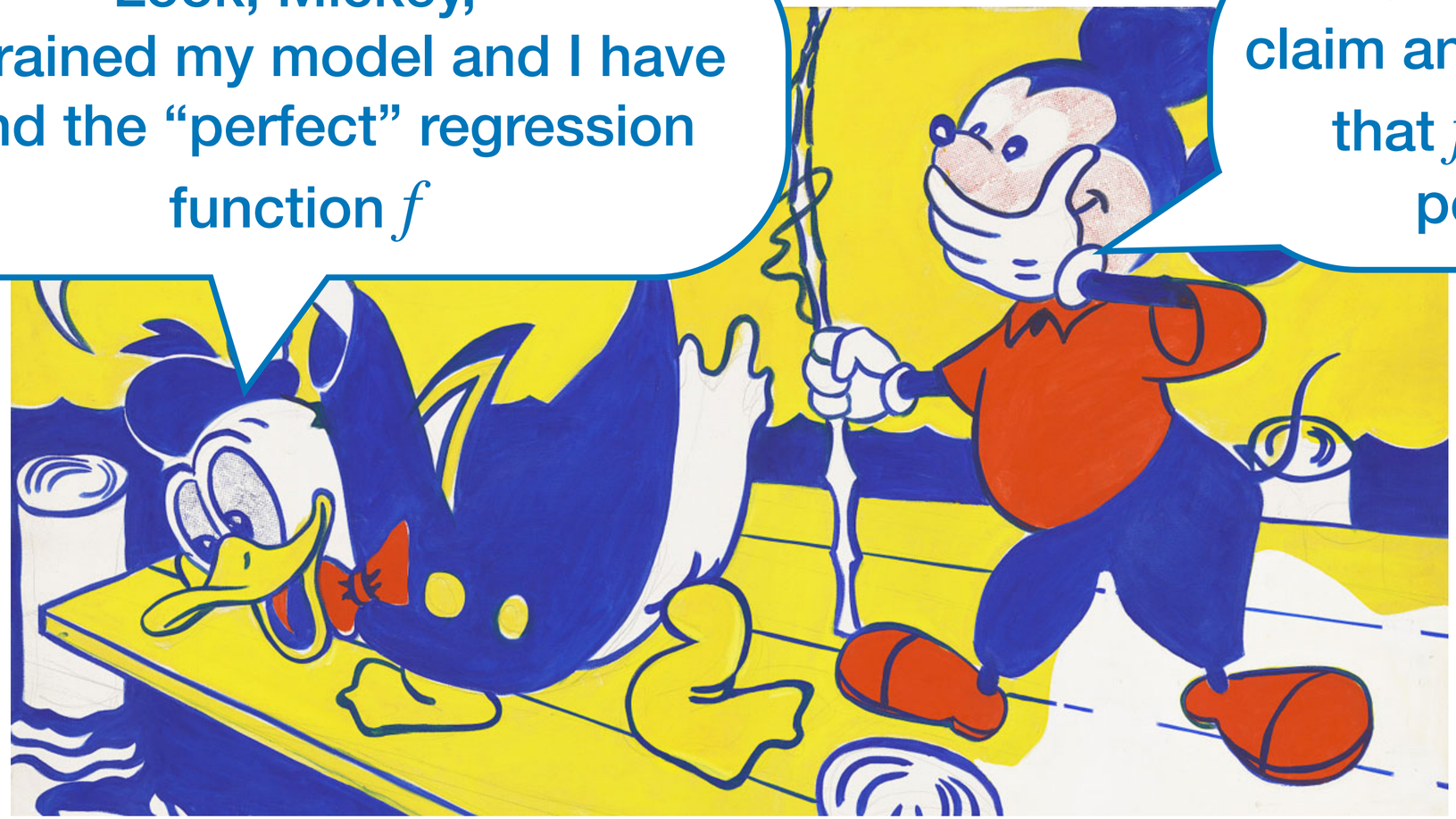
Generalization, Model Selection, and Validation

Machine Learning Course - CS-433
Oct 1, 2024
Martin Jaggi & Nicolas Flammarion

Generalization, validation?

Look, Mickey,
I've trained my model and I have found the "perfect" regression function f

How can I verify this claim and have confidence that f will have good performance?



What is the model selection problem?

- Ridge regression: $w_\lambda = \arg\min_w \frac{1}{2N}\sum_{n=1}^{N}(y_n - x_n^\top w)^2 + \lambda\|w\|_2^2$

  Hyperparameter

  - λ can be tuned to control the model complexity (to reduce overfitting)
  - In practice: $(\lambda_1, \cdots, \lambda_k) \longrightarrow$ Algorithm $\longrightarrow (w_1, \cdots, w_k)$
  - Which λ should we use?

- Polynomial feature expansion: $(x_{(1)}, x_{(2)}) \xrightarrow{\phi} (x_{(1)}, x_{(2)}, x_{(1)}^2 + x_{(2)}^2)$

  $(x_{(1)}, x_{(2)}, 5x_{(1)}^2 + 2x_{(2)}^2, x_{(2)}^3 + 2x_{(1)})$

  - Enrich the model complexity, by augmenting the feature vector x.
  - Here the degree d is the hyperparameter

We are facing the same problem: **how do we choose these hyperparameters?**

Model selection for neural networks

Algorithms?

SGD
Adam
Which step-size?
Which batch-size?
Which momentum?

Architectures?

FullyConnected
ConvNet
ResNet
Transformer
Which width?
Which depth?
Batch normalization?

Regularizations?

Weight decay?
Early stopping?
Data augmentations?

Probabilistic Setup

Data Model:

Unknown distribution $\mathscr{D}$ with range $\mathscr{X} \times \mathscr{Y}$
We see a dataset S of independent samples from $\mathscr{D}$:

$$S = \{(x_n, y_n)\}_{n=1}^{N} \sim \mathscr{D} \quad \text{i.i.d.}$$

Learning Algorithm:

$$\mathscr{A}(S) = f_S$$

Input Output

Ridge regression: gradient descent or least-squares estimator

Can add a subscript $f_{S,\lambda}$ to indicate the hyper parameter dependency

Generalization Error: how accurate is f at predicting?

We compute the ***expected error*** over all samples drawn from distribution $\mathscr{D}$:

$$L_{\mathscr{D}}(f) = \mathbb{E}_{(x,y)\sim\mathscr{D}}[\ell(y, f(x))]$$

where $\ell(\,\cdot\,,\cdot\,)$ is the loss function

- Ex: $\ell(y, y') = \frac{1}{2}(y - y')^2$, logistic loss, hinge loss

The quantity $L_{\mathscr{D}}(f)$ has many names: **True / Expected / Generalization** **Risk / Error / Loss**

This is the quantity we are fundamentally interested in

Generalization Error: how accurate is f at predicting?

We compute the ***expected error*** over all samples drawn from distribution $\mathscr{D}$:

$$L_{\mathscr{D}}(f) = \mathbb{E}_{(x,y)\sim\mathscr{D}}[\ell(y, f(x))]$$

where $\ell(\,\cdot\,,\cdot\,)$ is the loss function

- Ex: $\ell(y, y') = \frac{1}{2}(y - y')^2$, logistic loss, hinge loss

The quantity $L_{\mathscr{D}}(f)$ has many names: **True / Expected / Generalization** — **Risk / Error / Loss**

This is the quantity we are fundamentally interested in

Problem: $\mathscr{D}$ is unknown

Empirical Error: what we can compute

We can approximate the true error by **averaging the loss function over the dataset**

$$L_S(f) = \frac{1}{|S|} \sum_{(x_n,y_n)\in S} \ell(y_n, f(x_n)) \,.$$

Also called: **empirical risk/error/loss**

⚠ The samples are random thus $L_S(f)$ is a random variable

It is an unbiased estimator of the true error

Law of large number: $L_S(f) \underset{|S|\to\infty}{\to} L_{\mathcal{D}}(f)$ but fluctuations!

Generalization gap: $|L_{\mathcal{D}}(f) - L_S(f)|$

Training error: what we are minimizing

⚠ the prediction function f_S is itself a function of the data S

When the model has been trained on the same data it is applied to, the empirical error is called the ***training error:***

$$L_S(f_S) = \frac{1}{|S|} \sum_{(x_n, y_n) \in S} \ell(y_n, f_S(x_n))$$

This is the objective function you are minimizing to find the predictor

It might not be representative of the error we see on “fresh” samples

The reason that $L_S(f_S)$ might not be close to $L_{\mathcal{D}}(f_S)$ is of course overfitting

Splitting the data

Problem: Validating model on the same data we trained it on
Fix: **Split the data** into an independent *training and test set:*

$$S = S_{\text{train}} \cup S_{\text{test}}$$

1. We **learn** the function $f_{S_{\text{train}}}$ using the **train set**
2. We **validate** it computing the error on the **test set**

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n, x_n) \in S_{\text{test}}} \ell(y_n, f_{S_{\text{train}}}(x_n))$$

➡ Since S_{test} and S_{train} are independent: $L_{S_{\text{test}}}(f_{S_{\text{train}}}) \approx L_{\mathcal{D}}(f_{S_{\text{train}}})$

Splitting the data

Problem: Validating model on the same data we trained it on
Fix: **Split the data** into an independent *training and test set:*

$$S = S_{\text{train}} \cup S_{\text{test}}$$

1. We **learn** the function $f_{S_{\text{train}}}$ using the **train set**
2. We **validate** it computing the error on the **test set**

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n, x_n) \in S_{\text{test}}} \ell(y_n, f_{S_{\text{train}}}(x_n))$$

➡ Since S_{test} and S_{train} are independent: $L_{S_{\text{test}}}(f_{S_{\text{train}}}) \approx L_{\mathcal{D}}(f_{S_{\text{train}}})$

⚠ We have less data both for the learning and the validation tasks (tradeoff)

Generalization gap:
How far is the test from the true error?

Claim: given a model f and a test set $S_{\text{test}} \sim \mathscr{D}$ i.i.d. (not used to learn f) and a loss $\ell(\cdot, \cdot) \in [a, b]$:

$$\mathbb{P}\left[\left| \underbrace{L_{\mathscr{D}}(f) - L_{S_{\text{test}}}(f)}_{\text{Generalization Gap}} \right| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{\text{test}}|}} \right] \leq \delta$$

The error decreases as $\mathcal{O}(1/\sqrt{|S_{\text{test}}|})$ with the number of test points

High probability bound: δ is only in the $\ln$

➡The more data points we have, the more confident we are that the empirical loss we measure is close to the true loss

Why do you care?

- Given a predictor f and a dataset S you can control the expected risk:

$$\mathbb{P}\Bigg(\underbrace{L_{\mathscr{D}}(f)}_{\text{not computable}} \geq \underbrace{L_S(f)}_{\text{Computable}} + \underbrace{\sqrt{\frac{(a-b)^2 \ln(2/\delta)}{2\,|S_{test}|}}}_{\text{deviation}} \Bigg) \leq \delta$$

- Given a dataset S
  1. Split: $S = S_{\text{train}} \cup S_{\text{test}}$
  2. Train: $\mathscr{A}(S_{\text{train}}) = f_{S_{\text{train}}}$
  3. Validate:

$$\mathbb{P}\left(L_{\mathscr{D}}(f_{S_{\text{train}}}) \geq L_{S_{\text{test}}}(f_{S_{\text{train}}}) + \sqrt{\frac{(a-b)^2 \ln(2/\delta)}{2\,|S_{test}|}} \right) \leq \delta$$

➡We can obtain a probabilistic upper bound on the expected risk

The proof relies only on concentration inequalities

Since $(x_n, y_n) \in S_{\text{test}}$ are chosen independently, the associated losses $\Theta_n = \ell(y_n, f(x_n)) \in [a, b]$ given a fixed model f, are also i.i.d. random variables

Empirical loss: $\frac{1}{N}\sum_{n=1}^{N} \Theta_n = \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(x_n)) = L_{S_{\text{test}}}(f)$

True loss: $\mathbb{E}[\Theta_n] = \mathbb{E}[\ell(y_n, f(x_n))] = L_{\mathcal{D}}(f)$

What is the chance that the empirical loss $L_{S_{\text{test}}}(f)$ deviates from the true loss by more than a given constant?

➡ classically addressed using **concentration inequalities**

Hoeffding inequality: a simple concentration bound

Claim: Let $\Theta_1, \ldots, \Theta_N$ be a sequence of i.i.d. random variables with mean $\mathbb{E}[\Theta]$ and range $[a, b]$

$$\mathbb{P}\left[\left|\frac{1}{N}\sum_{n=1}^{N}\Theta_n - \mathbb{E}[\Theta]\right| \geq \varepsilon\right] \leq 2e^{-2N\varepsilon^2/(b-a)^2} \text{ for any } \varepsilon \geq 0$$

Concentration bound: the empirical mean is concentrated around its mean

A. Use it with $\Theta_n = \ell(y_n, f(x_n))$

B. Equating $\delta = 2e^{-2|S_{\text{test}}|\varepsilon^2/(b-a)^2}$ we get $\varepsilon = \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2|S_{\text{test}}|}}$ □

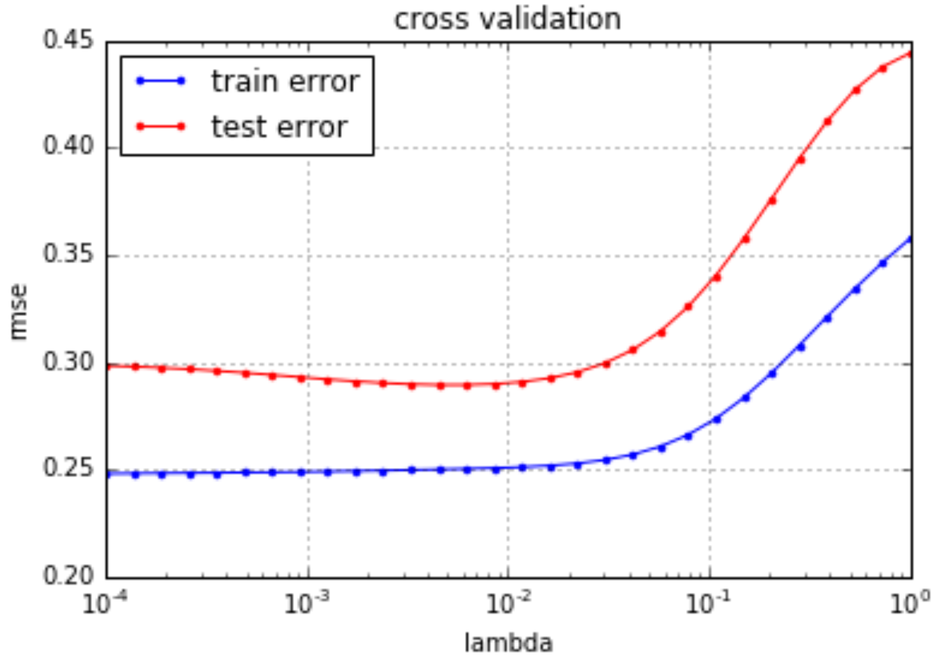
Model Selection: pick the best model

Goal: select the hyperparameters of our model (λ for ex. in ridge regression)
We have a set of candidate values $\{\lambda_k\}_{k=1}^{K}$. Which one should we choose?

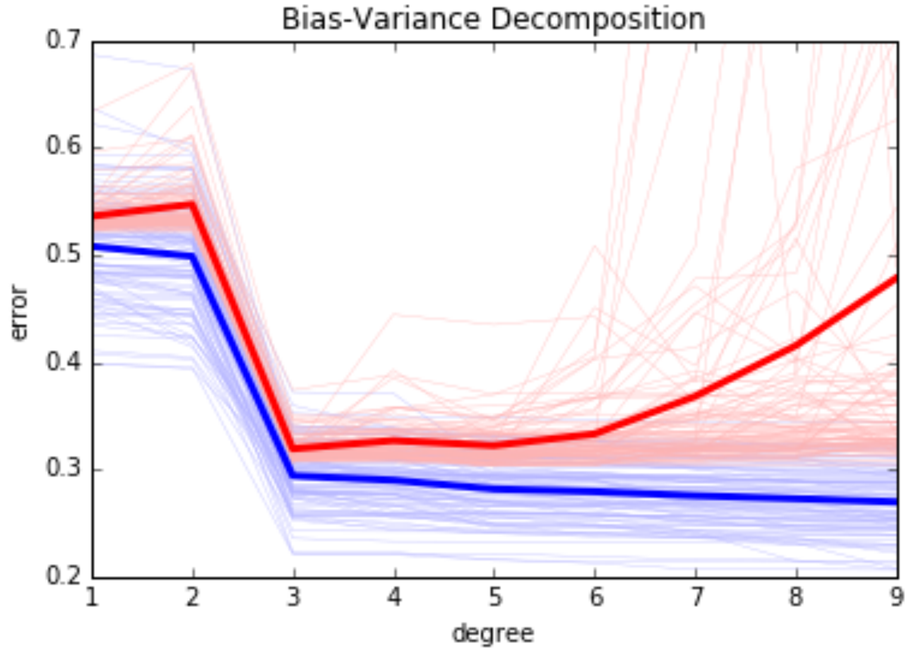
1. Split the data into $S = S_{\text{train}} \cup S_{\text{test}}$, generated independently from $\mathscr{D}$
2. Run the learning algorithm K times on the same training set S_{train} to compute the K prediction functions $f_{S_{\text{train}},\lambda_k}$
3. For each prediction function, compute the test error $L_{S_{\text{test}}}(f_{S_{\text{train}},\lambda_k})$

We then choose the value of the parameter λ giving the smallest test error

Examples



Ridge regression



Degree in case of a polynomial feature expansion

Does model selection work?

Two questions:

- How do we know that the best function $f_{S_{\text{train}},\lambda}$ is a good approximation of the best model within our function class?

- How do we know that $L_{S_{\text{test}}}(f_{S_{\text{train}},\lambda_k}) \approx L_{\mathcal{D}}(f_{S_{\text{train}},\lambda_k})$?

  We have discussed it for a single model
  What about several models?
  I.e., what is the justification that the min is actually good?

How far is each of the K test errors $L_{S_{\text{test}}}(f_k)$ from the true $L_{\mathcal{D}}(f_k)$?

Claim: we can bound the maximum deviation for all K candidates, by

$$\mathbb{P}\left[\max_k \left| L_{\mathcal{D}}(f_k) - L_{S_{\text{test}}}(f_k) \right| \geq \sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2\,|S_{\text{test}}|}} \right] \leq \delta$$

- The error decreases as $\mathcal{O}(1/\sqrt{|S_{\text{test}}|})$ with the number test points
- When testing K hyper-parameters, the error only goes up by $\sqrt{\ln(K)}$

➡So we can test many different models without incurring a large penalty

- It can be extended to infinitely many models

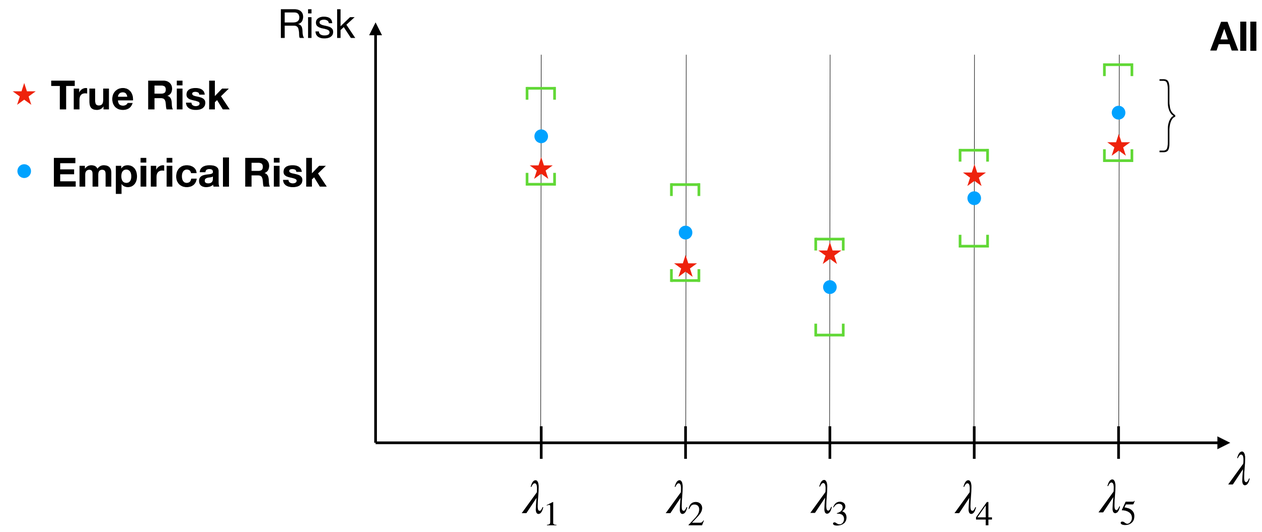
Proof: A simple union bound

The proof of this statement follows the proof of the special case $K = 1$

$$\begin{aligned}
\mathbb{P}\left[\ \max_k \left| L_{\mathscr{D}}(f_k) - L_{S_{\text{test}}}(f_k) \right| \geq \varepsilon \right] &= \mathbb{P}\left[\ \cup_k \left\{ \left| L_{\mathscr{D}}(f_k) - L_{S_{\text{test}}}(f_k) \right| \geq \varepsilon \right\} \right] \\
&\leq \sum_k \mathbb{P}\left[\left| L_{\mathscr{D}}(f_k) - L_{S_{\text{test}}}(f_k) \right| \geq \varepsilon \right] \\
&\leq 2Ke^{-2N\varepsilon^2/(b-a)^2}
\end{aligned}$$

Hence, equating $\delta = 2Ke^{-2N\varepsilon^2/(b-a)^2}$, we get $\varepsilon = \sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2N}}$ as stated

If we choose the “best” function according to the empirical risk then its true risk is not too far away from the true risk of the optimal choice



All the widths are the same

$$2\sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2\,|S_{\text{test}}|}}$$

Let $k^* = \text{argmin}_k L_{\mathcal{D}}(f_k)$ and $\hat{k} = \text{argmin}_k L_{S_{\text{test}}}(f_k)$ then

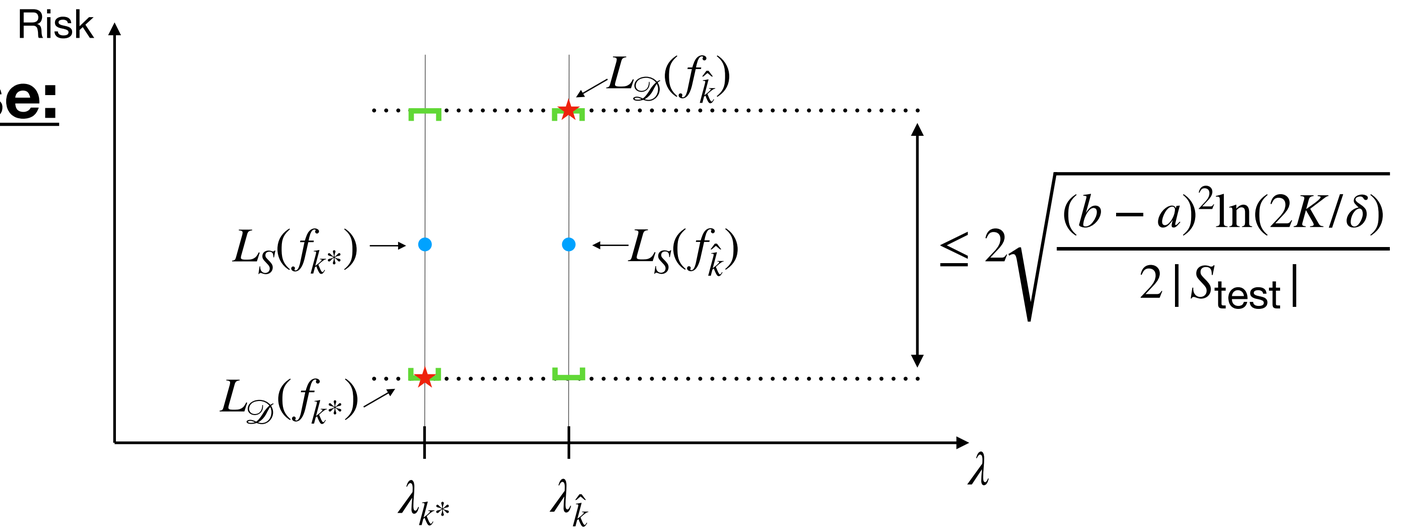
$$\mathbb{P}\left[\; L_{\mathcal{D}}(f_{\hat{k}}) \geq L_{\mathcal{D}}(f_{k^*}) + 2\sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2\,|S_{\text{test}}|}} \;\right] \leq \delta$$

Function with the smallest empirical risk

Function with the smallest true risk

If we choose the “best” function according to the empirical risk then its true risk is not too far away from the true risk of the optimal choice

Worst case:



Let $k^* = \text{argmin}_k L_{\mathcal{D}}(f_k)$ and $\hat{k} = \text{argmin}_k L_{S_{\text{test}}}(f_k)$ then

$$\mathbb{P}\left[L_{\mathcal{D}}(f_{\hat{k}}) \geq L_{\mathcal{D}}(f_{k*}) + 2\sqrt{\frac{(b-a)^2\ln(2K/\delta)}{2|S_{\text{test}}|}} \right] \leq \delta$$

Function with the smallest empirical risk

Function with the smallest true risk

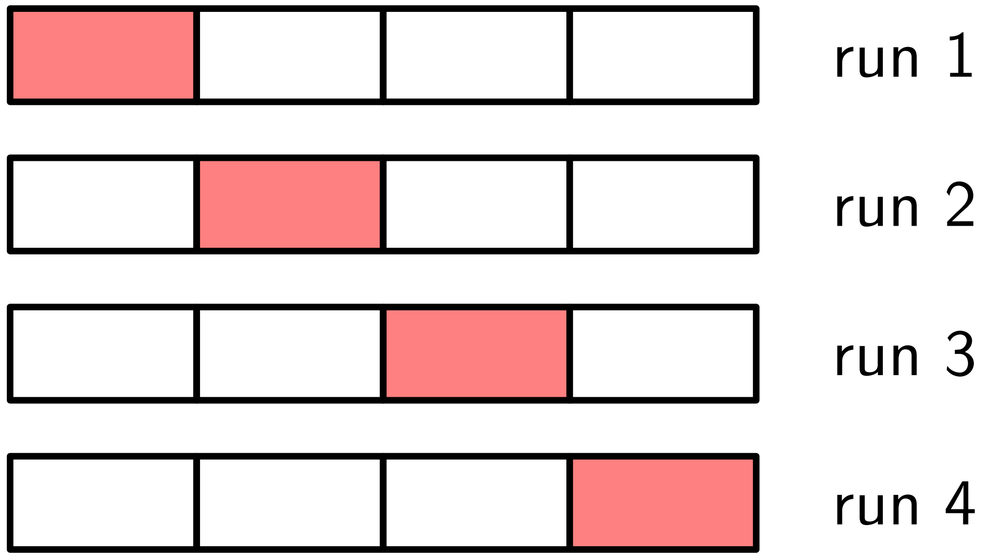
Cross-Validation

- Splitting the data once into two parts (one for training and one for testing) is not the most efficient way to use the data

- Cross-validation is a better way

K-fold Cross-Validation

1. Randomly partition the data into K groups
2. Train K times. Each time leave out exactly one of the K groups for testing and use the remaining $K-1$ groups for training.
3. Average the K results



- We have used all data for training, and all data for testing, and used each data point the same number of times
- Cross-validation returns an estimate of the *generalization-error* and its variance

Do we still have some time?

Hoeffding's inequality:

Let $\Theta_1, \ldots, \Theta_N$ be a sequence of i.i.d. random variables with mean $\mathbb{E}[\Theta]$ and range $[a, b]$. Then, for any $\varepsilon > 0$

$$\mathbb{P}\left[\left|\frac{1}{N}\sum_{n=1}^{N}\Theta_n - \mathbb{E}[\Theta]\right| \geq \varepsilon\right] \leq 2e^{-2N\varepsilon^2/(b-a)^2}$$

Proof (I)

- We equivalently assume that $\mathbb{E}[\Theta] = 0$ and that $\Theta_n \in [a, b]$
- We will only show that

$$\mathbb{P}\left\{ \frac{1}{N} \sum_{n=1}^{N} \Theta_n \geq \varepsilon \right\} \leq e^{-2N\varepsilon^2/(b-a)^2}$$

This, together with the equivalent bound

$$\mathbb{P}\left\{ \frac{1}{N} \sum_{n=1}^{N} \Theta_n \leq -\varepsilon \right\} \leq e^{-2N\varepsilon^2/(b-a)^2}$$

will prove the claim

Proof (II)

For any $s \geq 0$,

$$\mathbb{P}\left\{ \frac{1}{N} \sum_{n=1}^{N} \Theta_n \geq \varepsilon \right\} = \mathbb{P}\left\{ s\frac{1}{N} \sum_{n=1}^{N} \Theta_n \geq s\varepsilon \right\}$$

$$= \mathbb{P}\left\{ e^{s\frac{1}{N}\sum_{n=1}^{N}\Theta_n} \geq e^{s\varepsilon} \right\}$$

$$\leq \mathbb{E}[e^{s\frac{1}{N}\sum_{n=1}^{N}\Theta_n}]e^{-s\varepsilon} \qquad \text{(Markov inequality)}$$

$$= \prod_{n=1}^{N} \mathbb{E}[e^{\frac{s\Theta_n}{N}}]e^{-s\varepsilon} \qquad \text{(the r.v } \Theta_n \text{ are independent)}$$

$$= \mathbb{E}[e^{\frac{s\Theta}{N}}]^N e^{-s\varepsilon} \qquad \text{(the r.v } \Theta_n \text{ are i.d.)}$$

$$\leq e^{s^2(b-a)^2/(8N)}e^{-s\varepsilon} \qquad \text{(Hoeffding lemma)}$$

Proof (III)

What do we do now? We have for any $s \geq 0$

$$\mathbb{P}\left\{ \frac{1}{N} \sum_{n=1}^{N} \Theta_n \geq \varepsilon \right\} \leq e^{s^2(b-a)^2/(8N)} e^{-s\varepsilon}$$

In particular for the minimum value obtained for $s = \frac{4N\varepsilon}{(b-a)^2}$

$$\mathbb{P}\left\{ \frac{1}{N} \sum_{n=1}^{N} \Theta_n \geq \varepsilon \right\} \leq e^{-2N\varepsilon^2/(b-a)^2}$$

Hoeffding lemma

For any random variable X, with $\mathbb{E}[X] = 0$ and $X \in [a, b]$ we have

$$\mathbb{E}[e^{sX}] \leq e^{\frac{1}{8}s^2(b-a)^2} \text{ for any } s \geq 0$$

Proof outline:

Consider the convex function $s \mapsto e^{sx}$. In the range $[a, b]$ it is upper bounded by the chord

$$e^{sx} \leq \frac{x-a}{b-a}e^{sb} + \frac{b-x}{b-a}e^{sa}$$

Taking the expectation and recalling that $\mathbb{E}[X] = 0$, we get

$$\mathbb{E}[e^{sX}] \leq \frac{b}{b-a}e^{sa} - \frac{a}{b-a}e^{sb} \leq e^{s^2(b-a)^2/8}$$

Bias-Variance Decomposition

Machine Learning Course - CS-433
Oct 2, 2024
Martin Jaggi & Nicolas Flammarion

Last time

How can we judge if a given predictor is good?
How to select the best models of a family?

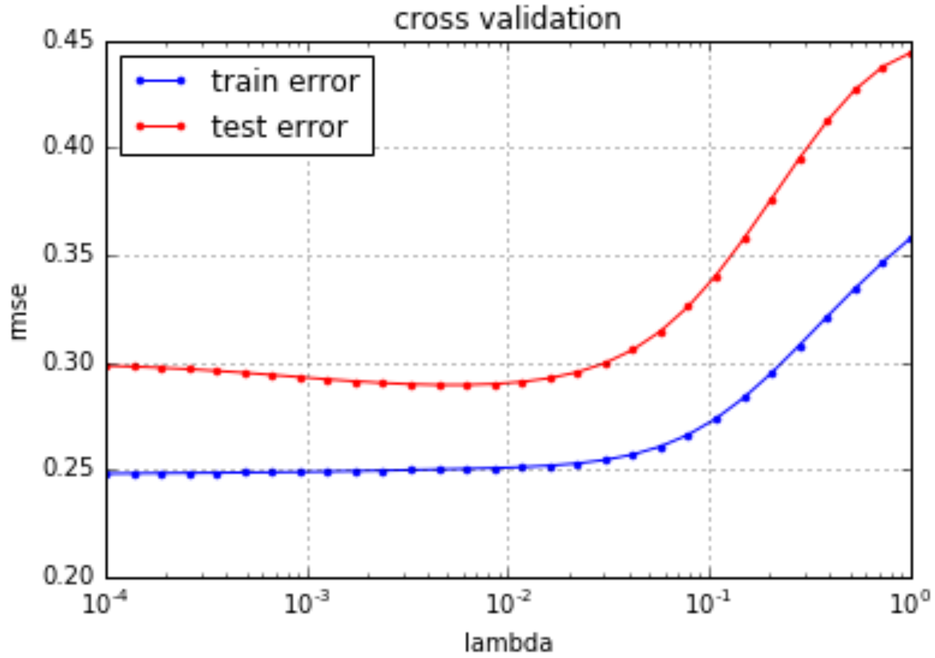
➡Bound the difference between the true and empirical risks

➡Split data into train and test sets (learn with the train and test on the test)

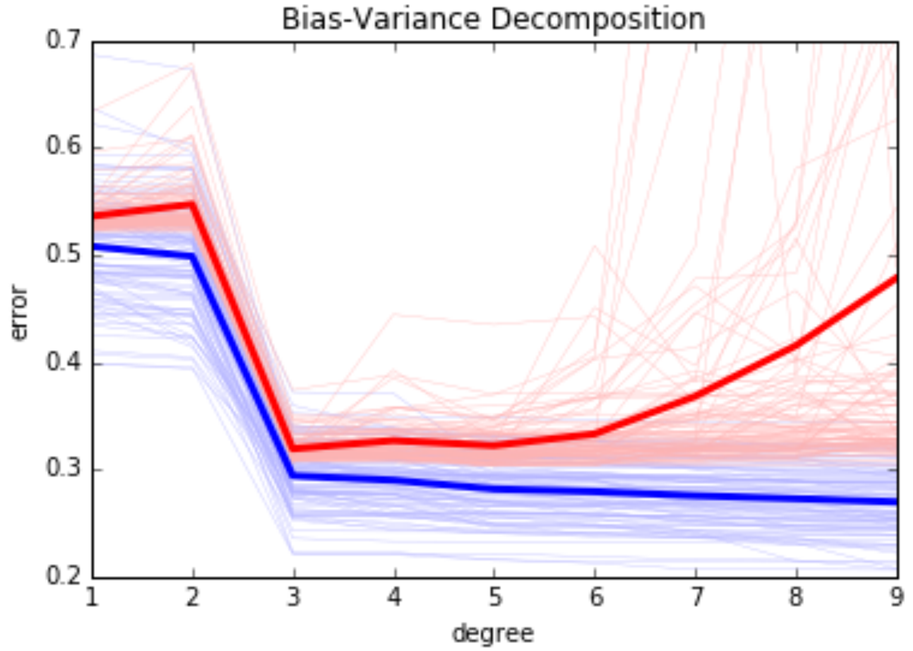
Motivation: Hyperparameters search (which often control the complexity)

But we haven’t investigated the role of the complexity of the class

Model selection curves

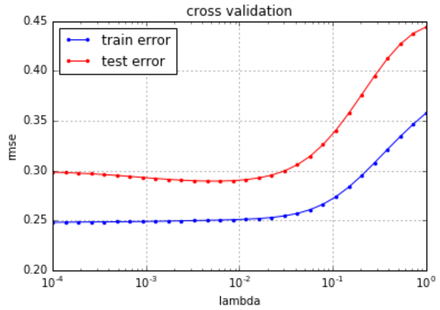


Ridge regression

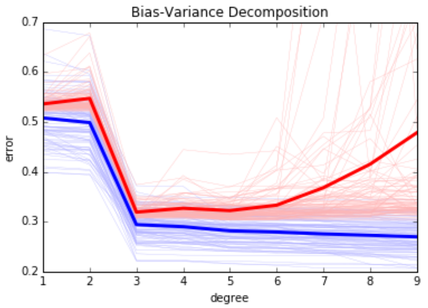


Degree in case of a polynomial feature expansion

Today



Ridge regression



Polynomial feature expansion

How does the risk behave as a function of the complexity of the model class?

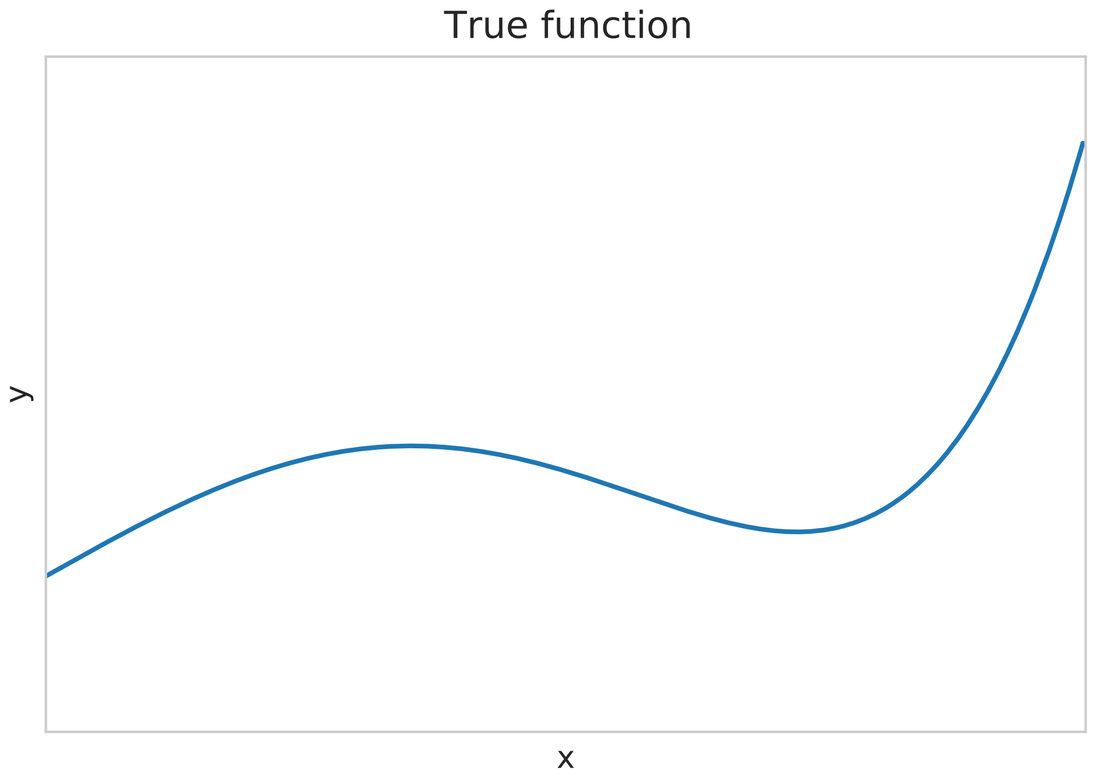
➡ ***Bias-Variance tradeoff***

It will help us to decide how complex and rich we should make our model

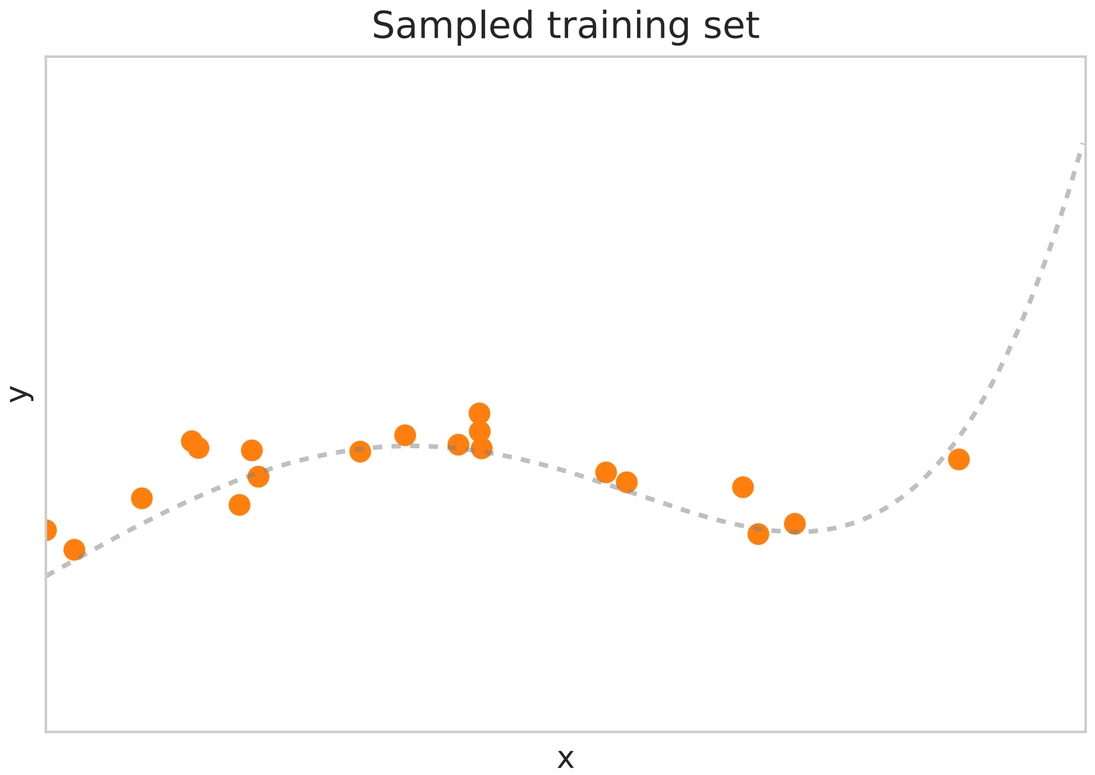
Before: quantitative

Now: ***qualitative***

A small experiment: 1D-regression



A small experiment: 1D-regression

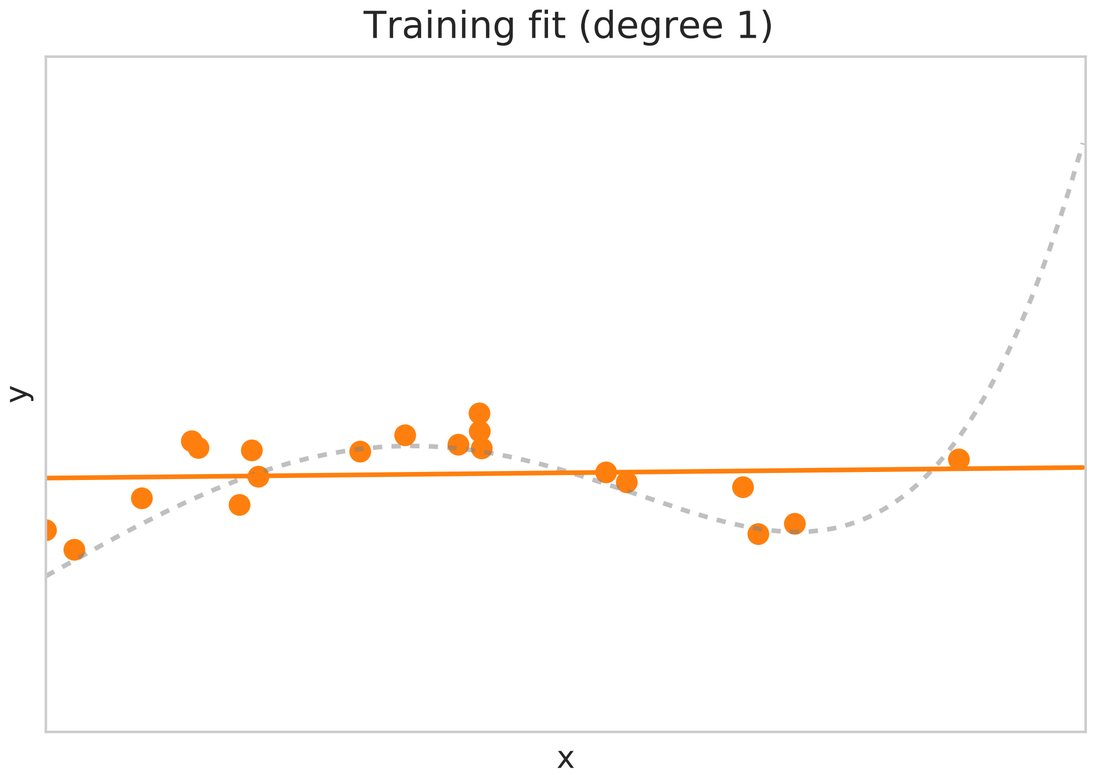


Linear regression using polynomial feature expansion $(x, x^2, x^3, \cdots, x^d)$

The maximum degree d measures the complexity of the class

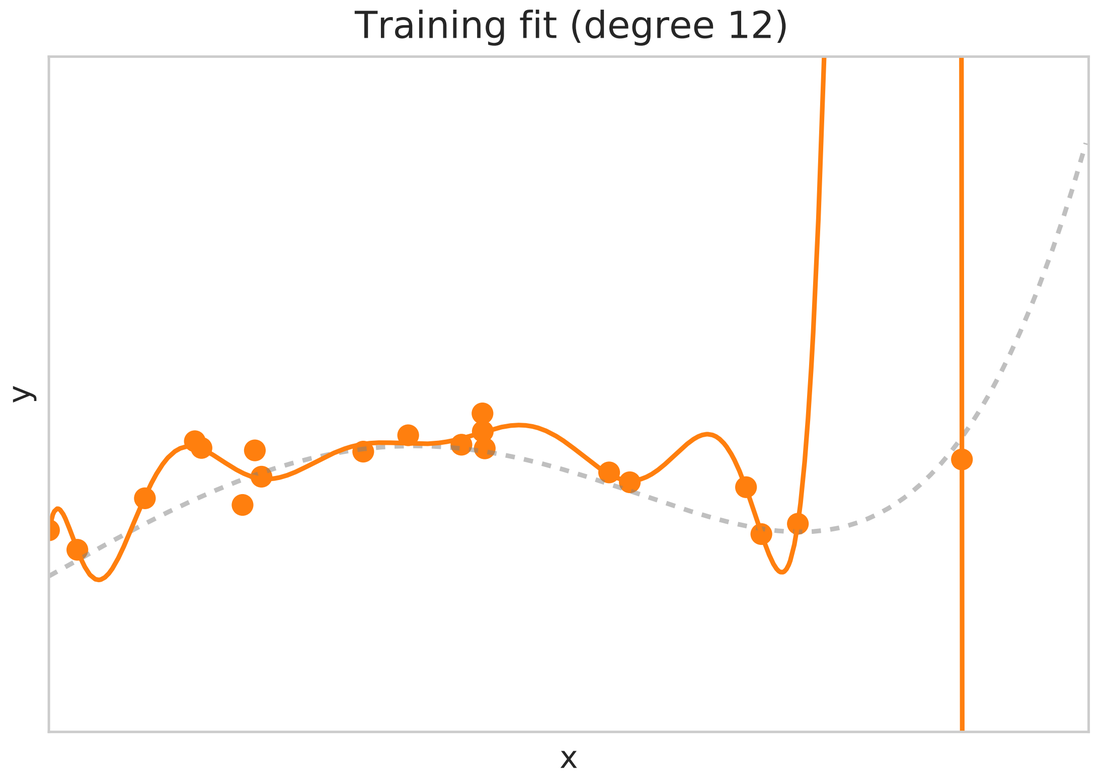
➡ How far should you go?

Simple model: bad fit



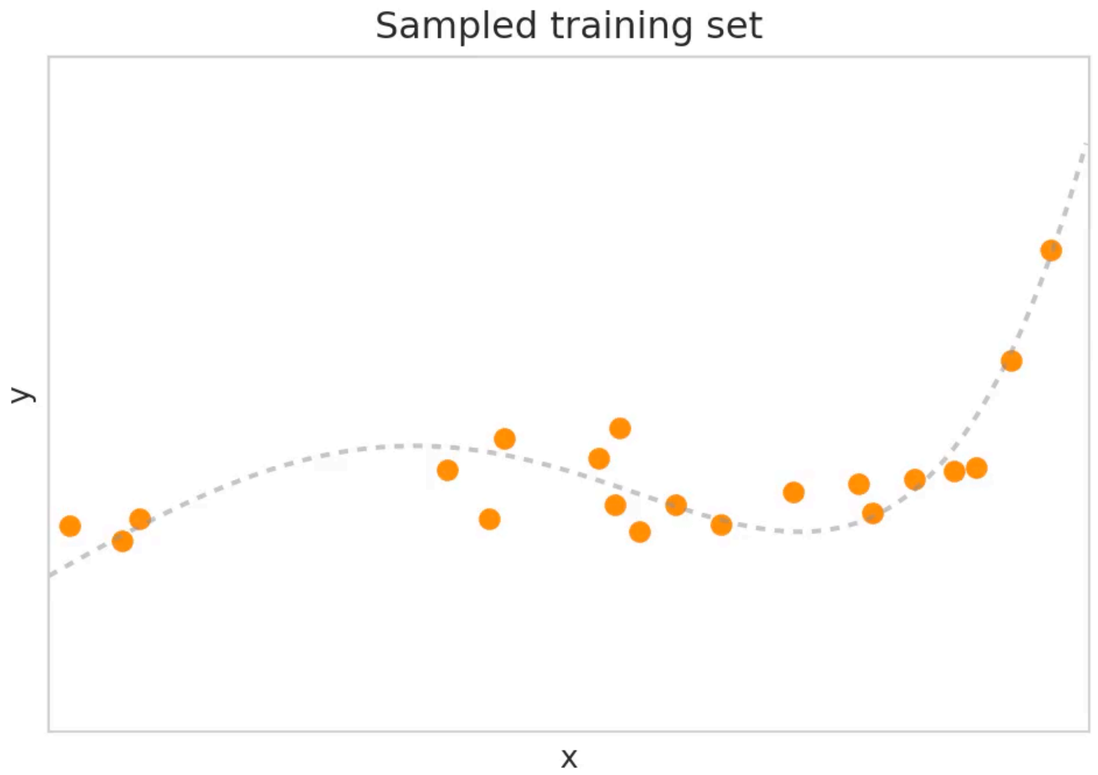
No linear function would be a good predictor. The model class is not rich enough

Complex model: good fit?



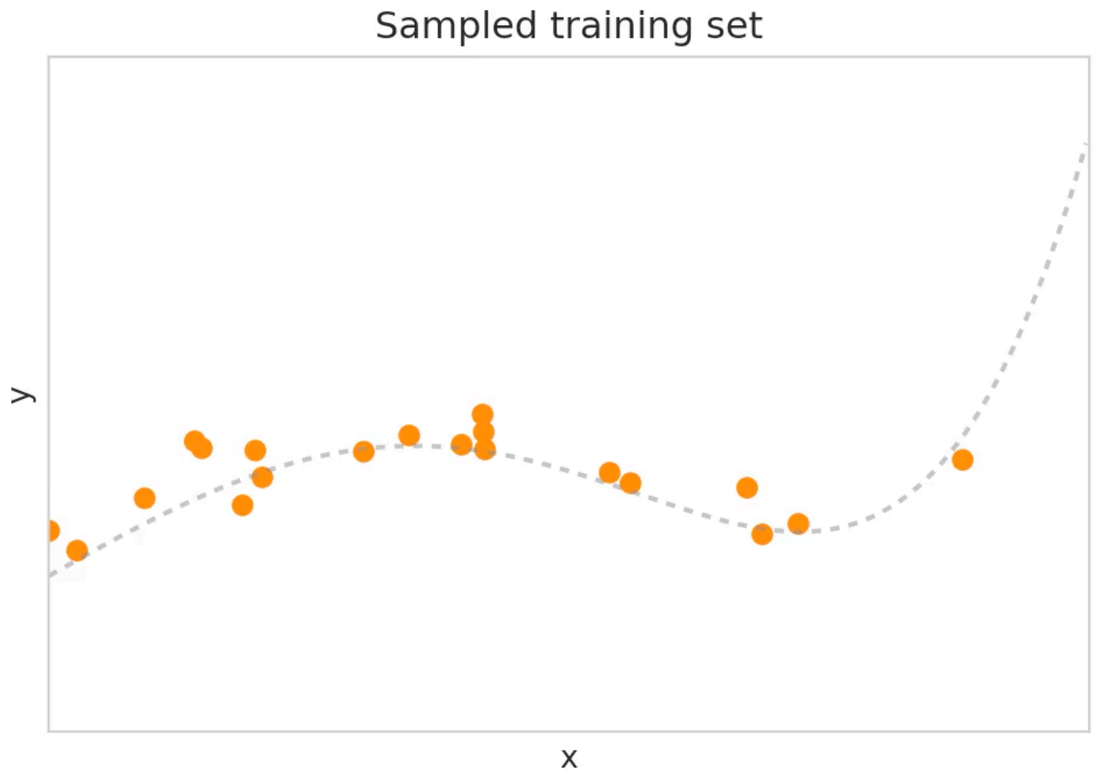
High degree polynomial will be a good fit. But?

But there is randomness in the data



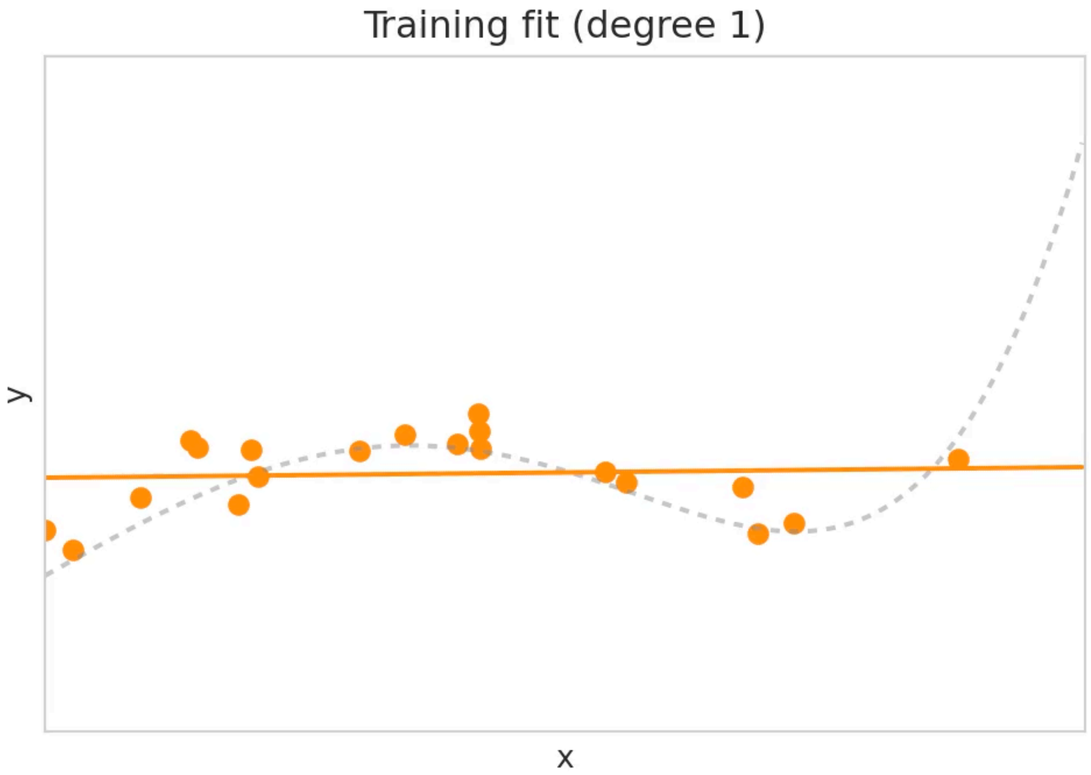
We have observed one particular S_{train} but we could have observed several others!

But there is randomness in the data



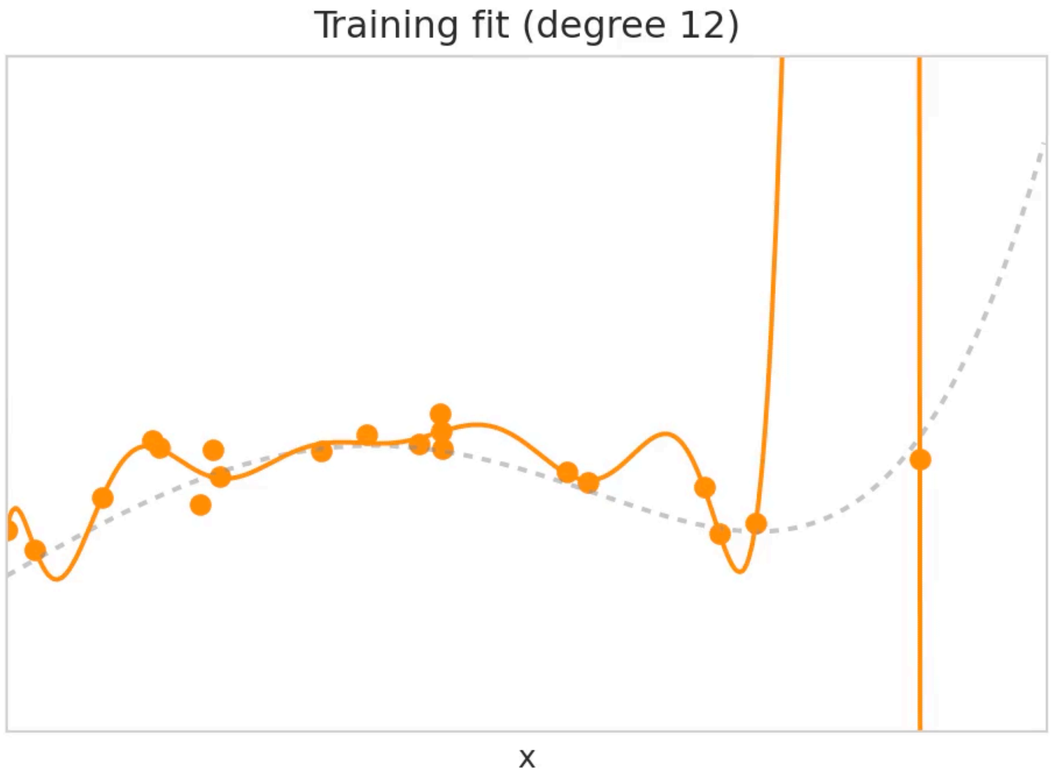
Even if we keep the same $(x_1, \cdots, x_n)$, we have variability in the observed $(y_1, \cdots, y_n)$

Thus there is randomness in the predictions



Moving a single observation will cause only a small shift in the position of the line

Underfitting

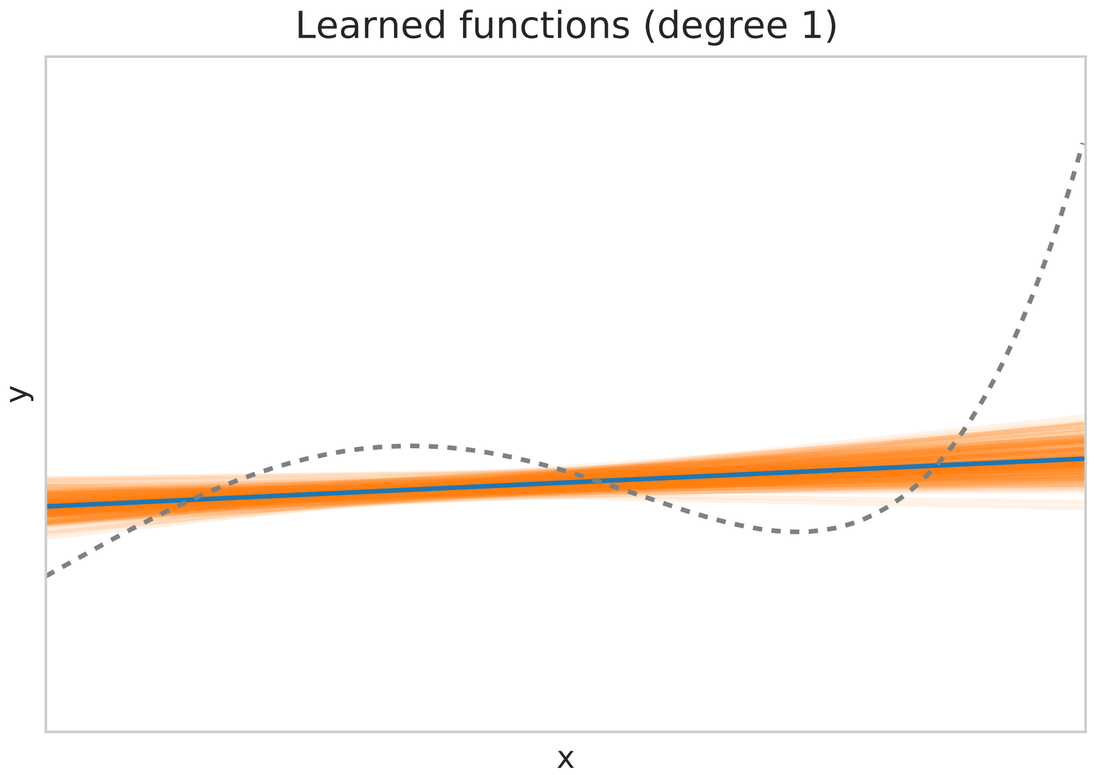


Changing one of the observations may change the prediction considerably

Overfitting

Simple models are less sensitive

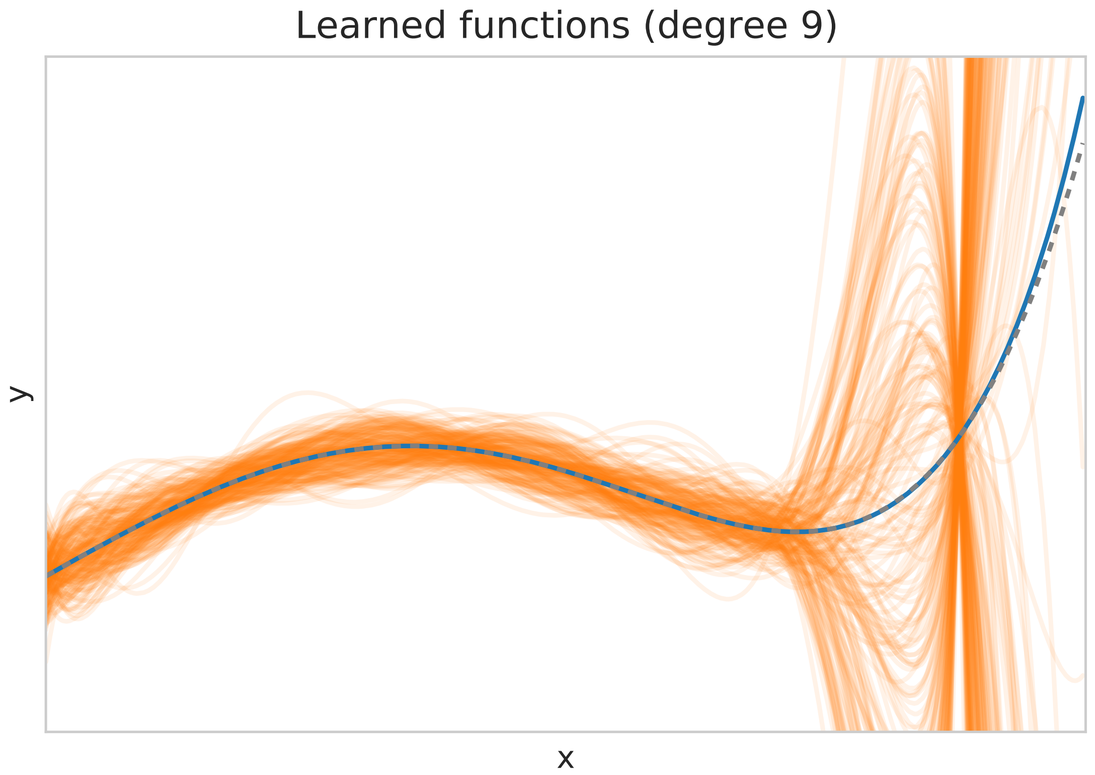
Simple models have large bias but low variance



The average of the predictions f_S does not fit well the data: **large bias**

The variance of the predictions f_S as a function of S is small: **small variance**

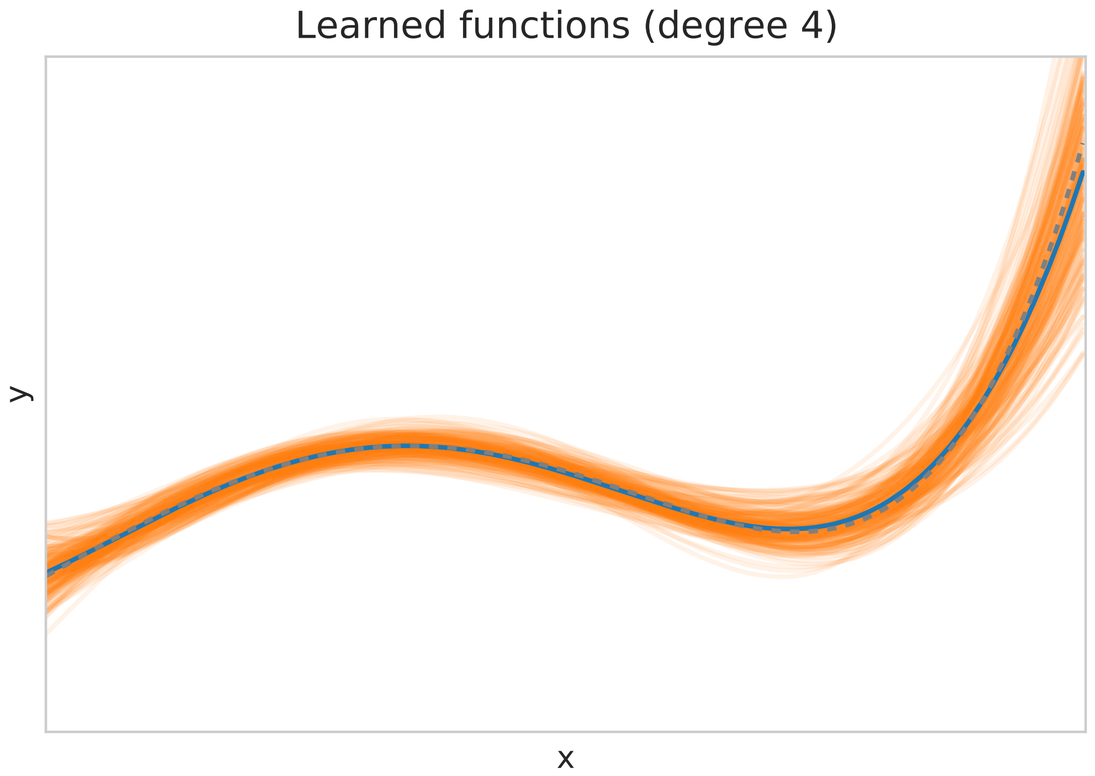
Complex models have low bias but high variance



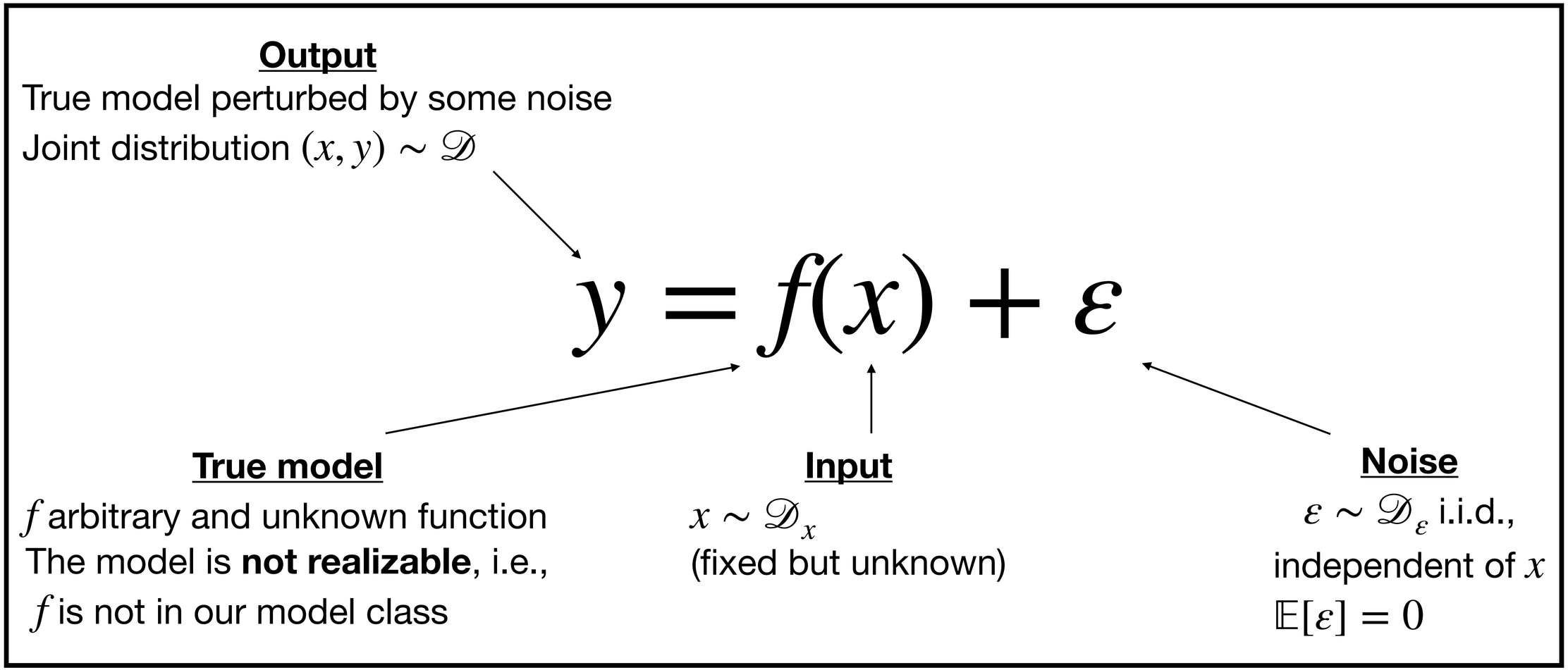
The average of the predictions f_S fits well the data: **small bias**

The variance of the predictions f_S as a function of S is large: **large variance**

We need to balance bias & variance correctly



Data model: output perturbed by some noise



We consider the square loss and will provide a decomposition of the true error

Error Decomposition

$S = \{(x_n, y_n)\}_{n=1}^{N} \sim \mathscr{D}$ i.i.d. $\quad \mathscr{A} \quad f_S = \mathscr{A}(S)$

We are interested in how the ***expected error*** of f_S:

$$\mathbb{E}_{(x,y)\sim\mathscr{D}}[(y - f_S(x))^2]$$

behaves as a ***function of the train set*** S and model class complexity

Error Decomposition

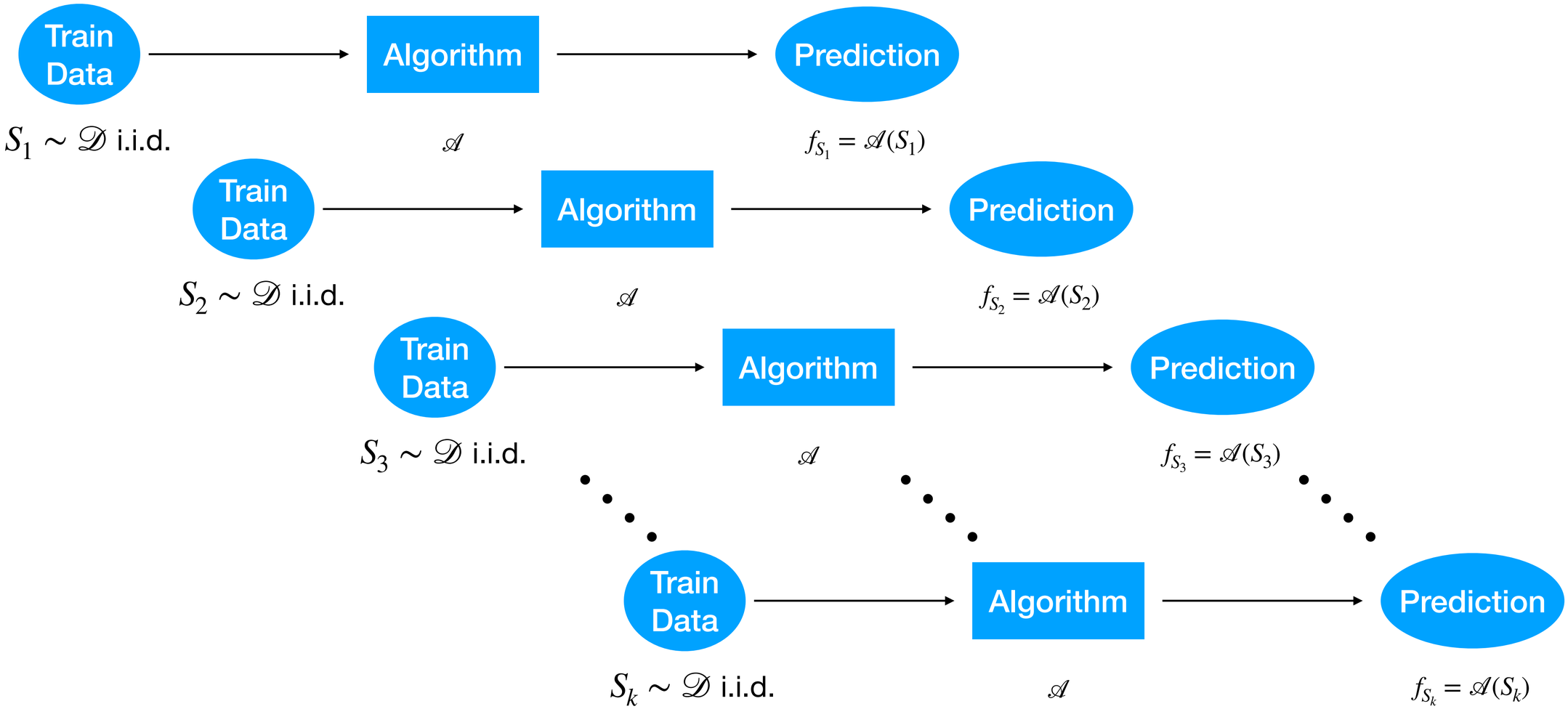
$S = \{(x_n, y_n)\}_{n=1}^{N} \sim \mathscr{D}$ i.i.d. $\quad \mathscr{A} \quad f_S = \mathscr{A}(S)$

The decomposition will hold true at ***every single point*** x. Therefore, to simplify, we consider the expected error of f_S for a fixed element x_0:

$$L(f_S) = \mathbb{E}_{\varepsilon \sim \mathscr{D}_\varepsilon}[(f(x_0) + \varepsilon - f_S(x_0))^2]$$

This is a random variable. The randomness comes for the train set S

We run the experiment many times



We are interested in the ***average*** and the ***variance*** of the ***predictions*** $(f_{S_1}, \cdots, f_{S_k})$ over these multiple runs

A decomposition in three terms

We are interested in the expectation of the true risk over the training set S

$$\begin{aligned}\mathbb{E}_{S\sim\mathscr{D}}[L(f_S)] &= \mathbb{E}_{S\sim\mathscr{D}}\left[\mathbb{E}_{\varepsilon\sim\mathscr{D}_\varepsilon}[(f(x_0)+\varepsilon-f_S(x_0))^2]\right] \\ &= \mathbb{E}_{S\sim\mathscr{D},\varepsilon\sim\mathscr{D}_\varepsilon}[(f(x_0)+\varepsilon-f_S(x_0))^2]\end{aligned}$$

We will decompose this quantity in ***three non-negative terms*** and will interpret each of these terms

First we expand the square:

$$\mathbb{E}_{S\sim\mathcal{D},\,\varepsilon\sim\mathcal{D}_\varepsilon}[(f(x_0)+\varepsilon-f_S(x_0))^2] = \mathbb{E}_{\varepsilon\sim\mathcal{D}_\varepsilon}[\varepsilon^2] + 2\mathbb{E}_{S\sim\mathcal{D},\,\varepsilon\sim\mathcal{D}_\varepsilon}[\varepsilon(f(x_0)-f_S(x_0))] + \mathbb{E}_{S\sim\mathcal{D}}[(f(x_0)-f_S(x_0))^2]$$

Using that $\mathbb{E}_{\varepsilon\sim\mathcal{D}_\varepsilon}[\varepsilon]=0$ and $\varepsilon \perp\!\!\!\perp S$:

- $\mathbb{E}_{\varepsilon\sim\mathcal{D}_\varepsilon}[\varepsilon^2] = \mathrm{Var}_{\varepsilon\sim\mathcal{D}_\varepsilon}[\varepsilon]$
- $\mathbb{E}_{S\sim\mathcal{D},\,\varepsilon\sim\mathcal{D}_\varepsilon}[\varepsilon(f(x_0)-f_S(x_0))] = \mathbb{E}_{\varepsilon\sim\mathcal{D}_\varepsilon}[\varepsilon]\times\mathbb{E}_{S\sim\mathcal{D}}[f(x_0)-f_S(x_0)] = 0$

Therefore

$$\boxed{\mathbb{E}_{S\sim\mathcal{D},\,\varepsilon\sim\mathcal{D}_\varepsilon}[(f(x_0)+\varepsilon-f_S(x_0))^2] = \mathrm{Var}_{\varepsilon\sim\mathcal{D}_\varepsilon}[\varepsilon] + \mathbb{E}_{S\sim\mathcal{D}}[(f(x_0)-f_S(x_0))^2]}$$

Trick: we add and subtract the constant term $\mathbb{E}_{S' \sim \mathcal{D}}[f_{S'}(x_0)]$, where S' is a second training set independent from S

$$\begin{aligned}
\mathbb{E}_{S\sim\mathcal{D}}[(f(x_0) - f_S(x_0))^2] &= \mathbb{E}_{S\sim\mathcal{D}}\left[(f(x_0) - \mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)] + \mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)] - f_S(x_0))^2\right] \\
&= \mathbb{E}_{S\sim\mathcal{D}}\Big[(f(x_0) - \mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)])^2 + (\mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)] - f_S(x_0))^2 \\
&\quad + 2(f(x_0) - \mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)])(\mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)] - f_S(x_0))\Big]
\end{aligned}$$

Cross-term:

$$\begin{aligned}
&\mathbb{E}_{S\sim\mathcal{D}}\Big[\big(f(x_0) - \mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)]\big) \cdot \big(\mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)] - f_S(x_0)\big)\Big] \\
&= \big(f(x_0) - \mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)]\big) \cdot \mathbb{E}_{S\sim\mathcal{D}}\big[(\mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)] - f_S(x_0))\big] \\
&= \big(f(x_0) - \mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)]\big) \cdot \big(\mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)] - \mathbb{E}_{S\sim\mathcal{D}}[f_S(x_0)]\big) = 0.
\end{aligned}$$

$$\boxed{\mathbb{E}_{S\sim\mathcal{D}}[(f(x_0) - f_S(x_0))^2] = (f(x_0) - \mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)])^2 + \mathbb{E}_{S\sim\mathcal{D}}[(\mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)] - f_S(x_0))^2]}$$

Bias-Variance Decomposition

We obtain the following decomposition into three positive terms:

$$\mathbb{E}_{S\sim\mathcal{D},\,\varepsilon\sim\mathcal{D}_\varepsilon}[(f(x_0)+\varepsilon-f_S(x_0))^2] \;=\; \mathrm{Var}_{\varepsilon\sim\mathcal{D}_\varepsilon}[\varepsilon] \leftarrow \textbf{Noise variance}$$

$$\textbf{Bias} \longrightarrow \;+\; (f(x_0)-\mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)])^2$$

$$\textbf{Variance} \longrightarrow \;+\; \mathbb{E}_{S\sim\mathcal{D}}\left[(f_S(x_0)-\mathbb{E}_{S'\sim\mathcal{D}}[f_{S'}(x_0)])^2\right]$$

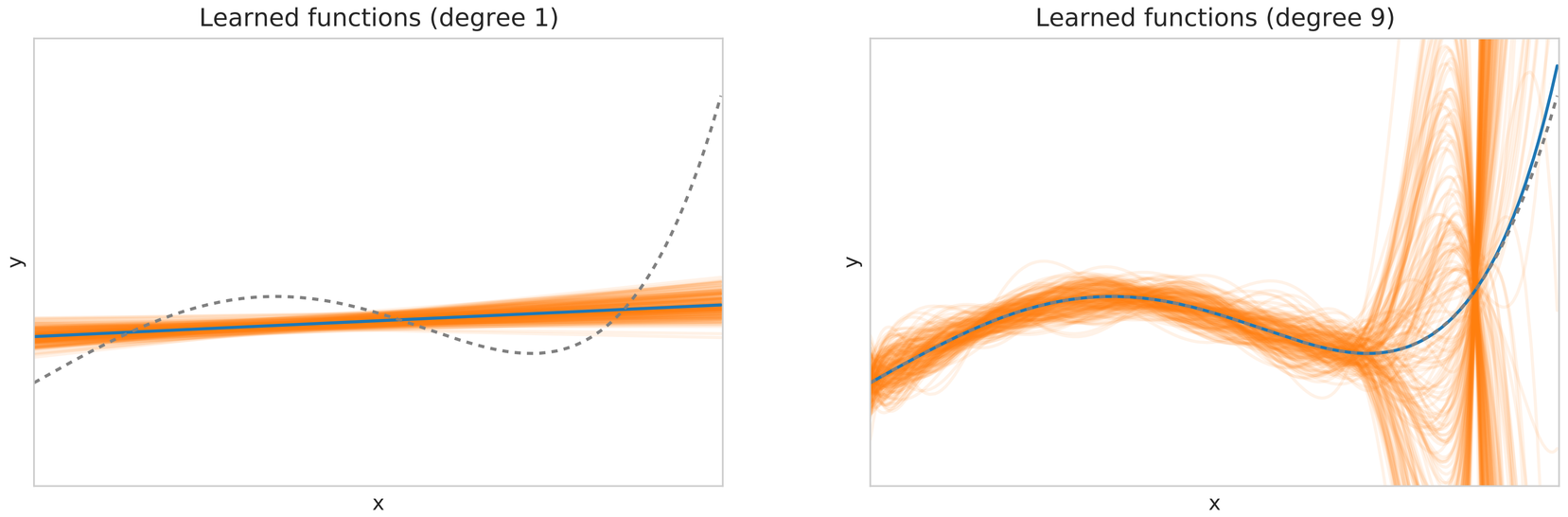
each of which always provides a lower bound of the true error

➡ To minimize the true error, we must choose a method that achieves **low bias and low variance** simultaneously

Noise: a strict lower bound on the achievable error

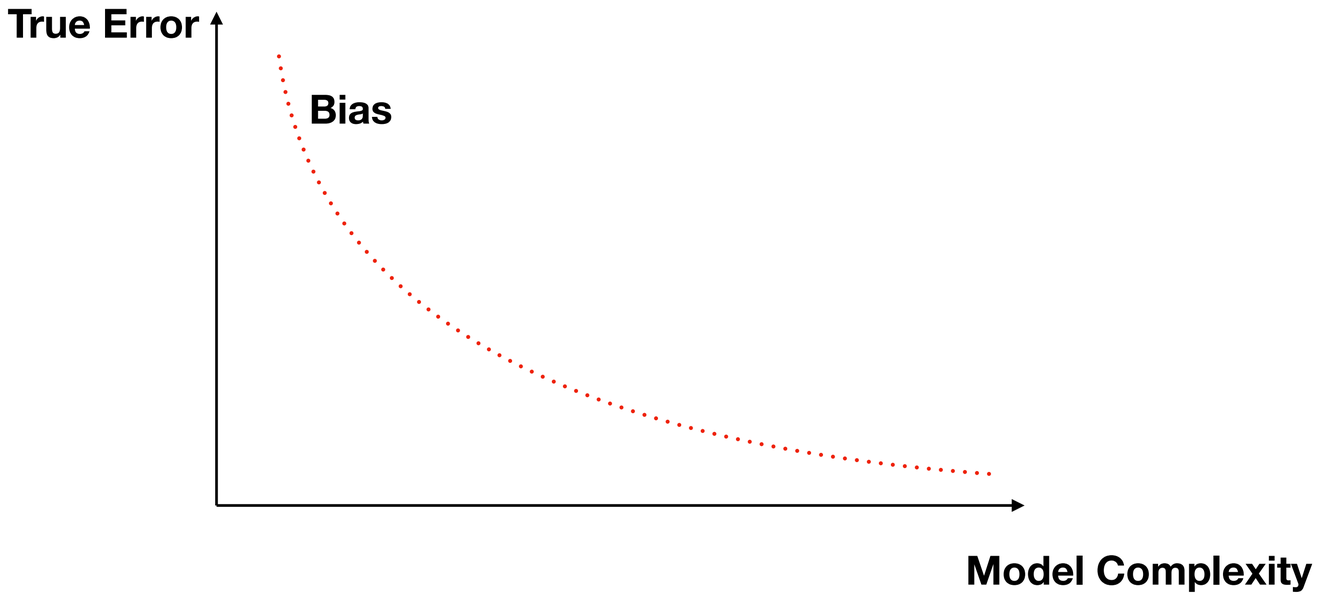
- It is not possible to go below the noise level
- Even if we know the true model f, we still suffer from the noise: $L(f) = \mathbb{E}[\varepsilon^2]$
- It is not possible to predict the noise from the data since they are independent

Bias: $(f(x_0) - \mathbb{E}_{S \sim \mathcal{D}}[f_S(x_0)])^2$



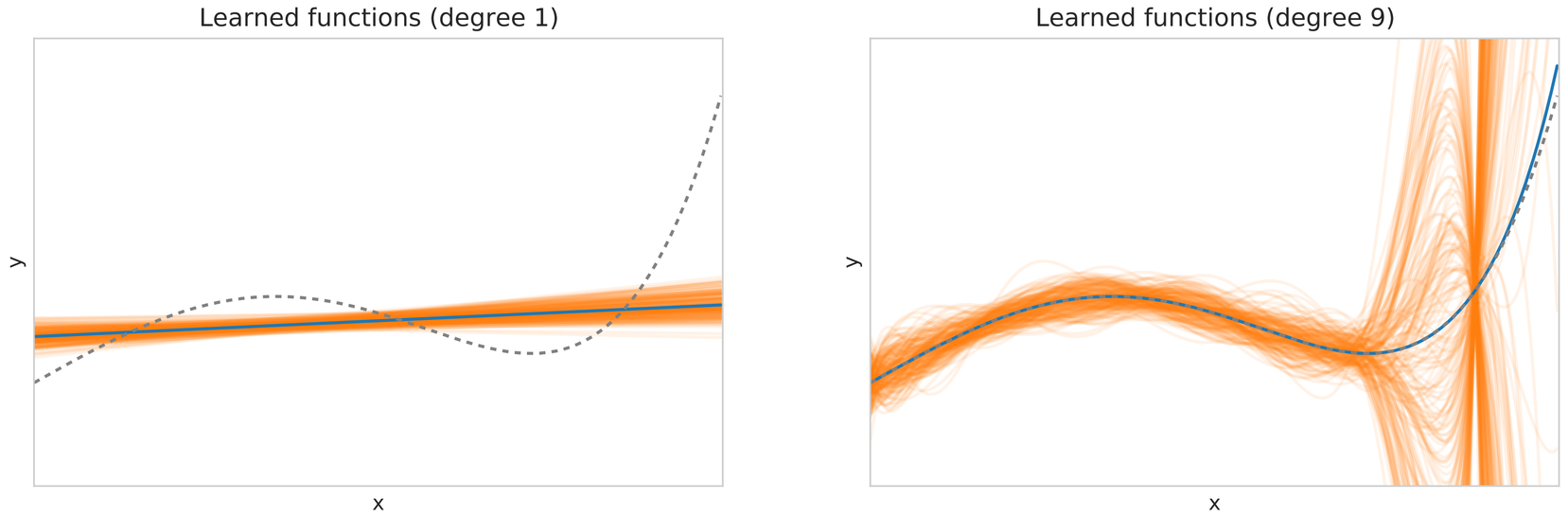
- Squared of the difference between the actual value $f(x_0)$ and the expected prediction
- It measures how far off in general the models' predictions are from the correct value
- If model **complexity** is **low**, **bias** is typically **high**
- If model **complexity** is **high**, **bias** Is typically **low**

Bias: $(f(x_0) - \mathbb{E}_{S \sim \mathscr{D}}[f_S(x_0)])^2$



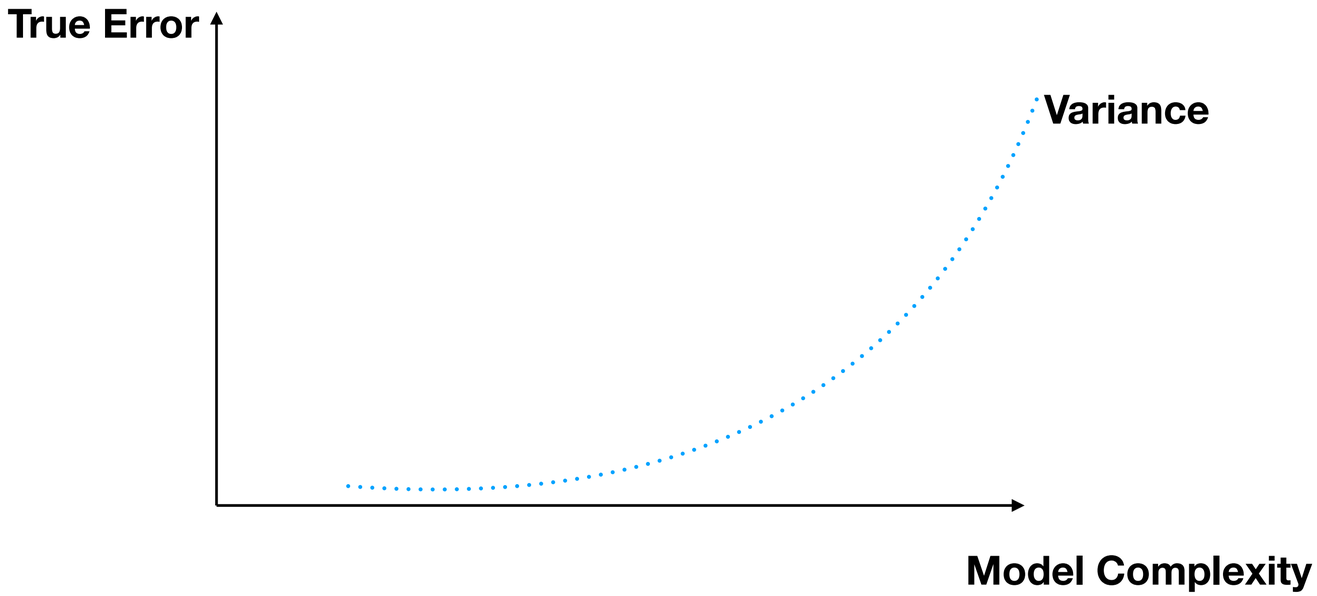
- Squared of the difference between the actual value $f(x_0)$ and the expected prediction
- It measures how far off in general the models' predictions are from the correct value
- If model **complexity** is **low**, **bias** is typically **high**
- If model **complexity** is **high**, **bias** is typically **low**

Variance: $\mathbb{E}_{S\sim\mathcal{D}}\left[(f_S(x_0) - \mathbb{E}_{S\sim\mathcal{D}}[f_S(x_0)])^2\right]$



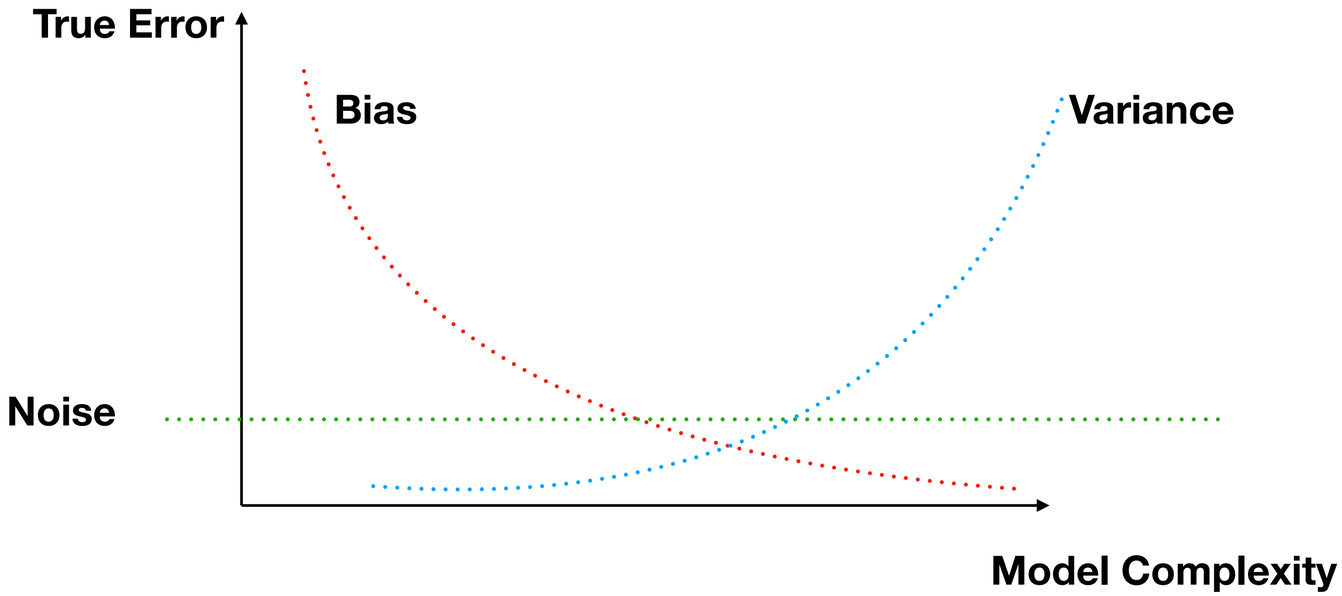
- Variance of the prediction function
- It measures the variability of predictions at a given point across different training set realizations
- If we consider complex models, small variations in the training set can lead to significant changes in the predictions

Variance: $\mathbb{E}_{S\sim\mathcal{D}}\left[(f_S(x_0) - \mathbb{E}_{S\sim\mathcal{D}}[f_S(x_0)])^2\right]$



- Variance of the prediction function
- It measures the variability of predictions at a given point across different training set realizations
- If we consider complex models, small variations in the training set can lead to significant changes in the predictions

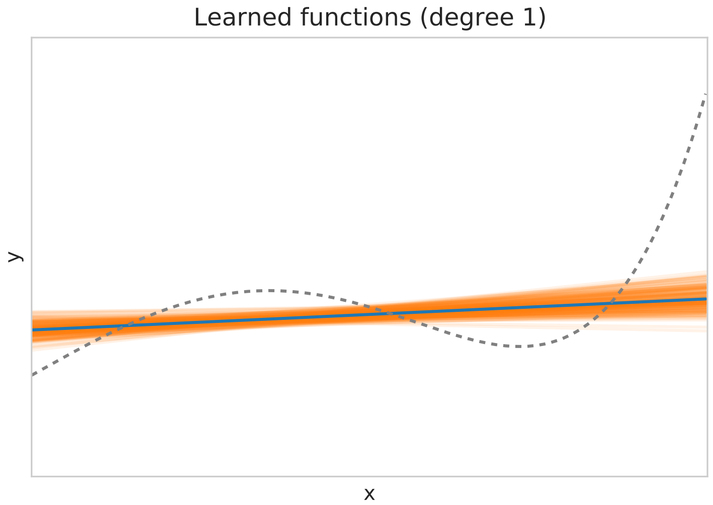
Bias Variance tradeoff and U-shape curve

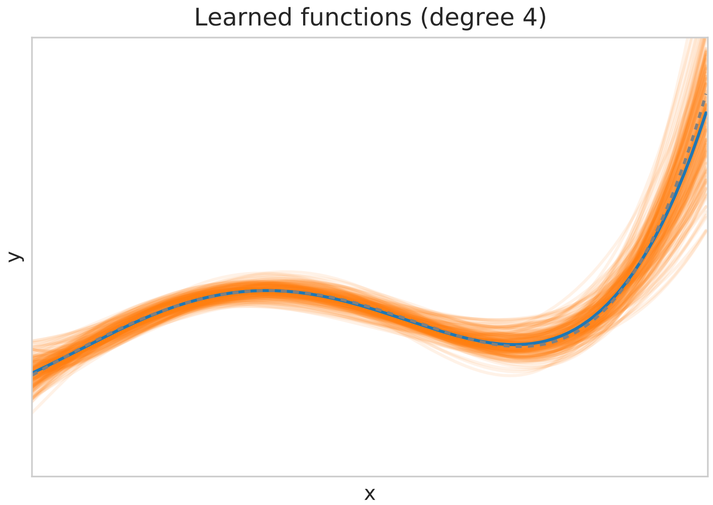


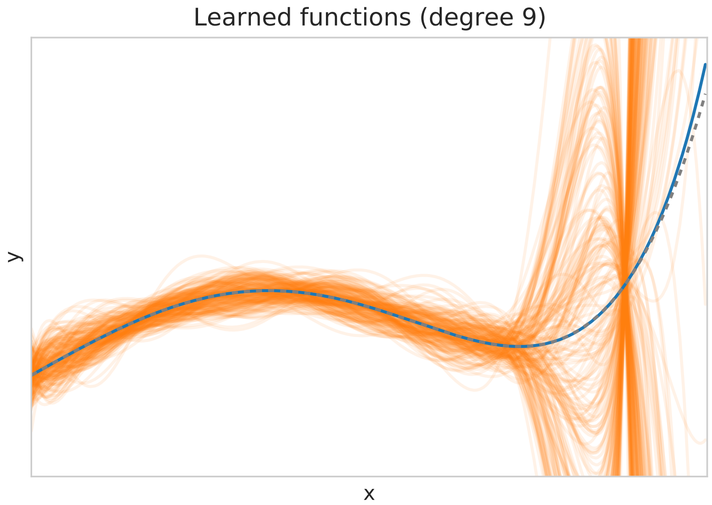
- If model complexity is too low, approximation will be poor (underfitting)
- If model complexity is too high, it may cause issues with variance (overfitting)

➡ This phenomenon is known as the bias-variance tradeoff

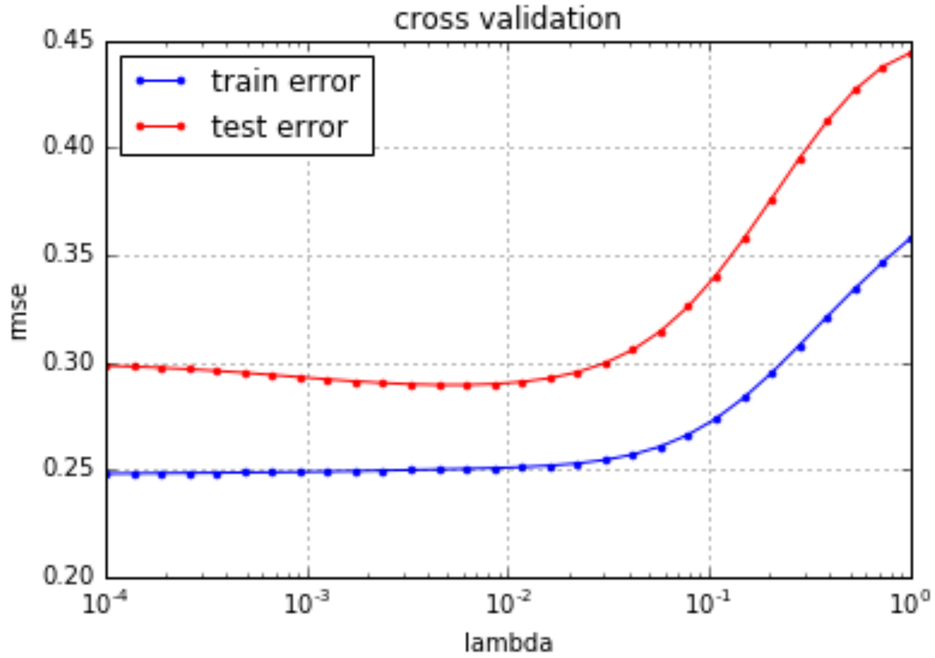
Challenge: Identify a method that ensures both low variance and low bias



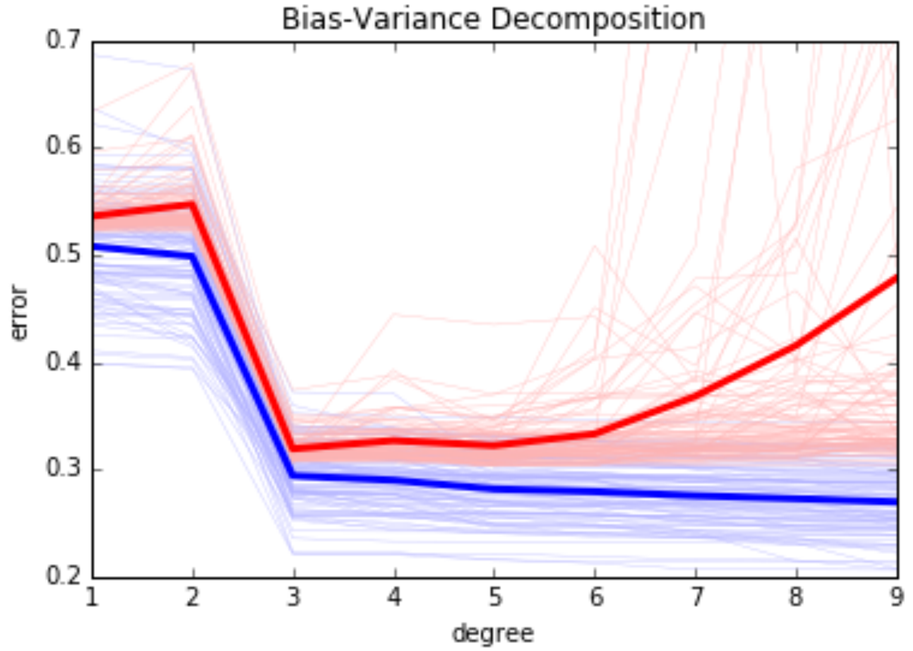




Model selection curves

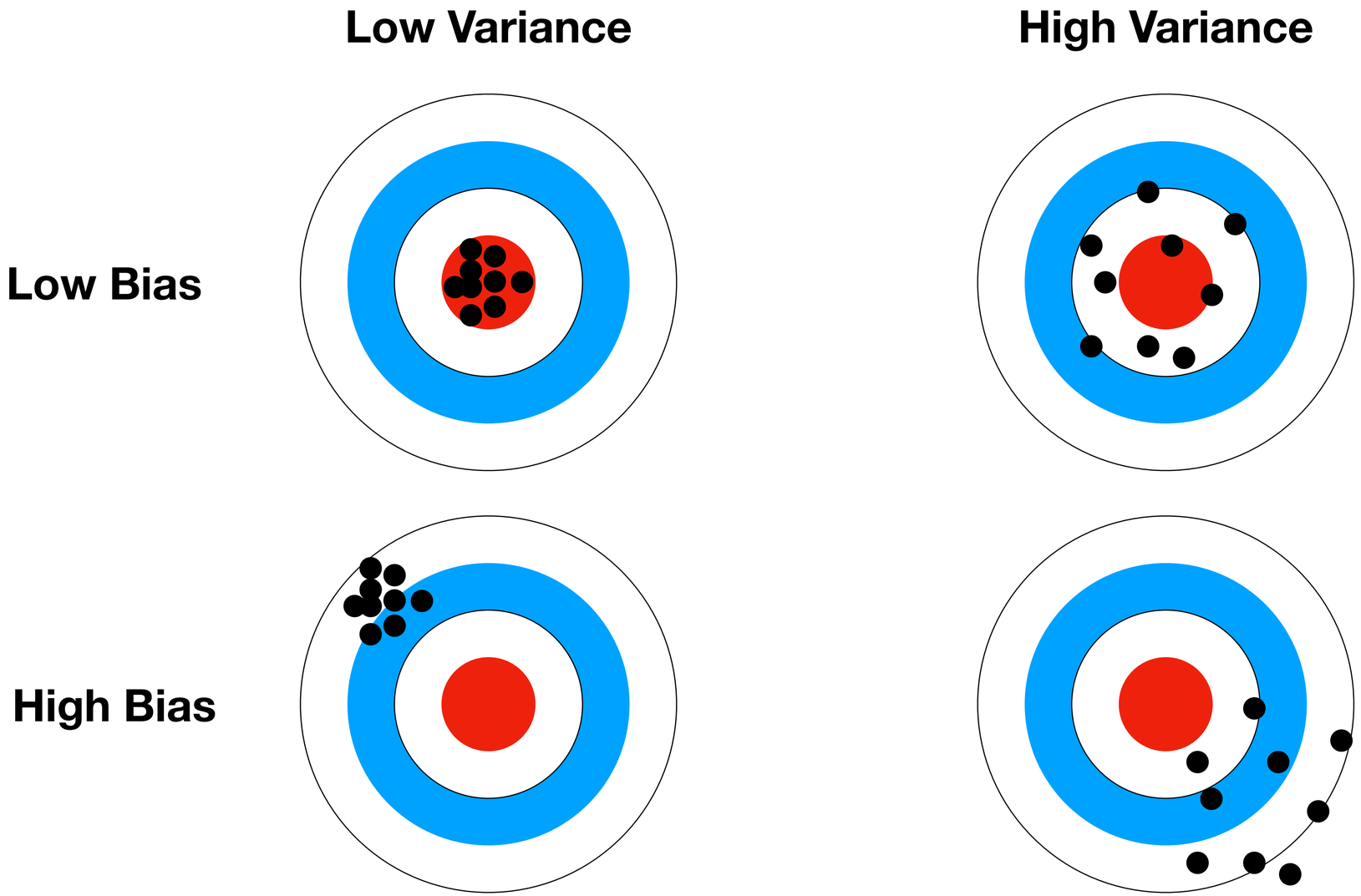


Ridge regression



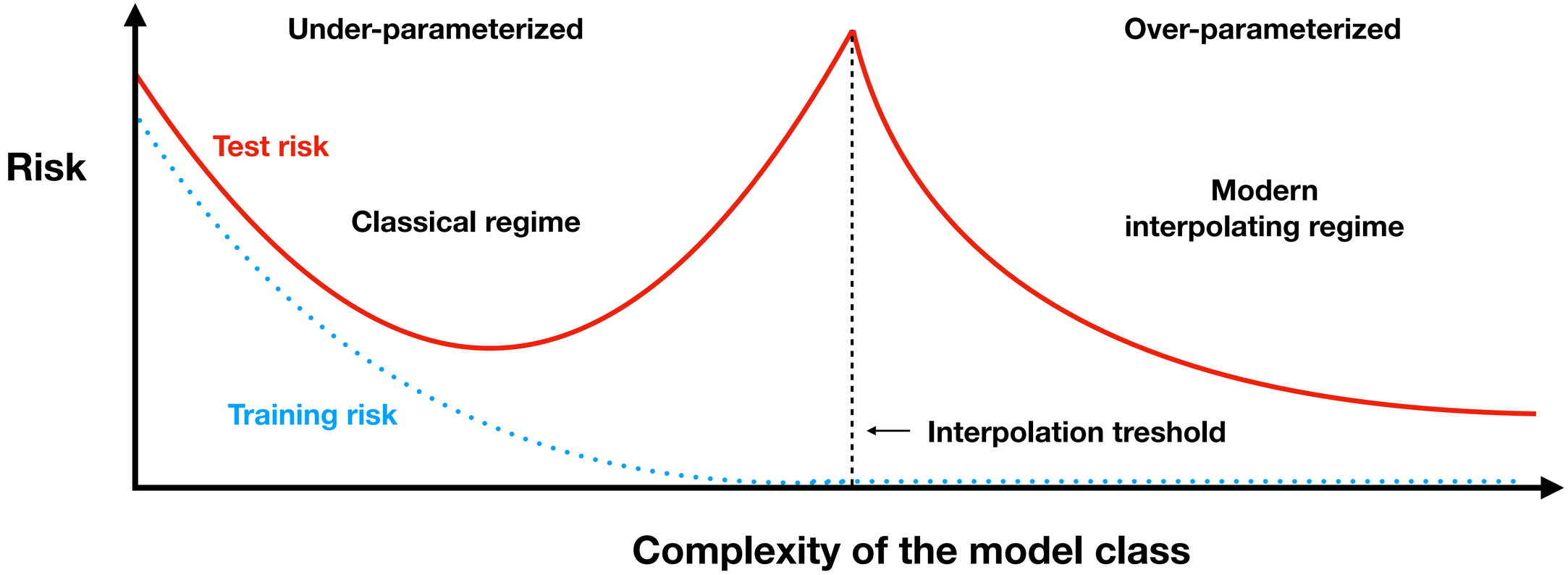
Degree in case of a polynomial feature expansion

Conclusion



But this depends on the algorithm!

Double descent curve



Reconciling modern machine-learning practice and the classical bias–variance trade-off

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal, PNAS, 2019