Generalization Error & Overfitting

CMPUT 296: Basics of Machine Learning

Textbook §8.1-8.2

Logistics

- Thought Questions #2 will be marked today
    - TQ#2 superthread in the discussion forum
- Quiz will be marked by the end of the week
- Assignment #2 is due next **Thursday (Oct 22)**

Recap: Solving Linear Regression

A **linear predictor** has the form $f(\mathbf{x}) = w_0 + w_1 x_1 + \ldots + w_d x_d = \sum_{j=0}^{d} w_j x_j = \mathbf{w}^T \mathbf{x}$

- **Linear regression** is the process of finding a vector $\mathbf{w}$ of weights that minimizes the expected cost of the prediction
- This can be solved **analytically** by solving a system of linear equations
  - But this can be very expensive for large d: $O(nd^2 + d^3)$
- More common solved **numerically** by **first-order gradient descent**
  - But this can also be very expensive for large n: $O(ndk)$ for k iterations
  - We can get around this using **stochastic gradient descent**
- Linear regression can be straightforwardly extended to **nonlinear regression**
  - Just do linear regression on a bunch of nonlinear features

Outline

1. Recap & Logistics
2. Overfitting
3. Estimating Generalization Error

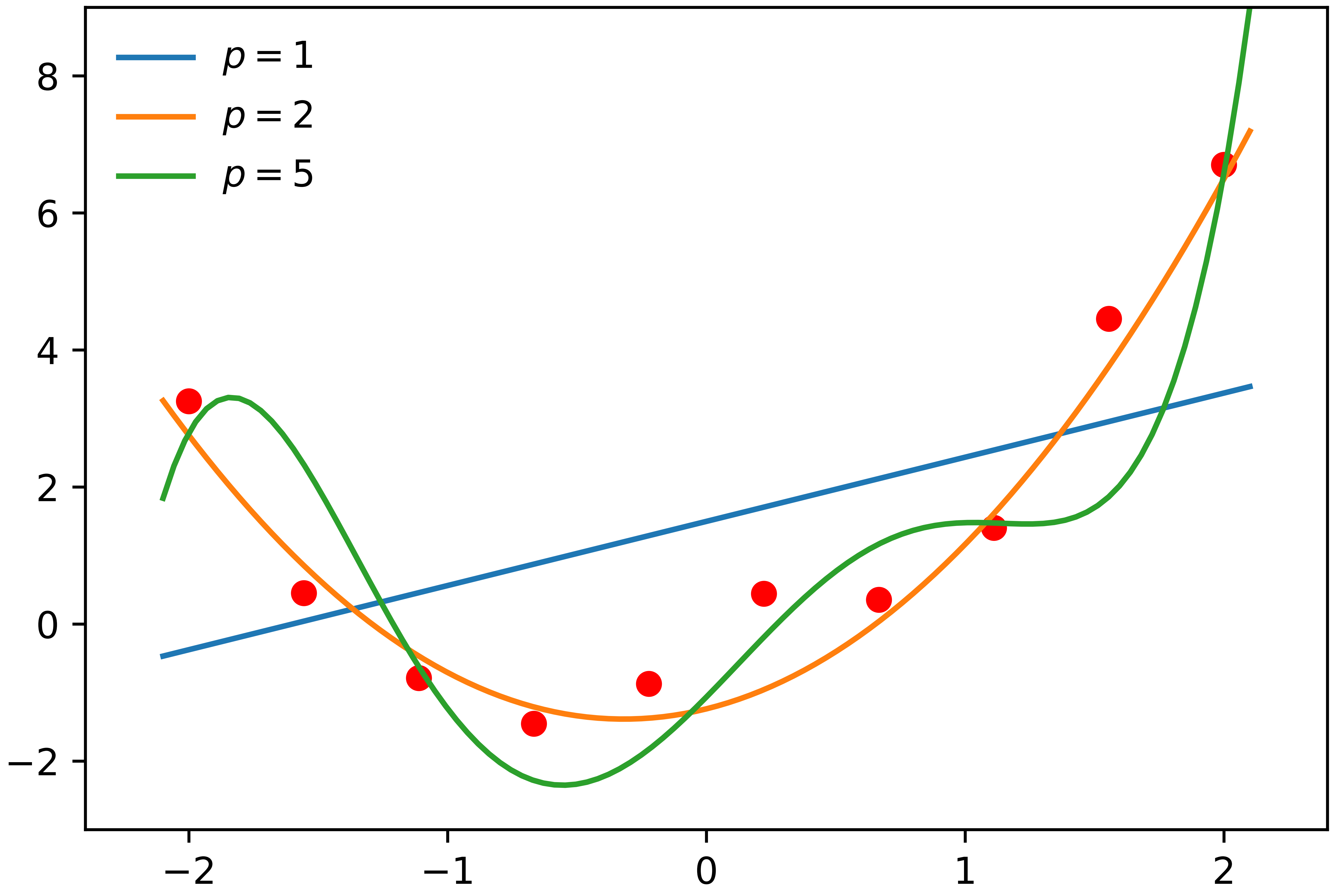
Comparing Models

- **Consistency** tells us about the behavior of a particular **estimator** in the limit of **infinite data**
- In the context of **parametric learning**, the estimate is the model
    - i.e., the "true parameter" vectory $\boldsymbol{\omega}$ is the unknown quantity being estimated
    - The MLE estimator $\mathbf{W}_{\text{MLE}}$ is a random variable, because it is a function of the dataset $\mathscr{D}$ (assumed to be an i.i.d. sample)
    - The actual estimate $\mathbf{w}_{\text{MLE}}$ is what we compute for a single realization of $\mathscr{D}$

Question: Given two specific models f_1 and f_2 computed from a **finite dataset** $\mathscr{D}$, is it even possible to tell which one is "better"?

Comparing Models: Polynomial Fits

Question: Which model is better?



Generalization Error

Question: What do we mean by one model being better than another?

Definition: Generalization error is a synonym for the expected cost:

$$\mathbb{E}[C] = \int_{\mathcal{X}\times\mathcal{Y}} p(\mathbf{x}, y)\mathrm{cost}\left(f(\mathbf{x}), y\right)\, d\mathbf{x}dy$$

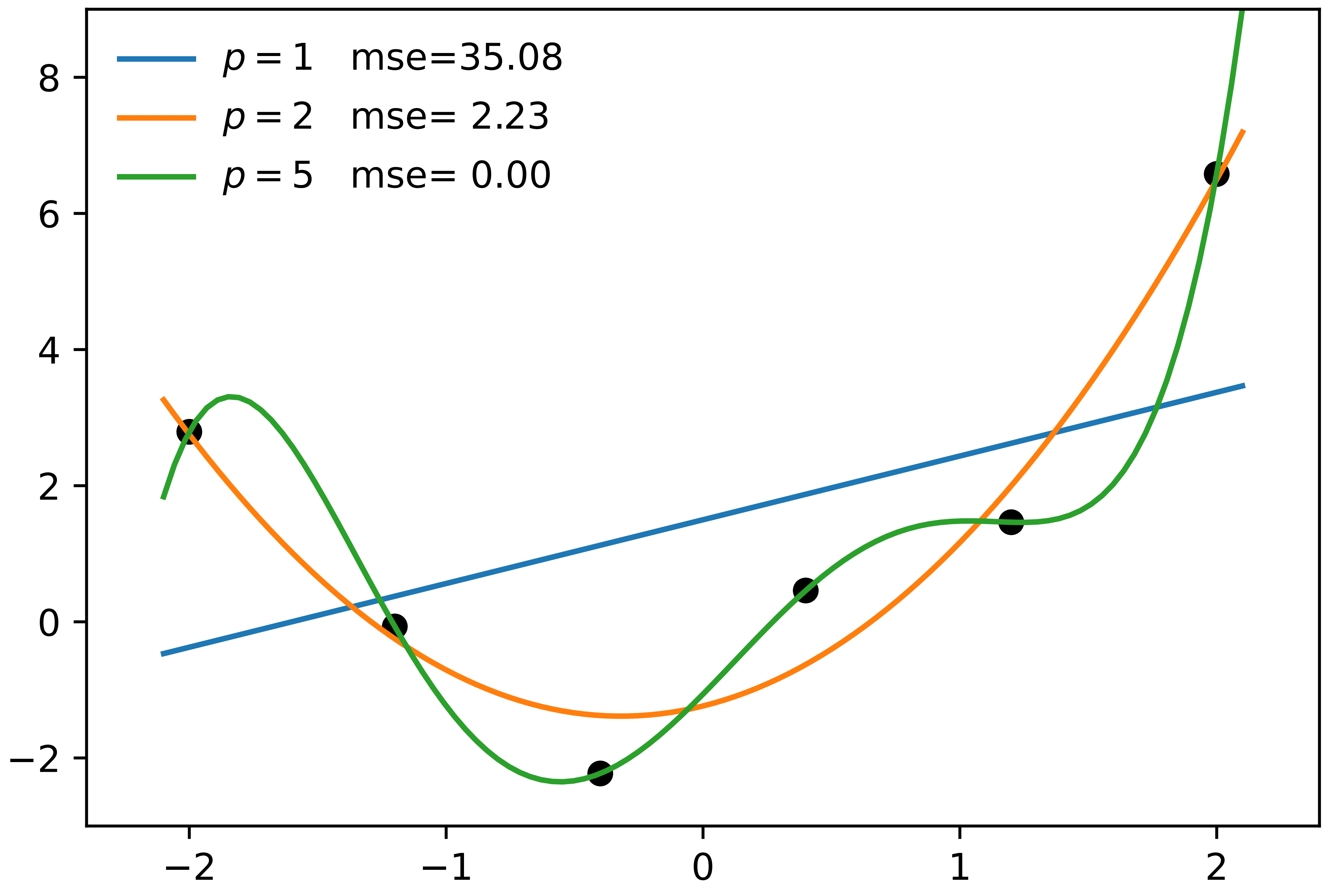
Question: How can we minimize generalization error?

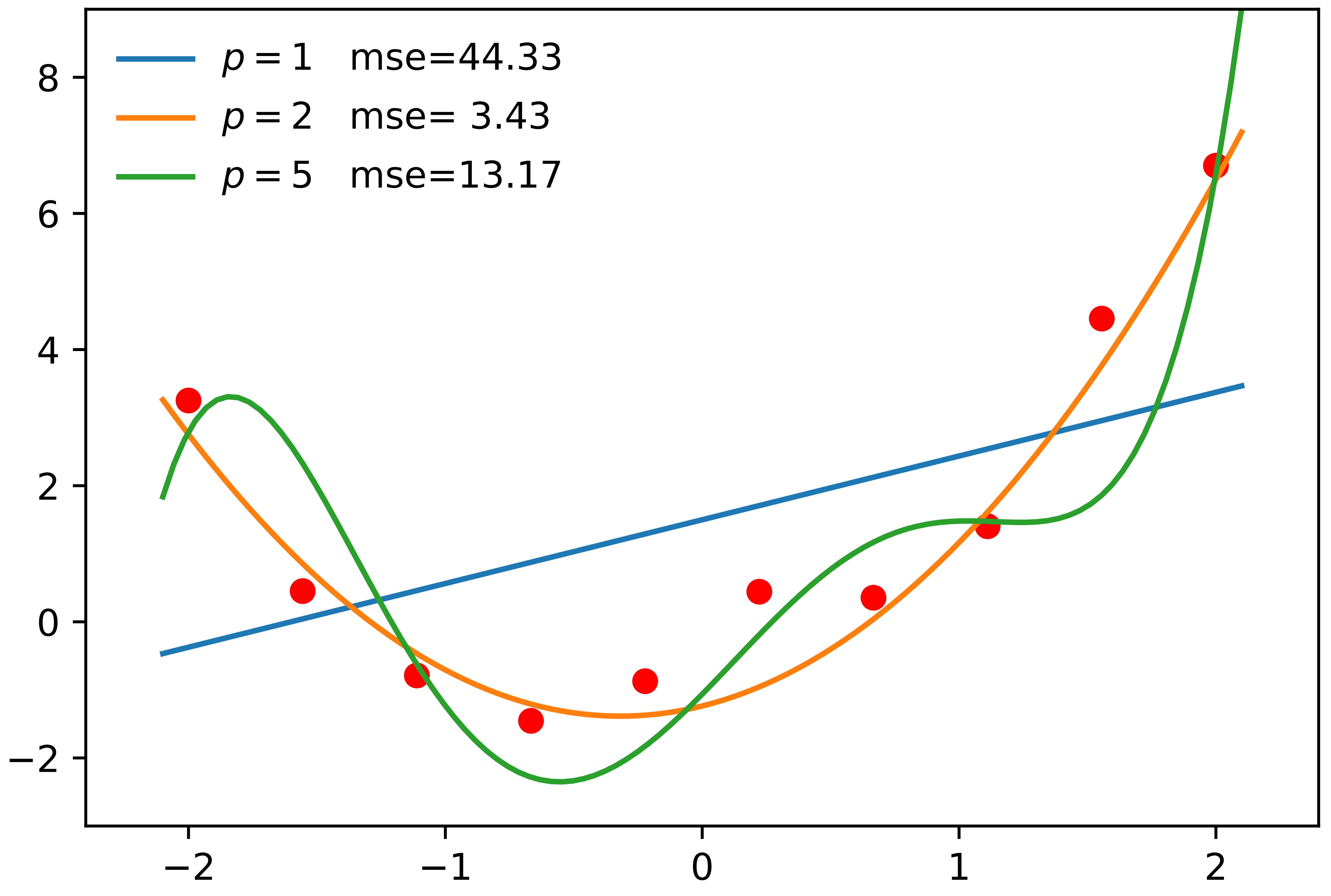
Definition: Empirical error is the cost realized on the training data:

$$\hat{C} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{cost}\left(f(\mathbf{x}_i), y_i\right)$$

Comparison Using Empirical Error

Question: Can we use empirical error to compare models?





Overfitting

Definition: Overfitting occurs when we select a model that has very good **empirical error** (possibly 0), but extremely poor **generalization error**.

Questions:

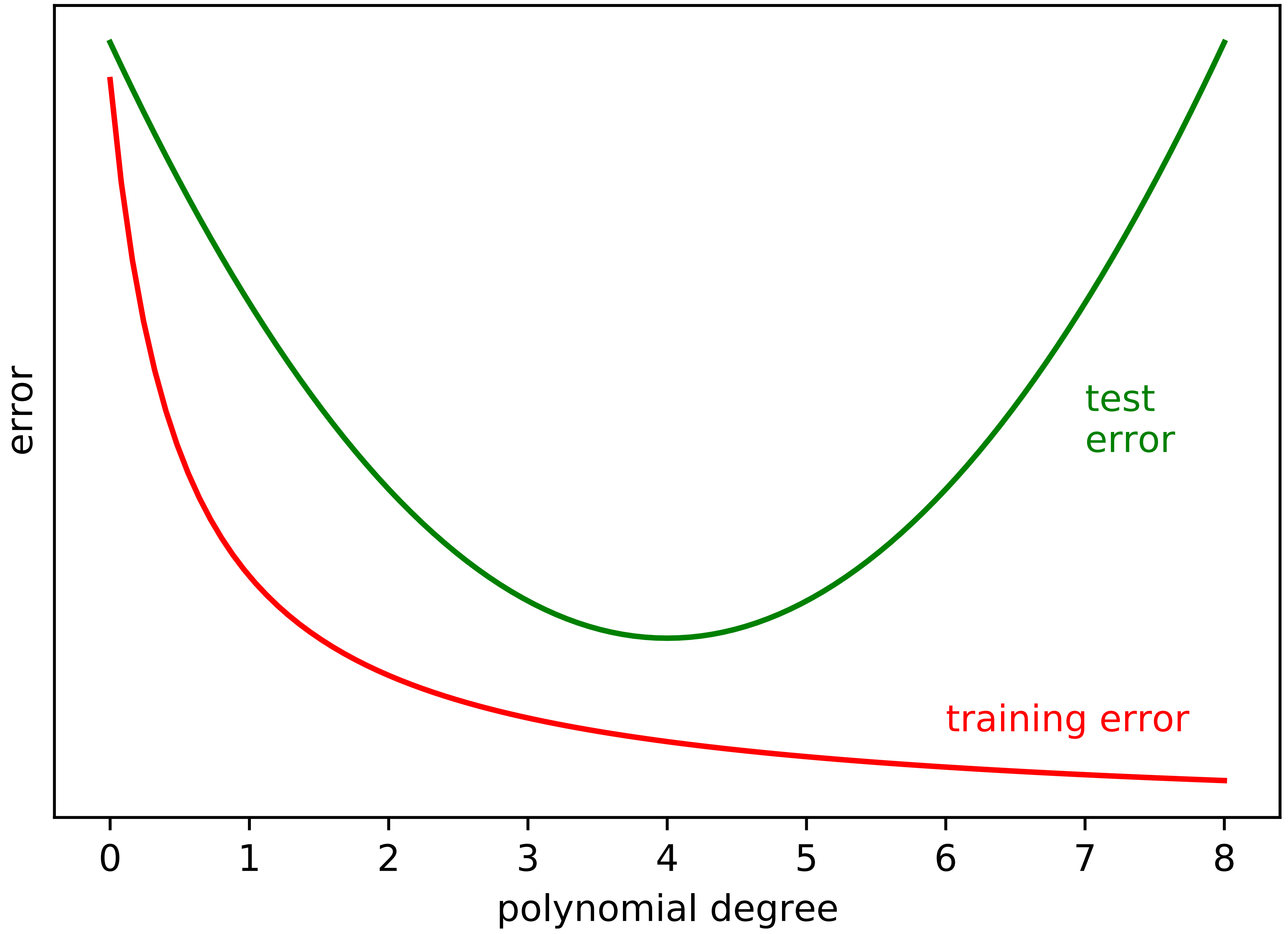
1. Can you guess which of $p = 1$, $p = 2$, or $p = 5$ will have lowest empirical error on my next crazy dataset, *before I tell you what the data even are*?
2. What if I tell you that the data were generated using a quadratic with Gaussian noise?
3. If we cannot estimate generalization error using empirical error, how can we avoid overfitting?

Estimating Generalization Error

- Turns out we *can* estimate generalization error using empirical error
- **Empirical error** on an i.i.d. dataset is an **unbiased estimator** of **generalization error**
- But the i.i.d. dataset ***must not*** be the same dataset that we used to train the model in the first place (**why?**)
- Instead, we **hold out** some of our dataset
  - The non-held-out data (the **training set**) is used to train the model
  - The held-out data (the **test set**) is used to estimate generalization error

Detecting Overfitting

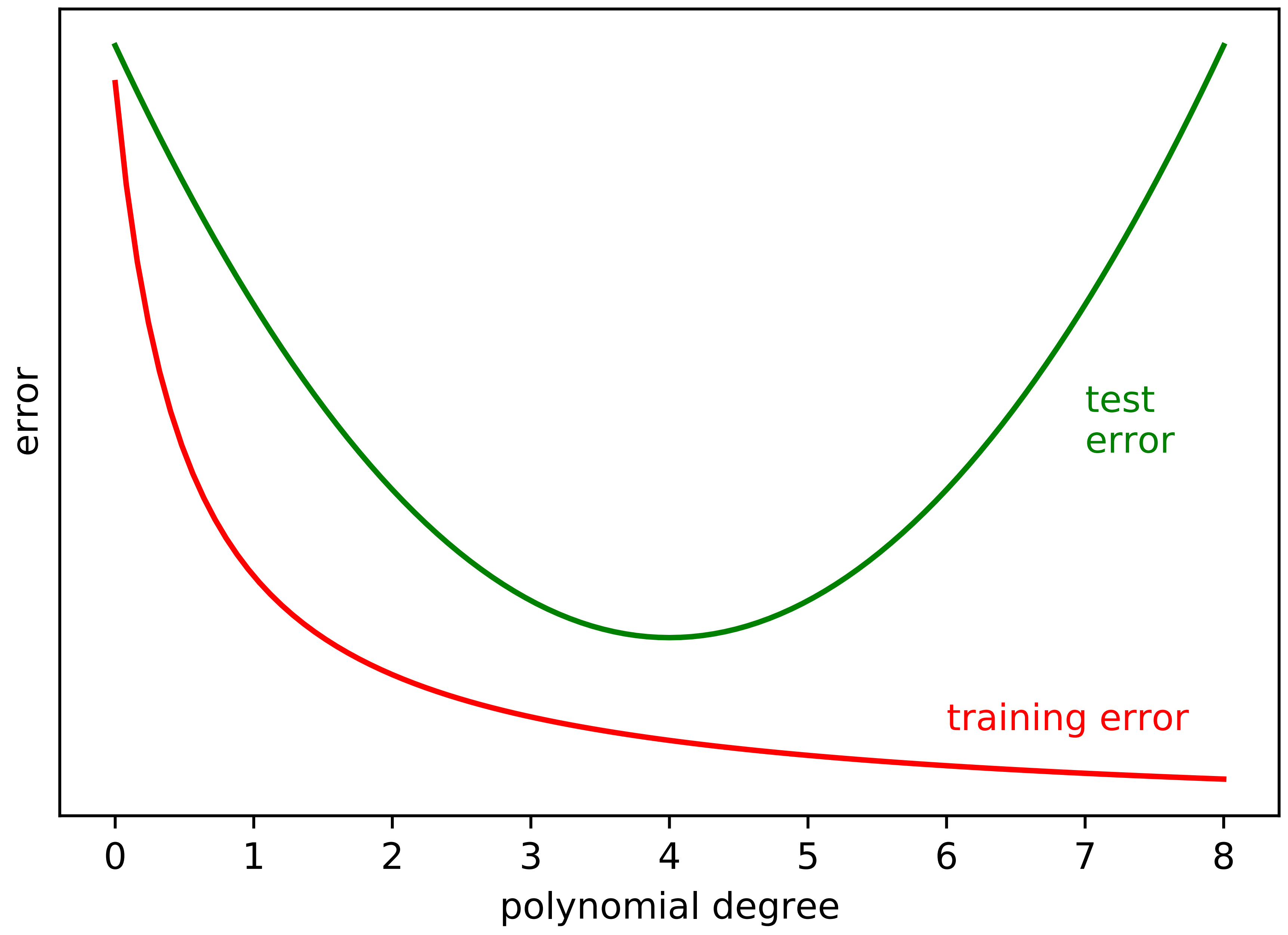
Question:
If the **training error** (the empirical error on the training set) is smaller than the **test error** (empirical error on the test set), does that indicate that we are overfitting?

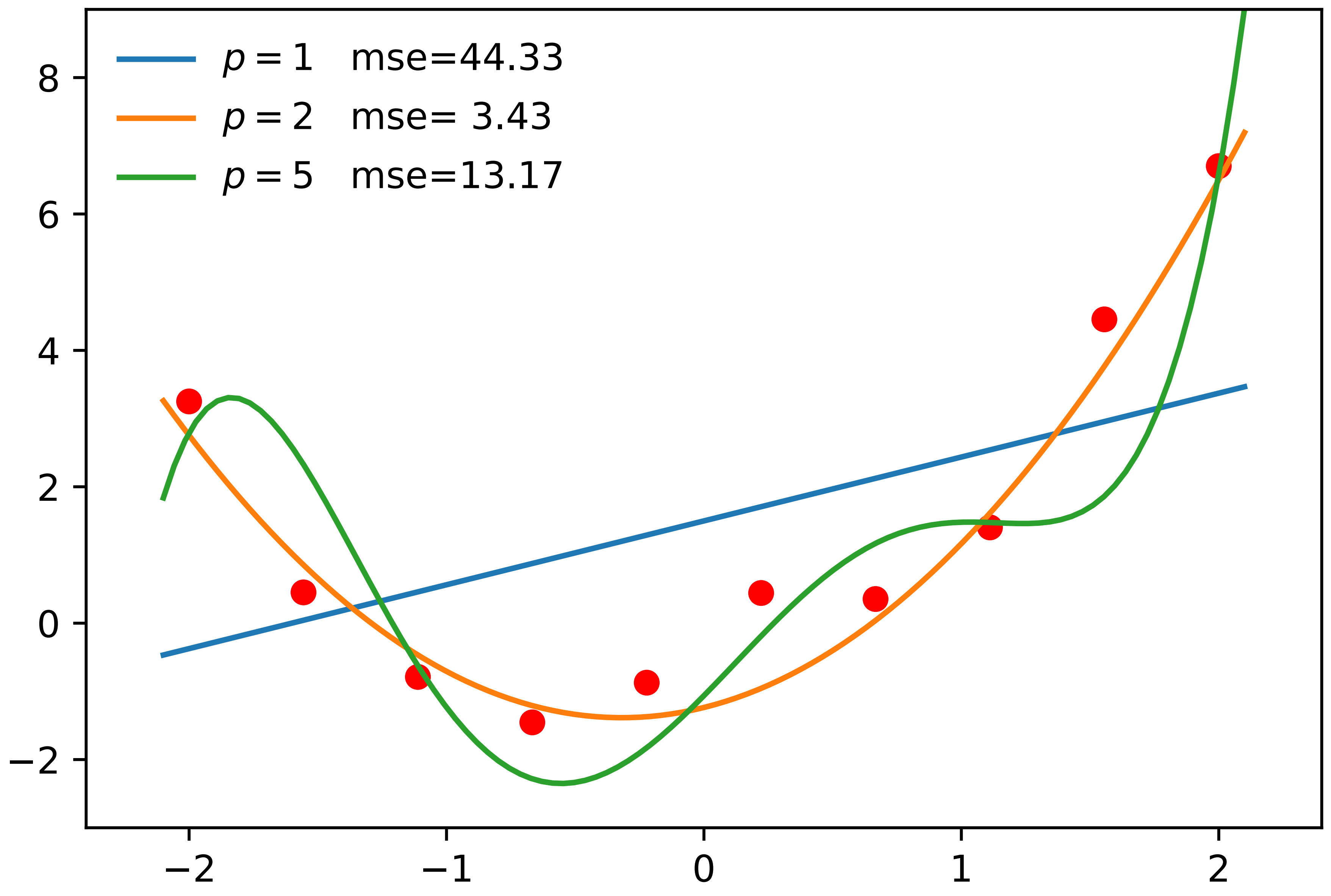


Question: At what point does this hypothetical regression start to overfit?

Underfitting

- **Overfitting** is the result of using an **overly complex model** (on **too little data**)
- **Question:** Can we guarantee good generalization performance by always using a very simple model?
- **Underfitting** is the result of using an **overly simple model**
- We need our model to be *complex enough* to capture the underlying process, but *simple enough* that it doesn't also learn noise from our training data





Drawbacks of Held-Out Data

Using a held-out test set has two main disadvantages:

1. **We want to use as much of our data for training as possible**
    - Every datapoint that we hold out for estimating generalization error is a datapoint that we can't train out model with
2. **We can only use a held-out test set once**
    - If you choose your hyperparameters (e.g., p for polynomial regression) using a test set, then you have effectively used it for **training**
    - If you use a dataset to choose your model, then generalization error estimates based on that dataset will inevitably be **optimistic**

Alternative: k-fold Cross-Validation

k-fold cross-validation

1. Randomly partition $\mathscr{D}$ into equal-sized disjoint subsets $\mathscr{D}^{(1)}, \ldots, \mathscr{D}^{(k)}$
2. For all $1 \leq j \leq k$, train a model $f^{(j)}$ using $\mathscr{D} \backslash \mathscr{D}^{(j)}$
3. For all $1 \leq j \leq k$, compute empirical error $\hat{C}^{(j)}$ of model $f^{(j)}$ on $\mathscr{D}^{(j)}$
4. Estimated generalization error is mean: $\dfrac{1}{k} \sum_{j=1}^{k} \hat{C}^{(j)}$

- **Every** datapoint gets used for testing **once**
- Extreme version: $k = n$ (aka **leave-one-out** cross-validation)
- Often used **on the training set** to choose hyperparameters (e.g., p for polynomial regression)
    - Since it's used on the training set, can use a separate held-out set to evaluate the final model

Alternative: Bootstrap Resampling

- **Bootstrapping** assumes that the data is a reasonable model of the underlying (true) distribution
- So to create a test/training split, sample from the dataset itself!

Bootstrap resampling

1. For $1 \leq j \leq k$, sample n datapoints **with replacement** from $\mathcal{D}$; call this $\mathcal{D}^{(j)}$
2. For all $1 \leq j \leq k$, train a model $f^{(j)}$ on $\mathcal{D}^{(j)}$
3. For all $1 \leq j \leq k$, compute empirical error $\hat{C}^{(j)}$ of model $f^{(j)}$ on $\mathcal{D} \setminus \mathcal{D}^{(j)}$
4. Estimated generalization error is mean: $\dfrac{1}{k}\sum_{j=1}^{k} \hat{C}^{(j)}$

- As with k-fold cross-validation, this can be used on the training set for selecting hyperparameters
- **Question:** How does this (or k-fold cross-validation) address the "only use test set once" issue?

Summary

- Our goal is to minimize **generalization error**: expected cost with respect to the **underlying distribution**
- But we only have access to **empirical error**: average cost on a dataset
- The empirical error of a model on its **training data** is a **biased, over-optimistic estimate** of generalization error
- Using an **overly complex** model leads to **overfitting**:
  High training performance at the expense of generalization performance
  - **Underfitting** comes from using an **overly simple** model
- A **held-out test set** gives an unbiased estimate of generalization error
  - But you can only use it **once**!
  - Alternatives: k-fold cross-validation; bootstrap resampling