Learning Theory & Regularization

Shan-Hung Wu

shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

Machine Learning
Outline

1. Learning Theory

2. Point Estimation: Bias and Variance
   - Consistency*

3. Decomposing Generalization Error

4. Regularization
   - Weight Decay
   - Validation
Outline

1 Learning Theory

2 Point Estimation: Bias and Variance
   • Consistency*

3 Decomposing Generalization Error

4 Regularization
   • Weight Decay
   • Validation
Which Polynomial Degree Is Better? I

- Given a training set \( X = \{(x^{(i)}, y^{(i)})\}_{i=1}^N \) i.i.d. sampled from of \( P(x, y) \)
- Assume \( P(x, y) = P(y|x)P(x) \), where
  - \( P(x) \sim \text{Uniform}(-1, 1) \)
  - \( y = \sin(\pi x) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2) \)
Which Polynomial Degree Is Better? II

Consider 3 unregularized polynomial regressors of degrees $P = 1$, $3$, and $10$.

Which one would you pick?

Note that $P = 10$ has zero training error. Any $N$ points can be perfectly fitted by a polynomial of degree $N - 1$. 

Shan-Hung Wu (CS, NTHU) Learning Theory & Regularization Machine Learning 5 / 44
Consider 3 unregularized polynomial regressors of degrees $P = 1, 3, \text{ and } 10$.

Which one would you pick? Probably not $P = 1$ nor $P = 10$. 

Note that $P = 10$ has zero training error. Any $N$ points can be perfectly fitted by a polynomial of degree $N - 1$. 

Shan-Hung Wu (CS, NTHU) 
Learning Theory & Regularization 
Machine Learning 5 / 44
Consider 3 unregularized polynomial regressors of degrees $P = 1$, 3, and 10

Which one would you pick? Probably not $P = 1$ nor $P = 10$

Note that $P = 10$ has zero training error

- Any $N$ points can be perfectly fitted by a polynomial of degree $N - 1$
Empirical Error vs. Generalization Error

- In ML, we usually “learn” a function by minimizing the *empirical error/risk* defined over a training set of size $N$:

$$C_N(w) \text{ or } C_N[f] = \frac{1}{N} \sum_{i=1}^{N} \text{loss} \left( f(x^{(i)}; w), y^{(i)} \right)$$

- E.g., $C_N(w) = \frac{1}{2} \sum_{i=1}^{N} \left( y^{(i)} - w^\top x^{(i)} \right)^2$ in linear regression
Empirical Error vs. Generalization Error

- In ML, we usually “learn” a function by minimizing the empirical error/risk defined over a training set of size $N$:

$$C_N(w) \text{ or } C_N[f] = \frac{1}{N} \sum_{i=1}^{N} \text{loss} \left( f(x^{(i)}; w), y^{(i)} \right)$$

- E.g., $C_N(w) = \frac{1}{2} \sum_{i=1}^{N} \left( y^{(i)} - w^\top x^{(i)} \right)^2$ in linear regression

- But our goal is to have a low generalization error/risk defined over the underlying data distribution:

$$C(w) \text{ or } C[f] = \int \text{loss} \left( f(x; w), y \right) dP(x, y)$$

- Can be estimated by the testing error

$$C_{N'}(w) = \frac{1}{N'} \sum_{i=1}^{N'} \text{loss} \left( f(x'^{(i)}; w), y'^{(i)} \right)$$ defined over the testing set

$\mathbb{X}' = \{ (x'^{(i)}, y'^{(i)}) \}_{i=1}^{N'}$
**Empirical Error vs. Generalization Error**

- In ML, we usually “learn” a function by minimizing the *empirical error/risk* defined over a training set of size $N$:

  $$ C_N(w) \text{ or } C_N[f] = \frac{1}{N} \sum_{i=1}^{N} \text{loss} \left( f(x^{(i)}; w), y^{(i)} \right) $$

- E.g., $C_N(w) = \frac{1}{2} \sum_{i=1}^{N} \left( y^{(i)} - w^\top x^{(i)} \right)^2$ in linear regression

- But our goal is to have a low *generalization error/risk* defined over the underlying data distribution:

  $$ C(w) \text{ or } C[f] = \int \text{loss} \left( f(x; w), y \right) dP(x, y) $$

- Can be estimated by the *testing error*

  $$ C_N'(w) = \frac{1}{N'} \sum_{i=1}^{N'} \text{loss} \left( f(x'^{(i)}; w), y'^{(i)} \right) \text{ defined over the testing set} $$

  $$ \mathbb{X}' = \{(x'^{(i)}, y'^{(i)})\}_{i=1}^{N'} $$

- Does a low $C_N[f]$ implies low $C[f]$?
Empirical Error vs. Generalization Error

- In ML, we usually “learn” a function by minimizing the *empirical error/risk* defined over a training set of size $N$:

\[
C_N(w) \text{ or } C_N[f] = \frac{1}{N} \sum_{i=1}^{N} \text{loss} \left( f(x^{(i)}; w), y^{(i)} \right)
\]

- E.g., $C_N(w) = \frac{1}{2} \sum_{i=1}^{N} \left( y^{(i)} - w^\top x^{(i)} \right)^2$ in linear regression

- But our goal is to have a low *generalization error/risk* defined over the underlying data distribution:

\[
C(w) \text{ or } C[f] = \int \text{loss} \left( f(x; w), y \right) dP(x, y)
\]

- Can be estimated by the *testing error*

\[
C_N'(w) = \frac{1}{N'} \sum_{i=1}^{N'} \text{loss} \left( f(x'^{(i)}; w), y'^{(i)} \right)
\]

\[
\mathbb{X}' = \{(x'^{(i)}, y'^{(i)})\}_{i=1}^{N'}
\]

- Does a low $C_N[f]$ implies low $C[f]$? No, as $P = 10$ indicates
No-Free-Lunch Theorem

- Why $C[f]$ is defined over a *particular* data generating distribution $P$?
No-Free-Lunch Theorem

Why $C[f]$ is defined over a particular data generating distribution $P$?

Theorem (No-Free-Lunch Theorem [4])

Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying unseen points.
No-Free-Lunch Theorem

- Why $C[f]$ is defined over a particular data generating distribution $P$?

**Theorem (No-Free-Lunch Theorem [4])**

Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying unseen points.

- No machine learning algorithm is better than any other universally
No-Free-Lunch Theorem

Why $C[f]$ is defined over a particular data generating distribution $P$?

Theorem (No-Free-Lunch Theorem [4])

Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying unseen points.

- No machine learning algorithm is better than any other universally
- The goal of ML is not to seek a universally good learning algorithm
- Instead, a good algorithm that performs well on data drawn from a particular $P$ we care about
Learning Theory

Let $f^* = \arg\min_f C[f]$ be the best possible function we can get.
Learning Theory

- Let \( f^* = \arg\min_f C[f] \) be the best possible function we can get.
- Since we are seeking a prediction function in a model (hypothesis space) \( \mathcal{F} \), this is what we can have at best: \( f_{\mathcal{F}}^* = \arg\min_{f \in \mathcal{F}} C[f] \)

Learning theory: how to characterize \( C[f_{\mathcal{F}^N}] = \int \text{loss}(f_{\mathcal{F}^N}(x; w), y) dP(x, y) \)?

Not to confuse \( C[f_{\mathcal{F}^N}] \) with \( C_{\mathcal{F}^N}[f] \).

Bounding methods
Decomposition methods
Learning Theory

- Let $f^* = \arg\min_f C[f]$ be the best possible function we can get.
- Since we are seeking a prediction function in a model (hypothesis space) $F$, this is what we can have at best: $f^*_F = \arg\min_{f \in F} C[f]$.
- But we only minimize empirical errors on limited examples of size $N$, this is what we actually have $f_N = \arg\min_{f \in F} C_N[f]$.
  - Ignoring numerical errors (due to, e.g., numerical optimization).
Learning Theory

- Let \( f^* = \arg \min f \mathcal{C}[f] \) be the best possible function we can get.
- Since we are seeking a prediction function in a model (hypothesis space) \( \mathbb{F} \), this is what can have at best: \( f^*_\mathbb{F} = \arg \min_{f \in \mathbb{F}} \mathcal{C}[f] \).
- But we only minimize empirical errors on limited examples of size \( N \), this is what we actually have: \( f_N = \arg \min_{f \in \mathbb{F}} \mathcal{C}_N[f] \).
  - Ignoring numerical errors (due to, e.g., numerical optimization).
- **Learning theory**: how to characterize

\[
\mathcal{C}[f_N] = \int \text{loss}(f_N(x; w), y) dP(x, y)\
\]

- Not to confuse \( \mathcal{C}[f_N] \) with \( \mathcal{C}_N[f] \).
Learning Theory

- Let $f^* = \arg \min_f C[f]$ be the best possible function we can get.
- Since we are seeking a prediction function in a model (hypothesis space) $\mathbb{F}$, this is what can have at best: $f^*_\mathbb{F} = \arg \min_{f \in \mathbb{F}} C[f]$
- But we only minimizes empirical errors on limited examples of size $N$, this is what we actually have $f_N = \arg \min_{f \in \mathbb{F}} C_N[f]$
  - Ignoring numerical errors (due to, e.g., numerical optimization)
- **Learning theory**: how to characterize $C[f_N] = \int \text{loss}(f_N(x; w), y) dP(x, y)$?
  - Not to confuse $C[f_N]$ with $C_N[f]$
- Bounding methods
- Decomposition methods
Bounding Methods I

- \( \min_f C[f] = C[f^*] \) is called the **Bayes error**
  - Larger than 0 when there is randomness in \( P(y|x) \)
  - E.g., in our regression problem: \( y = f^*(x; w) + \epsilon, \ \epsilon \sim \mathcal{N}(0, \sigma^2) \)

![Graph showing regression function and data points](image-url)
Bounding Methods 1

- \( \min_f C[f] = C[f^*] \) is called the \textbf{Bayes error}
  - Larger than 0 when there is randomness in \( P(y|x) \)
  - E.g., in our regression problem: \( y = f^*(x; w) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2) \)
  - \textbf{Cannot be avoided} even we know \( P(x, y) \) in the ground truth
Bounding Methods I

- \( \min_f C[f] = C[f^*] \) is called the **Bayes error**
  - Larger than 0 when there is randomness in \( P(y|x) \)
  - E.g., in our regression problem: \( y = f^*(x; w) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2) \)

- **Cannot be avoided** even we know \( P(x, y) \) in the ground truth

- So, our target is to make \( C[f_N] \) as close to \( C[f^*] \) as possible
Let $\mathcal{E} = C[f_N] - C[f^*]$ be the *excess error*.

We have

$$\mathcal{E} = C[f^*_F] - C[f^*] + C[f_N] - C[f^*_F]$$

Where $\mathcal{E}_{\text{app}}$ is called the *approximation error* and $\mathcal{E}_{\text{est}}$ is called the *estimation error*.

How to reduce these errors?

We can reduce $\mathcal{E}_{\text{app}}$ by choosing a more complex $F$. A complex $F$ has a larger capacity.

E.g., larger polynomial degree $P$ in polynomial regression.

How to reduce $\mathcal{E}_{\text{est}}$?
Bounding Methods II

- Let $\mathcal{E} = C[f_N] - C[f^*]$ be the excess error.

- We have

$$\mathcal{E} = C[f^*_F] - C[f^*] + C[f_N] - C[f^*_F]$$

  - $\mathcal{E}_{\text{app}}$ is called the approximation error.
  - $\mathcal{E}_{\text{est}}$ is called the estimation error.
Let $\mathcal{E} = C[f_N] - C[f^*]$ be the **excess error**.

We have

$$\mathcal{E} = C[f^*_F] - C[f^*] + C[f_N] - C[f^*_F]$$

- $\mathcal{E}_{\text{app}}$ is called the **approximation error**
- $\mathcal{E}_{\text{est}}$ is called the **estimation error**

How to reduce these errors?
Let $\mathcal{E} = C[f_N] - C[f^*]$ be the excess error.

We have

$$\mathcal{E} = \underbrace{C[f_F^*] - C[f^*]}_{\mathcal{E}_{\text{app}}} + \underbrace{C[f_N] - C[f_F^*]}_{\mathcal{E}_{\text{est}}}$$

$\mathcal{E}_{\text{app}}$ is called the approximation error.

$\mathcal{E}_{\text{est}}$ is called the estimation error.

How to reduce these errors?

We can reduce $\mathcal{E}_{\text{app}}$ by choosing a more complex $F$.

- A complex $F$ has a larger capacity.
- E.g., larger polynomial degree $P$ in polynomial regression.
Bounding Methods II

- Let $\mathcal{E} = C[f_N] - C[f^*]$ be the excess error
- We have
  \[ \mathcal{E} = C[f_F^*] - C[f^*] + C[f_N] - C[f_F^*] \]
  \[ \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} \]

- $\mathcal{E}_{\text{app}}$ is called the approximation error
- $\mathcal{E}_{\text{est}}$ is called the estimation error
- How to reduce these errors?
- We can reduce $\mathcal{E}_{\text{app}}$ by choosing a more complex $F$
  - A complex $F$ has a larger capacity
  - E.g., larger polynomial degree $P$ in polynomial regression
- How to reduce $\mathcal{E}_{\text{est}}$?
Bounding Methods III

- Bounds of $\mathcal{E}_{\text{est}}$ for, e.g., binary classifiers [1, 2, 3]:

$$\mathcal{E}_{\text{est}} = O \left[ \left( \frac{\text{Complexity}(F) \log N}{N} \right)^{\alpha} \right], \alpha \in \left[ \frac{1}{2}, 1 \right], \text{ with high probability}$$

- So, to reduce $\mathcal{E}_{\text{est}}$, we should either have
  - **Simpler model** (e.g., smaller polynomial degree $P$), or
  - Larger training set
Model Complexity, Overfit, and Underfit

- Too simple a model leads to high $\mathcal{E}_{\text{app}}$

- Too complex a model leads to high $\mathcal{E}_{\text{est}}$
Model Complexity, Overfit, and Underfit

- Too simple a model leads to high $\mathcal{E}_{\text{app}}$ due to **underfitting**
  - $f_N$ fails to capture the shape of $f^*$

- Too complex a model leads to high $\mathcal{E}_{\text{est}}$
Model Complexity, Overfit, and Underfit

- Too simple a model leads to high $\mathcal{E}_{\text{app}}$ due to **underfitting**
  - $f_N$ fails to capture the shape of $f^*$
  - *High* training error; *high* testing error (given a sufficiently large $N$)
- Too complex a model leads to high $\mathcal{E}_{\text{est}}$
Model Complexity, Overfit, and Underfit

- Too simple a model leads to high $\mathcal{E}_{\text{app}}$ due to *underfitting*
  - $f_N$ fails to capture the shape of $f^*$
  - **High** training error; **high** testing error (given a sufficiently large $N$)
- Too complex a model leads to high $\mathcal{E}_{\text{est}}$ due to *overfitting*
  - $f_N$ captures not only the shape of $f^*$ but also some spurious patterns (e.g., noise) local to a particular training set
Model Complexity, Overfit, and Underfit

- Too simple a model leads to high $\mathcal{E}_{\text{app}}$ due to underfitting
  - $f_N$ fails to capture the shape of $f^*$
  - High training error; high testing error (given a sufficiently large $N$)
- Too complex a model leads to high $\mathcal{E}_{\text{est}}$ due to overfitting
  - $f_N$ captures not only the shape of $f^*$ but also some spurious patterns (e.g., noise) local to a particular training set
  - Low training error; high testing error
Sample Complexity and Learning Curves

- How many training examples ($N$) are sufficient?
Sample Complexity and Learning Curves

- How many training examples ($N$) are sufficient?
- Different models/algorithms may have different *sample complexity*
  - I.e., the $N$ required to learn a target function with specified generalizability
Sample Complexity and Learning Curves

- How many training examples ($N$) are sufficient?
- Different models/algorithms may have different sample complexity
  - i.e., the $N$ required to learn a target function with specified generalizability
- Can be visualized using the learning curves

![Diagram showing learning curves and sample complexities](image)
Sample Complexity and Learning Curves

- How many training examples ($N$) are sufficient?
- Different models/algorithms may have different sample complexity
  - i.e., the $N$ required to learn a target function with specified generalizability
- Can be visualized using the learning curves
- Too small $N$ results in overfit regardless of model complexity

![Learning Curves Diagram]
Decomposition Methods

- Bounding methods analyze $C[f_N]$ qualitatively
  - General, as no (or weak) assumption on data distribution is made
Decomposition Methods

- Bounding methods analyze $C[f_N]$ qualitatively
  - General, as no (or weak) assumption on data distribution is made
- However, in practice, these bounds are too loose to quantify $C[f_N]$
Decomposition Methods

- Bounding methods analyze $C[f_N]$ qualitatively
  - General, as no (or weak) assumption on data distribution is made
- However, in practice, these bounds are too loose to quantify $C[f_N]$
- In some particular situations, we can decompose $C[f_N]$ into multiple meaningful terms
Decomposition Methods

- Bounding methods analyze $C[f_N]$ qualitatively
  - General, as no (or weak) assumption on data distribution is made
- However, in practice, these bounds are too loose to quantify $C[f_N]$
- In some particular situations, we can decompose $C[f_N]$ into multiple meaningful terms
- Assume particular
  - Loss function $\text{loss}(\cdot)$, and
  - Data generating distribution $P(x, y)$
Decomposition Methods

- Bounding methods analyze $C[f_N]$ qualitatively
  - General, as no (or weak) assumption on data distribution is made
- However, in practice, these bounds are too loose to quantify $C[f_N]$
- In some particular situations, we can decompose $C[f_N]$ into multiple meaningful terms
  - Assume particular
    - Loss function $\text{loss}(\cdot)$, and
    - Data generating distribution $P(x, y)$
- Require knowledge about the point estimation
Outline

1 Learning Theory

2 Point Estimation: Bias and Variance
   - Consistency*

3 Decomposing Generalization Error

4 Regularization
   - Weight Decay
   - Validation
Sample Mean and Variance

- **Point estimation** is the attempt to estimate some fixed but unknown quantity $\theta$ of a random variable by using sample data.
Sample Mean and Variance

- *Point estimation* is the attempt to estimate some fixed but unknown quantity $\theta$ of a random variable by using sample data.

- Let $X = \{x^{(1)}, \cdots, x^{(n)}\}$ be a set of $n$ i.i.d. samples of a random variable $x$, a *point estimator* or *statistic* is a function of the data:

  $$\hat{\theta}_n = g(x^{(1)}, \cdots, x^{(n)})$$

- The value $\hat{\theta}_n$ is called the *estimate* of $\theta$.
Sample Mean and Variance

- **Point estimation** is the attempt to estimate some fixed but unknown quantity $\theta$ of a random variable by using sample data.

- Let $X = \{x^{(1)}, \ldots, x^{(n)}\}$ be a set of $n$ i.i.d. samples of a random variable $x$, a **point estimator** or **statistic** is a function of the data:

  $$\hat{\theta}_n = g(x^{(1)}, \ldots, x^{(n)})$$

  - The value $\hat{\theta}_n$ is called the **estimate** of $\theta$.

- Sample mean: $\hat{\mu}_x = \frac{1}{n} \sum_i x^{(i)}$

- Sample variance: $\hat{\sigma}_x = \frac{1}{n} \sum_i (x^{(i)} - \hat{\mu}_x)^2$
Sample Mean and Variance

- **Point estimation** is the attempt to estimate some fixed but unknown quantity $\theta$ of a random variable by using sample data.
- Let $X = \{x^{(1)}, \ldots, x^{(n)}\}$ be a set of $n$ i.i.d. samples of a random variable $x$, a *point estimator* or *statistic* is a function of the data:

$$\hat{\theta}_n = g(x^{(1)}, \ldots, x^{(n)})$$

- The value $\hat{\theta}_n$ is called the *estimate* of $\theta$.
- Sample mean: $\hat{\mu}_x = \frac{1}{n} \sum_i x^{(i)}$
- Sample variance: $\hat{\sigma}_x = \frac{1}{n} \sum_i (x^{(i)} - \hat{\mu}_x)^2$
- How good are these estimators?
Bias & Variance

- **Bias** of an estimator:

\[
\text{bias}(\hat{\theta}_n) = \mathbb{E}_X(\hat{\theta}_n) - \theta
\]

- Here, the expectation is defined over *all possible* \(X\)’s of size \(n\), i.e.,
  \[
  \mathbb{E}_X(\hat{\theta}_n) = \int \hat{\theta}_n dP(X)
  \]

- We call a statistic *unbiased estimator* iff it has zero bias
Bias & Variance

- **Bias** of an estimator:

\[
\text{bias}(\hat{\theta}_n) = \mathbb{E}_X(\hat{\theta}_n) - \theta
\]

- Here, the expectation is defined over *all possible* \(X\)'s of size \(n\), i.e.,
\[
\mathbb{E}_X(\hat{\theta}_n) = \int \hat{\theta}_n dP(X)
\]

- We call a statistic *unbiased estimator* iff it has zero bias

- **Variance** of an estimator:

\[
\text{Var}_X(\hat{\theta}_n) = \mathbb{E}_X \left[ (\hat{\theta}_n - \mathbb{E}_X[\hat{\theta}_n])^2 \right]
\]
Bias & Variance

- **Bias** of an estimator:

\[
\text{bias}(\hat{\theta}_n) = E_X(\hat{\theta}_n) - \theta
\]

- Here, the expectation is defined over *all possible* \( X \)'s of size \( n \), i.e.,

\[
E_X(\hat{\theta}_n) = \int \hat{\theta}_n dP(X)
\]

- We call a statistic *unbiased estimator* iff it has zero bias

- **Variance** of an estimator:

\[
\text{Var}_X(\hat{\theta}_n) = E_X \left[ (\hat{\theta}_n - E_X[\hat{\theta}_n])^2 \right]
\]

- Is \( \hat{\mu}_x = \frac{1}{n} \sum_i x^{(i)} \) an unbiased estimator of \( \mu_x \)?
Bias & Variance

- **Bias** of an estimator:

\[
\text{bias}(\hat{\theta}_n) = E_X(\hat{\theta}_n) - \theta
\]

- Here, the expectation is defined over all possible X’s of size n, i.e.,

\[
E_X(\hat{\theta}_n) = \int \hat{\theta}_n dP(X)
\]

- We call a statistic *unbiased estimator* iff it has zero bias

- **Variance** of an estimator:

\[
\text{Var}_X(\hat{\theta}_n) = E_X \left[ (\hat{\theta}_n - E_X[\hat{\theta}_n])^2 \right]
\]

- Is \( \hat{\mu}_x = \frac{1}{n} \sum_i x^{(i)} \) an unbiased estimator of \( \mu_x \)? Yes [Homework]
Bias & Variance

- **Bias** of an estimator:
  \[ \text{bias}(\hat{\theta}_n) = E_{X}(\hat{\theta}_n) - \theta \]

  - Here, the expectation is defined over *all possible* \(X\)'s of size \(n\), i.e.,
    \[ E_{X}(\hat{\theta}_n) = \int \hat{\theta}_n dP(X) \]

  - We call a statistic **unbiased estimator** iff it has zero bias

- **Variance** of an estimator:
  \[ \text{Var}_{X}(\hat{\theta}_n) = E_{X} \left[ (\hat{\theta}_n - E_{X}[\hat{\theta}_n])^2 \right] \]

- Is \( \hat{\mu}_x = \frac{1}{n} \sum_i x^{(i)} \) an unbiased estimator of \( \mu_x \)? Yes [Homework]

- What much is \( \text{Var}_{X}(\hat{\mu}_x) \)?
Variance of $\hat{\mu}_x$

$$\text{Var}_x(\hat{\mu}) = E_x[(\hat{\mu} - E_x[\hat{\mu}])^2] = E[\hat{\mu}^2 - 2\hat{\mu}\mu + \mu^2] = E[\hat{\mu}^2] - \mu^2$$
Variance of $\hat{\mu}_X$

\[
\text{Var}_X(\hat{\mu}) = E_X[(\hat{\mu} - E_X[\hat{\mu}])^2] = E[\hat{\mu}^2 - 2\hat{\mu}\mu + \mu^2] = E[\hat{\mu}^2] - \mu^2 \\
= E\left[\frac{1}{n^2} \sum_{i,j} x^{(i)} x^{(j)}\right] - \mu^2 = \frac{1}{n^2} \sum_{i,j} E[x^{(i)} x^{(j)}] - \mu^2
\]
Variance of $\hat{\mu}_x$

\[
\text{Var}_x(\hat{\mu}) = E_x[(\hat{\mu} - E_x[\hat{\mu}])^2] = E[\hat{\mu}^2 - 2\hat{\mu}\mu + \mu^2] = E[\hat{\mu}^2] - \mu^2 \\
= E\left[\frac{1}{n^2} \sum_{i,j} x^{(i)}x^{(j)}\right] - \mu^2 = \frac{1}{n^2} \sum_{i,j} E[x^{(i)}x^{(j)}] - \mu^2 \\
= \frac{1}{n^2} \left(\sum_{i=j} E[x^{(i)}x^{(j)}] + \sum_{i\neq j} E[x^{(i)}x^{(j)}]\right) - \mu^2
\]
Variance of $\hat{\mu}_X$

$$\text{Var}_X(\hat{\mu}) = E_X[(\hat{\mu} - E_X[\hat{\mu}])^2] = E[\hat{\mu}^2 - 2\hat{\mu}\mu + \mu^2] = E[\hat{\mu}^2] - \mu^2$$

$$= E\left[\frac{1}{n^2} \sum_{i,j} x^{(i)} x^{(j)}\right] - \mu^2 = \frac{1}{n^2} \sum_{i,j} E[x^{(i)} x^{(j)}] - \mu^2$$

$$= \frac{1}{n^2} \left( \sum_{i=j} E[x^{(i)} x^{(j)}] + \sum_{i \neq j} E[x^{(i)} x^{(j)}] \right) - \mu^2$$

$$= \frac{1}{n^2} \left( \sum_i E[x^{(i)}]^2 + n(n-1)E[x^{(i)}]E[x^{(j)}] \right) - \mu^2$$
Variance of $\hat{\mu}_X$

$$\text{Var}_X(\hat{\mu}) = E_X[(\hat{\mu} - E_X[\hat{\mu}])^2] = E[\hat{\mu}^2 - 2\hat{\mu}\mu + \mu^2] = E[\hat{\mu}^2] - \mu^2$$
$$= E\left[\frac{1}{n^2} \sum_{i,j} x^{(i)} x^{(j)}\right] - \mu^2 = \frac{1}{n^2} \sum_{i,j} E[x^{(i)} x^{(j)}] - \mu^2$$
$$= \frac{1}{n^2} \left(\sum_{i=j} E[x^{(i)} x^{(j)}] + \sum_{i\neq j} E[x^{(i)} x^{(j)}]\right) - \mu^2$$
$$= \frac{1}{n^2} \left(\sum_{i} E[x^{(i)^2}] + n(n-1)E[x^{(i)}]E[x^{(j)}]\right) - \mu^2$$
$$= \frac{1}{n} E[x^2] + \frac{(n-1)}{n} \mu^2 - \mu^2 = \frac{1}{n} \left( E[x^2] - \mu^2 \right) = \frac{1}{n} \sigma_x^2$$
Variance of $\hat{\mu}_x$

$$\text{Var}_X(\hat{\mu}) = \mathbb{E}_X[(\hat{\mu} - \mathbb{E}_X[\hat{\mu}])^2] = \mathbb{E}[\hat{\mu}^2 - 2\hat{\mu}\mu + \mu^2] = \mathbb{E}[\hat{\mu}^2] - \mu^2$$

$$= \mathbb{E}\left[\frac{1}{n^2} \sum_{i,j} x^{(i)} x^{(j)}\right] - \mu^2 = \frac{1}{n^2} \sum_{i,j} \mathbb{E}[x^{(i)} x^{(j)}] - \mu^2$$

$$= \frac{1}{n^2} \left( \sum_{i=j} \mathbb{E}[x^{(i)} x^{(j)}] + \sum_{i\neq j} \mathbb{E}[x^{(i)} x^{(j)}] \right) - \mu^2$$

$$= \frac{1}{n^2} \left( \sum_i \mathbb{E}[x^{(i)}^2] + n(n-1)\mathbb{E}[x^{(i)}]\mathbb{E}[x^{(j)}] \right) - \mu^2$$

$$= \frac{1}{n} \mathbb{E}[x^2] + \frac{(n-1)}{n} \mu^2 - \mu^2 = \frac{1}{n} \left( \mathbb{E}[x^2] - \mu^2 \right) = \frac{1}{n} \sigma_x^2$$

- The variance of $\hat{\mu}_x$ diminishes as $n \to \infty$
Unbiased Estimator of $\sigma_x$

Is $\hat{\sigma}_x = \frac{1}{n} \sum_i (x^{(i)} - \hat{\mu}_x)^2$ and an unbiased estimator of $\sigma_x$?
Unbiased Estimator of $\sigma_x$

Is $\hat{\sigma}_x = \frac{1}{n} \sum_i (x^{(i)} - \hat{\mu}_x)^2$ and an unbiased estimator of $\sigma_x$? No

$$E_X[\hat{\sigma}] = E[\frac{1}{n} \sum_i (x^{(i)} - \hat{\mu})^2] = E[\frac{1}{n} (\sum_i x^{(i)} - 2 \sum_i x^{(i)} \hat{\mu} + \sum_i \hat{\mu}^2)]$$
Unbiased Estimator of $\sigma_x$

Is $\hat{\sigma}_x = \frac{1}{n} \sum_i (x^{(i)} - \hat{\mu}_x)^2$ and an unbiased estimator of $\sigma_x$?  \textit{No}

\[
\begin{align*}
E_x[\hat{\sigma}] &= E\left[\frac{1}{n} \sum_i (x^{(i)} - \hat{\mu})^2\right] = E\left[\frac{1}{n} (\sum_i x^{(i)^2} - 2 \sum_i x^{(i)} \hat{\mu} + \sum_i \hat{\mu}^2)\right] \\
&= E\left[\frac{1}{n} (\sum_i x^{(i)^2} - n \hat{\mu}^2)\right] = \frac{1}{n} (\sum_i E[x^{(i)^2}] - nE[\hat{\mu}^2])
\end{align*}
\]
Unbiased Estimator of $\sigma_x$

Is $\hat{\sigma}_x = \frac{1}{n} \sum_i (x^{(i)} - \hat{\mu}_x)^2$ and an unbiased estimator of $\sigma_x$? No

$$E_X[\hat{\sigma}] = E\left[\frac{1}{n} \sum_i (x^{(i)} - \hat{\mu})^2\right] = E\left[\frac{1}{n} (\sum_i x^{(i)2} - 2 \sum_i x^{(i)} \hat{\mu} + \sum_i \hat{\mu}^2)\right]$$
$$= E\left[\frac{1}{n} (\sum_i x^{(i)2} - n \hat{\mu}^2)\right] = \frac{1}{n} (\sum_i E[x^{(i)}] - nE[\hat{\mu}^2])$$
$$= E[x^2] - E[\hat{\mu}^2] = E[(x - \mu)^2 + 2x\mu - \mu^2] - E[\hat{\mu}^2]$$
$$= (\sigma^2 + \mu^2) - (\text{Var}[\hat{\mu}] + E[\hat{\mu}]^2)$$
Unbiased Estimator of $\sigma_x$

Is $\hat{\sigma}_x = \frac{1}{n} \sum_i (x^{(i)} - \hat{\mu}_x)^2$ and an unbiased estimator of $\sigma_x$? No

$$E_X[\hat{\sigma}] = E\left[\frac{1}{n} \sum_i (x^{(i)} - \hat{\mu})^2\right] = E\left[\frac{1}{n} (\sum_i x^{(i)}^2 - 2\sum_i x^{(i)} \hat{\mu} + \sum_i \hat{\mu}^2)\right]$$
$$= E\left[\frac{1}{n} (\sum_i x^{(i)}^2 - n\hat{\mu}^2)\right] = \frac{1}{n} (\sum_i E[x^{(i)}^2] - nE[\hat{\mu}^2])$$
$$= E[x^2] - E[\hat{\mu}^2] = E[(x - \mu)^2 + 2x\mu - \mu^2] - E[\hat{\mu}^2]$$
$$= (\sigma^2 + \mu^2) - (\text{Var}[\hat{\mu}] + E[\hat{\mu}]^2)$$
$$= \sigma^2 + \mu^2 - \frac{1}{n} \sigma^2 - \mu^2 = \frac{n-1}{n} \sigma^2 \neq \sigma^2$$
Unbiased Estimator of $\sigma_x$ 

- Is $\hat{\sigma}_x = \frac{1}{n} \sum_i (x^{(i)} - \hat{\mu}_x)^2$ and an unbiased estimator of $\sigma_x$? No

$$E_X[\hat{\sigma}] = E\left[ \frac{1}{n} \sum_i (x^{(i)} - \hat{\mu})^2 \right] = E\left[ \frac{1}{n} \left( \sum_i x^{(i)^2} - 2 \sum_i x^{(i)} \hat{\mu} + \sum_i \hat{\mu}^2 \right) \right]$$

$$= E\left[ \frac{1}{n} \left( \sum_i x^{(i)^2} - n \hat{\mu}^2 \right) \right] = \frac{1}{n} \left( \sum_i E[x^{(i)^2}] - n E[\hat{\mu}^2] \right)$$

$$= E[x^2] - E[\hat{\mu}^2] = E[(x - \mu)^2 + 2x\mu - \mu^2] - E[\hat{\mu}^2]$$

$$= (\sigma^2 + \mu^2) - (\text{Var}[\hat{\mu}] + E[\hat{\mu}]^2)$$

$$= \sigma^2 + \mu^2 - \frac{1}{n} \sigma^2 - \mu^2 = \frac{n-1}{n} \sigma^2 \neq \sigma^2$$

- What’s the unbiased estimator of $\sigma_x$?
Unbiased Estimator of $\sigma_x$

- Is $\hat{\sigma}_x = \frac{1}{n} \sum_i (x^{(i)} - \hat{\mu}_x)^2$ and an unbiased estimator of $\sigma_x$? **No**

$$
E_X[\hat{\sigma}] = E\left[\frac{1}{n} \sum_i (x^{(i)} - \hat{\mu})^2\right] = E\left[\frac{1}{n} \left(\sum_i x^{(i)} - 2 \sum_i x^{(i)} \hat{\mu} + \sum_i \hat{\mu}^2\right)\right] \\
= E\left[\frac{1}{n} (\sum_i x^{(i)} - n\hat{\mu})^2\right] = \frac{1}{n} (\sum_i E[x^{(i)}] - nE[\hat{\mu}^2]) \\
= E[x^2] - E[\hat{\mu}^2] = E[(x - \mu)^2 + 2x\mu - \mu^2] - E[\hat{\mu}^2] \\
= (\sigma^2 + \mu^2) - (\text{Var}[\hat{\mu}] + E[\hat{\mu}]^2) \\
= \sigma^2 + \mu^2 - \frac{1}{n} \sigma^2 - \mu^2 = \frac{n-1}{n} \sigma^2 \neq \sigma^2
$$

- What’s the unbiased estimator of $\sigma_x$?

$$
\hat{\sigma}_x = \frac{n}{n-1} \left(\frac{1}{n} \sum_i (x^{(i)} - \hat{\mu}_x)^2\right) = \frac{1}{n-1} \sum_i (x^{(i)} - \hat{\mu}_x)^2
$$
Mean Square Error

- **Mean square error** of an estimator:

\[
\text{MSE}(\hat{\theta}_n) = \mathbb{E}_X[(\hat{\theta}_n - \theta)^2]
\]
Mean Square Error

- **Mean square error** of an estimator:

\[
MSE(\hat{\theta}_n) = E_X[(\hat{\theta}_n - \theta)^2]
\]

- Can be decomposed into the bias and variance:

\[
E_X[(\hat{\theta}_n - \theta)^2] = E[(\hat{\theta}_n - E[\hat{\theta}_n] - E[\hat{\theta}_n] + \theta)^2]
\]
Mean Square Error

- **Mean square error** of an estimator:

\[
\text{MSE}(\hat{\theta}_n) = \mathbb{E}_X[(\hat{\theta}_n - \theta)^2]
\]

- Can be decomposed into the bias and variance:

\[
\mathbb{E}_X[(\hat{\theta}_n - \theta)^2] = \mathbb{E}[(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n] - \mathbb{E}[\hat{\theta}_n] + \theta)^2]
\]

\[
= \mathbb{E}[(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])^2 + (\mathbb{E}[\hat{\theta}_n] - \theta)^2 + 2(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])(\mathbb{E}[\hat{\theta}_n] - \theta)]
\]
Mean Square Error

- Mean square error of an estimator:

\[ \text{MSE}(\hat{\theta}_n) = \mathbb{E}_X \left[ (\hat{\theta}_n - \theta)^2 \right] \]

- Can be decomposed into the bias and variance:

\[
\mathbb{E}_X \left[ (\hat{\theta}_n - \theta)^2 \right] = \mathbb{E} \left[ (\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n] - \mathbb{E}[\hat{\theta}_n] + \theta)^2 \right] \\
= \mathbb{E} \left[ (\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])^2 + (\mathbb{E}[\hat{\theta}_n] - \theta)^2 + 2(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])(\mathbb{E}[\hat{\theta}_n] - \theta) \right] \\
= \mathbb{E} \left[ (\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])^2 \right] + \mathbb{E} \left[ (\mathbb{E}[\hat{\theta}_n] - \theta)^2 \right] + 2\mathbb{E} \left( \hat{\theta}_n - \mathbb{E}[\hat{\theta}_n] \right)(\mathbb{E}[\hat{\theta}_n] - \theta)
\]
Mean Square Error

- **Mean square error** of an estimator:

\[
\text{MSE}(\hat{\theta}_n) = \mathbb{E}_X [ (\hat{\theta}_n - \theta)^2 ]
\]

- Can be decomposed into the bias and variance:

\[
\mathbb{E}_X [ (\hat{\theta}_n - \theta)^2 ] = \mathbb{E} [ (\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n] - \mathbb{E}[\hat{\theta}_n] + \theta)^2 ]
\]
\[
= \mathbb{E} [ (\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])^2 + (\mathbb{E}[\hat{\theta}_n] - \theta)^2 + 2(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])(\mathbb{E}[\hat{\theta}_n] - \theta) ]
\]
\[
= \mathbb{E} [ (\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])^2 ] + \mathbb{E} [ (\mathbb{E}[\hat{\theta}_n] - \theta)^2 ] + 2\mathbb{E} (\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n]) (\mathbb{E}[\hat{\theta}_n] - \theta)
\]
\[
= \mathbb{E} [ (\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])^2 ] + (\mathbb{E}[\hat{\theta}_n] - \theta)^2 + 2 \cdot 0 \cdot (\mathbb{E}[\hat{\theta}_n] - \theta)
\]
Mean Square Error

- **Mean square error** of an estimator:
  \[
  \text{MSE}(\hat{\theta}_n) = E_X[(\hat{\theta}_n - \theta)^2]
  \]

- Can be decomposed into the bias and variance:
  \[
  E_X[(\hat{\theta}_n - \theta)^2] = E[(\hat{\theta}_n - E[\hat{\theta}_n] - E[\hat{\theta}_n] + \theta)^2]
  = E[(\hat{\theta}_n - E[\hat{\theta}_n])^2 + (E[\hat{\theta}_n] - \theta)^2 + 2(\hat{\theta}_n - E[\hat{\theta}_n])(E[\hat{\theta}_n] - \theta)]
  = E[(\hat{\theta}_n - E[\hat{\theta}_n])^2] + E[(E[\hat{\theta}_n] - \theta)^2] + 2E(\hat{\theta}_n - E[\hat{\theta}_n])(E[\hat{\theta}_n] - \theta)
  = E[(\hat{\theta}_n - E[\hat{\theta}_n])^2] + (E[\hat{\theta}_n] - \theta)^2 + 2 \cdot 0 \cdot (E[\hat{\theta}_n] - \theta)
  = \text{Var}_X(\hat{\theta}_n) + \text{bias}(\hat{\theta}_n)^2
  \]

- MSE of an unbiased estimator is its variance
Outline

1. Learning Theory

2. Point Estimation: Bias and Variance
   - Consistency*

3. Decomposing Generalization Error

4. Regularization
   - Weight Decay
   - Validation
Consistency

- So far, we discussed the “goodness” of an estimator based on samples of fixed size
Consistency

- So far, we discussed the “goodness” of an estimator based on samples of fixed size
- If we have more samples, will the estimate become more accurate?

An estimator is (weak) consistent iff:

$$\lim_{n \to \infty} \hat{\theta}_n \overset{Pr}{\to} \theta,$$

where $\overset{Pr}{\to}$ means “converge in probability”.

Strong consistent iff “converge almost surely”.
Consistency

- So far, we discussed the “goodness” of an estimator based on samples of fixed size.
- If we have more samples, will the estimate become more accurate?
- An estimator is (weak) consistent iff:

  \[
  \lim_{n \to \infty} \hat{\theta}_n \xrightarrow{\text{Pr}} \theta,
  \]

  where \( \xrightarrow{\text{Pr}} \) means “converge in probability”
Consistency

- So far, we discussed the “goodness” of an estimator based on samples of fixed size.
- If we have more samples, will the estimate become more accurate?
- An estimator is (weak) **consistent** iff:

\[
\lim_{n \to \infty} \hat{\theta}_n \overset{\text{Pr}}{\to} \theta,
\]

where \(\overset{\text{Pr}}{\to}\) means “converge in probability”

- Strong consistent iff “converge almost surely”
Law of Large Numbers

Theorem (Weak Law of Large Numbers)

The sample mean $\hat{\mu}_x = \frac{1}{n} \sum_i x^{(i)}$ is a consistent estimator of $\mu_x$, i.e.,

$$\lim_{n \to \infty} \Pr \left( \left| \hat{\mu}_{x,n} - \mu_x \right| < \epsilon \right) = 1$$

for any $\epsilon > 0$. 

Theorem (Strong Law of Large Numbers)

In addition, $\hat{\mu}_x$ is a strong consistent estimator:

$$\Pr \left( \lim_{n \to \infty} \hat{\mu}_{x,n} = \mu_x \right) = 1.$$
Theorem (Weak Law of Large Numbers)

The sample mean \( \hat{\mu}_x = \frac{1}{n} \sum_i x^{(i)} \) is a consistent estimator of \( \mu_x \), i.e.,
\[
\lim_{n \to \infty} \Pr \left( |\hat{\mu}_{x,n} - \mu_x| < \varepsilon \right) = 1 \text{ for any } \varepsilon > 0.
\]

Theorem (Strong Law of Large Numbers)

In addition, \( \hat{\mu}_x \) is a strong consistent estimator: \( \Pr \left( \lim_{n \to \infty} \hat{\mu}_{x,n} = \mu_x \right) = 1 \).
Outline

1. Learning Theory

2. Point Estimation: Bias and Variance
   - Consistency*

3. Decomposing Generalization Error

4. Regularization
   - Weight Decay
   - Validation
Expected Generalization Error

- In ML, we get $f_N = \arg \min_{f \in \mathcal{F}} C_N[f]$ by minimizing the empirical error over a training set of size $N$.
- How to decompose the generalization error $C[f_N]$?
Expected Generalization Error

- In ML, we get $f_N = \arg\min_{f \in F} C_N[f]$ by minimizing the empirical error over a training set of size $N$.

- How to decompose the generalization error $C[f_N]$?

- Regard $f_N(x)$ as an estimate of true label $y$ given $x$.
  - $f_N$ an estimator mapped from i.i.d. samples in the training set $X$.

- To evaluate the estimator $f_N$, we consider the expected generalization error:
  $$E_X (C[f_N]) = E_X \left[ \int \text{loss}(f_N(x) - y) dP(x, y) \right]$$
Expected Generalization Error

- In ML, we get $f_N = \arg\min_{f \in \mathcal{F}} C_N[f]$ by minimizing the empirical error over a training set of size $N$.
- How to decompose the generalization error $C[f_N]$?
- Regard $f_N(x)$ as an estimate of true label $y$ given $x$.
  - $f_N$ an estimator mapped from i.i.d. samples in the training set $\mathcal{X}$.
- To evaluate the estimator $f_N$, we consider the expected generalization error:
  \[
  E_X (C[f_N]) = E_X \left[ \int \text{loss}(f_N(x) - y) dP(x, y) \right] \\
  = E_{X, x, y} [\text{loss}(f_N(x) - y)]
  \]
Expected Generalization Error

- In ML, we get $f_N = \arg\min_{f \in F} C_N[f]$ by minimizing the empirical error over a training set of size $N$.
- How to decompose the generalization error $C[f_N]$?
- Regard $f_N(x)$ as an estimate of true label $y$ given $x$.
  - $f_N$ an estimator mapped from i.i.d. samples in the training set $X$.
- To evaluate the estimator $f_N$, we consider the expected generalization error:
  \[
  E_X(C[f_N]) = E_X \left[ \int \text{loss}(f_N(x) - y) dP(x,y) \right] \\
  = E_{X,x,y} [\text{loss}(f_N(x) - y)] \\
  = E_x \left( E_{X,y} [\text{loss}(f_N(x) - y) | x = x] \right)
  \]
### Expected Generalization Error

- In ML, we get \( f_N = \text{arg min}_{f \in F} C_N[f] \) by minimizing the empirical error over a training set of size \( N \)
- How to decompose the generalization error \( C[f_N] \)?
- Regard \( f_N(x) \) as an estimate of true label \( y \) given \( x \)
  - \( f_N \) an estimator mapped from i.i.d. samples in the training set \( \mathcal{X} \)
- To evaluate the estimator \( f_N \), we consider the expected generalization error:
  \[
  E_{\mathcal{X}} (C[f_N]) = E_{\mathcal{X}} [\int \text{loss}(f_N(x) - y)dP(x, y)] \\
  = E_{\mathcal{X},x,y} [\text{loss}(f_N(x) - y)] \\
  = E_{x} (E_{\mathcal{X},y} [\text{loss}(f_N(x) - y)|x = x])
  \]
- There’s a simple decomposition of \( E_{\mathcal{X},y} [\text{loss}(f_N(x) - y)|x] \) for linear/polynomial regression
Example: Linear/Polynomial Regression

- In linear/polynomial regression, we have
  - $\text{loss}(\cdot) = (\cdot)^2$ a squared loss
  - $y = f^*(x) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, thus $E_y[y|x] = f^*(x)$ and $\text{Var}_y[y|x] = \sigma^2$
Example: Linear/Polynomial Regression

In linear/polynomial regression, we have

- loss(·) = (·)^2 a squared loss
- \( y = f^*(x) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2) \), thus \( E_y[y|x] = f^*(x) \) and \( \text{Var}_y[y|x] = \sigma^2 \)
- We can decompose the mean square error:

\[
E_{x,y}[\text{loss}(f_N(x) - y)|x] = E_{x,y}[(f_N(x) - y)^2|x] \\
= E_{x,y}[y^2 + f_N(x)^2 - 2f_N(x)y|x]
\]
Example: Linear/Polynomial Regression

- In linear/polynomial regression, we have
  - loss(·) = (·)^2 a squared loss
  - \( y = f^*(x) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2) \), thus \( E_y[y|x] = f^*(x) \) and \( \text{Var}_y[y|x] = \sigma^2 \)
- We can decompose the mean square error:

\[
E_{X,Y}[\text{loss}(f_N(x) - y)|x] = E_{X,Y}[(f_N(x) - y)^2|x] \\
= E_{X,Y}[y^2 + f_N(x)^2 - 2f_N(x)y|x] \\
= E_y[y^2|x] + E_X[f_N(x)^2|x] - 2E_{X,Y}[f_N(x)y|x]
\]
Example: Linear/Polynomial Regression

- In linear/polynomial regression, we have
  - $\text{loss}(\cdot) = (\cdot)^2$ a squared loss
  - $y = f^*(x) + \varepsilon$, $\varepsilon \sim N(0, \sigma^2)$, thus $E_y[y|x] = f^*(x)$ and $\text{Var}_y[y|x] = \sigma^2$
- We can decompose the mean square error:
  
  $$E_{X,y}[\text{loss}(f_N(x) - y)|x] = E_{X,y}[(f_N(x) - y)^2|x]$$
  $$= E_{X,y}[y^2 + f_N(x)^2 - 2f_N(x)y|x]$$
  $$= E_y[y^2|x] + E_X[f_N(x)^2|x] - 2E_{X,y}[f_N(x)y|x]$$
  $$= (\text{Var}_y[y|x] + E_y[y|x]^2) + (\text{Var}_X[f_N(x)|x] + E_X[f_N(x)|x]^2)$$
  $$- 2E_y[y|x]E_X[f_N(x)|x]$$
Example: Linear/Polynomial Regression

- In linear/polynomial regression, we have
  - $\text{loss}(\cdot) = (\cdot)^2$ a squared loss
  - $y = f^*(x) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$, thus $E_y[y|x] = f^*(x)$ and $\text{Var}_y[y|x] = \sigma^2$

- We can decompose the mean square error:

\[
E_{X,Y}[\text{loss}(f_N(x) - y)|x] = E_{X,Y}[(f_N(x) - y)^2|x] \\
= E_{X,Y}[y^2 + f_N(x)^2 - 2f_N(x)y|x] \\
= E_y[y^2|x] + E_X[f_N(x)^2|x] - 2E_{X,Y}[f_N(x)y|x] \\
= (\text{Var}_y[y|x] + E_y[y|x]^2) + (\text{Var}_X[f_N(x)|x] + E_X[f_N(x)|x]^2) \\
- 2E_y[y|x]E_X[f_N(x)|x] \\
= \text{Var}_y[y|x] + \text{Var}_X[f_N(x)|x] + (E_X[f_N(x)|x] - E_y[y|x])^2
\]
Example: Linear/Polynomial Regression

In linear/polynomial regression, we have

- $\text{loss}(\cdot) = (\cdot)^2$ a squared loss
- $y = f^*(x) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, thus $E_y[y|x] = f^*(x)$ and $\text{Var}_y[y|x] = \sigma^2$

We can decompose the mean square error:

$$E_{X,y}[\text{loss}(f_N(x) - y)|x] = E_{X,y}[(f_N(x) - y)^2|x]$$
$$= E_{X,y}[y^2 + f_N(x)^2 - 2f_N(x)y|x]$$
$$= E_y[y^2|x] + E_X[f_N(x)^2|x] - 2E_{X,y}[f_N(x)y|x]$$
$$= (\text{Var}_y[y|x] + E_y[y|x]^2) + (\text{Var}_X[f_N(x)|x] + E_X[f_N(x)|x]^2)$$
$$\quad - 2E_y[y|x]E_X[f_N(x)|x]$$
$$= \text{Var}_y[y|x] + \text{Var}_X[f_N(x)|x] + (E_X[f_N(x)|x] - E_y[y|x])^2$$
$$= \text{Var}_y[y|x] + \text{Var}_X[f_N(x)|x] + E_X[f_N(x) - f^*(x)|x]^2$$
Example: Linear/Polynomial Regression

- In linear/polynomial regression, we have
  - loss(·) = (·)^2 a squared loss
  - \( y = f^*(x) + \epsilon \), \( \epsilon \sim \mathcal{N}(0, \sigma^2) \), thus \( \mathbb{E}_y[y|x] = f^*(x) \) and \( \text{Var}_y[y|x] = \sigma^2 \)
- We can decompose the mean square error:

\[
\mathbb{E}_{X,y}[\text{loss}(f_N(x) - y)|x] = \mathbb{E}_{X,y}[(f_N(x) - y)^2|x] \\
= \mathbb{E}_{X,y}[y^2 + f_N(x)^2 - 2f_N(x)y|x] \\
= \mathbb{E}_y[y^2|x] + \mathbb{E}_X[f_N(x)^2|x] - 2\mathbb{E}_{X,y}[f_N(x)y|x] \\
= (\text{Var}_y[y|x] + \mathbb{E}_y[y|x]^2) + (\text{Var}_X[f_N(x)|x] + \mathbb{E}_X[f_N(x)|x]^2) \\
- 2\mathbb{E}_y[y|x]\mathbb{E}_X[f_N(x)|x] \\
= \text{Var}_y[y|x] + \text{Var}_X[f_N(x)|x] + (\mathbb{E}_X[f_N(x)|x] - \mathbb{E}_y[y|x])^2 \\
= \text{Var}_y[y|x] + \text{Var}_X[f_N(x)|x] + \mathbb{E}_X[f_N(x) - f^*(x)|x]^2 \\
= \sigma^2 + \text{Var}_X[f_N(x)|x] + \text{bias}[f_N(x)|x]^2
\]
Bias-Variance Tradeoff I

\[ E_x(C[f_N]) = E_x(E_{x,y}[\text{loss}(f_N(x) - y)|x]) = E_x(\sigma^2 + \text{Var}_x[f_N(x)|x] + \text{bias}[f_N(x)|x]^2) \]
Bias-Variance Tradeoff I

\[ E_x(C[f_N]) = E_x(E_{x,y}[\text{loss}(f_N(x) - y)|x]) \\
= E_x(\sigma^2 + \text{Var}_x[f_N(x)|x] + \text{bias}[f_N(x)|x]^2) \]

- The first term cannot be avoided when \( P(y|x) \) is stochastic
Bias-Variance Tradeoff I

\[ E_X (C[f_N]) = E_X (E_{X,y} [\text{loss}(f_N(x) - y)|x]) = E_X (\sigma^2 + \text{Var}_X [f_N(x)|x] + \text{bias}[f_N(x)|x]^2) \]

- The first term cannot be avoided when \( P(y|x) \) is stochastic
- **Model complexity** controls the tradeoff between variance and bias
- E.g., polynomial regressors (dotted line = average training error):
Bias-Variance Tradeoff II

- Provides another way to understand the generalization/testing error

Too simple a model leads to high bias or underfitting
- High training error;
- High testing error (given a sufficiently large $N$)

Too complex a model leads to high variance or overfitting
- Low training error;
- High testing error
Bias-Variance Tradeoff II

- Provides another way to understand the generalization/testing error
- Too simple a model leads to high bias or underfitting
  - *High* training error; *high* testing error (given a sufficiently large $N$)
Bias-Variance Tradeoff II

- Provides another way to understand the generalization/testing error
- Too simple a model leads to high bias or underfitting
  - *High* training error; *high* testing error (given a sufficiently large $N$)
- Too complex a model leads to high variance or overfitting
  - *Low* training error; *high* testing error
Outline

1. Learning Theory

2. Point Estimation: Bias and Variance
   - Consistency*

3. Decomposing Generalization Error

4. Regularization
   - Weight Decay
   - Validation
We get $f_N = \arg\min_{f \in F} C_N[f]$ by minimizing the empirical error.
We get $f_N = \arg\min_{f \in F} C_N[f]$ by minimizing the empirical error. But what we really care about is the generalization error $C[f_N]$. Regularization refers to any technique designed to improve the generalizability of $f_N$. Any idea inspired by the learning theory?
We get $f_N = \arg\min_{f \in F} C_N[f]$ by minimizing the empirical error $C_N[f]$. But what we really care about is the generalization error $C[f_N]$. 

**Regularization** refers to any technique designed to improve the generalizability of $f_N$. Any idea inspired by the learning theory?
We get $f_N = \arg\min_{f \in F} C_N[f]$ by minimizing the empirical error. But what we really care about is the generalization error $C[f_N]$. *Regularization* refers to any technique designed to improve the generalizability of $f_N$. Any idea inspired by the learning theory? Regularization in the cost function: *weight decay*. Regularization during the training process: *validation*.
Outline

1. Learning Theory
2. Point Estimation: Bias and Variance
   - Consistency*
3. Decomposing Generalization Error
4. Regularization
   - Weight Decay
   - Validation
Panelizing Complex Functions

- **Occam’s razor**: among equal-performing models, the simplest one should be selected.
Panelizing Complex Functions

- **Occam’s razor**: among equal-performing models, the simplest one should be selected.

- Idea: to add a term in the cost function that panelizes complex functions.

- So, with sufficiently complex $F$:
  - Minimizing the empirical error term reduces bias.
  - Minimizing the penalty term reduces variance.
What to Panelize?

- What impacts Complexity($F$) in a model?
What to Panelize?

- What impacts Complexity($\mathcal{F}$) in a model?
- Some constants in the model $\mathcal{F}$
  - E.g., degree $P$ in polynomial regression
- Restricts the capacity of $\mathcal{F}$
What to Panelize?

- What impacts Complexity($F$) in a model?
- Some constants in the model $F$
  - E.g., degree $P$ in polynomial regression
- Restricts the capacity of $F$
- However, cannot be penalized in a cost function since fixed
What to Panelize?

• What impacts Complexity($F$) in a model?
• Some constants in the model $F$
  • E.g., degree $P$ in polynomial regression
• Restricts the capacity of $F$
• However, cannot be penalized in a cost function since fixed
• Alternatively, \textit{function parameters}
  • E.g., the parameter $w$ of a function $f(\cdot;w) \in F$
• Also restricts the capacity of $F$
• Can be penalized
What to Panelize?

- What impacts Complexity(\(F\)) in a model?
- Some constants in the model \(F\)
  - E.g., degree \(P\) in polynomial regression
- Restricts the capacity of \(F\)
- However, cannot be penalized in a cost function since fixed
- Alternatively, *function parameters*
  - E.g., the parameter \(w\) of a function \(f(\cdot;w) \in F\)
- Also restricts the capacity of \(F\)
- Can be penalized
- But which \(w\) implies a complex model?
Weight Decay

- In practice, $w = 0$ is usually the “simplest” function
  - E.g, in binary classification for labels $\{-1, 1\}$, a perceptron with $w = 0$ means random guessing

Weight decay: to penalize the norm of $w$, which is nonnegative and equals to 0 when $w = 0$.

E.g., the Ridge regression:

$$\arg\min_w b \quad \| y - (Xw - b) \|_2^2 \text{ subject to } \|w\|_2^2 \leq T$$

for some constant $T > 0$.

In practice, we usually solve a simpler problem:

$$\arg\min_w b \quad \| y - (Xw - b) \|_2^2 + \alpha \|w\|_2^2$$

where $\alpha > 0$ is a constant representing both $T$ and the KKT multiplier.

What does a larger $\alpha$ mean?
Weight Decay

- In practice, $w = 0$ is usually the “simplest” function
  - E.g., in binary classification for labels $\{-1, 1\}$, a perceptron with $w = 0$ means random guessing

- Weight decay: to penalize the norm of $w$, which is nonnegative and equals to 0 when $w = 0$
Weight Decay

- In practice, $w = 0$ is usually the “simplest” function
  - E.g., in binary classification for labels $\{-1, 1\}$, a perceptron with $w = 0$ means random guessing

- **Weight decay**: to penalize the norm of $w$, which is nonnegative and equals to 0 when $w = 0$

- E.g., the **Ridge regression**:

$$\arg\min_{w, b} \frac{1}{2} \|y - (Xw - b1)\|^2 \text{ subject to } \|w\|^2 \leq T$$

for some constant $T > 0$
Weight Decay

- In practice, $w = 0$ is usually the “simplest” function
  - E.g., in binary classification for labels $\{-1, 1\}$, a perceptron with $w = 0$ means random guessing

- *Weight decay*: to penalize the *norm* of $w$, which is nonnegative and equals to $0$ when $w = 0$

- E.g., the *Ridge regression*:

$$
\arg\min_{w,b} \frac{1}{2} \| y - (Xw - b1) \|^2 \text{ subject to } \|w\|^2 \leq T
$$

for some constant $T > 0$

- In practice, we usually solve a simpler problem:

$$
\arg\min_{w,b} \frac{1}{2N} \| y - (Xw - b1) \|^2 + \frac{\alpha}{2} \|w\|^2
$$

where $\alpha > 0$ is a constant representing both $T$ and the KKT multiplier
Weight Decay

- In practice, $w = 0$ is usually the “simplest” function
  - E.g., in binary classification for labels $\{-1, 1\}$, a perceptron with $w = 0$ means random guessing

- **Weight decay**: to penalize the norm of $w$, which is nonnegative and equals to 0 when $w = 0$

- E.g., the *Ridge regression*:

  $$\arg\min_{w, b} \frac{1}{2} ||y - (Xw - b1)||^2 \text{ subject to } ||w||^2 \leq T$$

  for some constant $T > 0$

- In practice, we usually solve a simpler problem:

  $$\arg\min_{w, b} \frac{1}{2N} ||y - (Xw - b1)||^2 + \frac{\alpha}{2} ||w||^2$$

  where $\alpha > 0$ is a constant representing both $T$ and the KKT multiplier

- What does a larger $\alpha$ means?
Weight Decay

- In practice, \( w = 0 \) is usually the “simplest” function
  - E.g., in binary classification for labels \( \{-1, 1\} \), a perceptron with \( w = 0 \) means random guessing

- **Weight decay**: to penalize the norm of \( w \), which is nonnegative and equals to 0 when \( w = 0 \)

- E.g., the **Ridge regression**:

  \[
  \arg \min_{w,b} \frac{1}{2} \| y - (Xw - b1) \|^2 \text{ subject to } \|w\|^2 \leq T
  \]

  for some constant \( T > 0 \)

- In practice, we usually solve a simpler problem:

  \[
  \arg \min_{w,b} \frac{1}{2N} \| y - (Xw - b1) \|^2 + \frac{\alpha}{2} \|w\|^2
  \]

  where \( \alpha > 0 \) is a constant representing both \( T \) and the KKT multiplier

- What does a larger \( \alpha \) means? We prefer a more simple function

Shan-Hung Wu (CS, NTHU) Learning Theory & Regularization Machine Learning
Flat Regressors

\[ \text{arg min}_{w,b} \frac{1}{2} \left( \| y - (Xw - b1) \|^2 + \alpha \| w \|^2 \right) \]

- The bias \( b \) is \textit{not} regularized, why?
Flat Regressors

\[ \arg\min_{w,b} \frac{1}{2} (\|y - (Xw - b1)\|^2 + \alpha \|w\|^2) \]

- The bias \(b\) is not regularized, why?
- We want the simplest function with \(w = 0\) means "a dummy regressor by averaging"
  - Remember \(R^2\) (coefficient of determination)?
- However, the label \(y\)'s may not be standardized to have zero mean
Flat Regressors

\[
\arg\min_{w,b} \frac{1}{2} (\|y - (Xw - b1)\|^2 + \alpha\|w\|^2)
\]

- The bias \(b\) is *not* regularized, why?
- We want the simplest function with \(w = 0\) means "a dummy regressor by averaging"
  - Remember \(R^2\) (coefficient of determination)?
- However, the label \(y\)'s may not be standardized to have zero mean
- This explains why we prefer a "flat" hyperplane in the previous lecture
- We have discussed how to solve the Ridge regression problem
Sparse Weight Decay

- Alternatively we can minimize the $L^1$-norm in weight decay
- E.g., LASSO (least absolute shrinkage and selection operator):

$$\arg\min_{w,b} \frac{1}{2N} \|y - (Xw - b1)\|^2 + \alpha \|w\|_1$$

for some constant $\alpha > 0$

- Usually results in sparse $w$ that has many zero attributes
- Why?
Sparsity

$$\arg \min_{w, b} \frac{1}{2N} \| y - (Xw - b1) \|^2 + \alpha \| w \|_1$$

- The surface of the cost function is the sum of SSE (blue contours) and 1-norm (red contours)
- Optimal point locates on some axes
Elastic Net**

- LASSO can be used as a feature selection technique
  - The sparse $w$ selects explanatory variables that are most correlated to the target variable

$$\text{argmin} w, b \frac{1}{2} \sum_{i=1}^{N} (y_i - (Xw - b))^2 + \alpha (\beta \|w\|_1 + (1-\beta) \|w\|_2^2)$$

for some constant $\beta \in (0, 1)$

Still gives a sparse $w$

Highly correlated variables will have similar values in $w$
Elastic Net**

- LASSO can be used as a feature selection technique
  - The sparse $w$ selects explanatory variables that are most correlated to the target variable

- Limitations:
  1. Selects at most $N$ variables if $D > N$
  2. No *group selection*
     - Important in some applications, e.g., gene selection problems
Elastic Net**

- LASSO can be used as a feature selection technique
  - The sparse $w$ selects explanatory variables that are most correlated to the target variable

- Limitations:
  1. Selects at most $N$ variables if $D > N$
  2. No group selection
    - Important in some applications, e.g., gene selection problems

- Elastic net combines Ridge and LASSO:

$$
\arg\min_{w, b} \frac{1}{2N} \| y - (Xw - b1) \|^2 + \alpha \left( \beta \| w \|_1 + \frac{1 - \beta}{2} \| w \|^2 \right)
$$

for some constant $\beta \in (0, 1)$
Elastic Net**

- LASSO can be used as a feature selection technique
  - The sparse $\mathbf{w}$ selects explanatory variables that are most correlated to the target variable
- Limitations:
  1. Selects at most $N$ variables if $D > N$
  2. No *group selection*
     - Important in some applications, e.g., gene selection problems
- **Elastic net** combines Ridge and LASSO:

$$
\arg\min_{\mathbf{w}, b} \frac{1}{2N} \| \mathbf{y} - (\mathbf{X} \mathbf{w} - b \mathbf{1}) \|^2 + \alpha \left( \beta \| \mathbf{w} \|_1 + \frac{1 - \beta}{2} \| \mathbf{w} \|_2^2 \right)
$$

for some constant $\beta \in (0, 1)$
- Still gives a sparse $\mathbf{w}$
- Highly correlated variables will have similar values in $\mathbf{w}$
Outline

1. Learning Theory

2. Point Estimation: Bias and Variance
   - Consistency*

3. Decomposing Generalization Error

4. Regularization
   - Weight Decay
   - Validation
In ML, we call the constants that are fixed in a model the \textit{hyperparameters}.

- Degree $P$ in polynomial regression
- Coefficient $\alpha$ of the weight decay term in the cost function of Ridge and LASSO, etc.
Tuning Hyperparameters

- In ML, we call the constants that are fixed in a model the **hyperparameters**
  - Degree $P$ in polynomial regression
  - Coefficient $\alpha$ of the weight decay term in the cost function of Ridge and LASSO, etc.
- Usually reflect some assumptions about the model
- Changing their values changes model complexity
  - And therefore generalization performance
In ML, we call the constants that are fixed in a model the **hyperparameters**
- Degree $P$ in polynomial regression
- Coefficient $\alpha$ of the weight decay term in the cost function of Ridge and LASSO, etc.

Usually reflect some assumptions about the model
Changing their values changes model complexity
    - And therefore generalization performance

How to set appropriate values?
In ML, we call the constants that are fixed in a model the **hyperparameters**
- Degree $P$ in polynomial regression
- Coefficient $\alpha$ of the weight decay term in the cost function of Ridge and LASSO, etc.

Usually reflect some assumptions about the model
- Changing their values changes model complexity
  - And therefore generalization performance

How to set appropriate values?
- Train a model many times with different hyperparameters, and choose the function with best generalizability
- Very time consuming, can we have heuristics to speed up the process?
Structured Risk Minimization

- Consider again the Occam’s razor
- *Structured risk minimization*: start from the simplest model, gradually increase its complexity, and stop when overfitting
Validation Set

- Pitfall:

we peep the testing set during the training process

The final function will overfit the testing set

Optimistic testing error

Fix?

Split a validation set from the training set and use it for hyperparameter selection
Validation Set

- Pitfall: we peep the testing set during the training process
  - The final function will overfit the testing set
  - Optimistic testing error

Fix:
- Split a validation set from the training set and use it for hyperparameter selection
Validation Set

- Pitfall: we peep the testing set during the training process
  - The final function will overfit the testing set
  - Optimistic testing error

- Fix?
Validation Set

- Pitfall: we peep the testing set during the training process
  - The final function will overfit the testing set
  - Optimistic testing error

- Fix? Split a *validation set* from the training set and use it for hyperparameter selection
Reference I

Concentration inequalities and empirical processes theory applied to the analysis of learning algorithms. 

Some applications of concentration inequalities to statistics. 

On the uniform convergence of relative frequencies of events to their probabilities. 
The lack of a priori distinctions between learning algorithms.