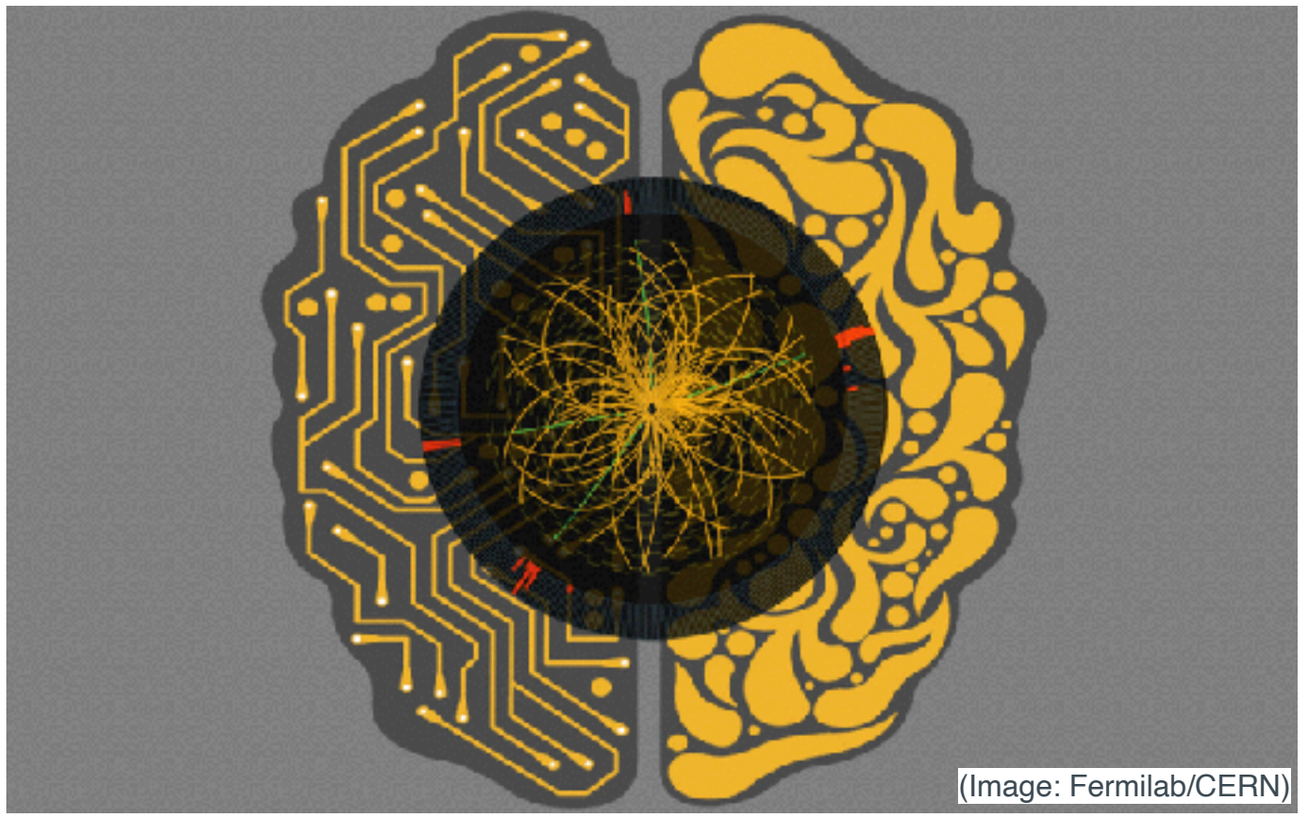
PHY 835: Collider Physics Phenomenology

Machine Learning in Fundamental Physics

Gary Shiu, UW-Madison



Lecture 2: Basics of Machine Learning

Recap of Lecture 1

- Be able to tell a friend examples of problems where ML can be used in collider physics (and physics in general).

- Where can ML be useful in Theoretical Physics?

- How can a physics problem be related to identifying cats and dots on images.

- Remember to start installing the software packages (Exercise 1) and get familiarized with them.

A Physicist's Definition of ML

- 0th order: ML is concerned with algorithms improving from data automatically.

- Different from **physics model fitting** to experimental data? Not entirely, but there are differences.

- ML focus on **predictions** (i.e. accuracy on **unseen data)**

- It is an **"experimental field"**: sometimes not clear why they are working (~mathematical precise definition of path integrals).

- As a physicist: It is about finding the right "phenomenological" model/ framework to describe data/address the task.

Typical ML Problem

- **Dataset** D=(**X**,**y**) where **X**=matrix of independent variables (input), **y**=vector of dependent variables (output).

- **Model** f(x; θ): **x** → y with parameter θ

- **Cost function** C(y, f(x; θ)): judges how well the model performs.

- Find model (parameters) which minimizes the cost function.

- Commonly used cost function is the **mean squared error**,

$$C(\mathbf{y}, f(\mathbf{X}; \theta) = \sum_i \left(y_1 - f(\mathbf{x}_i; \theta)\right)^2$$

Recipe

- Randomly divide the dataset D into 2 mutually exclusive groups: D_{train} and D_{test} (known as **cross-validation** in ML/statistics).

- Minimize the cost function with respect to the training set:

$$\hat{\theta} = \arg\ \min_{\theta}\{C\left(\mathbf{y}_{\text{train}}, f(\mathbf{X}_{\text{train}}; \theta)\right)\}$$

- Evaluate the performance on the test set:

$$C\left(\mathbf{y}_{\text{test}}, f(\mathbf{X}_{\text{test}}; \hat{\theta})\right)$$

- Define **in-sample error** and **out-of-sample error**:

$$E_{in} = C\left(\mathbf{y}_{\text{train}}, f(\mathbf{X}_{\text{train}}; \hat{\theta})\right) \quad , \quad E_{out} = C\left(\mathbf{y}_{\text{test}}, f(\mathbf{X}_{\text{test}}; \hat{\theta}))\right)$$

- Typically $E_{out} \geq E_{in}$.

Predicting vs Fitting

- In ML we do not know the underlying model or want to check one underlying model compared to many experimental setups.

- **Model comparison** done by performance on E_{out}.

- **Model selection:** model that minimizes E_{out}.

- Model with the lowest E_{out} does not usually have the lowest E_{in}.

- Let's get some intuition for why predicting and fitting are different things by trying out the polynomial regression examples in Notebook 1 of Mehta et al:

https://physics.bu.edu/%7Epankajm/MLnotebooks.html

Polynomial Regression

- Fitting data with polynomials of different order.

- Dataset is generated by drawing samples from:

$$y_i = f(x_i) + \eta_i$$

- $f(x_i)$ is some polynomial and η_i is Gaussian uncorrelated noise:

$$< \eta_i > = 0, \quad < \eta_i \eta_j > = \delta_{ij} \sigma^2$$

- σ is the **noise strength**. The larger σ is the noisier the data; $\sigma = 0$ corresponds to the noiseless case.

- Consider different model classes $f_\alpha(x; \theta_\alpha)$ to model the data and make predictions.

Complexity vs Predictivity

- The model class $f_\alpha(x;\theta_\alpha)$ encodes the **features** we choose to represent the data.

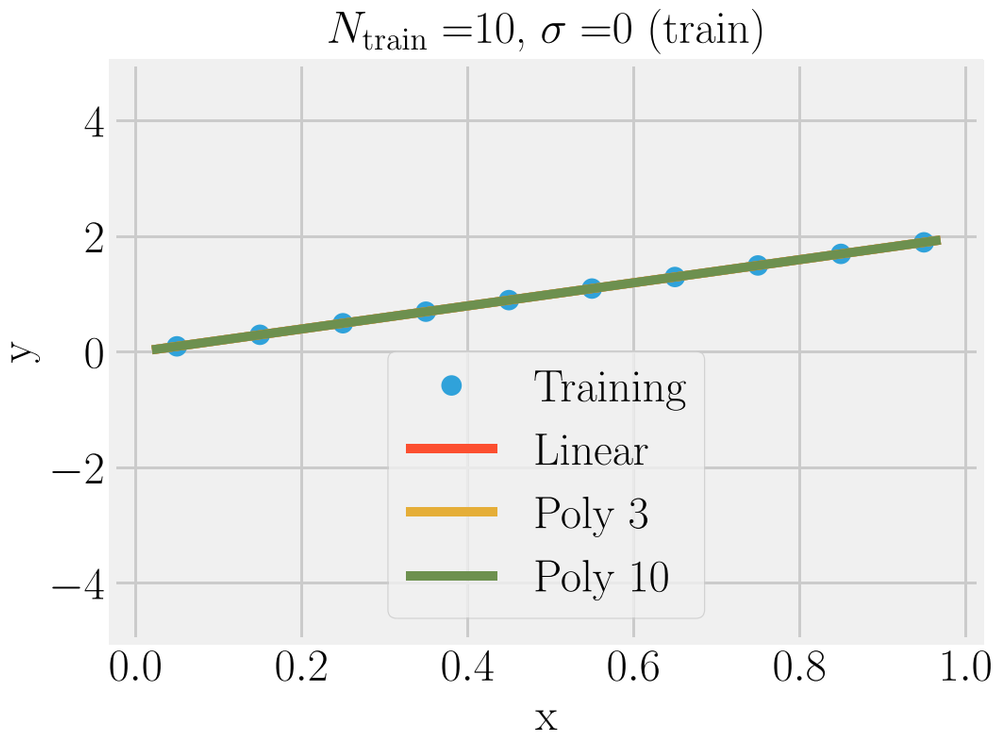
- Consider three model classes $f_\alpha(x;\theta_\alpha)$ with $\alpha = 1,3,10$ corresponding to all polynomials of order α.

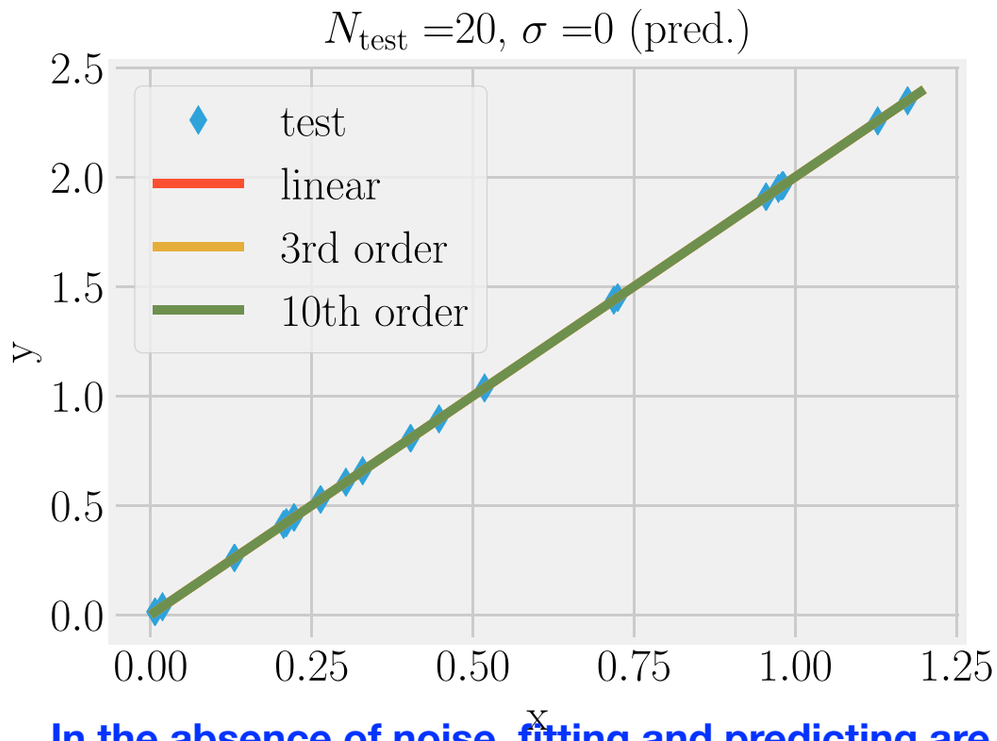
- Different **model complexities**: each term in the polynomial is a **feature** of the model; increasing the order increases # features.

- More complex model class may give better **predictive** power, but only if D_{train} is large enough to accurately learn these parameters.

Noiseless Case

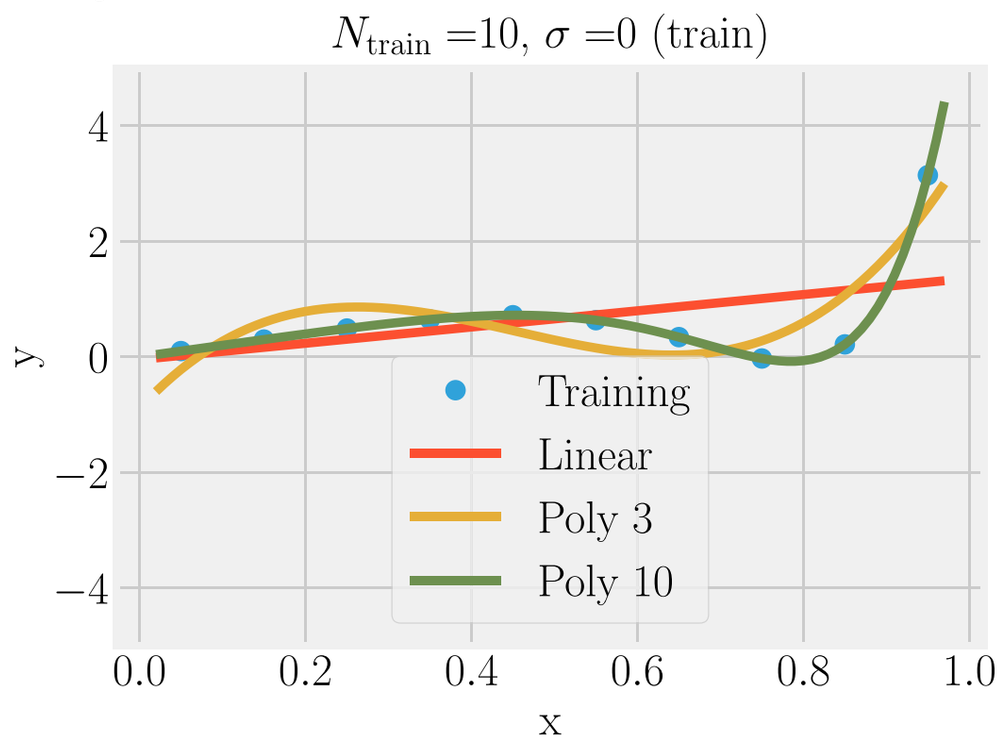
$f(x) = 2x, x \in [0,1]$

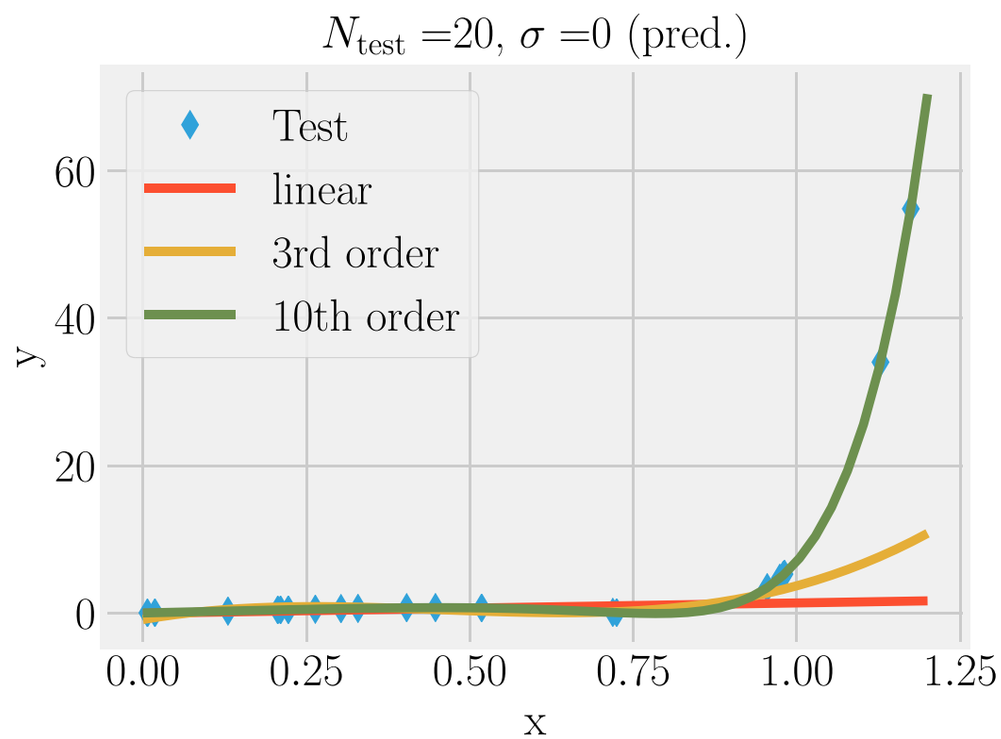




In the absence of noise, fitting and predicting are identical, provided model has enough flexibility.

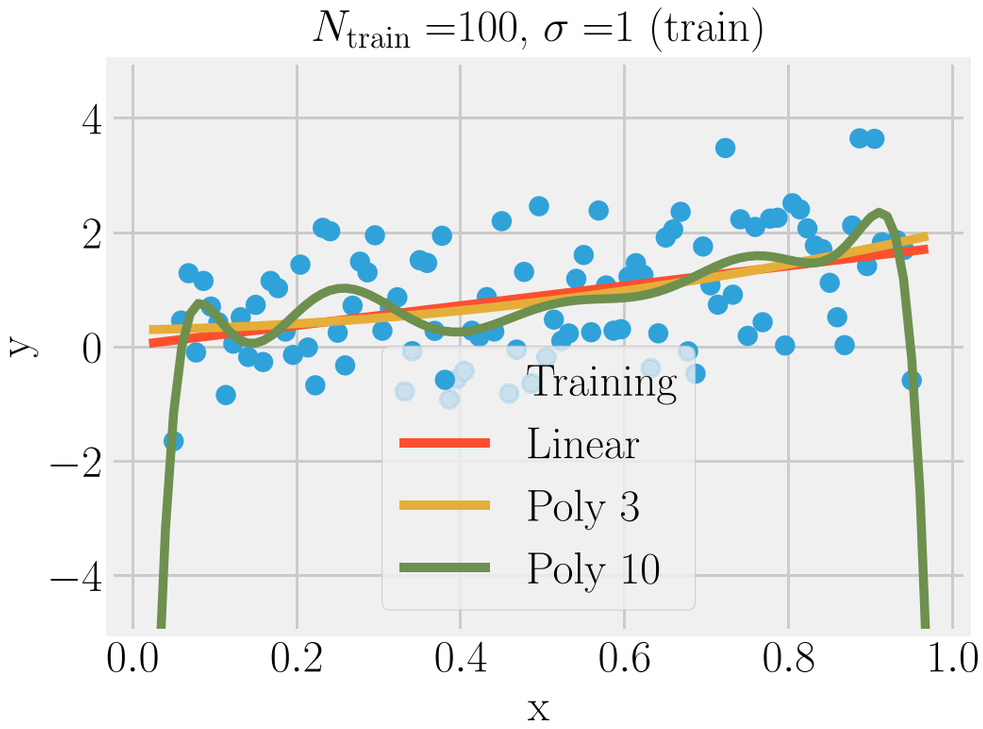
$f(x) = 2x - 10x^5 + 15x^{10}, x \in [0,1]$

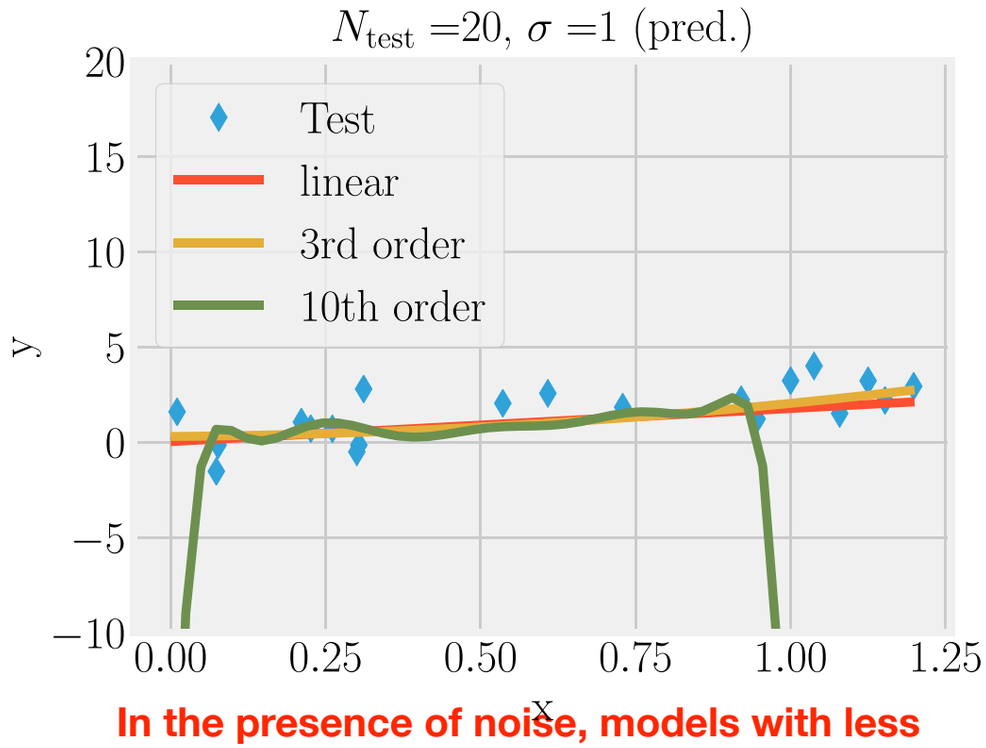




Noisy Case

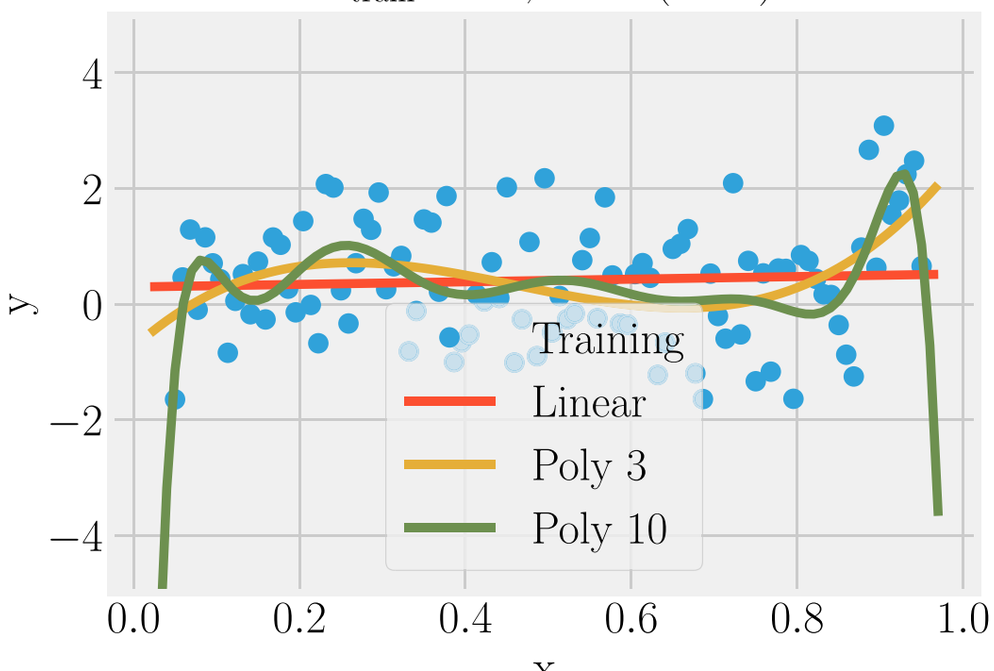
$f(x) = 2x, x \in [0,1]$

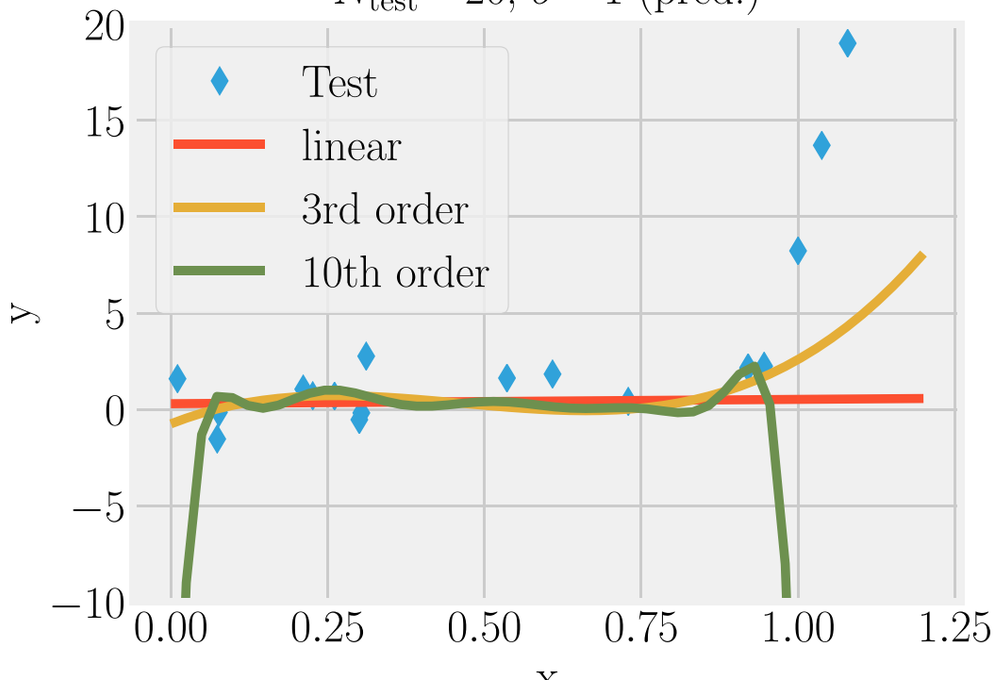




$f(x) = 2x - 10x^5 + 15x^{10}, x \in [0,1]$

In the presence of noise, models with less complexity can exhibit improved predictive power.





Overfitting

- The 10^{th} order model makes the worst out-of-sample predictions, even if the data was generated by a 10^{th} order polynomial.

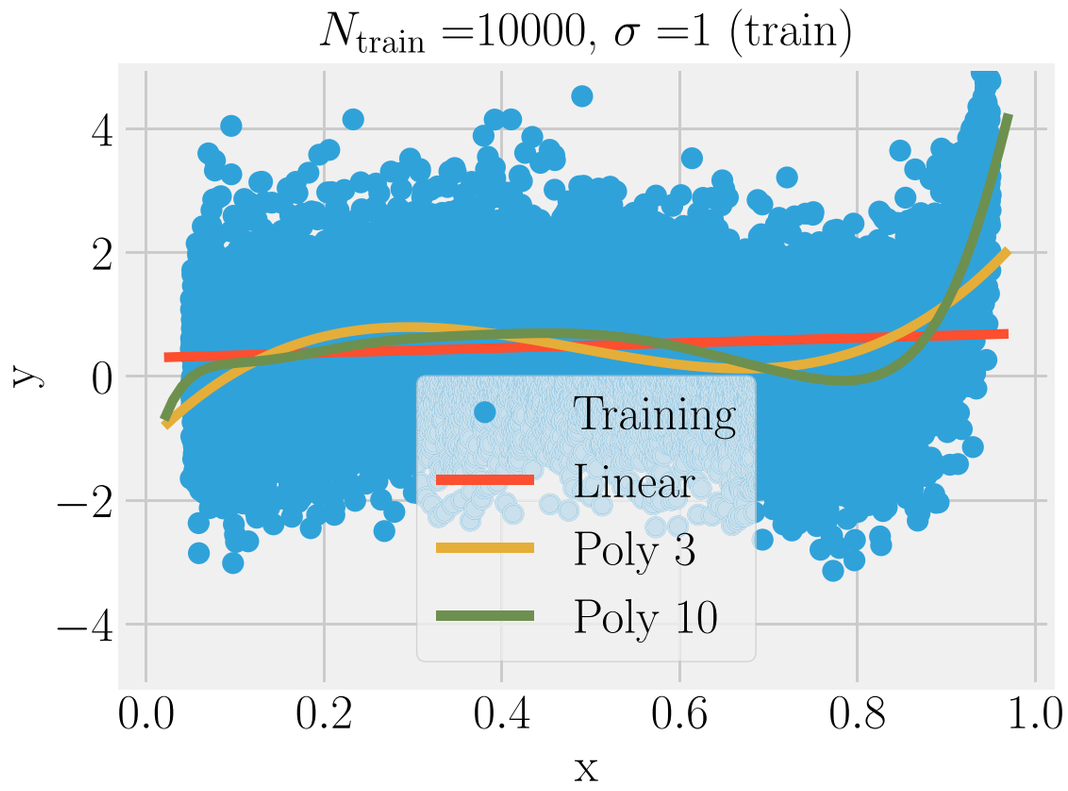
- At small sample sizes, noise can create fluctuations in the data that look like genuine pattern.

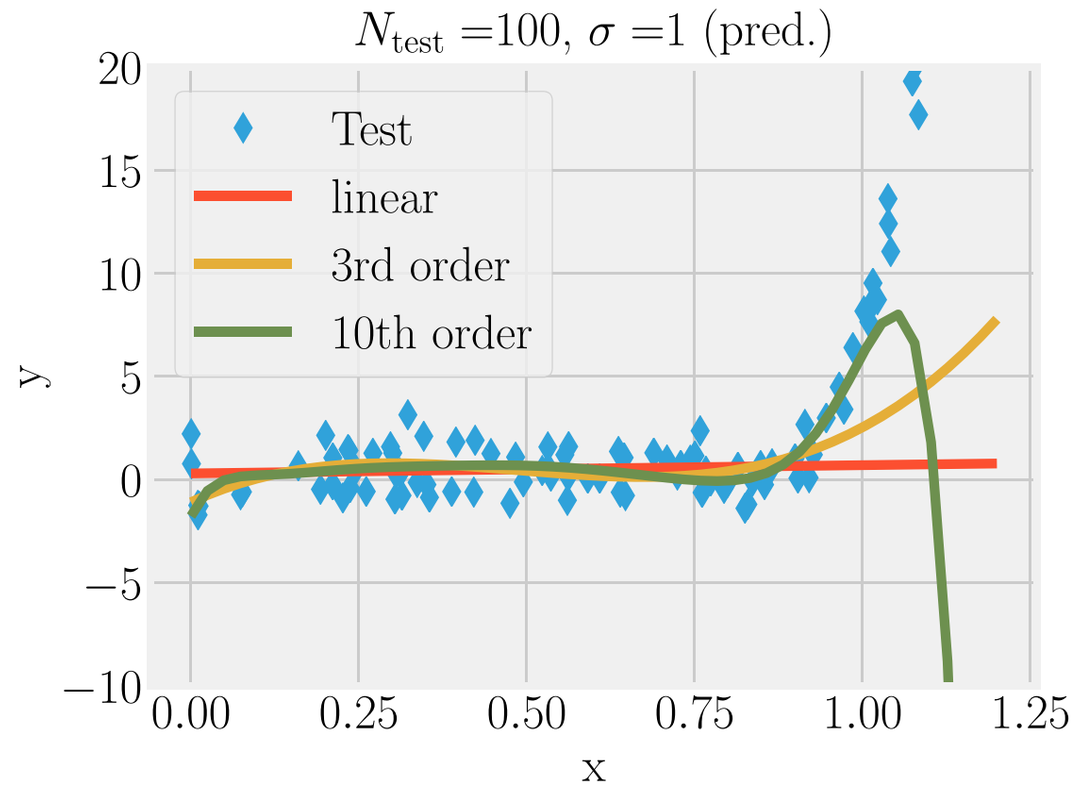
- Simple models (e.g., linear function) are forced to ignore the fluctuations and focus on the larger trend.

- Complex models can capture both the global trends and noise, and can be tricked into **“overfitting”**.

- Can avoid overfitting by 1) using less expressive models (regularization, more later) or 2) increasing the size of training set.

Larger Training Set





- With a larger training set, the 10th order polynomial gives both the best fits and the most predictive power over the entire training range $x \in [0,1]$ and even slightly beyond to ≈ 1.05.

- An illustration of **bias-variance tradeoff:** simple models have more **"bias"** but less **"variance"**.

Why is ML difficult?

- Fitting (existing data) is not predicting (unseen data).

- Using a complex model can result in **overfitting** when the training data size is small and the data is noisy.

- For complex datasets and small training sets, simple models can be better at prediction due to the **bias-variance trade-off**.

- Difficult to generalize beyond the situations (data range) encountered in the training set.

Basics of Statistical Learning

Statistical Learning

- **Hypothesis set** (contains all possible models we consider).

- The goal of **Statistical Learning** is to determine a function from the hypothesis set that approximates f(x) as best as possible.

- Intuitively, we want to learn a function that performs probably as well on new data as on training data.

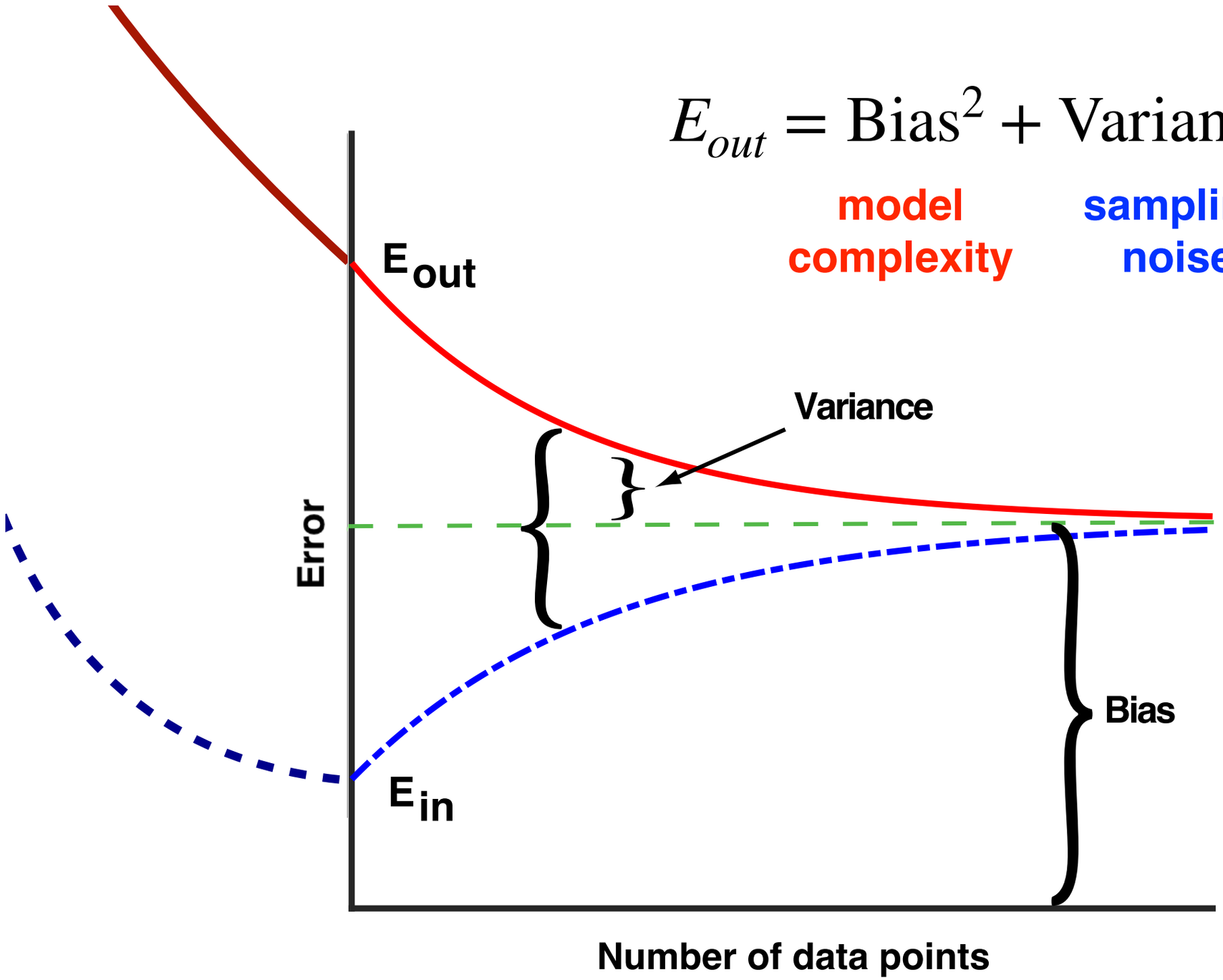
- What is the relation between E_{in} and E_{out} ?

- **Assumption:** cannot exactly learn the target function f(x).

Bias-Variance Decomposition

$$E_{out} = \text{Bias}^2 + \text{Variance}$$

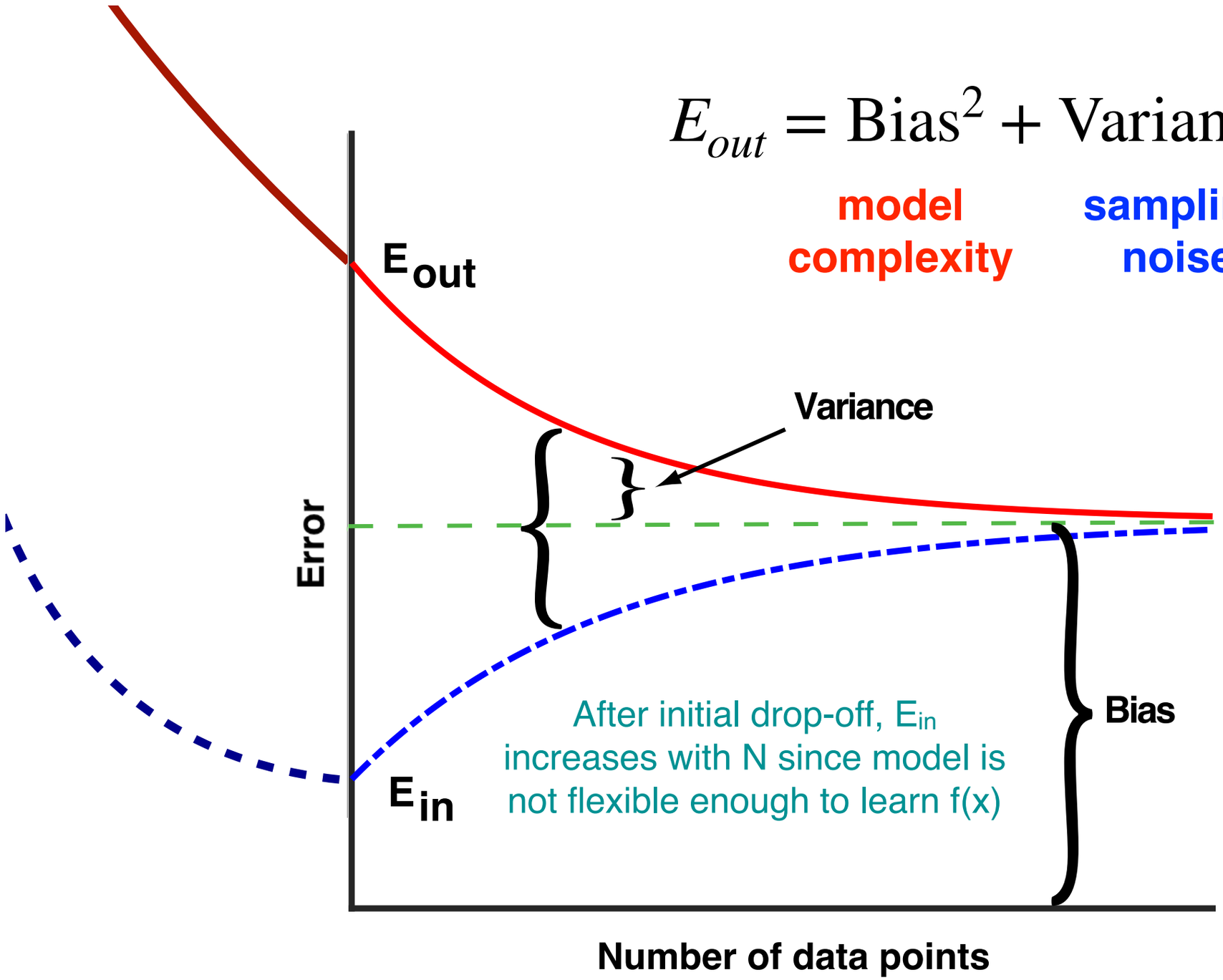
model complexity **sampling noise**



Bias-Variance Decomposition

$$E_{out} = \text{Bias}^2 + \text{Variance}$$

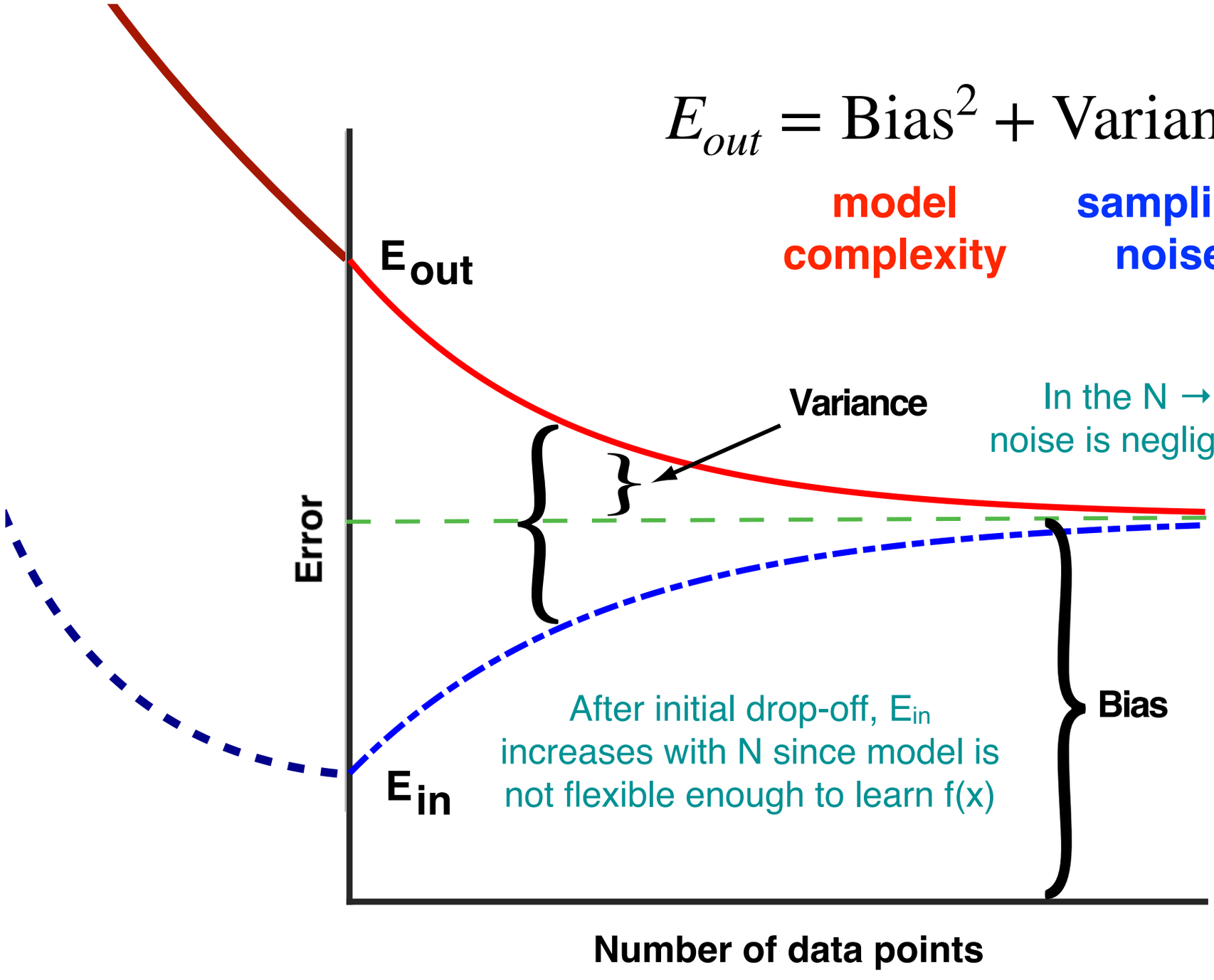
model complexity **sampling noise**



Bias-Variance Decomposition

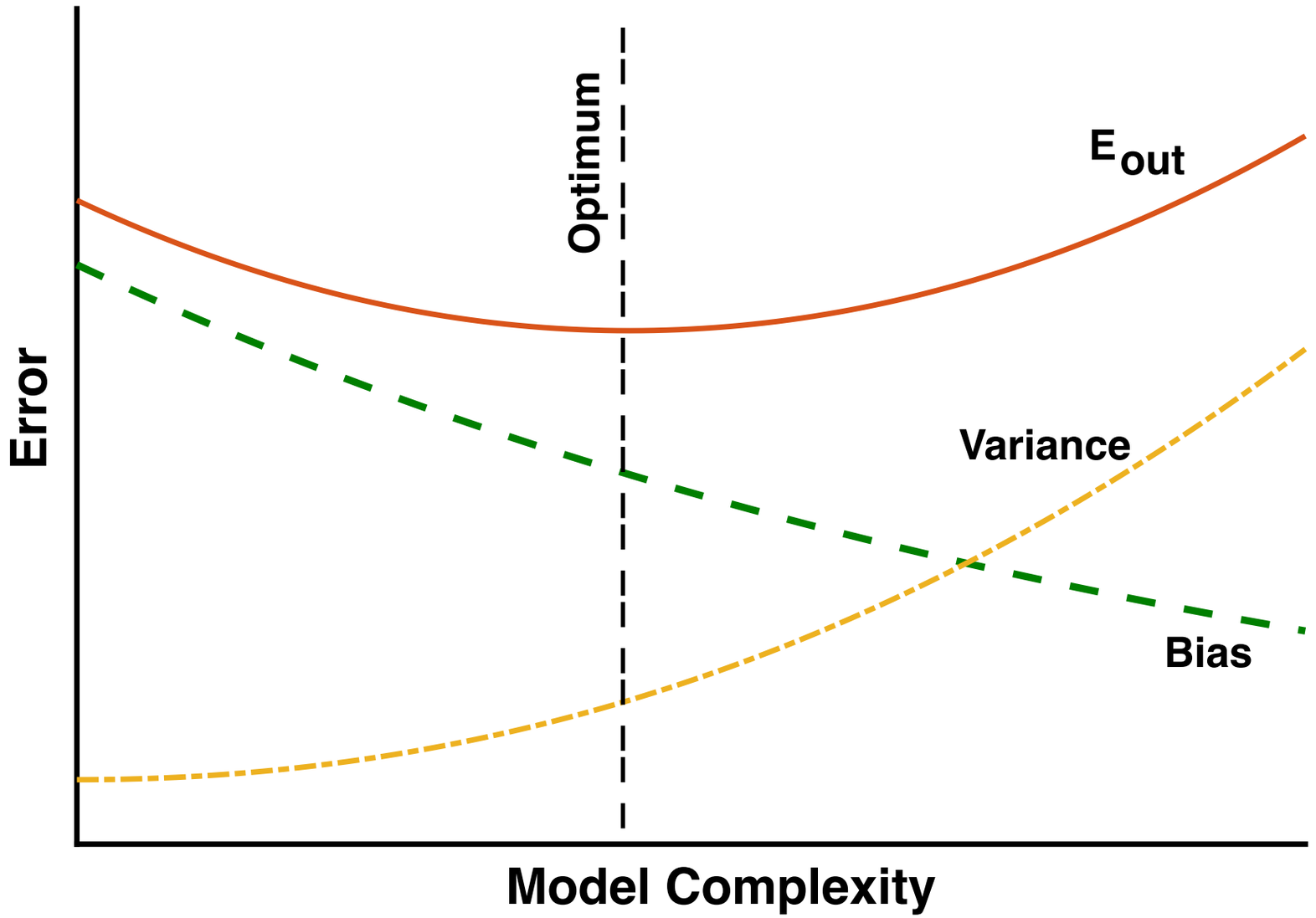
$$E_{out} = \text{Bias}^2 + \text{Variance}$$

model complexity **sampling noise**



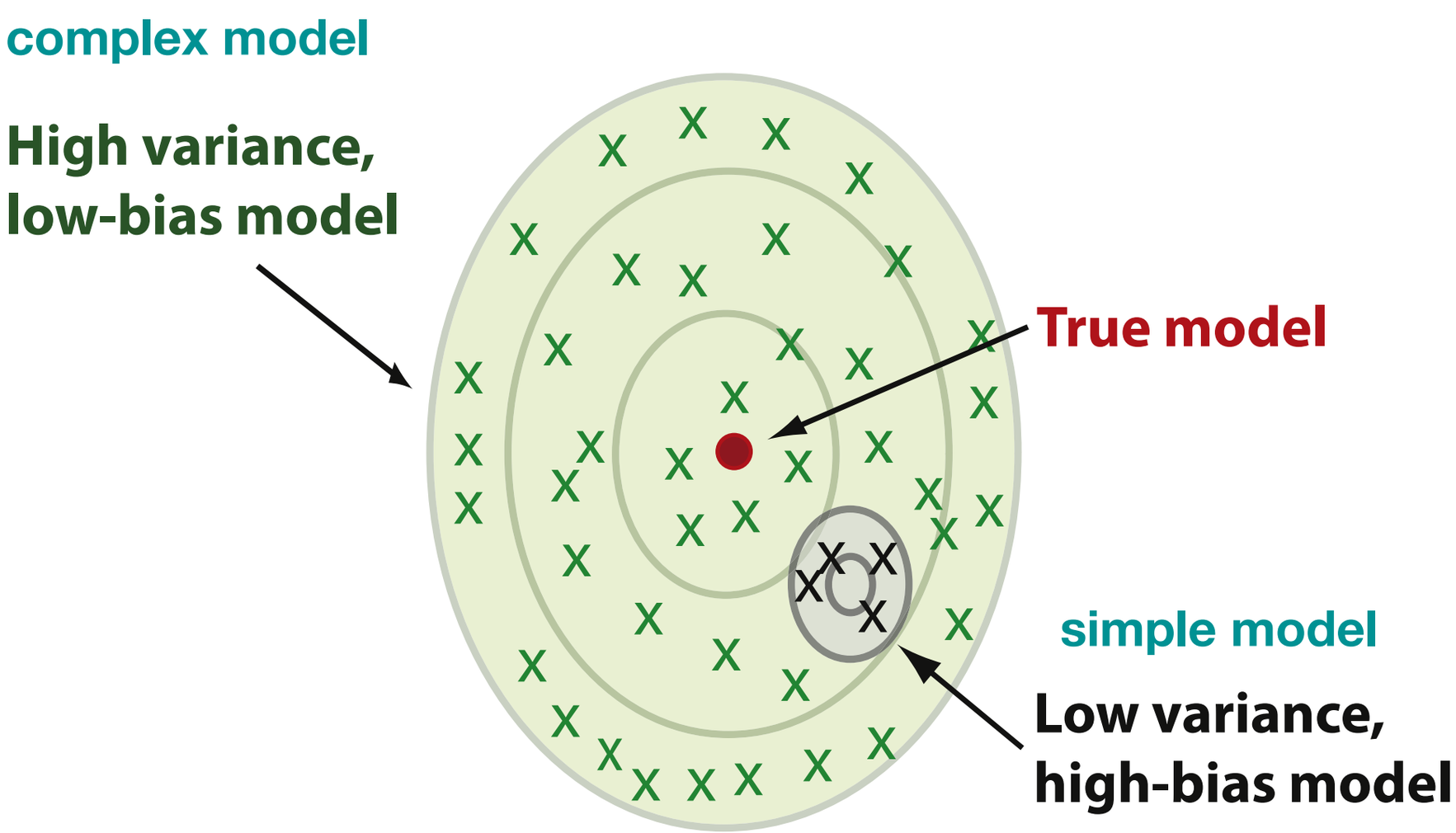
In the N → ∞ limit, the sampling noise is negligible and $E_{in} = E_{out} = \text{Bias}^2$

Bias-Variance Trade-off



Simple models have high-bias while complex models have high variance.
Models with intermediate complexity have the best performance.

Bias-Variance Trade-off



The average over all the trained complex (i.e., high variance, low-bias) models is closer to the true model.

Bias-Variance Decomposition

- Dataset D={(**X**,**y**)} obtained by drawing N samples from the true data distribution.

- True data is generated by a noisy model: $y = f(x) + \epsilon$ where ϵ is normally distributed with mean zero and standard deviation σ_ϵ.

- Cost function: $C(y, f(x, \theta)) = \sum_{i=1}^{N} \left(y_i - f(x_i; \theta)\right)^2$

- The estimate for the parameters: $\hat{\theta}_D = \arg\min_\theta C\left(y, f(\mathbf{X}; \theta)\right)$

  is a function of the dataset D. Thus we obtain a different error for each dataset. We take expectation value over all these datasets and over noise, and denote that as $\mathbb{E}_{D,\epsilon}$.

Bias-Variance Decomposition

- The expected generalization error:

$$\begin{aligned}
\mathbb{E}_{\mathcal{D},\epsilon}[\mathcal{C}(\boldsymbol{y}, f(\boldsymbol{X}; \hat{\boldsymbol{\theta}}_{\mathcal{D}}))] &= \mathbb{E}_{\mathcal{D},\epsilon}\left[\sum_i (y_i - f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}}))^2\right] \\
&= \mathbb{E}_{\mathcal{D},\epsilon}\left[\sum_i (y_i - f(\boldsymbol{x}_i) + f(\boldsymbol{x}_i) - f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}}))^2\right] \\
&= \sum_i \mathbb{E}_{\epsilon}[(y_i - f(\boldsymbol{x}_i))^2] + \mathbb{E}_{\mathcal{D},\epsilon}[(f(\boldsymbol{x}_i) - f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}}))^2] + 2\mathbb{E}_{\epsilon}[y_i - f(\boldsymbol{x}_i)]\mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}_i) - f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}})] \\
&= \sum_i \sigma_\epsilon^2 + \mathbb{E}_{\mathcal{D}}[(f(\boldsymbol{x}_i) - f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}}))^2],
\end{aligned}$$

- Further simplifying the last term:

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}[(f(\boldsymbol{x}_i) - f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}}))^2] &= \mathbb{E}_{\mathcal{D}}[\{f(\boldsymbol{x}_i) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}})] + \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}})] - f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}})\}^2] \\
&= \mathbb{E}_{\mathcal{D}}[\{f(\boldsymbol{x}_i) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}})]\}^2] + \mathbb{E}_{\mathcal{D}}[\{f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}}) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}})]\}^2] \\
&\quad - 2\mathbb{E}_{\mathcal{D}}[\{f(\boldsymbol{x}_i) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}})]\}\{f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}}) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}})]\}] \\
&= (f(\boldsymbol{x}_i) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}})])^2 + \mathbb{E}_{\mathcal{D}}[\{f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}}) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}})]\}^2].
\end{aligned}$$

Bias-Variance Decomposition

- We derived our asserted relation:

$$E_{out} = \text{Bias}^2 + \text{Variance} + \text{Noise}$$

- The **bias** defined below measures the deviation of the expectation value of our estimator from the true value:

$$Bias^2 = \sum_i (f(\boldsymbol{x}_i) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}})])^2$$

- The **variance** measures how much our estimator fluctuates due to finite-sample effects:

$$Var = \sum_i \mathbb{E}_{\mathcal{D}}[(f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}}) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}})])^2]$$

- The fundamental tension in ML between the complexity of a model and the amount of training data needed to train it.

- With a limited amount of data, it is beneficial to use a less-complex model with higher-bias (with a worse asymptotic performance).

Summary

- Typical problem in ML.
- Splitting data into training set and test set.
- In-sample error E_{in} may not equal out-of-sample error E_{out}
- **Bias-variance trade-off**:

$$< E_{out} > = \text{Bias}^2 + \text{Variance} + \text{Noise}$$

- Example: polynomial regression.