Polynomial Regression and Regularization

Administrivia

- If you still haven't enrolled in Moodle yet
  - Enrollment key on Piazza
  - If joined course recently, email me to get added to Piazza
- Homework 1 posted later tonight. Good Milestones:
  - **Problems 1-3 This Week**
  - Problems 4 and 5 Next Week

The RoadMap

- **Last Time:**
  - Regression Refresher (there was nothing fresh about it)
- **This Time:**
  - Polynomial Regression
  - Regularization (wiggles are bad, Man)
- **Next Time:**
  - Bias-Variance Trade-Off (what does it all **MEAN**?)

Previously on CSCI 4622

Given training data $(x_{i1}, x_{i2}, \ldots, x_{ip}, y_i)$ for $i = 1, 2, \ldots, n$ fit a regression of the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i \qquad \text{where} \qquad \epsilon_i \sim N(0, \sigma^2)$$

LOSS FUNCTION

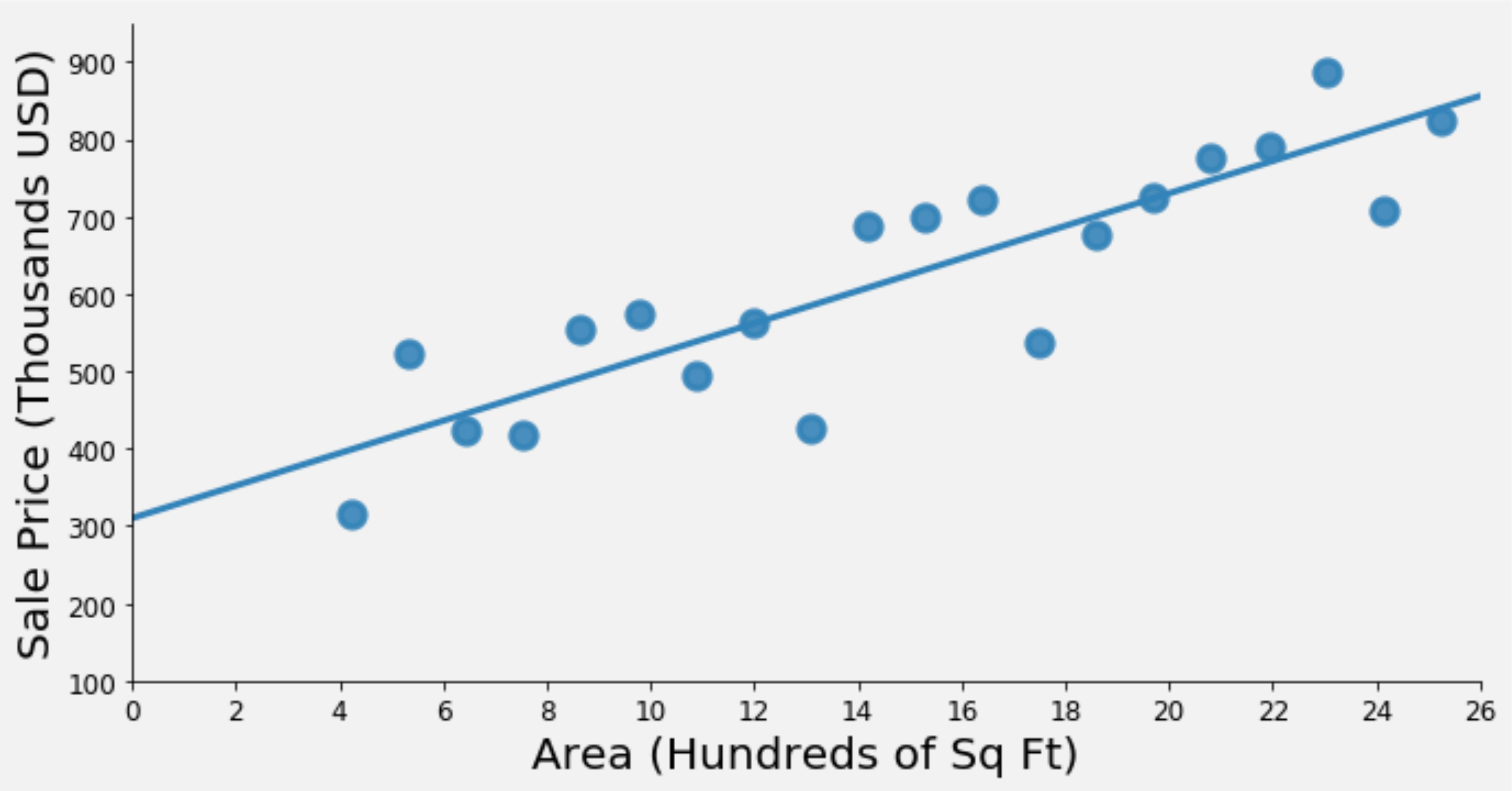
Estimates of the parameters are found by minimizing

$$\text{RSS} = \sum_{i=1}^{n} \left[(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) - y_i\right]^2 = \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|^2$$

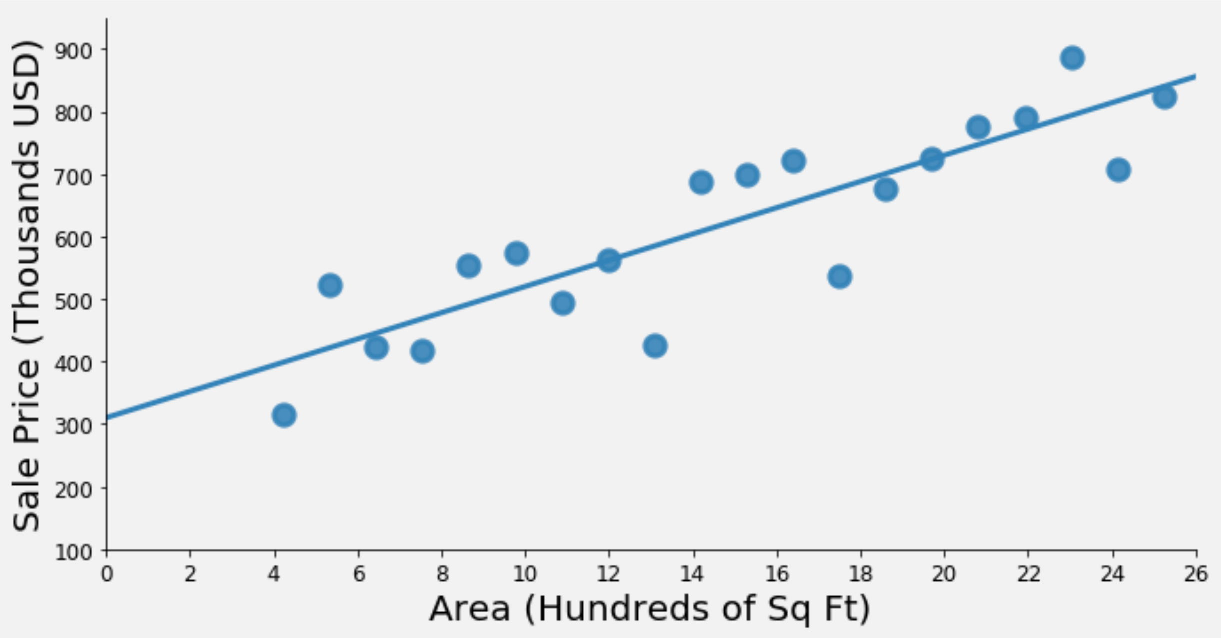
Construct the design matrix $\mathbf{X}$ by prepending column of 1s to data matrix

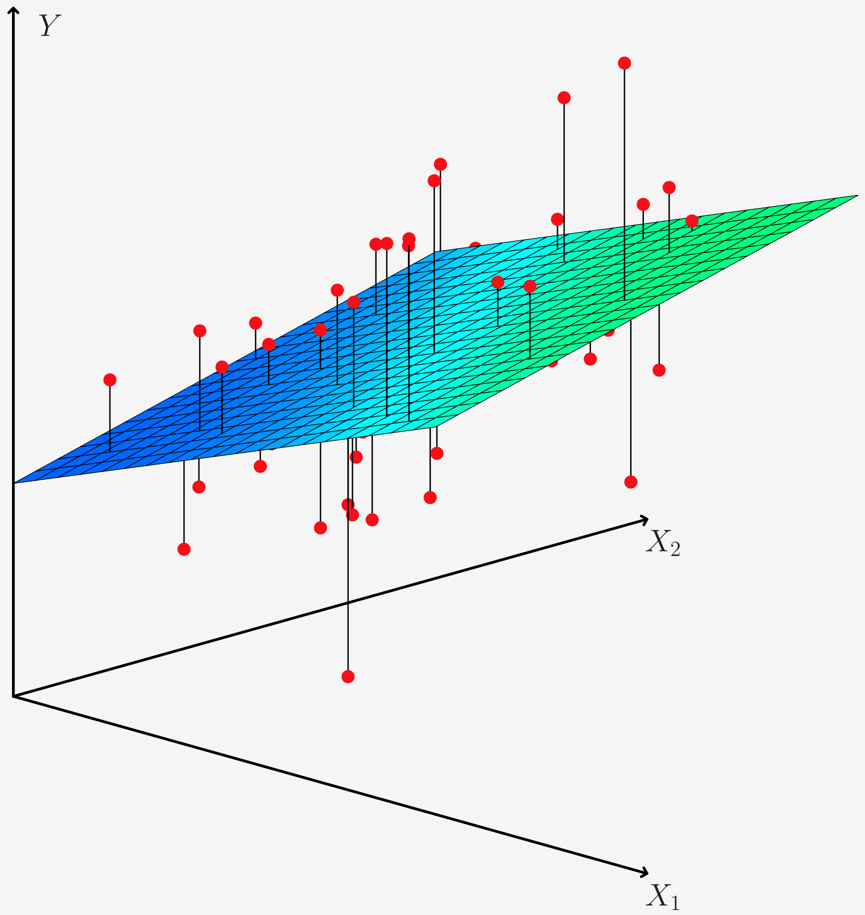
For the moment, solve via normal equations: $\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$

So, We Can Model Things Like This

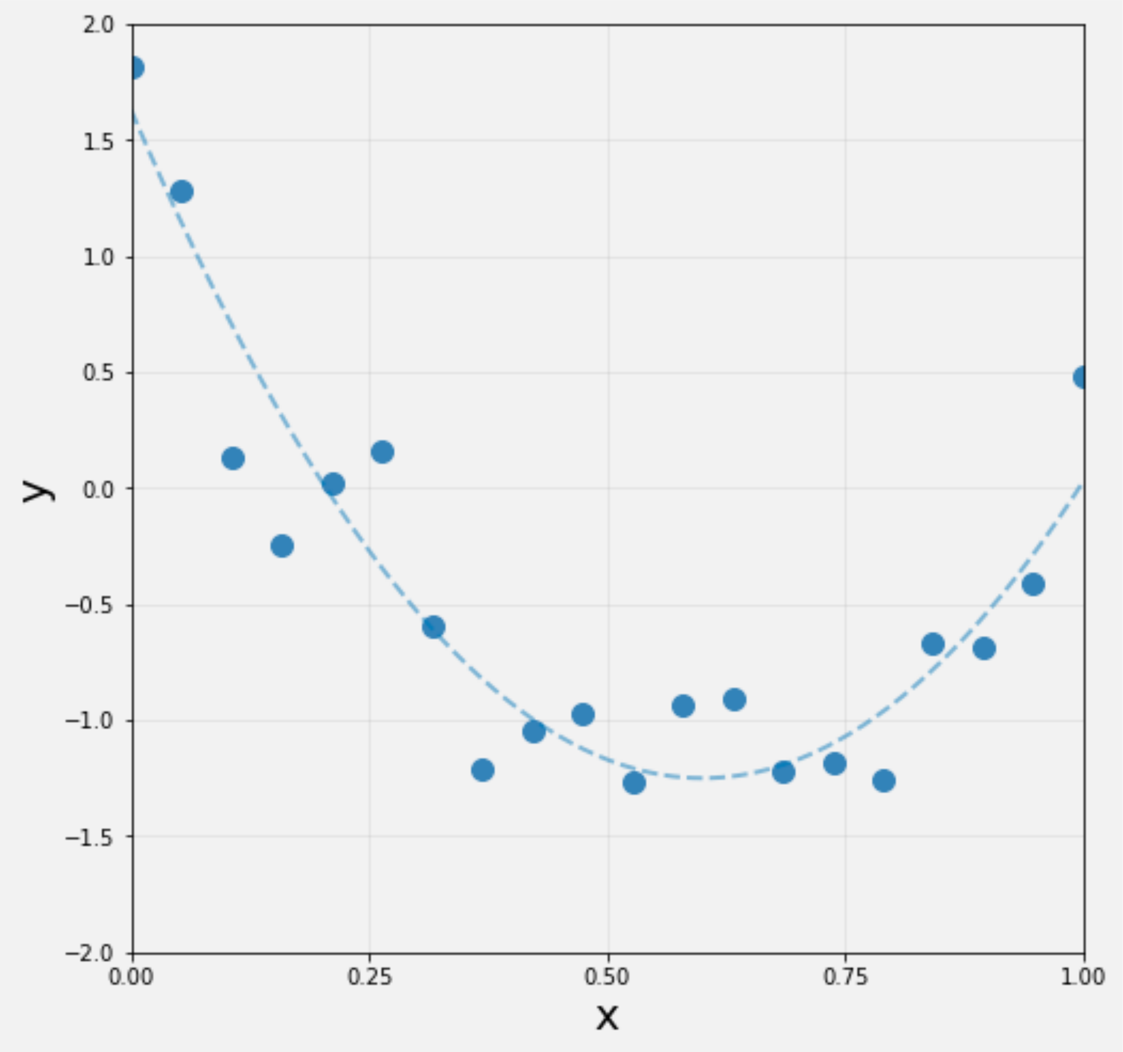


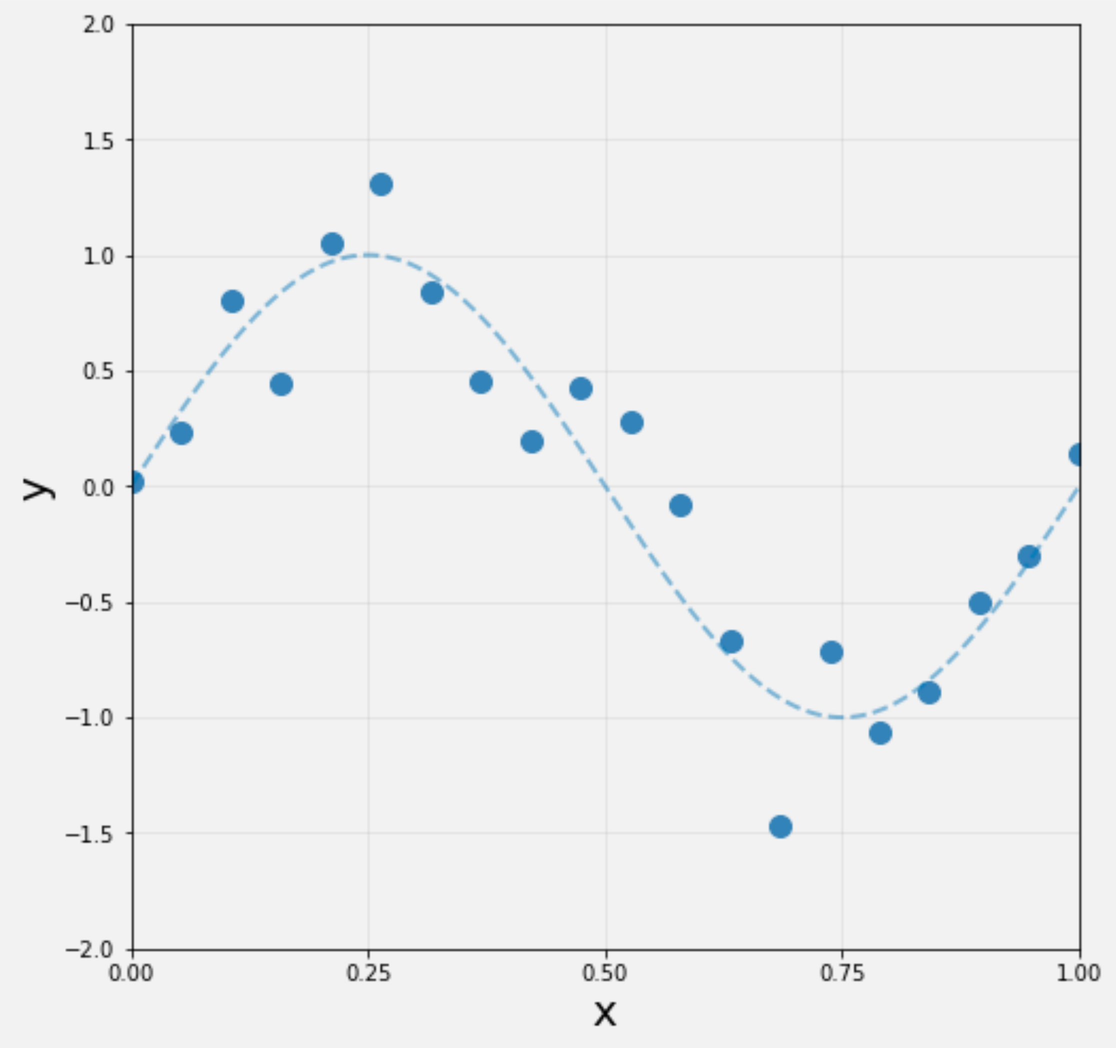
And Things Like This



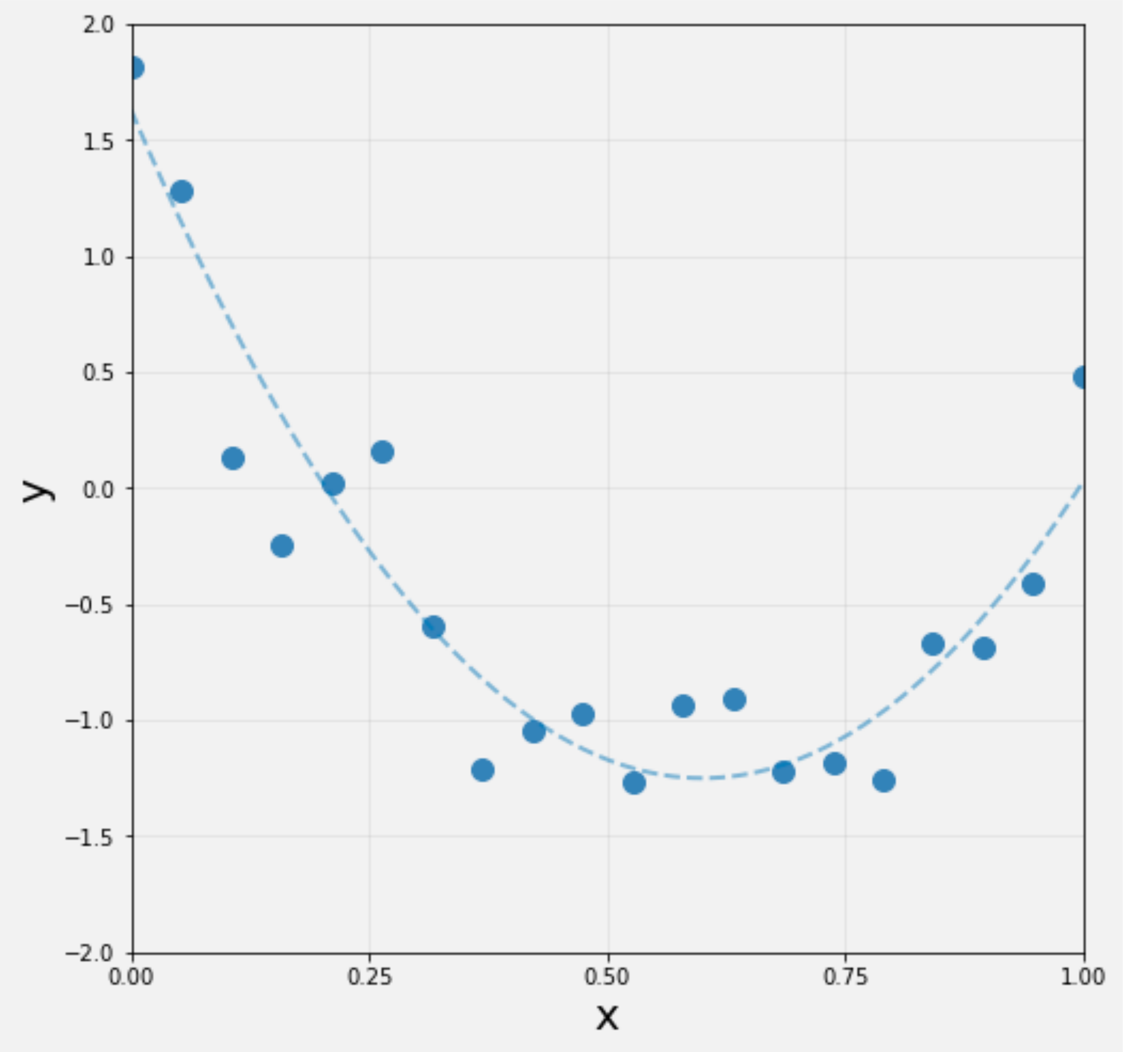


But What About Things Like This?



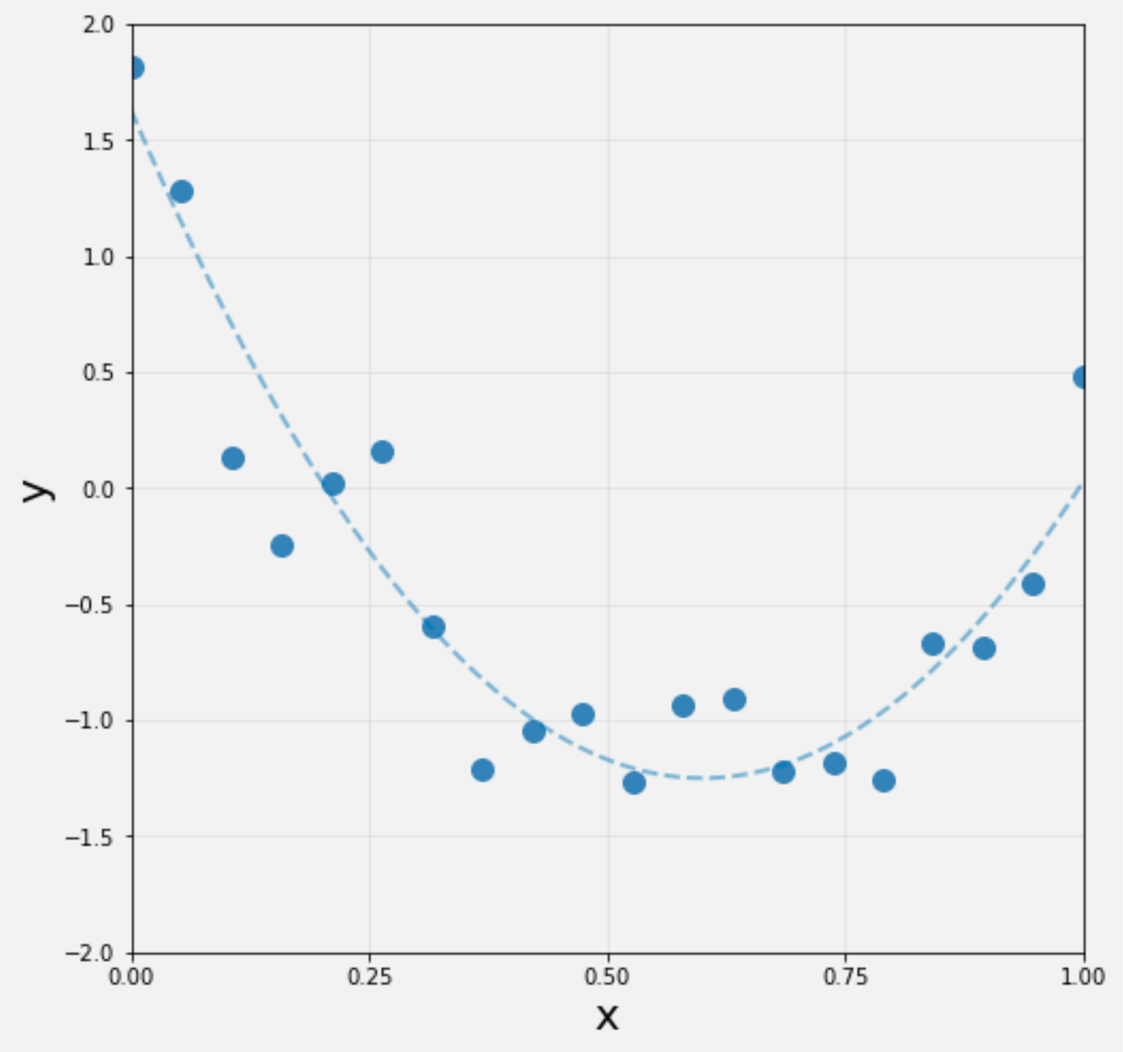


But What About Things Like This?



- So yeah, nonlinearity is a thing
- Clearly, the linear models of regression cannot handle this.
- Give me Neural Networks or Give me Death!

But What About Things Like This?



- So yeah, nonlinearity is a thing
- Clearly, the linear models of regression cannot handle this.
- Give me Neural Networks or Give me Death!
- Nah. Regression can totally do this

Start with the Obvious: Polynomials

- Suppose, as in the previous pictures, we have a single feature X
- We want to go from this: $Y = \beta_0 + \beta_1 X + \epsilon$
- To something like this: $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_p X^p + \epsilon$
- So what is the difference between these two?

Start with the Obvious: Polynomials

- Suppose, as in the previous pictures, we have a single feature X
- We want to go from this: $Y = \beta_0 + \beta_1 X + \epsilon$
- To something like this: $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_p X^p + \epsilon$
- So what is the difference between these two?
- Seems like it's closer to this: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$

The One Becomes the Many

- Start with a single feature X
- And derive new polynomial features: $X_1 = X, \quad X_2 = X^2, \quad \cdots \quad X_p = X^p$
- These two things: $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_p X^p + \epsilon$
- are exactly the same: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$

The Polynomial Regression Pipeline

- Start with a single feature X
- Derive new polynomial features: $X_1 = X, \quad X_2 = X^2, \quad \cdots \quad X_p = X^p$
- Solve the MLR in the usual way: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$
- **Question**: What does the Matrix Equation look like?

The Polynomial Regression Pipeline

- Start with a single feature X

- Derive new polynomial features: $X_1 = X, \quad X_2 = X^2, \quad \cdots \quad X_p = X^p$

- Solve the MLR in the usual way: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$

- **Question**: What does the Matrix Equation look like?

Before:

$$\begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ 1 & x_{31} & x_{32} & \cdots & x_{3p} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

The Polynomial Regression Pipeline

- Start with a single feature X
- Derive new polynomial features: $X_1 = X, \quad X_2 = X^2, \quad \cdots \quad X_p = X^p$
- Solve the MLR in the usual way: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$
- **Question**: What does the Matrix Equation look like?

After:
$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^p \\ 1 & x_2 & x_2^2 & \cdots & x_2^p \\ 1 & x_3 & x_3^2 & \cdots & x_3^p \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^p \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

The Polynomial Regression Pipeline

- Suppose I learn the parameters from the training set

- What if I want to make a prediction about data I haven't seen yet?

HAVE TEST point x_0

NEED to PERFORM SAME TRANSFORMATION on x_0 THAT WE PERFORMED on training data.

$$x_0 \to (1, x_0, x_0^2, \ldots, x_0^p)$$

THEN prediction is just $\hat{y}_0 = x_0^T \hat{\beta}$

The Polynomial Regression Pipeline

- Suppose I learn the parameters from the training set

- What if I want to make a prediction about data I haven't seen yet?

- **Important Theme #1:** If you transform your features for training, predicting is always as easy transforming the features of your new data in the same way.

	Input space		Feature space
Training	x	$\rightarrow$	$x, x^2, \ldots x^p$
Test	x_0	$\rightarrow$	$x_0, x_0^2, \ldots, x_0^p$

MAGIC HAPPENS HERE!

The Polynomial Regression Pipeline

- Suppose I learn the parameters from the training set

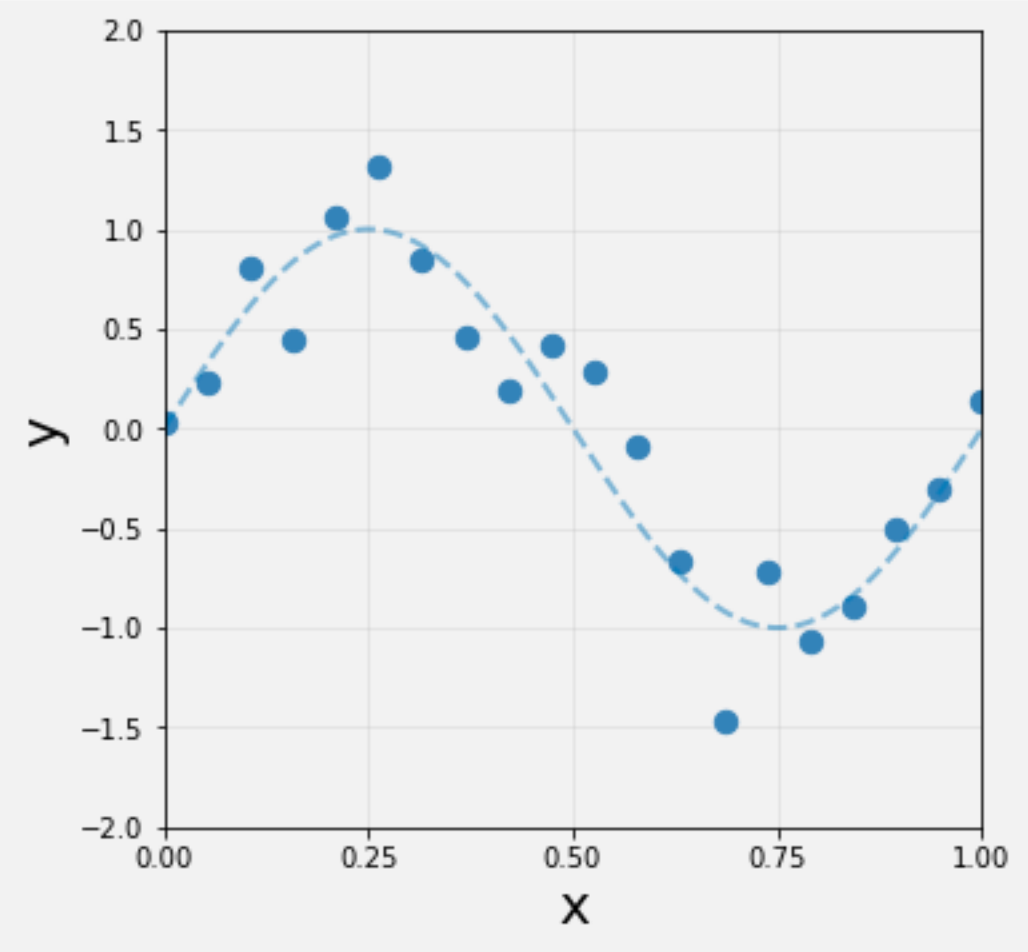
- What if I want to make a prediction about data I haven't seen yet?

- **Important Theme #1:** If you transform your features for training, predicting is always as easy transforming the features of your new data in the same way.

- **Important Theme #2:** If your current features are not flexible enough, moving into a higher dimensional space will make your model more flexible (but BEWARE!)

Example: Sinusoidal Data

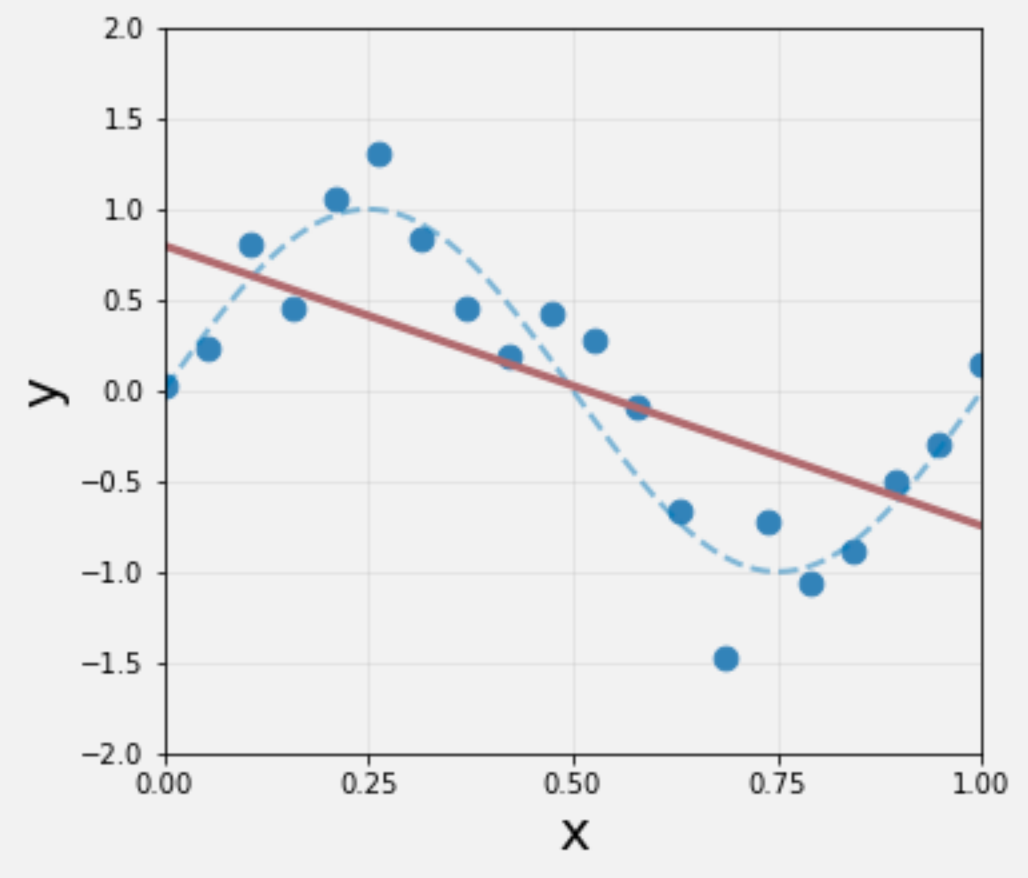
True Model is $f(x) = \sin(\pi x) + \epsilon$



- **Question:** What degree polynomial should I use?

Example: Sinusoidal Data

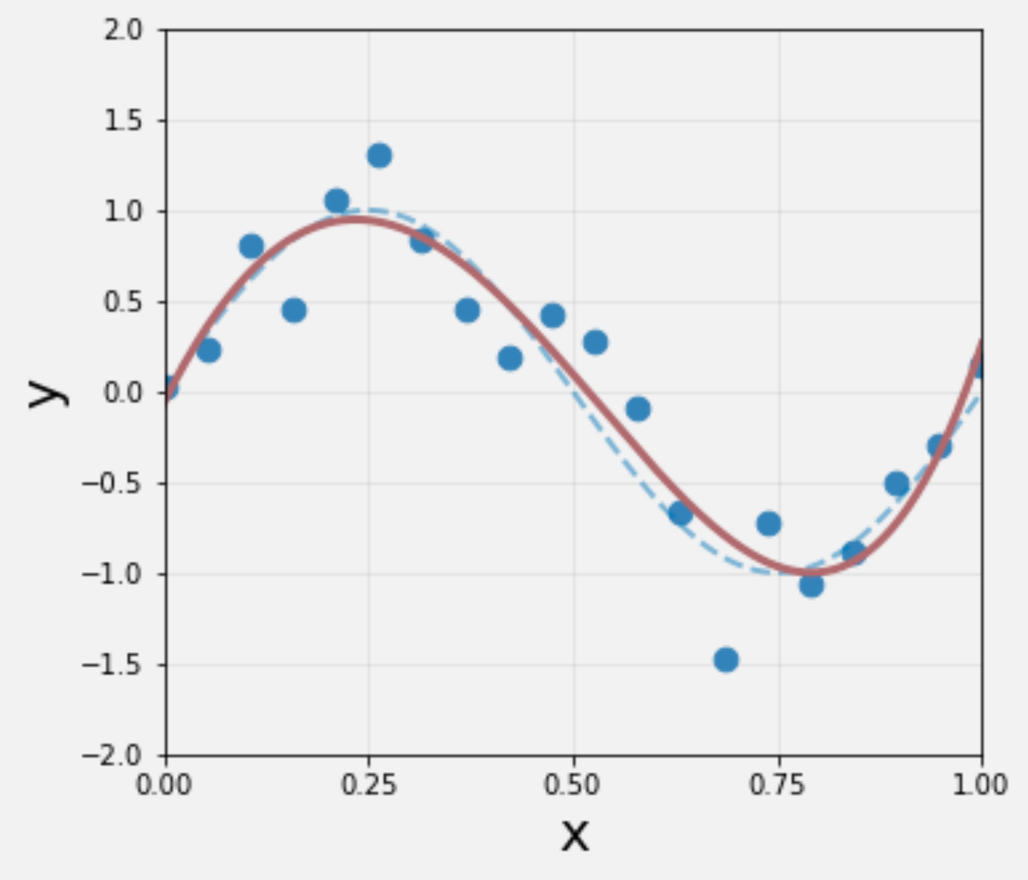
True Model is $f(x) = \sin(\pi x) + \epsilon$



- **Question:** What degree polynomial should I use?
- Degree = 1

Example: Sinusoidal Data

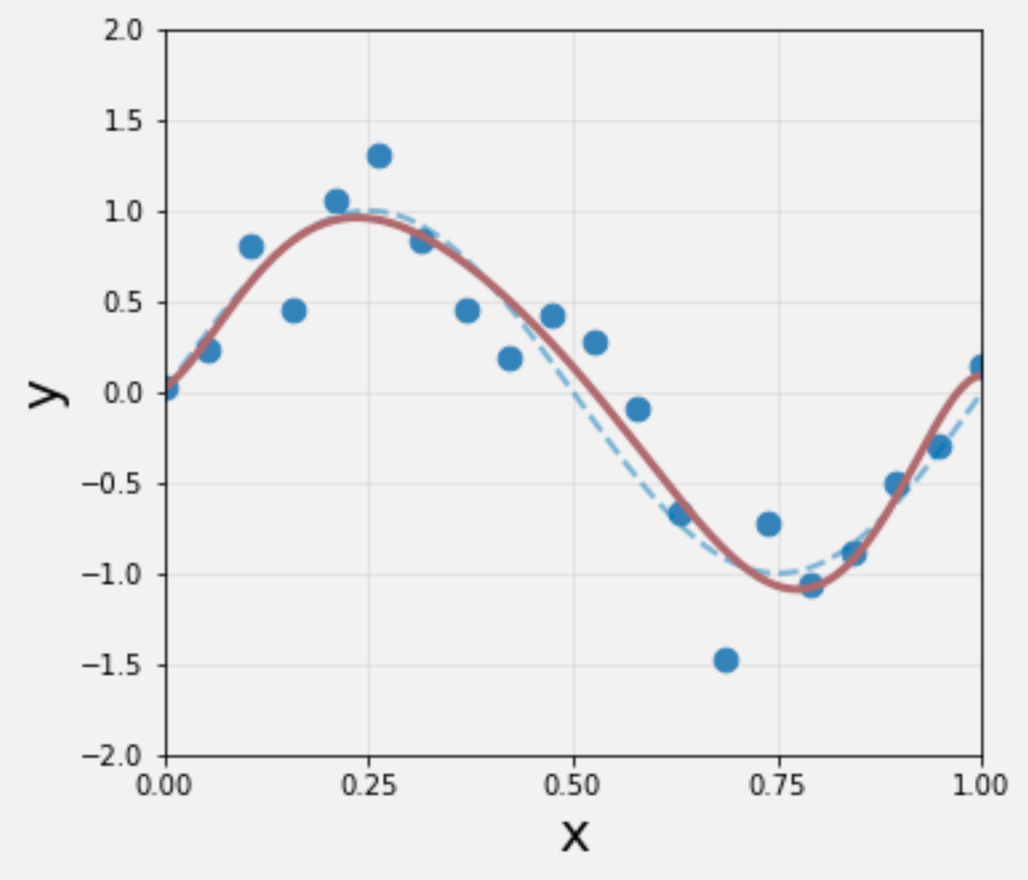
True Model is $f(x) = \sin(\pi x) + \epsilon$



- **Question**: What degree polynomial should I use?
- Degree = 4

Example: Sinusoidal Data

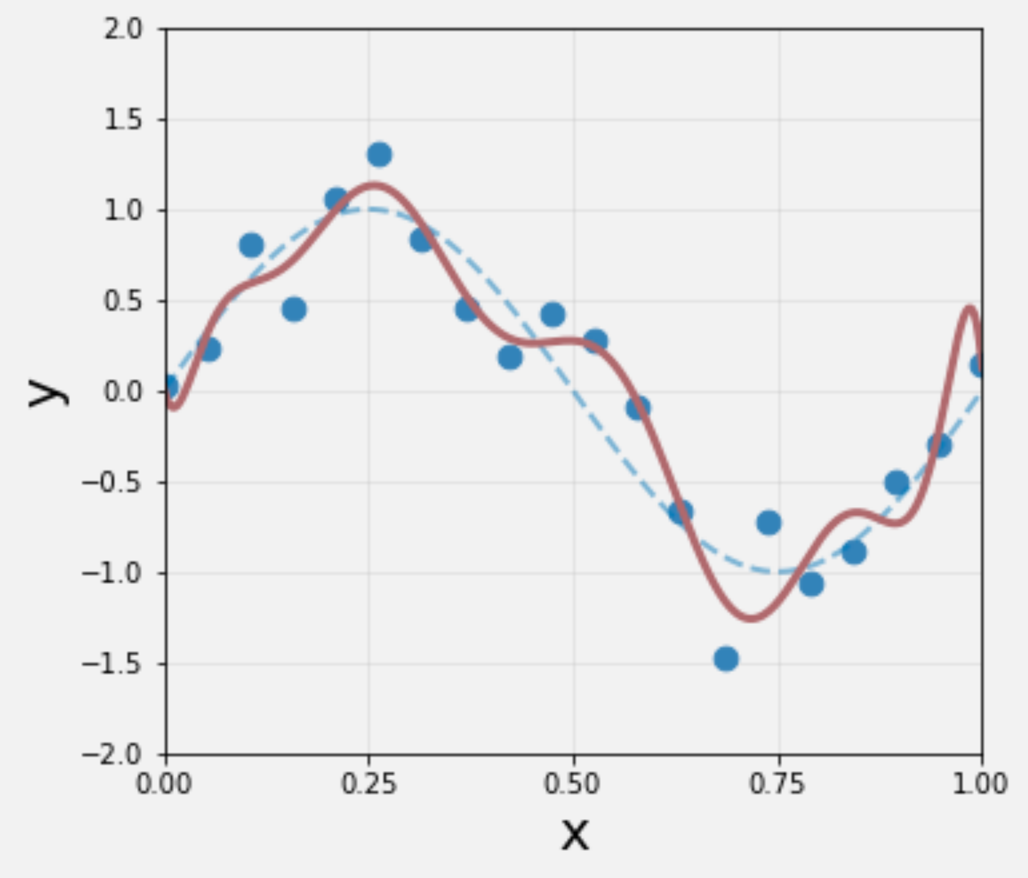
True Model is $f(x) = \sin(\pi x) + \epsilon$



- **Question**: What degree polynomial should I use?
- Degree = 7

Example: Sinusoidal Data

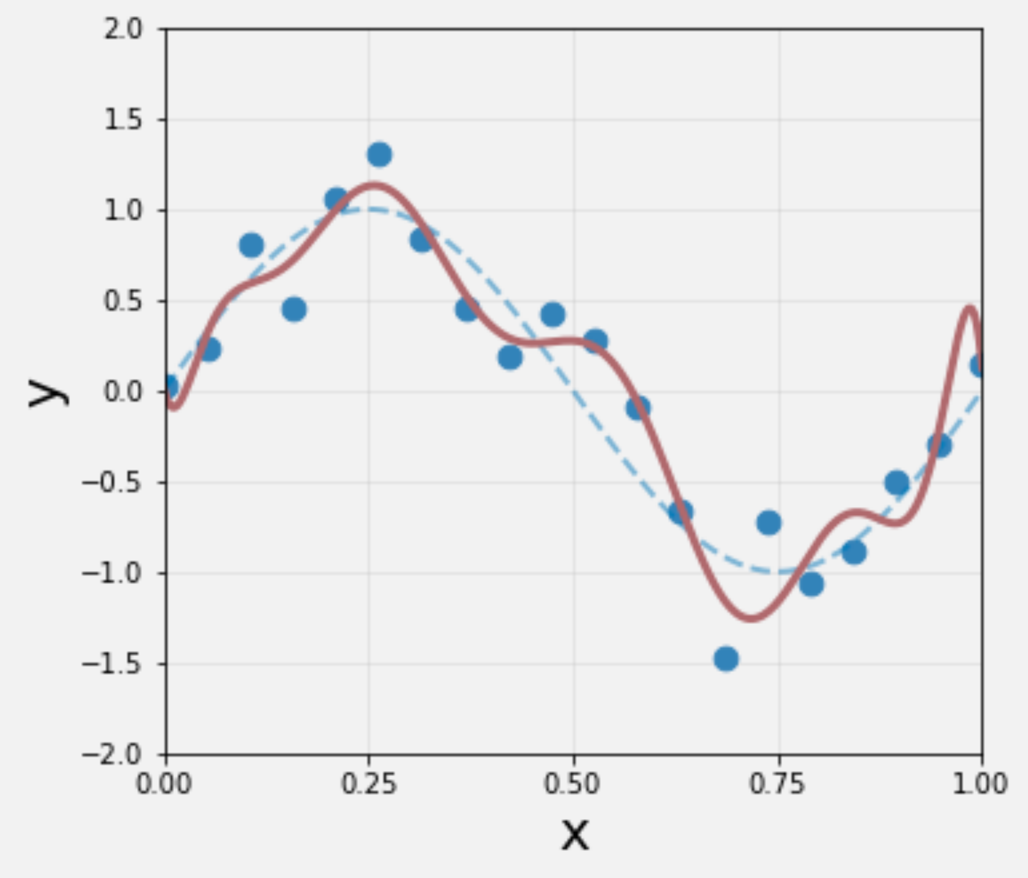
True Model is $f(x) = \sin(\pi x) + \epsilon$



- **Question**: What degree polynomial should I use?
- Degree = 11

Example: Sinusoidal Data

True Model is $f(x) = \sin(\pi x) + \epsilon$



- **Question**: What degree polynomial should I use?
- Degree = 11
- What Questions should we be asking?

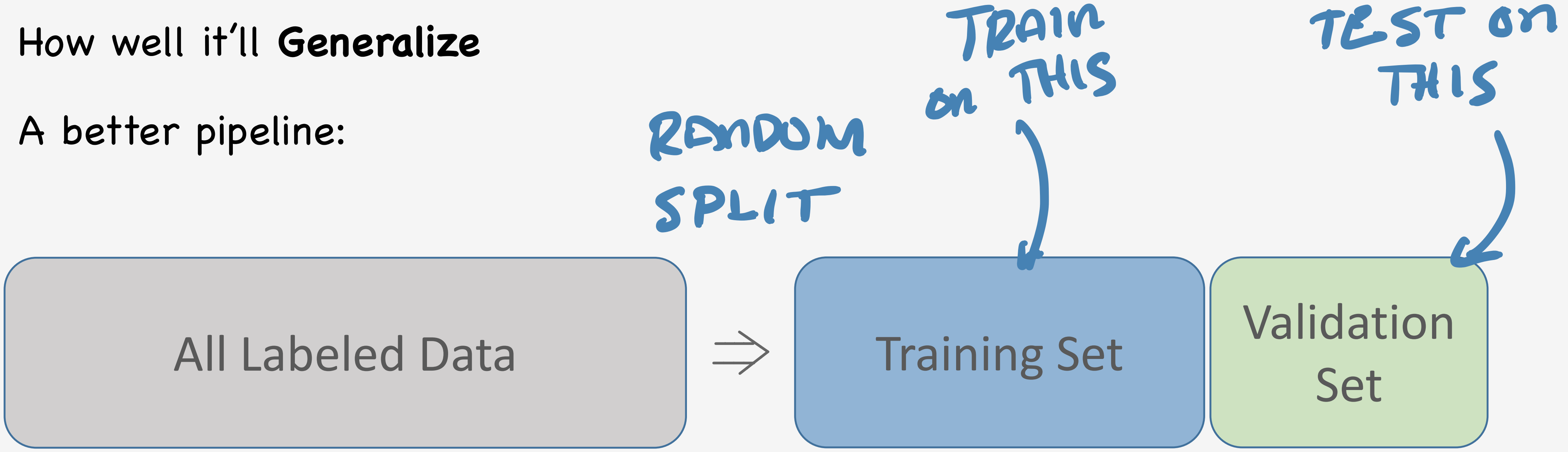
The Real ML Pipeline (Almost)

The more flexible (powerful) your model gets, the better it'll do on training set

But that's not useful. We want to know how model will do on new data

How well it'll **Generalize**

A better pipeline:



Train on training set. **Validate** (or evaluate) on validation set.

HELD-OUT DATA

The Real ML Pipeline (Almost)

Need a **performance measure:**

For Regression, we use the Mean-Squared Error:

TRUE RESPONSE

PREDICTIONS

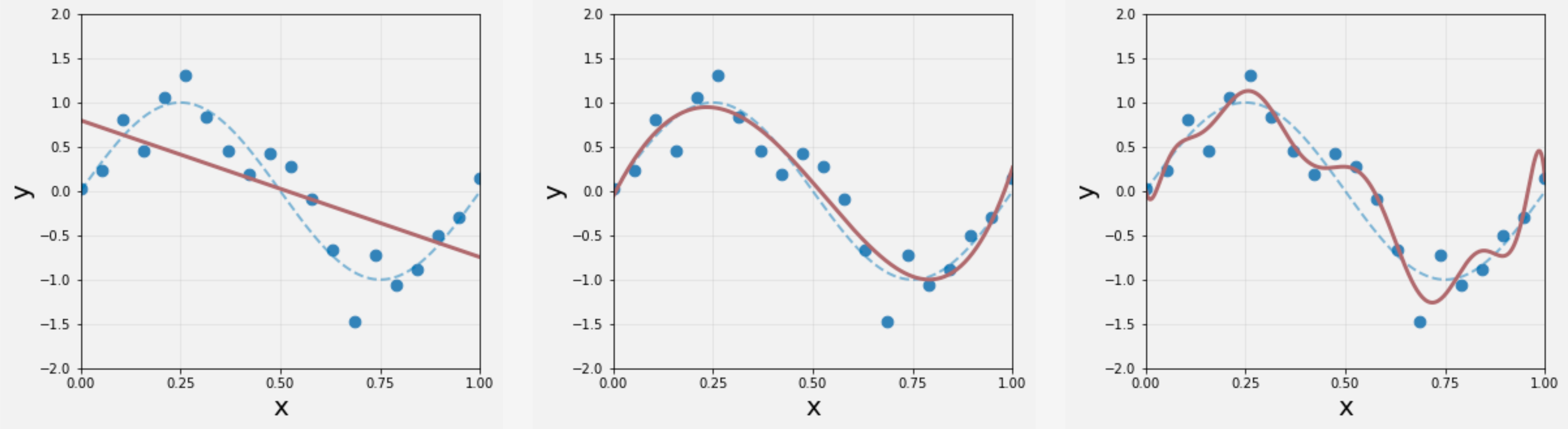
$$\mathrm{MSE} = \frac{1}{n}\sum_{I=1}^{n}(y_i - \hat{y}_i)^2$$

All Labeled Data $\Rightarrow$ Training Set | Validation Set

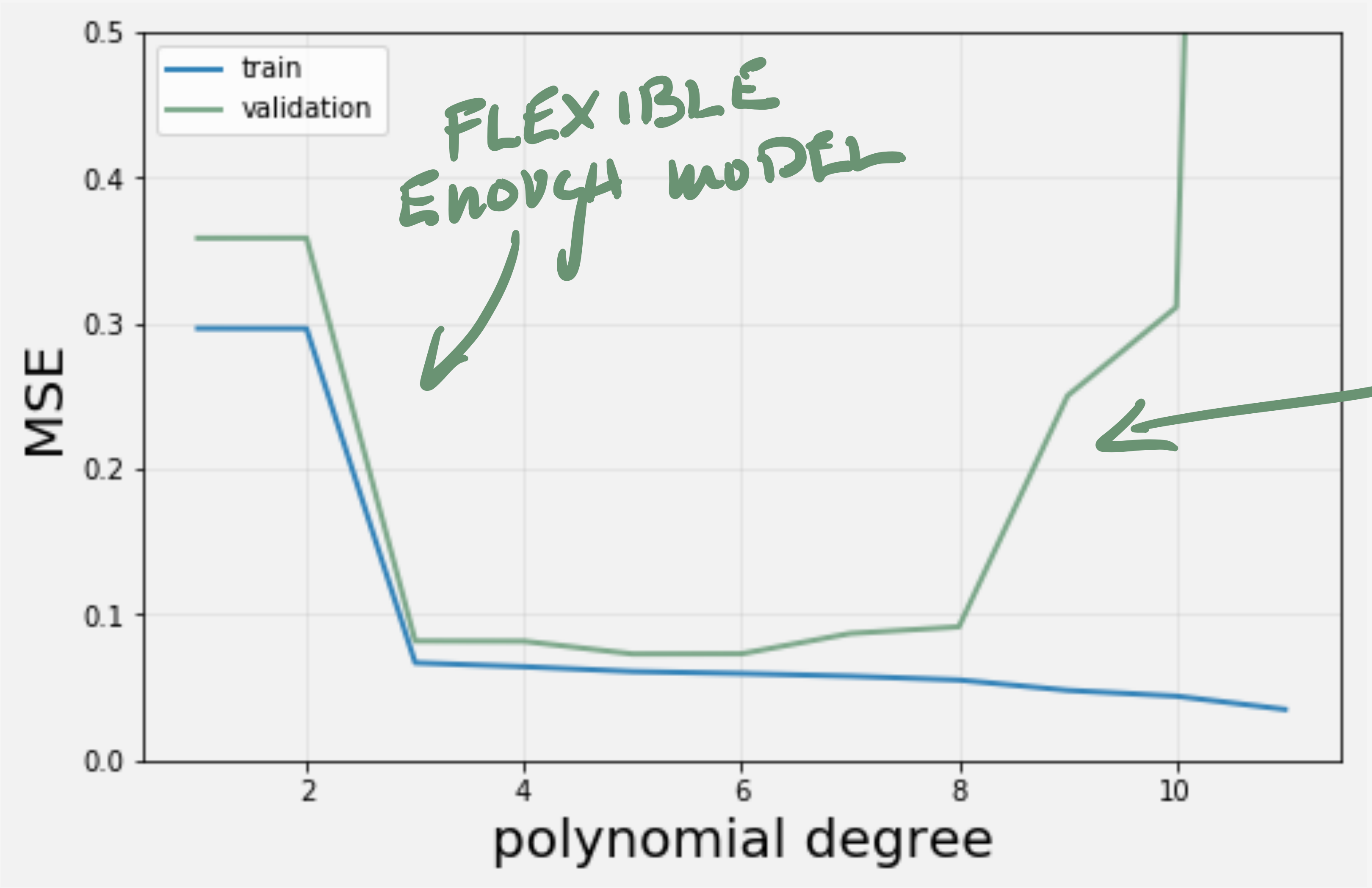
Example: Sinusoidal Data

What happens to the MSE on training set as p grows? MSE goes Down

What happens to MSE on validation set as p grows? MSE goes Down
AT FIRST, THEN BACK UP!



Example: Sinusoidal Data

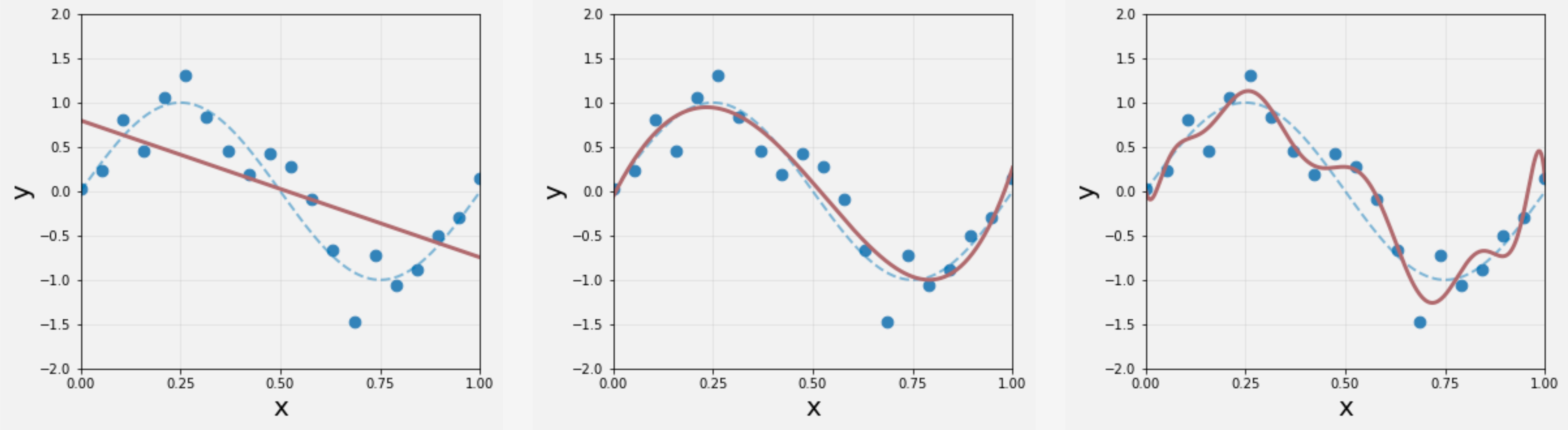


OVERFITTING HAS STARTED

Example: Sinusoidal Data

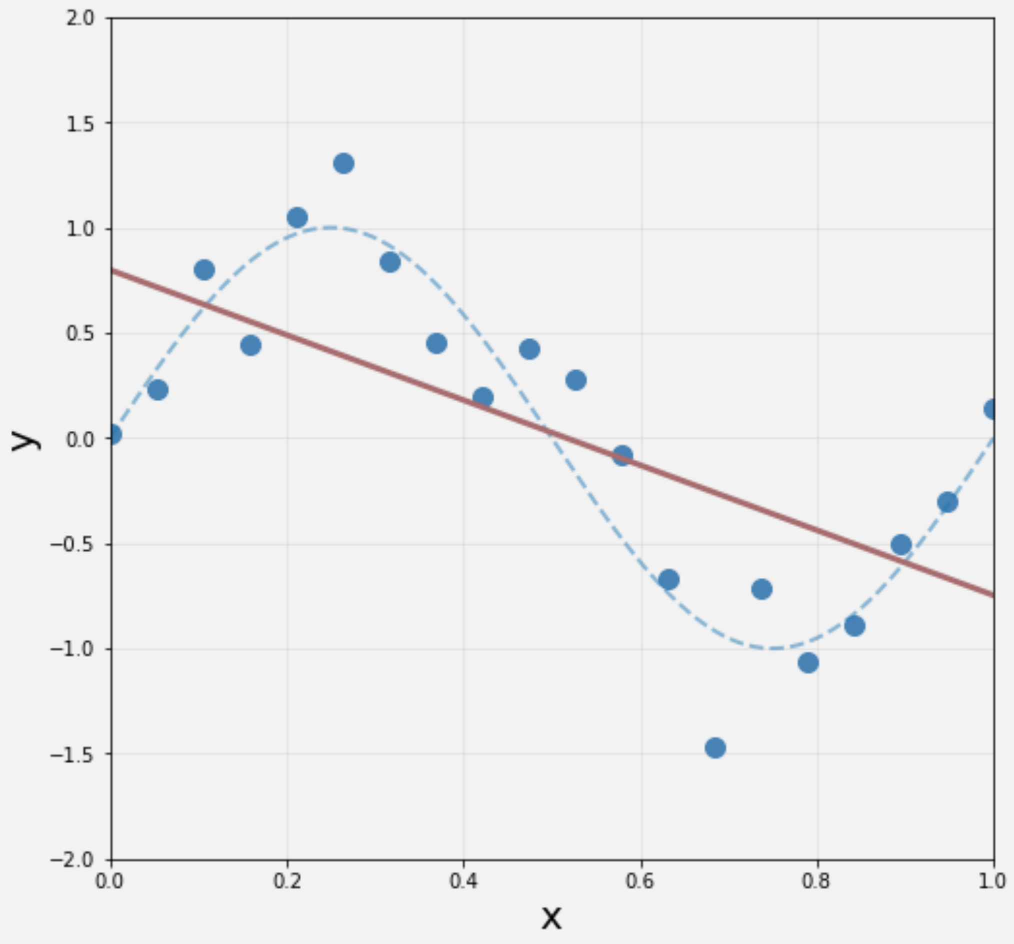
Revisit Polynomial Regression

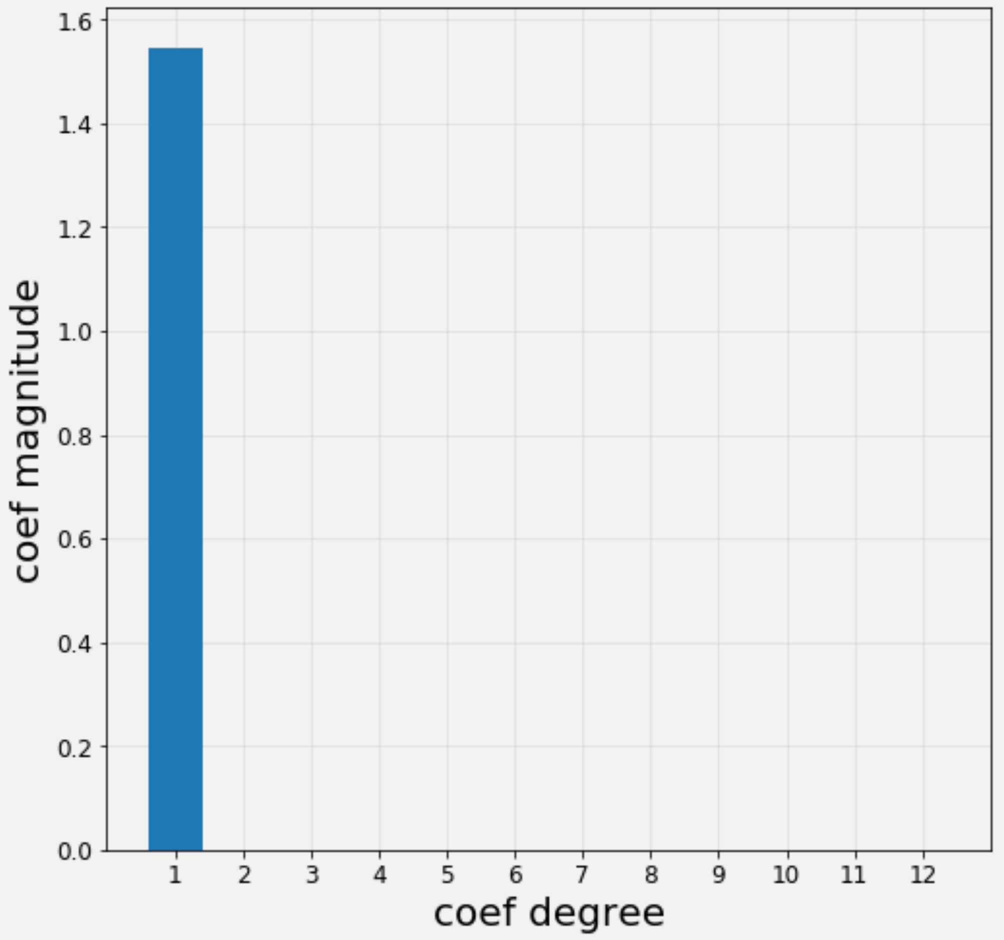
What causes those wiggles that are hurting us so bad?



Example: Sinusoidal Data

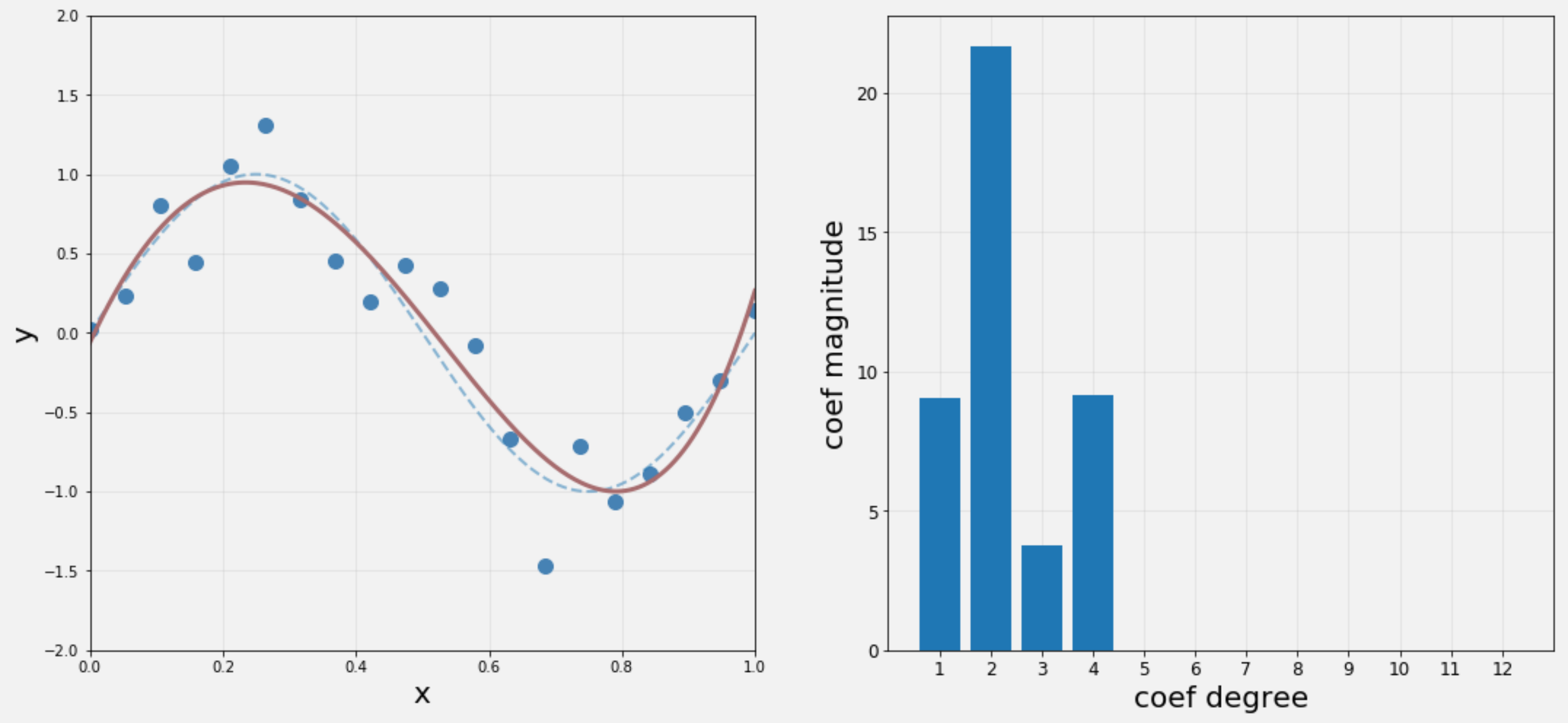
True Model is $f(x) = \sin(\pi x) + \epsilon$





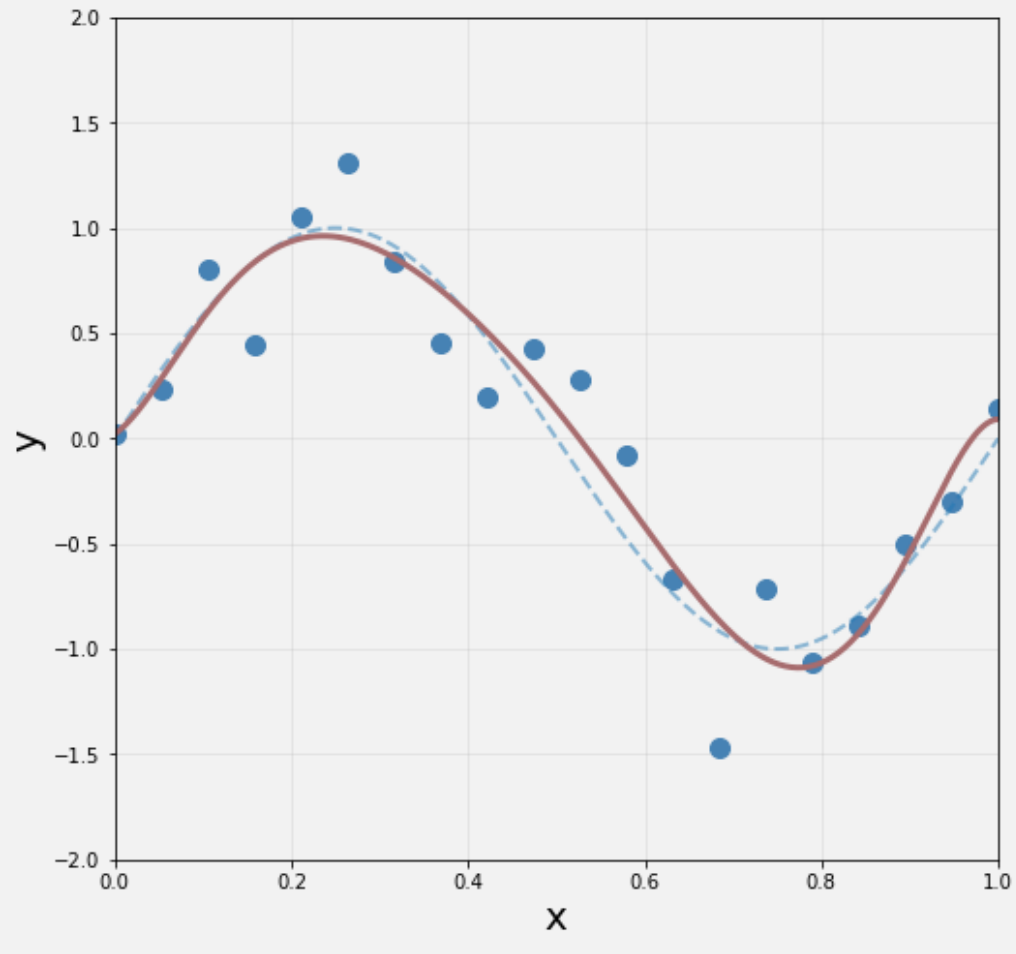
Example: Sinusoidal Data

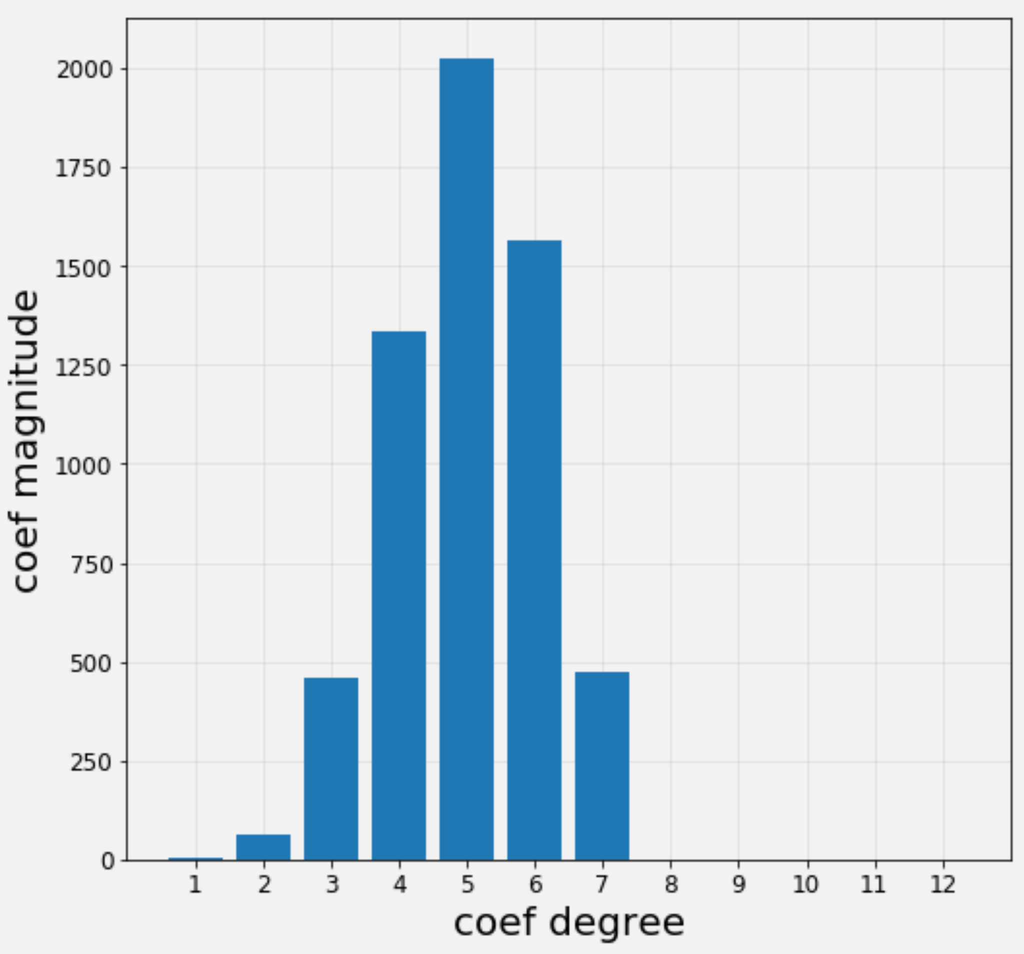
True Model is $f(x) = \sin(\pi x) + \epsilon$



Example: Sinusoidal Data

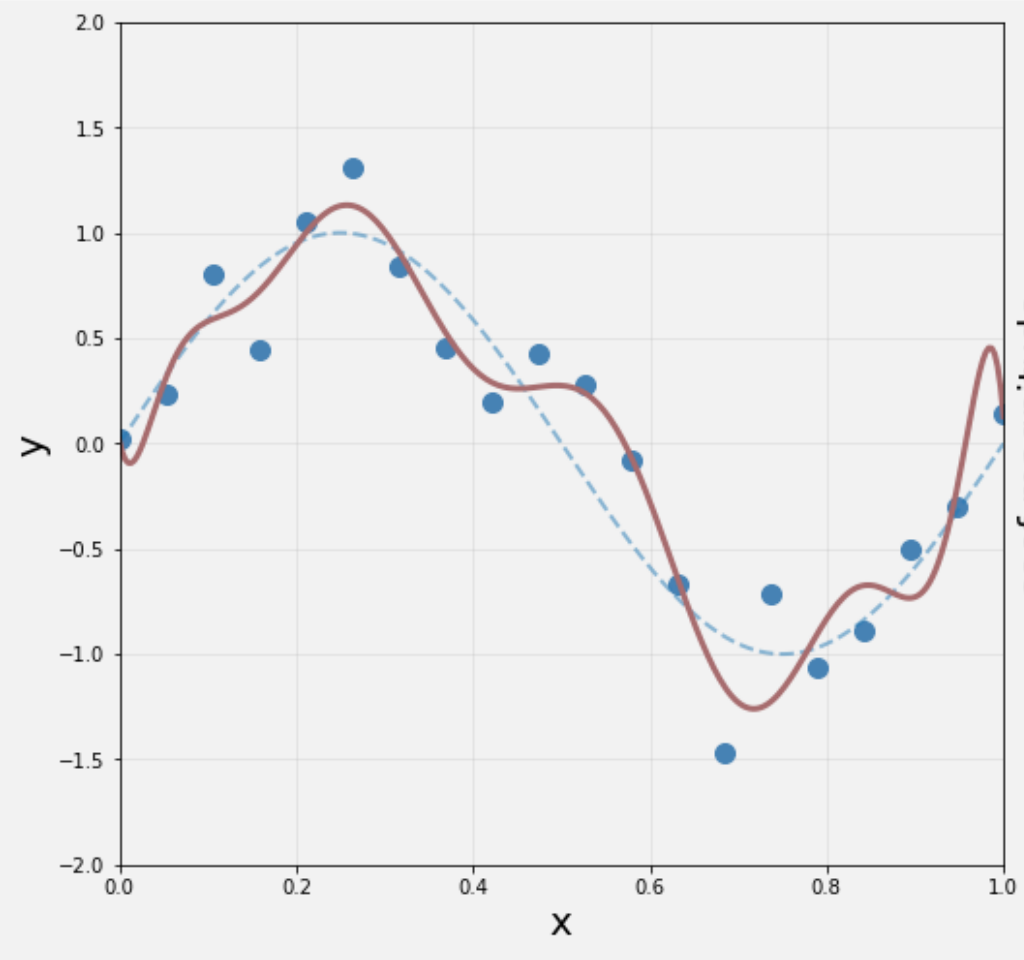
True Model is $f(x) = \sin(\pi x) + \epsilon$

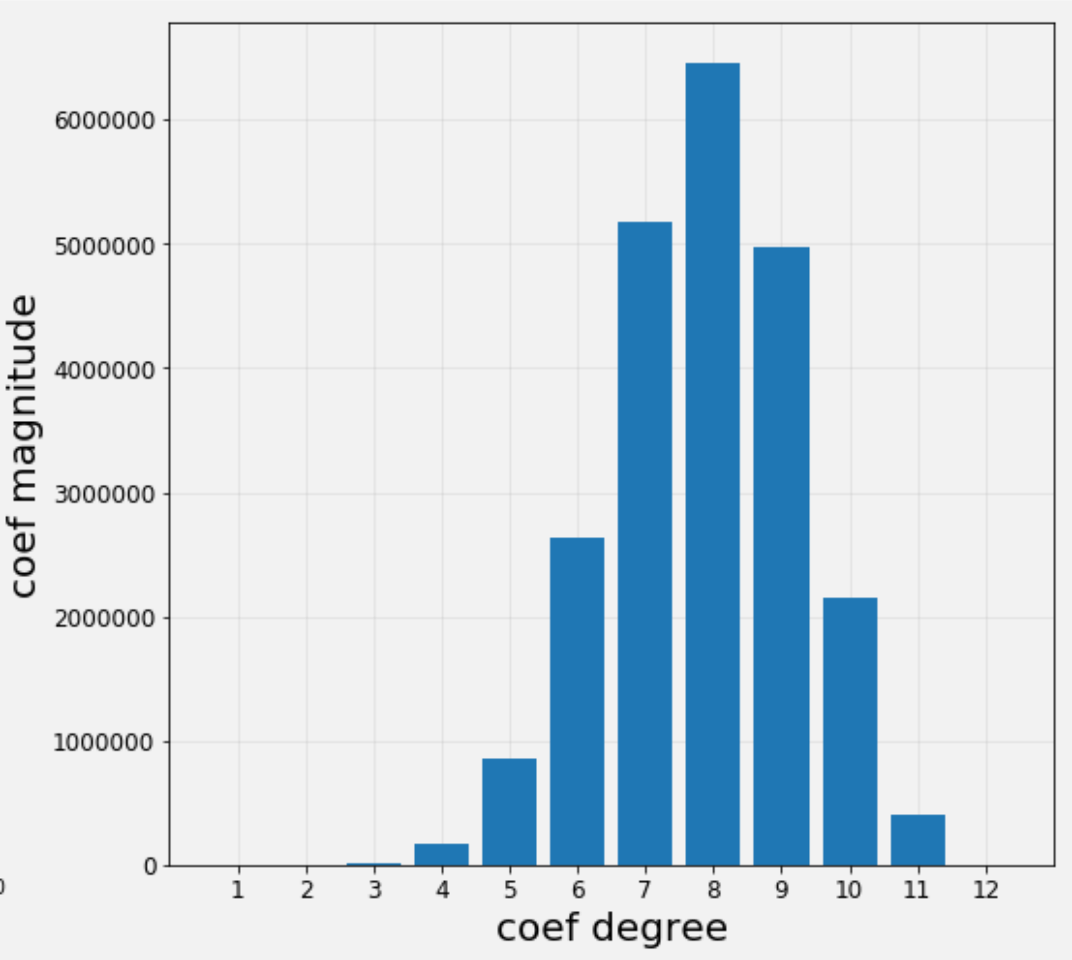




Example: Sinusoidal Data

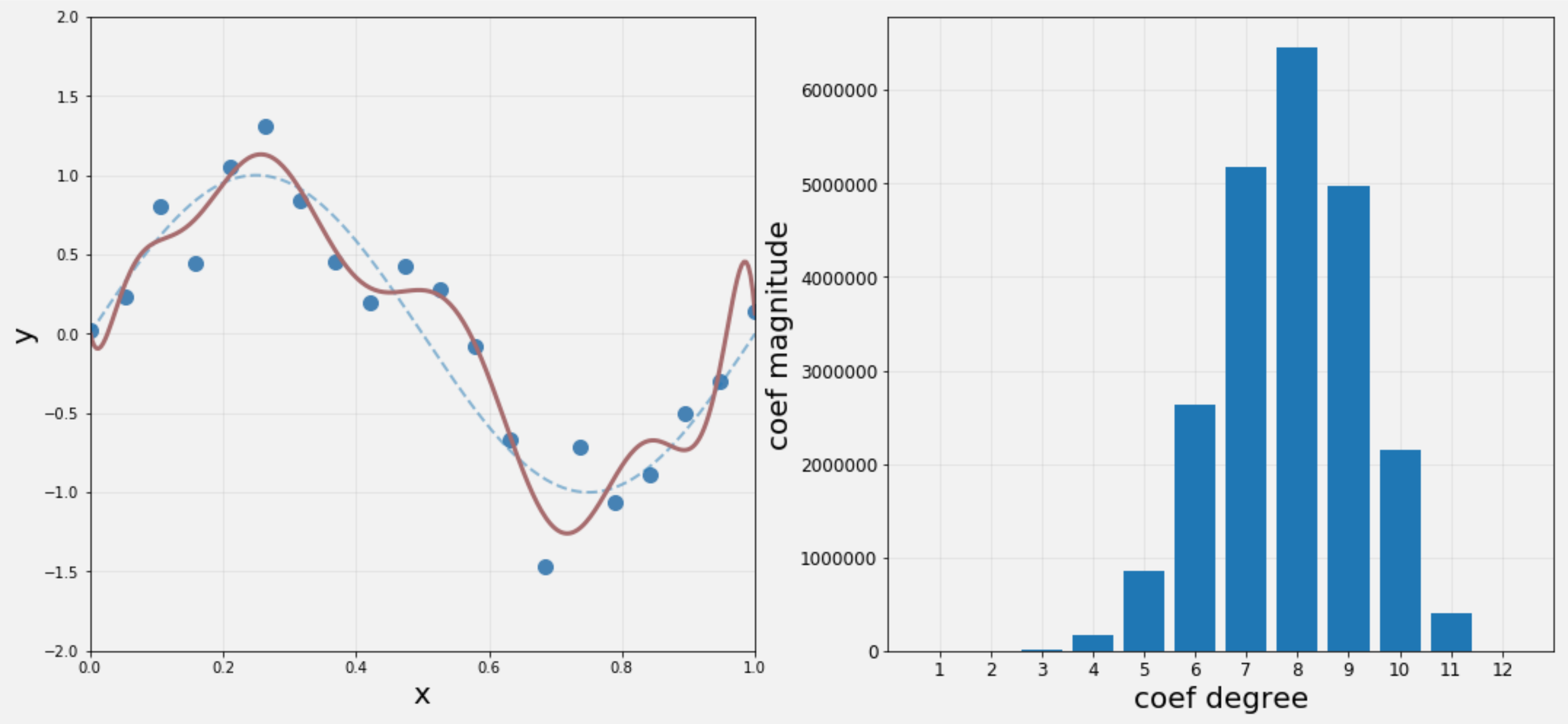
True Model is $f(x) = \sin(\pi x) + \epsilon$





Example: Sinusoidal Data

Coefficients grow **very** large. Need a way to control them. **Regulate** them even



Regularization

Goal: Keep model coefficients from growing too large

Idea: Put something in the loss function to dissuade them growing too much

$$\text{MINIMIZE} \quad RSS_\lambda = \underbrace{\sum_{i=1}^{n} [(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) - y_i]^2}_{\text{RSS}} + \underbrace{\lambda \sum_{k=1}^{p} \beta_k^2}_{\text{PENALTY}}$$

- Regularization weight λ is a tuning parameter
- Have to do regularization study to choose best value

Regularization

Goal: Keep model coefficients from growing too large

Idea: Put something in the loss function to dissuade them growing too much

$$RSS_\lambda = \sum_{i=1}^{n} [(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) - y_i]^2 + \lambda \sum_{k=1}^{p} \beta_k^2$$

- Why do we not regularize the bias parameter?

BIAS TELLS US ABOUT AVERAGE RESPONSE (HEIGHT OF DATA). REGULARIZING B_0 PULLS THE WHOLE MODEL DOWN FROM WHERE SHOULD BE

Ridge Regression

Goal: Keep model coefficients from growing too large

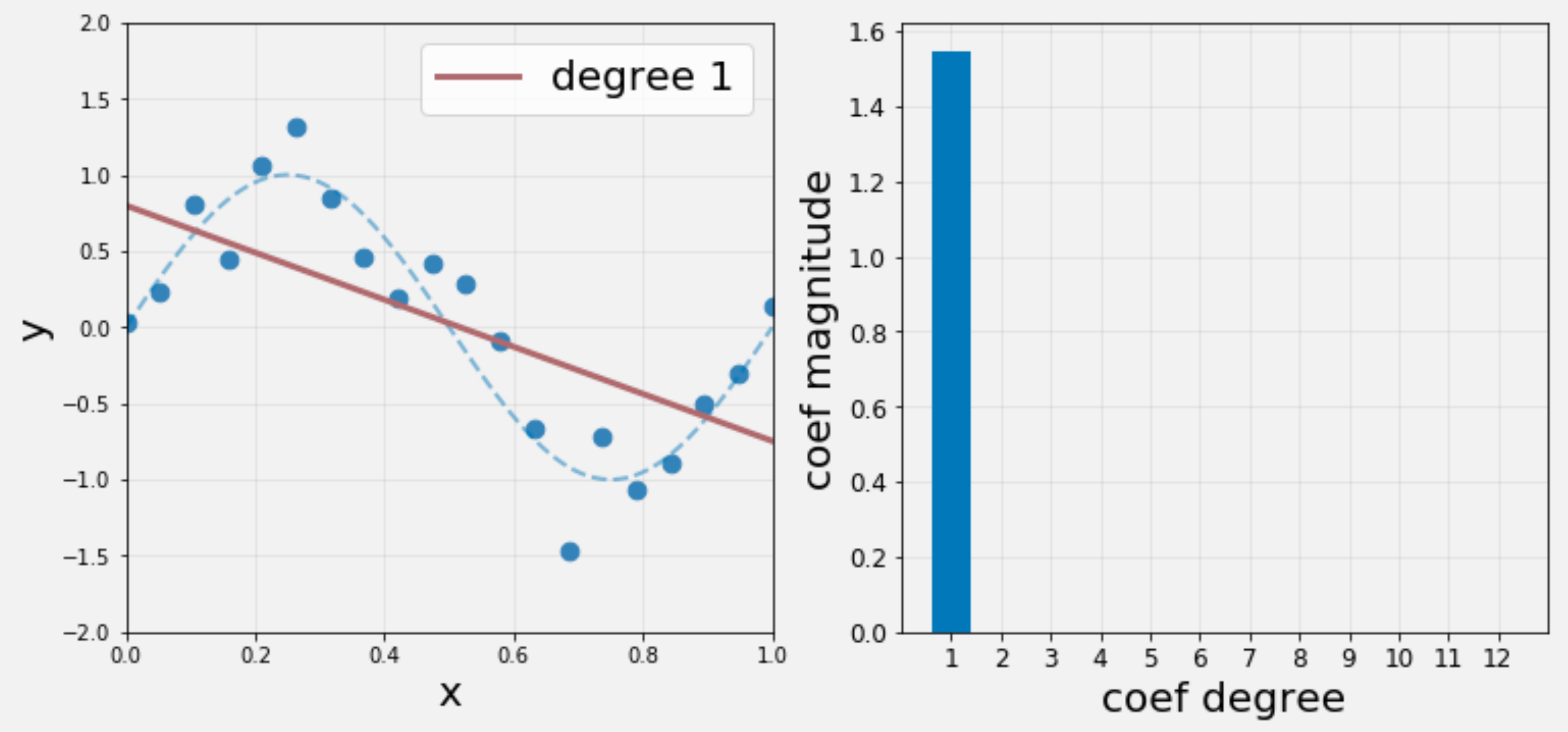
Idea: Put something in the loss function to dissuade them growing too much

$$RSS_\lambda = \sum_{i=1}^{n} \left[(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) - y_i\right]^2 + \lambda \sum_{k=1}^{p} \beta_k^2$$

- This form of penalty term is called Ridge Regression. But there are many more

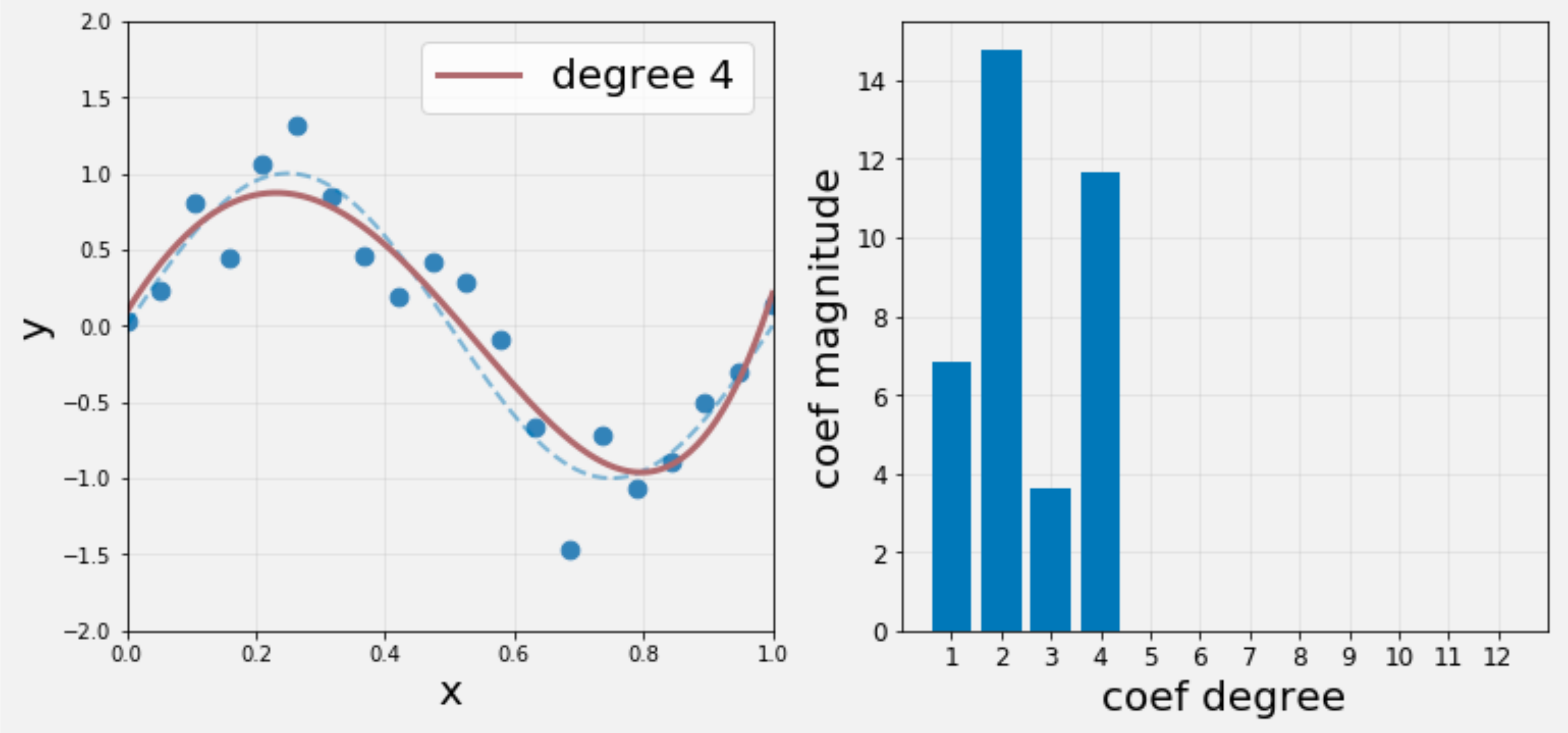
Sinusoidal Data with Ridge Regression

Watch the coefficients



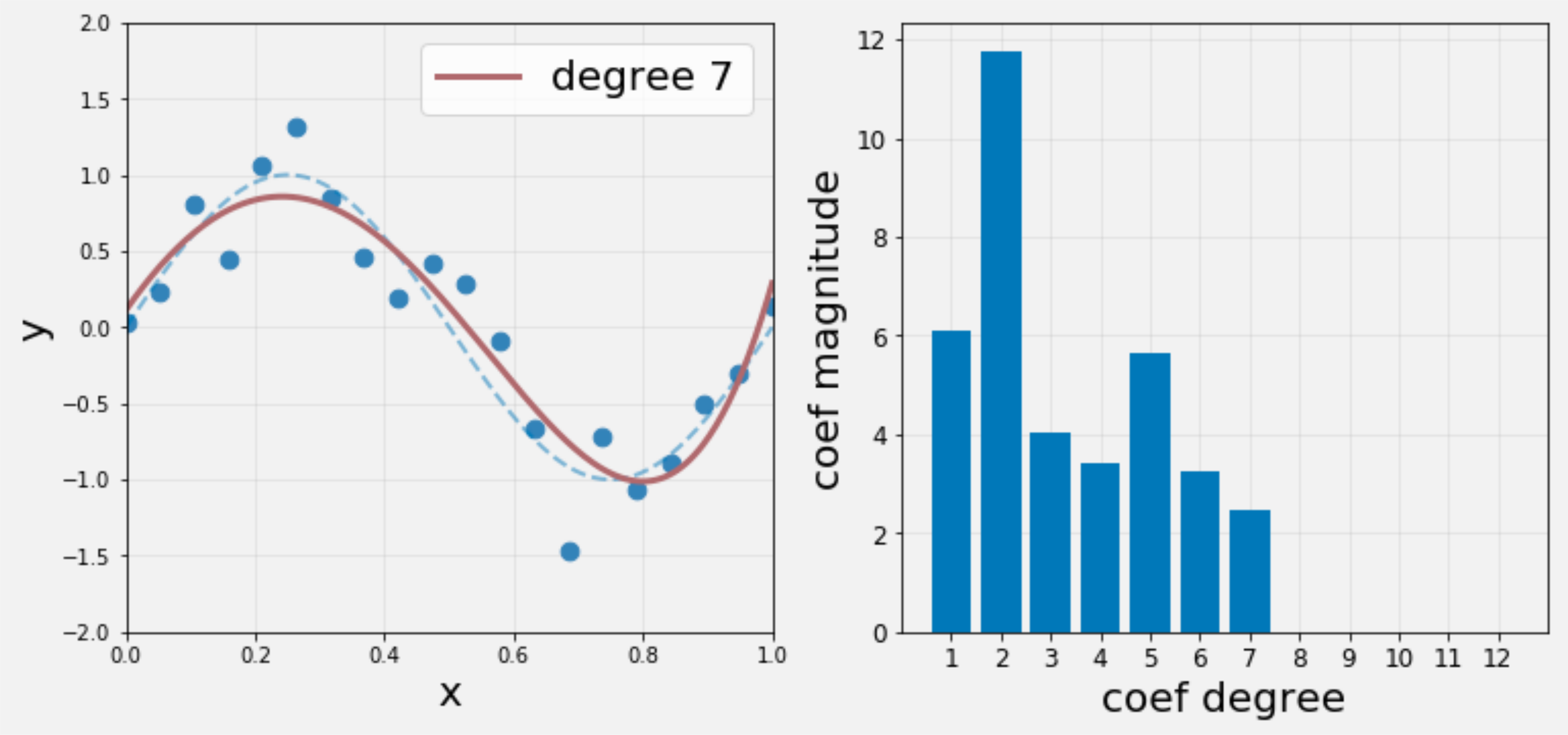
Sinusoidal Data with Ridge Regression

Watch the coefficients



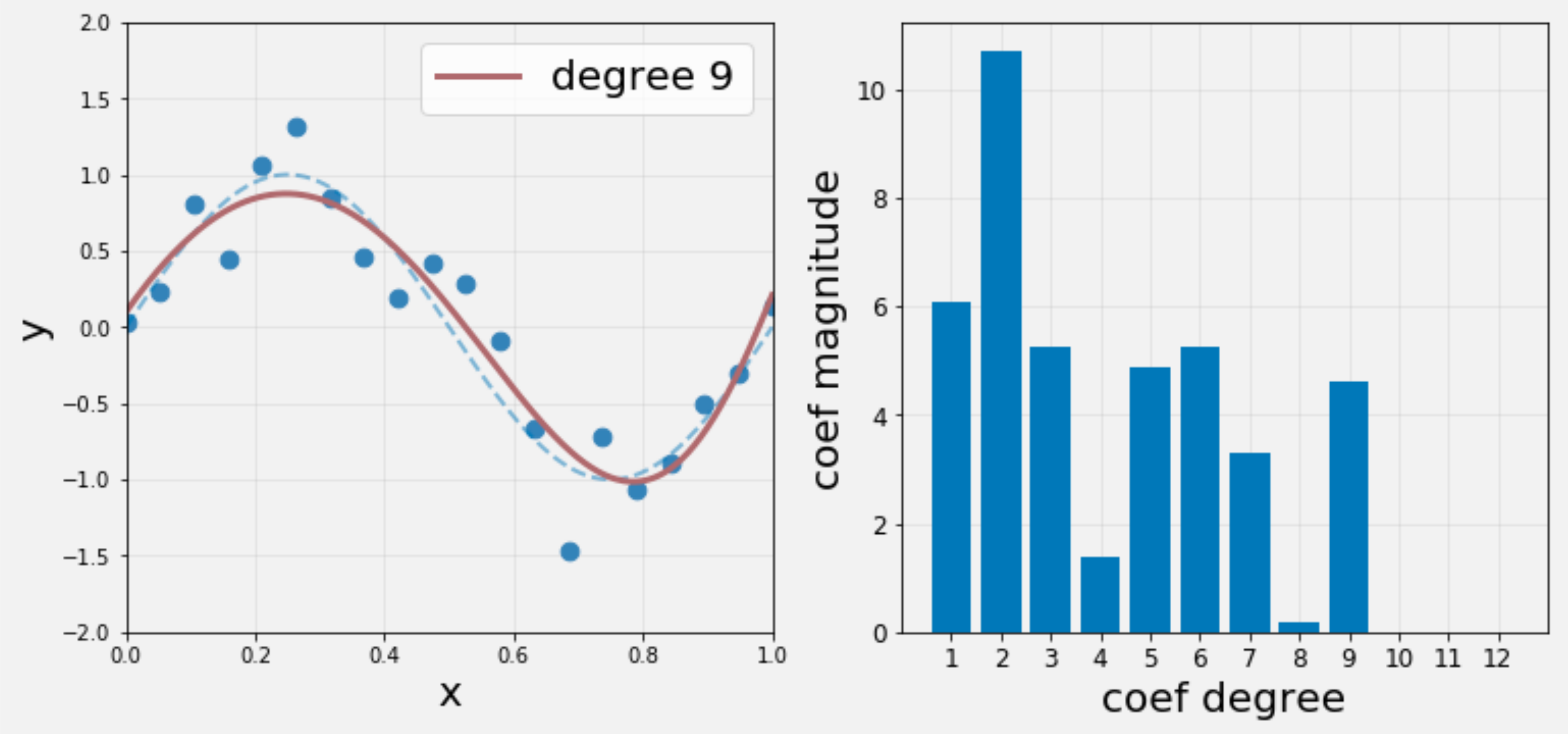
Sinusoidal Data with Ridge Regression

Watch the coefficients



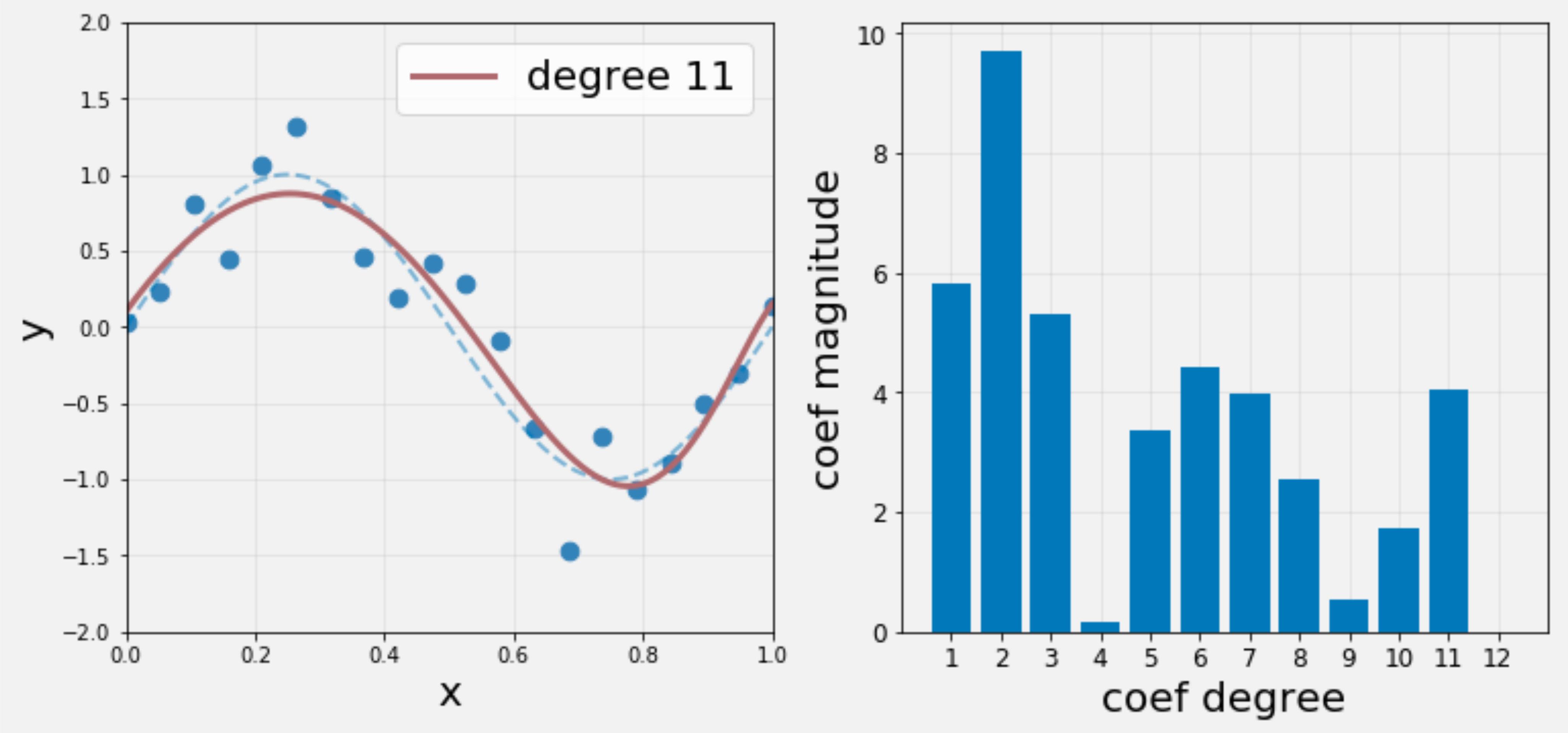
Sinusoidal Data with Ridge Regression

Watch the coefficients



Sinusoidal Data with Ridge Regression

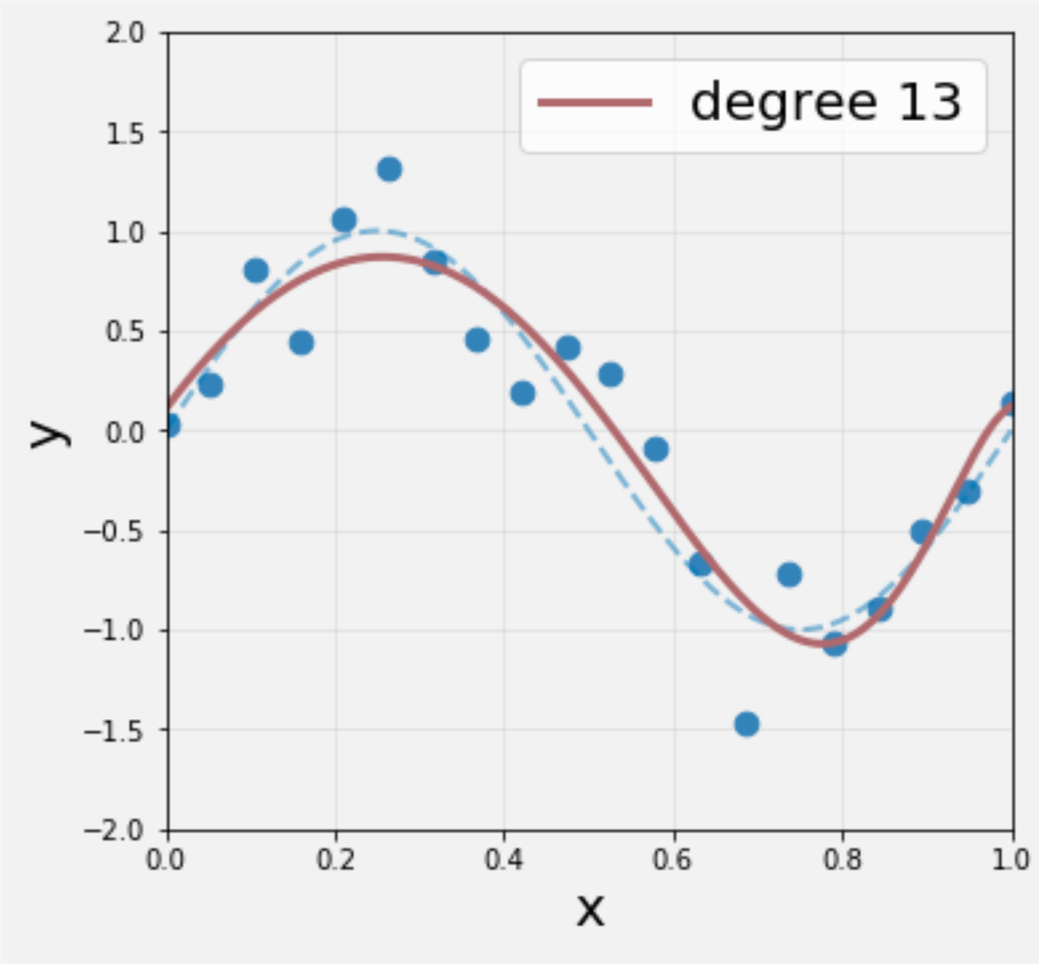
Watch the coefficients

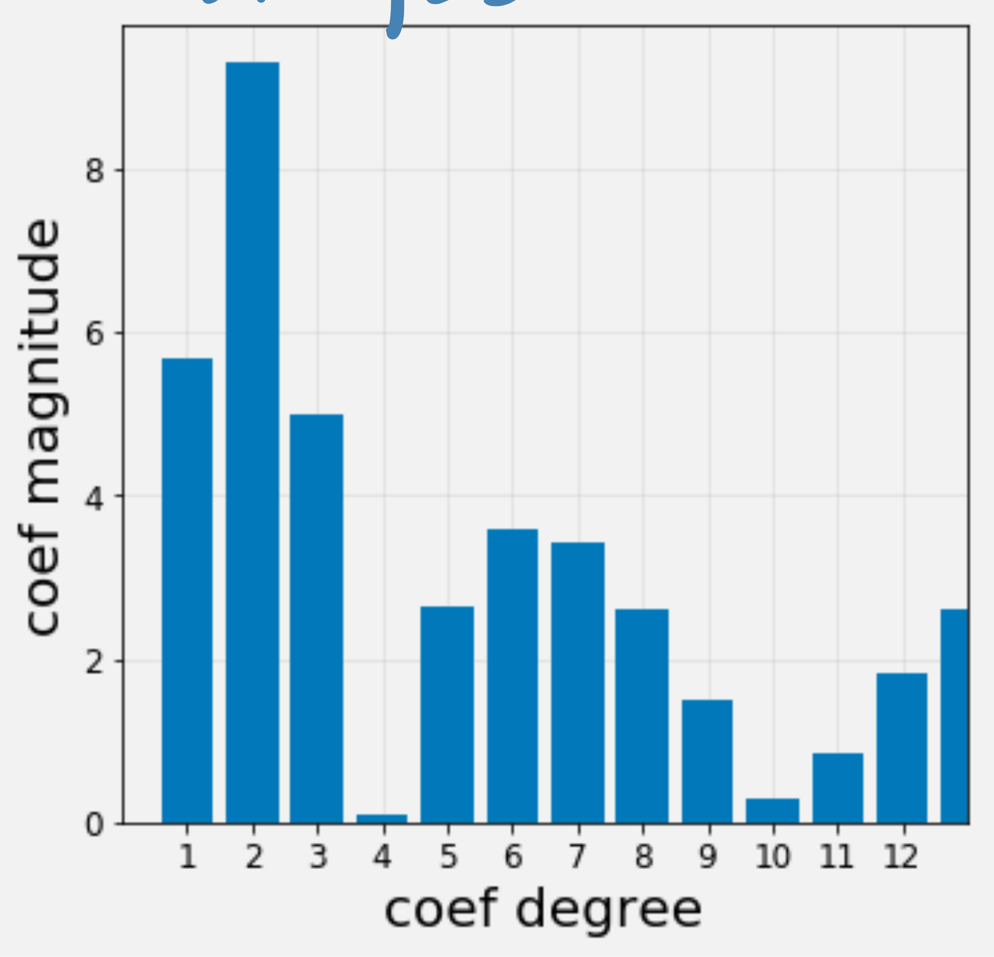


Sinusoidal Data with Ridge Regression

Watch the coefficients

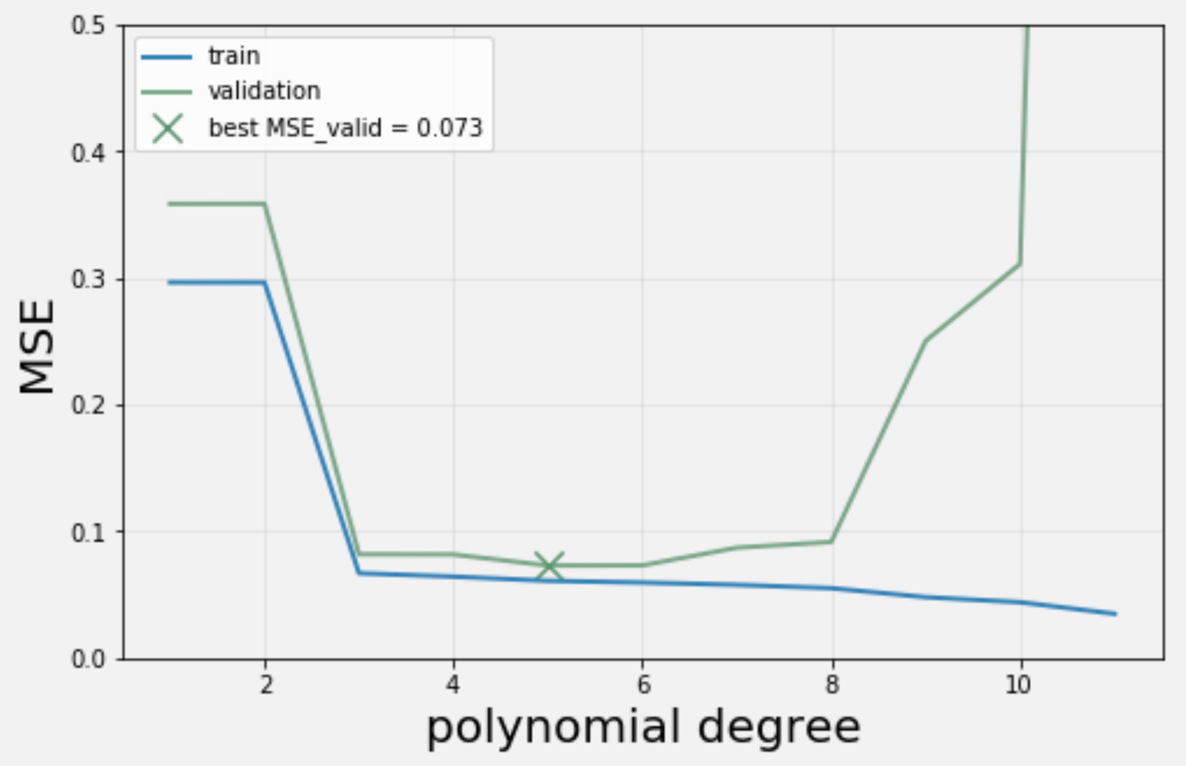
NOTE: COEFFICIENTS STAYED SMALL!



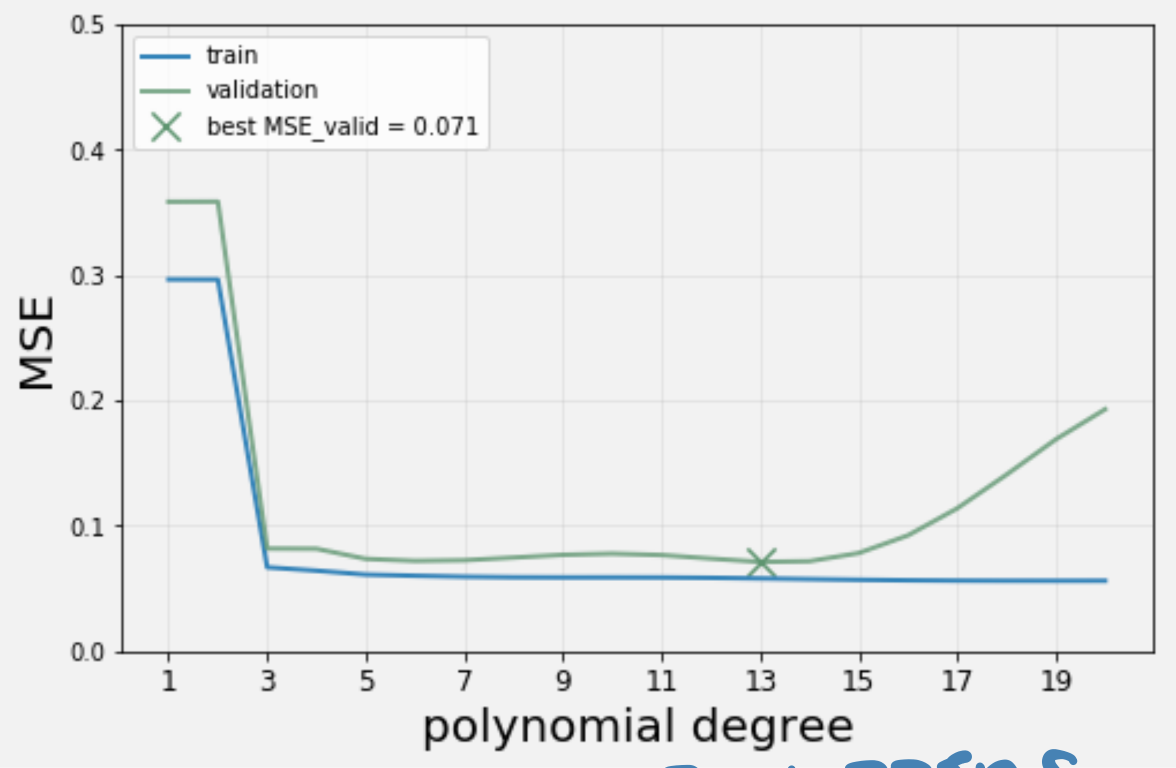


Example: Sinusoidal Data

No Regularization

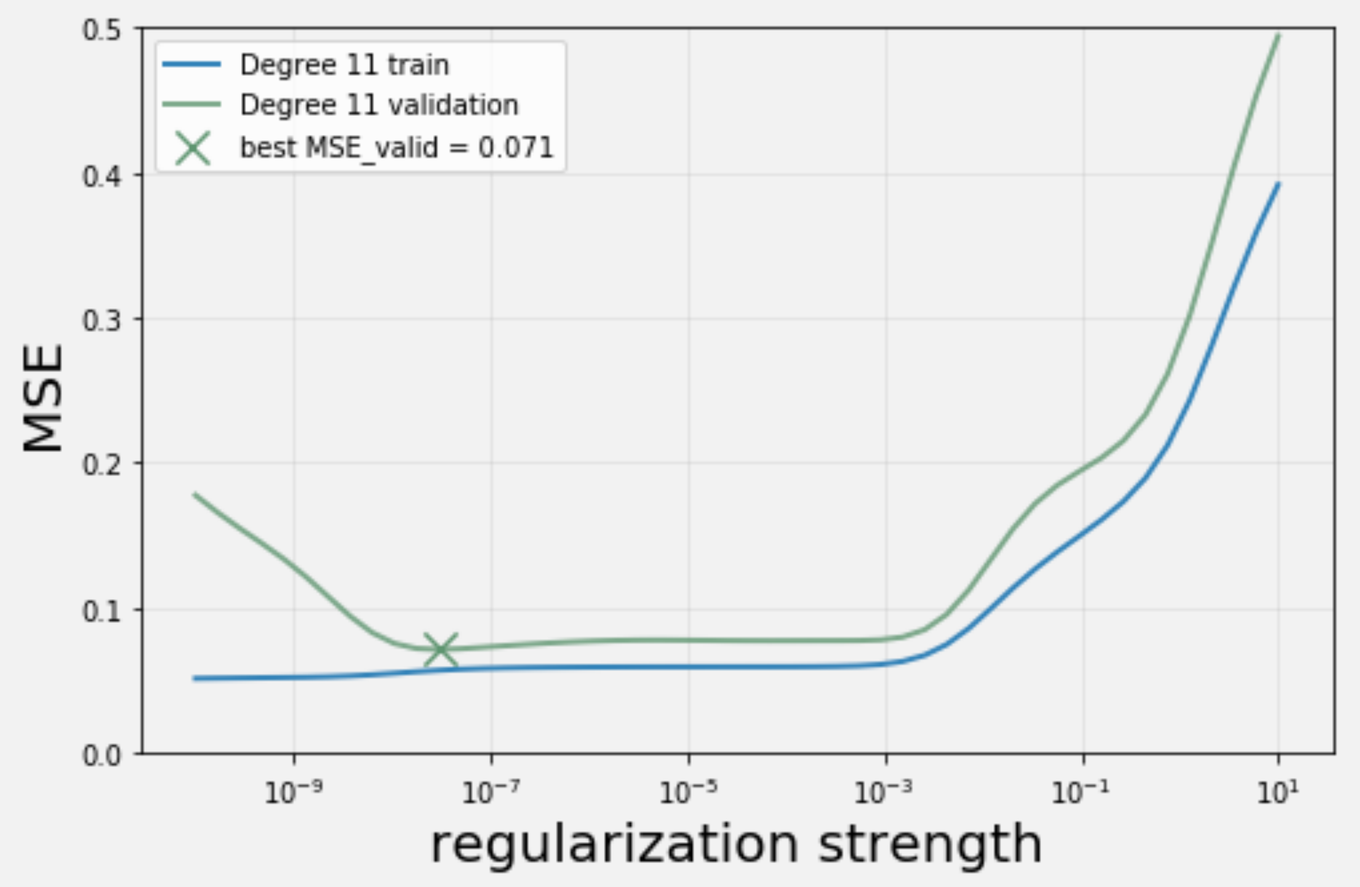


Ridge Regression



BLOW-UP HAPPENS MUCH LATER !

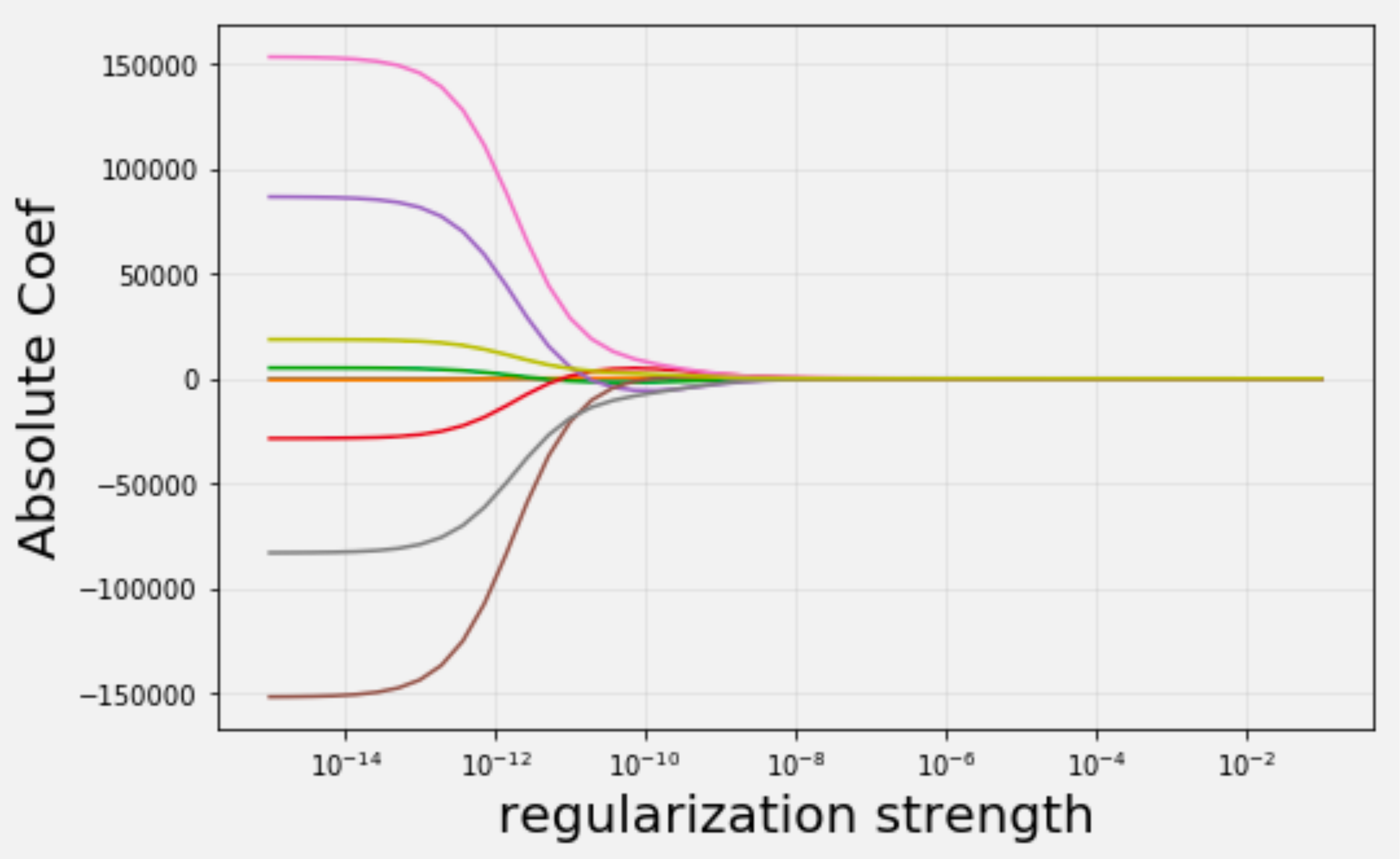
Example: Sinusoidal Data



Regularization Study:

- Choose poly degree
- Vary over reg strength
- Look for best validation MSE

Example: Sinusoidal Data



Regularization Study:

- Choose poly degree
- Vary over reg strength
- Look at size of coefficients

COEFFS SHRINK AS $\lambda \to \infty$

Regularization Wrap-Up

You should always do regularization.

If you choose λ carefully, it will always help Generalization

Next Time:

- Talk more about Ridge Regression Details
- Learning Curves and what they tell us about the Bias-Variance Trade-Off