Validation

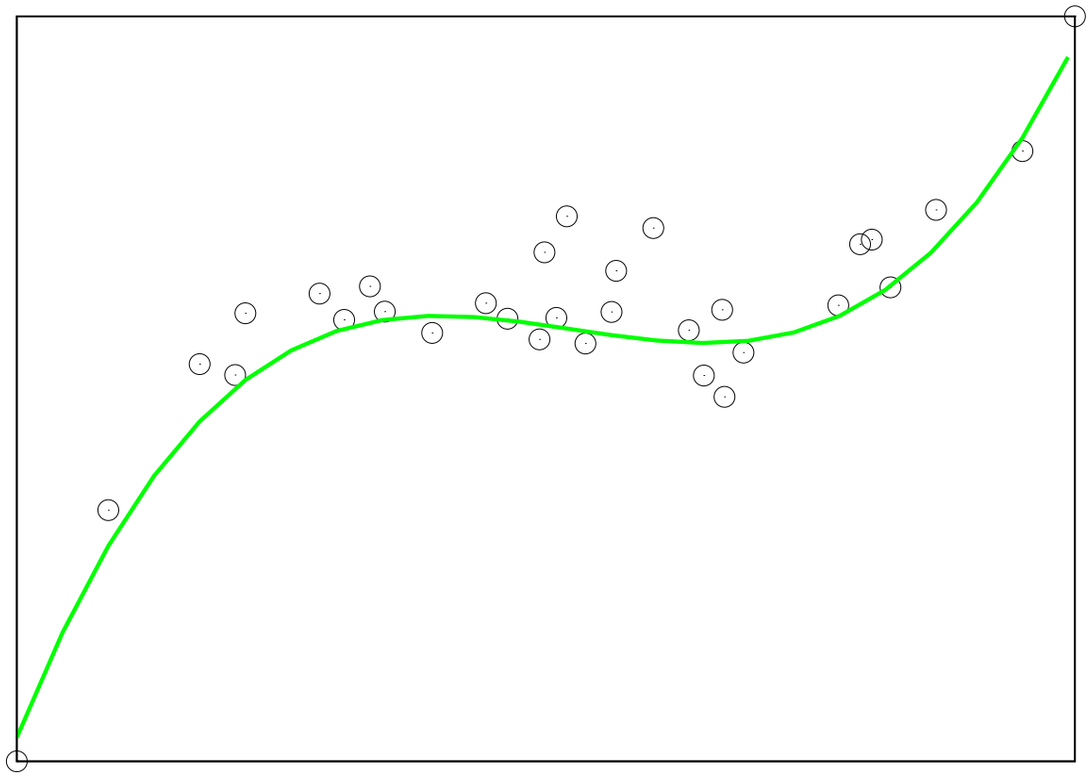
Sewoong Oh

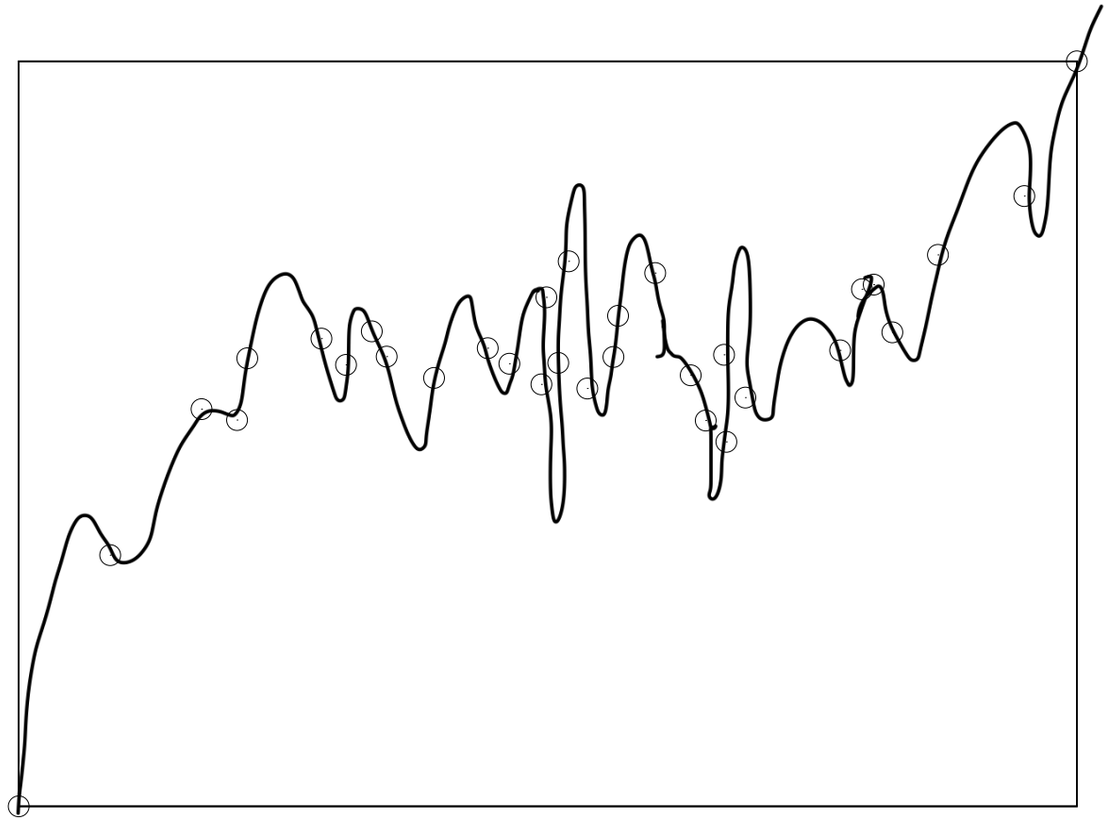
CSE/STAT 416
University of Washington

Generalization:
how do we validate which model is better?

Generalization

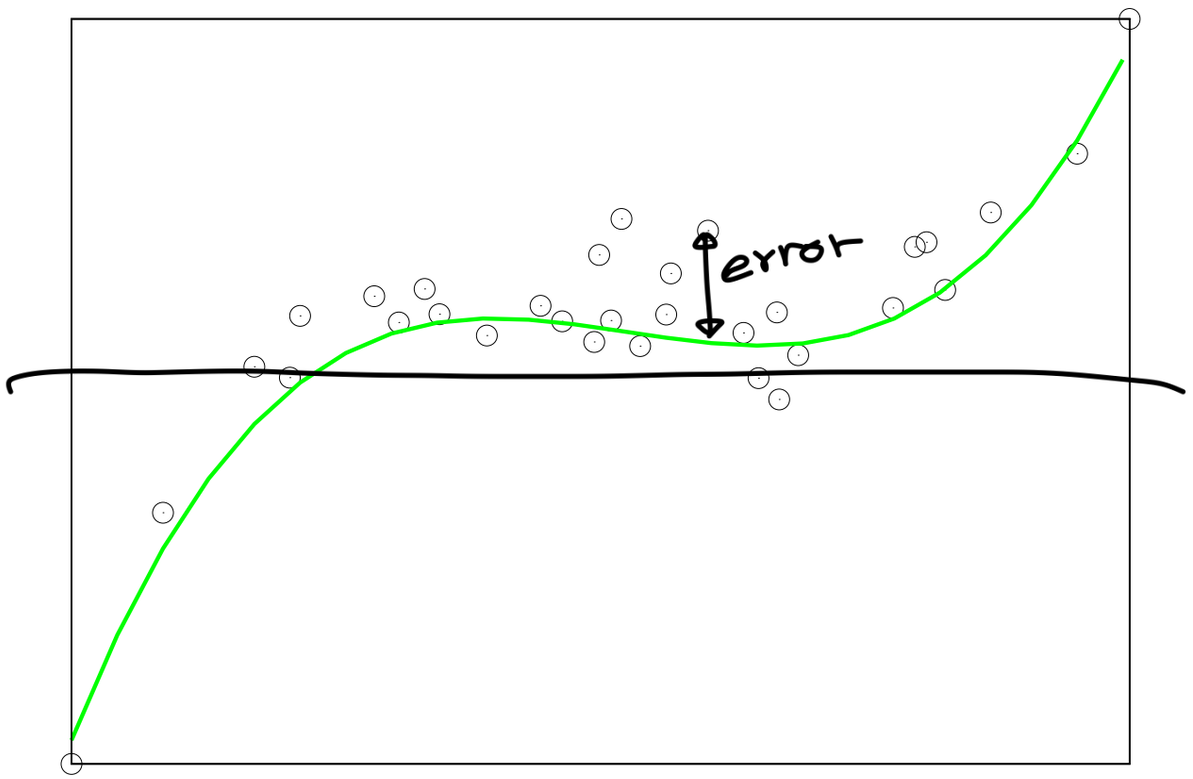
- we say a predictor **generalizes** if it performs well on unseen data
- formal mathematical definition involves probabilistic assumptions
- first, we study practical methods for assessing generalization

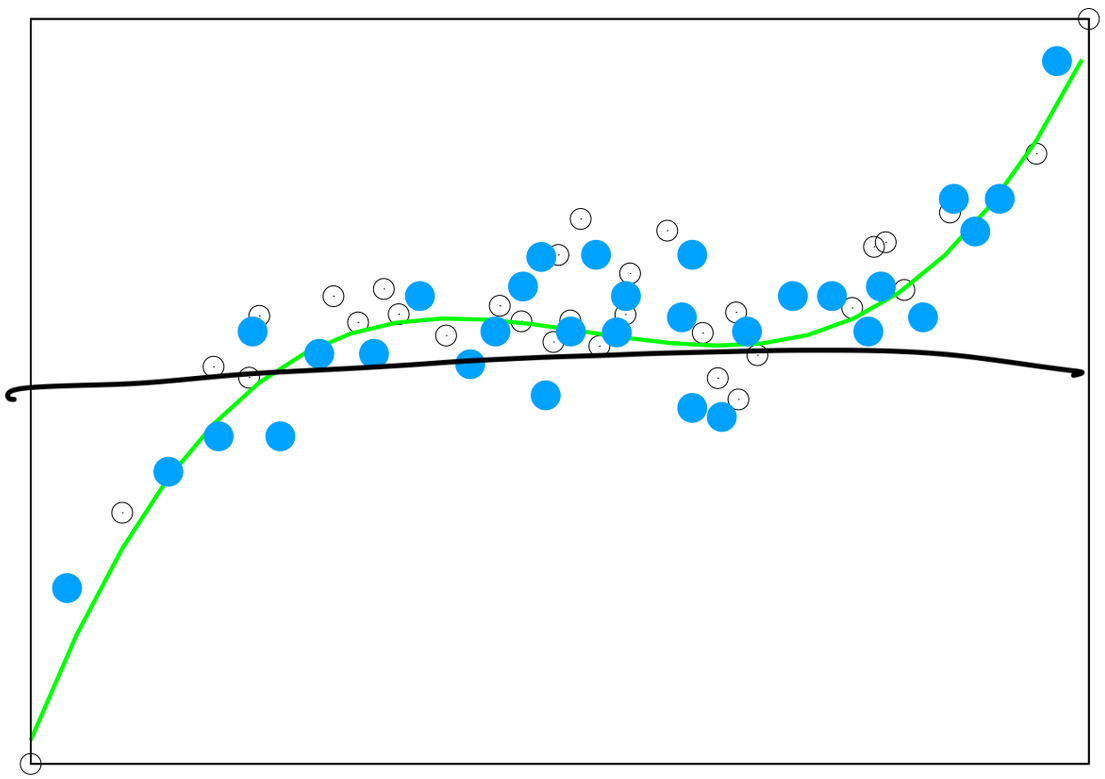




In-sample and out-of-sample data

- the data used to construct a predictor is **training data** or **in-sample data**
- we want the predictor to work on **out-of-sample data**
- we say a predictor **fails to generalize** if it does not perform well on out-of-sample data





- **train** a cubic predictor on 32 (in-sample) black circles: MSE 174
- **predict** *y* for 30 (out-of-sample) blue points: MSE 192
- conclude this predictor generalizes: in-sample MSE $\approx$ out-of-sample MSE

Validation

- a way to mimic how the predictor performs on unseen data
- key idea: divide the data into two set for **training** and **testing**

- **training set** used to construct (“train”) the predictor
- **test set** or **validation set** used to evaluate the predictor
  - based on the assumption that test set is similar to unseen data

Validation

- we use **training error** for optimization (or finding the model)

$$\text{MSE}_{\text{train}} = \frac{1}{|S_{\text{train}}|} \sum_{i \in S_{\text{train}}} (f(x_i) - y_i)^2$$

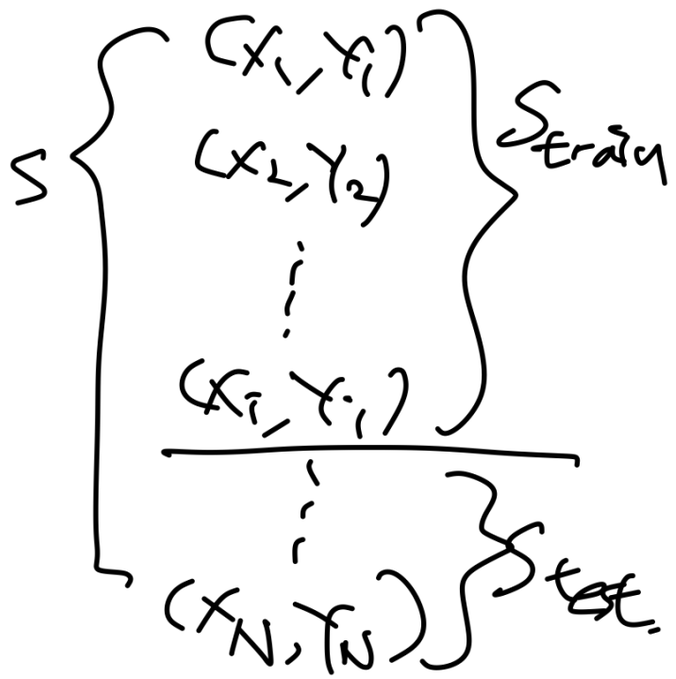
- we use **test error** for validation

$$\text{MSE}_{\text{test}} = \frac{1}{|S_{\text{test}}|} \sum_{i \in S_{\text{test}}} (f(x_i) - y_i)^2$$

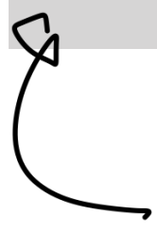
- selecting train/test sets should be **random** (80/20 or 90/10 are common)

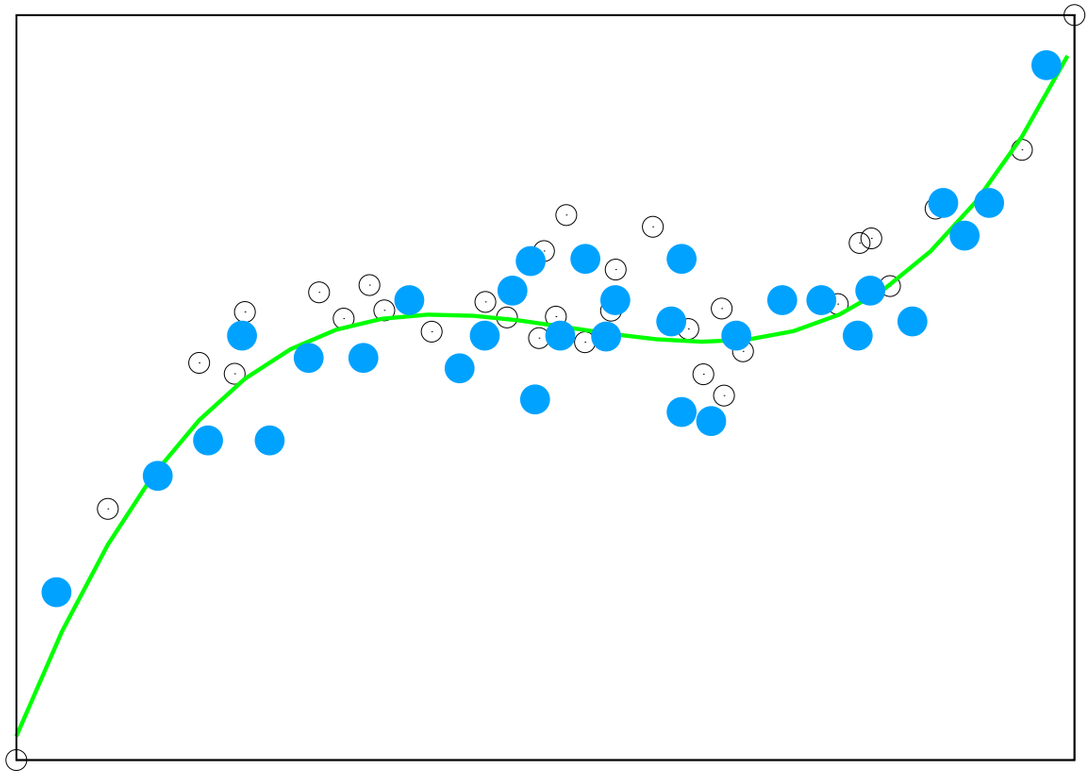
- we say a model or predictor is **overfit** if

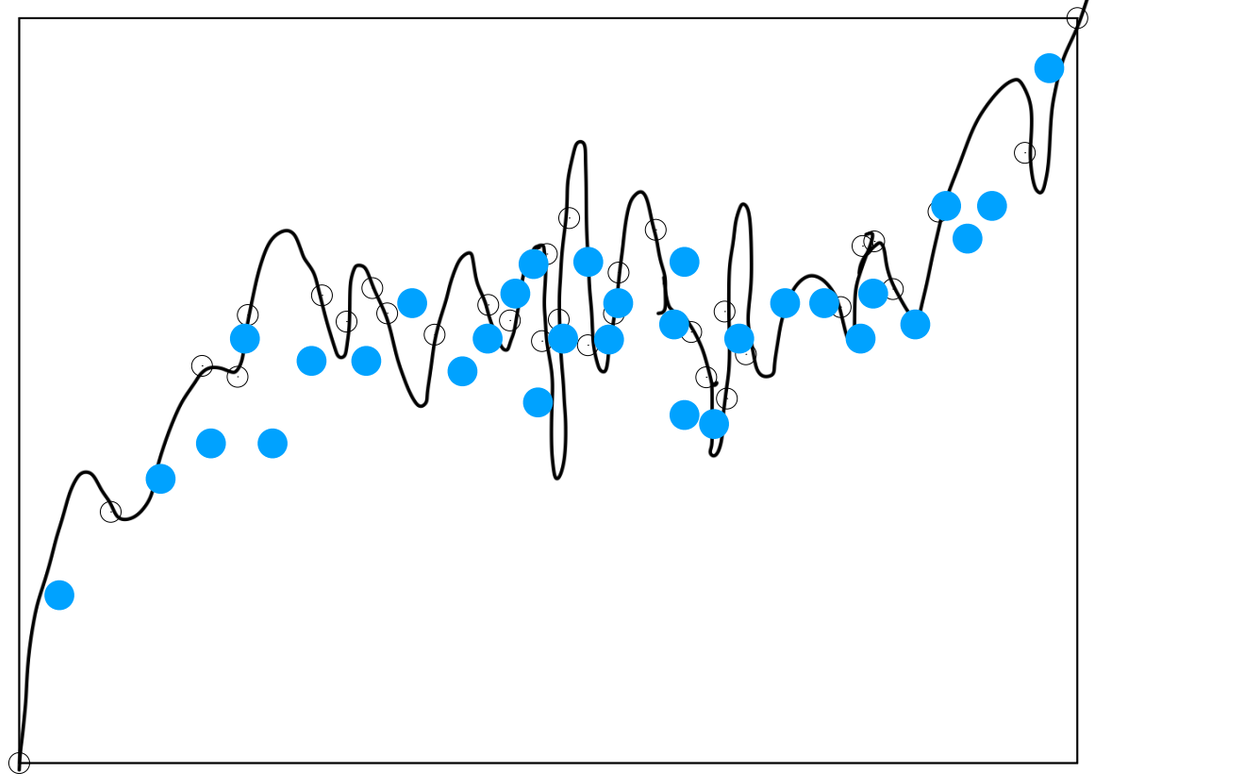
$$\text{MSE}_{\text{test}} \gg \text{MSE}_{\text{train}}$$



	small training error	large training error
small test error	model performs well generalizes well	?
large test error	fit data well no generalization	perform poorly generalize well







$w_0 + w_1 x + \text{----} + w_p x^p$

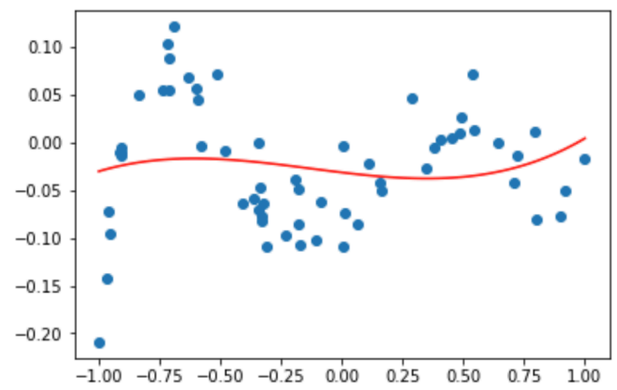
- between two models, the one with smaller test error should be chosen

Overfitting

- a model that fits the training data well but performs poorly on test data suffers from **overfitting**
- overfitting happens if we use a model with high **model complexity**
- for example, for linear regression with polynomial features

$$\hat{y} = f(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_p x^p$$

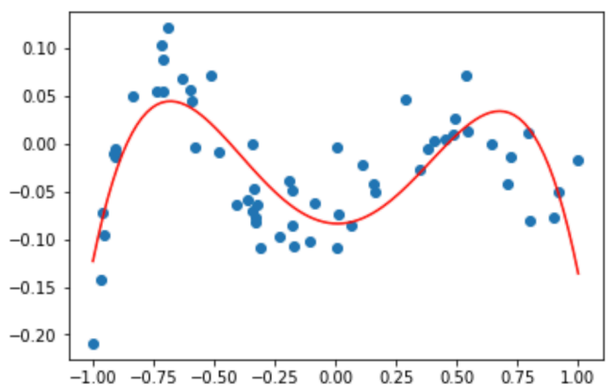
- N = 60 data points, and $p \in \{3, 4, 5, 20\}$



0.004011490884146126

0.003831010290504173

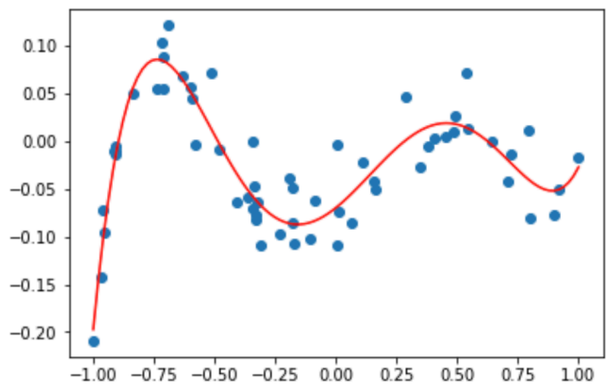
degree 3



0.0019177229761741974

0.002200492720447942

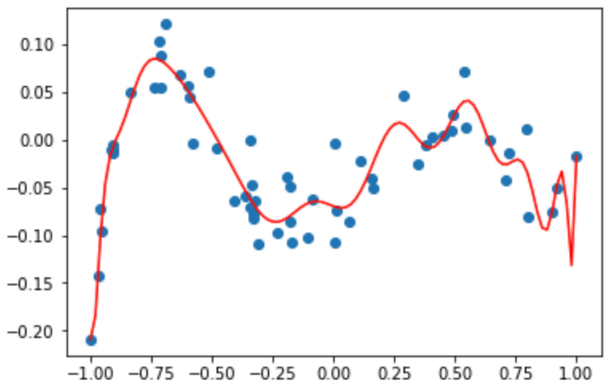
degree 4



0.0007426853089970962

0.0010239915404334238

degree 5

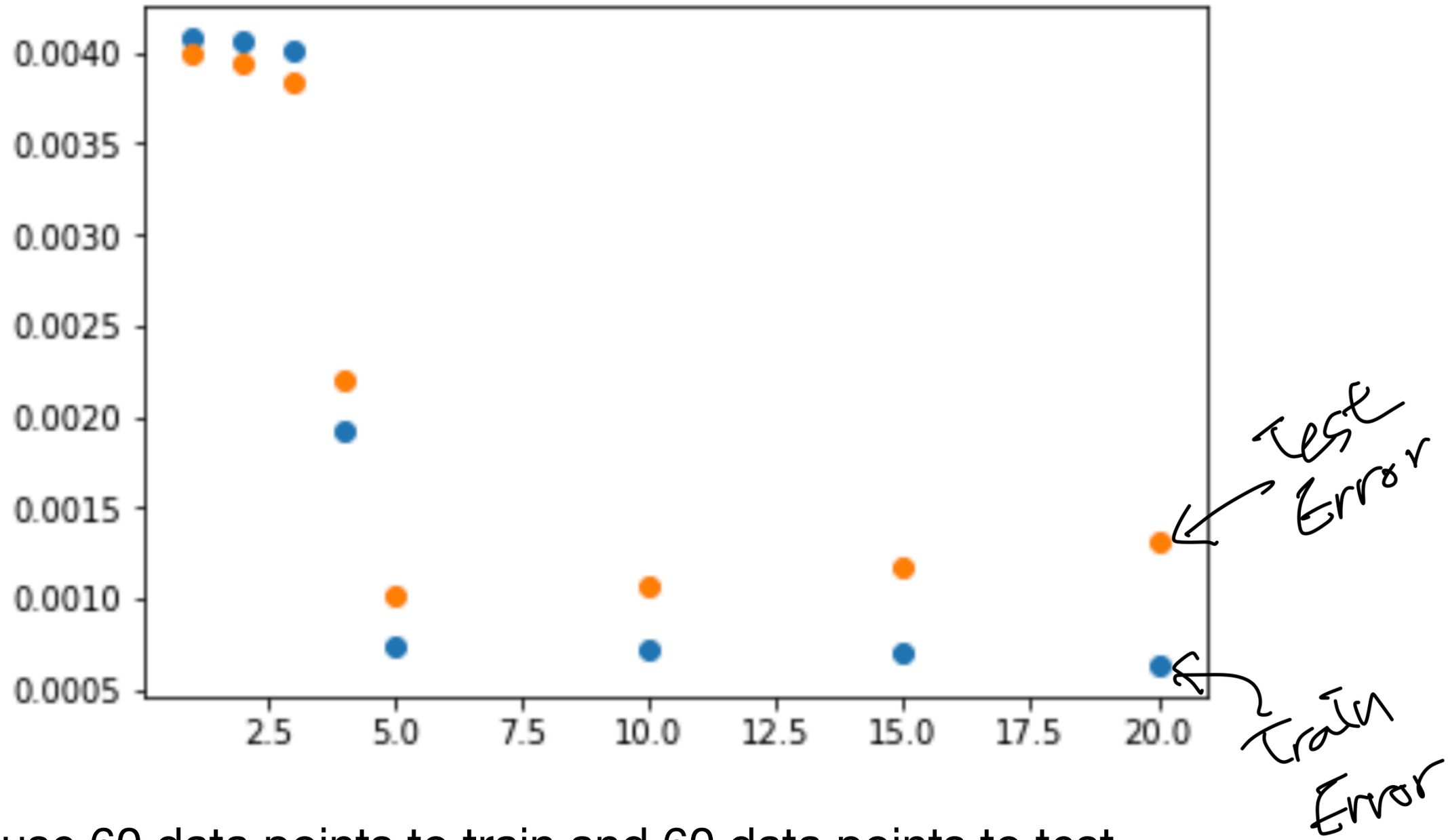


0.0006350545819906561 $\text{MSE}_{\text{train}}$

0.0013118370515986446 MSE_{test}

degree 20

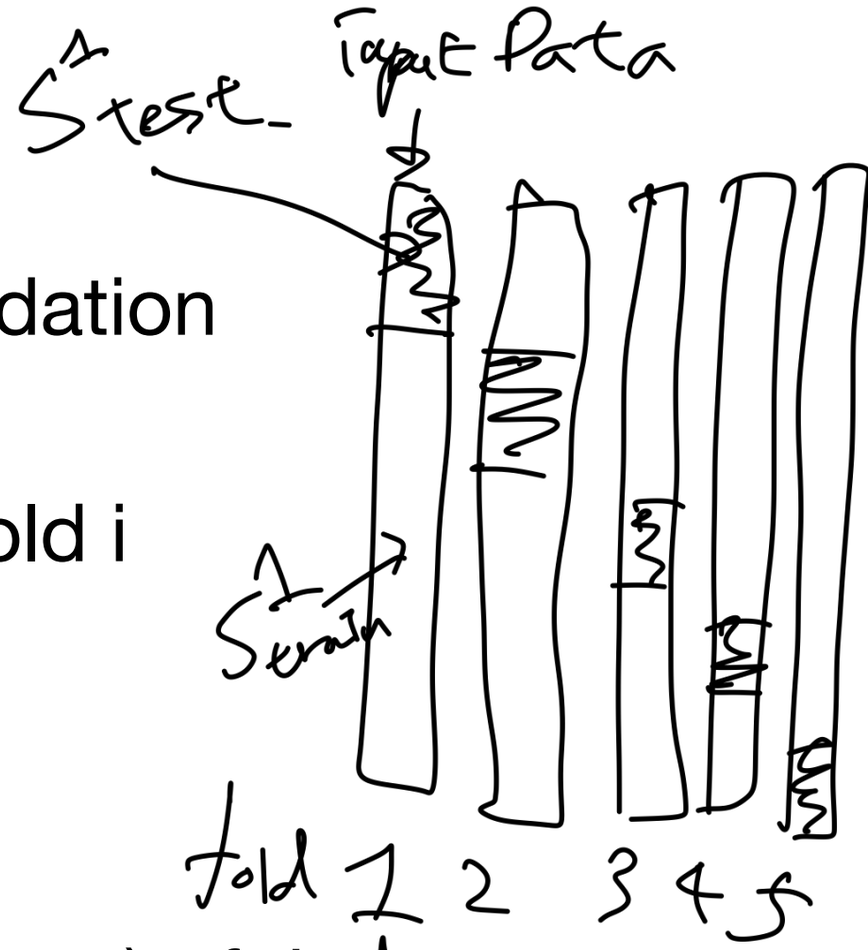
How does one choose which model to use?



- first use 60 data points to train and 60 data points to test
- then choose degree 5 as per the above test error
- now re-train on all 120 data points with degree 5 polynomial model

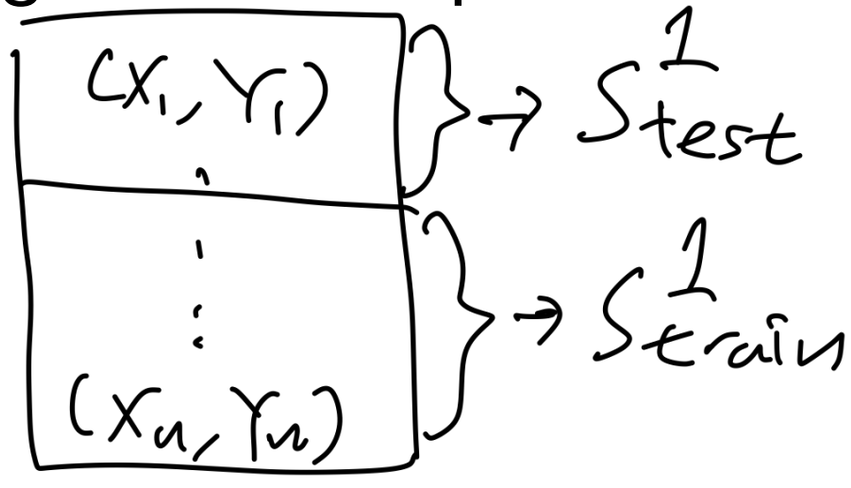
Cross validation

- systematic method for out-of-sample validation
  - divide the data into *k* **folds**
  - for each i, fit predictor on all data but fold i
  - compute test error on fold i
  - average the test error across the folds



- gives some idea of the variability (or variance) of the test error

- we can estimate **stability** of the ML pipeline, by comparing the model parameters found in each fold



pseudo code for k-fold cross validation

- Input: data $D=\{(x_i,y_i)\}_{i=1,\ldots,N}$, number of folds **k**, model
- Output: trained model parameters **w**, training error, test error

- D <- shuffle(D)
- D_1=D[0:N/k],D_2=D[N/k:2N/k],…
- For j=1,…,k do
  - w <- fit({D_1,…,D_{j-1},D_{j+1},…,D_k})
  - $\mathrm{MSE}_{\mathrm{train}}^{(j)} \leftarrow \frac{k}{(k-1)N} \sum_{i \in \text{training data}} (w^T x_i - y_i)^2$
  - $\mathrm{MSE}_{\mathrm{test}}^{(j)} \leftarrow \frac{k}{N} \sum_{i \in D_j} (w^T x_i - y_i)^2$
- $\mathrm{MSE}_{\mathrm{train}} \leftarrow \frac{1}{k} \sum_{j=1}^{k} \mathrm{MSE}_{\mathrm{train}}^{(j)}$
- $\mathrm{MSE}_{\mathrm{test}} \leftarrow \frac{1}{k} \sum_{j=1}^{k} \mathrm{MSE}_{\mathrm{test}}^{(j)}$

- shuffle the data randomly
- split the data into k groups
- repeat for each fold
- train on the rest of data
- compute training error
- compute test error

How to choose k?

- k = 5 to 10 seems to work well in practice

- small k leads to overestimating the test error
  - because we are training on much smaller data size

- if k=N it is called Leave-one-out (LOO) cross validation

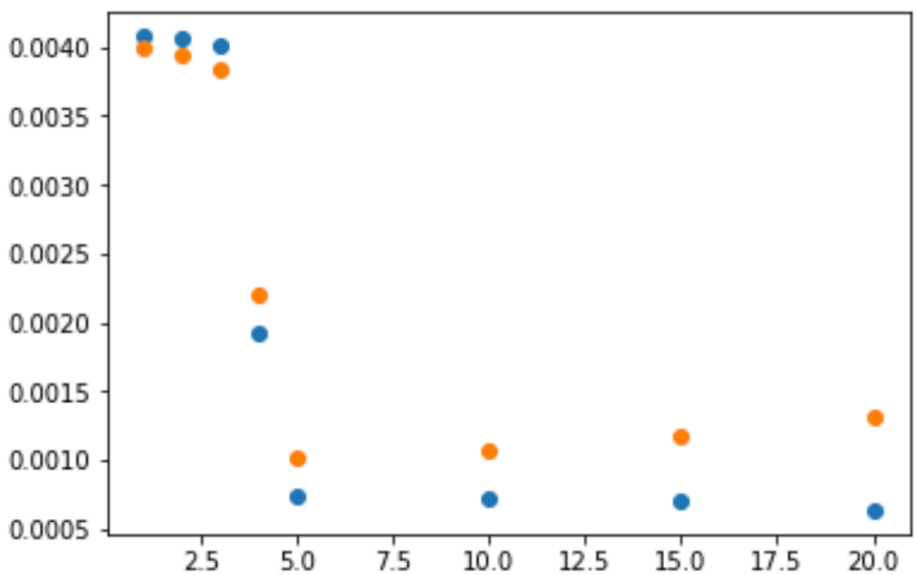
- it takes longer to cross validate for large k

Model selection via (cross) validation

- whether you use cross validation or just validation on a single split data, you can compute the $\mathrm{MSE}_{\mathrm{train}}$ and $\mathrm{MSE}_{\mathrm{test}}$

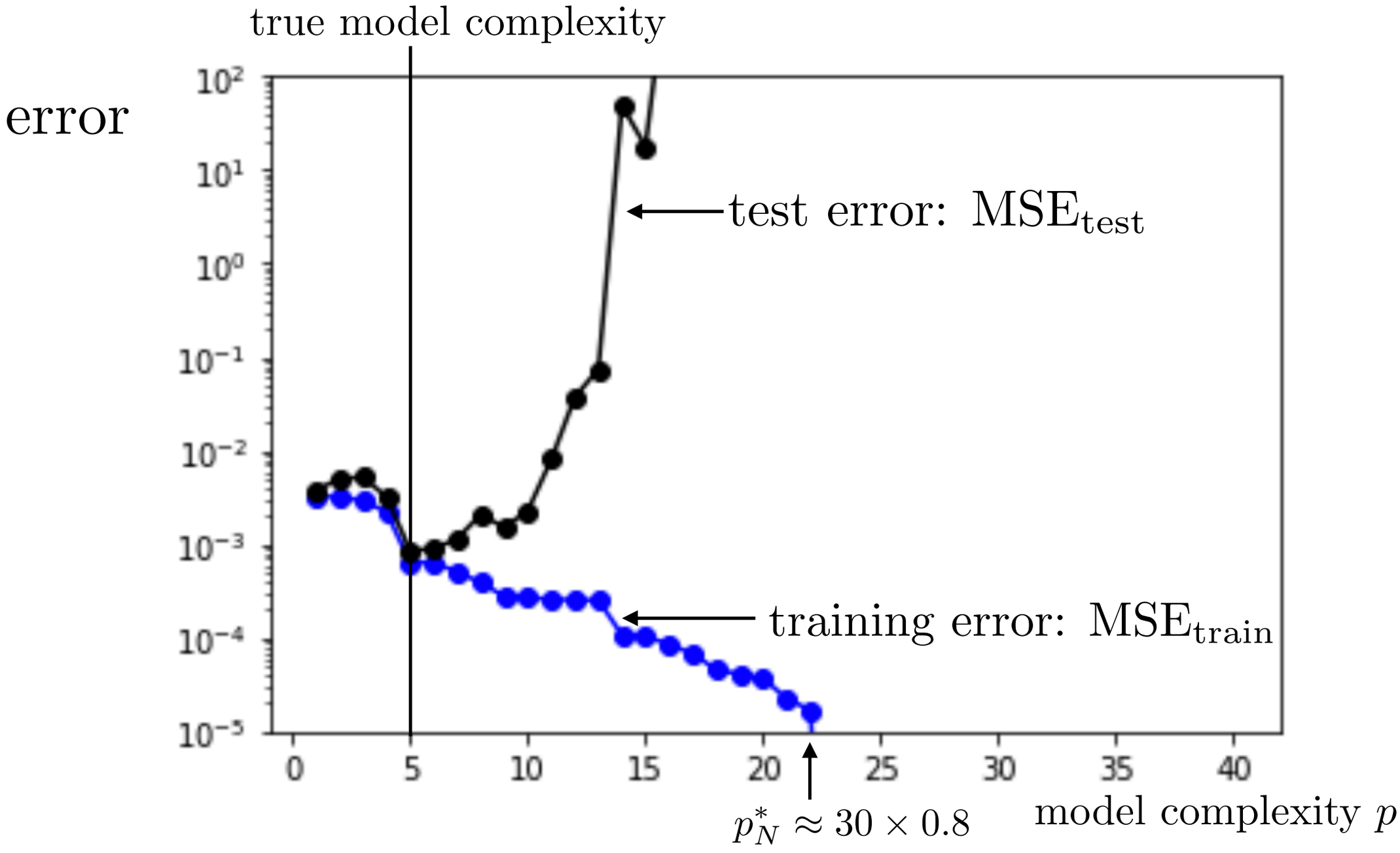
- such validation this requires:
  - dataset of N samples
  - choice of k, if using cross validation
  - choices of model classes to be validated

  (e.g. linear regression with quadratic loss, and polynomial features of degrees $p = 3, 4, 5, 10, 15,$ and 20)



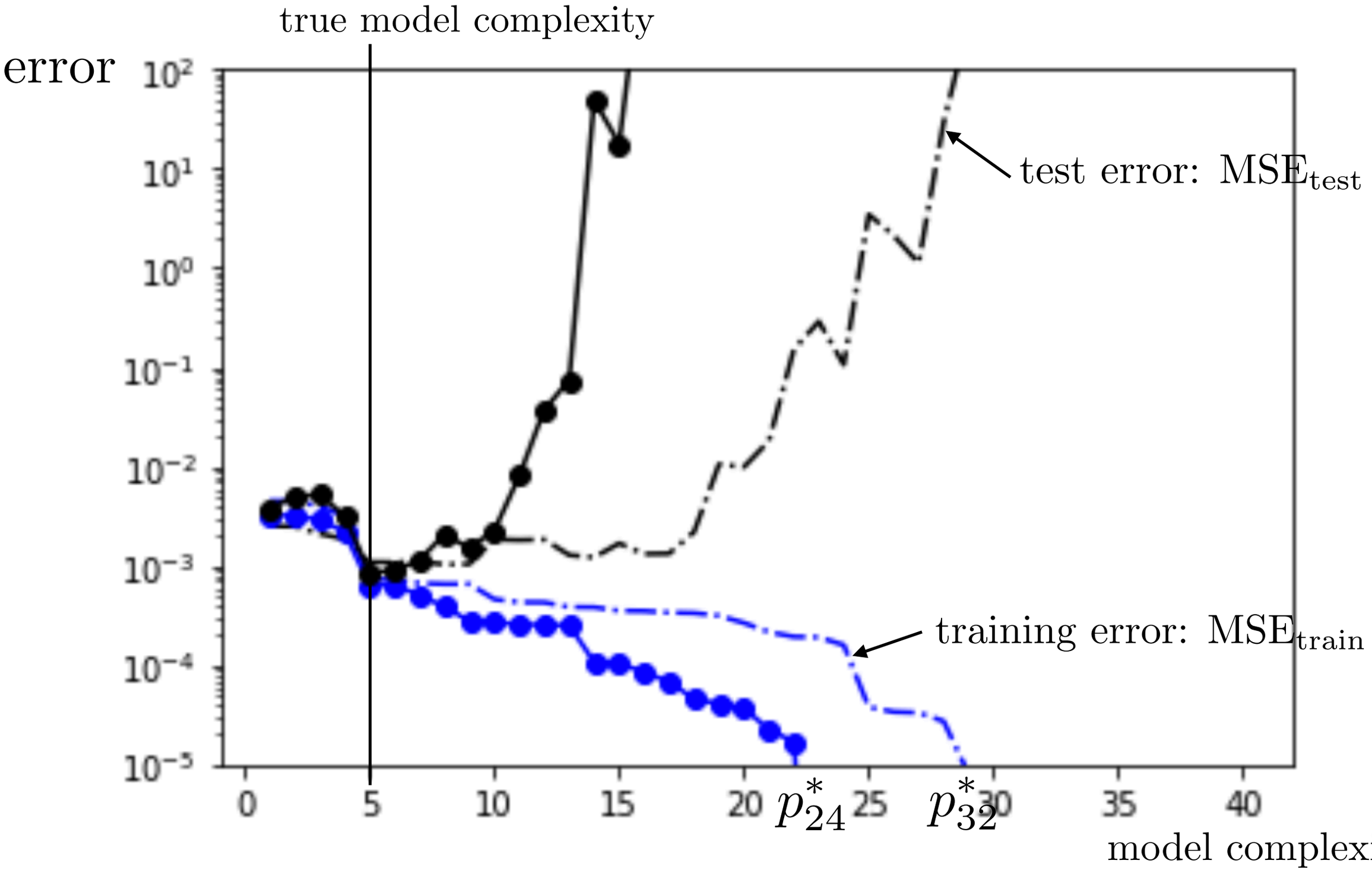
- ignoring ***k*** for now, as it is not too sensitive to this choice as long as it is large enough, the computed MSEs are functions of sample size ***N*** and model complexity ***p***

- let us first fix sample size ***N=30***, collect one dataset of size N, randomly shuffle the dataset, and fix one training set S_{train} and test set S_{test} via 80/20 split
- then we run multiple validations and plot the computed MSEs for all values of ***p*** that we are interested in



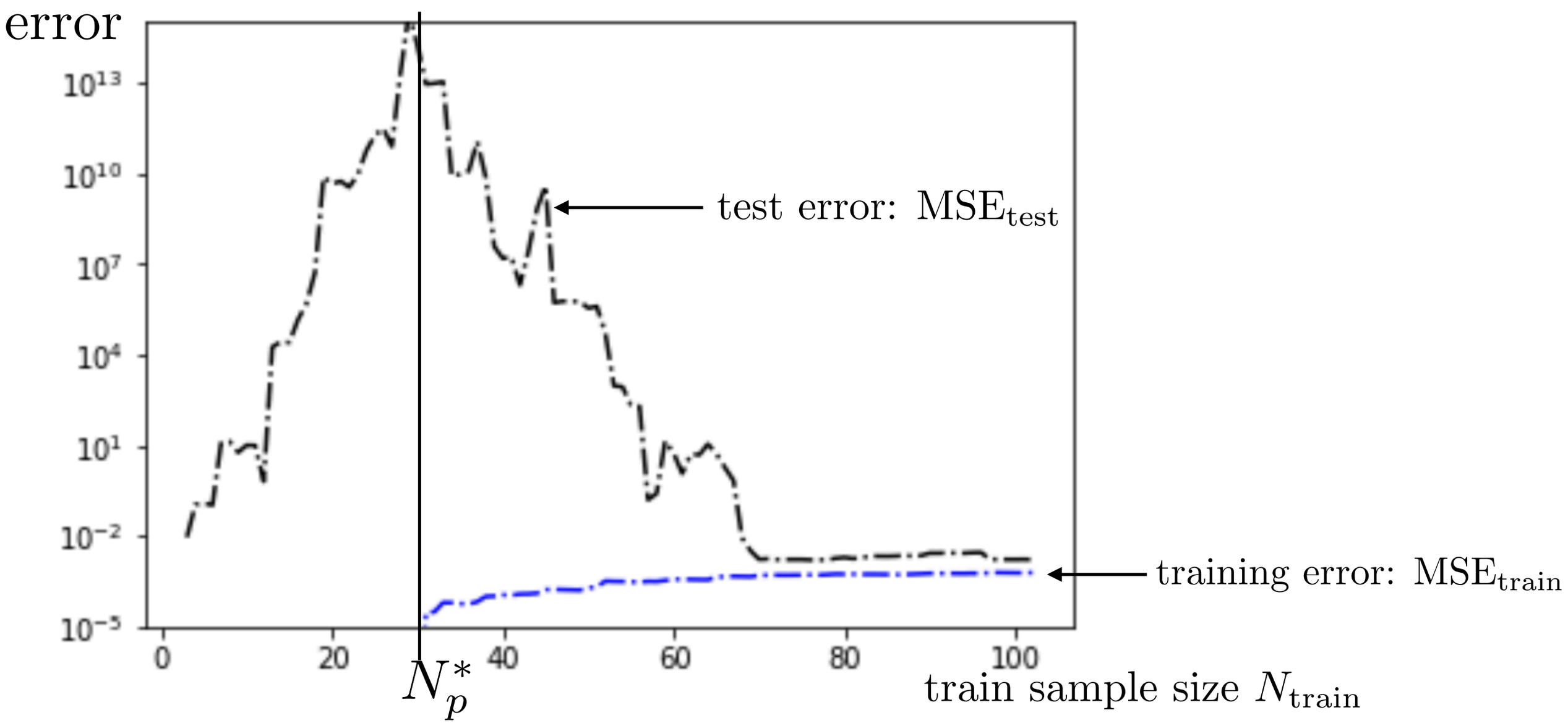
- Given sample size N there is a threshold where training error is zero
- Training error is **always** monotonically non-increasing
- Test error has a trend of going down and up, but fluctuates

- let us now repeat the process changing the sample size to **N=40** , and see how the curves change



- The threshold moves right
- Training error tends to increase: more points need to fit
- Test error tends to decrease: with more samples complex models can be fit well

- let us now fix model complexity ***p=30***, collect multiple datasets by starting with 3 samples and adding one sample at a time to the training set, but keeping a large enough test set fixed
- then we run multiple validations and plot the computed MSEs for all values of train sample size N_{train} that we are interested in



- There is a threshold below which training error is zero (extreme overfit)
- Below this threshold, test error is meaningless
- Test error tends to decrease
- Training error tends to increase

From practice to theory

Why does test error go down and up with increasing model complexity?

$\mathrm{MSE}_{\mathrm{test}}(w) = \frac{1}{N}\sum_{i=1}^{N}(\underbrace{f_w(x_i)}_{\hat{y}_i} - \underbrace{(f_0(x_i)+\varepsilon_i)}_{y_i})^2$ is a random variable

to understand the behavior, we consider the expected error:

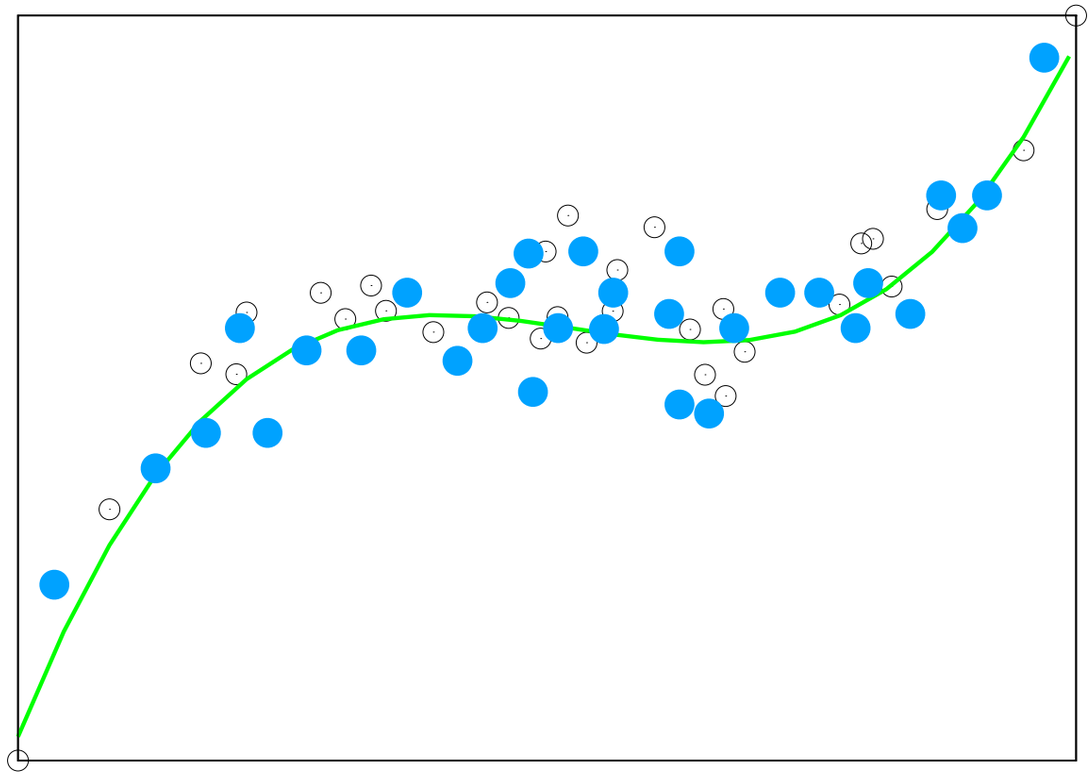
$$
\begin{aligned}
\overline{\mathrm{MSE}_{\mathrm{test}}(x_i)} &= \frac{1}{|W|}\sum_{w\in W}(f_w(x_i)-(f_0(x_i)+\varepsilon_i))^2 \\
&= \mathbb{E}[(f_w(x_i)-(f_0(x_i)+\varepsilon_i))^2] \\
&= \mathrm{Var}[f_w(x_i)-(f_0(x_i)+\varepsilon_i)] + \mathbb{E}[f_w(x_i)-(f_0(x_i)+\varepsilon_i)]^2 \\
&= \mathrm{Var}[f_w(x_i)-f_0(x_i)] + \mathrm{Var}[\varepsilon_i] + \mathbb{E}[f_w(x_i)-f_0(x_i)]^2 \\
&\qquad \mathrm{MSE}_{\mathrm{variance}} \qquad \mathrm{MSE}_{\mathrm{noise}} \qquad \mathrm{MSE}_{\mathrm{bias}}
\end{aligned}
$$

Why does test error go down and up with increasing model complexity?

- three sources of error: **noise**, **bias**, and **variance**

$$\mathrm{MSE}_{\mathrm{true}} = \mathrm{MSE}_{\mathrm{noise}} + \mathrm{MSE}_{\mathrm{bias}} + \mathrm{MSE}_{\mathrm{variance}}$$

- error from **noise** in the data cannot be reduced

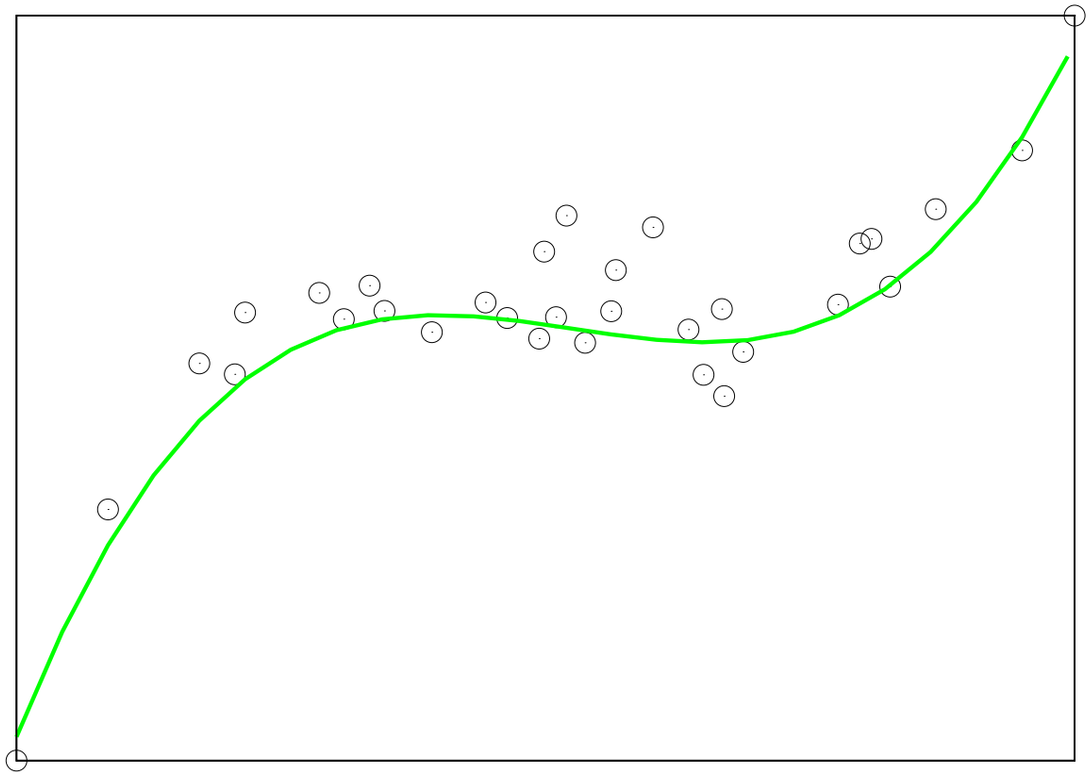


$$y = f_0(x) + \varepsilon$$

$$\mathrm{MSE}_{\mathrm{noise}} = \mathbb{E}[\varepsilon^2]$$

Low complexity models

- suppose we train a constant function, many times each with *N* samples from $y = f_0(x) + \varepsilon$

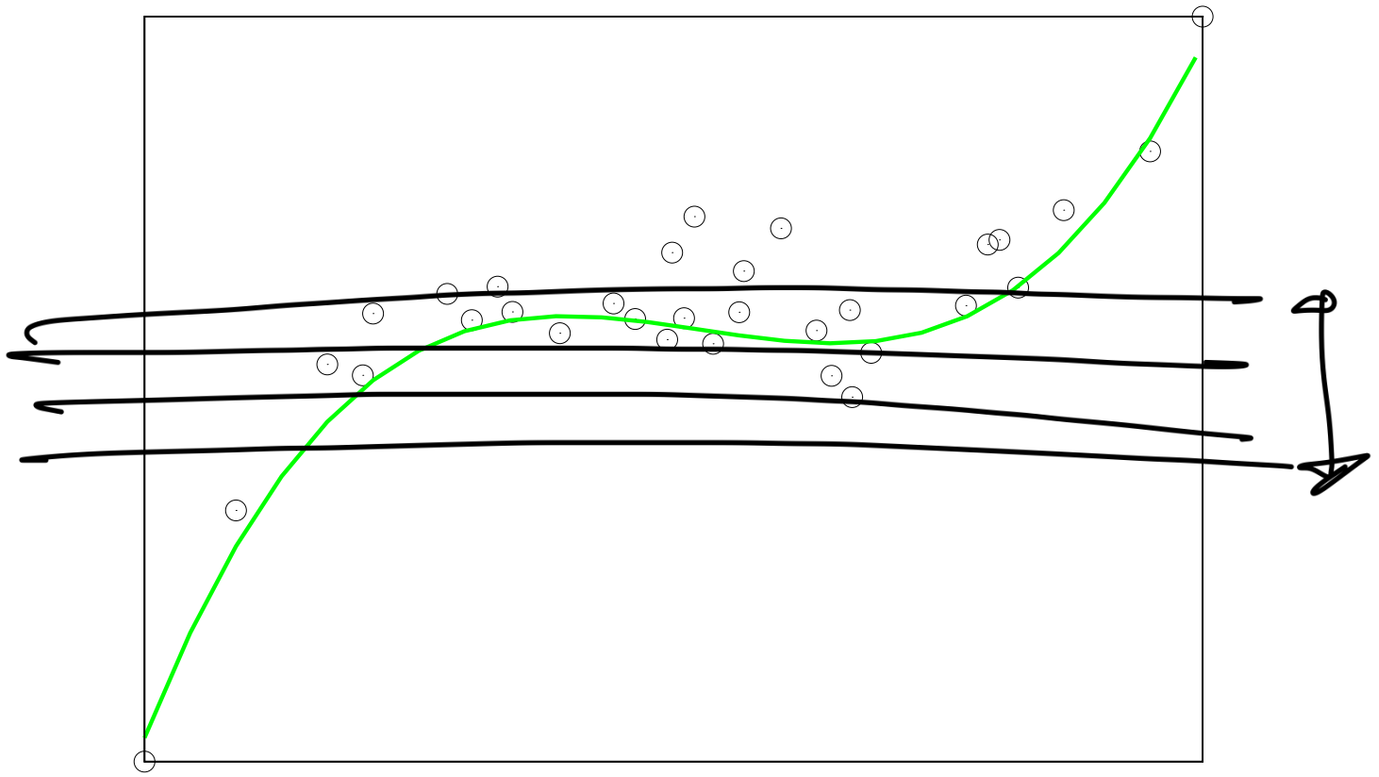


$$\mathrm{MSE}_{\mathrm{bias}}(x_i) = \mathrm{Bias}(x_i)^2 \text{ where } \mathrm{Bias}(x_i) = \mathbb{E}[f_w(x_i) - f_0(x_i)]$$

- low complexity model has a large **bias**

Low complexity models

- suppose we train a constant function, many times each with *N* samples from $y = f_0(x) + \varepsilon$

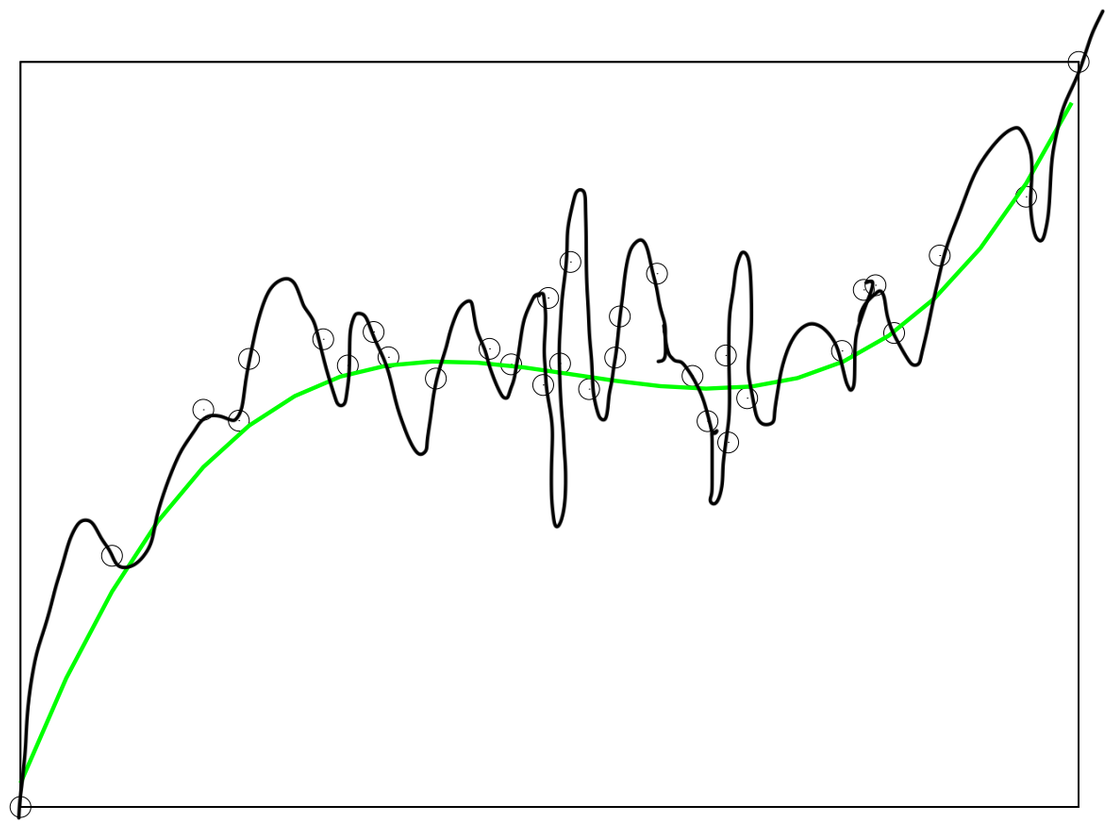


$$\mathrm{MSE}_{\mathrm{variance}}(x_i) = \mathrm{Var}[f_w(x_i)]$$

- low complexity model has a small **variance**

High complexity models

- suppose we train a high-degree polynomial function, many times each with *N* samples from $y = f_0(x) + \varepsilon$



$w_0 + w_1 x + \cdots + w_p x^p$

$$\mathrm{MSE}_{\mathrm{variance}}(x_i) = \mathrm{Var}[f_w(x_i)]$$

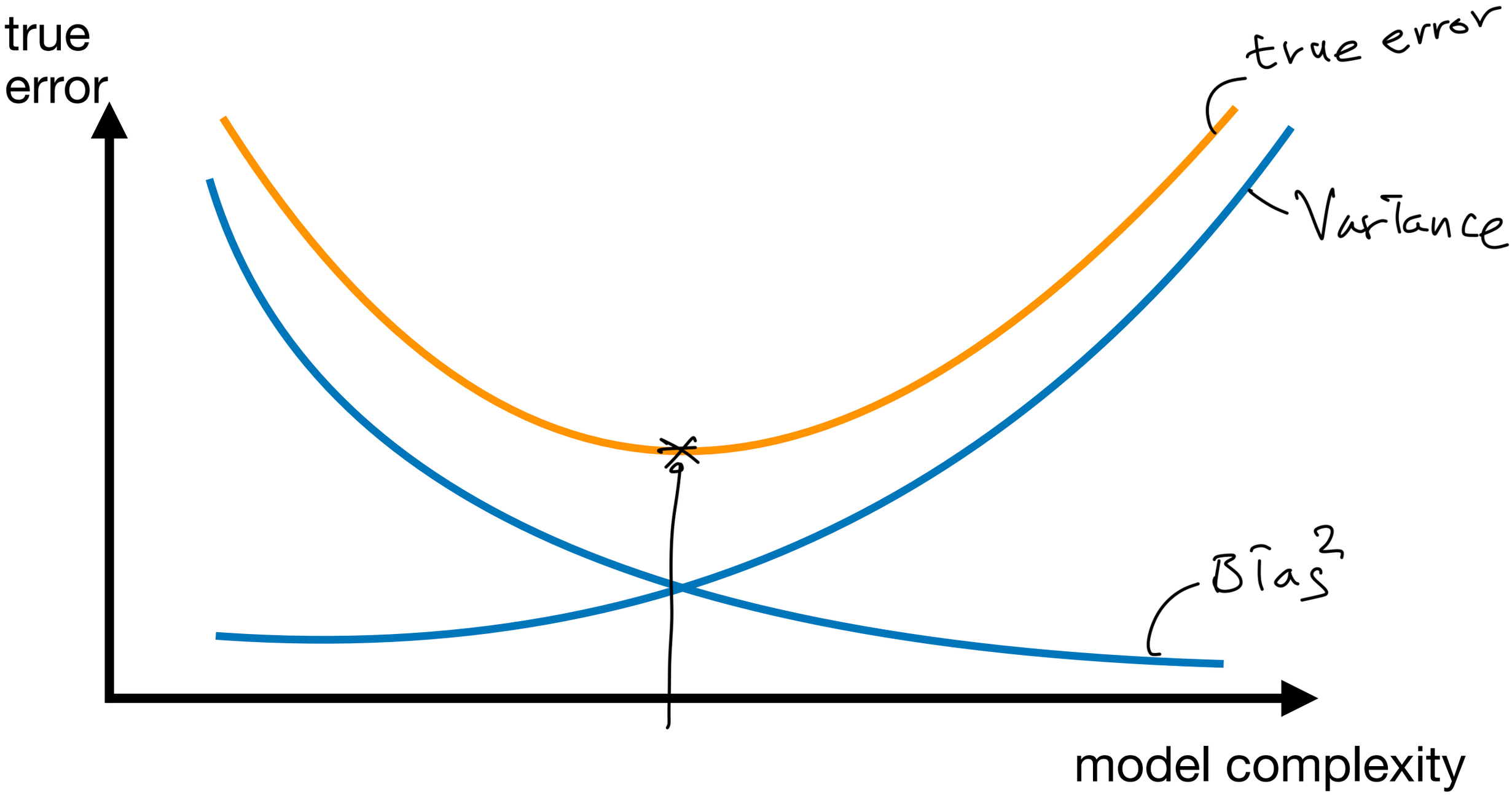
$$\mathrm{MSE}_{\mathrm{bias}}(x_i) = \mathrm{Bias}(x_i)^2 \text{ where } \mathrm{Bias}(x_i) = \mathbb{E}[f_w(x_i) - f_0(x_i)]$$

- high complexity model has a large **variance** but small **bias**

Bias-Variance tradeoff

- for fixed sample size *N*,

$$\mathrm{MSE}_{\mathrm{true}} = \mathrm{MSE}_{\mathrm{noise}} + \mathrm{MSE}_{\mathrm{bias}} + \mathrm{MSE}_{\mathrm{variance}}$$



For fixed model complexity

- suppose we fix model complexity such that

$$f_0(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5 + \varepsilon$$

$$f(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$