Lecture 4: Generalization and Model Selection

Tao LIN

SoE, Westlake University

March 17, 2025

Reading materials

- Chapter 3.2, Bishop, Pattern Recognition and Machine Learning
- Chapter 2, Stanford CS-229,
  https://cs229.stanford.edu/notes2022fall/main_notes.pdf
- Bias-Variance decomposition by Scott Fortmann-Roe:
  http://scott.fortmann-roe.com/docs/BiasVariance.html
- Double-descent phenomenon by Mikhail Belkin et al:
  https://www.pnas.org/content/116/32/15849.short

Reference

- EPFL, CS-433 Machine Learning, `https://github.com/epfml/ML_course`

Table of Contents

Table of Contents

Definition 1 (*Learning* problem can be formulated as optimization problem)

Given a cost function $\mathcal{L}(\mathbf{w})$, we wish to find $\mathbf{w}^\star$ which minimizes the cost:

$$\min_{\mathbf{w}} \ \mathcal{L}(\mathbf{w}) \quad \text{subject to } \mathbf{w} \in \mathbb{R}^D \tag{1}$$

Definition 1 (*Learning* problem can be formulated as optimization problem)

Given a cost function $\mathcal{L}(\mathbf{w})$, we wish to find $\mathbf{w}^\star$ which minimizes the cost:

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \quad \text{subject to } \mathbf{w} \in \mathbb{R}^D \tag{1}$$

We will use an optimization algorithm (e.g., Gradient Descent) to find a good $\mathbf{w}$.

Definition 1 (*Learning* problem can be formulated as optimization problem)

Given a cost function $\mathcal{L}(\mathbf{w})$, we wish to find $\mathbf{w}^\star$ which minimizes the cost:

$$\min_{\mathbf{w}} \ \mathcal{L}(\mathbf{w}) \quad \text{subject to } \mathbf{w} \in \mathbb{R}^D \tag{1}$$

We will use an optimization algorithm (e.g., Gradient Descent) to find a good $\mathbf{w}$.

Considering a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ and learnable weights $\mathbf{w} \in \mathbb{R}^D$ for $f_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w}$.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N, \qquad \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix} \in \mathbb{R}^{N \times D} \tag{2}$$

Using Gradient Descent for Linear Regression with MSE

The MSE is defined as:

$$\mathcal{L}(\mathbf{w}) := \frac{1}{2N} \sum_{n=1}^{N} \left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 = \tfrac{1}{2N} \mathbf{e}^\top \mathbf{e}\,, \tag{3}$$

where the error vector $\mathbf{e}$ is defined as:

$$\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{w} = \begin{pmatrix} e_1 \\ \vdots \\ e_N \end{pmatrix} \in \mathbb{R}^N\,, \tag{4}$$

and $e_i := y_n - \mathbf{x}_n^\top \mathbf{w}$.

Using Gradient Descent for Linear Regression with MSE

The MSE is defined as:

$$\mathcal{L}(\mathbf{w}) := \frac{1}{2N}\sum_{n=1}^{N}\left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 = \tfrac{1}{2N}\mathbf{e}^\top \mathbf{e}\,, \tag{3}$$

where the error vector $\mathbf{e}$ is defined as:

$$\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{w} = \begin{pmatrix} e_1 \\ \vdots \\ e_N \end{pmatrix} \in \mathbb{R}^N\,, \tag{4}$$

and $e_i := y_n - \mathbf{x}_n^\top \mathbf{w}$. The gradient is given by

$$\nabla\mathcal{L}(\mathbf{w}) = -\frac{1}{N}\mathbf{X}^\top \mathbf{e} \tag{5}$$

A probabilistic model for linear regression

Definition 2 (Data generation process)

We assume that the data is generated by the model,

$$y_n = \mathbf{x}_n^\top \mathbf{w} + \epsilon_n, \tag{6}$$

where

- the ϵ_n (the noise) is a zero-mean Gaussian random variable with variance σ^2
- the noise is independent of each other and independent of the input.
- the model $\mathbf{w}$ is unknown.

A probabilistic model for linear regression

Definition 2 (Data generation process)

We assume that the data is generated by the model,

$$y_n = \mathbf{x}_n^\top \mathbf{w} + \epsilon_n, \tag{6}$$

where

- the ϵ_n (the noise) is a zero-mean Gaussian random variable with variance σ^2
- the noise is independent of each other and independent of the input.
- the model $\mathbf{w}$ is unknown.

The likelihood of the data vector $\mathbf{y} = (y_1, \cdots, y_N)$ given the input $\mathbf{X}$ and the model $\mathbf{w}$ is

$$p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n \mid \mathbf{x}_n, \mathbf{w}) = \prod_{n=1}^{N} \mathcal{N}(y_n \mid \mathbf{x}_n^\top \mathbf{w}, \sigma^2). \tag{7}$$

A probabilistic model for linear regression

Definition 2 (Data generation process)

We assume that the data is generated by the model,

$$y_n = \mathbf{x}_n^\top \mathbf{w} + \epsilon_n, \tag{6}$$

where

- the ϵ_n (the noise) is a zero-mean Gaussian random variable with variance σ^2
- the noise is independent of each other and independent of the input.
- the model $\mathbf{w}$ is unknown.

The likelihood of the data vector $\mathbf{y} = (y_1, \cdots, y_N)$ given the input $\mathbf{X}$ and the model $\mathbf{w}$ is

$$p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n \mid \mathbf{x}_n, \mathbf{w}) = \prod_{n=1}^{N} \mathcal{N}(y_n \mid \mathbf{x}_n^\top \mathbf{w}, \sigma^2). \tag{7}$$

The probabilistic view point: maximize this likelihood over the choice of model $\mathbf{w}$.

The probabilistic interpretation of least-squares/linear regression

Least-Squares linear regression can be interpreted as the Maximum Likelihood Estimator:

$$\mathbf{w}_{\text{lse}} \overset{(a)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}, \mathbf{X}|\mathbf{w})$$ (by the definition of log-likelihood)

(8)

The probabilistic interpretation of least-squares/linear regression

Least-Squares linear regression can be interpreted as the Maximum Likelihood Estimator:

$$\begin{aligned}
\mathbf{w}_{\text{lse}} &\overset{(a)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}, \mathbf{X}|\mathbf{w}) && \text{(by the definition of log-likelihood)} \\
&\overset{(b)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{X}|\mathbf{w}) p(\mathbf{y}|\mathbf{X}, \mathbf{w}) && \text{(by factoring the likelihood)}
\end{aligned} \tag{8}$$

The probabilistic interpretation of least-squares/linear regression

Least-Squares linear regression can be interpreted as the Maximum Likelihood Estimator:

$$
\begin{aligned}
\mathbf{w}_{\text{lse}} &\overset{(a)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}, \mathbf{X} | \mathbf{w}) && \text{(by the definition of log-likelihood)} \\
&\overset{(b)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{X} | \mathbf{w}) p(\mathbf{y} | \mathbf{X}, \mathbf{w}) && \text{(by factoring the likelihood)} \\
&\overset{(c)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{X}) p(\mathbf{y} | \mathbf{X}, \mathbf{w}) && \text{(the choice of the input } \mathbf{x}_n \text{ is independent of } \mathbf{w}\text{)}
\end{aligned}
\tag{8}
$$

The probabilistic interpretation of least-squares/linear regression

Least-Squares linear regression can be interpreted as the Maximum Likelihood Estimator:

$$
\begin{aligned}
\mathbf{w}_{\text{lse}} &\overset{(a)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}, \mathbf{X}|\mathbf{w}) && \text{(by the definition of log-likelihood)} \\
&\overset{(b)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{X}|\mathbf{w}) p(\mathbf{y}|\mathbf{X}, \mathbf{w}) && \text{(by factoring the likelihood)} \\
&\overset{(c)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{X}) p(\mathbf{y}|\mathbf{X}, \mathbf{w}) && \text{(the choice of the input } \mathbf{x}_n \text{ is independent of } \mathbf{w}) \\
&\overset{(d)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}|\mathbf{X}, \mathbf{w}) && (p(\mathbf{X}) \text{ is independent of } \mathbf{w})
\end{aligned}
\tag{8}
$$

The probabilistic interpretation of least-squares/linear regression

Least-Squares linear regression can be interpreted as the Maximum Likelihood Estimator:

$$
\begin{aligned}
\mathbf{w}_{\text{lse}} &\overset{(a)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}, \mathbf{X}|\mathbf{w}) && \text{(by the definition of log-likelihood)} \\
&\overset{(b)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{X}|\mathbf{w}) p(\mathbf{y}|\mathbf{X}, \mathbf{w}) && \text{(by factoring the likelihood)} \\
&\overset{(c)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{X}) p(\mathbf{y}|\mathbf{X}, \mathbf{w}) && \text{(the choice of the input } \mathbf{x}_n \text{ is independent of } \mathbf{w}) \\
&\overset{(d)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}|\mathbf{X}, \mathbf{w}) && (p(\mathbf{X}) \text{ is independent of } \mathbf{w}) \\
&\overset{(e)}{=} \arg\min_{\mathbf{w}} \quad -\log \left[\prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathbf{w}) \right] && \text{(we assume samples are iid.)}
\end{aligned}
\tag{8}
$$

The probabilistic interpretation of least-squares/linear regression

Least-Squares linear regression can be interpreted as the Maximum Likelihood Estimator:

$$
\begin{aligned}
\mathbf{w}_{\text{lse}} &\overset{(a)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}, \mathbf{X}|\mathbf{w}) && \text{(by the definition of log-likelihood)}\\
&\overset{(b)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{X}|\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) && \text{(by factoring the likelihood)}\\
&\overset{(c)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{X})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) && \text{(the choice of the input } \mathbf{x}_n \text{ is independent of } \mathbf{w})\\
&\overset{(d)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}|\mathbf{X}, \mathbf{w}) && (p(\mathbf{X}) \text{ is independent of } \mathbf{w})\\
&\overset{(e)}{=} \arg\min_{\mathbf{w}} \quad -\log \left[\prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathbf{w})\right] && \text{(we assume samples are iid.)}\\
&= \arg\min_{\mathbf{w}} \quad \frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 && \text{(by definition and calculus)}
\end{aligned}
\tag{8}
$$

Maximum-Likelihood Estimator (MLE)

Instead of maximizing the likelihood, we can maximize the logarithm of the likelihood, i.e., log-likelihood (LL):

$$\mathcal{L}_{\mathrm{LL}}(\mathbf{w}) := \log p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \mathrm{cnst}\,. \tag{9}$$

Maximum-Likelihood Estimator (MLE)

Instead of maximizing the likelihood, we can maximize the logarithm of the likelihood, i.e., log-likelihood (LL):

$$\mathcal{L}_{\text{LL}}(\mathbf{w}) := \log p(\mathbf{y} \,|\, \mathbf{X}, \mathbf{w}) = -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \text{cnst} \,. \tag{9}$$

Compare the LL to the MSE (Mean Squared Error)

$$\mathcal{L}_{\text{LL}}(\mathbf{w}) = -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \text{cnst} \tag{10}$$

$$\mathcal{L}_{\text{MSE}}(\mathbf{w}) = \frac{1}{2N} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 \tag{11}$$

Maximum-Likelihood Estimator (MLE)

Instead of maximizing the likelihood, we can maximize the logarithm of the likelihood, i.e., log-likelihood (LL):

$$\mathcal{L}_{\text{LL}}(\mathbf{w}) := \log p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \text{cnst}\,. \tag{9}$$

Compare the LL to the MSE (Mean Squared Error)

$$\mathcal{L}_{\text{LL}}(\mathbf{w}) = -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \text{cnst} \tag{10}$$

$$\mathcal{L}_{\text{MSE}}(\mathbf{w}) = \frac{1}{2N} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 \tag{11}$$

Maximizing the LL is equivalent to minimizing the MSE:

$$\arg\min_{\mathbf{w}} \mathcal{L}_{\text{MSE}}(\mathbf{w}) = \arg\max_{\mathbf{w}} \mathcal{L}_{\text{LL}}(\mathbf{w})\,. \tag{12}$$

Table of Contents

Motivation

- In rare cases, one can compute the optimum of the cost function analytically.
- Linear regression using an MSE cost function is one such case.
- Here its solution can be obtained explicitly, by solving a linear system of equations.

$\Rightarrow$ These equations are sometimes called the normal equations.

$\Rightarrow$ Solving the normal equations is called the least squares.

Normal Equations

To derive the normal equations,

1. we first show that the problem is convex.

2. we then use the optimality conditions for convex functions, i.e.,

$$\nabla \mathcal{L}(\mathbf{w}^\star) = \mathbf{0}\,, \tag{13}$$

   where $\mathbf{w}^\star$ corresponds to the parameter at the optimum point.

Normal Equations

To derive the normal equations,

1. we first show that the problem is convex.

2. we then use the optimality conditions for convex functions, i.e.,

$$\nabla \mathcal{L}(\mathbf{w}^\star) = \mathbf{0}\,, \tag{13}$$

where $\mathbf{w}^\star$ corresponds to the parameter at the optimum point.

Given the definition $\mathcal{L}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 = \frac{1}{2N}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w})$, we have

$$\nabla \mathcal{L}(\mathbf{w}) = -\frac{1}{N}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = 0\,, \tag{14}$$

where we can get the normal equations for linear regression.

Least Squares

We need to solve the linear system of the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$, where

$$\mathbf{X}^\top y = \underbrace{\mathbf{X}^\top\mathbf{X}}_{\text{Gram matrix}} \mathbf{w} \tag{15}$$

Least Squares

We need to solve the linear system of the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$, where

$$\mathbf{X}^\top y = \underbrace{\mathbf{X}^\top\mathbf{X}}_{\text{Gram matrix}} \mathbf{w} \tag{15}$$

If the Gram matrix is invertible, we can multiply the normal equation by the inverse of the Gram matrix from the left:

$$\mathbf{w}^\star = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}\,, \tag{16}$$

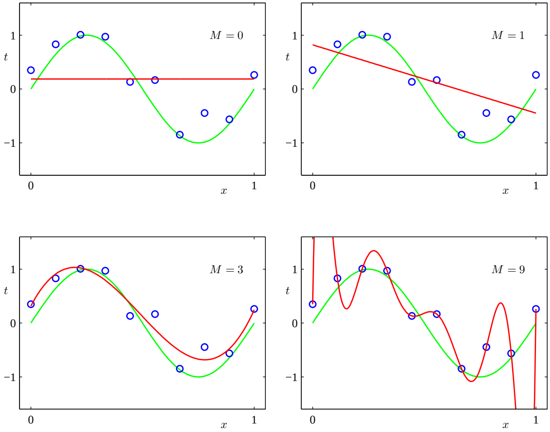
where we can get a closed-form expression for the minimum.

Table of Contents

Definitions of under-fit and over-fit

Let's consider the polynomial regression problem (for a one-dimensional input x_n):

$$
\begin{aligned}
y_n &\approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M \\
&=: \phi(x_n)^\top \mathbf{w} \,.
\end{aligned}
\tag{17}
$$

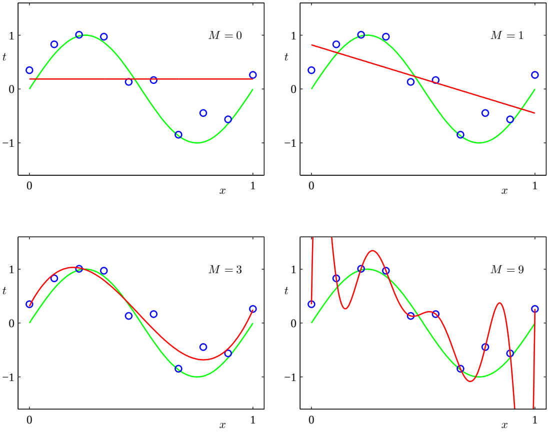


Definitions of under-fit and over-fit

Let's consider the polynomial regression problem (for a one-dimensional input x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M$$
$$=: \phi(x_n)^\top \mathbf{w} \,. \tag{17}$$

- under-fit: cannot find the *the underlying function* of the data.

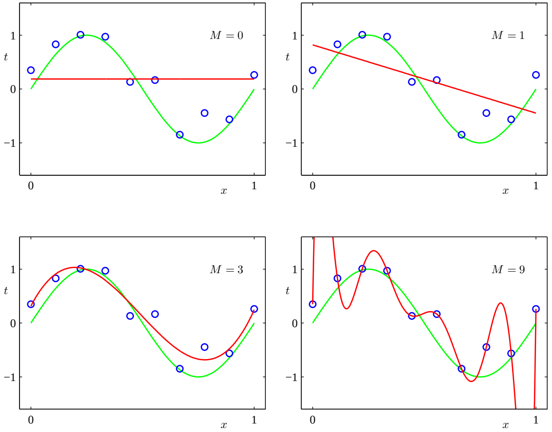


Definitions of under-fit and over-fit

Let's consider the polynomial regression problem (for a one-dimensional input x_n):

$$\begin{aligned} y_n &\approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M \\ &=: \phi(x_n)^\top \mathbf{w} \,. \end{aligned} \tag{17}$$

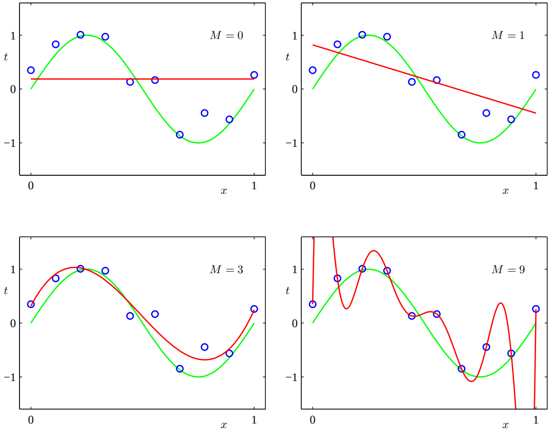
- under-fit: cannot find the *the underlying function* of the data.
- over-fit: can fit both *the underlying function* and *the noise in the data*.



Definitions of under-fit and over-fit

Let's consider the polynomial regression problem (for a one-dimensional input x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M \\ =: \phi(x_n)^\top \mathbf{w}\,. \tag{17}$$



- under-fit: cannot find the *the underlying function* of the data.
- over-fit: can fit both *the underlying function* and *the noise in the data*.

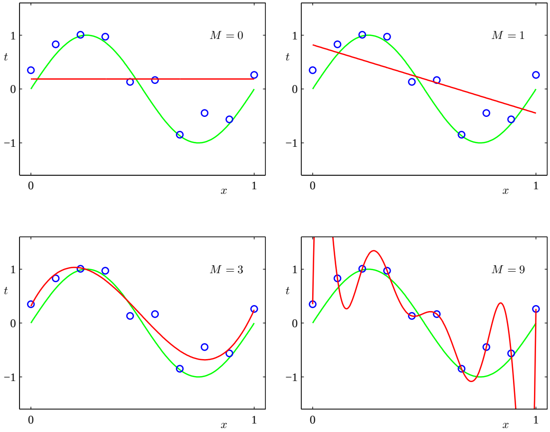
Discussion:

- Both of these phenomena (under-fit and over-fit) are undesirable.

Definitions of under-fit and over-fit

Let's consider the polynomial regression problem (for a one-dimensional input x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M \\ =: \phi(x_n)^\top \mathbf{w} \,. \tag{17}$$



- under-fit: cannot find the *the underlying function* of the data.
- over-fit: can fit both *the underlying function* and *the noise in the data*.

Discussion:

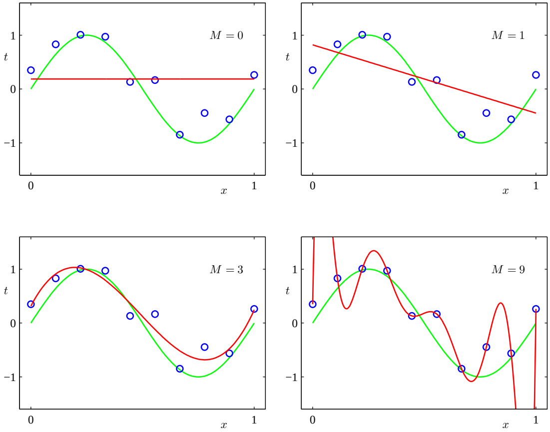
- Both of these phenomena (under-fit and over-fit) are undesirable.
- This discussion is made more difficult:

Definitions of under-fit and over-fit

Let's consider the polynomial regression problem (for a one-dimensional input x_n):

$$
\begin{aligned}
y_n &\approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M \\
&=: \phi(x_n)^\top \mathbf{w}\,.
\end{aligned}
\tag{17}
$$

- under-fit: cannot find the *the underlying function* of the data.
- over-fit: can fit both *the underlying function* and *the noise in the data*.



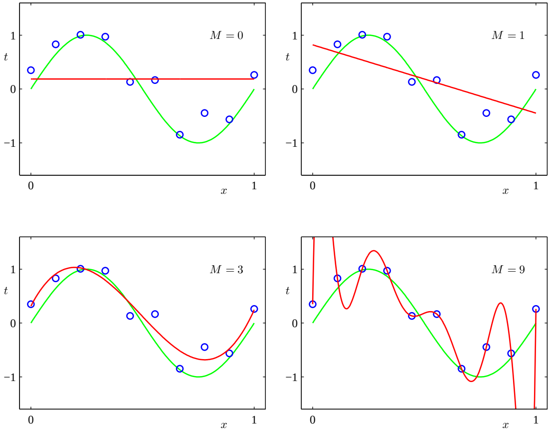
Discussion:

- Both of these phenomena (under-fit and over-fit) are undesirable.
- This discussion is made more difficult:
  - since all we have is data.

Definitions of under-fit and over-fit

Let's consider the polynomial regression problem (for a one-dimensional input x_n):

$$
\begin{aligned}
y_n &\approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M \\
&=: \phi(x_n)^\top \mathbf{w} \, .
\end{aligned} \tag{17}
$$



- under-fit: cannot find the *the underlying function* of the data.
- over-fit: can fit both *the underlying function* and *the noise in the data*.

Discussion:

- Both of these phenomena (under-fit and over-fit) are undesirable.
- This discussion is made more difficult:
  - since all we have is data.
  - we do not know a priori what part is the underlying signal and what part is noise.

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{18}$$

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{18}$$

- Ω is a regularizer, and will measure the complexity of the model given by $\mathbf{w}$.

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{18}$$

- Ω is a regularizer, and will measure the complexity of the model given by $\mathbf{w}$.
- **Examples:**

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{18}$$

- Ω is a regularizer, and will measure the complexity of the model given by $\mathbf{w}$.
- **Examples:**
  - L_2-regularization (Ridge Regression)

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_2^2 \; , \tag{19}$$

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{18}$$

- Ω is a regularizer, and will measure the complexity of the model given by $\mathbf{w}$.
- **Examples:**
  - L_2-regularization (Ridge Regression)

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_2^2 \; , \tag{19}$$

Exercise $\rightarrow$

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{18}$$

- Ω is a regularizer, and will measure the complexity of the model given by $\mathbf{w}$.
- **Examples:**
  - L_2-regularization (Ridge Regression)

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \left\|\mathbf{y} - \mathbf{X}\mathbf{w}\right\|_2^2 \; + \; \lambda \left\|\mathbf{w}\right\|_2^2 \,, \tag{19}$$

Exercise $\rightarrow$ the explicit solution is defined as $\mathbf{w}^{\star}_{\text{ridge}} = (\mathbf{X}^{\top}\mathbf{X} + \lambda'\mathbf{I})^{-1}\mathbf{X}^{\top}\mathbf{y}$.

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{18}$$

- Ω is a regularizer, and will measure the complexity of the model given by $\mathbf{w}$.
- **Examples:**
  - L_2-regularization (Ridge Regression)

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_2^2 \; , \tag{19}$$

Exercise $\rightarrow$ the explicit solution is defined as $\mathbf{w}^{\star}_{\text{ridge}} = (\mathbf{X}^{\top}\mathbf{X} + \lambda'\mathbf{I})^{-1}\mathbf{X}^{\top}\mathbf{y}$.

  - L_1-regularization (Lasso Regression)

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_1 \; . \tag{20}$$

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{18}$$

- Ω is a regularizer, and will measure the complexity of the model given by $\mathbf{w}$.
- **Examples:**
  - L_2-regularization (Ridge Regression)

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_2^2 \;, \tag{19}$$

    Exercise $\rightarrow$ the explicit solution is defined as $\mathbf{w}^\star_{\text{ridge}} = (\mathbf{X}^\top \mathbf{X} + \lambda' \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$.
  - L_1-regularization (Lasso Regression)

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_1 \;. \tag{20}$$

- Trade-off between under-fitting and over-fitting. In practice,

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{18}$$

- Ω is a regularizer, and will measure the complexity of the model given by $\mathbf{w}$.
- **Examples:**
  - L_2-regularization (Ridge Regression)

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_2^2 \; , \tag{19}$$

Exercise $\rightarrow$ the explicit solution is defined as $\mathbf{w}^\star_{\text{ridge}} = (\mathbf{X}^\top\mathbf{X} + \lambda'\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$.

  - L_1-regularization (Lasso Regression)

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_1 \; . \tag{20}$$

- Trade-off between under-fitting and over-fitting. In practice,
  - λ can control the model complexity.

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{18}$$

- Ω is a regularizer, and will measure the complexity of the model given by $\mathbf{w}$.
- **Examples:**
  - L_2-regularization (Ridge Regression)

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_2^2 \;, \tag{19}$$

    Exercise $\rightarrow$ the explicit solution is defined as $\mathbf{w}^{\star}_{\text{ridge}} = (\mathbf{X}^{\top}\mathbf{X} + \lambda'\mathbf{I})^{-1}\mathbf{X}^{\top}\mathbf{y}$.
  - L_1-regularization (Lasso Regression)

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_1 \;. \tag{20}$$

- Trade-off between under-fitting and over-fitting. In practice,
  - λ can control the model complexity.
  - The polynomial feature extension can enrich the complexity.

The probabilistic interpretation of Ridge Regression

We start with the posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{y})$ and chose $\mathbf{w}$ to maximize this posterior.

The Maximum-A-Posteriori (MAP) estimate:

$$\mathbf{w}_{\text{ridge}} = \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \qquad \text{(by the definition of posterior)}$$

$$(21)$$

The probabilistic interpretation of Ridge Regression

We start with the posterior $p(\mathbf{w}|\mathbf{X},\mathbf{y})$ and chose $\mathbf{w}$ to maximize this posterior.

The Maximum-A-Posteriori (MAP) estimate:

$$\begin{aligned}
\mathbf{w}_{\text{ridge}} &= \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{w}|\mathbf{X},\mathbf{y}) && \text{(by the definition of posterior)} \\
&\overset{(a)}{=} \arg\min_{\mathbf{w}} \quad -\log \frac{p(\mathbf{y},\mathbf{X}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y},\mathbf{X})} && \text{(by the Bayes' law)}
\end{aligned} \tag{21}$$

The probabilistic interpretation of Ridge Regression

We start with the posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{y})$ and chose $\mathbf{w}$ to maximize this posterior.

The Maximum-A-Posteriori (MAP) estimate:

$$
\begin{aligned}
\mathbf{w}_{\text{ridge}} &= \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{w}|\mathbf{X}, \mathbf{y}) && \text{(by the definition of posterior)} \\
&\overset{(a)}{=} \arg\min_{\mathbf{w}} \quad -\log \frac{p(\mathbf{y}, \mathbf{X}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}, \mathbf{X})} && \text{(by the Bayes' law)} \\
&\overset{(b)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}, \mathbf{X}|\mathbf{w})p(\mathbf{w}) && \text{(eliminate quantities that do not depend on } \mathbf{w}\text{)}
\end{aligned}
\tag{21}
$$

The probabilistic interpretation of Ridge Regression

We start with the posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{y})$ and chose $\mathbf{w}$ to maximize this posterior.

The Maximum-A-Posteriori (MAP) estimate:

$$
\begin{aligned}
\mathbf{w}_{\text{ridge}} &= \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{w}|\mathbf{X}, \mathbf{y}) && \text{(by the definition of posterior)} \\
&\overset{(a)}{=} \arg\min_{\mathbf{w}} \quad -\log \frac{p(\mathbf{y}, \mathbf{X}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}, \mathbf{X})} && \text{(by the Bayes' law)} \\
&\overset{(b)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}, \mathbf{X}|\mathbf{w})p(\mathbf{w}) && \text{(eliminate quantities that do not depend on } \mathbf{w}\text{)} \\
&\overset{(c)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w}) && \text{(eliminate quantities that do not depend on } \mathbf{w}\text{)}
\end{aligned}
\tag{21}
$$

The probabilistic interpretation of Ridge Regression

We start with the posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{y})$ and chose $\mathbf{w}$ to maximize this posterior.

The Maximum-A-Posteriori (MAP) estimate:

$$
\begin{aligned}
\mathbf{w}_{\text{ridge}} &= \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{w}|\mathbf{X}, \mathbf{y}) && \text{(by the definition of posterior)} \\
&\overset{(a)}{=} \arg\min_{\mathbf{w}} \quad -\log \frac{p(\mathbf{y}, \mathbf{X}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}, \mathbf{X})} && \text{(by the Bayes' law)} \\
&\overset{(b)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}, \mathbf{X}|\mathbf{w})p(\mathbf{w}) && \text{(eliminate quantities that do not depend on } \mathbf{w}\text{)} \\
&\overset{(c)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w}) && \text{(eliminate quantities that do not depend on } \mathbf{w}\text{)} \\
&= \arg\min_{\mathbf{w}} \quad \sum_{n=1}^{N} \frac{1}{2\sigma^2}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 . && (21)
\end{aligned}
$$

How do we trade-off the under-fitting and over-fitting?

Last lecture:

- Normal Equation and Least Squares
- Probabilistic Interpretation of Linear Regression
- Maximum Likelihood Estimation (MLE)
- Over-fitting and Under-fitting
- Polynomial Regression, Ridge Regression, and Lasso

Last lecture:

- Normal Equation and Least Squares
- Probabilistic Interpretation of Linear Regression
- Maximum Likelihood Estimation (MLE)
- Over-fitting and Under-fitting
- Polynomial Regression, Ridge Regression, and Lasso

This lecture:

- Generalization Gap and Model Selection
- Bias-Variance Decomposition

Table of Contents

Table of Contents

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \quad + \quad \lambda \, \|\mathbf{w}\|^2 \tag{22}$$

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \quad + \quad \boxed{\lambda}\, \|\mathbf{w}\|^2 \tag{22}$$

- λ can be tuned to control the model complexity

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \quad + \quad \boxed{\lambda}\, \|\mathbf{w}\|^2 \tag{22}$$

- λ can be tuned to control the model complexity
- In practice: $(\lambda_1, \ldots, \lambda_k) \longrightarrow$ Algorithm $\longrightarrow (\mathbf{w}_1, \ldots, \mathbf{w}_k)$.

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \quad + \quad \boxed{\lambda} \, \|\mathbf{w}\|^2 \tag{22}$$

- λ can be tuned to control the model complexity
- In practice: $(\lambda_1, \ldots, \lambda_k) \longrightarrow$ Algorithm $\longrightarrow (\mathbf{w}_1, \ldots, \mathbf{w}_k)$.
- Which λ should we use?

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \quad + \quad \boxed{\lambda}\,\|\mathbf{w}\|^2 \tag{22}$$

- λ can be tuned to control the model complexity
- In practice: $(\lambda_1, \ldots, \lambda_k) \longrightarrow$ Algorithm $\longrightarrow (\mathbf{w}_1, \ldots, \mathbf{w}_k)$.
- Which λ should we use?

Similarly, in Polynomial Regression: $(x_1, x_2) \xrightarrow{\phi}$ $(x_1, x_2, x_1^2 + x_2^2)$
$(x_1, x_2, 5x_1^2 + 2x_2^2, x_2^3 + 2x_1)$

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \quad + \quad \boxed{\lambda}\,\|\mathbf{w}\|^2 \tag{22}$$

- λ can be tuned to control the model complexity
- In practice: $(\lambda_1, \ldots, \lambda_k) \longrightarrow$ Algorithm $\longrightarrow (\mathbf{w}_1, \ldots, \mathbf{w}_k)$.
- Which λ should we use?

Similarly, in Polynomial Regression: $(x_1, x_2) \xrightarrow{\phi} \begin{matrix} (x_1, x_2, x_1^2 + x_2^2) \\ (x_1, x_2, 5x_1^2 + 2x_2^2, x_2^3 + 2x_1) \end{matrix}$

- we can enrich the model complexity, by augmenting the feature vector $\mathbf{x}$

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \quad + \quad \boxed{\lambda} \, \|\mathbf{w}\|^2 \tag{22}$$

- λ can be tuned to control the model complexity
- In practice: $(\lambda_1, \ldots, \lambda_k) \longrightarrow$ Algorithm $\longrightarrow (\mathbf{w}_1, \ldots, \mathbf{w}_k)$.
- Which λ should we use?

Similarly, in Polynomial Regression: $(x_1, x_2) \xrightarrow{\phi} \begin{matrix} (x_1, x_2, x_1^2 + x_2^2) \\ (x_1, x_2, 5x_1^2 + 2x_2^2, x_2^3 + 2x_1) \end{matrix}$

- we can enrich the model complexity, by augmenting the feature vector $\mathbf{x}$

How do we choose these hyper-parameters?

Model selection for neural networks

Optimization Algorithms?

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

Neural Architectures?

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

Regularizations?

Weight decay?
Dropout?
Early stopping?
Data augmentation?

To give a meaningful answer to the above questions,
we first need to specify our data model!

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{23}$$

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{23}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{24}$$

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{23}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{24}$$

- f_S denotes the output.

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{23}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{24}$$

- f_S denotes the output.
- $\mathcal{A}$ denotes the learning algorithm.

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{23}$$

Learning algorithm:

$$\boxed{f_S} = \boxed{\mathcal{A}}(\boxed{S}) \tag{24}$$

- f_S denotes the output.
- $\mathcal{A}$ denotes the learning algorithm.
- S denotes the input (dataset).

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{23}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{24}$$

- f_S denotes the output.
- $\mathcal{A}$ denotes the learning algorithm.
- S denotes the input (dataset).
- Can add a subscript $f_{S,\lambda}$ to indicate the model dependency (for Ridge Regression).

Given a model f, how can we assess if f is any good?

Given a model f, how can we assess if f is any good?

- We should compute the **expected error** over all samples chosen according to $\mathcal{D}$:

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] \qquad \text{where } \ell(\cdot,\cdot) \text{ is our loss function} \tag{25}$$

Given a model f, how can we assess if f is any good?

- We should compute the **expected error** over all samples chosen according to $\mathcal{D}$:

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] \qquad \text{where } \ell(\cdot,\cdot) \text{ is our loss function} \tag{25}$$

- The quantity $\mathcal{L}_{\mathcal{D}}(f)$ has many names $\left\{ \begin{array}{l} \textit{True} \\ \textit{Expected} \end{array} \right. \left\{ \begin{array}{l} \textit{Risk} \\ \textit{Error} \\ \textit{Loss} \end{array} \right.$

Given a model f, how can we assess if f is any good?

- We should compute the **expected error** over all samples chosen according to $\mathcal{D}$:

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] \qquad \text{where } \ell(\cdot,\cdot) \text{ is our loss function} \tag{25}$$

- The quantity $\mathcal{L}_{\mathcal{D}}(f)$ has many names $\begin{cases} \textit{True} \\ \textit{Expected} \end{cases} \begin{cases} \textit{Risk} \\ \textit{Error} \\ \textit{Loss} \end{cases}$

This is the quantity we are fundamentally interested in

Given a model f, how can we assess if f is any good?

- We should compute the **expected error** over all samples chosen according to $\mathcal{D}$:

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] \qquad \text{where } \ell(\cdot,\cdot) \text{ is our loss function} \tag{25}$$

- The quantity $\mathcal{L}_{\mathcal{D}}(f)$ has many names $\left\{\begin{array}{l} \textit{True} \\ \textit{Expected} \end{array}\right. \quad \left\{\begin{array}{l} \textit{Risk} \\ \textit{Error} \\ \textit{Loss} \end{array}\right.$

This is the quantity we are fundamentally interested in
but we cannot estimate it since $\mathcal{D}$ is not known.

Empirical Error: what we can compute

- We are given some data S

Empirical Error: what we can compute

- We are given some data S
- We can approximate the true error by **averaging the loss function on the dataset** S

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) \,. \tag{26}$$

Empirical Error: what we can compute

- We are given some data S
- We can approximate the true error by **averaging the loss function on the dataset** S

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) \,. \tag{26}$$

- This is called *empirical risk/error/loss*.

Empirical Error: what we can compute

- We are given some data S
- We can approximate the true error by **averaging the loss function on the dataset** S

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) \,. \tag{26}$$

- This is called *empirical risk/error/loss*.
- The samples are random thus $L_S(f)$ is a random variable.

Empirical Error: what we can compute

- We are given some data S
- We can approximate the true error by **averaging the loss function on the dataset** S

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) \,. \tag{26}$$

- This is 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.

Empirical Error: what we can compute

- We are given some data S
- We can approximate the true error by **averaging the loss function on the dataset** S

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) \,. \tag{26}$$

- This is 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.
- **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.

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

- The empirical error is called the **training error**:

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

- The empirical error is called the **training error**: when the model has been trained on the same data it is applied to, namely,

Training Error: what we are minimizing

> The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

- The empirical error is called the **training error**: when the model has been trained on the same data it is applied to, namely,
  - If we first learn the model from S, i.e., we compute $f_S = \mathcal{A}(S)$,

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

- The empirical error is called the **training error**: when the model has been trained on the same data it is applied to, namely,
    - If we first learn the model from S, i.e., we compute $f_S = \mathcal{A}(S)$,
    - and then we compute the empirical risk of f_S using the same data S.

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

- The empirical error is called the **training error**: when the model has been trained on the same data it is applied to, namely,
  - If we first learn the model from S, i.e., we compute $f_S = \mathcal{A}(S)$,
  - and then we compute the empirical risk of f_S using the same data S.
  - Then in fact we are computing

$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f_S(\mathbf{x}_n)) \,. \tag{27}$$

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

- The empirical error is called the **training error**: when the model has been trained on the same data it is applied to, namely,
  - If we first learn the model from S, i.e., we compute $f_S = \mathcal{A}(S)$,
  - and then we compute the empirical risk of f_S using the same data S.
  - Then in fact we are computing

$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f_S(\mathbf{x}_n)) \,. \tag{27}$$

  - This is called *the training (risk/loss/error)*.

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

- The empirical error is called the **training error**: when the model has been trained on the same data it is applied to, namely,
  - If we first learn the model from S, i.e., we compute $f_S = \mathcal{A}(S)$,
  - and then we compute the empirical risk of f_S using the same data S.
  - Then in fact we are computing

$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f_S(\mathbf{x}_n)) \,. \tag{27}$$

  - This is called *the training (risk/loss/error)*.

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

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

- **Problem:** Validating model on the same data subset we trained it on!

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

- **Problem:** Validating model on the same data subset we trained it on!
- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{28}$$

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

- **Problem:** Validating model on the same data subset we trained it on!
- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{28}$$

1. We **learn** the function $f_{S_{\text{train}}}$ using the train set S_{train}

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

- **Problem:** Validating model on the same data subset we trained it on!
- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{28}$$

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

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n, \mathbf{x}_n) \in S_{\text{test}}} \ell\left(y_n, f_{S_{\text{train}}}(\mathbf{x}_n)\right). \tag{29}$$

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

- **Problem:** Validating model on the same data subset we trained it on!
- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{28}$$

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

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n, \mathbf{x}_n) \in S_{\text{test}}} \ell\left(y_n, f_{S_{\text{train}}}(\mathbf{x}_n)\right). \tag{29}$$

- Since S_{train} and S_{test} are independent, we hope that $L_{S_{\text{test}}}(f_{S_{\text{train}}}) \approx L_{\mathcal{D}}(f_{S_{\text{train}}})$

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

- **Problem:** Validating model on the same data subset we trained it on!
- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{28}$$

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

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n, \mathbf{x}_n) \in S_{\text{test}}} \ell\left(y_n, f_{S_{\text{train}}}(\mathbf{x}_n)\right). \tag{29}$$

- Since S_{train} and S_{test} are independent, we hope that $L_{S_{\text{test}}}(f_{S_{\text{train}}}) \approx L_{\mathcal{D}}(f_{S_{\text{train}}})$
- Issues: we have fewer data both for the learning and validation tasks (trade-off)

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

Assume that we have a model f, and that our loss function $\ell(\cdot, \cdot)$ is bounded.

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

Assume that we have a model f, and that our loss function $\ell(\cdot, \cdot)$ is bounded.

- We are given a test set S_{test} chosen i.i.d. from the underlying distribution $\mathcal{D}$ (and this test set was *not* used to train the model).

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

Assume that we have a model f, and that our loss function $\ell(\cdot, \cdot)$ is bounded.

- We are given a test set S_{test} chosen i.i.d. from the underlying distribution $\mathcal{D}$ (and this test set was *not* used to train the model).
- The test/empirical error is

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n) \in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{30}$$

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

Assume that we have a model f, and that our loss function $\ell(\cdot,\cdot)$ is bounded.

- We are given a test set S_{test} chosen i.i.d. from the underlying distribution $\mathcal{D}$ (and this test set was *not* used to train the model).
- The test/empirical error is

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n) \in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{30}$$

- The true error is

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right].$$

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

Assume that we have a model f, and that our loss function $\ell(\cdot,\cdot)$ is bounded.

- We are given a test set S_{test} chosen i.i.d. from the underlying distribution $\mathcal{D}$ (and this test set was *not* used to train the model).
- The test/empirical error is

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n) \in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{30}$$

- The true error is

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right].$$

- How far are these apart?

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

Assume that we have a model f, and that our loss function $\ell(\cdot, \cdot)$ is bounded.

- We are given a test set S_{test} chosen i.i.d. from the underlying distribution $\mathcal{D}$ (and this test set was *not* used to train the model).
- The test/empirical error is

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n) \in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{30}$$

- The true error is

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x}) \sim \mathcal{D}} \left[\ell\left(y, f(\mathbf{x})\right)\right].$$

- How far are these apart?

Definition 3 (Generalization error)

The *generalization error* is given by

$$\left|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)\right|.$$

Table of Contents

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

- **True Error:**

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f\left(\mathbf{x}\right)\right)\right]. \tag{31}$$

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

- **True Error:**

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right]. \tag{31}$$

- **Test/Empirical Error:**

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n)\in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{32}$$

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

- **True Error:**

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right]. \tag{31}$$

- **Test/Empirical Error:**

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n)\in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{32}$$

- **Generalization Error:**

$$\left|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)\right|. \tag{33}$$

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

- **True Error:**

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right]. \tag{31}$$

- **Test/Empirical Error:**

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n)\in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{32}$$

- **Generalization Error:**

$$\left|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)\right|. \tag{33}$$

In expectation, they are the same:

$$L_{\mathcal{D}}(f) = \mathbb{E}_{S_{\text{test}}\sim\mathcal{D}}\left[L_{S_{\text{test}}}(f)\right], \tag{34}$$

where the expectation is over the samples of the test set.

The variation of the $|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)|$ matters.

The variation of the $|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)|$ matters.

Theorem 4

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

$$\Pr\left[|L_{\mathcal{D}}(f) - L_{S_{test}}(f)| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{35}$$

The variation of the $|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)|$ matters.

Theorem 4

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

$$\Pr\left[|L_{\mathcal{D}}(f) - L_{S_{test}}(f)| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{35}$$

- Exercise:

The variation of the $|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)|$ matters.

Theorem 4

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

$$\Pr\left[|L_{\mathcal{D}}(f) - L_{S_{test}}(f)| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{test}|}} \right] \leq \delta \tag{35}$$

- Exercise: The error decreases as $\mathcal{O}\left(1/\sqrt{S_{\text{test}}}\right)$ with the number test points.

The variation of the $|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)|$ matters.

Theorem 4

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

$$\Pr\left[|L_{\mathcal{D}}(f) - L_{S_{test}}(f)| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{35}$$

- Exercise: The error decreases as $\mathcal{O}\left(1/\sqrt{S_{\text{test}}}\right)$ with the number test points.

$\Rightarrow$ 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?

Why do you care?

Given a predictor f and a dataset S, we can control the expected risk:

$$\Pr\left[\underbrace{L_{\mathcal{D}}(f)}_{\text{not computable}} \geq \underbrace{L_{S_{\text{test}}}(f)}_{\text{computable}} + \underbrace{\sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{\text{test}}|}}}_{\text{deviation}}\right] \leq \delta\,. \tag{36}$$

Why do you care?

Given a predictor f and a dataset S, we can control the expected risk:

$$\Pr\left[\underbrace{L_{\mathcal{D}}(f)}_{\text{not computable}} \geq \underbrace{L_{S_{\text{test}}}(f)}_{\text{computable}} + \underbrace{\sqrt{\frac{(b-a)^2\ln(2/\delta)}{2\left|S_{\text{test}}\right|}}}_{\text{deviation}}\right] \leq \delta\,. \tag{36}$$

In words, we can compute a probabilistic upper bound on the true risk.

Why do you care?

Given a predictor f and a dataset S, we can control the expected risk:

$$\Pr\left[\underbrace{L_{\mathcal{D}}(f)}_{\text{not computable}} \geq \underbrace{L_{S_{\text{test}}}(f)}_{\text{computable}} + \underbrace{\sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\left|S_{\text{test}}\right|}}}_{\text{deviation}}\right] \leq \delta\,. \tag{36}$$

In words, we can compute a probabilistic upper bound on the true risk.

A concrete example: given a dataset S

Why do you care?

Given a predictor f and a dataset S, we can control the expected risk:

$$\Pr\left[\underbrace{L_{\mathcal{D}}(f)}_{\text{not computable}} \geq \underbrace{L_{S_{\text{test}}}(f)}_{\text{computable}} + \underbrace{\sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\left|S_{\text{test}}\right|}}}_{\text{deviation}}\right] \leq \delta\,. \tag{36}$$

In words, we can compute a probabilistic upper bound on the true risk.

A concrete example: given a dataset S

- Split: $S = S_{\text{train}} \cup S_{\text{test}}$

Why do you care?

Given a predictor f and a dataset S, we can control the expected risk:

$$\Pr\left[\underbrace{L_{\mathcal{D}}(f)}_{\text{not computable}} \geq \underbrace{L_{S_{\text{test}}}(f)}_{\text{computable}} + \underbrace{\sqrt{\frac{(b-a)^2\ln(2/\delta)}{2\left|S_{\text{test}}\right|}}}_{\text{deviation}}\right] \leq \delta\,. \tag{36}$$

In words, we can compute a probabilistic upper bound on the true risk.

A concrete example: given a dataset S

- Split: $S = S_{\text{train}} \cup S_{\text{test}}$
- Train: $\mathcal{A}(S_{\text{train}}) = f_{S_{\text{train}}}$

Why do you care?

Given a predictor f and a dataset S, we can control the expected risk:

$$\Pr\left[\underbrace{L_{\mathcal{D}}(f)}_{\text{not computable}} \geq \underbrace{L_{S_{\text{test}}}(f)}_{\text{computable}} + \underbrace{\sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{\text{test}}|}}}_{\text{deviation}}\right] \leq \delta\,. \quad (36)$$

In words, we can compute a probabilistic upper bound on the true risk.

A concrete example: given a dataset S

- Split: $S = S_{\text{train}} \cup S_{\text{test}}$
- Train: $\mathcal{A}(S_{\text{train}}) = f_{S_{\text{train}}}$
- Validate:

$$\Pr\left[L_{\mathcal{D}}(f_{S_{\text{train}}}) \geq L_{S_{\text{test}}}(f_{S_{\text{train}}}) + \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{\text{test}}|}}\right] \leq \delta\,. \quad (37)$$

Table of Contents

Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

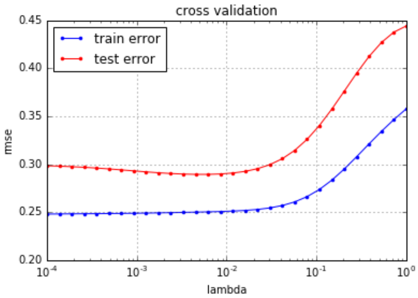
Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

Challenges: We have a set of values $\{\lambda_k\}_{k=1}^{K}$, which one to choose?

Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

Challenges: We have a set of values $\{\lambda_k\}_{k=1}^{K}$, which one to choose?

Solutions:

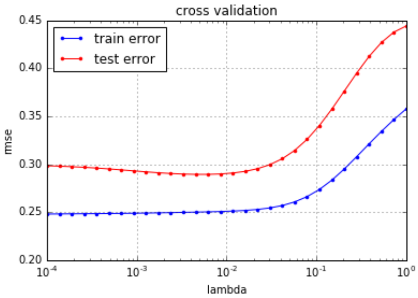


Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

Challenges: We have a set of values $\{\lambda_k\}_{k=1}^{K}$, which one to choose?

Solutions:

- Split the data into $S = S_{\text{train}} \cup S_{\text{test}}$.

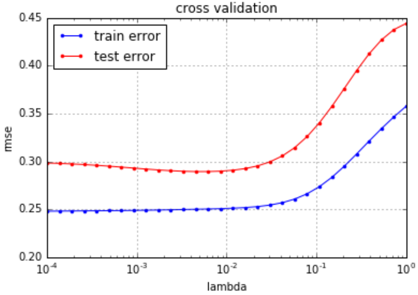


Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

Challenges: We have a set of values $\{\lambda_k\}_{k=1}^K$, which one to choose?

Solutions:

- Split the data into $S = S_{\text{train}} \cup S_{\text{test}}$.
- Run the learning algorithm K times on the same training set S_{train} to compute the K prediction functions $f_{S_{\text{train}},\lambda_k}$.

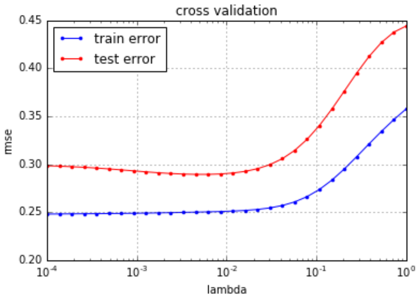


Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

Challenges: We have a set of values $\{\lambda_k\}_{k=1}^{K}$, which one to choose?

Solutions:

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

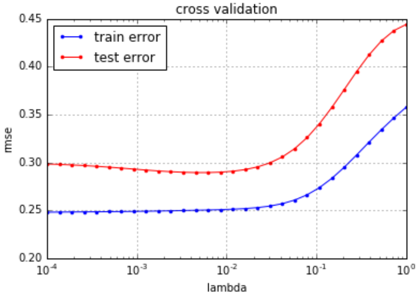


Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

Challenges: We have a set of values $\{\lambda_k\}_{k=1}^K$, which one to choose?

Solutions:

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



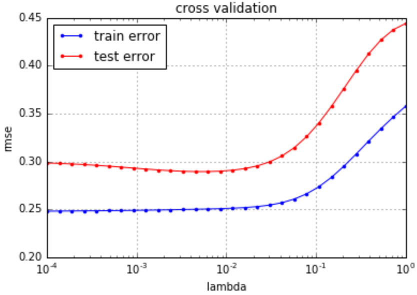
We then choose the value of the parameter λ which gives us the smallest such test error.

Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

Challenges: We have a set of values $\{\lambda_k\}_{k=1}^K$, which one to choose?

Solutions:

- Split the data into $S = S_{\text{train}} \cup S_{\text{test}}$.
- Run the learning algorithm K times on the same training set S_{train} to compute the K prediction functions $f_{S_{\text{train}},\lambda_k}$.
- 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 which gives us the smallest such test error.

Issues: the existence of the generalization gap $|L_{S_{\text{test}}}(f_{S_{\text{train}},\lambda_k}) - L_{\mathcal{D}}(f_{S_{\text{train}},\lambda_k})|$!

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

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

Theorem 5

We can bound the maximum deviation for all K candidates, by

$$\Pr\left[\max_k |L_{\mathcal{D}}(f_k) - L_{S_{test}}(f_k)| \geq \sqrt{\frac{(b-a)^2 \ln(2\boxed{K}/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{38}$$

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

Theorem 5

We can bound the maximum deviation for all K candidates, by

$$\Pr\left[\max_k |L_{\mathcal{D}}(f_k) - L_{S_{test}}(f_k)| \geq \sqrt{\frac{(b-a)^2 \ln(2\boxed{K}/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{38}$$

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

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

Theorem 5

We can bound the maximum deviation for all K candidates, by

$$\Pr\left[\max_k \left|L_{\mathcal{D}}(f_k) - L_{S_{test}}(f_k)\right| \geq \sqrt{\frac{(b-a)^2 \ln(2\boxed{K}/\delta)}{2\left|S_{test}\right|}}\right] \leq \delta \tag{38}$$

- 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 $\sqrt{\ln(K)}$.

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

Theorem 5

We can bound the maximum deviation for all K candidates, by

$$\Pr\left[\max_k |L_{\mathcal{D}}(f_k) - L_{S_{test}}(f_k)| \geq \sqrt{\frac{(b-a)^2 \ln(2\boxed{K}/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{38}$$

- 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 $\sqrt{\ln(K)}$.

$\Rightarrow$ So we can test many different models without incurring a large penalty.

- $k^\star = \arg\min_k L_{\mathcal{D}}(f_k)$, i.e., $f_{k^\star}$ denotes the function with the smallest true risk.
- $\hat{k} = \arg\min_k L_{S_{\text{test}}}(f_k)$, i.e., $f_{\hat{k}}$ denotes the function with the smallest empirical risk.

$$\Pr\left[L_{\mathcal{D}}(f_{\hat{k}}) \geq L_{\mathcal{D}}(f_{k^\star}) + \sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2\left|S_{\text{test}}\right|}}\right] \leq \delta \tag{39}$$

- $k^\star = \arg\min_k L_{\mathcal{D}}(f_k)$, i.e., $f_{k^\star}$ denotes the function with the smallest true risk.
- $\hat{k} = \arg\min_k L_{S_{\text{test}}}(f_k)$, i.e., $f_{\hat{k}}$ denotes the function with the smallest empirical risk.

$$\Pr\left[L_{\mathcal{D}}(f_{\hat{k}}) \geq L_{\mathcal{D}}(f_{k^\star}) + \sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2\left|S_{\text{test}}\right|}}\right] \leq \delta \tag{39}$$

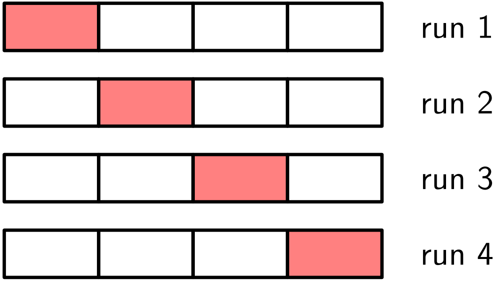
If we choose the “best” function according to the empirical risk,
then its true risk is not too far away from the true risk of the optimal choice.

Table of Contents

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

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

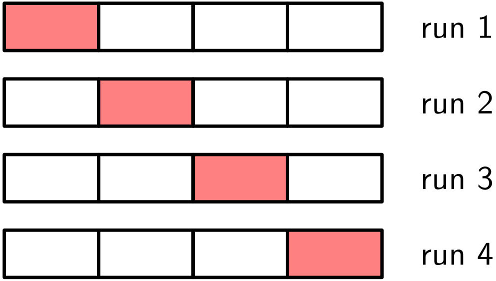
K-fold cross-validation:



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

K-fold cross-validation:

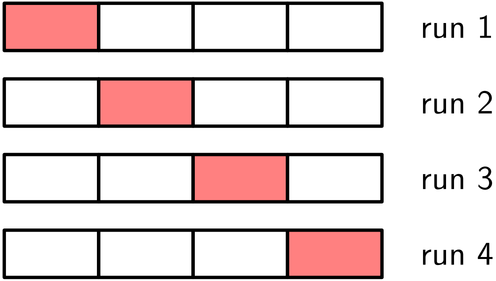
1. Randomly partition the data into K groups



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

K-fold cross-validation:

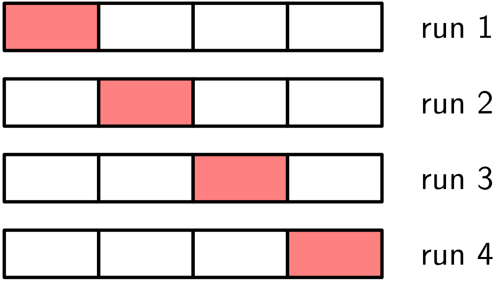
1. Randomly partition the data into K groups
2. Train K times.



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

K-fold cross-validation:

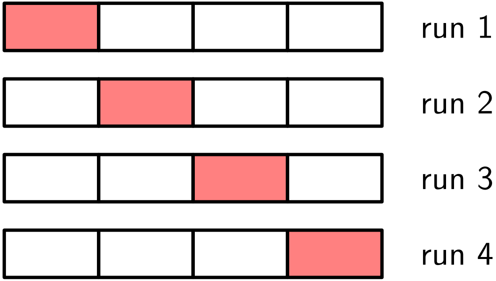
1. Randomly partition the data into K groups
2. Train K times. Each time leave out exactly one of the K groups for testing



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

K-fold cross-validation:

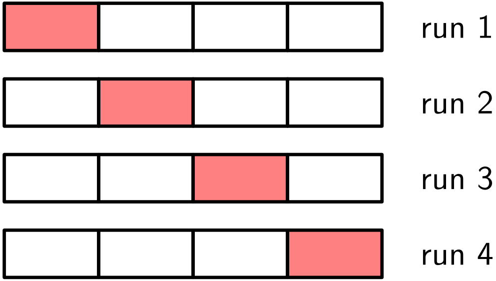
1. Randomly partition the data into K groups
2. Train K times. Each time leave out exactly one of the K groups for testing and use the remaining $K-1$ groups for training.



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

K-fold cross-validation:

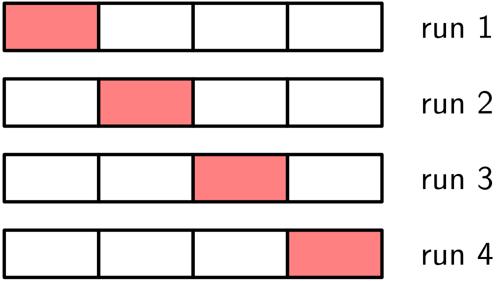
1. Randomly partition the data into K groups
2. Train K times. Each time leave out exactly one of the K groups for testing and use the remaining $K-1$ groups for training.
3. Average the K results



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

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

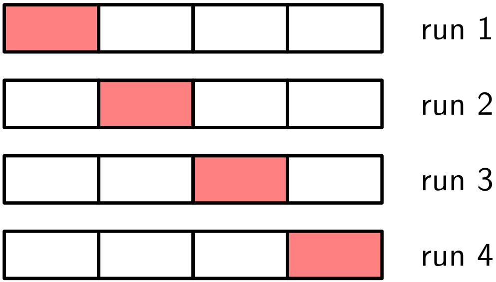


Benefits:

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

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



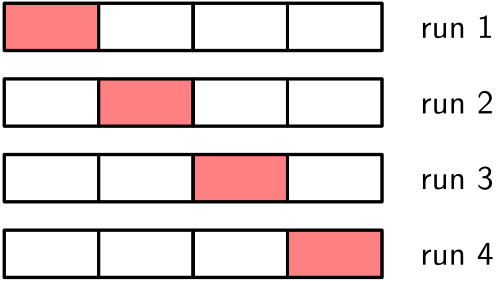
Benefits:

- We have used all data for training, and all data for testing, and used each data point the same number of times.

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

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



Benefits:

- 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 unbiased estimate of the generalization error and its variance.

Table of Contents

Previously:

Previously:

- **Motivation:** Hyperparameters search (which often control the complexity)

Previously:

- **Motivation:** Hyperparameters search (which often control the complexity)
- **Questions:**

Previously:

- **Motivation:** Hyperparameters search (which often control the complexity)
- **Questions:**
  - How can we judge if a given predictor is good?

Previously:

- **Motivation:** Hyperparameters search (which often control the complexity)
- **Questions:**
  - How can we judge if a given predictor is good?
  - How to select the best models of a family?

Previously:

- **Motivation:** Hyperparameters search (which often control the complexity)
- **Questions:**
  - How can we judge if a given predictor is good?
  - How to select the best models of a family?
- **Solutions:**

Previously:

- **Motivation:** Hyperparameters search (which often control the complexity)
- **Questions:**
  - How can we judge if a given predictor is good?
  - How to select the best models of a family?
- **Solutions:**

  $\Rightarrow$ Bound the difference between the true and empirical risks.

Previously:

- **Motivation:** Hyperparameters search (which often control the complexity)
- **Questions:**
  - How can we judge if a given predictor is good?
  - How to select the best models of a family?
- **Solutions:**

$\Rightarrow$ Bound the difference between the true and empirical risks.
$\Rightarrow$ Split data into train and test sets (learn with the train and test on the test).

Previously:

- **Motivation:** Hyperparameters search (which often control the complexity)
- **Questions:**
  - How can we judge if a given predictor is good?
  - How to select the best models of a family?
- **Solutions:**

$\Rightarrow$ Bound the difference between the true and empirical risks.
$\Rightarrow$ Split data into train and test sets (learn with the train and test on the test).

This part:

Previously:

- **Motivation:** Hyperparameters search (which often control the complexity)
- **Questions:**
  - How can we judge if a given predictor is good?
  - How to select the best models of a family?
- **Solutions:**

  $\Rightarrow$ Bound the difference between the true and empirical risks.

  $\Rightarrow$ Split data into train and test sets (learn with the train and test on the test).

This part:

- the role of the complexity of the class

Previously:

- **Motivation:** Hyperparameters search (which often control the complexity)
- **Questions:**
  - How can we judge if a given predictor is good?
  - How to select the best models of a family?
- **Solutions:**

  $\Rightarrow$ Bound the difference between the true and empirical risks.
  $\Rightarrow$ Split data into train and test sets (learn with the train and test on the test).

This part:

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

Previously:

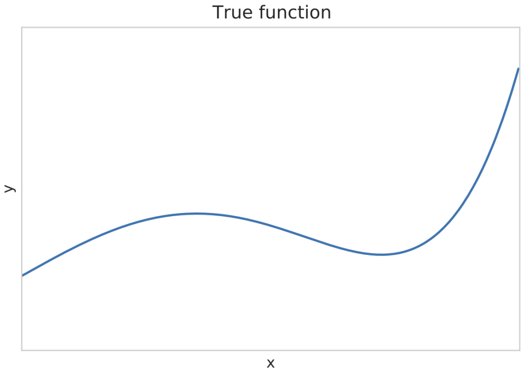
- **Motivation:** Hyperparameters search (which often control the complexity)
- **Questions:**
  - How can we judge if a given predictor is good?
  - How to select the best models of a family?
- **Solutions:**

$\Rightarrow$ Bound the difference between the true and empirical risks.
$\Rightarrow$ Split data into train and test sets (learn with the train and test on the test).

This part:

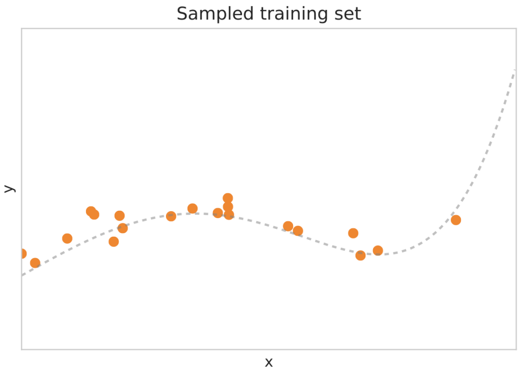
- the role of the complexity of the class
- How does the risk behave as a function of the complexity of the model class?
- It will help us to decide how complex and rich we should make our model

Motivation example: 1D-regression



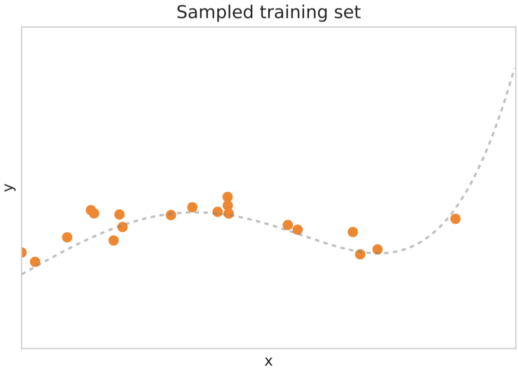
We have a true underlying function in blue we would like to recover.

Motivation example: 1D-regression



We have a true underlying function in blue we would like to recover.

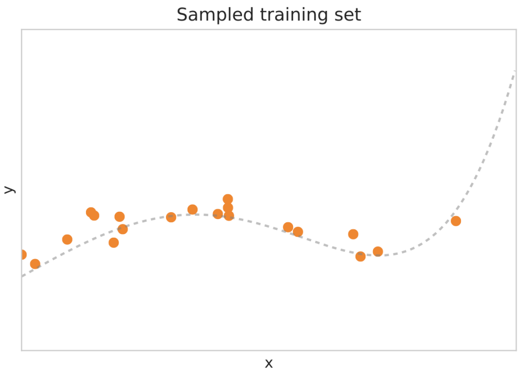
Motivation example: 1D-regression



We have a true underlying function in blue we would like to recover.

- Considering Linear Regression with polynomial feature expansion $x, x^2, x^3, \ldots, x^d$.

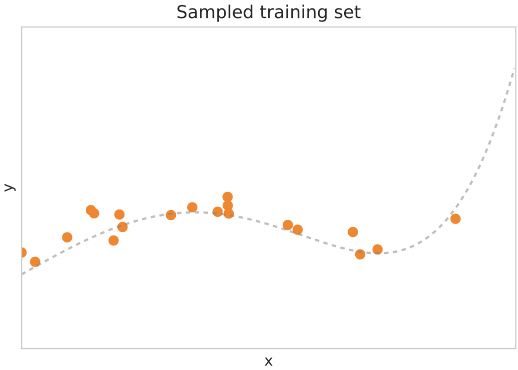
Motivation example: 1D-regression



We have a true underlying function in blue we would like to recover.

- Considering Linear Regression with polynomial feature expansion $x, x^2, x^3, \ldots, x^d$.
- The maximum degree d measures the complexity of the class

Motivation example: 1D-regression

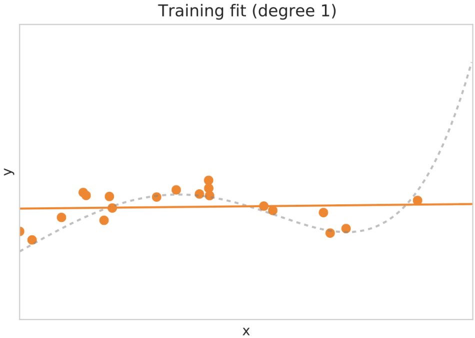


We have a true underlying function in blue we would like to recover.

- Considering Linear Regression with polynomial feature expansion $x, x^2, x^3, \ldots, x^d$.
- The maximum degree d measures the complexity of the class
  - $\Rightarrow$ How far should we go?

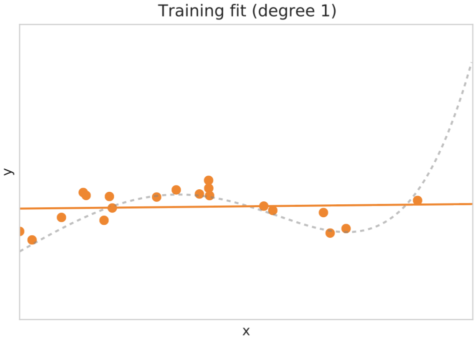
Motivation example: 1D-regression

If we restrict ourselves to simple models:



Motivation example: 1D-regression

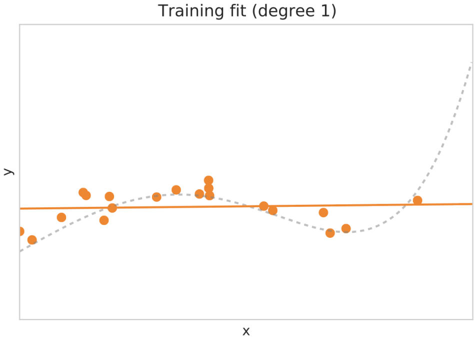
If we restrict ourselves to simple models:



- No linear function would be a good predictor.

Motivation example: 1D-regression

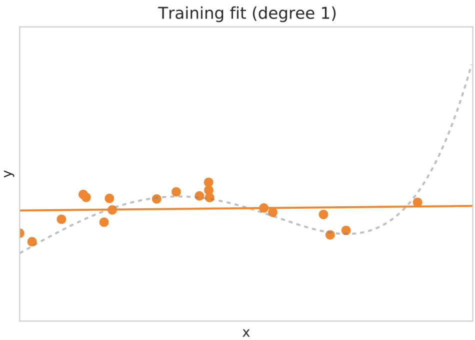
If we restrict ourselves to simple models:



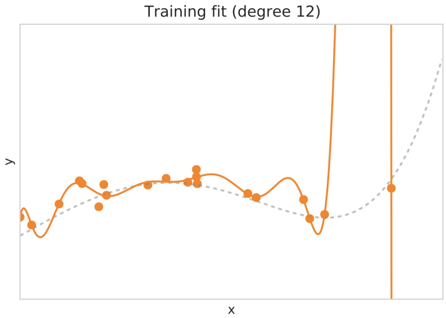
- No linear function would be a good predictor.
- The model is not rich/expressive enough.

Motivation example: 1D-regression

If we restrict ourselves to simple models:



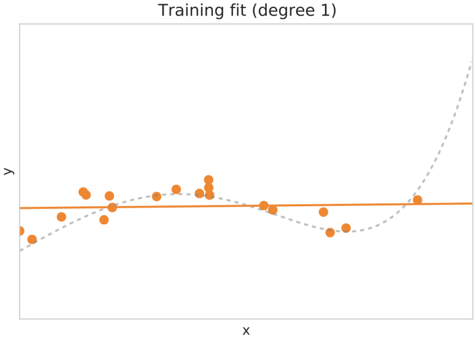
If we consider a high-degree polynomial:



- No linear function would be a good predictor.
- The model is not rich/expressive enough.

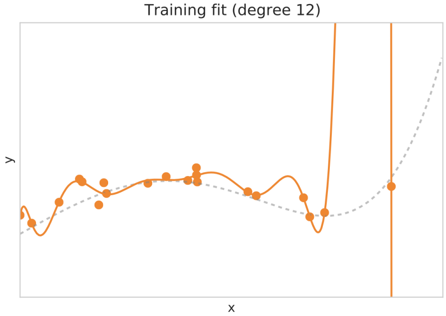
Motivation example: 1D-regression

If we restrict ourselves to simple models:



- No linear function would be a good predictor.
- The model is not rich/expressive enough.

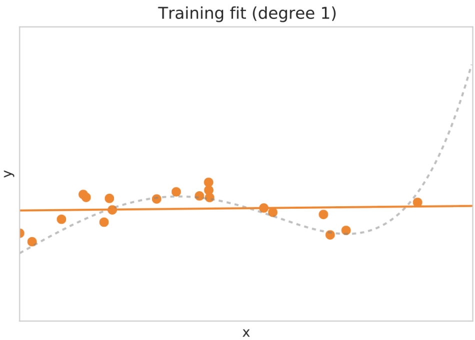
If we consider a high-degree polynomial:



- We can find a model that fits the data well

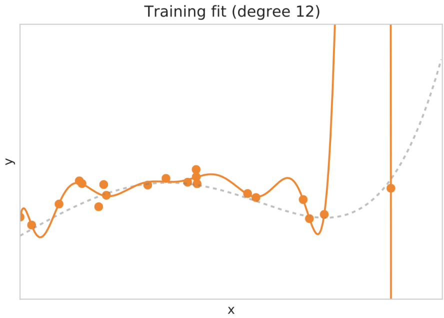
Motivation example: 1D-regression

If we restrict ourselves to simple models:



- No linear function would be a good predictor.
- The model is not rich/expressive enough.

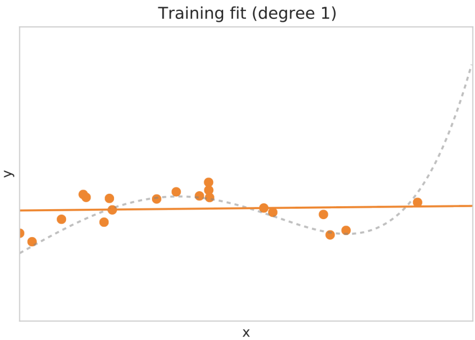
If we consider a high-degree polynomial:



- We can find a model that fits the data well
- Is it a good predictor?

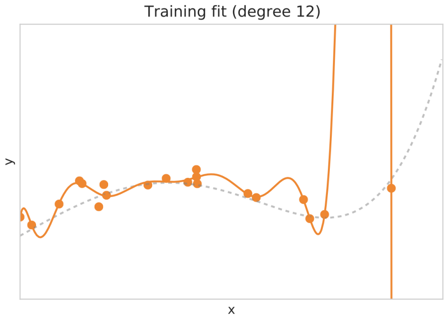
Motivation example: 1D-regression

If we restrict ourselves to simple models:



- No linear function would be a good predictor.
- The model is not rich/expressive enough.

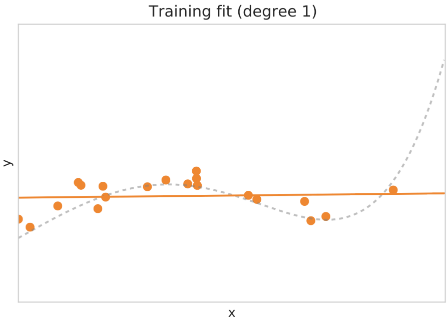
If we consider a high-degree polynomial:

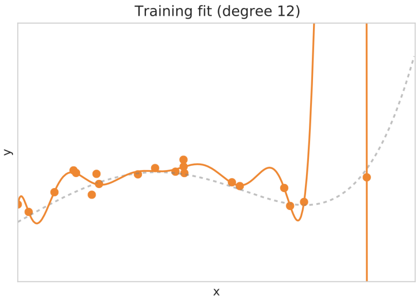


- We can find a model that fits the data well
- Is it a good predictor?
- No: Large generalization error

Motivation example: 1D-regression

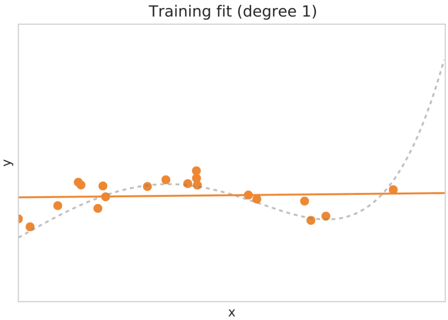
Simple models are less sensitive.

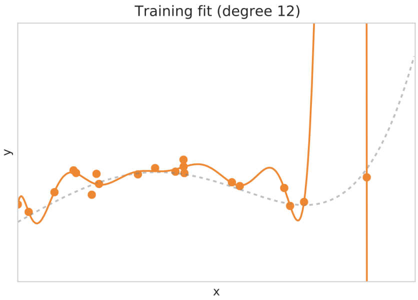




Motivation example: 1D-regression

Simple models are less sensitive.

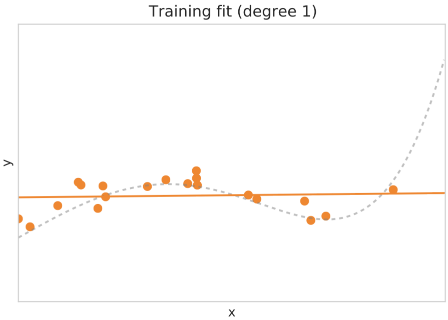


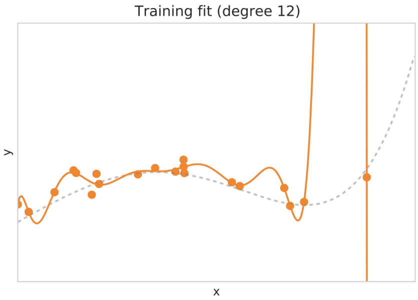


- moving a single observation will cause only a small shift in the position of the line

Motivation example: 1D-regression

Simple models are less sensitive.

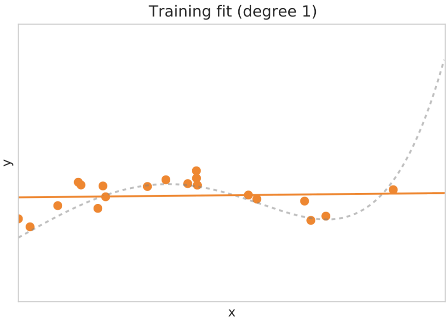


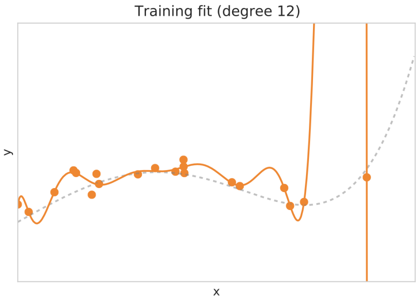


- moving a single observation will cause only a small shift in the position of the line
- under-fitting

Motivation example: 1D-regression

Simple models are less sensitive.



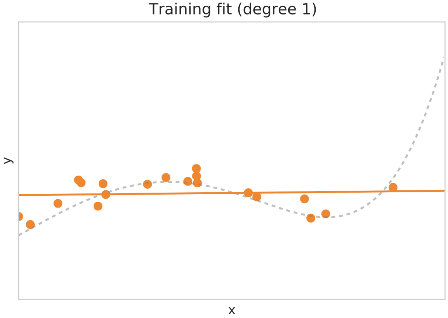


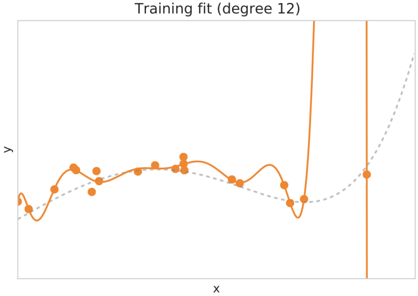
- moving a single observation will cause only a small shift in the position of the line
- under-fitting

- Changing one of the data points may change the prediction considerable

Motivation example: 1D-regression

Simple models are less sensitive.





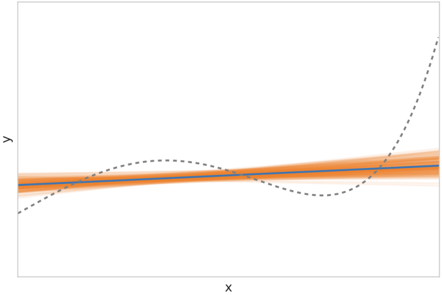
- moving a single observation will cause only a small shift in the position of the line
- under-fitting

- Changing one of the data points may change the prediction considerable
- over-fitting

Motivation example: 1D-regression

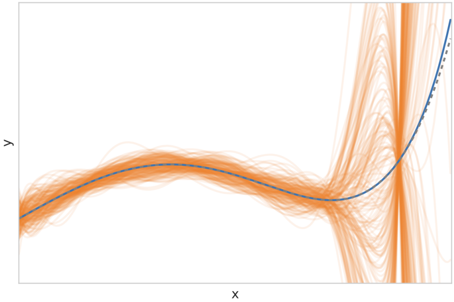
Simple models have large biases but low variance.

Learned functions (degree 1)



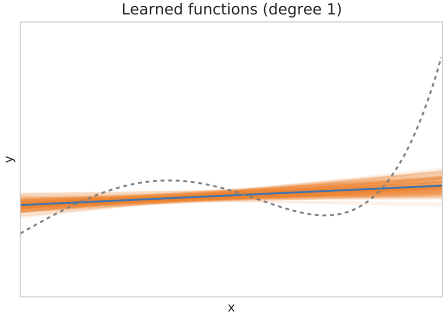
Complex models have low bias but high variance.

Learned functions (degree 9)



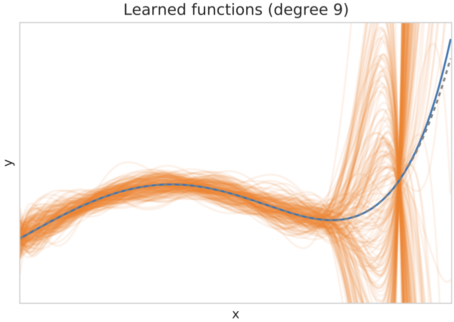
Motivation example: 1D-regression

Simple models have large biases but low variance.



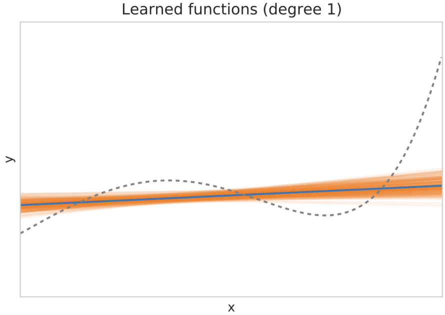
- large bias: the average of the predictions f_S does not fit well the data

Complex models have low bias but high variance.



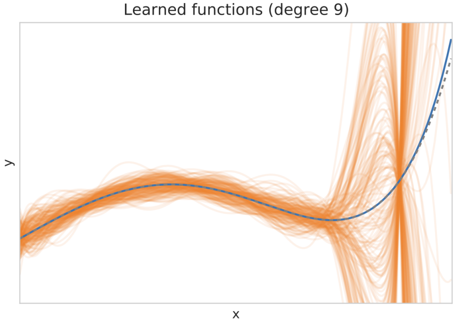
Motivation example: 1D-regression

Simple models have large biases but low variance.



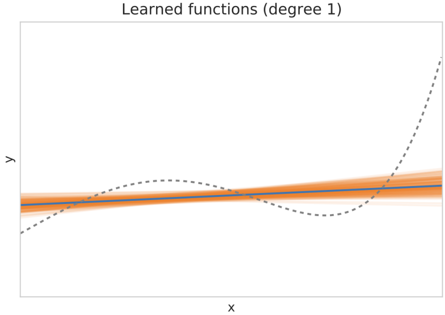
- large bias: the average of the predictions f_S does not fit well the data
- small variance: the variance of the predictions f_S as a function of S is small

Complex models have low bias but high variance.



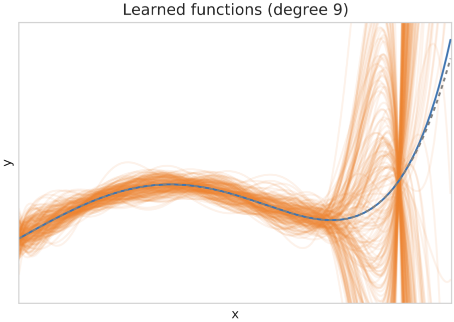
Motivation example: 1D-regression

Simple models have large biases but low variance.



- large bias: the average of the predictions f_S does not fit well the data
- small variance: the variance of the predictions f_S as a function of S is small

Complex models have low bias but high variance.

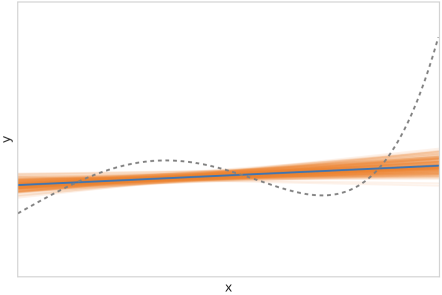


- small bias

Motivation example: 1D-regression

Simple models have large biases but low variance.

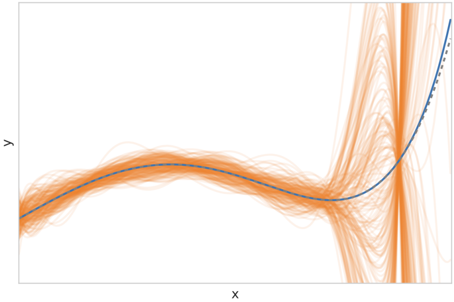
Learned functions (degree 1)



- large bias: the average of the predictions f_S does not fit well the data
- small variance: the variance of the predictions f_S as a function of S is small

Complex models have low bias but high variance.

Learned functions (degree 9)



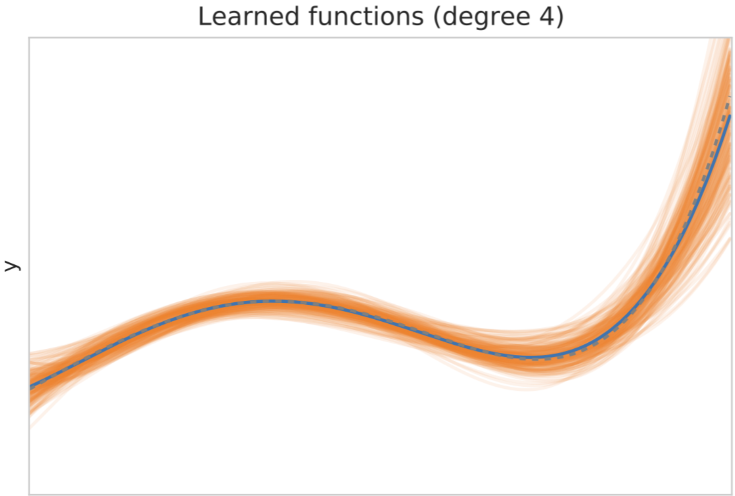
- small bias
- large variance

Motivation example: 1D-regression

We need to balance bias & variance correctly

Motivation example: 1D-regression

We need to balance bias & variance correctly



Data model

We assume that the data forms a joint distribution $(\mathbf{x}, y) \sim \mathcal{D}$, and is generated as

$$y = f(x) + \epsilon \tag{40}$$

Data model

We assume that the data forms a joint distribution $(\mathbf{x}, y) \sim \mathcal{D}$, and is generated as

- **True model:** f is an arbitrary and unknown function.

$$\boxed{y} = \boxed{f}(\boxed{x}) + \boxed{\epsilon} \quad (40)$$

Data model

We assume that the data forms a joint distribution $(\mathbf{x}, y) \sim \mathcal{D}$, and is generated as

- **True model:** f is an arbitrary and unknown function.
- **Output:** true model y is perturbed by some noise.

$$y = f(x) + \epsilon \tag{40}$$

Data model

We assume that the data forms a joint distribution $(\mathbf{x}, y) \sim \mathcal{D}$, and is generated as

- **True model:** f is an arbitrary and unknown function.
- **Output:** true model y is perturbed by some noise.

$$y = f(x) + \epsilon \tag{40}$$

- **Input:** $\mathbf{x} \sim \mathcal{D}$ (fixed but unknown)

Data model

We assume that the data forms a joint distribution $(\mathbf{x}, y) \sim \mathcal{D}$, and is generated as

- **True model:** f is an arbitrary and unknown function.
- **Output:** true model y is perturbed by some noise.

$$\boxed{y} = \boxed{f}(\boxed{x}) + \boxed{\epsilon} \tag{40}$$

- **Input:** $\mathbf{x} \sim \mathcal{D}$ (fixed but unknown)
- **Noise:** $\epsilon \in \mathcal{D}_\epsilon$ i.i.d., independent of $\mathbf{x}$ and $\mathbb{E}[\epsilon] = 0$

Error decomposition

Error decomposition

Training data
$S_{\text{train}} \sim \mathcal{D}$ i.i.d.

Error decomposition

$$\boxed{\begin{array}{c}\text{Training data}\\ S_{\text{train}} \sim \mathcal{D} \text{ i.i.d.}\end{array}} \longrightarrow \boxed{\begin{array}{c}\text{Algorithm}\\ \mathcal{A}\end{array}}$$

Error decomposition

Training data $S_{\text{train}} \sim \mathcal{D}$ i.i.d.	$\rightarrow$	Algorithm $\mathcal{A}$	$\rightarrow$	Prediction $f_{S_{\text{train}}} = \mathcal{A}(S_{\text{train}})$

Error decomposition

$$\boxed{\begin{array}{c}\text{Training data}\\ S_{\text{train}} \sim \mathcal{D} \text{ i.i.d.}\end{array}} \rightarrow \boxed{\begin{array}{c}\text{Algorithm}\\ \mathcal{A}\end{array}} \rightarrow \boxed{\begin{array}{c}\text{Prediction}\\ f_{S_{\text{train}}} = \mathcal{A}(S_{\text{train}})\end{array}}$$

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

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\left(y - f_S(\mathbf{x})\right)^2\right] \tag{41}$$

Error decomposition

$$\boxed{\begin{array}{c}\text{Training data}\\ S_{\text{train}} \sim \mathcal{D} \text{ i.i.d.}\end{array}} \rightarrow \boxed{\begin{array}{c}\text{Algorithm}\\ \mathcal{A}\end{array}} \rightarrow \boxed{\begin{array}{c}\text{Prediction}\\ f_{S_{\text{train}}} = \mathcal{A}(S_{\text{train}})\end{array}}$$

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

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[(y - f_S(\mathbf{x}))^2\right] \tag{41}$$

  behaves as a function of

Error decomposition

$$\begin{array}{c} \text{Training data} \\ S_{\text{train}} \sim \mathcal{D} \text{ i.i.d.} \end{array} \rightarrow \begin{array}{c} \text{Algorithm} \\ \mathcal{A} \end{array} \rightarrow \begin{array}{c} \text{Prediction} \\ f_{S_{\text{train}}} = \mathcal{A}(S_{\text{train}}) \end{array}$$

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

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\left(y - f_S(\mathbf{x})\right)^2\right] \tag{41}$$

behaves as a function of

1. the train set S

Error decomposition

Training data $S_{\text{train}} \sim \mathcal{D}$ i.i.d.	$\rightarrow$	Algorithm $\mathcal{A}$	$\rightarrow$	Prediction $f_{S_{\text{train}}} = \mathcal{A}(S_{\text{train}})$

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

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\left(y - f_S(\mathbf{x})\right)^2\right] \tag{41}$$

behaves as a function of

1. the train set S
2. the complexity of the model class

Error decomposition

$$\boxed{\text{Training data } S_{\text{train}} \sim \mathcal{D} \text{ i.i.d.}} \rightarrow \boxed{\text{Algorithm } \mathcal{A}} \rightarrow \boxed{\text{Prediction } f_{S_{\text{train}}} = \mathcal{A}(S_{\text{train}})}$$

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

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\left(y - f_S(\mathbf{x})\right)^2\right] \tag{41}$$

  behaves as a function of
  1. the train set S
  2. the complexity of the model class
- The decomposition will be true for every single point $\mathbf{x}$.

Error decomposition

Training data $S_{\text{train}} \sim \mathcal{D}$ i.i.d. → Algorithm $\mathcal{A}$ → Prediction $f_{S_{\text{train}}} = \mathcal{A}(S_{\text{train}})$

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

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\left(y - f_S(\mathbf{x})\right)^2\right] \tag{41}$$

  behaves as a function of

  1. the train set S
  2. the complexity of the model class

- The decomposition will be true for every single point $\mathbf{x}$.
- To simplify, we consider the expected error of f_S for a fixed element $\mathbf{x}_0$:

$$L(f_{S_{\text{train}}}) = \mathbb{E}_{\epsilon\sim\mathcal{D}_\epsilon}\left[\left(f(\mathbf{x}_0) + \epsilon - f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right] \tag{42}$$

A decomposition in three terms

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

$$\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] \tag{43}$$

(44)

(45)

A decomposition in three terms

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

$$\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] \tag{43}$$

$$:= \mathbb{E}_{S_{\text{train}} \in \mathcal{D}, \boldsymbol{\epsilon} \in \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\left(f(\mathbf{x}_0) + \boldsymbol{\epsilon} - \boxed{f_{S_{\text{train}}}}(\mathbf{x}_0) \right)^2 \right] \tag{44}$$

$$\tag{45}$$

A decomposition in three terms

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

$$\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] \tag{43}$$

$$:= \mathbb{E}_{S_{\text{train}} \in \mathcal{D}, \boldsymbol{\epsilon} \in \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\left(f(\mathbf{x}_0) + \boldsymbol{\epsilon} - \boxed{f_{S_{\text{train}}}}(\mathbf{x}_0) \right)^2 \right] \tag{44}$$

$$= \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\boldsymbol{\epsilon}^2 \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D},\, \boldsymbol{\epsilon} \sim \mathcal{D}_{\boldsymbol{\epsilon}}} \left[2\boldsymbol{\epsilon}(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0)) \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2 \right] \tag{45}$$

A decomposition in three terms

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

$$\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] \tag{43}$$

$$:= \mathbb{E}_{S_{\text{train}} \in \mathcal{D}, \boldsymbol{\epsilon} \in \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\left(f(\mathbf{x}_0) + \boldsymbol{\epsilon} - \boxed{f_{S_{\text{train}}}}(\mathbf{x}_0) \right)^2 \right] \tag{44}$$

$$= \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\boldsymbol{\epsilon}^2 \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D},\, \boldsymbol{\epsilon} \sim \mathcal{D}_{\boldsymbol{\epsilon}}} \left[2\boldsymbol{\epsilon} (f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0)) \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2 \right] \tag{45}$$

Using that $\mathbb{E}_{\boldsymbol{\epsilon} \in \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\boldsymbol{\epsilon} \right] = 0$ and $\boldsymbol{\epsilon}$ is independent from S_{train}:

A decomposition in three terms

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

$$
\begin{aligned}
&\mathbb{E}_{S_{\text{train}}\in\mathcal{D}}\left[L(f_{S_{\text{train}}})\right] && (43)\\
&:= \mathbb{E}_{S_{\text{train}}\in\mathcal{D},\epsilon\in\mathcal{D}_\epsilon}\left[\left(f(\mathbf{x}_0)+\epsilon-\boxed{f_{S_{\text{train}}}}(\mathbf{x}_0)\right)^2\right] && (44)\\
&= \mathbb{E}_{\epsilon\sim\mathcal{D}_\epsilon}\left[\epsilon^2\right] + \mathbb{E}_{S_{\text{train}}\sim\mathcal{D},\,\epsilon\sim\mathcal{D}_\epsilon}\left[2\epsilon(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))\right] + \mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] && (45)
\end{aligned}
$$

Using that $\mathbb{E}_{\epsilon\in\mathcal{D}_\epsilon}[\epsilon]=0$ and ϵ is independent from S_{train}:

- $\mathbb{E}_{\epsilon\sim\mathcal{D}_\epsilon}\left[\epsilon^2\right] = \text{Var}_{\epsilon\in\mathcal{D}_\epsilon}[\epsilon]$

A decomposition in three terms

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

$$
\begin{aligned}
&\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] && (43) \\
&:= \mathbb{E}_{S_{\text{train}} \in \mathcal{D}, \epsilon \in \mathcal{D}_\epsilon} \left[\left(f(\mathbf{x}_0) + \epsilon - f_{S_{\text{train}}}(\mathbf{x}_0) \right)^2 \right] && (44) \\
&= \mathbb{E}_{\epsilon \sim \mathcal{D}_\epsilon} \left[\epsilon^2 \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D},\, \epsilon \sim \mathcal{D}_\epsilon} \left[2\epsilon \left(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0) \right) \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[\left(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0) \right)^2 \right] && (45)
\end{aligned}
$$

Using that $\mathbb{E}_{\epsilon \in \mathcal{D}_\epsilon}[\epsilon] = 0$ and ϵ is independent from S_{train}:

- $\mathbb{E}_{\epsilon \sim \mathcal{D}_\epsilon}\left[\epsilon^2\right] = \text{Var}_{\epsilon \in \mathcal{D}_\epsilon}[\epsilon]$
- $\mathbb{E}_{S_{\text{train}} \sim \mathcal{D},\, \epsilon \sim \mathcal{D}_\epsilon}\left[2\epsilon\left(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0)\right)\right] = 0.$

A decomposition in three terms

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

$$\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] \tag{43}$$

$$:= \mathbb{E}_{S_{\text{train}} \in \mathcal{D}, \epsilon \in \mathcal{D}_\epsilon} \left[\left(f(\mathbf{x}_0) + \epsilon - \boxed{f_{S_{\text{train}}}}(\mathbf{x}_0) \right)^2 \right] \tag{44}$$

$$= \mathbb{E}_{\epsilon \sim \mathcal{D}_\epsilon} \left[\epsilon^2 \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D},\, \epsilon \sim \mathcal{D}_\epsilon} \left[2\epsilon (f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0)) \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2 \right] \tag{45}$$

Using that $\mathbb{E}_{\epsilon \in \mathcal{D}_\epsilon} [\epsilon] = 0$ and ϵ is independent from S_{train}:

- $\mathbb{E}_{\epsilon \sim \mathcal{D}_\epsilon} \left[\epsilon^2 \right] = \text{Var}_{\epsilon \in \mathcal{D}_\epsilon} [\epsilon]$
- $\mathbb{E}_{S_{\text{train}} \sim \mathcal{D},\, \epsilon \sim \mathcal{D}_\epsilon} \left[2\epsilon (f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0)) \right] = 0.$

Therefore

$$\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] = \underbrace{\text{Var}_{\epsilon \in \mathcal{D}_\epsilon} [\epsilon]}_{\text{noise variance}} + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2 \right] \tag{46}$$

We can further decompose $\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right]$ into two terms:

$$\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{47}$$

$$= \mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\left(\qquad\qquad + \qquad\qquad \right)^2\right] \tag{48}$$

(50)

(52)

We can further decompose $\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right]$ into two terms:

$$\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{47}$$

$$= \mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}}[f_{S_{\text{train'}}}(\mathbf{x}_0)] + \qquad)^2\right] \tag{48}$$

(50)

(52)

We can further decompose $\mathbb{E}_{S_{\text{train}} \sim \mathcal{D}}\left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right]$ into two terms:

$$\mathbb{E}_{S_{\text{train}} \sim \mathcal{D}}\left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{47}$$

$$= \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}}\left[(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] + \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{48}$$

(50)

(52)

We can further decompose $\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right]$ into two terms:

$$
\begin{aligned}
&\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] && (47)\\
&=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\left(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]+\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right] && (48)\\
&=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\qquad\qquad\qquad\qquad\right] && (49)\\
& && (50)\\
& && (52)
\end{aligned}
$$

We can further decompose $\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right]$ into two terms:

$$\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{47}$$

$$=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]+\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{48}$$

$$=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right])^2\right] \tag{49}$$

(50)

(52)

We can further decompose $\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right]$ into two terms:

$$
\begin{aligned}
&\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] && (47)\\
&=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\left(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]+\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right] && (48)\\
&=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\left(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]\right)^2+\left(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right] && (49)\\
& && (50)
\end{aligned}
$$

$$(52)$$

We can further decompose $\mathbb{E}_{S_{\text{train}} \sim \mathcal{D}}\left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right]$ into two terms:

$$\mathbb{E}_{S_{\text{train}} \sim \mathcal{D}}\left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{47}$$

$$= \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}}\left[(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] + \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{48}$$

$$= \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}}\left[\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]\right)^2 + \left(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right] \tag{49}$$

$$\underbrace{+\mathbb{E}_{S_{\text{train}} \sim \mathcal{D}}\left[2\left(\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]\right)\right)\left(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0)\right)\right)\right]}_{0} \tag{50}$$

(52)

We can further decompose $\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right]$ into two terms:

$$\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{47}$$

$$=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]+\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{48}$$

$$=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right])^2+(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{49}$$

$$\underbrace{+\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[2\left((f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right])\left(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0)\right)\right)\right]}_{0} \tag{50}$$

$$=\underbrace{\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right])^2\right]}_{\text{bias}} \tag{51}$$

$$\tag{52}$$

We can further decompose $\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right]$ into two terms:

$$
\begin{aligned}
&\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] && (47)\\
&=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]+\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] && (48)\\
&=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right])^2+(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] && (49)\\
&\quad\underbrace{+\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[2\left((f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]\right)\left(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0)\right)\right)\right]}_{0} && (50)\\
&=\underbrace{\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right])^2\right]}_{\text{bias}} && (51)\\
&\quad+\underbrace{\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right]}_{\text{variance}} && (52)
\end{aligned}
$$

Bias-Variance Decomposition

$$\begin{aligned}
\mathbb{E}_{S_{\text{train}} \in \mathcal{D}}\left[L(f_{S_{\text{train}}})\right] &= \mathrm{Var}_{\epsilon\sim\mathcal{D}_\epsilon}[\epsilon] && \text{(noise variance)}\\
&+ \left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train}'}}\left[f_{S_{\text{train}'}}(\mathbf{x}_0)\right]\right)^2 && \text{(Bias)}\\
&+ \mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\left(\mathbb{E}_{S_{\text{train}'}}\left[f_{S_{\text{train}'}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right], && \text{(Variance)}
\end{aligned}$$

Bias-Variance Decomposition

$$
\begin{aligned}
\mathbb{E}_{S_{\text{train}} \in \mathcal{D}}\left[L(f_{S_{\text{train}}})\right] &= \text{Var}_{\epsilon \sim \mathcal{D}_\epsilon}[\epsilon] && \text{(noise variance)} \\
&+ \left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train}'}}\left[f_{S_{\text{train}'}}(\mathbf{x}_0)\right]\right)^2 && \text{(Bias)} \\
&+ \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}}\left[\left(\mathbb{E}_{S_{\text{train}'}}\left[f_{S_{\text{train}'}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right], && \text{(Variance)}
\end{aligned}
$$

which always lowers bound the true error.

Bias-Variance Decomposition

$$
\begin{aligned}
\mathbb{E}_{S_{\text{train}} \in \mathcal{D}}\left[L(f_{S_{\text{train}}})\right] &= \text{Var}_{\epsilon \sim \mathcal{D}_\epsilon}[\epsilon] && \text{(noise variance)} \\
&+ \left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train}'}}\left[f_{S_{\text{train}'}}(\mathbf{x}_0)\right]\right)^2 && \text{(Bias)} \\
&+ \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}}\left[\left(\mathbb{E}_{S_{\text{train}'}}\left[f_{S_{\text{train}'}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right], && \text{(Variance)}
\end{aligned}
$$

which always lowers bound the true error.

$\Rightarrow$ In order to minimize the true error, we need to select a method that **simultaneously achieves low bias and low variance**.

Component 1: Noise $\text{Var}_{\epsilon \sim \mathcal{D}_\epsilon}[\epsilon]$—a strict lower bound on what error we can achieve

Component 1: Noise $\mathrm{Var}_{\epsilon\sim\mathcal{D}_\epsilon}[\epsilon]$—a strict lower bound on what error we can achieve

- It is not possible to go below the noise level

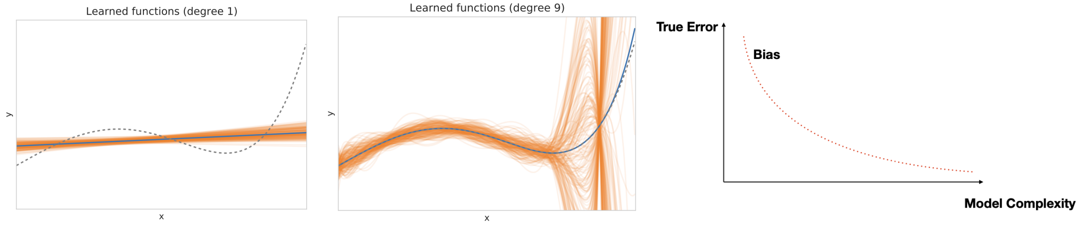
Component 1: Noise $\mathrm{Var}_{\epsilon \sim \mathcal{D}_\epsilon}[\epsilon]$—a strict lower bound on what error we can achieve

- It is not possible to go below the noise level
- Even if we know the true model f, we still suffer from $L(f) = \mathbb{E}\left[\epsilon^2\right]$

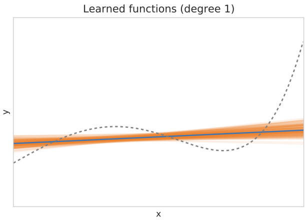
Component 1: Noise $\mathrm{Var}_{\epsilon \sim \mathcal{D}_\epsilon}[\epsilon]$—a strict lower bound on what error we can achieve

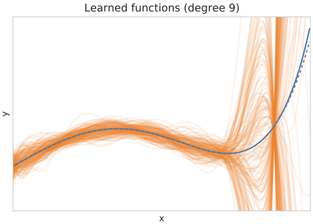
- It is not possible to go below the noise level
- Even if we know the true model f, we still suffer from $L(f) = \mathbb{E}\left[\epsilon^2\right]$
- It is not possible to predict the noise from the data since they are independent

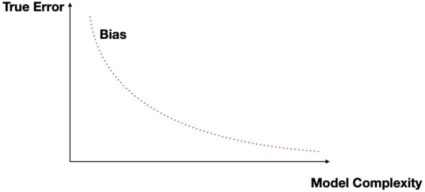
Component 2: Bias $\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train}'}}\left[f_{S_{\text{train}'}}(\mathbf{x}_0)\right]\right)^2$



Component 2: Bias $\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train}}}\left[f_{S_{\text{train}}}(\mathbf{x}_0)\right]\right)^2$

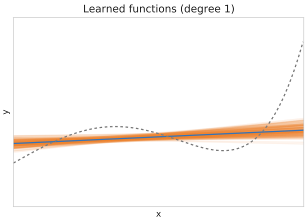


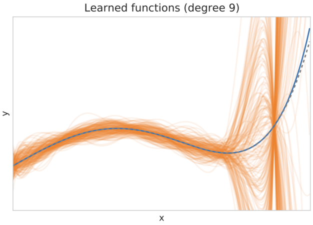


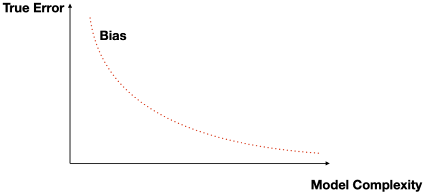


- It measures how far off in general the models' predictions are from the correct value.

Component 2: Bias $(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train}'}}[f_{S_{\text{train}'}}(\mathbf{x}_0)])^2$

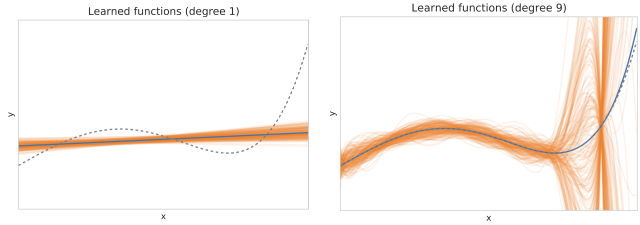


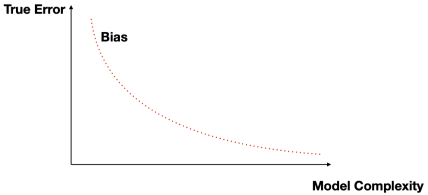




- It measures how far off in general the models' predictions are from the correct value.
- If complexity is small, then high bias.

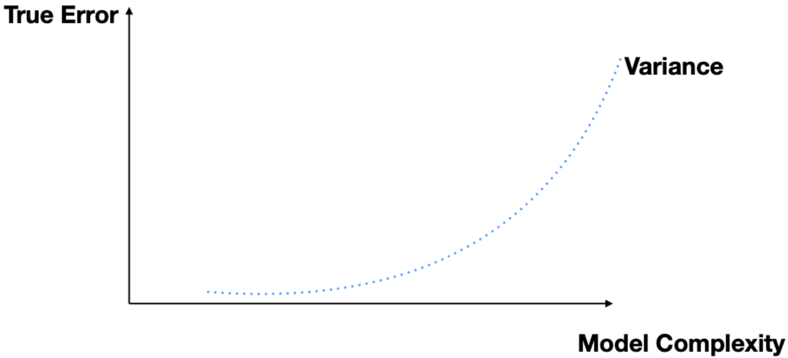
Component 2: Bias $\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train}}}\left[f_{S_{\text{train}}}(\mathbf{x}_0)\right]\right)^2$



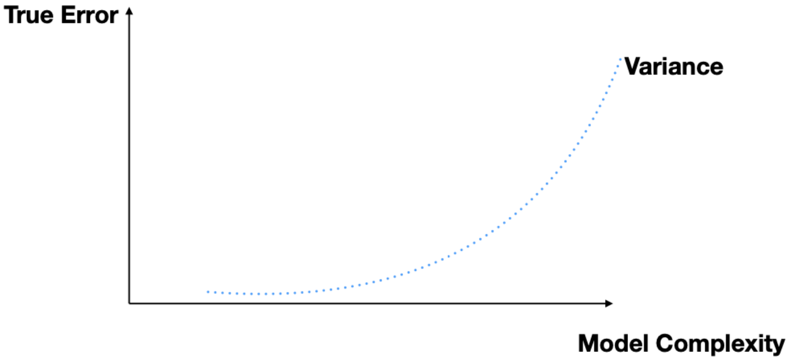


- It measures how far off in general the models' predictions are from the correct value.
- If complexity is small, then high bias.
- If complexity is high, then low bias.

Component 3: Variance $\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\left(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right]$

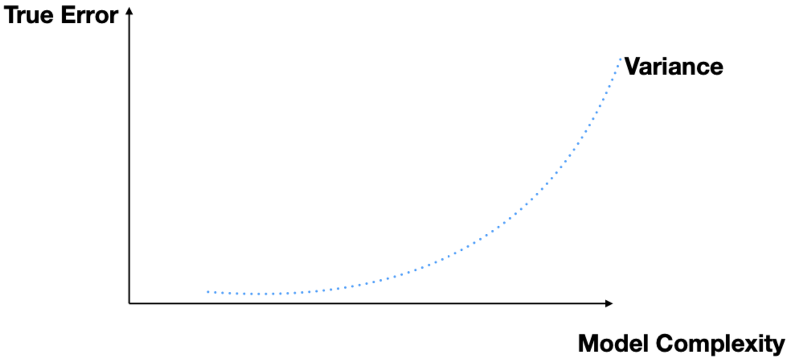


Component 3: Variance $\mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[\left(\mathbb{E}_{S_{\text{train'}}} \left[f_{S_{\text{train'}}}(\mathbf{x}_0) \right] - f_{S_{\text{train}}}(\mathbf{x}_0) \right)^2 \right]$



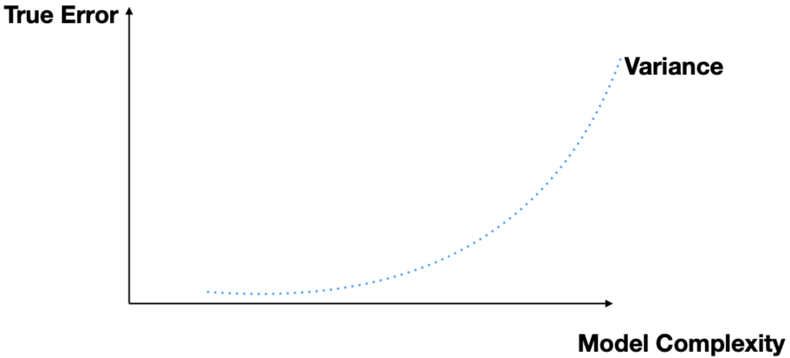
- Variance of the prediction function.

Component 3: Variance $\mathbb{E}_{S_{\text{train}} \sim \mathcal{D}}\left[\left(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right]$



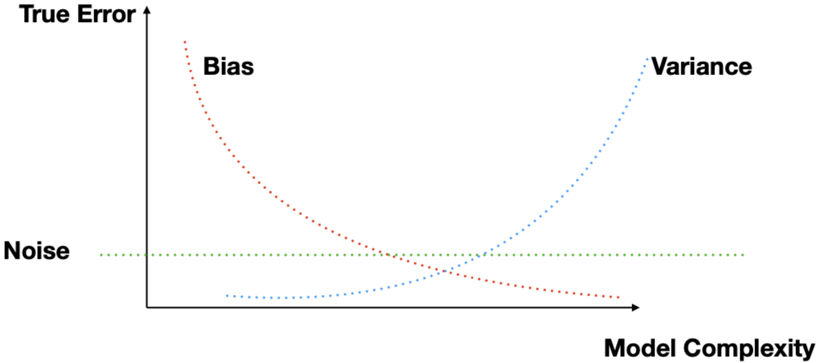
- Variance of the prediction function.
- It is how much the predictions for a given point vary between different realizations of the training set.

Component 3: Variance $\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\left(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right]$

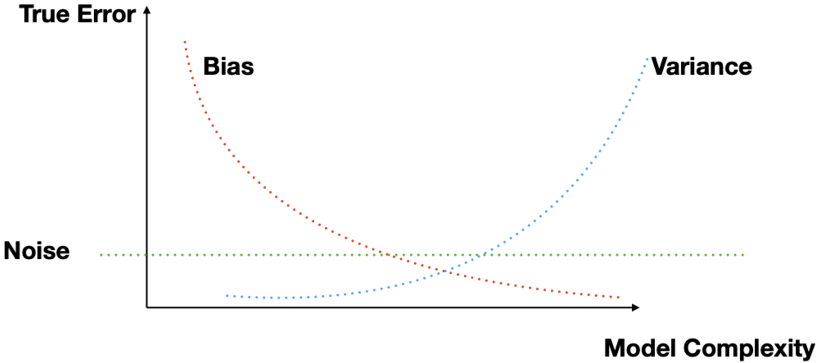


- Variance of the prediction function.
- It is how much the predictions for a given point vary between different realizations of the training set.
- If we consider complicated models, then small variations in the training set can result in large changes in the prediction.

Bias Variance tradeoff and U-shape curve

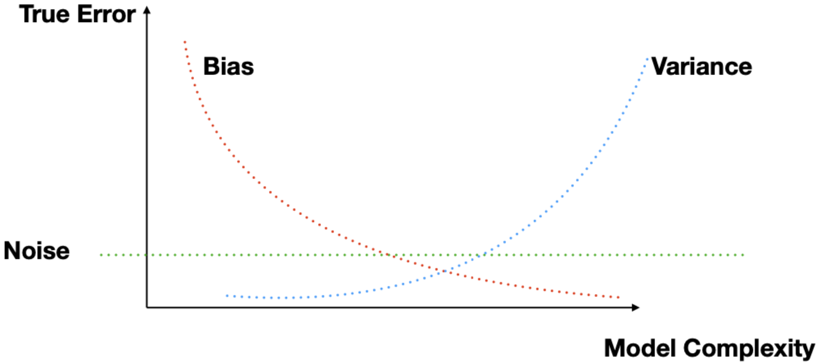


Bias Variance tradeoff and U-shape curve

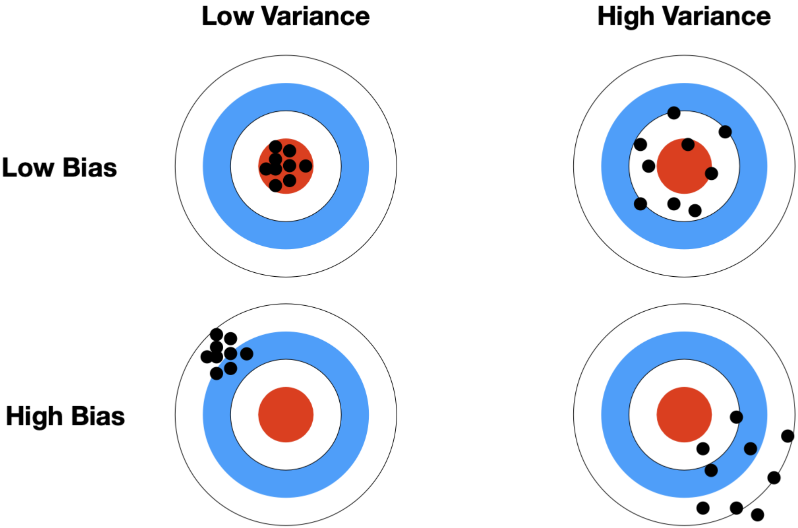


- If the complexity is too low, you cannot approximate well (under-fitting)

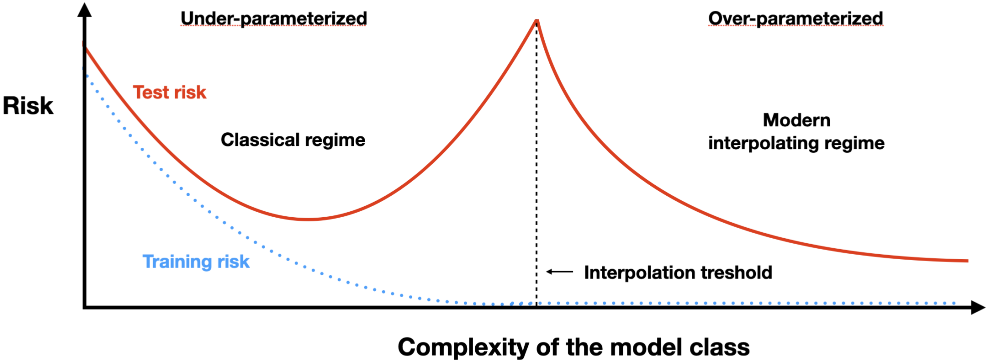
Bias Variance tradeoff and U-shape curve



- If the complexity is too low, you cannot approximate well (under-fitting)
- If the complexity is too large, you have a problem with the variance (over-fitting)


Low Variance
High Variance
Low Bias
High Bias

Double descent curve in Deep Learning



Last lecture:

- Normal Equation and Least Squares
- Probabilistic Interpretation of Linear Regression
- Maximum Likelihood Estimation (MLE)
- Over-fitting and Under-fitting
- Polynomial Regression, Ridge Regression, and Lasso

This lecture:

- Generalization Gap and Model Selection
- Bias-Variance Decomposition