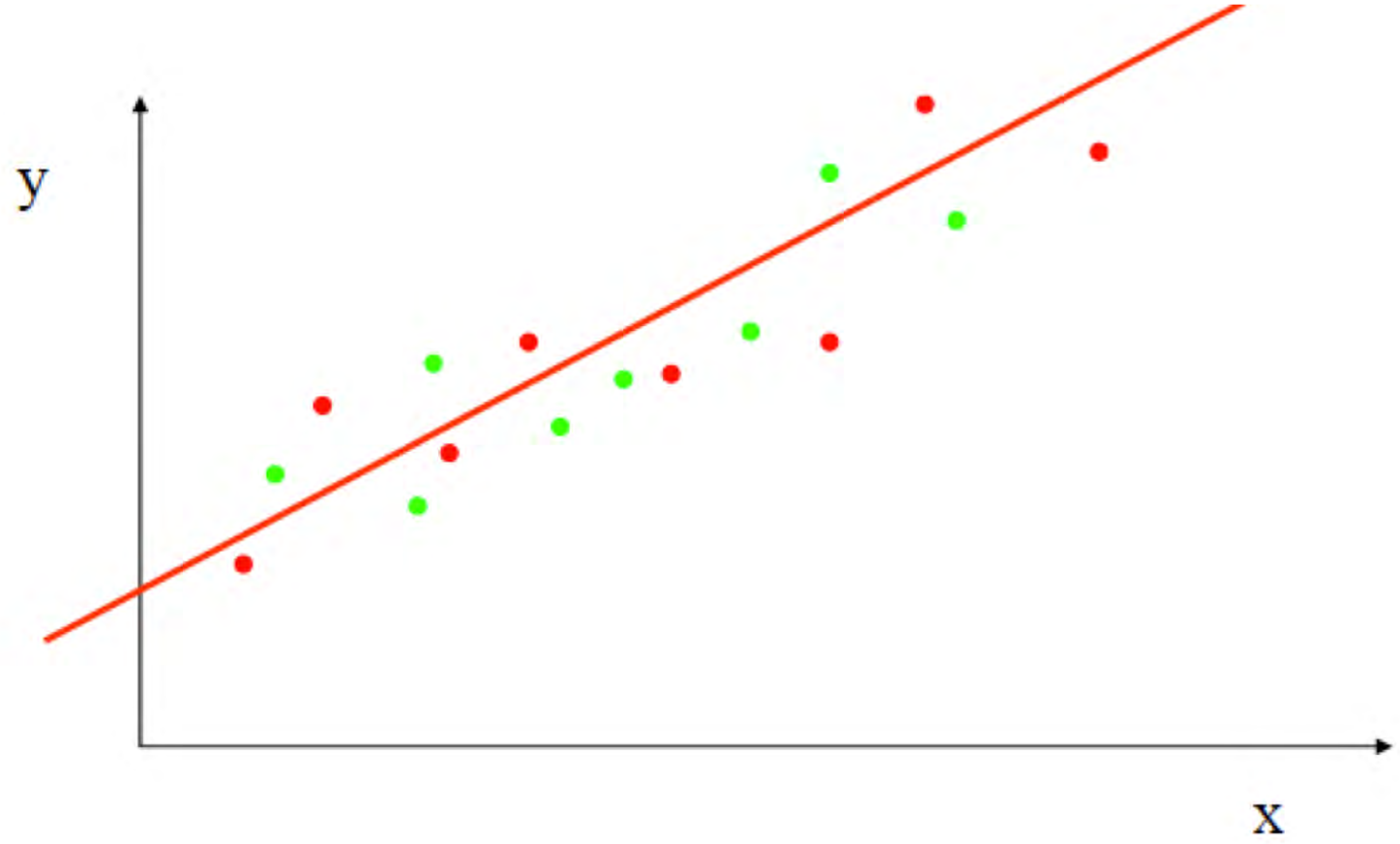
$$\theta^* = (\Phi^T \Phi)^{-1} \Phi^T y$$

$$\Phi = \begin{bmatrix} 1 & \phi_1(x_1) & \ldots & \phi_{G-1}(x_1) \\ 1 & \phi_1(x_2) & \ldots & \phi_{G-1}(x_2) \\ \vdots & & \ddots & \\ 1 & \phi_1(x_N) & \ldots & \phi_{G-1}(x_N) \end{bmatrix}$$

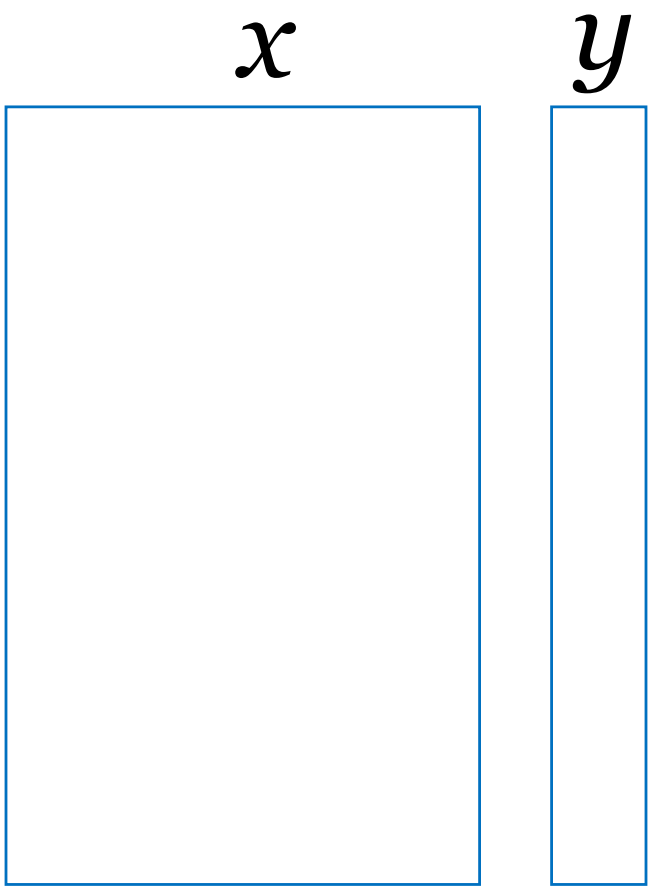
N x G matrix

Cross Validation

Generalize: sample to population



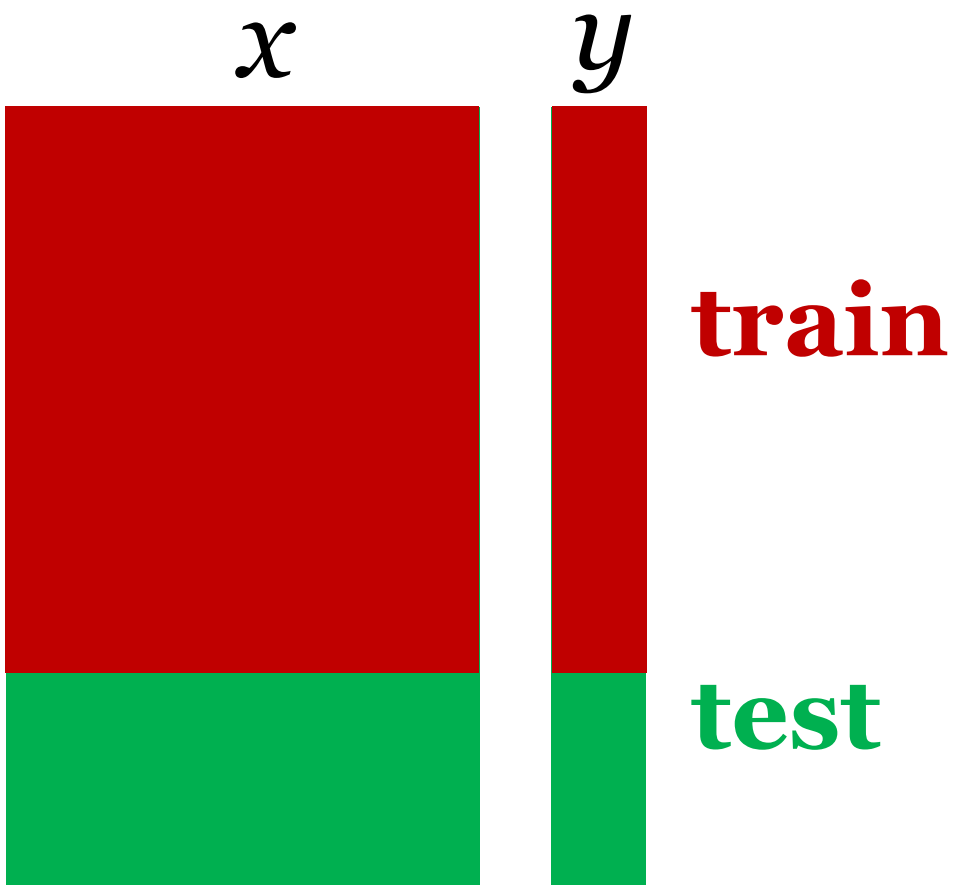
Labeled dataset



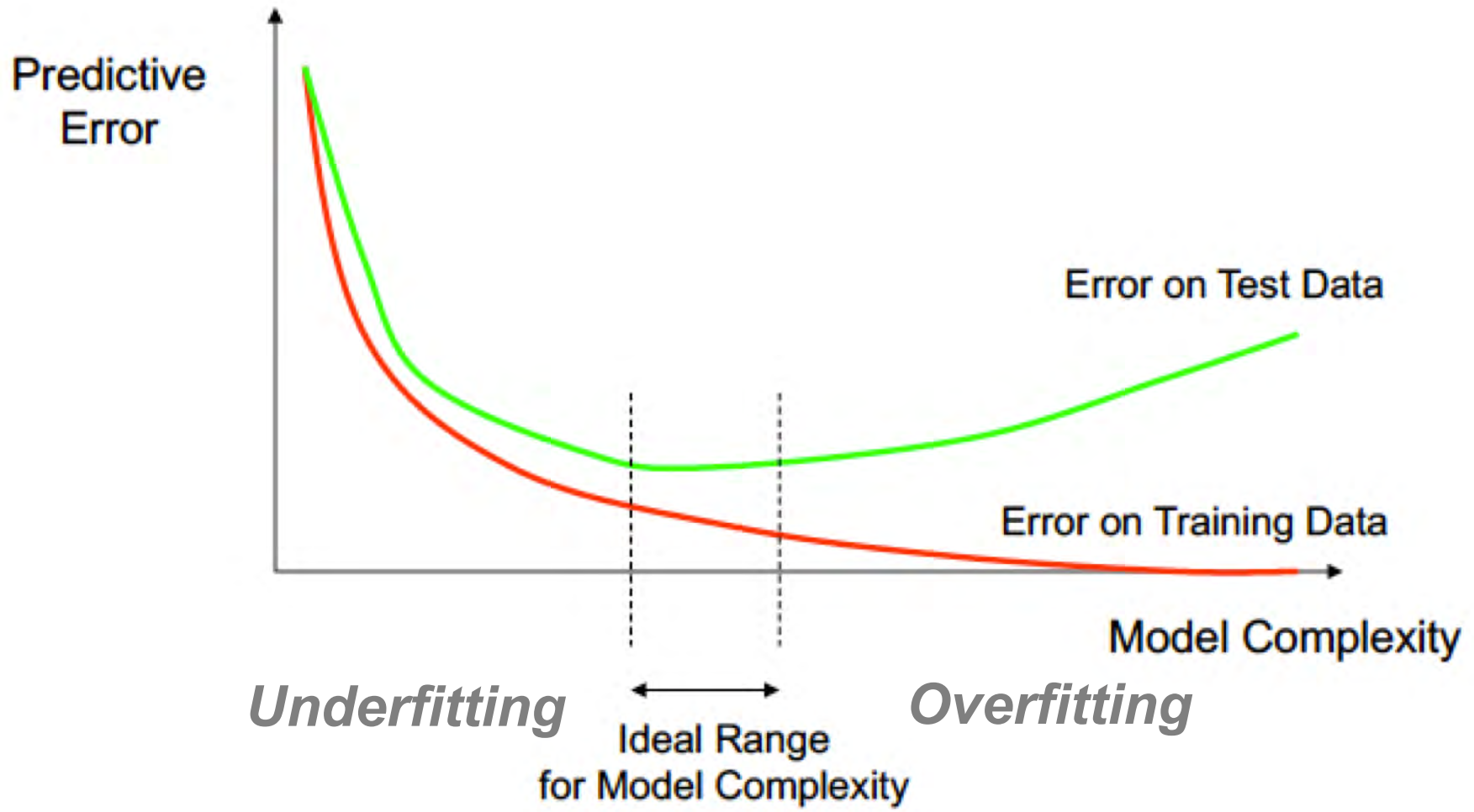
Each row represents one example

Assume rows are arranged "uniformly at random" (order doesn't matter)

Split into train and test



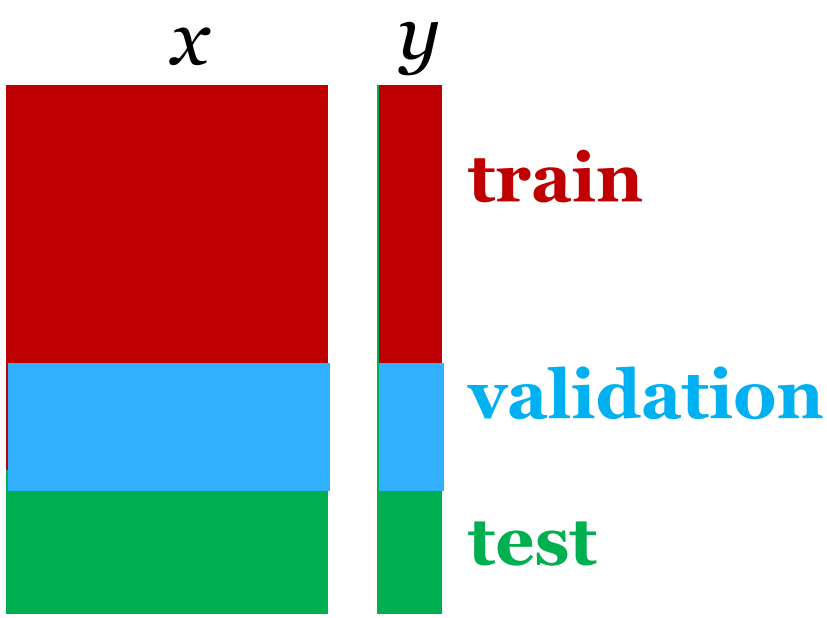
Model Complexity vs Error



How to fit best model?

Option: Fit on train, select on **validation**

1) Fit each model to **training** data
2) Evaluate each model on **validation** data
3) Select model with lowest **validation** error
4) Report error on **test** set



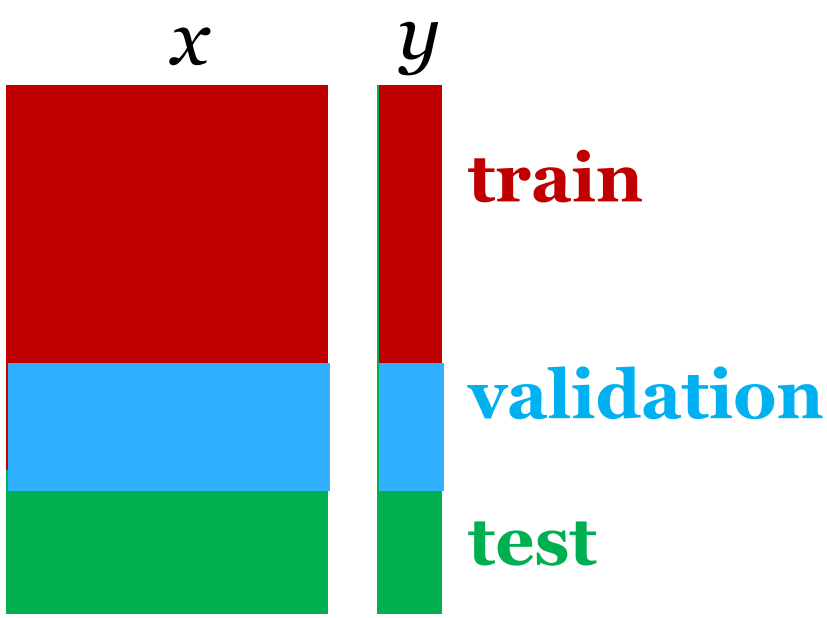
How to fit best model?

Option: Fit on train, select on **validation**

1) Fit each model to **training** data
2) Evaluate each model on **validation** data
3) Select model with lowest **validation** error
4) Report error on **test** set

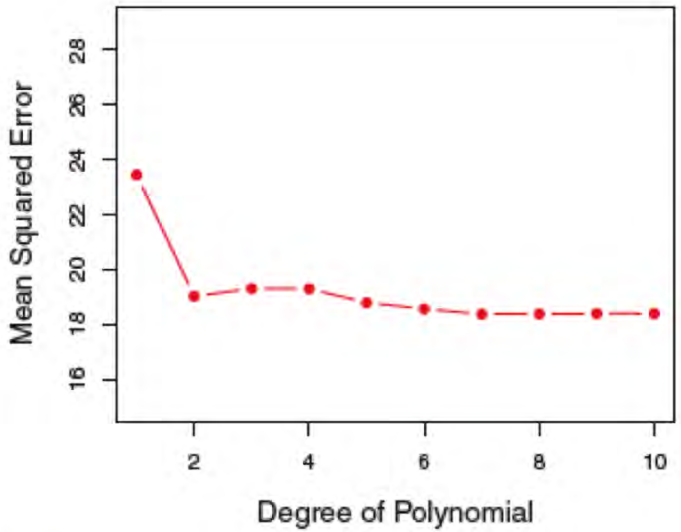
Concerns

- Will train be too small?
- Make better use of data?



Estimating Heldout Error with Fixed Validation Set

Single random split



10 other random splits

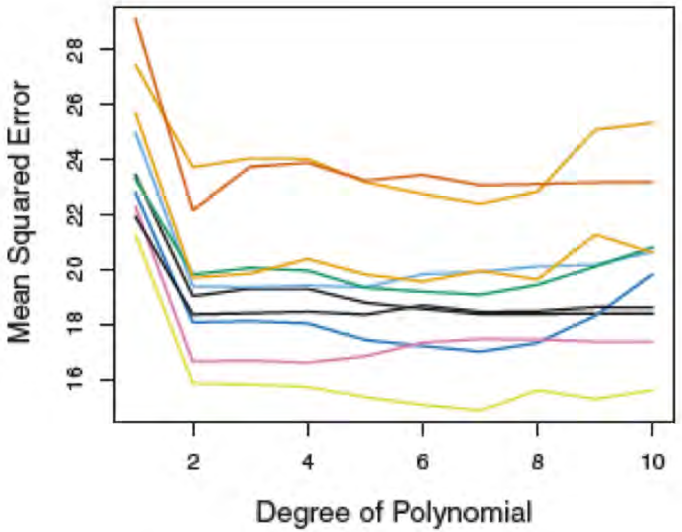
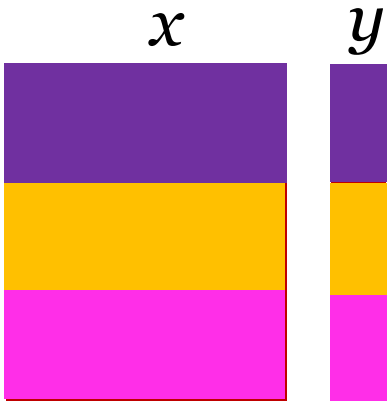


FIGURE 5.2. *The validation set approach was used on the* `Auto` *data set in order to estimate the test error that results from predicting* `mpg` *using polynomial functions of* `horsepower`. Left: *Validation error estimates for a single split into training and validation data sets.* Right: *The validation method was repeated ten times, each time using a different random split of the observations into a training set and a validation set. This illustrates the variability in the estimated test MSE that results from this approach.*

Credit: ISL Textbook, Chapter 5

3-fold Cross Validation

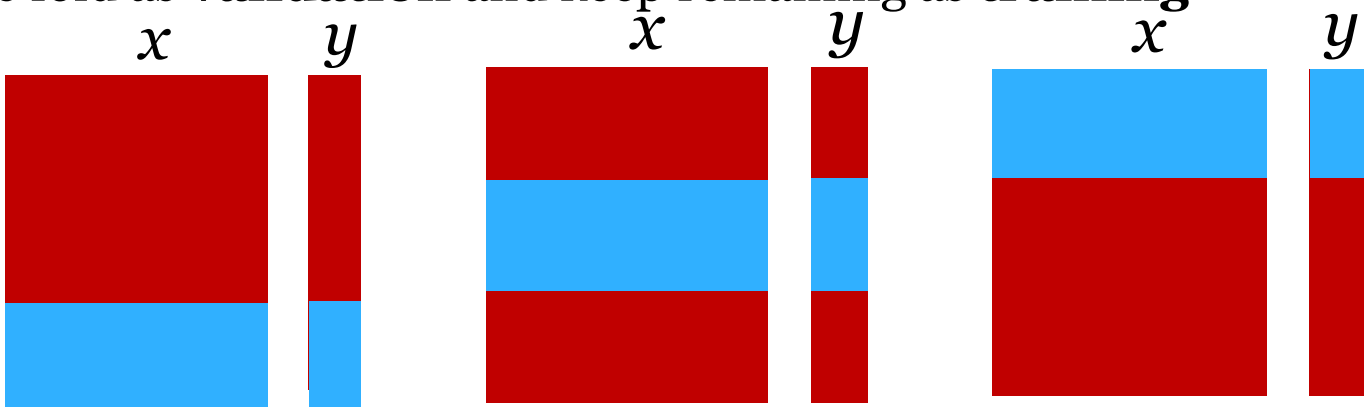
Divide labeled dataset
into 3 even-sized parts



Fit model 3 independent times.
Each time leave one fold as **validation** and keep remaining as **training**

train

validation



Heldout error estimate: average of the validation error across all 3 fits

K-fold CV: How many folds *K*?

- Can do as low as 2 fold
- Can do as high as N-1 folds ("Leave one out")
- Usual rule of thumb: 5-fold or 10-fold CV

- Computation runtime **scales linearly** with *K*
  - Larger K also means each fit uses more train data, so each fit might take longer too

- Each fit is independent and **parallelizable**

Estimating Heldout Error with Cross Validation

Leave-one-out (N-1 folds)

9 separate splits
Each one with 10 folds

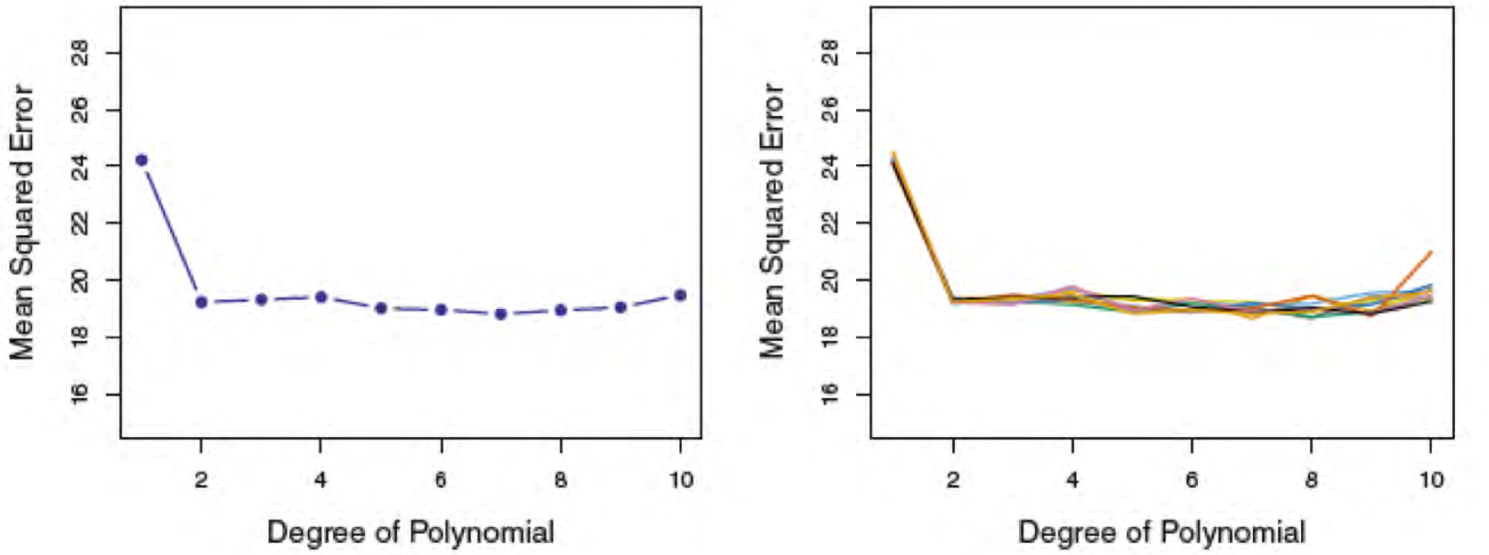


FIGURE 5.4. *Cross-validation was used on the* `Auto` *data set in order to estimate the test error that results from predicting* `mpg` *using polynomial functions of* `horsepower`. Left: *The LOOCV error curve.* Right: 10-*fold CV was run nine separate times, each with a different random split of the data into ten parts. The figure shows the nine slightly different CV error curves.*

Credit: ISL Textbook, Chapter 5

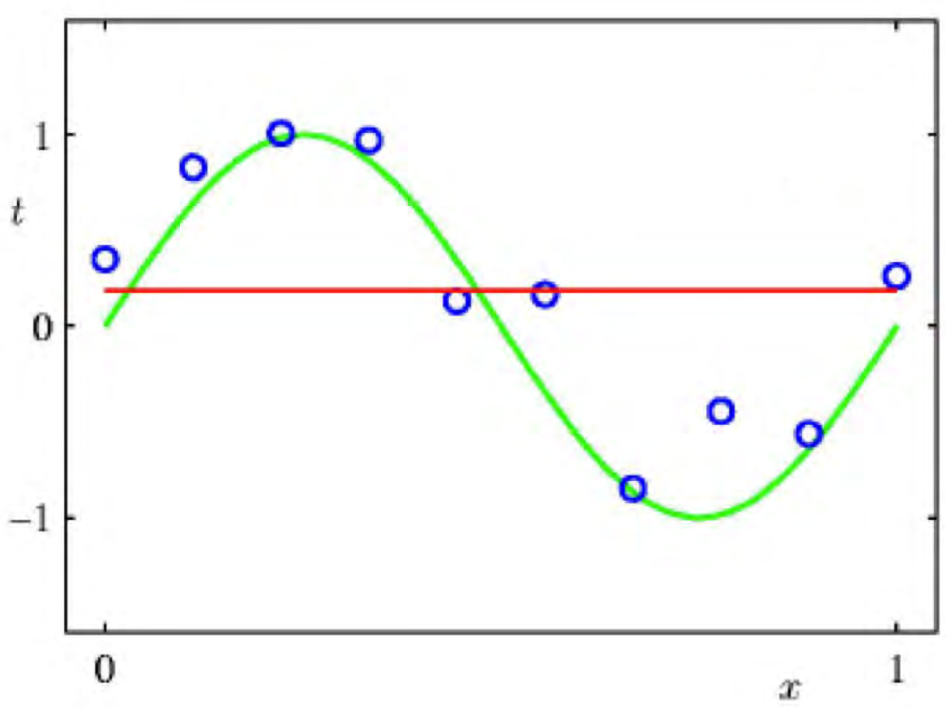
What to do about underfitting?

- Increase model complexity
  - Add more features!

What to do about overfitting?

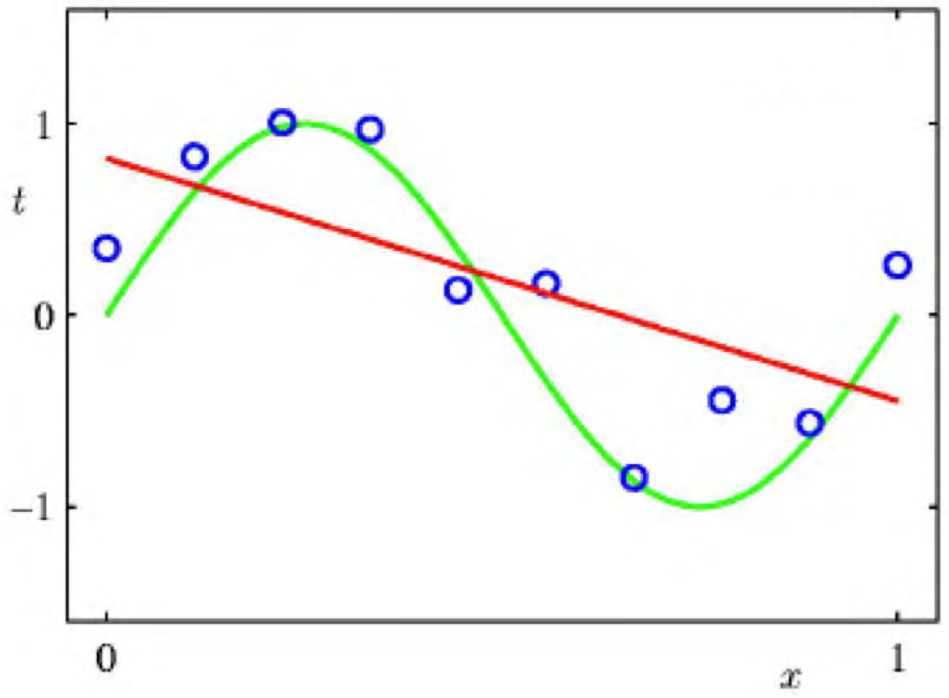
- Select complexity with cross validation
- Control single-fit complexity with a penalty!

Zero degree polynomial



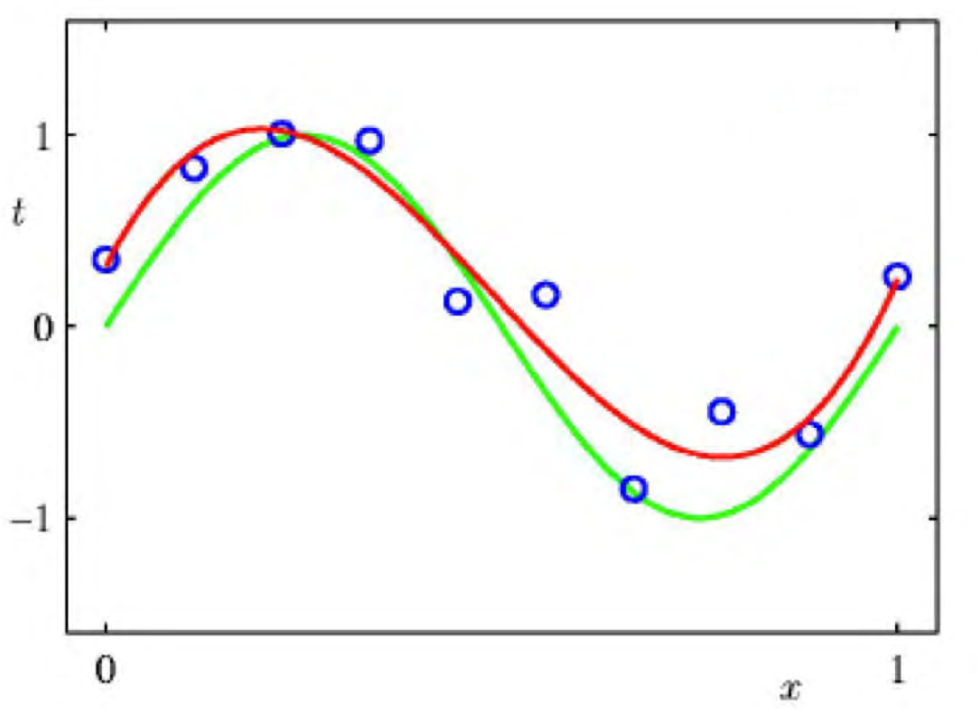
Credit: Slides from course by Prof. Erik Sudderth (UCI)

1st degree polynomial



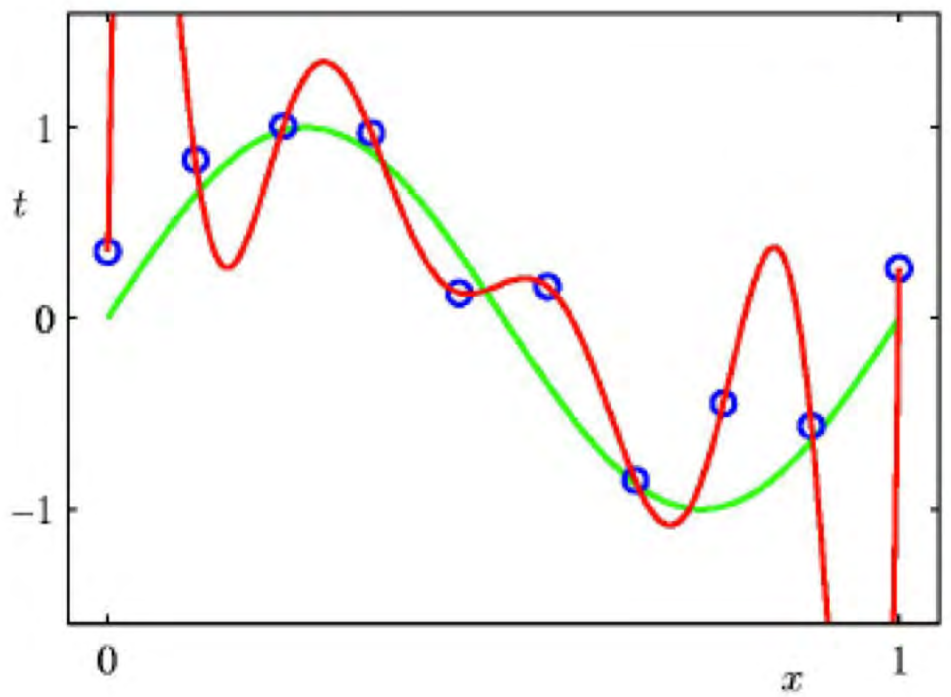
Credit: Slides from course by Prof. Erik Sudderth (UCI)

3rd degree polynomial



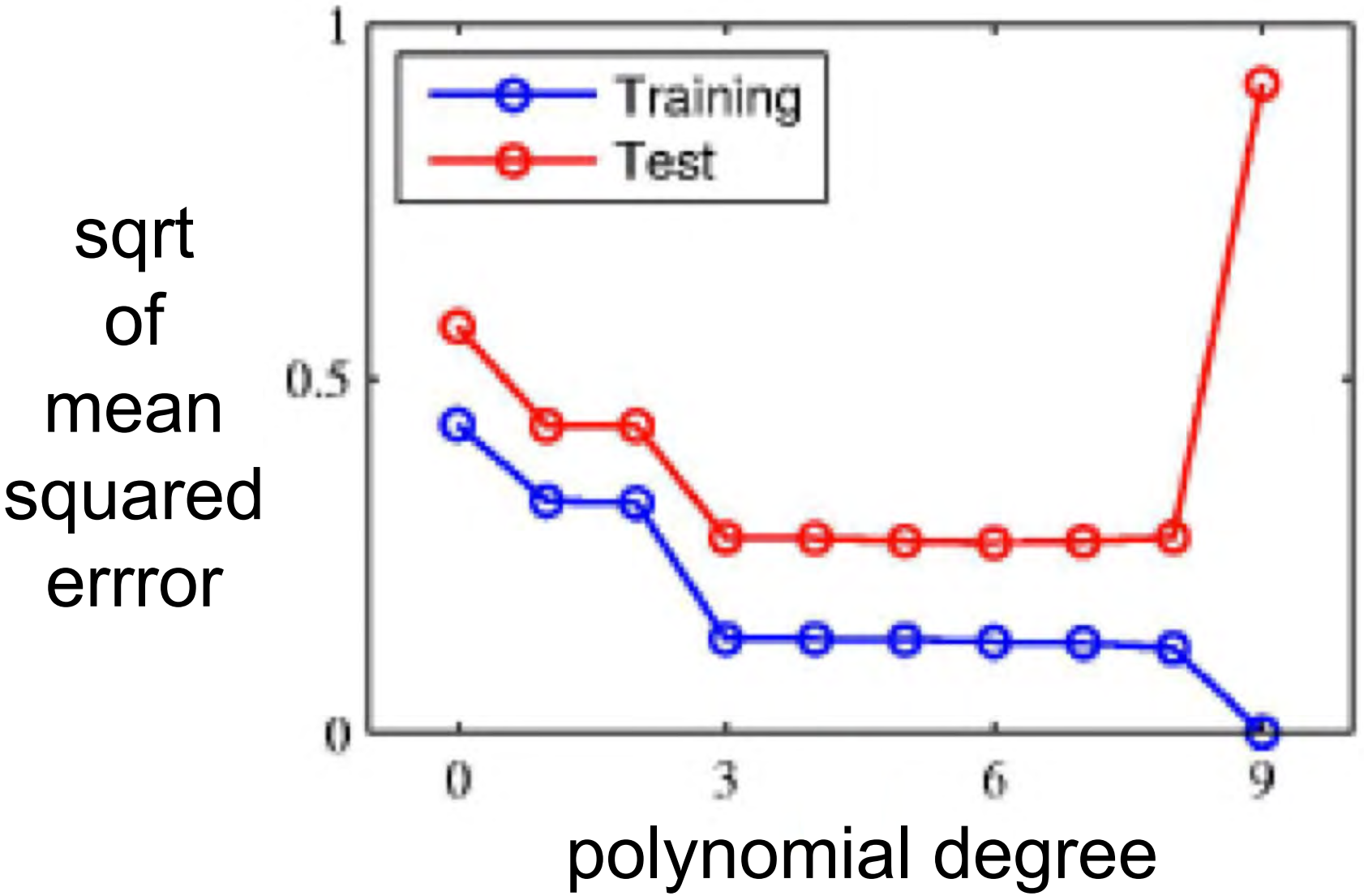
Credit: Slides from course by Prof. Erik Sudderth (UCI)

9^{th} degree polynomial



Credit: Slides from course by Prof. Erik Sudderth (UCI)

Error vs Complexity



Polynomial degree

Estimated Regression Coeffients θ	0	1	3	9
	0.19	0.82	0.31	0.35
		-1.27	7.99	232.37
			-25.43	-5321.83
			17.37	48568.31
				-231639.30
				640042.26
				-1061800.52
				1042400.18
				-557682.99
				125201.43

Credit: Slides from course by Prof. Erik Sudderth (UCI)

Idea:
Penalize magnitude of weights

$$J(\theta) = \frac{1}{2}\sum_{n=1}^{N}(y_n - \theta^T \tilde{x}_n)^2 \quad + \alpha \sum_{f} \theta_f^2$$

Penalty strength: $\alpha \geq 0$

Larger alpha means we prefer smaller magnitude weights

Idea:
Penalize magnitude of weights

$$J(\theta) = \frac{1}{2}\sum_{n=1}^{N}(y_n - \theta^T \tilde{x}_n)^2 \quad + \alpha \sum_{f} \theta_f^2$$

Written via matrix/vector product notation:

$$J(\theta) = \frac{1}{2}(y - \tilde{X}\theta)^T(y - \tilde{X}\theta) + \alpha\theta^T\theta$$

Exact solution for L2 penalized linear regression

Optimization problem: "Penalized Least Squares"

$$\min_{\theta} \quad \frac{1}{2}(y - \tilde{X}\theta)^T(y - \tilde{X}\theta) + \alpha\theta^T\theta$$

Solution:

$$\theta^* = (\tilde{X}^T\tilde{X} + \alpha I)^{-1}\tilde{X}^T y$$

If alpha > 0 , this is always invertible!

Slides on L1/L2 penalties

See slides 71-82 from UC-Irvine course here:

https://canvas.eee.uci.edu/courses/8278/files/2735313/

Pair Coding Activity

https://github.com/tufts-ml-courses/comp135-19s-assignments/blob/master/labs/GradientDescentDemo.ipynb

- Try existing gradient descent code:
  - Optimizes scalar slope to produce minimum error
  - Try step sizes of 0.0001, 0.02, 0.05, 0.1
- Add L2 penalty with alpha > 0
  - Write calc_penalized_loss and calc_penalized_grad
  - What happens to estimated slope value w?
- Repeat with L1 penalty with alpha > 0