Model Selection and Overfitting

Ninghao Liu

University of Georgia

February 8, 2024

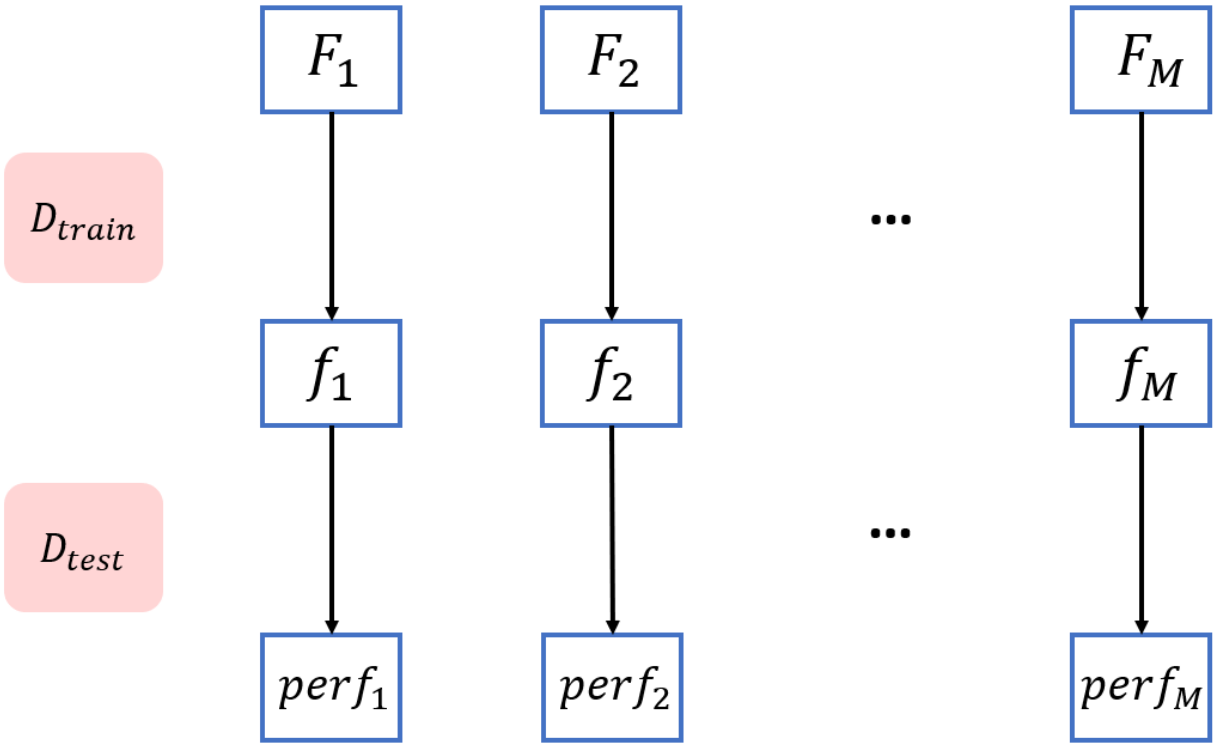
Till now ...

We have learned multiple models, such as linear models, NBC, decision trees.

Also, for the same model type, different hyper-parameters could lead to different model instantiation.

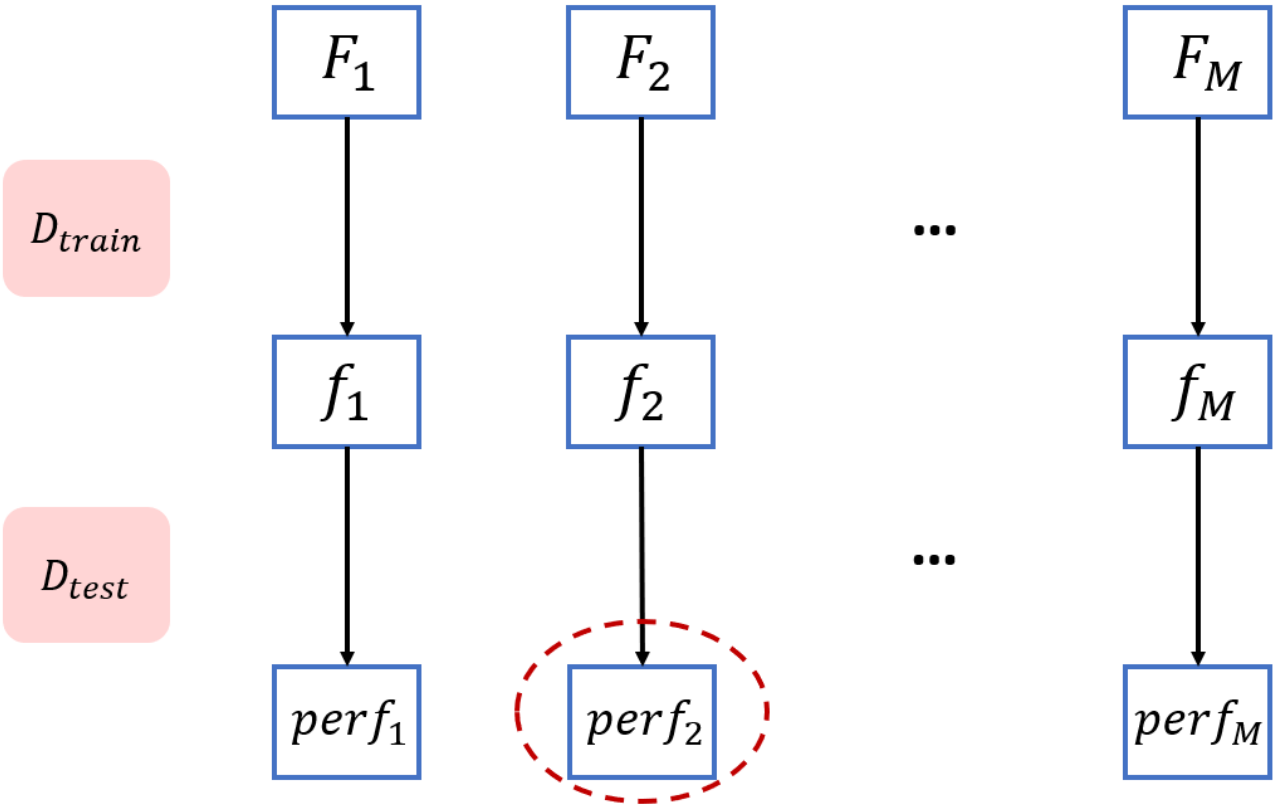
- How to select the best model for a given task?
- Are complex models always better than simple models?

Model Selection



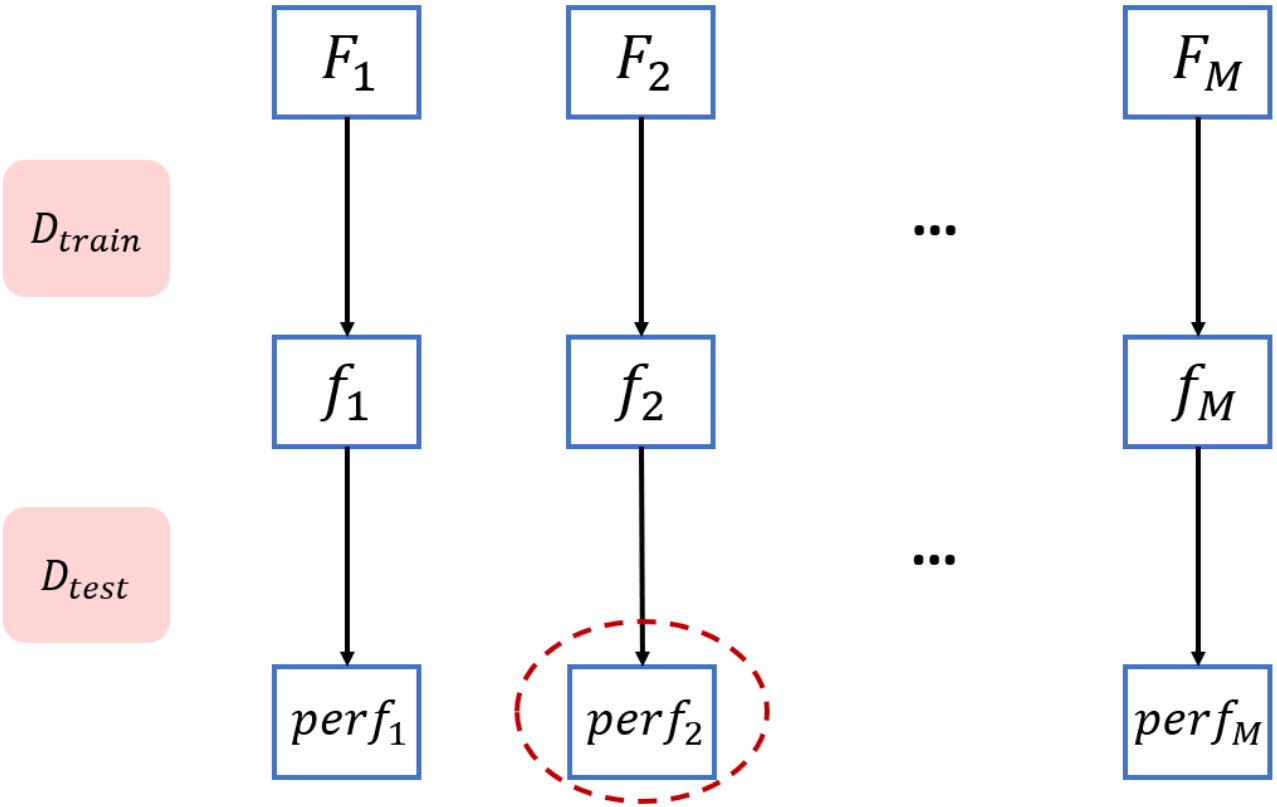
- Given the whole dataset, we split it into a training set and a test set. The test set is used to evaluate model performances.

Model Selection



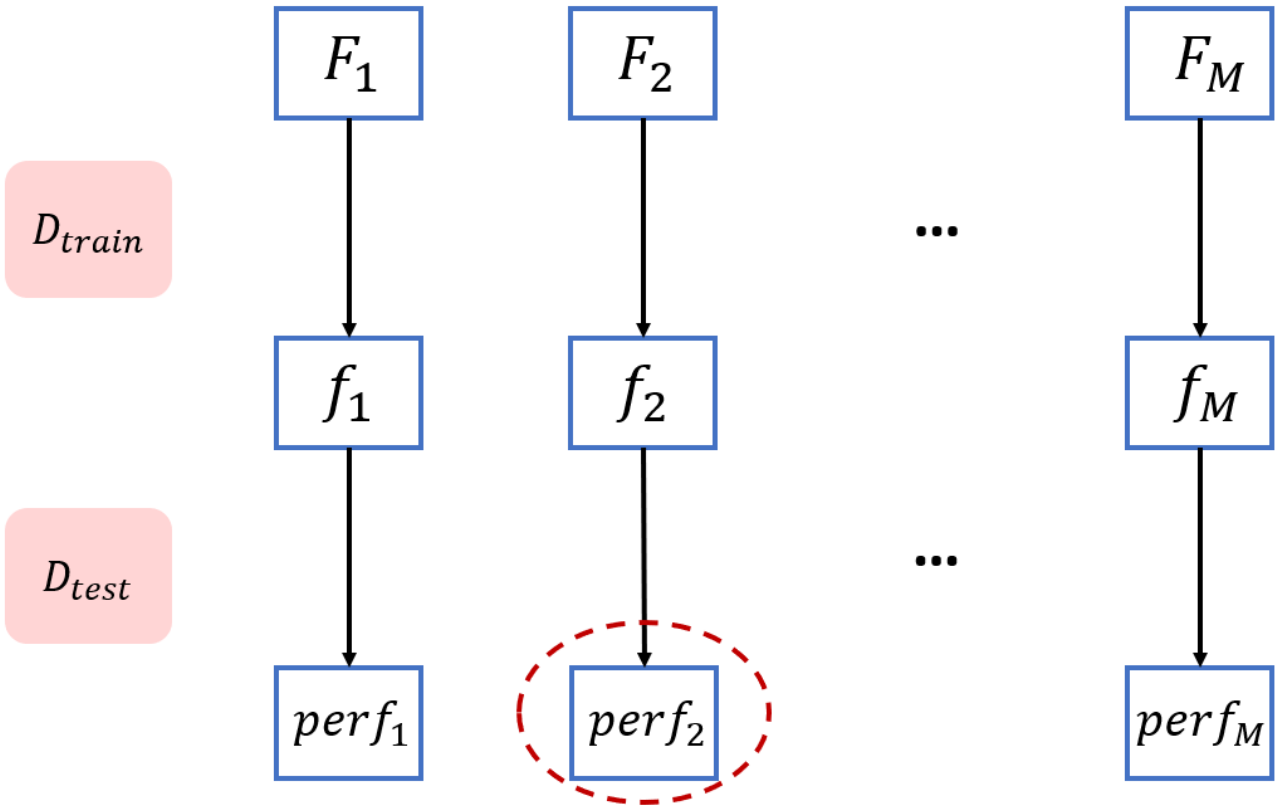
- Claim 1: We should select the model having the best test performance.

Model Selection



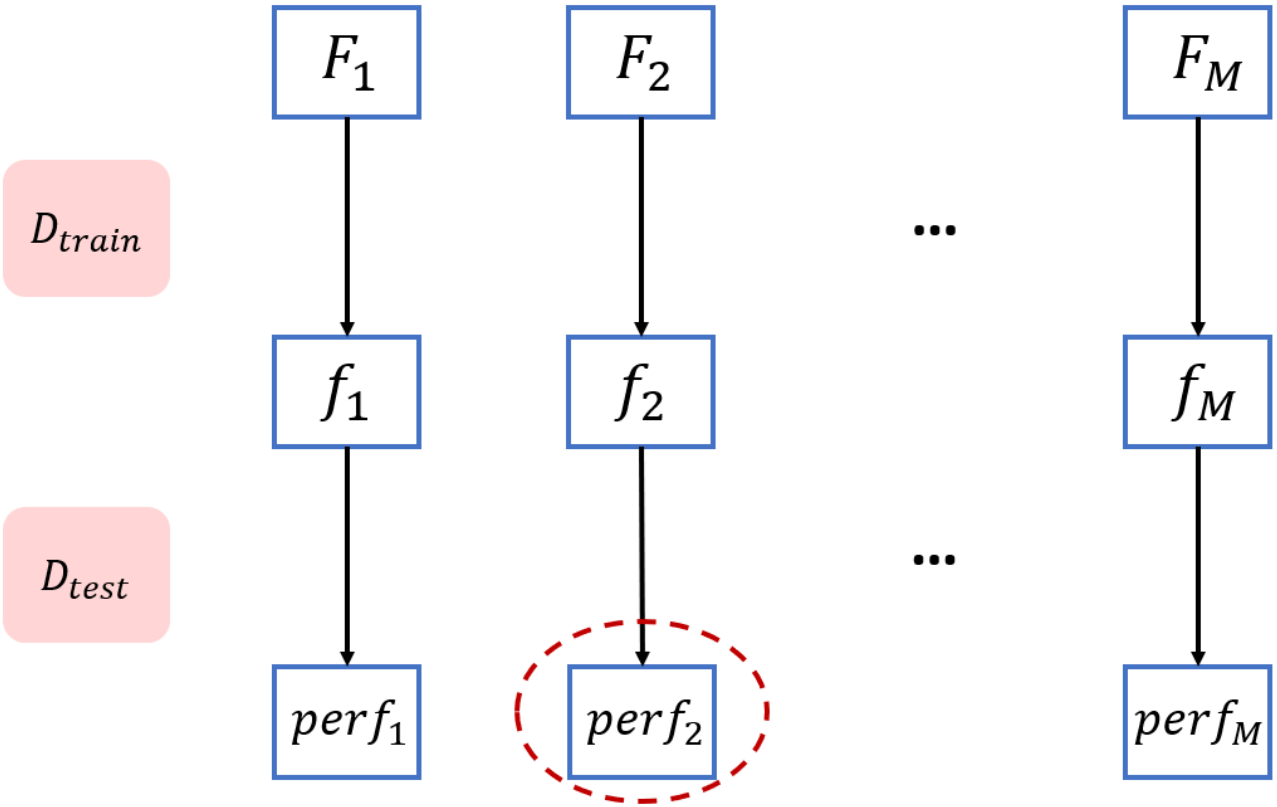
- Claim 2: If we use test performance as the evidence for model selection, after deploying the selected model, the expected model performance will be the test performance.

Model Selection



- Claim 2: If we use test performance as the evidence for model selection, after deploying the selected model, the expected model performance will be the test performance. (wrong!)

Bias of the winner



The model performance after deployment is usually worse than the test performance.

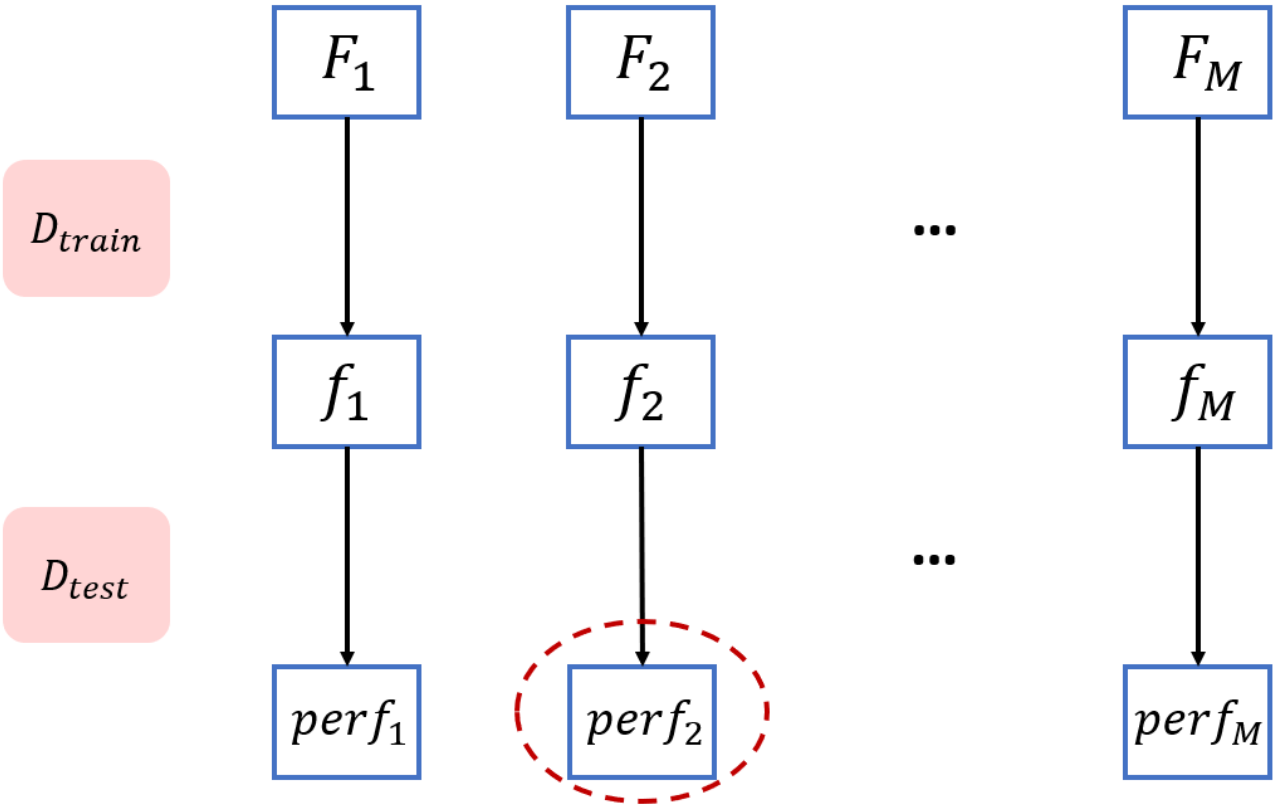
- Optimistic bias.

Bias of the winner

Let's do a simulation.

- Assume we have 100 models.
- These models have equal capability. The expected performance is 0.7.
- Due to the randomness of model training and testing, the test performance could vary.
- We always choose the model with the best test performance. We assume the best test performance will reflect the true performance of the model after deployment.
- Let's see what will happen through a Python simulation program.

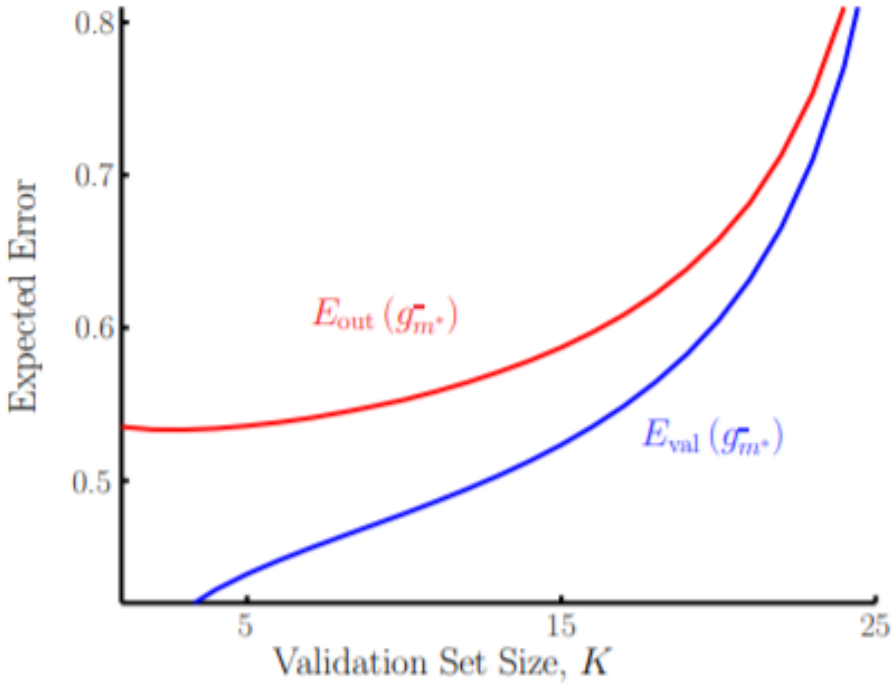
Bias of the winner



Optimistic bias.

- The winner could be just lucky.

Bias of the winner



There is a gap between the real test performance (red) and the performance estimated by the best test performance (blue) during model selection.

- The gap is smaller as the validation set grows (but will affect training).

A new question

Then, how to estimate the performance of a selected model after deployment?

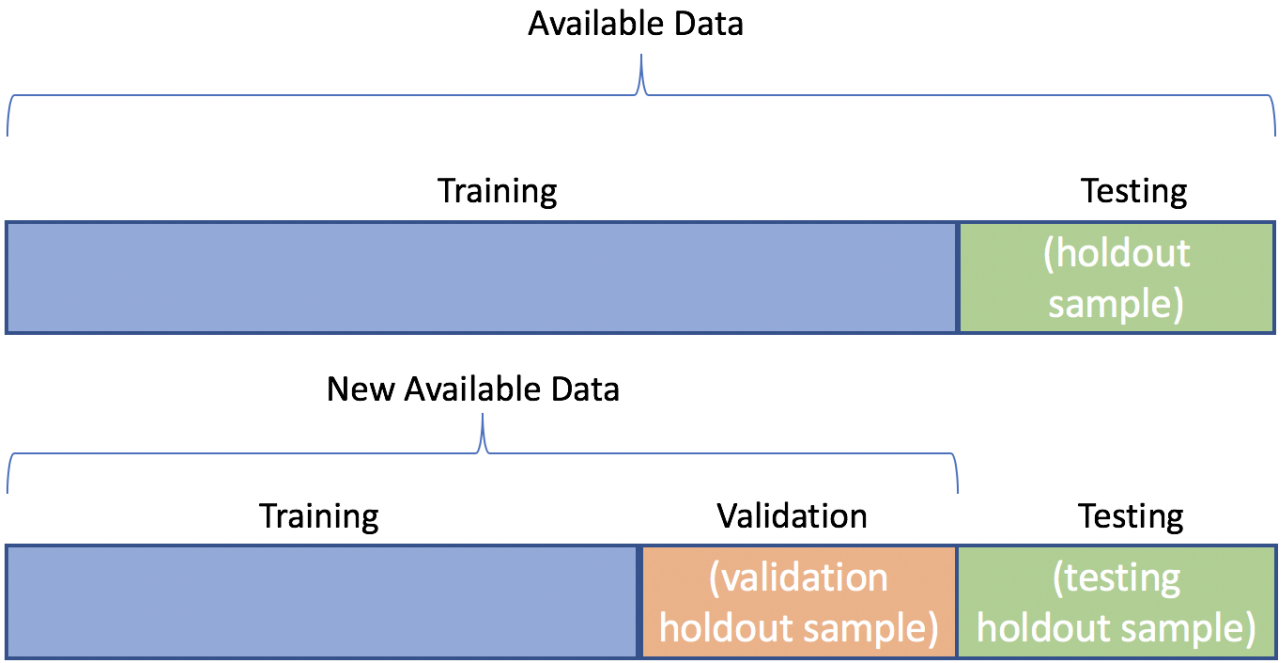
- We do not have any data left...

A new question

Then, how to estimate the performance of a selected model after deployment?

- We do not have any data left ...
- We wish there is another test dataset ...

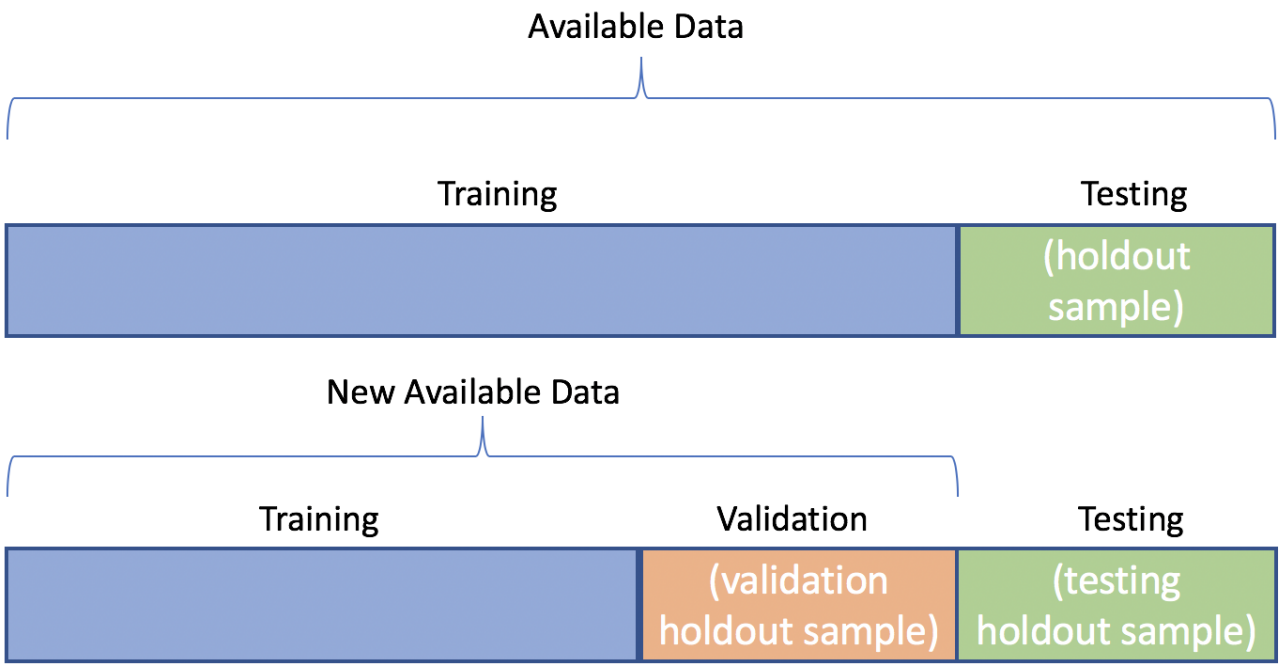
Training-Validation-Testing



Then, how to estimate the performance of a selected model after deployment?

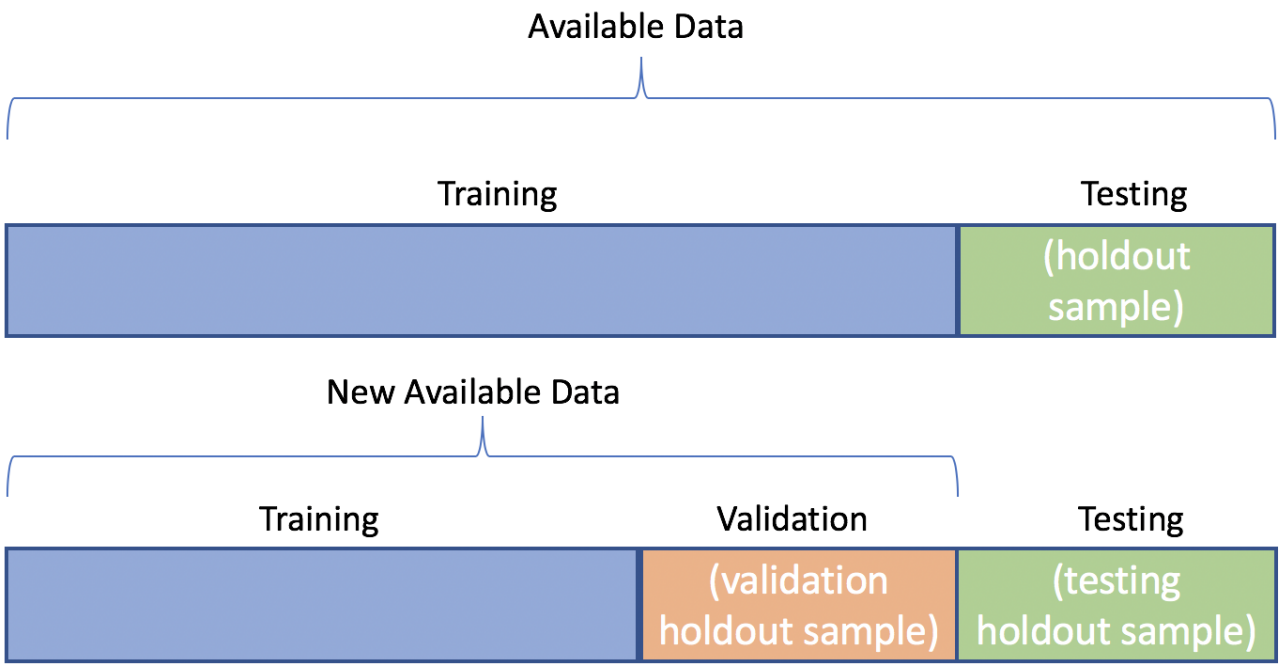
- Let's preserve two "test" datasets.
- One is used for model selection. Another one is used for "real" testing.

Training-Validation-Testing



- Training Data: For model training.
- Validation Data: For model selection, and hyper-parameter tuning.
- Test Data: For estimating model performance.

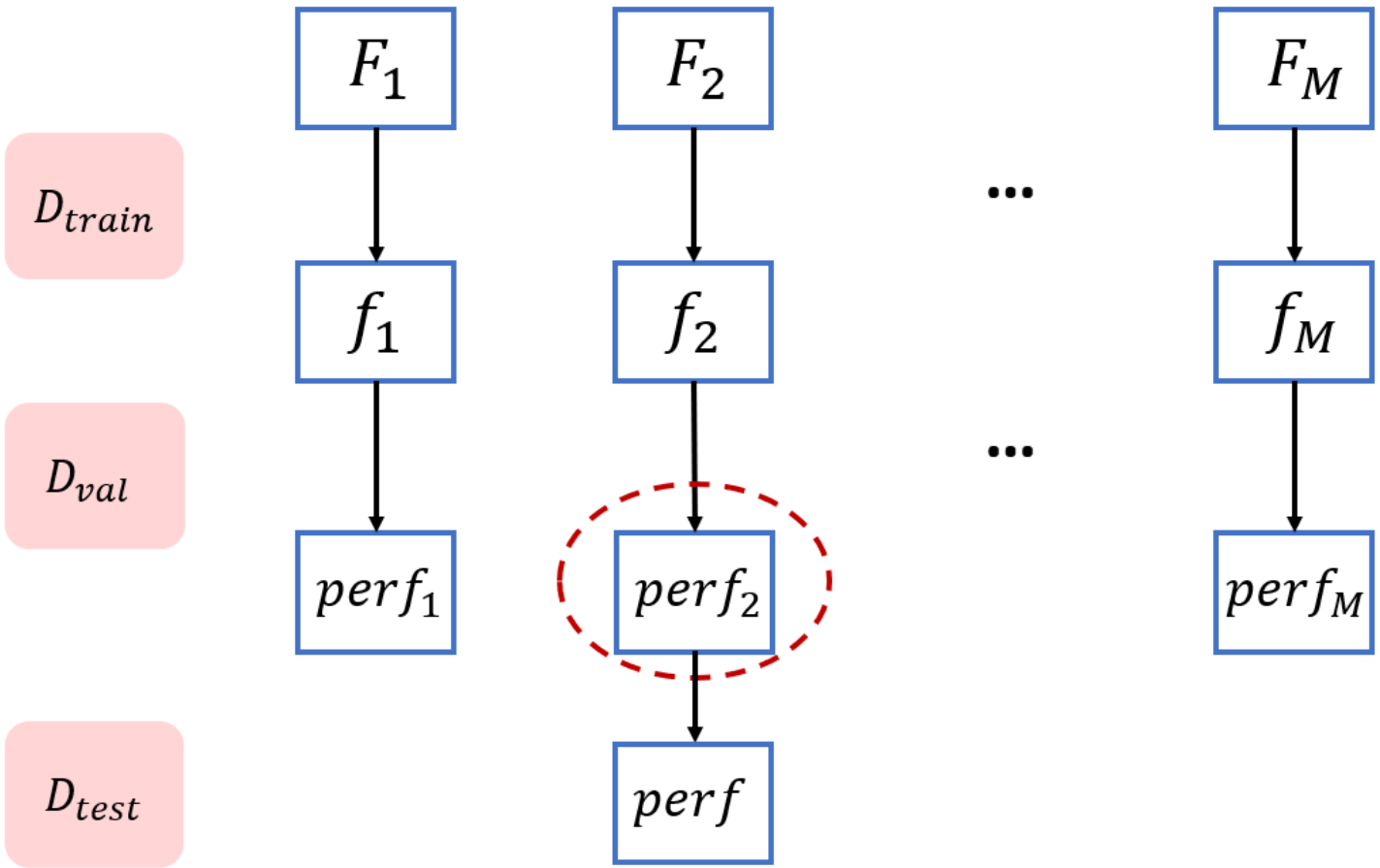
Training-Validation-Testing



Another valuable point of view:

- There is no "training and validation".
- There is only "training".
- "validation" is part of training.

Training-Validation-Testing

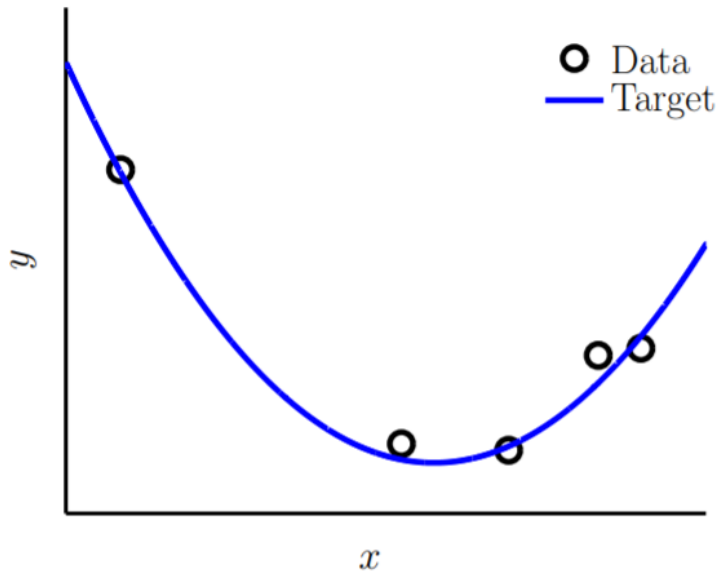


Till now ...

- How to select the best model for a given task? (✓)
- Are complex models always better than simple models?

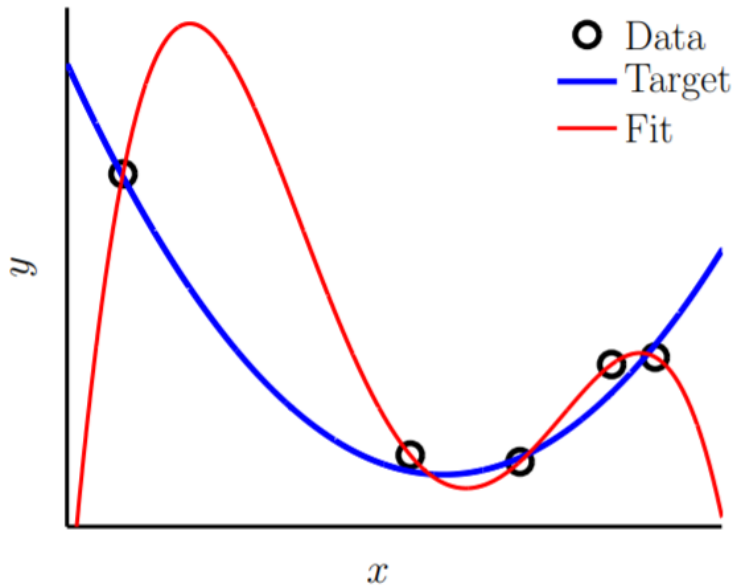
Contents adopted from "Learning from data" by Yaser Abu-Mostafa.

Overfitting: An Example



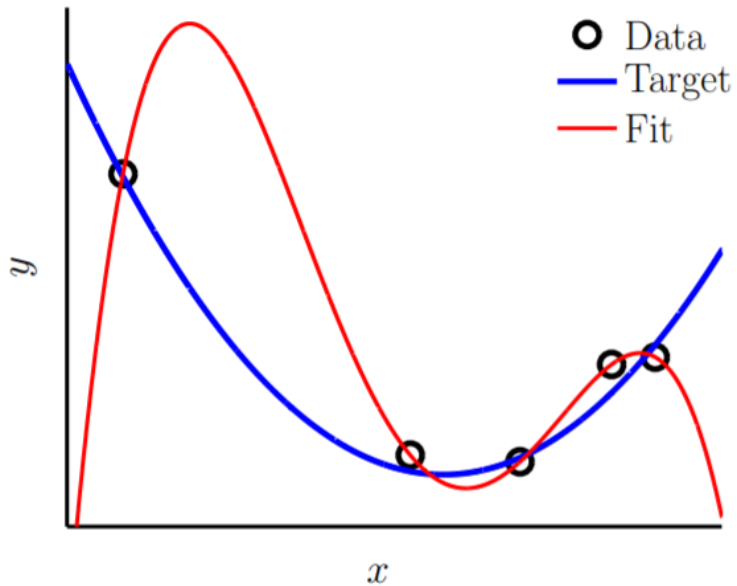
- Quadratic f
- Sample 5 data points on f.
- Add a little noise (measurement error).

Overfitting: An Example



- Target: Quadratic f
- Sample 5 data points on f.
- Add a little noise (measurement error).
- **4th order polynomial fit**.

Overfitting

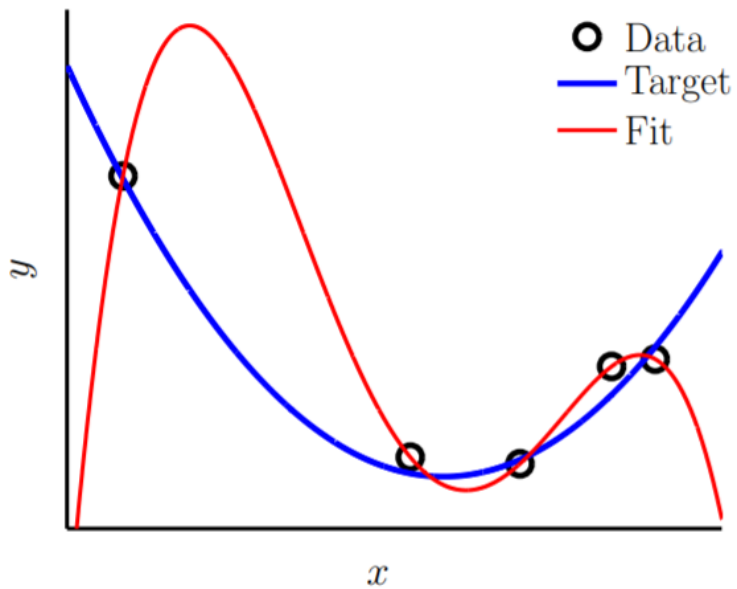


- Classic **overfitting**: simple target with excessively complex **hypothesis** (i.e., 4th order polynomial).

$$E_{in} \approx 0, E_{out} \gg 0$$

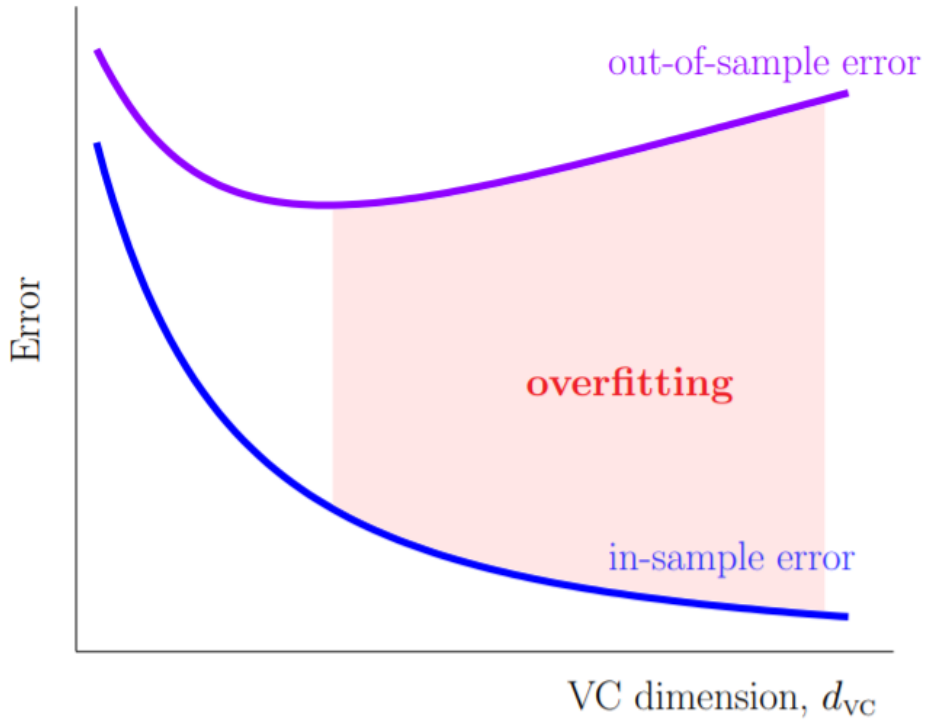
  - Hypothesis $\mathcal{H}$: The type of function that we guess should describes the target.
  - E_{in}: in-sample error (training); E_{out}: out-of-sample error (test).

Overfitting



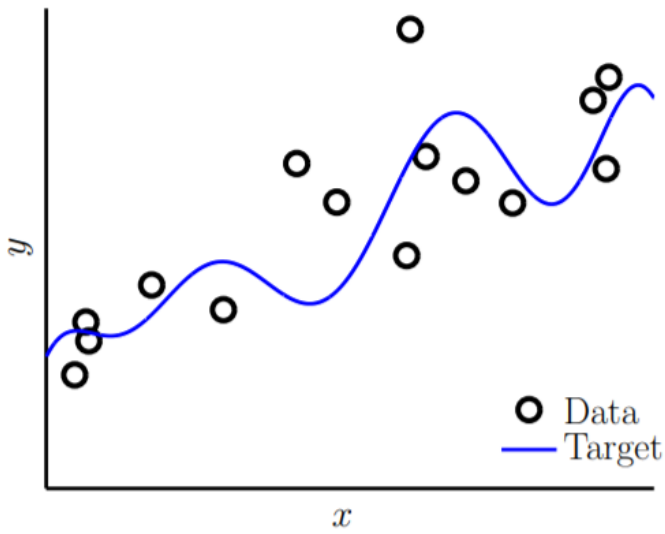
- The **noise** did us in.
- Fitting the data more than is warranted.

Overfitting

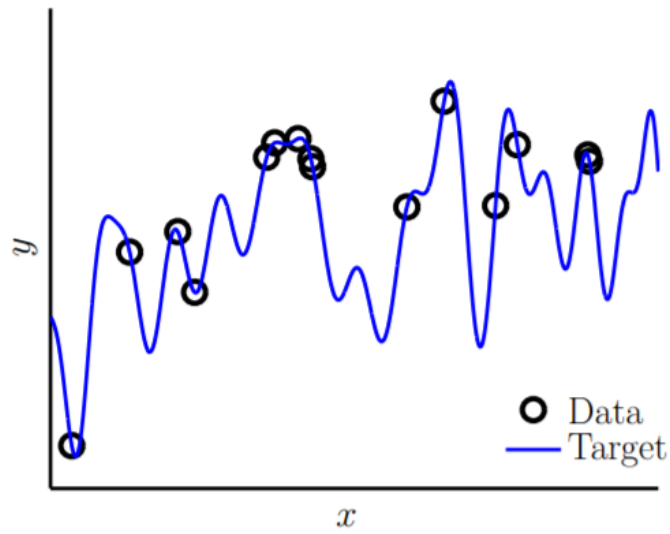


Overfitting:
Going for lower and lower E_{in} results in higher and higher E_{out}.

Overfitting: Another Example

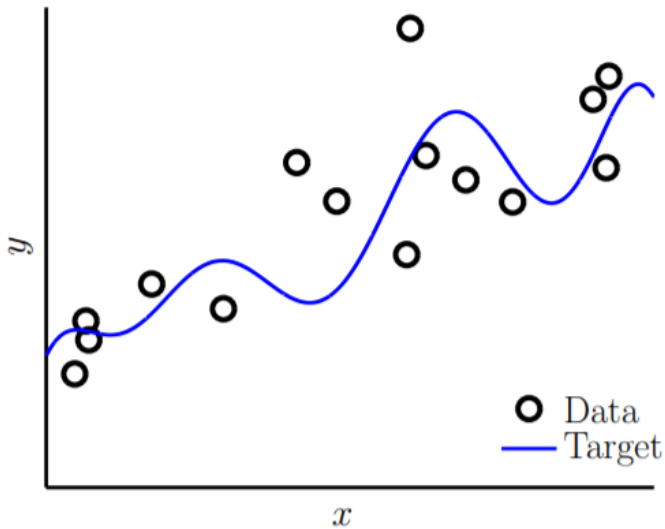


10th order f with noise.

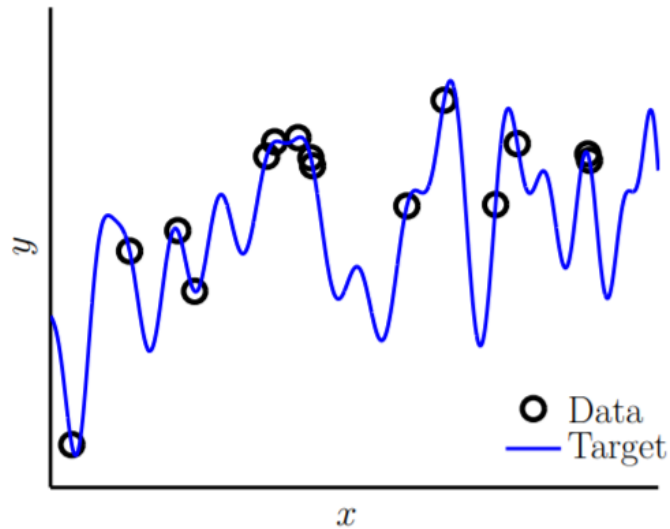


50th order f with no noise.

Overfitting: Another Example



10th order f with noise.



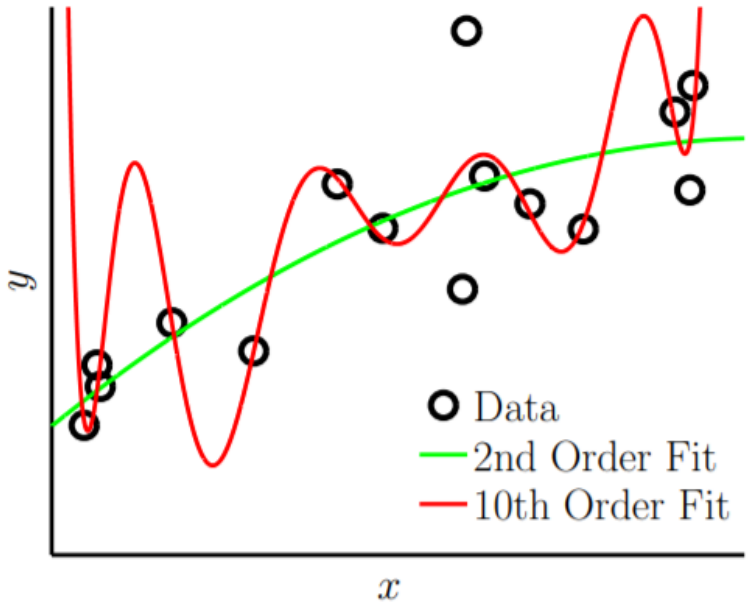
50th order f with no noise.

Two hypothesis:

- $\mathcal{H}_2$: 2-nd order polynomial fit.
- $\mathcal{H}_{10}$: 10-th order polynomial fit

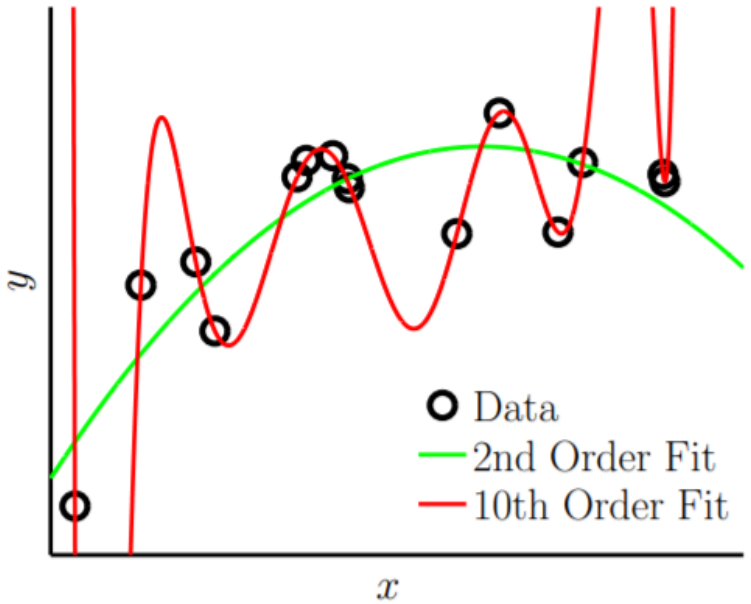
Which model do you pick for which problem and why?

Overfitting: Another Example



simple noisy target

	2nd Order	10th Order
E_{in}	0.050	0.034
E_{out}	0.127	**9.00**

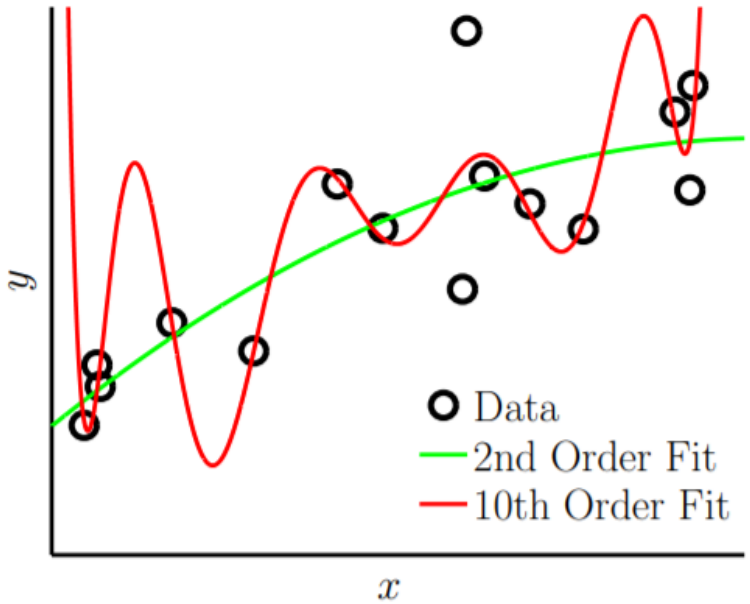


complex noiseless target

	2nd Order	10th Order
E_{in}	0.029	10^{-5}
E_{out}	0.120	**7680**

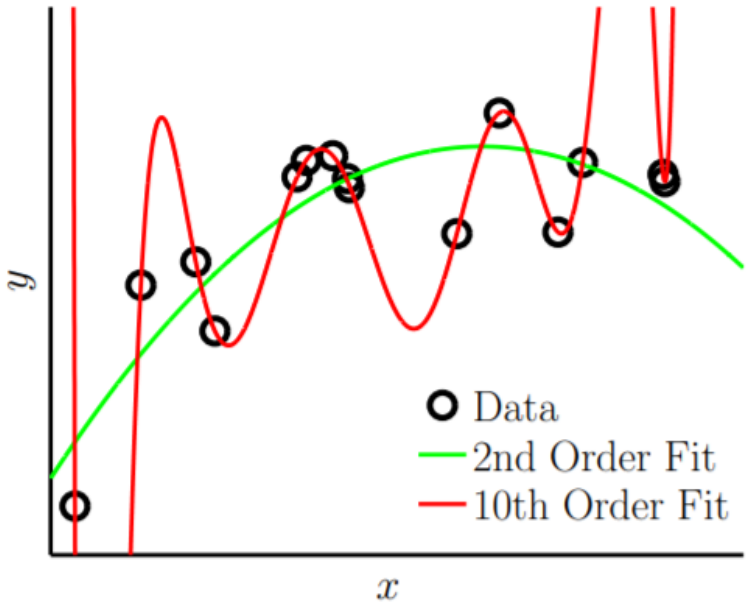
Simpler $\mathcal{H}$ is better even for the more complex target with no noise!

Overfitting



simple noisy target

	2nd Order	10th Order
E_{in}	0.050	0.034
E_{out}	0.127	**9.00**

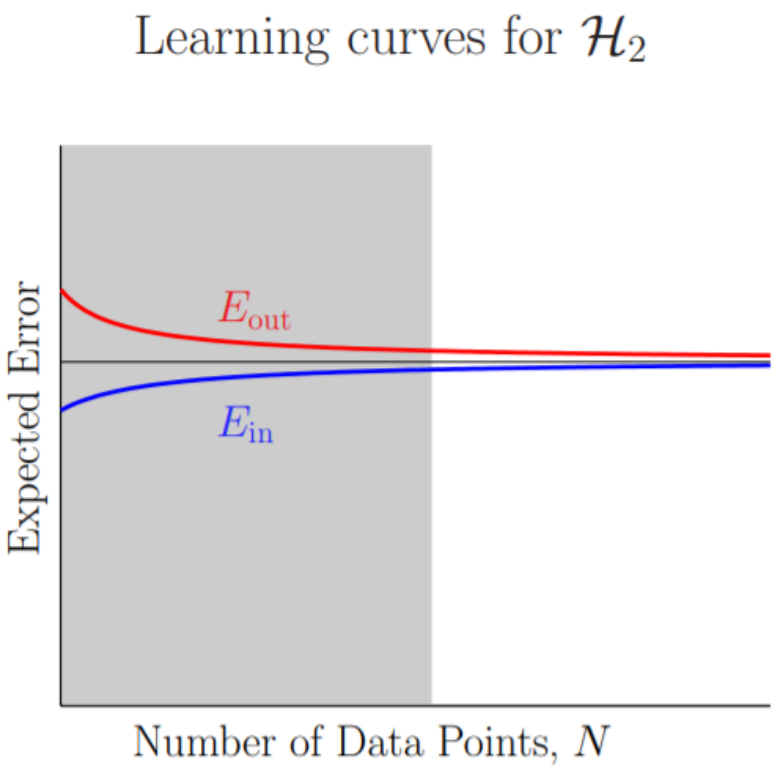


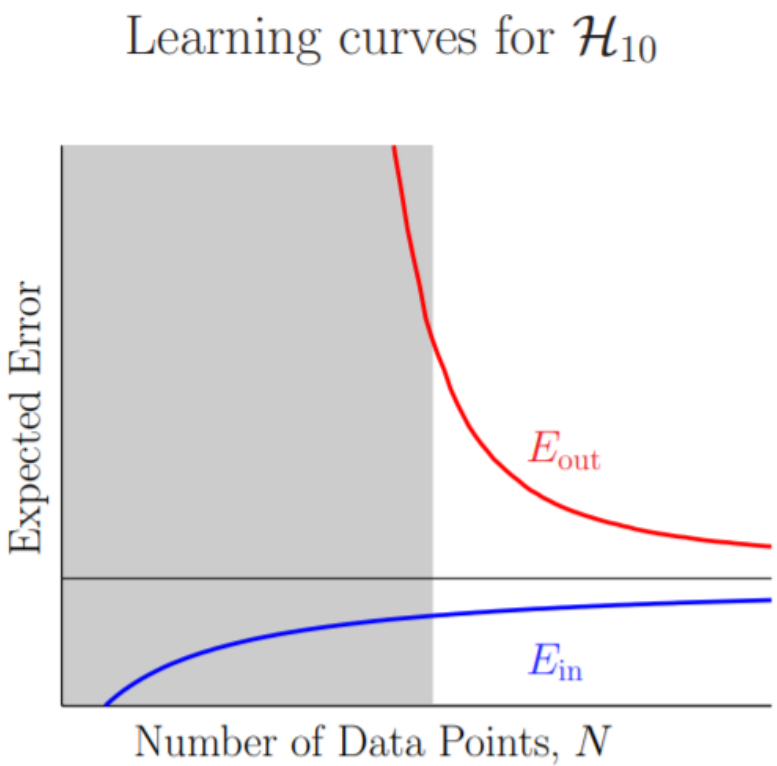
complex noiseless target

	2nd Order	10th Order
E_{in}	0.029	10^{-5}
E_{out}	0.120	**7680**

$\mathcal{H}$ should match **quantity** and **quality** of data, not just f.

Overfitting





When is $\mathcal{H}_{10}$ better than $\mathcal{H}_2$?

Overfitting

- **Conclusion**: We should choose models with reasonable complexity.
- **Wait...**: Deep models are super complex. Then, why do they still work well?