W1. Introduction and Machine Learning Pipeline

Guang Cheng

University of California, Los Angeles

guangcheng@ucla.edu

Week 1

The goal of this course

- Learn basic methods and concepts of machine learning

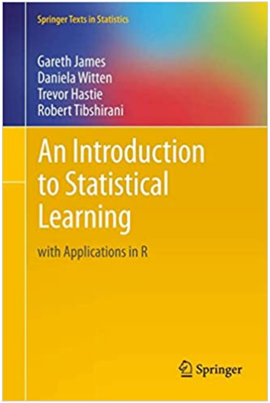
The goal of this course

- Learn basic methods and concepts of machine learning
- Apply machine learning methods to real data

The goal of this course

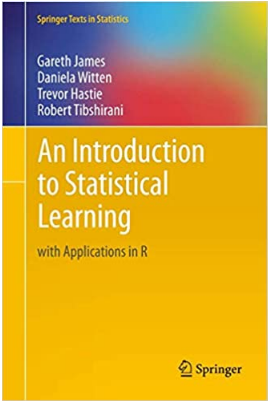
- Learn basic methods and concepts of machine learning
- Apply machine learning methods to real data
- No programming in the class

Recommended textbook



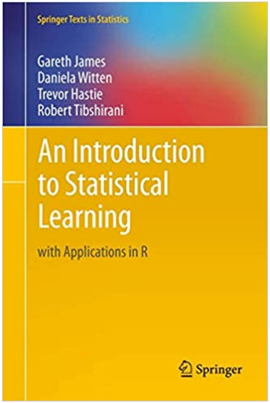
- Free to download: https://www.statlearning.com/

Recommended textbook



- Free to download: https://www.statlearning.com/
- Cover weeks 1, 2, 3, 4, 5, 6, 8, and 9.

Recommended textbook



- Free to download: https://www.statlearning.com/
- Cover weeks 1, 2, 3, 4, 5, 6, 8, and 9.
- R or Python

Course instructor

- **Guang Cheng**:

Course instructor

- **Guang Cheng**:
  - Email: guangcheng@ucla.edu

Course instructor

- **Guang Cheng**:
  - Email: guangcheng@ucla.edu
  - Office: Math 8105H (Math Building 8th floor)

Course instructor

- **Guang Cheng**:
  - Email: guangcheng@ucla.edu
  - Office: Math 8105H (Math Building 8th floor)
  - Instruction time: 2:30-3:45 each Tuesday & Thursday at Bunche Hall 2221E

Course instructor

- **Guang Cheng**:
  - Email: guangcheng@ucla.edu
  - Office: Math 8105H (Math Building 8th floor)
  - Instruction time: 2:30-3:45 each Tuesday & Thursday at Bunche Hall 2221E
  - Office hours: 1:30-2:30 each Tuesday & Thursday in Math 8105H

Course instructor

- **Guang Cheng**:
  - Email: guangcheng@ucla.edu
  - Office: Math 8105H (Math Building 8th floor)
  - Instruction time: 2:30-3:45 each Tuesday & Thursday at Bunche Hall 2221E
  - Office hours: 1:30-2:30 each Tuesday & Thursday in Math 8105H
  - During office hour, I expect questions mostly on lecture notes.

Course instructor

- **Guang Cheng**:
    - Email: guangcheng@ucla.edu
    - Office: Math 8105H (Math Building 8th floor)
    - Instruction time: 2:30-3:45 each Tuesday & Thursday at Bunche Hall 2221E
    - Office hours: 1:30-2:30 each Tuesday & Thursday in Math 8105H
    - During office hour, I expect questions mostly on lecture notes.
    - I will try to upload lecture notes before each Tuesday class.

Lab instructor

- **Dr. Samuel Borghese**:
  - Email: srborghese@g.ucla.edu

Lab instructor

- **Dr. Samuel Borghese**:
  - Email: srborghese@g.ucla.edu
  - Lab time: each Wed 2-3:15pm (teaching) & 3:30pm-4:45pm (classwork / lab application) in Bunche 2221E.

Lab instructor

- **Dr. Samuel Borghese**:
  - Email: srborghese@g.ucla.edu
  - Lab time: each Wed 2-3:15pm (teaching) & 3:30pm-4:45pm (classwork / lab application) in Bunche 2221E.
  - Office hours: 11am-12:30 each Thursday at Bunche 4353

Lab instructor

- **Dr. Samuel Borghese**:
  - Email: srborghese@g.ucla.edu
  - Lab time: each Wed 2-3:15pm (teaching) & 3:30pm-4:45pm (classwork / lab application) in Bunche 2221E.
  - Office hours: 11am-12:30 each Thursday at Bunche 4353
  - Please refer questions on lab tasks and programming to Sam

Tutor

- **Agarwal Tripti**:
  - Email: triptiagarwal@g.ucla.edu
  - Office hour: 11-1 each Tuesday
  - Location: Bunche 2221E
  - Responsibilities: other than hold office hour, Agarwal is also available to students via email and/or additional in-person meetings to answer any questions regarding coursework, homeworks, conceptual help, programming etc. Working hours is no more than 10 hours/week.

Grader (special reader)

- **TBD**:
  - Email: TBD
  - Please refer any questions about grading to him.

Grading scheme

- Homework: 60%

Grading scheme

- Homework: 60%
    - 8 homework assignments (except for weeks 7, 10)

Grading scheme

- Homework: 60%
    - 8 homework assignments (except for weeks 7, 10)
    - Typically assigned on Thursday after class before 6pm, and it will be due one week after that, say next Thursday 6pm. Homework will be announced in Bruinlearn, and you need to submit hwks there as well

Grading scheme

- Homework: 60%
  - 8 homework assignments (except for weeks 7, 10)
  - Typically assigned on Thursday after class before 6pm, and it will be due one week after that, say next Thursday 6pm. Homework will be announced in Bruinlearn, and you need to submit hwks there as well
  - Late hwk will be penalized as follows (unless there are any compelling reasons such as doctor note): if late within 3 days of due time, say before 6pm on Sunday, the score is calculated as 70% * nomial score; if late more than 3 days, it will receive a grade of zero.

Grading scheme

- Homework: 60%
  - 8 homework assignments (except for weeks 7, 10)
  - Typically assigned on Thursday after class before 6pm, and it will be due one week after that, say next Thursday 6pm. Homework will be announced in Bruinlearn, and you need to submit hwks there as well
  - Late hwk will be penalized as follows (unless there are any compelling reasons such as doctor note): if late within 3 days of due time, say before 6pm on Sunday, the score is calculated as 70% * nomial score; if late more than 3 days, it will receive a grade of zero.
  - Graders will finish grading within one week after due time, and return with solutions in the Bruinlearn

Grading scheme

- Homework: 60%
  - 8 homework assignments (except for weeks 7, 10)
  - Typically assigned on Thursday after class before 6pm, and it will be due one week after that, say next Thursday 6pm. Homework will be announced in Bruinlearn, and you need to submit hwks there as well
  - Late hwk will be penalized as follows (unless there are any compelling reasons such as doctor note): if late within 3 days of due time, say before 6pm on Sunday, the score is calculated as 70% * nomial score; if late more than 3 days, it will receive a grade of zero.
  - Graders will finish grading within one week after due time, and return with solutions in the Bruinlearn
- Lab assignment: 20% Lab Grade (8 HW/CW Assignments)

Grading scheme

- Homework: 60%
  - 8 homework assignments (except for weeks 7, 10)
  - Typically assigned on Thursday after class before 6pm, and it will be due one week after that, say next Thursday 6pm. Homework will be announced in Bruinlearn, and you need to submit hwks there as well
  - Late hwk will be penalized as follows (unless there are any compelling reasons such as doctor note): if late within 3 days of due time, say before 6pm on Sunday, the score is calculated as 70% * nomial score; if late more than 3 days, it will receive a grade of zero.
  - Graders will finish grading within one week after due time, and return with solutions in the Bruinlearn
- Lab assignment: 20% Lab Grade (8 HW/CW Assignments)

Grading scheme

- Homework: 60%
  - 8 homework assignments (except for weeks 7, 10)
  - Typically assigned on Thursday after class before 6pm, and it will be due one week after that, say next Thursday 6pm. Homework will be announced in Bruinlearn, and you need to submit hwks there as well
  - Late hwk will be penalized as follows (unless there are any compelling reasons such as doctor note): if late within 3 days of due time, say before 6pm on Sunday, the score is calculated as 70% * nomial score; if late more than 3 days, it will receive a grade of zero.
  - Graders will finish grading within one week after due time, and return with solutions in the Bruinlearn
- Lab assignment: 20% Lab Grade (8 HW/CW Assignments)
- Final exam: 20%. (in-person, open book, 60 mins, and in the final week)

Syllabus

- Week 1: Introduction and ML Pipeline

Syllabus

- Week 1: Introduction and ML Pipeline
- Week 2: Discrete Classification I - Logistic Regression

Syllabus

- Week 1: Introduction and ML Pipeline
- Week 2: Discrete Classification I - Logistic Regression
- Week 3: Linear Models, Regularization, and Hyperparameter Tuning

Syllabus

- Week 1: Introduction and ML Pipeline
- Week 2: Discrete Classification I - Logistic Regression
- Week 3: Linear Models, Regularization, and Hyperparameter Tuning
- Week 4: Discrete Classification II - Decision Trees

Syllabus

- Week 1: Introduction and ML Pipeline
- Week 2: Discrete Classification I - Logistic Regression
- Week 3: Linear Models, Regularization, and Hyperparameter Tuning
- Week 4: Discrete Classification II - Decision Trees
- Week 5: Dealing with Imbalanced Data

Syllabus

- Week 1: Introduction and ML Pipeline
- Week 2: Discrete Classification I - Logistic Regression
- Week 3: Linear Models, Regularization, and Hyperparameter Tuning
- Week 4: Discrete Classification II - Decision Trees
- Week 5: Dealing with Imbalanced Data
- Week 6: Neural Nets

Syllabus

- Week 1: Introduction and ML Pipeline
- Week 2: Discrete Classification I - Logistic Regression
- Week 3: Linear Models, Regularization, and Hyperparameter Tuning
- Week 4: Discrete Classification II - Decision Trees
- Week 5: Dealing with Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models

Syllabus

- Week 1: Introduction and ML Pipeline
- Week 2: Discrete Classification I - Logistic Regression
- Week 3: Linear Models, Regularization, and Hyperparameter Tuning
- Week 4: Discrete Classification II - Decision Trees
- Week 5: Dealing with Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models
- Week 8: Bagging and Boosting

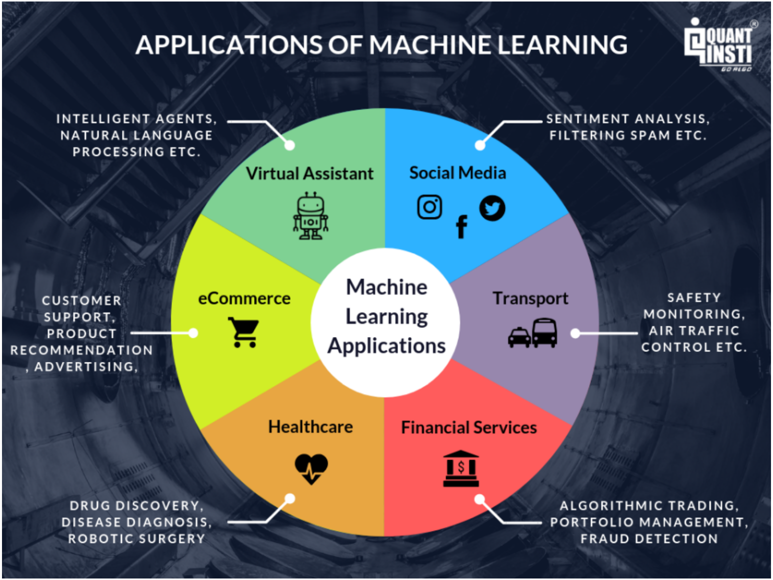
Syllabus

- Week 1: Introduction and ML Pipeline
- Week 2: Discrete Classification I - Logistic Regression
- Week 3: Linear Models, Regularization, and Hyperparameter Tuning
- Week 4: Discrete Classification II - Decision Trees
- Week 5: Dealing with Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models
- Week 8: Bagging and Boosting
- Week 9: Unsupervised Learning - Clustering & PCA

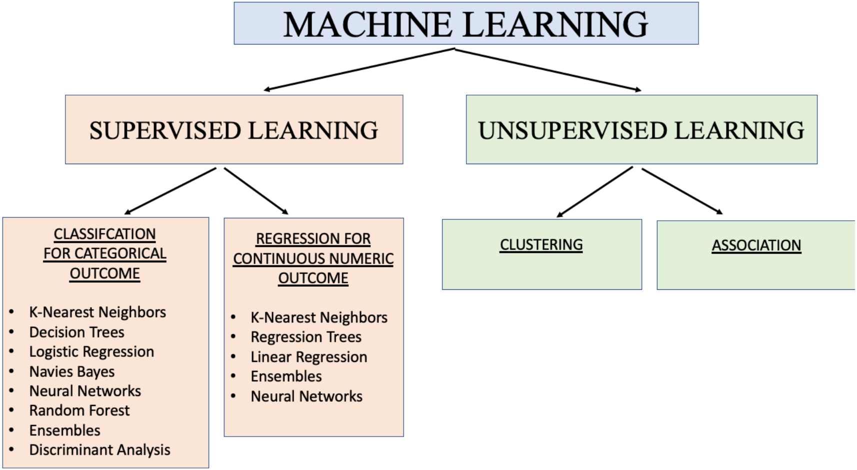
Syllabus

- Week 1: Introduction and ML Pipeline
- Week 2: Discrete Classification I - Logistic Regression
- Week 3: Linear Models, Regularization, and Hyperparameter Tuning
- Week 4: Discrete Classification II - Decision Trees
- Week 5: Dealing with Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models
- Week 8: Bagging and Boosting
- Week 9: Unsupervised Learning - Clustering & PCA
- Week 10: Introduction of Reinforcement Learning

Machine learning in the real world

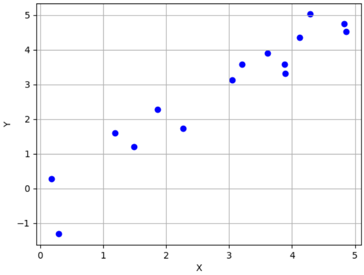


ML Quadrants



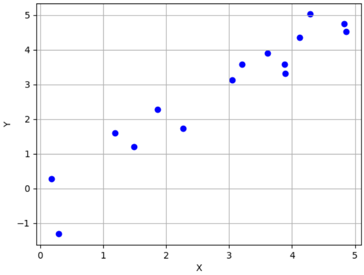
Machine Learning and Statistical Models

- A simple example: Suppose we observe a dataset:



Machine Learning and Statistical Models

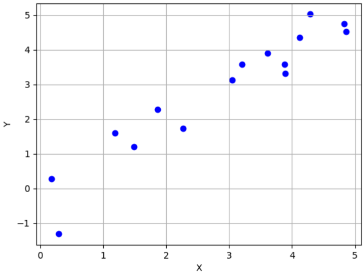
- A simple example: Suppose we observe a dataset:



- What is machine learning?

Machine Learning and Statistical Models

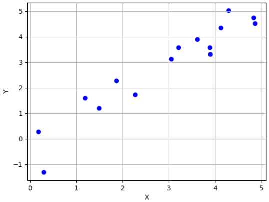
- A simple example: Suppose we observe a dataset:



- What is machine learning?
- What is a statistical model?

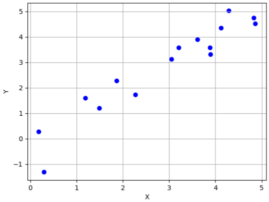
Machine Learning and Statistical Models

- Suppose we observe a dataset:



Machine Learning and Statistical Models

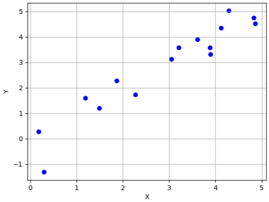
- Suppose we observe a dataset:



- **What is machine learning**?

Machine Learning and Statistical Models

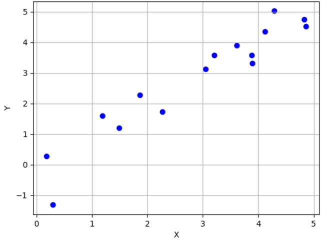
- Suppose we observe a dataset:



- **What is machine learning**?
  - Machine Learning is the study of computational algorithms that often applies to (unstructured) big data, e.g., image and text, with a particular focus on prediction.

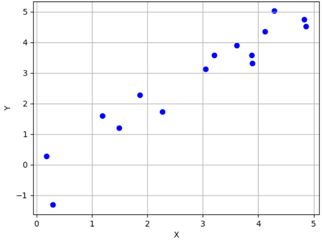
Statistical Model

Statistical model: is a mathematical model (built up a set of statistical assumptions) and is mostly concerned about estimation and inference, e.g., hypothesis testing. It is mostly useful for small data and applies to scenarios that demands interpretability.



Statistical Model

Statistical model: is a mathematical model (built up a set of statistical assumptions) and is mostly concerned about estimation and inference, e.g., hypothesis testing. It is mostly useful for small data and applies to scenarios that demands interpretability.

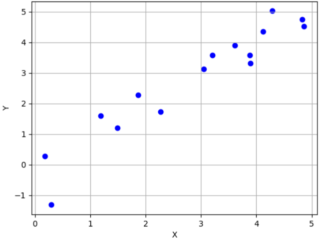


Assumptions:

- $Y = f(X) + \epsilon$, where f is a true model

Statistical Model

Statistical model: is a mathematical model (built up a set of statistical assumptions) and is mostly concerned about estimation and inference, e.g., hypothesis testing. It is mostly useful for small data and applies to scenarios that demands interpretability.

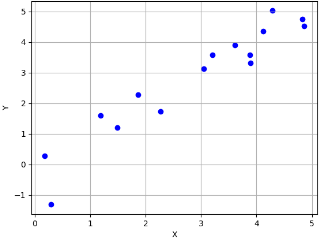


Assumptions:

- $Y = f(X) + \epsilon$, where f is a true model
- X and ϵ are independent

Statistical Model

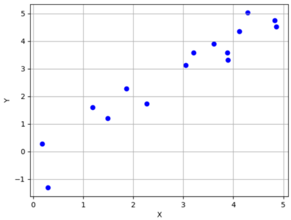
Statistical model: is a mathematical model (built up a set of statistical assumptions) and is mostly concerned about estimation and inference, e.g., hypothesis testing. It is mostly useful for small data and applies to scenarios that demands interpretability.

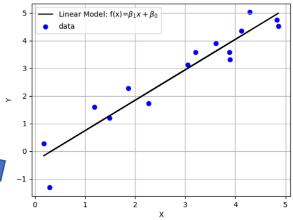


Assumptions:

- $Y = f(X) + \epsilon$, where f is a true model
- X and ϵ are independent
- $\mathbb{E}(\epsilon) = 0$ and $Var(\epsilon) = \sigma^2$

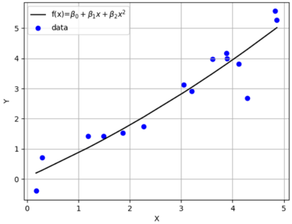
An example: how to determine the form of f





Assumption: the relationship between X and Y is Linear!

Assumption: the relationship between X and Y is Quadratic!



Statistical machine learning

Nowadays, two different cultures merge as the so called "Statistical Machine Learning," which is the main focus of this course.

Statistical machine learning

Nowadays, two different cultures merge as the so called "Statistical Machine Learning," which is the main focus of this course.

In what follows, I will introduce some basic concepts in the statistical machine learning.

Statistical machine learning

Nowadays, two different cultures merge as the so called "Statistical Machine Learning," which is the main focus of this course.

In what follows, I will introduce some basic concepts in the statistical machine learning.

- Parametric models vs non-parametric models

Statistical machine learning

Nowadays, two different cultures merge as the so called "Statistical Machine Learning," which is the main focus of this course.

In what follows, I will introduce some basic concepts in the statistical machine learning.

- Parametric models vs non-parametric models
- Training data vs testing data

Statistical machine learning

Nowadays, two different cultures merge as the so called "Statistical Machine Learning," which is the main focus of this course.

In what follows, I will introduce some basic concepts in the statistical machine learning.

- Parametric models vs non-parametric models
- Training data vs testing data
- Bias and variance tradeoff

Statistical machine learning

Nowadays, two different cultures merge as the so called "Statistical Machine Learning," which is the main focus of this course.

In what follows, I will introduce some basic concepts in the statistical machine learning.

- Parametric models vs non-parametric models
- Training data vs testing data
- Bias and variance tradeoff
- Model validation

Parametric models v.s. Non-parametric models

- **Parametric models**: Situations like linear regression, in which we can describe the functional form of $f(x)$ using *a fixed number of parameters* are called parametric models. Like

$$f(x) = \beta_0 + \beta^T x$$

Parametric models v.s. Non-parametric models

- **Parametric models**: Situations like linear regression, in which we can describe the functional form of $f(x)$ using *a fixed number of parameters* are called parametric models. Like

$$f(x) = \beta_0 + \beta^T x$$

- Once we know assume the parametric form of f, the estimation of f reduces to estimating the parameters β_0 and β.

Parametric models v.s. Non-parametric models

- **Non-Parametric models**: Simply, a model that is not parametric. There are many different interpretations to this statement, e.g., having an increasing number of parameters.

Parametric models v.s. Non-parametric models

- **Non-Parametric models**: Simply, a model that is not parametric. There are many different interpretations to this statement, e.g., having an increasing number of parameters.
- For example, the value of k in the K-nearest neighbor classifier that grows as you see more and more data. Other examples include the depth in decision tree, and the number of layers and the width in deep neural networks.

Parametric models v.s. Non-parametric models

- **Non-Parametric models**: Simply, a model that is not parametric. There are many different interpretations to this statement, e.g., having an increasing number of parameters.
- For example, the value of k in the K-nearest neighbor classifier that grows as you see more and more data. Other examples include the depth in decision tree, and the number of layers and the width in deep neural networks.
- In this course, a non-parametric models is one that does not make explicit assumptions about the form of f.

Training dataset v.s. Testing dataset

- **Training dataset**: data used to fit a model

Training dataset v.s. Testing dataset

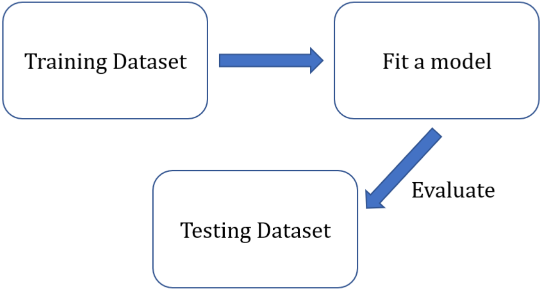
- **Training dataset**: data used to fit a model
- **Testing dataset**: data that were NOT used in the fitting process, but are used to test how well your model performs on unseen data.

Training dataset v.s. Testing dataset

- **Training dataset**: data used to fit a model
- **Testing dataset**: data that were NOT used in the fitting process, but are used to test how well your model performs on unseen data.

Training dataset v.s. Testing dataset

- **Training dataset**: data used to fit a model
- **Testing dataset**: data that were NOT used in the fitting process, but are used to test how well your model performs on unseen data.



Let us take regression as an example

- **Predictors/feature/covariate**: $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)$ is p-dimensional random variable

Let us take regression as an example

- **Predictors/feature/covariate**: $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)$ is p-dimensional random variable
- **Response/label**: Y is a quantitative random variable. Generally, Y is something we want to predict, say this image is a cat or dog or your starting salary after graduation.

Let us take regression as an example

- **Predictors/feature/covariate**: $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)$ is p-dimensional random variable

- **Response/label**: Y is a quantitative random variable. Generally, Y is something we want to predict, say this image is a cat or dog or your starting salary after graduation.

- **The relationship between $\boldsymbol{X}$ and** Y:

$$Y = f^*(\boldsymbol{X}) + \epsilon, \tag{1}$$

where $\mathbb{E}(\epsilon) = 0$ and $Var(\epsilon) = \sigma^2$.

Statistical machine learning for regression

- **Goal**: Find a function $f(\boldsymbol{X})$ for predicting Y (or approximate f^* well)
- **Loss function**: square loss

$$L(f(\boldsymbol{X}), Y) = (Y - f(\boldsymbol{X}))^2$$

Statistical machine learning for regression

- **Goal**: Find a function $f(\boldsymbol{X})$ for predicting Y (or approximate f^* well)
- **Loss function**: square loss

$$L(f(\boldsymbol{X}), Y) = (Y - f(\boldsymbol{X}))^2$$

- The averaged loss (expected error, also called as "risk") of f:

$$R(f) = \mathbb{E}_{\boldsymbol{X},Y}\left[L(f(\boldsymbol{X}), Y)\right] = \mathbb{E}\left[(Y - f(\boldsymbol{X}))^2\right]$$

Deeper look at the risk function R

- The expected squared loss can be written as

$$R(f) = \mathbb{E}\big[(Y - f(\boldsymbol{X}))^2\big] = \int\int (Y - f(\boldsymbol{X}))^2 \mathbb{P}(\boldsymbol{X}, Y) d\boldsymbol{X} dY.$$

Deeper look at the risk function R

- The expected squared loss can be written as

$$R(f) = \mathbb{E}\big[(Y - f(\boldsymbol{X}))^2\big] = \int\int (Y - f(\boldsymbol{X}))^2 \mathbb{P}(\boldsymbol{X}, Y) d\boldsymbol{X} dY.$$

- We can decompose $R(f)$ into

$$\mathbb{E}\big[(Y - f(\boldsymbol{X}))^2\big] = \int\int (Y - \mathbb{E}(Y|\boldsymbol{X}))^2 \mathbb{P}(\boldsymbol{X}, Y) d\boldsymbol{X} dY + \int\int (\mathbb{E}(Y|\boldsymbol{X}) - f(\boldsymbol{X}))^2 \mathbb{P}(\boldsymbol{X}, Y) d\boldsymbol{X} dY,$$

where $\mathbb{P}(\boldsymbol{X}, Y)$ is the joint distribution of $(\boldsymbol{X}, Y)$.

Deeper look at the risk function R

- The expected squared loss can be written as

$$R(f) = \mathbb{E}\big[(Y - f(\boldsymbol{X}))^2\big] = \int\int (Y - f(\boldsymbol{X}))^2 \mathbb{P}(\boldsymbol{X}, Y) d\boldsymbol{X} dY.$$

- We can decompose $R(f)$ into

$$\begin{aligned}\mathbb{E}\big[(Y - f(\boldsymbol{X}))^2\big] &= \int\int (Y - \mathbb{E}(Y|\boldsymbol{X}))^2 \mathbb{P}(\boldsymbol{X}, Y) d\boldsymbol{X} dY \\ &+ \int\int (\mathbb{E}(Y|\boldsymbol{X}) - f(\boldsymbol{X}))^2 \mathbb{P}(\boldsymbol{X}, Y) d\boldsymbol{X} dY,\end{aligned}$$

  where $\mathbb{P}(\boldsymbol{X}, Y)$ is the joint distribution of $(\boldsymbol{X}, Y)$.
- From the above decomp, we can tell $R(f)$ attains its minimum at

$$f^*(\boldsymbol{X}) = \mathbb{E}(Y|\boldsymbol{X}).$$

How to estimate f in practice?

- In practice, we do not know the exact from of $\mathbb{E}(Y|\boldsymbol{X})$.
- **Question**: What do we usually do?

How to estimate f in practice?

- In practice, we do not know the exact from of $\mathbb{E}(Y|\boldsymbol{X})$.
- **Question**: What do we usually do?
  - Impose a structure on f, for example

$$f(\boldsymbol{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p.$$

How to estimate f in practice?

- In practice, we do not know the exact from of $\mathbb{E}(Y|\boldsymbol{X})$.
- **Question**: What do we usually do?
  - Impose a structure on f, for example

$$f(\boldsymbol{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p.$$

  - Suppose a hypothesis space

$$\mathcal{F} = \{f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i : \beta_i \in \mathbb{R}, i = 0, \ldots, p\}$$

.

How to estimate f in practice?

- In practice, we do not know the exact from of $\mathbb{E}(Y|\boldsymbol{X})$.
- **Question**: What do we usually do?
  - Impose a structure on f, for example

$$f(\boldsymbol{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p.$$

  - Suppose a hypothesis space

$$\mathcal{F} = \{f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i : \beta_i \in \mathbb{R}, i = 0, \ldots, p\}$$

.

  - Minimize the averaged squared loss on a training dataset $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(\boldsymbol{x}_i) - y_i)^2$$

How to evaluate $\widehat{f}$: bias and variance tradeoff

- Based on the training dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, we obtain an estimator $\widehat{f}$

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(\boldsymbol{x}_i) - y_i)^2$$

How to evaluate $\widehat{f}$: bias and variance tradeoff

- Based on the training dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, we obtain an estimator $\widehat{f}$

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(\boldsymbol{x}_i) - y_i)^2$$

- Assess the quality of $\widehat{f}$ at $\boldsymbol{X} = \boldsymbol{x}_0$ (note that $Y = f^*(x_0) + \epsilon$):

$$\begin{aligned}
&\mathbb{E}_\epsilon\left[(\widehat{f}(\boldsymbol{X}) - Y)^2 | \boldsymbol{X} = \boldsymbol{x}_0\right] \\
=&\left[\widehat{f}(\boldsymbol{x}_0) - \mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}_0)\right]^2 + \mathbb{E}_\epsilon\left[Y - \mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}_0)\right]^2 \\
=&\underbrace{\left[\widehat{f}(\boldsymbol{x}_0) - \mathbb{E}_\epsilon(Y|\boldsymbol{X} = \boldsymbol{x}_0)\right]^2}_{Reducible} + \underbrace{\sigma^2}_{non-reducible}
\end{aligned}$$

How to evaluate $\widehat{f}$: bias and variance tradeoff

- Based on the training dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, we obtain an estimator $\widehat{f}$

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(\boldsymbol{x}_i) - y_i)^2$$

- Assess the quality of $\widehat{f}$ at $\boldsymbol{X} = \boldsymbol{x}_0$ (note that $Y = f^*(x_0) + \epsilon$):

$$\begin{aligned}
&\mathbb{E}_\epsilon\left[(\widehat{f}(\boldsymbol{X}) - Y)^2 | \boldsymbol{X} = \boldsymbol{x}_0\right] \\
=&\left[\widehat{f}(\boldsymbol{x}_0) - \mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}_0)\right]^2 + \mathbb{E}_\epsilon\left[Y - \mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}_0)\right]^2 \\
=&\underbrace{\left[\widehat{f}(\boldsymbol{x}_0) - \mathbb{E}_\epsilon(Y|\boldsymbol{X} = \boldsymbol{x}_0)\right]^2}_{Reducible} + \underbrace{\sigma^2}_{non-reducible}
\end{aligned}$$

- Here, $(\boldsymbol{x}_0, Y)$ is the testing dataset.

Bias and Variance tradeoff

- Reducible part can be decomposed into two components

$$
\begin{aligned}
&\mathbb{E}_D\left[\widehat{f}(\boldsymbol{x}_0) - \mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}_0)\right]^2 \\
=&\underbrace{\mathbb{E}_D\left[\widehat{f}(\boldsymbol{x}_0) - \mathbb{E}(\widehat{f}(\boldsymbol{X})\right]^2}_{Variance} + \underbrace{\left[\mathbb{E}_D(\widehat{f}(\boldsymbol{x}_0)) - \mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}_0)\right]^2}_{Bias^2},
\end{aligned}
$$

where the expectation is taken with respect to the training dataset D (that takes care of the randomness in $\widehat{f}$.)

Bias and Variance tradeoff

- Reducible part can be decomposed into two components

$$\begin{aligned}&\mathbb{E}_D\left[\widehat{f}(\boldsymbol{x}_0) - \mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}_0)\right]^2\\ =&\underbrace{\mathbb{E}_D\left[\widehat{f}(\boldsymbol{x}_0) - \mathbb{E}(\widehat{f}(\boldsymbol{X})\right]^2}_{Variance} + \underbrace{\left[\mathbb{E}_D(\widehat{f}(\boldsymbol{x}_0)) - \mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}_0)\right]^2}_{Bias^2},\end{aligned}$$

  where the expectation is taken with respect to the training dataset D (that takes care of the randomness in $\widehat{f}$.)

  - **Variance**: represents the variability of the predicted value. The randomness comes from the training dataset.

- Reducible part can be decomposed into two components

$$\mathbb{E}_D\left[\widehat{f}(\boldsymbol{x}_0) - \mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}_0)\right]^2$$
$$= \underbrace{\mathbb{E}_D\left[\widehat{f}(\boldsymbol{x}_0) - \mathbb{E}(\widehat{f}(\boldsymbol{X})\right]^2}_{Variance} + \underbrace{\left[\mathbb{E}_D(\widehat{f}(\boldsymbol{x}_0)) - \mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}_0)\right]^2}_{Bias^2},$$

where the expectation is taken with respect to the training dataset D (that takes care of the randomness in $\widehat{f}$.)

  - **Variance**: represents the variability of the predicted value. The randomness comes from the training dataset.
  - **Squared Bias**: The second term is the squared bias. If $\mathcal{F}$ is chosen well, so that the mean across all training data sets is the true function, then bias is 0.

Training MSE v.s. Testing MSE

- Let $D_r = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and $D_e = \{(\boldsymbol{x}_i', y_i')\}_{i=1}^m$ be training and testing datasets, respectively. Train an estimator from D_r

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(\boldsymbol{x}_i) - y_i)^2$$

Training MSE v.s. Testing MSE

- Let $D_r = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ and $D_e = \{(\boldsymbol{x}_i', y_i')\}_{i=1}^{m}$ be training and testing datasets, respectively. Train an estimator from D_r

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(\boldsymbol{x}_i) - y_i)^2$$

- Evaluate $\widehat{f}$ by the mean squared error (MSE):

$$\text{Training MSE} : \frac{1}{n} \sum_{i=1}^{n} (\widehat{f}(\boldsymbol{x}_i) - y_i)^2$$

$$\text{Testing MSE} : \frac{1}{m} \sum_{i=1}^{m} (\widehat{f}(\boldsymbol{x}_i') - y_i')^2$$

Training MSE v.s. Testing MSE

- Let $D_r = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and $D_e = \{(\boldsymbol{x}'_i, y'_i)\}_{i=1}^m$ be training and testing datasets, respectively. Train an estimator from D_r

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(\boldsymbol{x}_i) - y_i)^2$$

- Evaluate $\widehat{f}$ by the mean squared error (MSE):

$$\text{Training MSE} : \frac{1}{n} \sum_{i=1}^{n} (\widehat{f}(\boldsymbol{x}_i) - y_i)^2$$

$$\text{Testing MSE} : \frac{1}{m} \sum_{i=1}^{m} (\widehat{f}(\boldsymbol{x}'_i) - y'_i)^2$$

- **Question**: Which one can be used for assessing the quality of $\widehat{f}$?

An example.

- We generate $\{(x_i, y_i)\}_{i=1}^n$ in the following way
$$y_i = sin(x_i) + \epsilon_i$$

An example.

- We generate $\{(x_i, y_i)\}_{i=1}^n$ in the following way

$$y_i = sin(x_i) + \epsilon_i$$

  - $x_i \sim \text{Unif}(-2\pi, 2\pi)$

An example.

- We generate $\{(x_i, y_i)\}_{i=1}^{n}$ in the following way
$$y_i = sin(x_i) + \epsilon_i$$
    - $x_i \sim \mathrm{Unif}(-2\pi, 2\pi)$
    - $\epsilon \sim N(0, 0.5)$

An example.

- We generate $\{(x_i, y_i)\}_{i=1}^n$ in the following way

$$y_i = sin(x_i) + \epsilon_i$$

  - $x_i \sim \text{Unif}(-2\pi, 2\pi)$
  - $\epsilon \sim N(0, 0.5)$
  - Set $n = 30$

An example.

- We generate $\{(x_i, y_i)\}_{i=1}^n$ in the following way

$$y_i = sin(x_i) + \epsilon_i$$

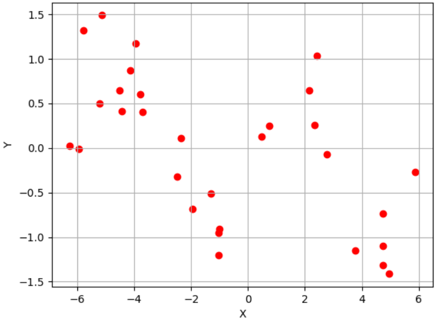
  - $x_i \sim \text{Unif}(-2\pi, 2\pi)$
  - $\epsilon \sim N(0, 0.5)$
  - Set $n = 30$

An example.

- We generate $\{(x_i, y_i)\}_{i=1}^{n}$ in the following way

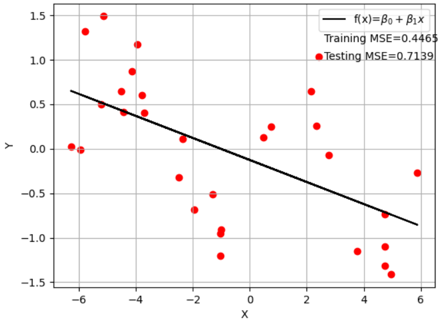
$$y_i = sin(x_i) + \epsilon_i$$

  - $x_i \sim \text{Unif}(-2\pi, 2\pi)$
  - $\epsilon \sim N(0, 0.5)$
  - Set $n = 30$



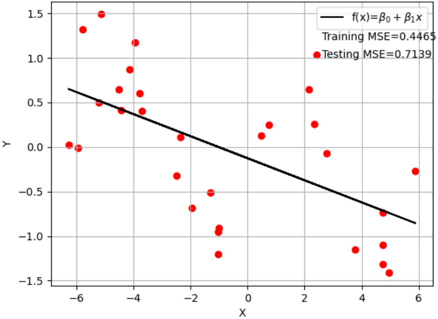
An example: linear regression model

- We fit a linear model $f(x) = \beta_0 + \beta_1 x$



An example: linear regression model

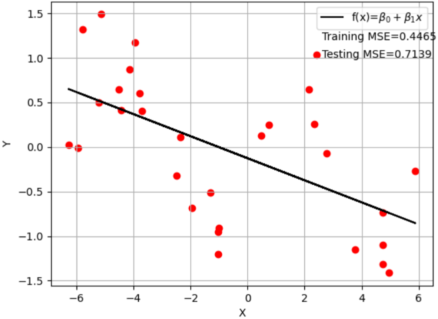
- We fit a linear model $f(x) = \beta_0 + \beta_1 x$



- Training MSE is 0.4465

An example: linear regression model

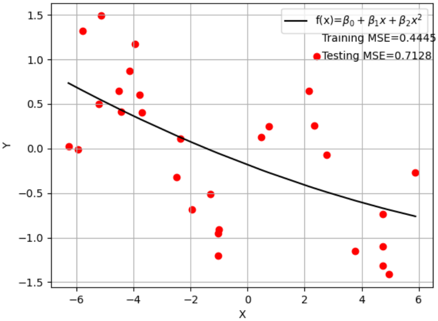
- We fit a linear model $f(x) = \beta_0 + \beta_1 x$



- Training MSE is 0.4465
- Testing MSE is 0.7139

An example: quadratic regression

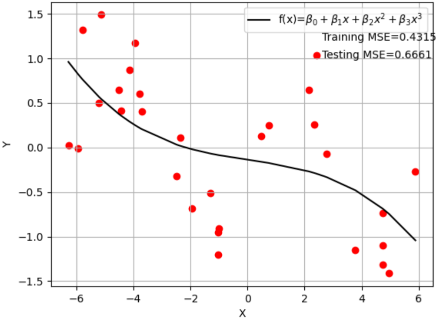
- We fit a quadratic model $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2$



- Training MSE is 0.4445 (improve 0.0020)
- Testing MSE is 0.7128 (improve by 0.0019)

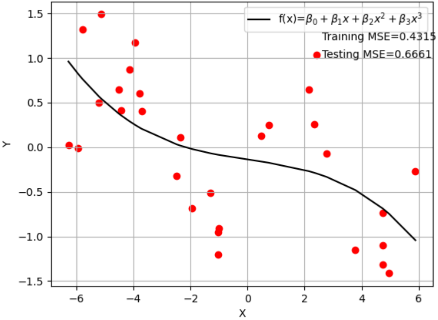
An example: polynomial regression

- We fit a polynomial model (with order 3)
  $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$



An example: polynomial regression

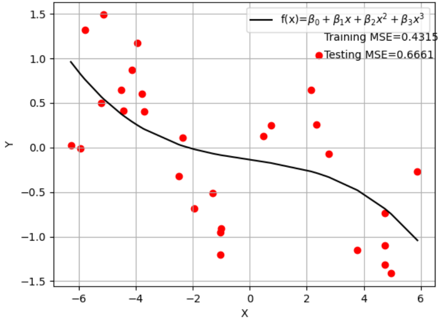
- We fit a polynomial model (with order 3) $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$



- Training MSE is 0.4315 (improve 0.0130)

An example: polynomial regression

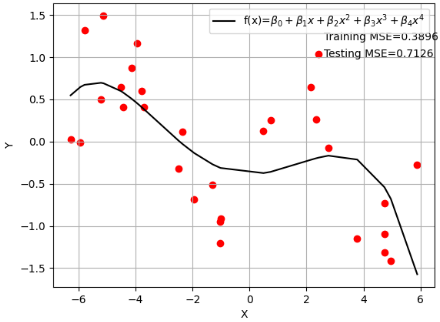
- We fit a polynomial model (with order 3) $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$



- Training MSE is 0.4315 (improve 0.0130)
- Testing MSE is 0.6661(improve by 0.0467)

An example: higher order polynomial

- We fit a polynomial model (with order 4)$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$



- Training MSE is 0.3896 (improve 0.0419)
- Testing MSE is 0.7126 (increase by 0.0464): start fitting noise ratehr than signal....

An example: conclusion

Metrics	Model 1	Model 2	Model 3	Model 4
Training MSE	0.4465	0.4445	0.4315	0.3896
Testing MSE	0.7139	0.7128	0.6661	0.7126

- **Conclusions**:

An example: conclusion

Metrics	Model 1	Model 2	Model 3	Model 4
Training MSE	0.4465	0.4445	0.4315	0.3896
Testing MSE	0.7139	0.7128	0.6661	0.7126

- **Conclusions**:
  (1) Training MSE is non-increasing with respect to the flexibility of model, i.e., as training model becomes more flexible, training MSE always becomes smaller.

An example: conclusion

Metrics	Model 1	Model 2	Model 3	Model 4
Training MSE	0.4465	0.4445	0.4315	0.3896
Testing MSE	0.7139	0.7128	0.6661	0.7126

- **Conclusions**:
  (1) Training MSE is non-increasing with respect to the flexibility of model, i.e., as training model becomes more flexible, training MSE always becomes smaller.
  (2) Testing MSE (*which is what we really care*) decreases first and then increases with respect to the flexibility of model.

An example: conclusion

Metrics	Model 1	Model 2	Model 3	Model 4
Training MSE	0.4465	0.4445	0.4315	0.3896
Testing MSE	0.7139	0.7128	0.6661	0.7126

- **Conclusions**:
  (1) Training MSE is non-increasing with respect to the flexibility of model, i.e., as training model becomes more flexible, training MSE always becomes smaller.
  (2) Testing MSE (*which is what we really care*) decreases first and then increases with respect to the flexibility of model.
- **The behavior of Testing MSE**: Bias-variance trade-off

An example: conclusion

Metrics	Model 1	Model 2	Model 3	Model 4
Training MSE	0.4465	0.4445	0.4315	0.3896
Testing MSE	0.7139	0.7128	0.6661	0.7126

- **Conclusions**:
  (1) Training MSE is non-increasing with respect to the flexibility of model, i.e., as training model becomes more flexible, training MSE always becomes smaller.
  (2) Testing MSE (*which is what we really care*) decreases first and then increases with respect to the flexibility of model.
- **The behavior of Testing MSE**: Bias-variance trade-off
  (1) Models with greater flexibility have a smaller bias.

An example: conclusion

Metrics	Model 1	Model 2	Model 3	Model 4
Training MSE	0.4465	0.4445	0.4315	0.3896
Testing MSE	0.7139	0.7128	0.6661	0.7126

- **Conclusions**:
  (1) Training MSE is non-increasing with respect to the flexibility of model, i.e., as training model becomes more flexible, training MSE always becomes smaller.
  (2) Testing MSE (*which is what we really care*) decreases first and then increases with respect to the flexibility of model.
- **The behavior of Testing MSE**: Bias-variance trade-off
  (1) Models with greater flexibility have a smaller bias.
  (2) More flexible methods have a greater variance

How to choose a proper parametric model?

- So far, we have discussed how to fit a statistical machine learning model properly, say strike the trade-off between bias and variance;

How to choose a proper parametric model?

- So far, we have discussed how to fit a statistical machine learning model properly, say strike the trade-off between bias and variance;
- All discussions rely on the choice/assumption of models to be fit;

How to choose a proper parametric model?

- So far, we have discussed how to fit a statistical machine learning model properly, say strike the trade-off between bias and variance;
- All discussions rely on the choice/assumption of models to be fit;
- However, in practice, we do need to choose the models to be fit. For example, shall we choose f as linear function vs quadratic function vs cubic function?

How to choose a proper parametric model?

- So far, we have discussed how to fit a statistical machine learning model properly, say strike the trade-off between bias and variance;
- All discussions rely on the choice/assumption of models to be fit;
- However, in practice, we do need to choose the models to be fit. For example, shall we choose f as linear function vs quadratic function vs cubic function?
- General questions: how to select the best fitting model from a bunch of candidate models?

How to choose a proper parametric model?

- So far, we have discussed how to fit a statistical machine learning model properly, say strike the trade-off between bias and variance;
- All discussions rely on the choice/assumption of models to be fit;
- However, in practice, we do need to choose the models to be fit. For example, shall we choose f as linear function vs quadratic function vs cubic function?
- General questions: how to select the best fitting model from a bunch of candidate models?
- Use model validation!

Model validation

- **Training Set**: The training set is a subset of the dataset used to train the machine learning model. The model learns patterns, relationships, and features from this set.

Model validation

- **Validation Set**: The validation set is a separate subset of data that is not used for training the model. It is used during the training phase to assess the model's performance on data it has not seen before.

Model validation

- **Test Set**: The test set (or holdout set) is another independent subset of data that is not used during training or validation. It is reserved for the final evaluation of the model's performance after training is complete.
- In other words, we have three datasets: training, validation and testing.

Validation dataset

- **Validation dataset** is a sample of data held back from training your model that is used to give an *estimate of model performance* given the current model's hyperparameters, e.g., order of polynomial in f.

Validation split

- **Definition**: Randomly split the dataset into two parts: one for training and the other one for validating. The error on validation dataset can be viewed as an estimate of **testing error**.

Validation split

- **Definition**: Randomly split the dataset into two parts: one for training and the other one for validating. The error on validation dataset can be viewed as an estimate of **testing error**.
- **Practice**: Split the raw dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ at 80:20 or 66:34 ratio.

Validation split

- **Definition**: Randomly split the dataset into two parts: one for training and the other one for validating. The error on validation dataset can be viewed as an estimate of **testing error**.
- **Practice**: Split the raw dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ at 80:20 or 66:34 ratio.

Validation split

- **Definition**: Randomly split the dataset into two parts: one for training and the other one for validating. The error on validation dataset can be viewed as an estimate of **testing error**.
- **Practice**: Split the raw dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ at 80:20 or 66:34 ratio.
- Split $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ into $D_r = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ and $D_v = \{(\boldsymbol{x}_i, y_i)\}_{i=m+1}^{n}$.

Validation split

- **Definition**: Randomly split the dataset into two parts: one for training and the other one for validating. The error on validation dataset can be viewed as an estimate of **testing error**.
- **Practice**: Split the raw dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ at 80:20 or 66:34 ratio.
- Split $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ into $D_r = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ and $D_v = \{(\boldsymbol{x}_i, y_i)\}_{i=m+1}^n$.
    - D_r: training dataset (Fit your model using this dataset)

Validation split

- **Definition**: Randomly split the dataset into two parts: one for training and the other one for validating. The error on validation dataset can be viewed as an estimate of **testing error**.
- **Practice**: Split the raw dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ at 80:20 or 66:34 ratio.
- Split $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ into $D_r = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ and $D_v = \{(\boldsymbol{x}_i, y_i)\}_{i=m+1}^n$.
  - D_r: training dataset (Fit your model using this dataset)
  - D_v: validation dataset (Evaluate your model using this dataset)

Simulation: how well validation estimates testing error?

- The diabetes dataset consists of 768 samples.

Simulation: how well validation estimates testing error?

- The diabetes dataset consists of 768 samples.
- Use 500 samples for training and 268 samples as testing data

Simulation: how well validation estimates testing error?

- The diabetes dataset consists of 768 samples.
- Use 500 samples for training and 268 samples as testing data
- The true test error (testing MSE) is 0.2350 evaluated using the 268 testing samples

Simulation

- We repeat the following experiment multiple times to see how well the validation error estimates the true testing error: 0.2350

Simulation

- We repeat the following experiment multiple times to see how well the validation error estimates the true testing error: 0.2350
- For each repetition, we further split the training dataset into "training" (350 samples) and validation (150 samples)

Simulation

- We repeat the following experiment multiple times to see how well the validation error estimates the true testing error: 0.2350
- For each repetition, we further split the training dataset into "training" (350 samples) and validation (150 samples)
- Fit a logistic regression model (the details will be covered in Week 2) based on "training" set (350)

Simulation

- We repeat the following experiment multiple times to see how well the validation error estimates the true testing error: 0.2350
- For each repetition, we further split the training dataset into "training" (350 samples) and validation (150 samples)
- Fit a logistic regression model (the details will be covered in Week 2) based on "training" set (350)
- Compute the **validation error** (i.e., testing error computed based on the validation set) and compare it with the true testing error

Simulation

If we try training-validation split multiple times.

```
0.3133333 0.2350746
0.3000000 0.2350746
0.2600000 0.2350746
0.2933333 0.2350746
0.2800000 0.2350746
0.2666667 0.2350746
0.3466667 0.2350746
0.2933333 0.2350746
0.3200000 0.2350746
0.2666667 0.2350746
0.3000000 0.2350746
0.3066667 0.2350746
0.2933333 0.2350746
0.2800000 0.2350746
0.2866667 0.2350746
```

- **Left**: Validation error and **Right**: True Testing error

Conclusion:

- Disadvantages of this approach:

  (1) the **validation error** (that supposed to approximate the test error) is **highly variable**, depending on how you split the dataset.

  (2) In the validation approach, only a subset of training set are used to fit the model (350 out of 500). Since statistical methods tend to perform worse when trained on fewer observations, this suggests that the validation error tends to **overestimate** the test error since the training set and testing set are often larger.

Solution: cross-validation

- Cross validation: repeat the training validation split multiple times

Solution: cross-validation

- Cross validation: repeat the training validation split multiple times
- **Objective**: Estimate the test error associated with a given statistical learning method in order to

Solution: cross-validation

- Cross validation: repeat the training validation split multiple times
- **Objective**: Estimate the test error associated with a given statistical learning method in order to
  - evaluate its performance with less variability (model assessment)

Solution: cross-validation

- Cross validation: repeat the training validation split multiple times
- **Objective**: Estimate the test error associated with a given statistical learning method in order to
  - evaluate its performance with less variability (model assessment)
  - select the appropriate level of flexibility (select the best model or parameters)

Solution: cross-validation

- Cross validation: repeat the training validation split multiple times
- **Objective**: Estimate the test error associated with a given statistical learning method in order to
    - evaluate its performance with less variability (model assessment)
    - select the appropriate level of flexibility (select the best model or parameters)
- **Mechanism**: holding out a subset of the training observations from the fitting process and then applying the fitted model to those held out observations.

K-Fold Cross-Validation

- **K-Fold Cross-Validation**: The dataset is divided into K subsets (or folds). The model is trained on K-1 folds and validated on the remaining fold. This process is repeated K times, with each fold serving as the validation set exactly once.

Stratified K-Fold Cross-Validation

- **Stratified K-Fold Cross-Validation**: Similar to K-Fold, but the data is divided into K folds while ensuring that each fold maintains the same class distribution as the original dataset. This is particularly useful for imbalanced datasets (to be covered in week 5).

Leave-One-Out Cross-Validation

- **Leave-One-Out Cross-Validation (LOOCV)**: In LOOCV, only one data point is used for validation, and the model is trained on the remaining data. This process is repeated for each data point in the dataset. So, if there are 100 datapoints, we will repeat this procedure 100 times.

Leave-One-Out Cross-Validation

- **Leave-One-Out Cross-Validation (LOOCV)**: In LOOCV, only one data point is used for validation, and the model is trained on the remaining data. This process is repeated for each data point in the dataset. So, if there are 100 datapoints, we will repeat this procedure 100 times.
- Computationally expensive (or even infeasible) when the number of observations in the training data is large. Except if you are using linear regression (where an explicit formula is available).

Leave-One-Out Cross-Validation

- **Leave-One-Out Cross-Validation (LOOCV)**: In LOOCV, only one data point is used for validation, and the model is trained on the remaining data. This process is repeated for each data point in the dataset. So, if there are 100 datapoints, we will repeat this procedure 100 times.
- Computationally expensive (or even infeasible) when the number of observations in the training data is large. Except if you are using linear regression (where an explicit formula is available).
- The validation MSE from LOOCV is based on averaging n individual fold-based error estimates. Each of these individual estimates is based on almost the same data. Therefore, these estimates are highly correlated with each other.

K-Fold Cross-Validation: implementation details

- Divide the sample data into k parts.

K-Fold Cross-Validation: implementation details

- Divide the sample data into k parts.
- Use $k - 1$ of the parts for training, and 1 for testing.

K-Fold Cross-Validation: implementation details

- Divide the sample data into k parts.
- Use $k - 1$ of the parts for training, and 1 for testing.
- Repeat the procedure k times, rotating the test set.

K-Fold Cross-Validation: implementation details

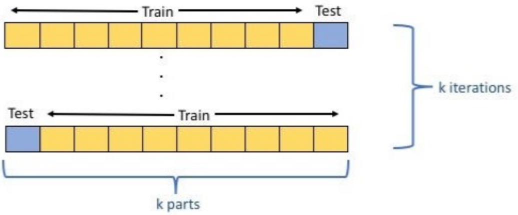
- Divide the sample data into k parts.
- Use $k-1$ of the parts for training, and 1 for testing.
- Repeat the procedure k times, rotating the test set.
- Determine an expected performance metric (mean square error, misclassfication error rate, confidence interval, or other appropriate metric) based on the results across the iterations.

K-Fold Cross-Validation: implementation details

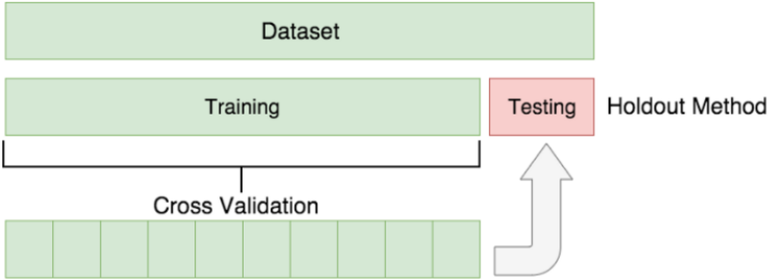
- Divide the sample data into k parts.
- Use $k-1$ of the parts for training, and 1 for testing.
- Repeat the procedure k times, rotating the test set.
- Determine an expected performance metric (mean square error, misclassfication error rate, confidence interval, or other appropriate metric) based on the results across the iterations.

K-Fold Cross-Validation: implementation details

- Divide the sample data into k parts.
- Use $k-1$ of the parts for training, and 1 for testing.
- Repeat the procedure k times, rotating the test set.
- Determine an expected performance metric (mean square error, misclassfication error rate, confidence interval, or other appropriate metric) based on the results across the iterations.



The complete picture on training, validation and testing



An example: 6-fold Cross-Validation

1 Suppose a dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{6n}$ is given. We split it into 6 parts

$$D_1 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}, D_2 = \{(\boldsymbol{x}_i, y_i)\}_{i=n+1}^{2n}, D_2 = \{(\boldsymbol{x}_i, y_i)\}_{i=2n+1}^{3n}$$
$$D_4 = \{(\boldsymbol{x}_i, y_i)\}_{i=3n+1}^{4n}, D_5 = \{(\boldsymbol{x}_i, y_i)\}_{i=4n+1}^{5n}, D_6 = \{(\boldsymbol{x}_i, y_i)\}_{i=5n+1}^{6n}$$

An example: 6-fold Cross-Validation

1. Suppose a dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{6n}$ is given. We split it into 6 parts

$$D_1 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}, D_2 = \{(\boldsymbol{x}_i, y_i)\}_{i=n+1}^{2n}, D_2 = \{(\boldsymbol{x}_i, y_i)\}_{i=2n+1}^{3n}$$
$$D_4 = \{(\boldsymbol{x}_i, y_i)\}_{i=3n+1}^{4n}, D_5 = \{(\boldsymbol{x}_i, y_i)\}_{i=4n+1}^{5n}, D_6 = \{(\boldsymbol{x}_i, y_i)\}_{i=5n+1}^{6n}$$

2. We repeat the following step from $j = 1, 2, 3, 4, 5, 6$: (1) Construct a dataset $D_{-j} = \cup_{i \neq j} D_i$; (2) Train a function via

$$\widehat{f}_{-j} = \arg\min_{f \in \mathcal{F}} \frac{1}{5n} \sum_{i \in D_{-j}} (f(\boldsymbol{x}_i) - y_i)^2$$

An example: 6-fold Cross-Validation

1 Suppose a dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{6n}$ is given. We split it into 6 parts

$$D_1 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}, D_2 = \{(\boldsymbol{x}_i, y_i)\}_{i=n+1}^{2n}, D_2 = \{(\boldsymbol{x}_i, y_i)\}_{i=2n+1}^{3n}$$
$$D_4 = \{(\boldsymbol{x}_i, y_i)\}_{i=3n+1}^{4n}, D_5 = \{(\boldsymbol{x}_i, y_i)\}_{i=4n+1}^{5n}, D_6 = \{(\boldsymbol{x}_i, y_i)\}_{i=5n+1}^{6n}$$

2 We repeat the following step from $j = 1, 2, 3, 4, 5, 6$: (1) Construct a dataset $D_{-j} = \cup_{i \neq j} D_i$; (2) Train a function via

$$\widehat{f}_{-j} = \arg\min_{f \in \mathcal{F}} \frac{1}{5n} \sum_{i \in D_{-j}} (f(\boldsymbol{x}_i) - y_i)^2$$

3 Compute the validation error of $\widehat{f}_{-j}, j = 1, \ldots, 6$

$$VE_j(\widehat{f}_{-j}) = \frac{1}{n} \sum_{i=(j-1)n+1}^{jn} (\widehat{f}_{-j}(\boldsymbol{x}_i) - y_i)^2$$

6-fold Cross-Validation: Procedure

4 Use the averaged validation errors as an estimate of testing error

$$\text{Testing Error Estimate} : \frac{1}{6}\sum_{j=1}^{6} VE_j(\widehat{f}_{-j})$$