Lecture 4: Model Selection and Bias-Variance Decomposition

Tao LIN

March 8, 2023

WESTLAKE UNIVERSITY | SCHOOL OF ENGINEERING

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

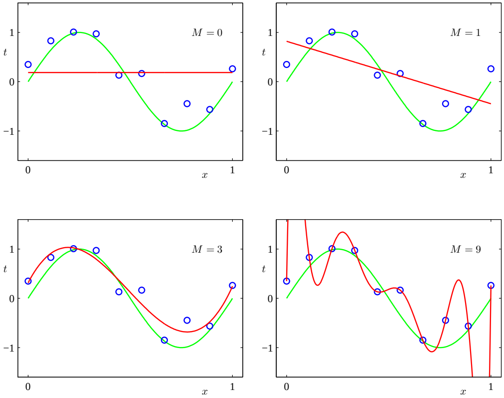
- EPFL, CS-433 Machine Learning, `https://github.com/epfml/ML_course`

Table of Contents

Definitions of under-fit and over-fit

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

$$
\begin{aligned}
y_n &\approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M \\
&=: \phi(x_n)^\top \mathbf{w}\,.
\end{aligned}
\tag{1}
$$

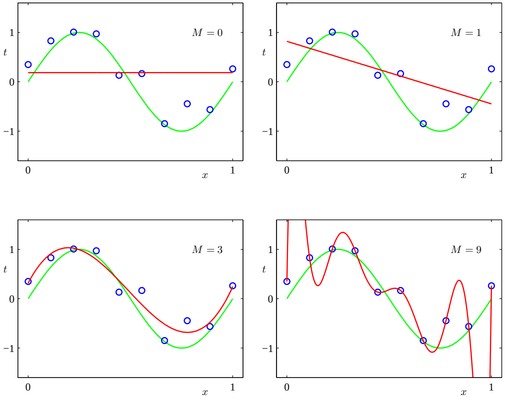


Definitions of under-fit and over-fit

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

$$
\begin{aligned}
y_n &\approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M \\
&=: \phi(x_n)^\top \mathbf{w} \,.
\end{aligned} \tag{1}
$$

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

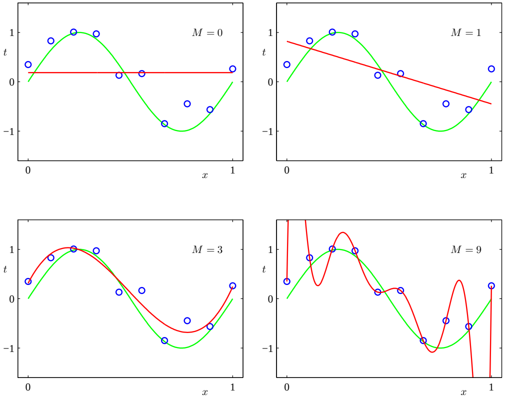


Definitions of under-fit and over-fit

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

$$
\begin{aligned}
y_n &\approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M \\
&=: \phi(x_n)^\top \mathbf{w}\,.
\end{aligned}
\tag{1}
$$

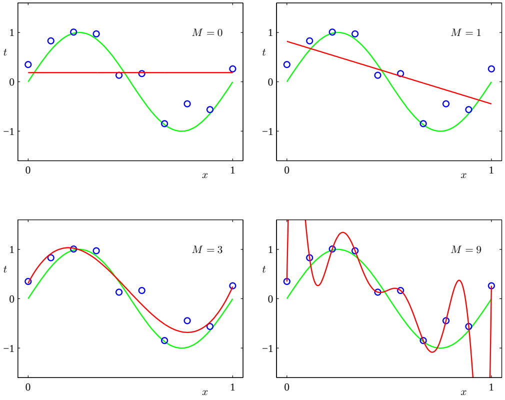
- under-fit: cannot find the *the underlying function* of the data.
- over-fit: can fit both *the underlying function* and *the noise in the data*.



Definitions of under-fit and over-fit

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

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w}\,. \tag{1}$$



- 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*.

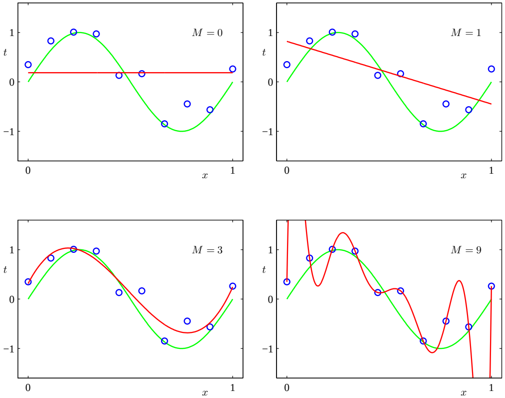
Discussion:

- 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):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w} \,. \tag{1}$$



- 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*.

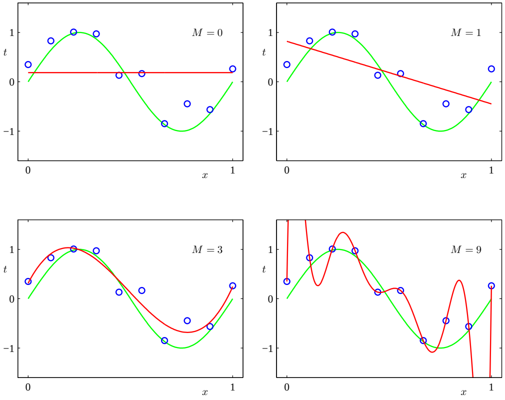
Discussion:

- 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):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M \\ =: \phi(x_n)^\top \mathbf{w}\,. \quad (1)$$



- 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:

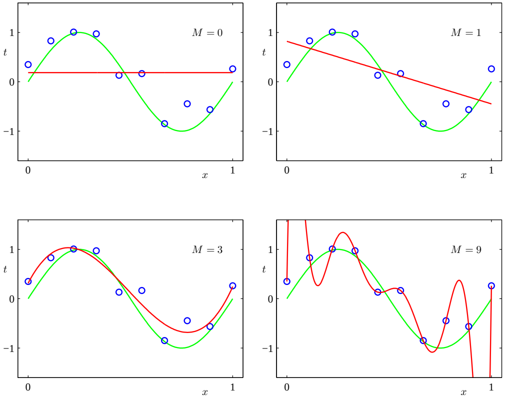
- Both of these phenomena (under-fit and over-fit) are undesirable.
- This discussion is made more difficult:
  - since all we have is data.

Definitions of under-fit and over-fit

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

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M \\ =: \phi(x_n)^\top \mathbf{w} \,. \tag{1}$$

- 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:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{2}$$

Regularization

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

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{2}$$

- 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:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{2}$$

- 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:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{2}$$

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

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_2^2 \;, \tag{3}$$

where 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:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{2}$$

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

    $$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \, \|\mathbf{w}\|_2^2 \; , \tag{3}$$

    where 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)

    $$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \, \|\mathbf{w}\|_1 \; . \tag{4}$$

Regularization

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

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{2}$$

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

    $$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \, \|\mathbf{w}\|_2^2 \; , \tag{3}$$

    where 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)

    $$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \, \|\mathbf{w}\|_1 \; . \tag{4}$$

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

Regularization

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

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{2}$$

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

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_2^2 \; , \tag{3}$$

    where 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)

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_1 \; . \tag{4}$$

- 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:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{2}$$

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

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \ + \ \lambda \|\mathbf{w}\|_2^2 \ , \tag{3}$$

    where 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)

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \ + \ \lambda \|\mathbf{w}\|_1 \ . \tag{4}$$

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

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

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 $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{5}$$

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 $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{5}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{6}$$

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 $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{5}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{6}$$

- 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 $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{5}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{6}$$

- 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 $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{5}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{6}$$

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

True Error, Empirical Error, and Training Error

- **True Error** (the **expected error** over all samples chosen according to $\mathcal{D}$):

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] . \tag{7}$$

  We cannot estimate it due to the unknown $\mathcal{D}$.

True Error, Empirical Error, and Training Error

- **True Error** (the **expected error** over all samples chosen according to $\mathcal{D}$):

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] . \tag{7}$$

  We cannot estimate it due to the unknown $\mathcal{D}$.

- **Empirical Error** (what we can compute—the approximated true error via dataset S):

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) . \tag{8}$$

True Error, Empirical Error, and Training Error

- **True Error** (the **expected error** over all samples chosen according to $\mathcal{D}$):

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] . \tag{7}$$

  We cannot estimate it due to the unknown $\mathcal{D}$.

- **Empirical Error** (what we can compute—the approximated true error via dataset S):

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) . \tag{8}$$

- **Training Error** (what we are minimizing—when the model has been trained on the same data it is applied to):

$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f_S(\mathbf{x}_n)) . \tag{9}$$

True Error, Empirical Error, and Training Error

- **True Error** (the **expected error** over all samples chosen according to $\mathcal{D}$):

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] . \tag{7}$$

  We cannot estimate it due to the unknown $\mathcal{D}$.

- **Empirical Error** (what we can compute—the approximated true error via dataset S):

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) . \tag{8}$$

- **Training Error** (what we are minimizing—when the model has been trained on the same data it is applied to):

$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f_S(\mathbf{x}_n)) . \tag{9}$$

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

Problem: validating the model on the same data subset we trained it on!

Splitting the data and Test Error

- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{10}$$

Splitting the data and Test Error

- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{10}$$

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

Splitting the data and Test Error

- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{10}$$

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}:

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n, \mathbf{x}_n) \in S_{\text{test}}} \ell\left(y_n, f_{S_{\text{train}}}(\mathbf{x}_n)\right). \tag{11}$$

Splitting the data and Test Error

- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{10}$$

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}:

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n, \mathbf{x}_n) \in S_{\text{test}}} \ell\left(y_n, f_{S_{\text{train}}}(\mathbf{x}_n)\right). \tag{11}$$

- 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

- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{10}$$

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}:

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n, \mathbf{x}_n) \in S_{\text{test}}} \ell\left(y_n, f_{S_{\text{train}}}(\mathbf{x}_n)\right). \tag{11}$$

- 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)

Last lecture:

- Over-fitting and Under-fitting
- Polynomial Regression, Ridge Regression, and Lasso Regression
- Basic Concepts for Generalization and Model Selection

Last lecture:

- Over-fitting and Under-fitting
- Polynomial Regression, Ridge Regression, and Lasso Regression
- Basic Concepts for Generalization and Model Selection

This lecture:

- Generalization Gap and Model Selection
- Bias-Variance Decomposition
- Before Introducing Multilayer Perceptron: Logistic Regression

Table of Contents

Table of Contents

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

- **True Error:**

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f\left(\mathbf{x}\right)\right)\right]. \tag{12}$$

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

- **True Error:**

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right]. \tag{12}$$

- **Test/Empirical Error:**

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n)\in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{13}$$

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

- **True Error:**

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right]. \tag{12}$$

- **Test/Empirical Error:**

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|}\sum_{(\mathbf{x}_n, y_n)\in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{13}$$

- **Generalization Error:**

$$\left|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)\right|. \tag{14}$$

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

- **True Error:**

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right]. \tag{12}$$

- **Test/Empirical Error:**

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n)\in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{13}$$

- **Generalization Error:**

$$\left|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)\right| . \tag{14}$$

In expectation, they are the same:

$$L_{\mathcal{D}}(f) = \mathbb{E}_{S_{\text{test}}\sim\mathcal{D}}\left[L_{S_{\text{test}}}(f)\right] , \tag{15}$$

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 1

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

$$\Pr\left[|L_{\mathcal{D}}(f) - L_{S_{test}}(f)| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{16}$$

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

Theorem 1

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

$$\Pr\left[|L_{\mathcal{D}}(f) - L_{S_{test}}(f)| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{16}$$

- 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 1

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

$$\Pr\left[|L_{\mathcal{D}}(f) - L_{S_{test}}(f)| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{16}$$

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

Theorem 1

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

$$\Pr\left[|L_{\mathcal{D}}(f) - L_{S_{test}}(f)| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{16}$$

- 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:

$$\Pr\left[\underbrace{L_{\mathcal{D}}(f)}_{\text{not computable}} \geq \underbrace{L_{S_{\text{test}}}(f)}_{\text{computable}} + \underbrace{\sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\left|S_{\text{test}}\right|}}}_{\text{deviation}}\right] \leq \delta\,. \tag{17}$$

Why do you care?

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

$$\Pr\left[\underbrace{L_{\mathcal{D}}(f)}_{\text{not computable}} \geq \underbrace{L_{S_{\text{test}}}(f)}_{\text{computable}} + \underbrace{\sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{\text{test}}|}}}_{\text{deviation}}\right] \leq \delta\,. \tag{17}$$

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:

$$\Pr\left[\underbrace{L_{\mathcal{D}}(f)}_{\text{not computable}} \geq \underbrace{L_{S_{\text{test}}}(f)}_{\text{computable}} + \underbrace{\sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{\text{test}}|}}}_{\text{deviation}}\right] \leq \delta\,. \tag{17}$$

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:

$$\Pr\left[\underbrace{L_{\mathcal{D}}(f)}_{\text{not computable}} \geq \underbrace{L_{S_{\text{test}}}(f)}_{\text{computable}} + \underbrace{\sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{\text{test}}|}}}_{\text{deviation}}\right] \leq \delta\,. \tag{17}$$

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:

$$\Pr\left[\underbrace{L_{\mathcal{D}}(f)}_{\text{not computable}} \geq \underbrace{L_{S_{\text{test}}}(f)}_{\text{computable}} + \underbrace{\sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{\text{test}}|}}}_{\text{deviation}}\right] \leq \delta\,. \tag{17}$$

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:

$$\Pr\left[\underbrace{L_{\mathcal{D}}(f)}_{\text{not computable}} \geq \underbrace{L_{S_{\text{test}}}(f)}_{\text{computable}} + \underbrace{\sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{\text{test}}|}}}_{\text{deviation}}\right] \leq \delta\,. \tag{17}$$

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:

$$\Pr\left[L_{\mathcal{D}}(f_{S_{\text{train}}}) \geq L_{S_{\text{test}}}(f_{S_{\text{train}}}) + \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2\,|S_{\text{test}}|}}\right] \leq \delta\,. \tag{18}$$

Table of Contents

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

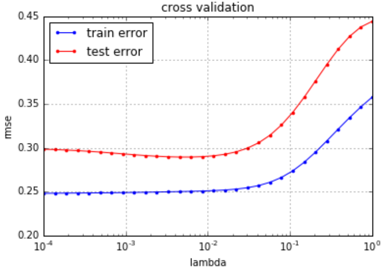
Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

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:

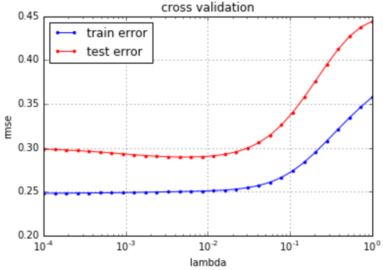


Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

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

Solutions:

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

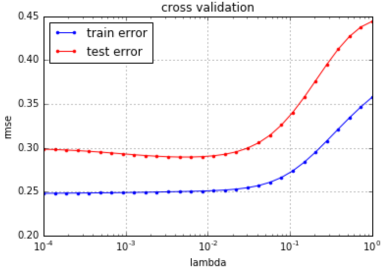


Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

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 $f_{S_{\text{train}},\lambda_k}$.

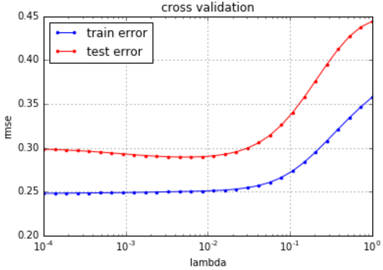


Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

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 $L_{S_{\text{test}}}(f_{S_{\text{train}},\lambda_k})$



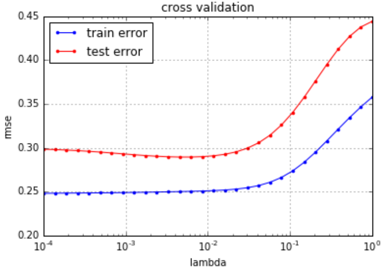
Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

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.



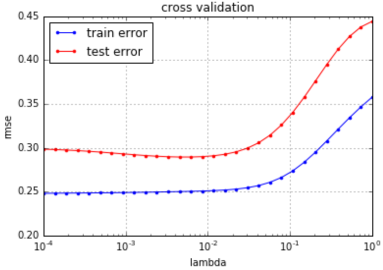
Goal: select the hyper-parameters of our model (e.g., λ for the ridge regression).

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 2

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

$$\Pr\left[\max_k \left|L_{\mathcal{D}}(f_k) - L_{S_{test}}(f_k)\right| \geq \sqrt{\frac{(b-a)^2 \ln(2\boxed{K}/\delta)}{2\left|S_{test}\right|}}\right] \leq \delta \tag{19}$$

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

Theorem 2

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

$$\Pr\left[\max_k |L_{\mathcal{D}}(f_k) - L_{S_{test}}(f_k)| \geq \sqrt{\frac{(b-a)^2 \ln(2\boxed{K}/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{19}$$

- 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 2

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

$$\Pr\left[\max_k |L_{\mathcal{D}}(f_k) - L_{S_{test}}(f_k)| \geq \sqrt{\frac{(b-a)^2 \ln(2\,K\,/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{19}$$

- 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 2

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

$$\Pr\left[\max_k |L_{\mathcal{D}}(f_k) - L_{S_{test}}(f_k)| \geq \sqrt{\frac{(b-a)^2 \ln(2\boxed{K}/\delta)}{2\,|S_{test}|}}\right] \leq \delta \tag{19}$$

- 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.

$$\Pr\left[L_{\mathcal{D}}(f_{\hat{k}}) \geq L_{\mathcal{D}}(f_{k^\star}) + \sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2\,|S_{\text{test}}|}}\right] \leq \delta \tag{20}$$

- $k^\star = \arg\min_k L_{\mathcal{D}}(f_k)$, 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.

$$\Pr\left[L_{\mathcal{D}}(f_{\hat{k}}) \geq L_{\mathcal{D}}(f_{k^\star}) + \sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2\,|S_{\text{test}}|}}\right] \leq \delta \tag{20}$$

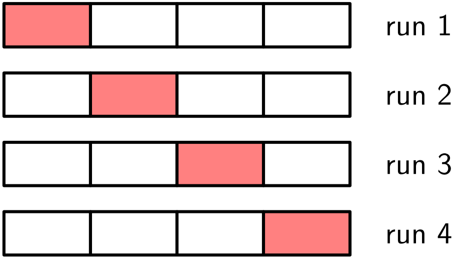
If we choose the “best” function according to the empirical risk, then its true risk is not too far away from the true risk of the optimal choice.

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!

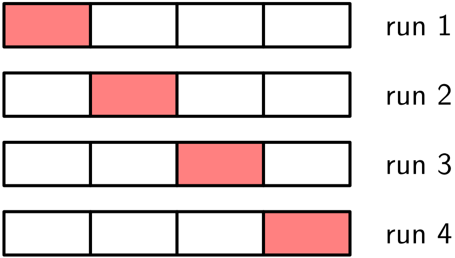
K-fold cross-validation:



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:

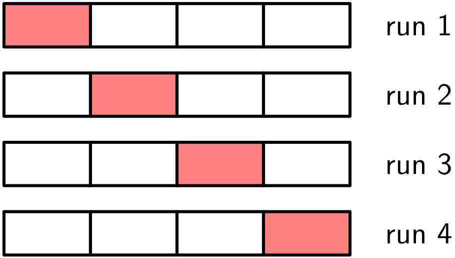
1. Randomly partition the data into K groups



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:

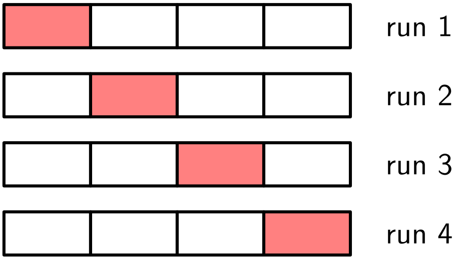
1. Randomly partition the data into K groups
2. Train K times.



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:

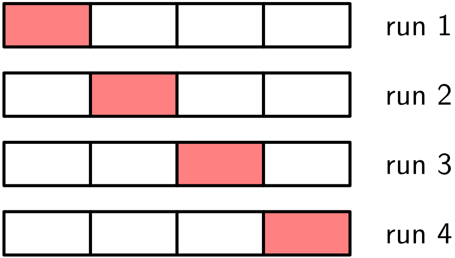
1. Randomly partition the data into K groups
2. Train K times. Each time leave out exactly one of the K groups for testing



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:

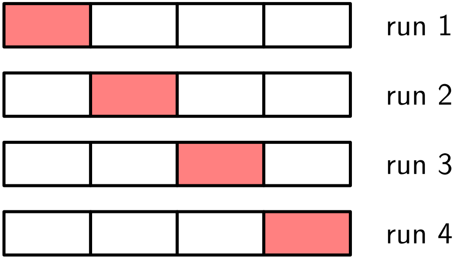
1. Randomly partition the data into K groups
2. Train K times. Each time leave out exactly one of the K groups for testing and use the remaining $K-1$ groups for training.



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:

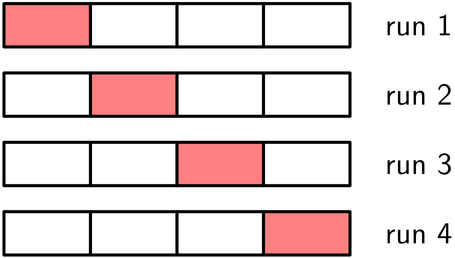
1. Randomly partition the data into K groups
2. Train K times. Each time leave out exactly one of the K groups for testing and use the remaining $K - 1$ groups for training.
3. Average the K results



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

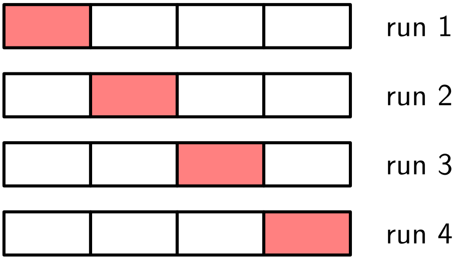


Benefits:

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



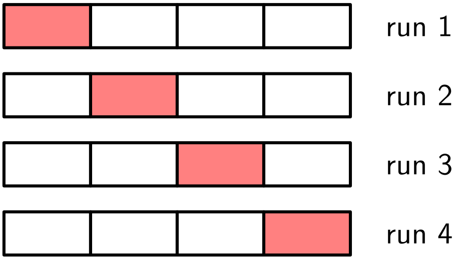
Benefits:

- 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:**

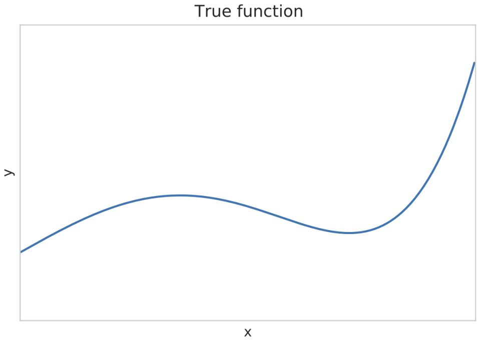
  $\Rightarrow$ Bound the difference between the true and empirical risks.

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

This part:

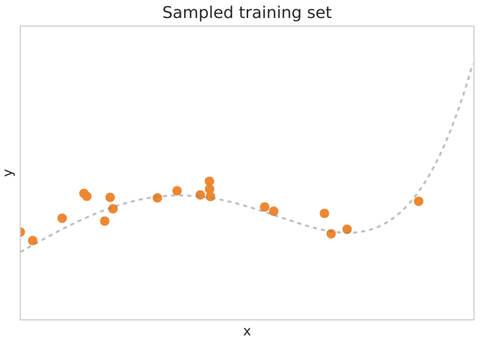
- the role of the complexity of the class
- How does the risk behave as a function of the complexity of the model class?
- It will help us to decide how complex and rich we should make our model

Motivation example: 1D-regression



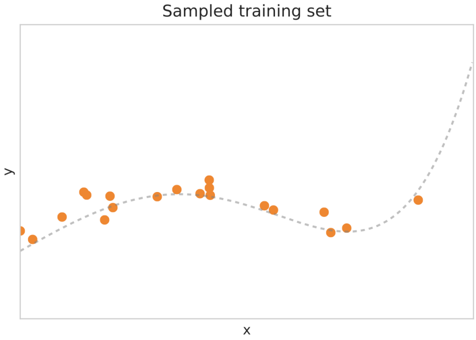
We have a true underlying function in blue we would like to recover.

Motivation example: 1D-regression



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

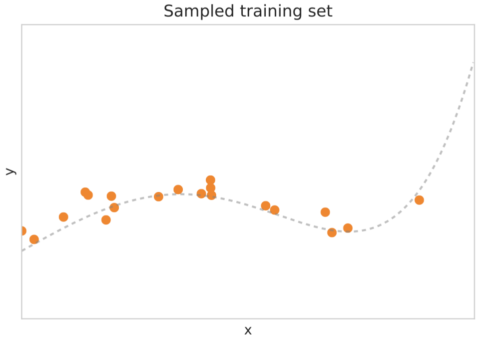
Motivation example: 1D-regression



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

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

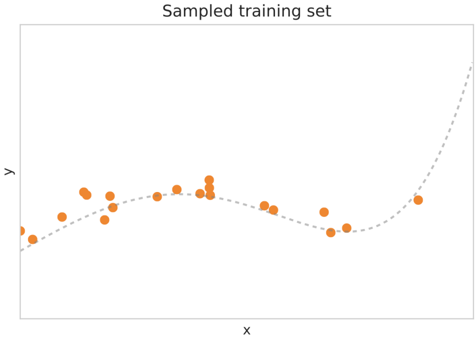
Motivation example: 1D-regression



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

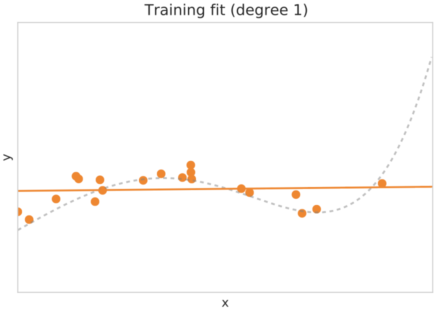


We have a true underlying function in blue we would like to recover.

- 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?

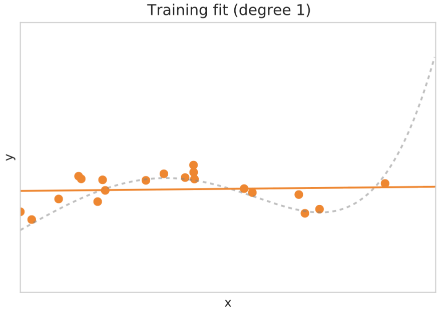
Motivation example: 1D-regression

If we restrict ourselves to simple models:



Motivation example: 1D-regression

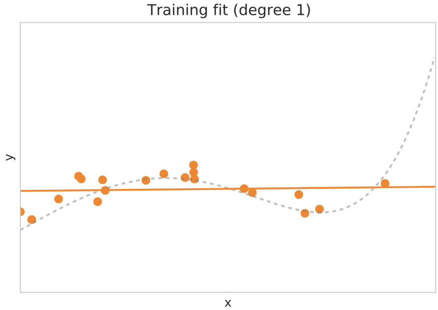
If we restrict ourselves to simple models:



- No linear function would be a good predictor.

Motivation example: 1D-regression

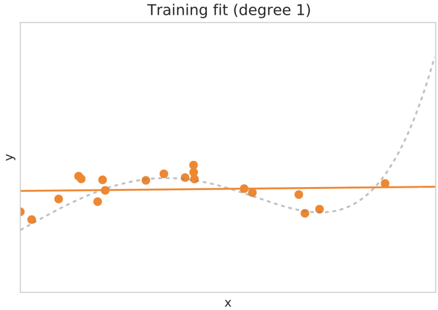
If we restrict ourselves to simple models:



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

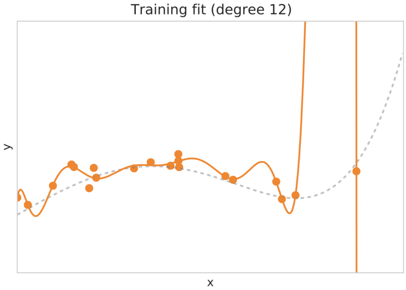
Motivation example: 1D-regression

If we restrict ourselves to simple models:



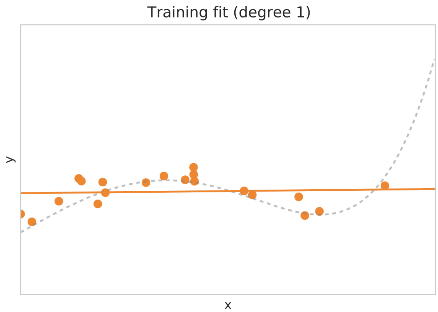
- No linear function would be a good predictor.
- The model is not rich/expressive enough.

If we consider a high-degree polynomial:



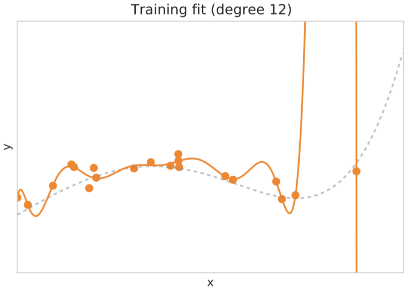
Motivation example: 1D-regression

If we restrict ourselves to simple models:



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

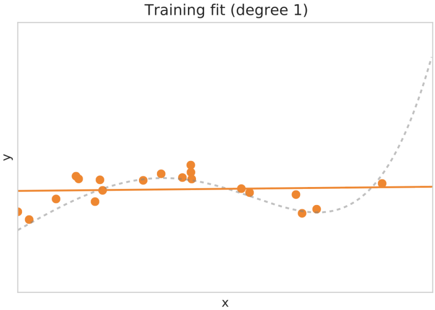
If we consider a high-degree polynomial:



- We can find a model that fits the data well

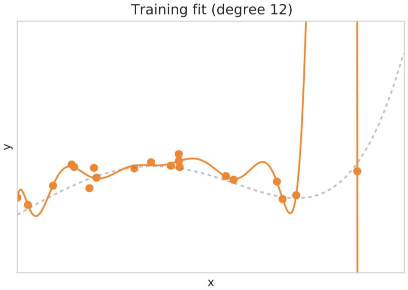
Motivation example: 1D-regression

If we restrict ourselves to simple models:



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

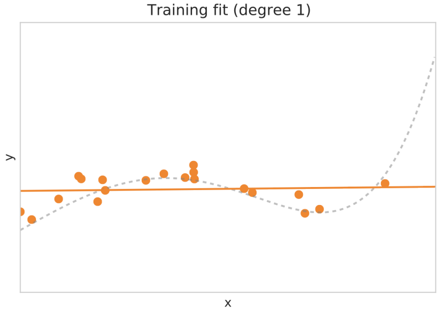
If we consider a high-degree polynomial:



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

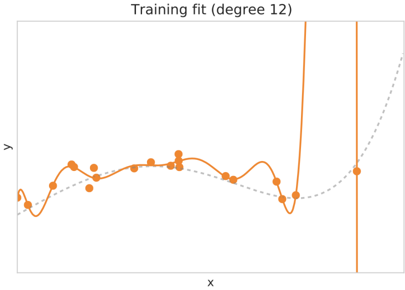
Motivation example: 1D-regression

If we restrict ourselves to simple models:



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

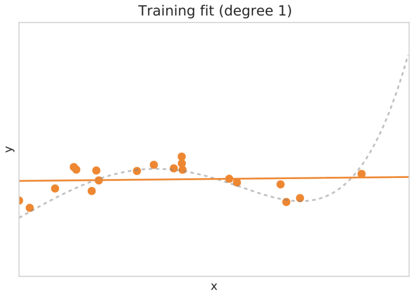
If we consider a high-degree polynomial:

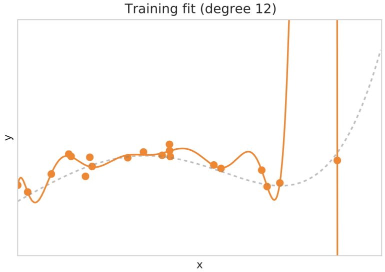


- We can find a model that fits the data well
- Is it a good predictor?
- No: Large generalization error

Motivation example: 1D-regression

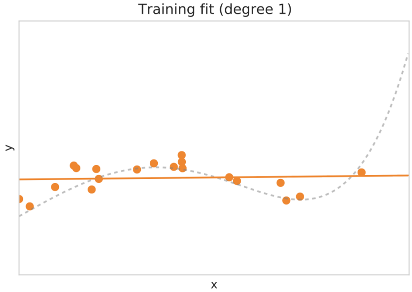
Simple models are less sensitive.

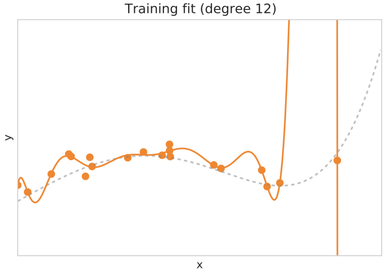




Motivation example: 1D-regression

Simple models are less sensitive.

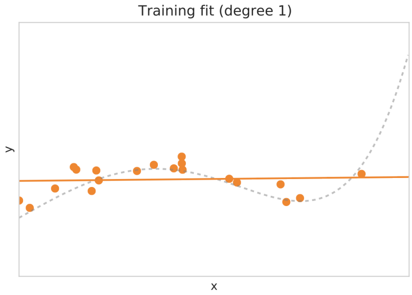


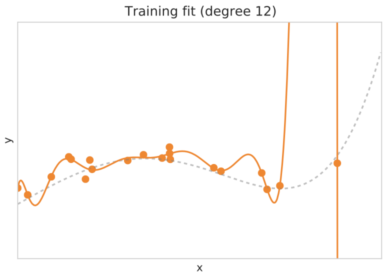


- moving a single observation will cause only a small shift in the position of the line

Motivation example: 1D-regression

Simple models are less sensitive.

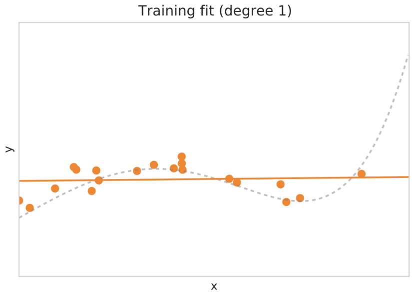


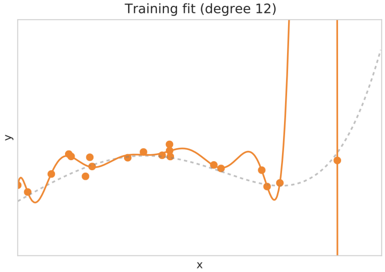


- moving a single observation will cause only a small shift in the position of the line
- under-fitting

Motivation example: 1D-regression

Simple models are less sensitive.



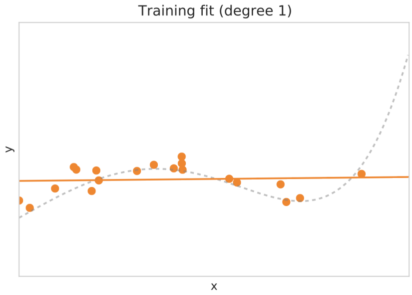


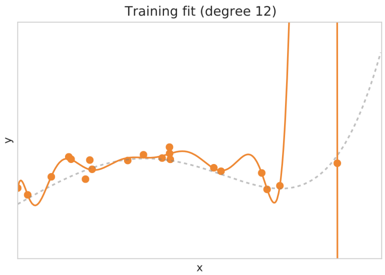
- moving a single observation will cause only a small shift in the position of the line
- under-fitting

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

Motivation example: 1D-regression

Simple models are less sensitive.



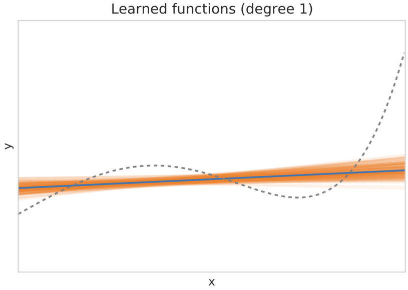


- moving a single observation will cause only a small shift in the position of the line
- under-fitting

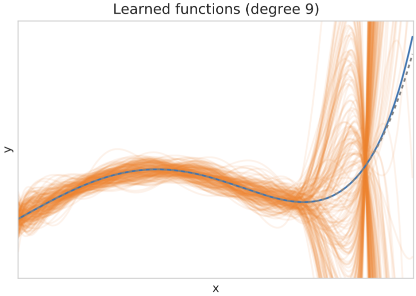
- Changing one of the data points may change the prediction considerable
- over-fitting

Motivation example: 1D-regression

Simple models have large biases but low variance.

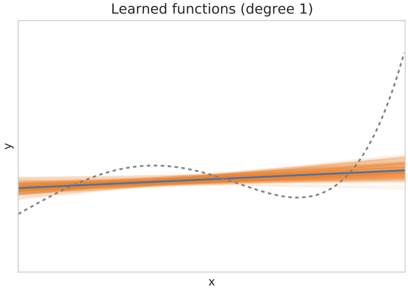


Complex models have low bias but high variance.

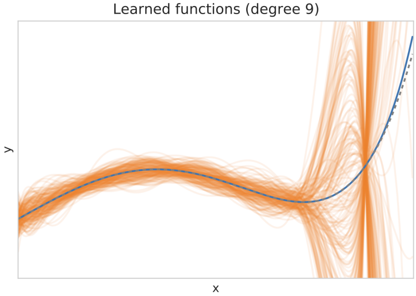


Motivation example: 1D-regression

Simple models have large biases but low variance.



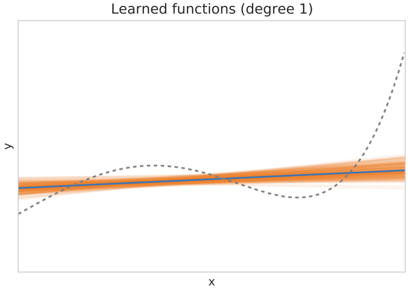
Complex models have low bias but high variance.



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

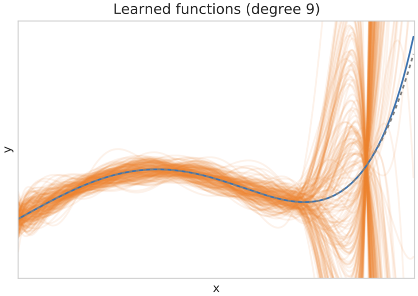
Motivation example: 1D-regression

Simple models have large biases but low variance.



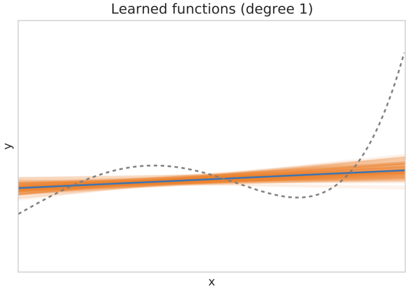
- large bias: the average of the predictions f_S does not fit well the data
- small variance: the variance of the predictions f_S as a function of S is small

Complex models have low bias but high variance.



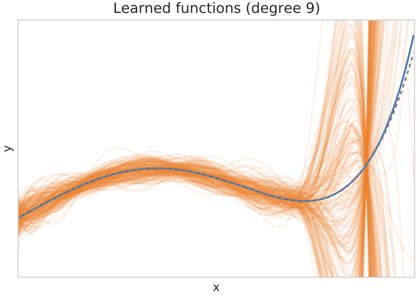
Motivation example: 1D-regression

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

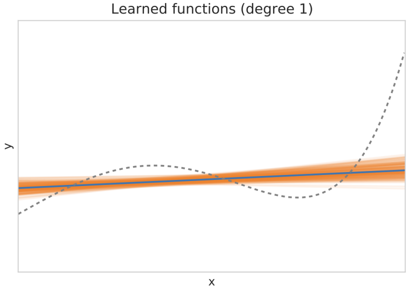
Complex models have low bias but high variance.



- small bias

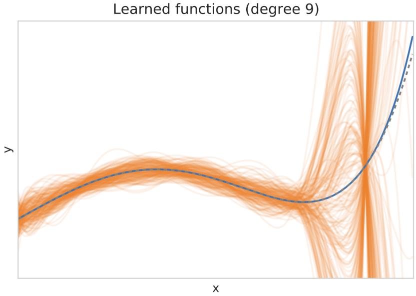
Motivation example: 1D-regression

Simple models have large biases but low variance.



- 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.



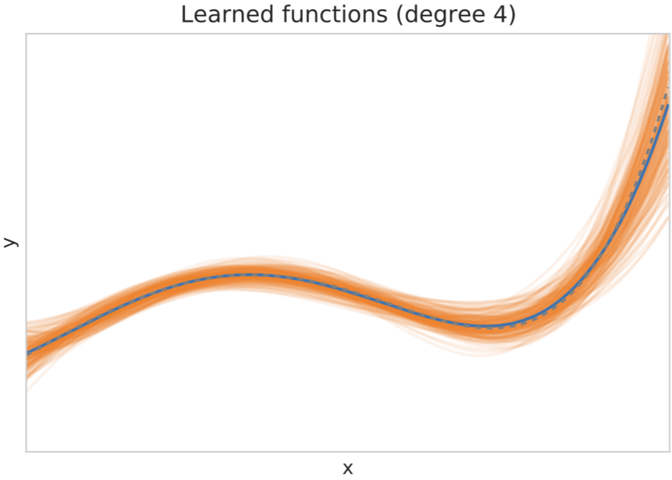
- small bias
- large variance

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

$$\boxed{y} = \boxed{f}(\boxed{x}) + \boxed{\epsilon} \tag{21}$$

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.

$$y = f(x) + \epsilon \tag{21}$$

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.

$$y = f(x) + \epsilon \tag{21}$$

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.

$$y = f(x) + \epsilon \tag{21}$$

- **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.

$$\boxed{y} = \boxed{f}(\boxed{x}) + \boxed{\boldsymbol{\epsilon}} \tag{21}$$

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

Error decomposition

Error decomposition

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

Error decomposition

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

Error decomposition

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

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:

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[(y - f_S(\mathbf{x}))^2\right] \tag{22}$$

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:

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[(y - f_S(\mathbf{x}))^2\right] \tag{22}$$

behaves as a function of

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:

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[(y - f_S(\mathbf{x}))^2\right] \tag{22}$$

behaves as a function of

1. the train set S

Error decomposition

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

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

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\left(y - f_S(\mathbf{x})\right)^2\right] \tag{22}$$

  behaves as a function of

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

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:

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\left(y - f_S(\mathbf{x})\right)^2\right] \tag{22}$$

  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

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

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

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[(y - f_S(\mathbf{x}))^2\right] \tag{22}$$

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 $\mathbf{x}_0$:

$$L(f_{S_{\text{train}}}) = \mathbb{E}_{\epsilon\sim\mathcal{D}_\epsilon}\left[(f(\mathbf{x}_0) + \epsilon - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{23}$$

A decomposition in three terms

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

$$\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] \tag{24}$$

(25)

(26)

A decomposition in three terms

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

$$\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] \tag{24}$$

$$:= \mathbb{E}_{S_{\text{train}} \in \mathcal{D}, \boldsymbol{\epsilon} \in \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\left(f(\mathbf{x}_0) + \boldsymbol{\epsilon} - \boxed{f_{S_{\text{train}}}}(\mathbf{x}_0) \right)^2 \right] \tag{25}$$

$$\tag{26}$$

A decomposition in three terms

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

$$\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] \tag{24}$$

$$:= \mathbb{E}_{S_{\text{train}} \in \mathcal{D}, \boldsymbol{\epsilon} \in \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\left(f(\mathbf{x}_0) + \boldsymbol{\epsilon} - f_{S_{\text{train}}}(\mathbf{x}_0) \right)^2 \right] \tag{25}$$

$$= \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\boldsymbol{\epsilon}^2 \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D},\, \boldsymbol{\epsilon} \sim \mathcal{D}_{\boldsymbol{\epsilon}}} \left[2\boldsymbol{\epsilon}(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0)) \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2 \right] \tag{26}$$

A decomposition in three terms

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

$$
\begin{aligned}
&\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] && (24) \\
&:= \mathbb{E}_{S_{\text{train}} \in \mathcal{D}, \boldsymbol{\epsilon} \in \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\left(f(\mathbf{x}_0) + \boldsymbol{\epsilon} - f_{S_{\text{train}}}(\mathbf{x}_0) \right)^2 \right] && (25) \\
&= \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\boldsymbol{\epsilon}^2 \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D},\, \boldsymbol{\epsilon} \sim \mathcal{D}_{\boldsymbol{\epsilon}}} \left[2\boldsymbol{\epsilon}(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0)) \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2 \right] && (26)
\end{aligned}
$$

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

$$
\begin{aligned}
&\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] && (24) \\
&:= \mathbb{E}_{S_{\text{train}} \in \mathcal{D}, \boldsymbol{\epsilon} \in \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\left(f(\mathbf{x}_0) + \boldsymbol{\epsilon} - \boxed{f_{S_{\text{train}}}}(\mathbf{x}_0) \right)^2 \right] && (25) \\
&= \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\boldsymbol{\epsilon}^2 \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D},\, \boldsymbol{\epsilon} \sim \mathcal{D}_{\boldsymbol{\epsilon}}} \left[2\boldsymbol{\epsilon}(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0)) \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2 \right] && (26)
\end{aligned}
$$

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

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

A decomposition in three terms

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

$$\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] \tag{24}$$

$$:= \mathbb{E}_{S_{\text{train}} \in \mathcal{D}, \boldsymbol{\epsilon} \in \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\left(f(\mathbf{x}_0) + \boldsymbol{\epsilon} - \boxed{f_{S_{\text{train}}}}(\mathbf{x}_0) \right)^2 \right] \tag{25}$$

$$= \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{D}_{\boldsymbol{\epsilon}}} \left[\boldsymbol{\epsilon}^2 \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D},\, \boldsymbol{\epsilon} \sim \mathcal{D}_{\boldsymbol{\epsilon}}} \left[2\boldsymbol{\epsilon}(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0)) \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2 \right] \tag{26}$$

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

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

A decomposition in three terms

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

$$\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] \tag{24}$$

$$:= \mathbb{E}_{S_{\text{train}} \in \mathcal{D}, \epsilon \in \mathcal{D}_\epsilon} \left[\left(f(\mathbf{x}_0) + \epsilon - f_{S_{\text{train}}}(\mathbf{x}_0) \right)^2 \right] \tag{25}$$

$$= \mathbb{E}_{\epsilon \sim \mathcal{D}_\epsilon} \left[\epsilon^2 \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D},\, \epsilon \sim \mathcal{D}_\epsilon} \left[2\epsilon (f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0)) \right] + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2 \right] \tag{26}$$

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

$$\mathbb{E}_{S_{\text{train}} \in \mathcal{D}} \left[L(f_{S_{\text{train}}}) \right] = \underbrace{\text{Var}_{\epsilon \in \mathcal{D}_\epsilon} [\epsilon]}_{\text{noise variance}} + \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2 \right] \tag{27}$$

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:

$$\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{28}$$

$$= \mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(\qquad\qquad + \qquad\qquad)^2\right] \tag{29}$$

(31)

(33)

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:

$$\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{28}$$

$$= \mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] + \qquad)^2\right] \tag{29}$$

$$\tag{31}$$

$$\tag{33}$$

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:

$$\mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2 \right] \tag{28}$$

$$= \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}} \left[f_{S_{\text{train'}}}(\mathbf{x}_0) \right] + \mathbb{E}_{S_{\text{train'}}} \left[f_{S_{\text{train'}}}(\mathbf{x}_0) \right] - f_{S_{\text{train}}}(\mathbf{x}_0) \right)^2 \right] \tag{29}$$

(31)

(33)

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:

$$\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{28}$$

$$= \mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] + \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{29}$$

$$= \mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\qquad\qquad\qquad\qquad\qquad\right] \tag{30}$$

$$\tag{31}$$

$$\tag{33}$$

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:

$$\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{28}$$

$$= \mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] + \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] \tag{29}$$

$$= \mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]\right)^2 \qquad\qquad \right] \tag{30}$$

$$\tag{31}$$

$$\tag{33}$$

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:

$$
\begin{aligned}
&\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] && (28)\\
&=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]+\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0))^{2}\right] && (29)\\
&=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\left(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]\right)^2+\left(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]-f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right] && (30)\\
& && (31)
\end{aligned}
$$

$$(33)$$

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:

$$
\begin{aligned}
&\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] && (28)\\
&=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}[f_{S_{\text{train'}}}(\mathbf{x}_0)]+\mathbb{E}_{S_{\text{train'}}}[f_{S_{\text{train'}}}(\mathbf{x}_0)]-f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] && (29)\\
&=\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\left(f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}[f_{S_{\text{train'}}}(\mathbf{x}_0)]\right)^2+\left(\mathbb{E}_{S_{\text{train'}}}[f_{S_{\text{train'}}}(\mathbf{x}_0)]-f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right] && (30)\\
&\quad\underbrace{+\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[2\left((f(\mathbf{x}_0)-\mathbb{E}_{S_{\text{train'}}}[f_{S_{\text{train'}}}(\mathbf{x}_0)]\right)\left(\mathbb{E}_{S_{\text{train'}}}[f_{S_{\text{train'}}}(\mathbf{x}_0)]-f_{S_{\text{train}}}(\mathbf{x}_0)\right))\right]}_{0} && (31)\\
& && (33)
\end{aligned}
$$

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:

$$
\begin{aligned}
&\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] && (28)\\
&= \mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] + \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0))^2\right] && (29)\\
&= \mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]\right)^2 + \left(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right] && (30)\\
&\qquad \underbrace{+\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[2\left(\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]\right)\left(\mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0)\right)\right)\right]}_{0} && (31)\\
&= \underbrace{\mathbb{E}_{S_{\text{train}}\sim\mathcal{D}}\left[\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}}\left[f_{S_{\text{train'}}}(\mathbf{x}_0)\right]\right)^2\right]}_{\text{bias}} && (32)\\
& && (33)
\end{aligned}
$$

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:

$$
\begin{aligned}
&\mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[(f(\mathbf{x}_0) - f_{S_{\text{train}}}(\mathbf{x}_0))^2 \right] && (28)\\
&= \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}} \left[f_{S_{\text{train'}}}(\mathbf{x}_0) \right] + \mathbb{E}_{S_{\text{train'}}} \left[f_{S_{\text{train'}}}(\mathbf{x}_0) \right] - f_{S_{\text{train}}}(\mathbf{x}_0))^2 \right] && (29)\\
&= \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}} \left[f_{S_{\text{train'}}}(\mathbf{x}_0) \right]\right)^2 + \left(\mathbb{E}_{S_{\text{train'}}} \left[f_{S_{\text{train'}}}(\mathbf{x}_0) \right] - f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2 \right] && (30)\\
&\quad \underbrace{+ \mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[2 \left(\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}} \left[f_{S_{\text{train'}}}(\mathbf{x}_0) \right]\right) \left(\mathbb{E}_{S_{\text{train'}}} \left[f_{S_{\text{train'}}}(\mathbf{x}_0) \right] - f_{S_{\text{train}}}(\mathbf{x}_0) \right) \right) \right]}_{0} && (31)\\
&= \underbrace{\mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train'}}} \left[f_{S_{\text{train'}}}(\mathbf{x}_0) \right]\right)^2 \right]}_{\text{bias}} && (32)\\
&\quad + \underbrace{\mathbb{E}_{S_{\text{train}} \sim \mathcal{D}} \left[\left(\mathbb{E}_{S_{\text{train'}}} \left[f_{S_{\text{train'}}}(\mathbf{x}_0) \right] - f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2 \right]}_{\text{variance}} && (33)
\end{aligned}
$$

Bias-Variance Decomposition

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

Bias-Variance Decomposition

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

which always lowers bound the true error.

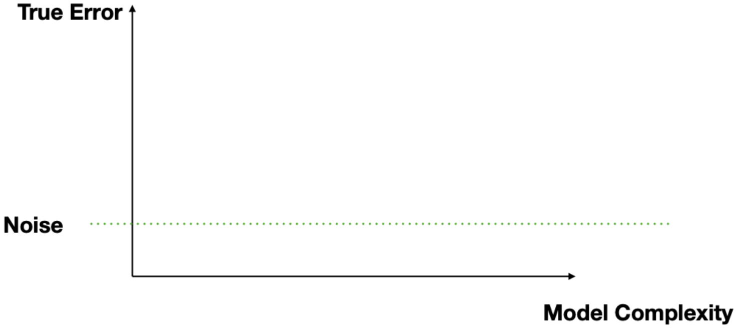
Bias-Variance Decomposition

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

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

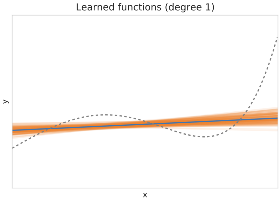
Component 1: Noise $\text{Var}_{\epsilon\sim\mathcal{D}_\epsilon}[\epsilon]$—a strict lower bound on what error we can achieve

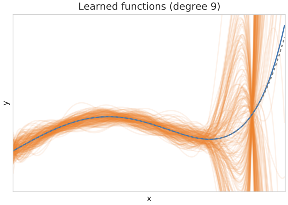
- It is not possible to go below the noise level
- Even if we know the true model f, we still suffer from $L(f) = \mathbb{E}\left[\epsilon^2\right]$

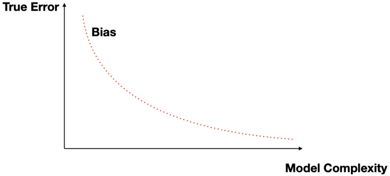
Component 1: Noise $\text{Var}_{\epsilon \sim \mathcal{D}_\epsilon}[\epsilon]$—a strict lower bound on what error we can achieve

- 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

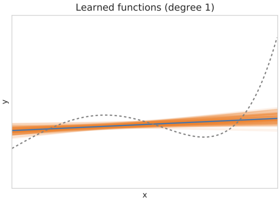
Component 2: Bias $\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train}}}\left[f_{S_{\text{train}}}(\mathbf{x}_0)\right]\right)^2$

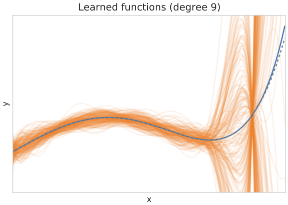


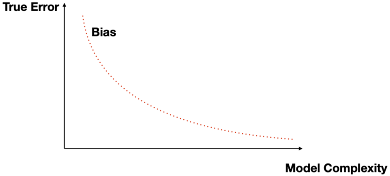




Component 2: Bias $\left(f(\mathbf{x}_0) - \mathbb{E}_{S_{\text{train}'}}\left[f_{S_{\text{train}'}}(\mathbf{x}_0)\right]\right)^2$

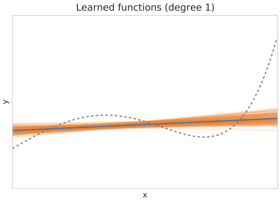


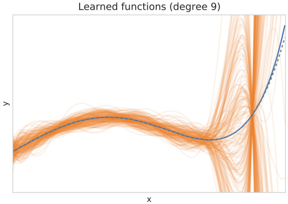


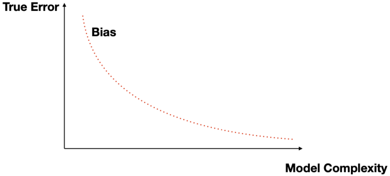


- It measures how far off in general the models' predictions are from the correct value.

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

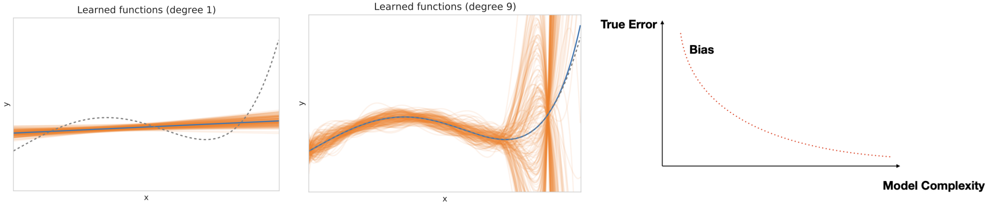






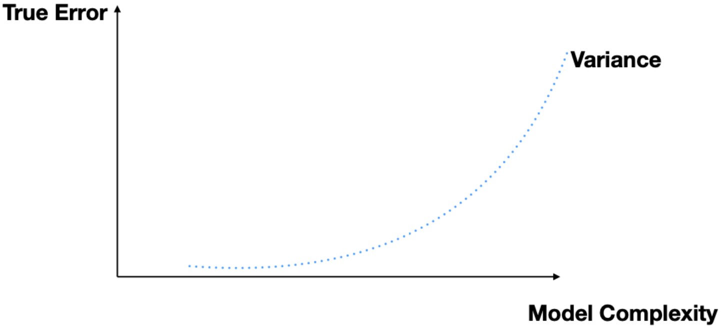
- It measures how far off in general the models' predictions are from the correct value.
- If complexity is small, then high bias.

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

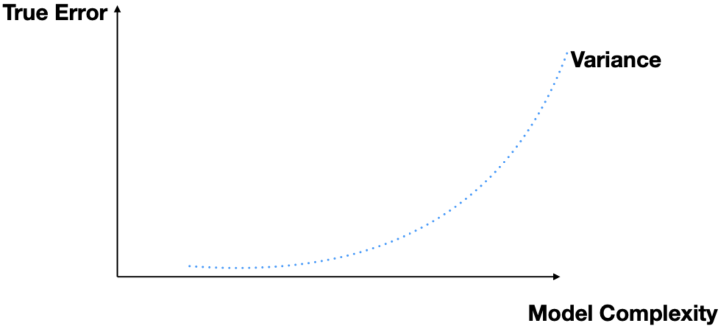


- It measures how far off in general the models' predictions are from the correct value.
- If complexity is small, then high bias.
- If complexity is high, then low bias.

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

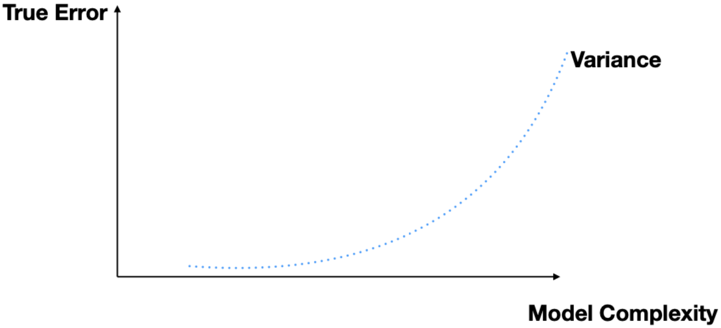


Component 3: Variance $\mathbb{E}_{S_{\text{train}} \sim \mathcal{D}}\left[\left(\mathbb{E}_{S_{\text{train}'}}\left[f_{S_{\text{train}'}}(\mathbf{x}_0)\right] - f_{S_{\text{train}}}(\mathbf{x}_0)\right)^2\right]$



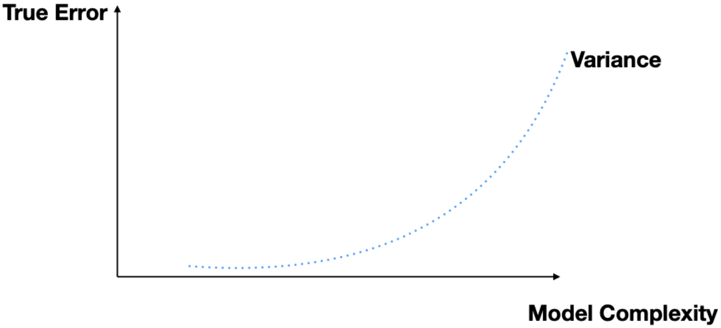
- Variance of the prediction function.

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



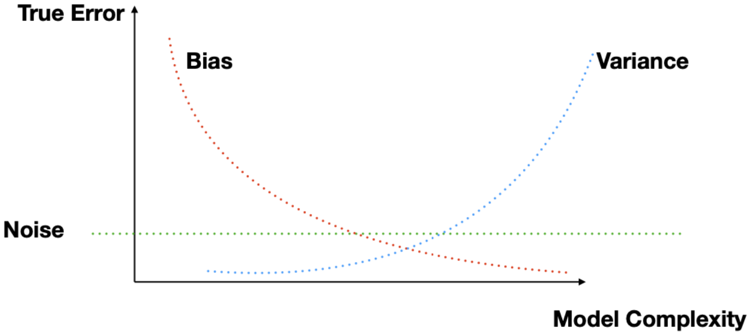
- Variance of the prediction function.
- It is how much the predictions for a given point vary between different realizations of the training set.

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

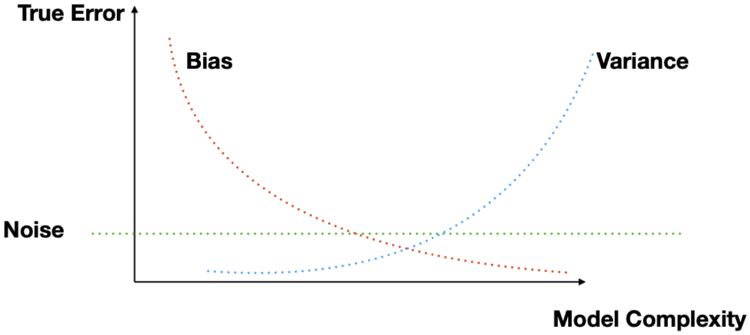


- Variance of the prediction function.
- It is how much the predictions for a given point vary between different realizations of the training set.
- If we consider complicated models, then small variations in the training set can result in large changes in the prediction.

Bias Variance tradeoff and U-shape curve

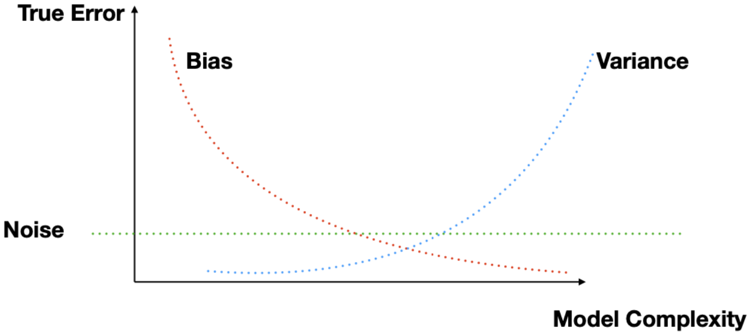


Bias Variance tradeoff and U-shape curve

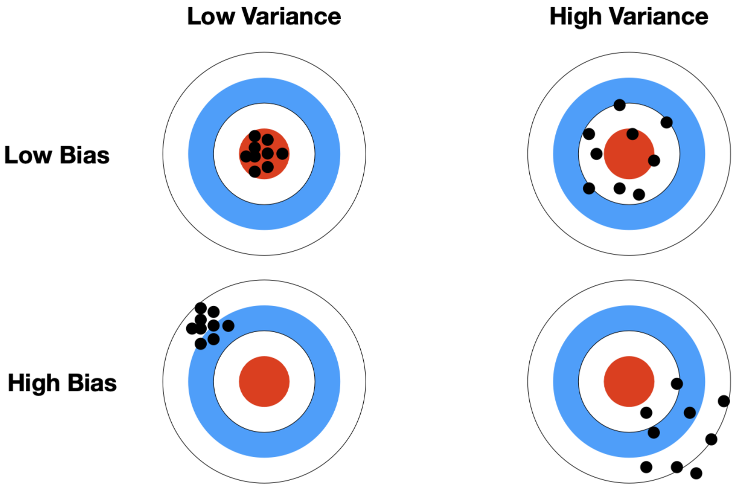


- If the complexity is too low, you cannot approximate well (under-fitting)

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

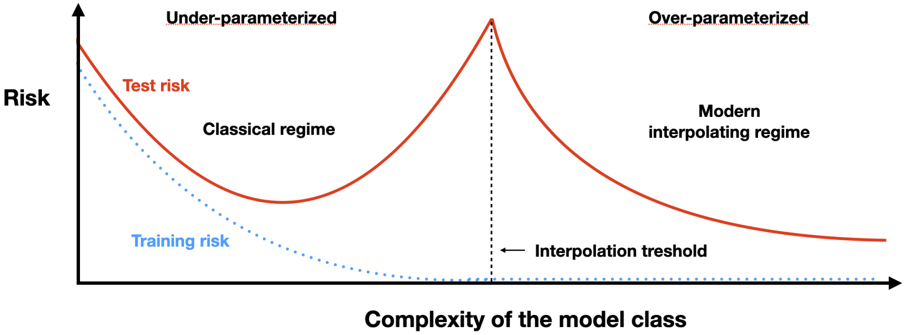


Table of Contents

Table of Contents

Classifier

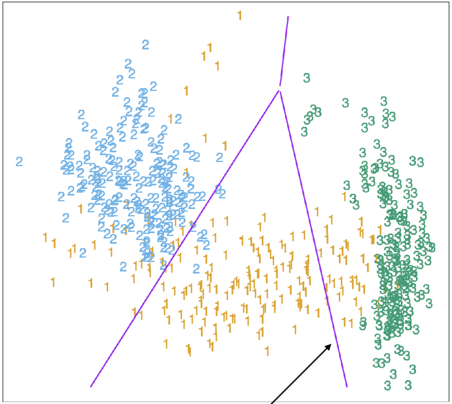
A classifier $f : \mathcal{X} \to \mathcal{Y}$

Classifier

A classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$ divides the input space into a collection of regions belonging to each class.

Classifier

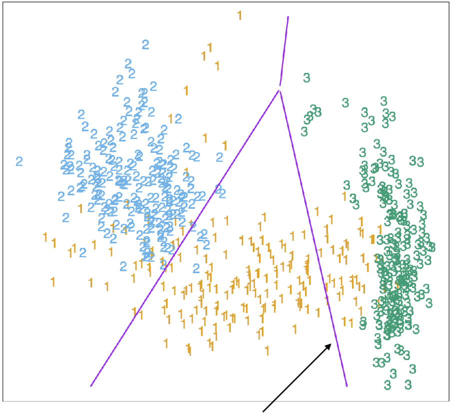
A classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$ divides the input space into a collection of regions belonging to each class.



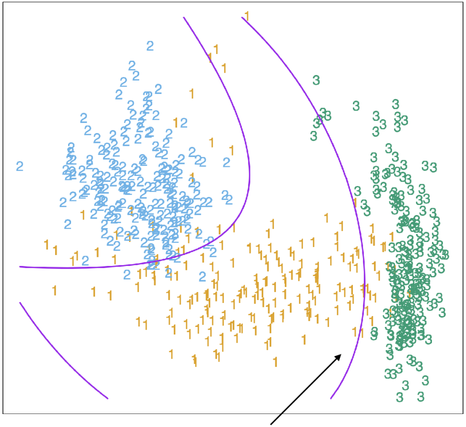
Linear Decision boundary

Classifier

A classifier $f : \mathcal{X} \to \mathcal{Y}$ divides the input space into a collection of regions belonging to each class.



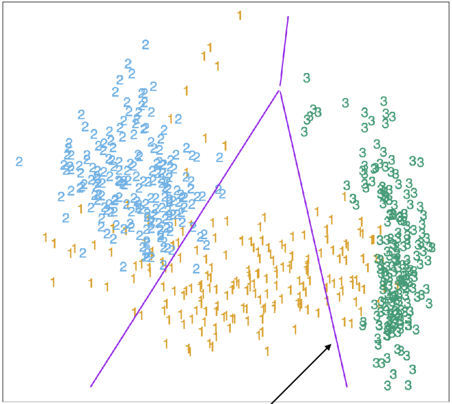
Linear Decision boundary



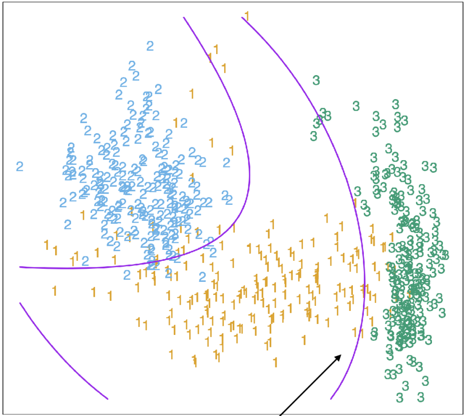
Nonlinear Decision boundary

Classifier

A classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$ divides the input space into a collection of regions belonging to each class.



Linear Decision boundary



Nonlinear Decision boundary

The boundaries of these regions are called decision boundaries.

Classification: a special case of regression?

Classification is a **regression problem** with discrete labels:

$$(\mathbf{x}, y) \in \mathcal{X} \times \{0, 1\} \subset \mathcal{X} \times \mathbb{R} \tag{34}$$

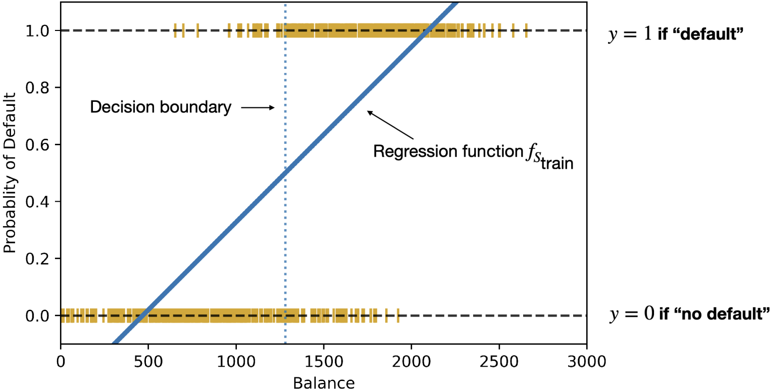
Classification: a special case of regression?

Classification is a **regression problem** with discrete labels:

$$(\mathbf{x}, y) \in \mathcal{X} \times \{0, 1\} \subset \mathcal{X} \times \mathbb{R} \tag{34}$$

Could we use previously seen regression methods to solve it?

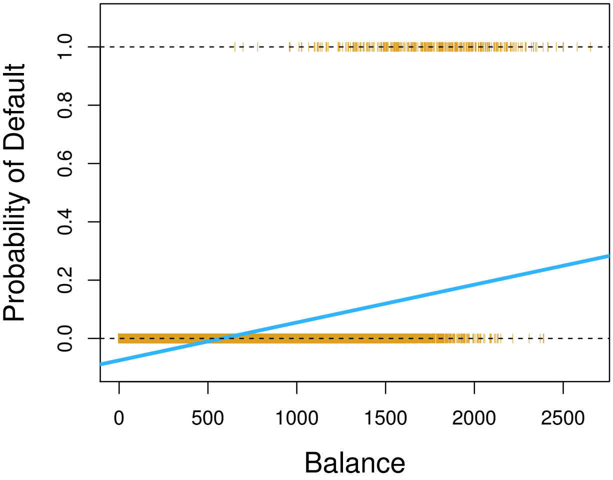
Is it a good idea to use some regression methods?



We label the output as probability for sake of interpretation

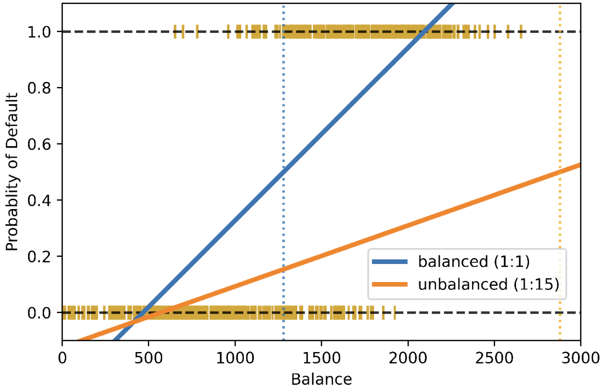
Classification is not just a special form of regression

- The predicted values are not probabilities (not in $[0, 1]$)



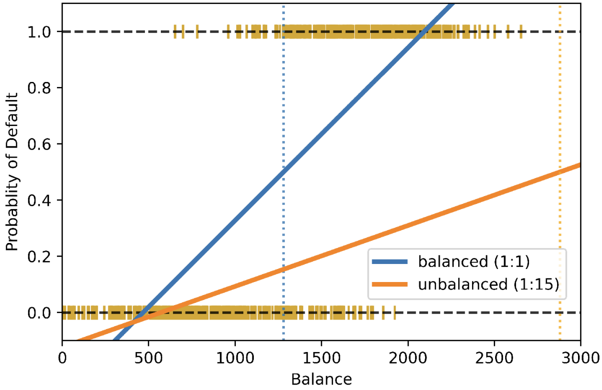
Classification is not just a special form of regression

- Sensitivity to unbalanced data



Classification is not just a special form of regression

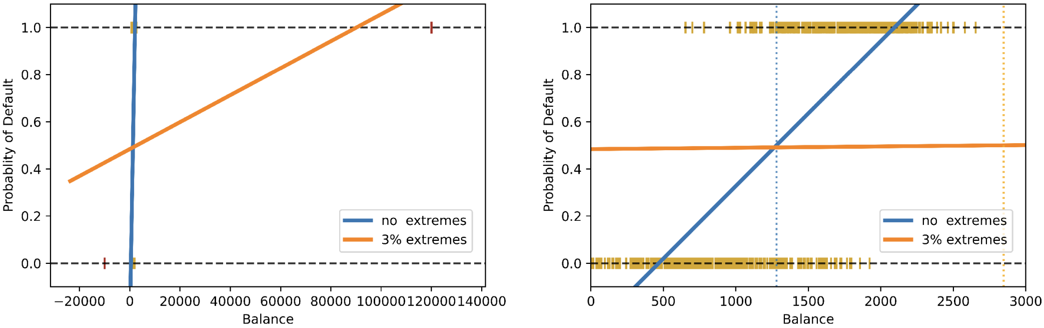
- Sensitivity to unbalanced data



The position of the line depends crucially on how many points are in each class.

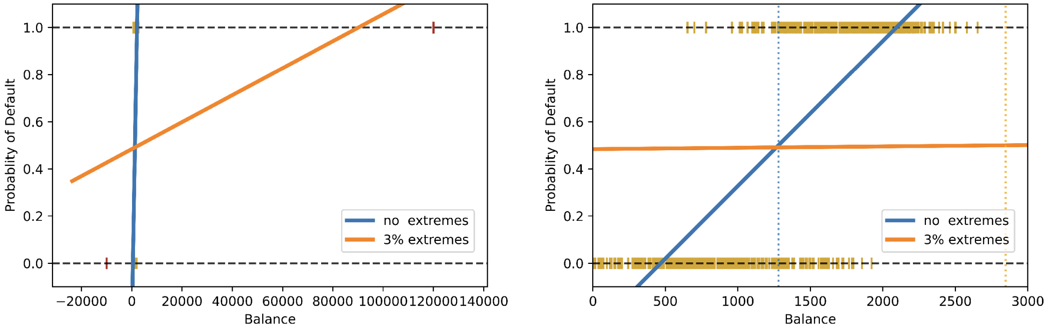
Classification is not just a special form of regression

- Sensitivity to extreme values:



Classification is not just a special form of regression

- Sensitivity to extreme values:



The position of the line depends crucially on where the points lie.

Optimal classification for known generating model

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

Optimal classification for known generating model

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.

Optimal classification for known generating model

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP):

Optimal classification for known generating model

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP): If we want to maximize the probability of guessing the correct label,

Optimal classification for known generating model

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP): If we want to maximize the probability of guessing the correct label, then we should choose the decision rule

Optimal classification for known generating model

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP): If we want to maximize the probability of guessing the correct label, then we should choose the decision rule

$$\hat{y}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}) \tag{35}$$

Optimal classification for known generating model

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP): If we want to maximize the probability of guessing the correct label, then we should choose the decision rule

$$\hat{y}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}) \tag{35}$$

  This classifier is also called the *Bayes classifier*.

Optimal classification for known generating model

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP): If we want to maximize the probability of guessing the correct label, then we should choose the decision rule

$$\hat{y}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}) \tag{35}$$

  This classifier is also called the *Bayes classifier*.

- In practice, we do not know the joint distribution $p(\mathbf{x}, y)$,

Optimal classification for known generating model

Assume that we know the joint distribution $p(\mathbf{x}, y)$.

- For a given input $\mathbf{x}$, *the probability that the "correct" label is* y is $p(y|\mathbf{x})$.
- Maximum A-Posteriori (MAP): If we want to maximize the probability of guessing the correct label, then we should choose the decision rule

$$\hat{y}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}) \tag{35}$$

  This classifier is also called the *Bayes classifier*.

- In practice, we do not know the joint distribution $p(\mathbf{x}, y)$, but we can use the data to learn the distribution (by assuming the data distribution).

Table of Contents

Motivation for Logistic Regression

Rather than modeling the output Y directly,

Motivation for Logistic Regression

Rather than modeling the output Y directly,
we can **model the probability** that Y belongs to a particular class.

Motivation for Logistic Regression

Rather than modeling the output Y directly,
we can **model the probability** that Y belongs to a particular class.

Previously, we used a linear regression model $\Pr(Y = 1|X = x) = \mathbf{x}^\top \mathbf{w} + w_0$, but

Motivation for Logistic Regression

Rather than modeling the output Y directly,
we can **model the probability** that Y belongs to a particular class.

Previously, we used a linear regression model $\Pr(Y = 1|X = x) = \mathbf{x}^\top \mathbf{w} + w_0$, but

- The predicted value is not in $[0, 1]$.

Motivation for Logistic Regression

Rather than modeling the output Y directly,
we can **model the probability** that Y belongs to a particular class.

Previously, we used a linear regression model $\Pr(Y = 1|X = x) = \mathbf{x}^\top \mathbf{w} + w_0$, but

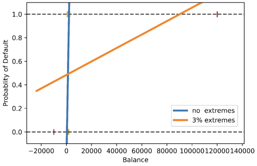
- The predicted value is not in $[0, 1]$.
- Very large ($y \gg 1$) or very small ($y \ll 0$) values of the prediction will contribute to the error if we use the squared loss.

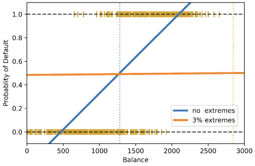
Motivation for Logistic Regression

Rather than modeling the output Y directly,
we can **model the probability** that Y belongs to a particular class.

Previously, we used a linear regression model $\Pr(Y = 1|X = x) = \mathbf{x}^\top \mathbf{w} + w_0$, but

- The predicted value is not in $[0, 1]$.
- Very large ($y \gg 1$) or very small ($y \ll 0$) values of the prediction will contribute to the error if we use the squared loss.



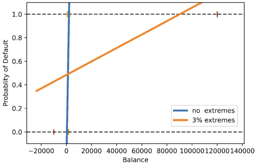


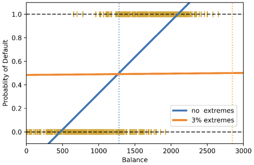
Motivation for Logistic Regression

Rather than modeling the output Y directly,
we can **model the probability** that Y belongs to a particular class.

Previously, we used a linear regression model $\Pr(Y = 1|X = x) = \mathbf{x}^\top \mathbf{w} + w_0$, but

- The predicted value is not in $[0, 1]$.
- Very large ($y \gg 1$) or very small ($y \ll 0$) values of the prediction will contribute to the error if we use the squared loss.

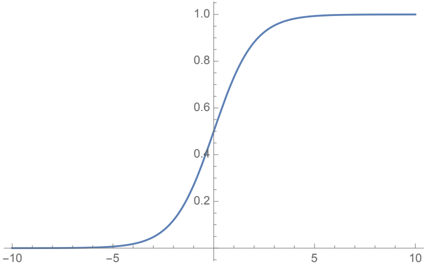




Solution: Transforming the predictions that take values in $(-\infty, \infty)$ into $[0, 1]$.

The logistic function

Consider first of all the case of two classes.

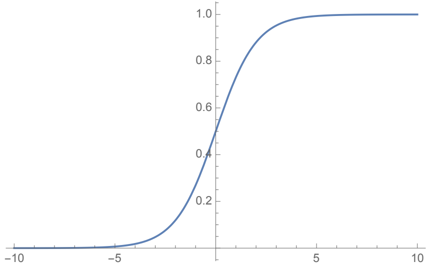


The logistic function

Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (36)$$

(37)

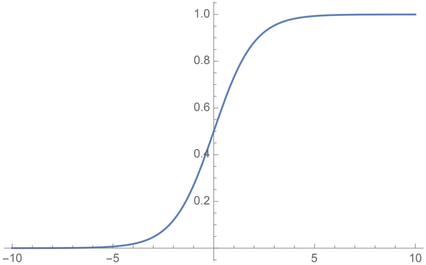


The logistic function

Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (36)$$

$$= \frac{1}{1 + \exp(-\eta)} \quad (37)$$

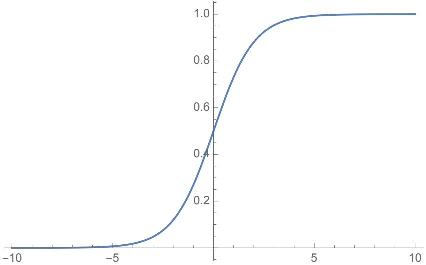


The logistic function

Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \tag{36}$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \tag{37}$$



The logistic function

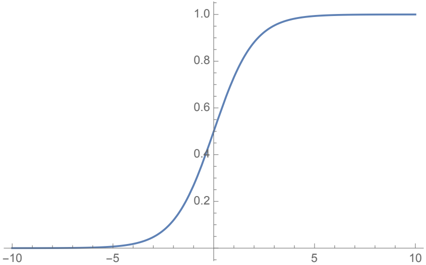
Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (36)$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \quad (37)$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (38)$$



The logistic function

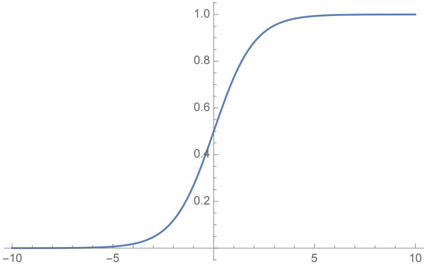
Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (36)$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \quad (37)$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^\eta}{1+e^\eta} \quad (38)$$



The logistic function

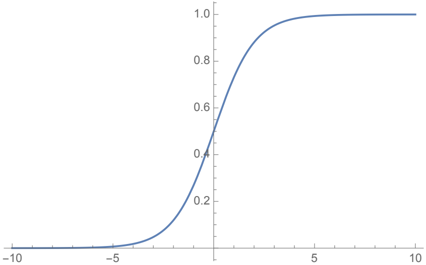
Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (36)$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \quad (37)$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^\eta}{1+e^\eta} \quad (38)$$



Properties of the logistic function:

The logistic function

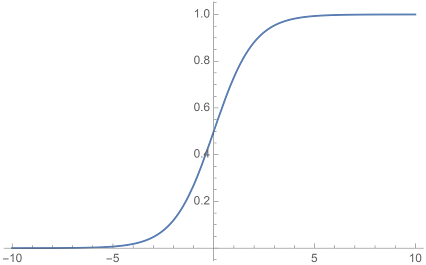
Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (36)$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \quad (37)$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^\eta}{1+e^\eta} \quad (38)$$



Properties of the logistic function:

- $1 - \sigma(\eta) = \sigma(-\eta)$

The logistic function

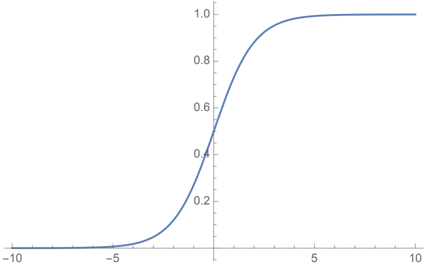
Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \tag{36}$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \tag{37}$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^{\eta}}{1+e^{\eta}} \tag{38}$$



Properties of the logistic function:

- $1 - \sigma(\eta) = \sigma(-\eta)$
- $\sigma'(\eta) = \sigma(\eta)\,(1 - \sigma(\eta))$

The logistic function

Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

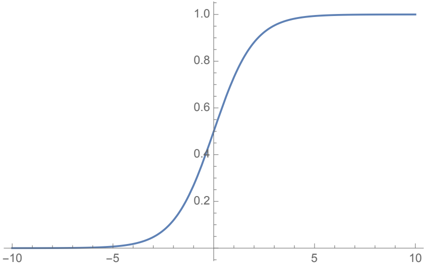
$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (36)$$

$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \quad (37)$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^\eta}{1+e^\eta} \quad (38)$$

For the case of $K > 2$ classes, we have



Properties of the logistic function:

- $1 - \sigma(\eta) = \sigma(-\eta)$
- $\sigma'(\eta) = \sigma(\eta)\,(1 - \sigma(\eta))$

The logistic function

Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (36)$$

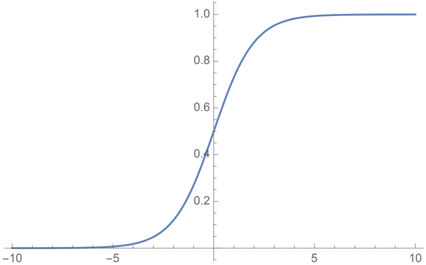
$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \quad (37)$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^\eta}{1+e^\eta} \quad (38)$$

For the case of $K > 2$ classes, we have

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} \quad (39)$$



Properties of the logistic function:

- $1 - \sigma(\eta) = \sigma(-\eta)$
- $\sigma'(\eta) = \sigma(\eta)\,(1-\sigma(\eta))$

The logistic function

Consider first of all the case of two classes.
The posterior probability for class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (36)$$

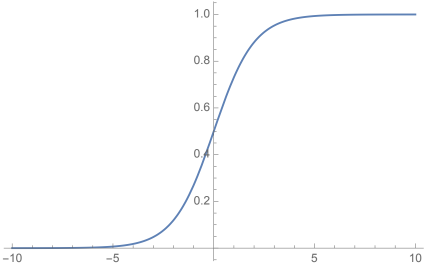
$$= \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \quad (37)$$

where we have defined

$$\eta = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ and } \sigma(\eta) := \frac{e^\eta}{1+e^\eta} \quad (38)$$

For the case of $K > 2$ classes, we have

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} = \frac{\exp(\eta_k)}{\sum_j \exp(\eta_j)} \quad (39)$$



Properties of the logistic function:

- $1 - \sigma(\eta) = \sigma(-\eta)$
- $\sigma'(\eta) = \sigma(\eta)\,(1 - \sigma(\eta))$

Logistic Regression

Given a “new” feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr\left[Y = 1|\mathbf{X} = \mathbf{x}\right] = \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \tag{40}$$

$$p(0|\mathbf{x}) := \Pr\left[Y = 0|\mathbf{X} = \mathbf{x}\right] = 1 - \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \ , \tag{41}$$

Logistic Regression

Given a “new” feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr\left[Y = 1|\mathbf{X} = \mathbf{x}\right] = \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \tag{40}$$

$$p(0|\mathbf{x}) := \Pr\left[Y = 0|\mathbf{X} = \mathbf{x}\right] = 1 - \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right)\ , \tag{41}$$

where we predict a real value (a probability) and not a label.

Logistic Regression

Given a "new" feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr\left[Y = 1|\mathbf{X} = \mathbf{x}\right] = \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \tag{40}$$

$$p(0|\mathbf{x}) := \Pr\left[Y = 0|\mathbf{X} = \mathbf{x}\right] = 1 - \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) , \tag{41}$$

where we predict a real value (a probability) and not a label.

Label prediction: quantize the probability

$$\text{if } p(1|\mathbf{x}) \geq 1/2 \Rightarrow \text{predict the class } 1 \tag{42}$$

$$\text{if } p(1|\mathbf{x}) < 1/2 \Rightarrow \text{predict the class } 0 \tag{43}$$

Logistic Regression

Given a "new" feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr\left[Y = 1|\mathbf{X} = \mathbf{x}\right] = \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \tag{40}$$

$$p(0|\mathbf{x}) := \Pr\left[Y = 0|\mathbf{X} = \mathbf{x}\right] = 1 - \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \ , \tag{41}$$

where we predict a real value (a probability) and not a label.

Label prediction: quantize the probability

$$\text{if } p(1|\mathbf{x}) \geq 1/2 \Rightarrow \text{predict the class } 1 \tag{42}$$

$$\text{if } p(1|\mathbf{x}) < 1/2 \Rightarrow \text{predict the class } 0 \tag{43}$$

Interpretation:

Logistic Regression

Given a “new” feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr\left[Y = 1|\mathbf{X} = \mathbf{x}\right] = \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \tag{40}$$

$$p(0|\mathbf{x}) := \Pr\left[Y = 0|\mathbf{X} = \mathbf{x}\right] = 1 - \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \ , \tag{41}$$

where we predict a real value (a probability) and not a label.

Label prediction: quantize the probability

$$\text{if } p(1|\mathbf{x}) \geq 1/2 \Rightarrow \text{predict the class } 1 \tag{42}$$

$$\text{if } p(1|\mathbf{x}) < 1/2 \Rightarrow \text{predict the class } 0 \tag{43}$$

Interpretation:

- Very large $\mathbf{x}^\top \mathbf{w} + w_0$ corresponds to $p(1|\mathbf{x})$ very close to 0 or 1 (high confidence).

Logistic Regression

Given a "new" feature vector $\mathbf{x}$, we predict the (posterior) probability of the two class labels given $\mathbf{x}$ by means of

$$p(1|\mathbf{x}) := \Pr\left[Y = 1|\mathbf{X} = \mathbf{x}\right] = \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right) \tag{40}$$

$$p(0|\mathbf{x}) := \Pr\left[Y = 0|\mathbf{X} = \mathbf{x}\right] = 1 - \sigma\left(\mathbf{x}^\top \mathbf{w} + w_0\right)\ , \tag{41}$$

where we predict a real value (a probability) and not a label.

Label prediction: quantize the probability

$$\text{if } p(1|\mathbf{x}) \geq 1/2 \Rightarrow \text{predict the class } 1 \tag{42}$$

$$\text{if } p(1|\mathbf{x}) < 1/2 \Rightarrow \text{predict the class } 0 \tag{43}$$

Interpretation:

- Very large $\mathbf{x}^\top \mathbf{w} + w_0$ corresponds to $p(1|\mathbf{x})$ very close to 0 or 1 (high confidence).
- Small $\left|\mathbf{x}^\top \mathbf{w} + w_0\right|$ corresponds to $p(1|\mathbf{x})$ very close to 0.5 (low confidence).

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^{N}$ (fixed but unknown $\mathcal{D}$).

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^{N}$ (fixed but unknown $\mathcal{D}$).

The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \tag{44}$$

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^{N}$ (fixed but unknown $\mathcal{D}$).

The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := p(\{\mathbf{x}_n, y_n\}_{n=1}^{N} | \mathbf{w}) \tag{44}$$

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^N$ (fixed but unknown $\mathcal{D}$).

The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := p(\{\mathbf{x}_n, y_n\}_{n=1}^N|\mathbf{w}) = \prod_{n=1}^N p(\{\mathbf{x}_n, y_n\}|\mathbf{w}) \,, \tag{44}$$

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^N$ (fixed but unknown $\mathcal{D}$).

The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := p(\{\mathbf{x}_n, y_n\}_{n=1}^N|\mathbf{w}) = \prod_{n=1}^N p(\{\mathbf{x}_n, y_n\}|\mathbf{w}) \,, \tag{44}$$

or equivalently,

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} [-\log \mathcal{L}(\mathbf{w})] = \arg\min \sum_{n=1}^N -\log \left(p(\{\mathbf{x}_n, y_n\}|\mathbf{w}) \right) \,. \tag{45}$$

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^N$ (fixed but unknown $\mathcal{D}$).

The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := p(\{\mathbf{x}_n, y_n\}_{n=1}^N | \mathbf{w}) = \prod_{n=1}^N p(\{\mathbf{x}_n, y_n\} | \mathbf{w}) \,, \tag{44}$$

or equivalently,

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} [-\log \mathcal{L}(\mathbf{w})] = \arg\min \sum_{n=1}^N -\log \left(p(\{\mathbf{x}_n, y_n\} | \mathbf{w}) \right) \,. \tag{45}$$

This estimator is **consistent** (under mild condition):

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^N$ (fixed but unknown $\mathcal{D}$).

The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := p(\{\mathbf{x}_n, y_n\}_{n=1}^N | \mathbf{w}) = \prod_{n=1}^N p(\{\mathbf{x}_n, y_n\}|\mathbf{w}) \,, \tag{44}$$

or equivalently,

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} [-\log \mathcal{L}(\mathbf{w})] = \arg\min \sum_{n=1}^N -\log\left(p(\{\mathbf{x}_n, y_n\}|\mathbf{w})\right) \,. \tag{45}$$

This estimator is **consistent** (under mild condition):
$\Rightarrow$ if the data are generated according to the model,

MLE is a method of estimating the parameters of a statistical model

Assume a training set S_{train}, consisting of i.i.d. samples $\{\mathbf{x}_n, y_n\}_{n=1}^N$ (fixed but unknown $\mathcal{D}$).

The MLE finds the parameters $\mathbf{w}^\star$ under which $\{\mathbf{x}_n, y_n\}$ are the most likely:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := p(\{\mathbf{x}_n, y_n\}_{n=1}^N | \mathbf{w}) = \prod_{n=1}^N p(\{\mathbf{x}_n, y_n\} | \mathbf{w}) \,, \tag{44}$$

or equivalently,

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} [-\log \mathcal{L}(\mathbf{w})] = \arg\min \sum_{n=1}^N - \log \left(p(\{\mathbf{x}_n, y_n\} | \mathbf{w}) \right) \,. \tag{45}$$

This estimator is **consistent** (under mild condition):
$\Rightarrow$ if the data are generated according to the model,
the MLE converges to the true parameter when $N \to \infty$.

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

$$p(\mathbf{y}, \mathbf{X}|\mathbf{w}) = p(\mathbf{X}|\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = p(\mathbf{X})p(\mathbf{y}|\mathbf{X}, \mathbf{w})\,, \tag{46}$$

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

$$p(\mathbf{y}, \mathbf{X}|\mathbf{w}) = p(\mathbf{X}|\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = p(\mathbf{X})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \,, \tag{46}$$

where $\mathbf{X}$ does not depend on $\mathbf{w}$, i.e., $p(\mathbf{W})$ is a constant w.r.t. arbitrary $\mathbf{w}$.

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

$$p(\mathbf{y}, \mathbf{X}|\mathbf{w}) = p(\mathbf{X}|\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = p(\mathbf{X})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \,, \tag{46}$$

where $\mathbf{X}$ does not depend on $\mathbf{w}$, i.e., $p(\mathbf{W})$ is a constant w.r.t. arbitrary $\mathbf{w}$.

For Linear Regression, we have:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n)$$

$$\tag{48}$$

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

$$p(\mathbf{y}, \mathbf{X}|\mathbf{w}) = p(\mathbf{X}|\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = p(\mathbf{X})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \,, \tag{46}$$

where $\mathbf{X}$ does not depend on $\mathbf{w}$, i.e., $p(\mathbf{W})$ is a constant w.r.t. arbitrary $\mathbf{w}$.

For Linear Regression, we have:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n) = \prod_{n:y_n=1} p(y_n = 1|\mathbf{x}_n) \prod_{n:y_n=0} p(y_n = 0|\mathbf{x}_n) \tag{47}$$

$$\tag{48}$$

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

$$p(\mathbf{y}, \mathbf{X}|\mathbf{w}) = p(\mathbf{X}|\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = p(\mathbf{X})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \,, \tag{46}$$

where $\mathbf{X}$ does not depend on $\mathbf{w}$, i.e., $p(\mathbf{W})$ is a constant w.r.t. arbitrary $\mathbf{w}$.

For Linear Regression, we have:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n) = \prod_{n:y_n=1} p(y_n = 1|\mathbf{x}_n) \prod_{n:y_n=0} p(y_n = 0|\mathbf{x}_n) \tag{47}$$

$$= \prod_{n=1}^{N} \sigma(\mathbf{x}_n^\top \mathbf{w})^{y_n} \left[1 - \sigma(\mathbf{x}_n^\top \mathbf{w})\right]^{1-y_n} \tag{48}$$

MLE for Logistic Regression

The likelihood of the data $\{\mathbf{y}, \mathbf{X}\}$ given the parameter $\mathbf{w}$, i.e., $p(\mathbf{y}, \mathbf{X}|\mathbf{w})$.

$$p(\mathbf{y}, \mathbf{X}|\mathbf{w}) = p(\mathbf{X}|\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = p(\mathbf{X})p(\mathbf{y}|\mathbf{X}, \mathbf{w})\,, \tag{46}$$

where $\mathbf{X}$ does not depend on $\mathbf{w}$, i.e., $p(\mathbf{W})$ is a constant w.r.t. arbitrary $\mathbf{w}$.

For Linear Regression, we have:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n) = \prod_{n:y_n=1} p(y_n = 1|\mathbf{x}_n) \prod_{n:y_n=0} p(y_n = 0|\mathbf{x}_n) \tag{47}$$

$$= \prod_{n=1}^{N} \sigma(\mathbf{x}_n^\top \mathbf{w})^{y_n} \left[1 - \sigma(\mathbf{x}_n^\top \mathbf{w})\right]^{1-y_n} \tag{48}$$

As a result,

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \arg\min_{\mathbf{w}} \left(- \log p(\mathbf{y}|\mathbf{X}, \mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \right) \tag{49}$$

Gradient of the negative log likelihood

Recall that

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \left(\mathcal{L}(\mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \right) \tag{50}$$

Gradient of the negative log likelihood

Recall that

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \left(\mathcal{L}(\mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \right) \tag{50}$$

Let's minimize $\mathcal{L}(\mathbf{w})$ through the property of stationary points.

$$\nabla \mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \left(\sigma(\mathbf{x}_n^\top \mathbf{w}) - y_n \right) \tag{51}$$

$$\tag{52}$$

Gradient of the negative log likelihood

Recall that

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \left(\mathcal{L}(\mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \right) \tag{50}$$

Let's minimize $\mathcal{L}(\mathbf{w})$ through the property of stationary points.

$$\begin{aligned} \nabla \mathcal{L}(\mathbf{w}) &= \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \left(\sigma(\mathbf{x}_n^\top \mathbf{w}) - y_n \right) && (51) \\ &= \frac{1}{N} \mathbf{X}^\top \left[\sigma(\mathbf{X}\mathbf{w}) - \mathbf{y} \right] , && (52) \end{aligned}$$

Gradient of the negative log likelihood

Recall that

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \left(\mathcal{L}(\mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \right) \tag{50}$$

Let's minimize $\mathcal{L}(\mathbf{w})$ through the property of stationary points.

$$\begin{aligned} \nabla \mathcal{L}(\mathbf{w}) &= \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \left(\sigma(\mathbf{x}_n^\top \mathbf{w}) - y_n \right) && (51) \\ &= \frac{1}{N} \mathbf{X}^\top \left[\sigma(\mathbf{X}\mathbf{w}) - \mathbf{y} \right], && (52) \end{aligned}$$

where $\mathbf{X} \in \mathbb{R}^{N \times d}$.

Gradient of the negative log likelihood

Recall that

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \left(\mathcal{L}(\mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n^\top \mathbf{w} + \log(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) \right) \tag{50}$$

Let's minimize $\mathcal{L}(\mathbf{w})$ through the property of stationary points.

$$\nabla \mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \left(\sigma(\mathbf{x}_n^\top \mathbf{w}) - y_n \right) \tag{51}$$

$$= \frac{1}{N} \mathbf{X}^\top \left[\sigma(\mathbf{X}\mathbf{w}) - \mathbf{y} \right] , \tag{52}$$

where $\mathbf{X} \in \mathbb{R}^{N \times d}$.

$\Rightarrow$ It has no closed-form solution to $\nabla \mathcal{L}(\mathbf{w}) = 0$.

This lecture:

- Generalization Gap and Model Selection
- Bias-Variance Decomposition
- Before Introducing Multilayer Perceptron: Logistic Regression

This lecture:

- Generalization Gap and Model Selection
- Bias-Variance Decomposition
- Before Introducing Multilayer Perceptron: Logistic Regression

Next lecture:

- Exponential Families and Generalized Linear Models
- Multi-Layer Perceptron
- Back-Propagation
- Introduction to Deep Learning Optimization