Econ 425 Week 1

Machine Learning Pipeline

Grigory Franguridi

UCLA Econ

USC CESR

franguri@usc.edu

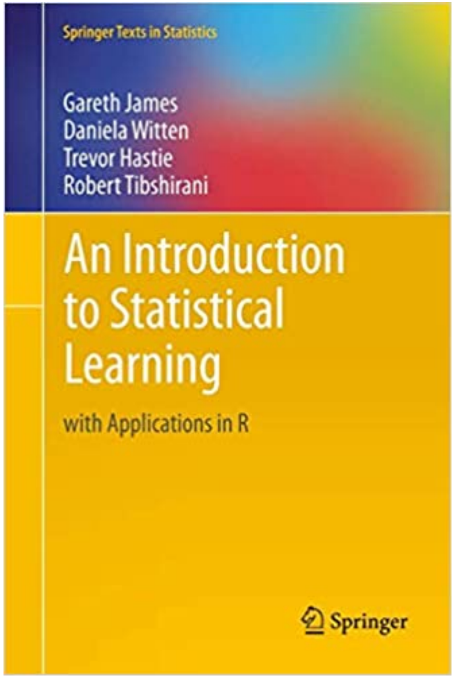
Our goals

- Learn basic methods and concepts of ML
- Apply ML to solve real-world problems

Our goals

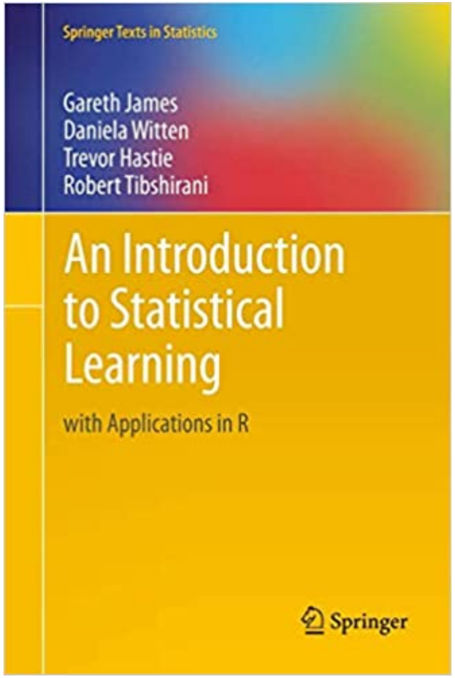
- Learn basic methods and concepts of ML
- Apply ML to solve real-world problems

Recommended textbook



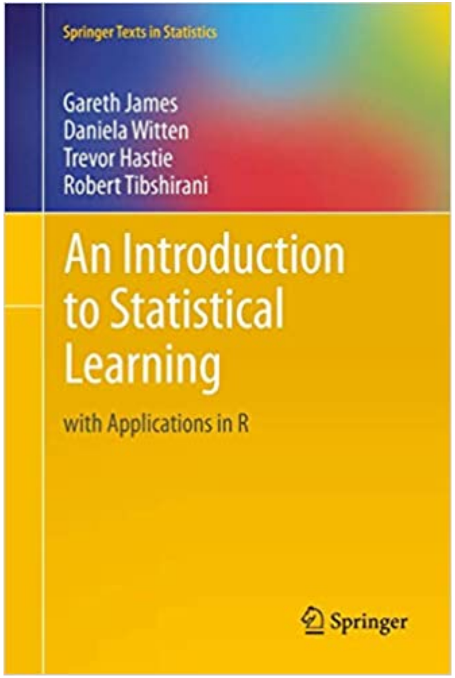
- Free to download: https://www.statlearning.com/
- Cover weeks 1, 2, 3, 4, 5, 6, 8, and 9
- R or Python

Recommended textbook



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

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

- **Grigory Franguridi**
  - Email: franguri@usc.edu
  - Class time: MW 9:00-10:15 @Bunche Hall 2221E
  - Office hours: W 10:15–12:15 @Bunche Hall 3169
  - During office hours, I expect questions mostly on lecture notes
  - lecture notes uploaded before class on Mon

Course instructor

- **Grigory Franguridi**
  - Email: franguri@usc.edu
  - Class time: MW 9:00-10:15 @Bunche Hall 2221E
  - Office hours: W 10:15–12:15 @Bunche Hall 3169
  - During office hours, I expect questions mostly on lecture notes
  - lecture notes uploaded before class on Mon

Course instructor

- **Grigory Franguridi**
  - Email: franguri@usc.edu
  - Class time: MW 9:00-10:15 @Bunche Hall 2221E
  - Office hours: W 10:15–12:15 @Bunche Hall 3169
  - During office hours, I expect questions mostly on lecture notes
  - lecture notes uploaded before class on Mon

Course instructor

- **Grigory Franguridi**
  - Email: franguri@usc.edu
  - Class time: MW 9:00-10:15 @Bunche Hall 2221E
  - Office hours: W 10:15–12:15 @Bunche Hall 3169
  - During office hours, I expect questions mostly on lecture notes
  - lecture notes uploaded before class on Mon

Course instructor

- **Grigory Franguridi**
  - Email: franguri@usc.edu
  - Class time: MW 9:00-10:15 @Bunche Hall 2221E
  - Office hours: W 10:15–12:15 @Bunche Hall 3169
  - During office hours, I expect questions mostly on lecture notes
  - lecture notes uploaded before class on Mon

Course instructor

- **Grigory Franguridi**
  - Email: franguri@usc.edu
  - Class time: MW 9:00-10:15 @Bunche Hall 2221E
  - Office hours: W 10:15–12:15 @Bunche Hall 3169
  - During office hours, I expect questions mostly on lecture notes
  - lecture notes uploaded before class on Mon

Lab instructor

- **Samuel Borghese**
  - Email: srborghese@g.ucla.edu
  - Lab time: each Wed 2:00-3:15pm (teaching) & 3:30pm-4:45pm (classwork / lab application) @Bunche 2221E
  - Office hours: 11:00-12:30 each Thursday @Bunche 4353
  - Please address questions on lab tasks and programming to Sam

Lab instructor

- **Samuel Borghese**
  - Email: srborghese@g.ucla.edu
  - Lab time: each Wed 2:00-3:15pm (teaching) & 3:30pm-4:45pm (classwork / lab application) @Bunche 2221E
  - Office hours: 11:00-12:30 each Thursday @Bunche 4353
  - Please address questions on lab tasks and programming to Sam

Lab instructor

- **Samuel Borghese**
  - Email: srborghese@g.ucla.edu
  - Lab time: each Wed 2:00-3:15pm (teaching) & 3:30pm-4:45pm (classwork / lab application) @Bunche 2221E
  - Office hours: 11:00-12:30 each Thursday @Bunche 4353
  - Please address questions on lab tasks and programming to Sam

Lab instructor

- **Samuel Borghese**
  - Email: srborghese@g.ucla.edu
  - Lab time: each Wed 2:00-3:15pm (teaching) & 3:30pm-4:45pm (classwork / lab application) @Bunche 2221E
  - Office hours: 11:00-12:30 each Thursday @Bunche 4353
  - Please address questions on lab tasks and programming to Sam

Tutor

- **Agarwal Tripti**
  - Email: triptiagarwal@g.ucla.edu
  - Office hour: 11:00-1:00 each Tuesday @Bunche 2221E
  - also available to students via email and/or additional in-person meetings to answer any questions regarding coursework, homeworks, conceptual help, programming etc.

Grader (special reader)

- **TBD**
    - Email: TBD
    - Please refer any questions about grading to the grader

Grading scheme

- **Homework**: 60%
  - 8 homework assignments (except for weeks 7, 10)
  - assigned on Thursdays 6pm
  - due one week after, i.e. next Thursday 6pm
  - announcement/submission in Bruinlearn
  - late submissions (except compelling reasons such as doctor note): within 3 days of due time: 70%*score; late more than 3 days: zero
  - graded within one week after due time
- **Lab assignment**: 20% Lab Grade (8 HW/CW assignments)
- **Final exam**: 20% (in-person, open book, 60 mins in the final week)

Grading scheme

- **Homework**: 60%
  - 8 homework assignments (except for weeks 7, 10)
  - assigned on Thursdays 6pm
  - due one week after, i.e. next Thursday 6pm
  - announcement/submission in Bruinlearn
  - late submissions (except compelling reasons such as doctor note): within 3 days of due time: 70%*score; late more than 3 days: zero
  - graded within one week after due time
- **Lab assignment**: 20% Lab Grade (8 HW/CW assignments)
- **Final exam**: 20% (in-person, open book, 60 mins in the final week)

Grading scheme

- **Homework**: 60%
  - 8 homework assignments (except for weeks 7, 10)
  - assigned on Thursdays 6pm
  - due one week after, i.e. next Thursday 6pm
  - announcement/submission in Bruinlearn
  - late submissions (except compelling reasons such as doctor note): within 3 days of due time: 70%*score; late more than 3 days: zero
  - graded within one week after due time
- **Lab assignment**: 20% Lab Grade (8 HW/CW assignments)
- **Final exam**: 20% (in-person, open book, 60 mins in the final week)

Grading scheme

- **Homework**: 60%
  - 8 homework assignments (except for weeks 7, 10)
  - assigned on Thursdays 6pm
  - due one week after, i.e. next Thursday 6pm
  - announcement/submission in Bruinlearn
  - late submissions (except compelling reasons such as doctor note): within 3 days of due time: 70%*score; late more than 3 days: zero
  - graded within one week after due time
- **Lab assignment**: 20% Lab Grade (8 HW/CW assignments)
- **Final exam**: 20% (in-person, open book, 60 mins in the final week)

Grading scheme

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

    - assigned on Thursdays 6pm
    - due one week after, i.e. next Thursday 6pm
    - announcement/submission in Bruinlearn
    - late submissions (except compelling reasons such as doctor note): within 3 days of due time: 70%*score; late more than 3 days: zero
    - graded within one week after due time

- **Lab assignment**: 20% Lab Grade (8 HW/CW assignments)
- **Final exam**: 20% (in-person, open book, 60 mins in the final week)

Grading scheme

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

  - assigned on Thursdays 6pm
  - due one week after, i.e. next Thursday 6pm
  - announcement/submission in Bruinlearn
  - late submissions (except compelling reasons such as doctor note): within 3 days of due time: 70%*score; late more than 3 days: zero
  - graded within one week after due time

- **Lab assignment**: 20% Lab Grade (8 HW/CW assignments)
- **Final exam**: 20% (in-person, open book, 60 mins in the final week)

Grading scheme

- **Homework**: 60%
  - 8 homework assignments (except for weeks 7, 10)
  - assigned on Thursdays 6pm
  - due one week after, i.e. next Thursday 6pm
  - announcement/submission in Bruinlearn
  - late submissions (except compelling reasons such as doctor note): within 3 days of due time: 70%*score; late more than 3 days: zero
  - graded within one week after due time
- **Lab assignment**: 20% Lab Grade (8 HW/CW assignments)
- **Final exam**: 20% (in-person, open book, 60 mins in the final week)

Grading scheme

- **Homework**: 60%
  - 8 homework assignments (except for weeks 7, 10)
  - assigned on Thursdays 6pm
  - due one week after, i.e. next Thursday 6pm
  - announcement/submission in Bruinlearn
  - late submissions (except compelling reasons such as doctor note): within 3 days of due time: 70%*score; late more than 3 days: zero
  - graded within one week after due time
- **Lab assignment**: 20% Lab Grade (8 HW/CW assignments)
- **Final exam**: 20% (in-person, open book, 60 mins in the final week)

Grading scheme

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

  - assigned on Thursdays 6pm
  - due one week after, i.e. next Thursday 6pm
  - announcement/submission in Bruinlearn
  - late submissions (except compelling reasons such as doctor note): within 3 days of due time: 70%*score; late more than 3 days: zero
  - graded within one week after due time
- **Lab assignment**: 20% Lab Grade (8 HW/CW assignments)
- **Final exam**: 20% (in-person, open book, 60 mins in the final week)

Grading scheme

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

  - assigned on Thursdays 6pm
  - due one week after, i.e. next Thursday 6pm
  - announcement/submission in Bruinlearn
  - late submissions (except compelling reasons such as doctor note): within 3 days of due time: 70%*score; late more than 3 days: zero
  - graded within one week after due time

- **Lab assignment**: 20% Lab Grade (8 HW/CW assignments)
- **Final exam**: 20% (in-person, open book, 60 mins in the final week)

Syllabus

- Week 1: 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: Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models
- Week 8: Bagging and Boosting
- Week 9: Unsupervised Learning - Clustering & PCA
- Week 10: Reinforcement Learning

Syllabus

- Week 1: 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: Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models
- Week 8: Bagging and Boosting
- Week 9: Unsupervised Learning - Clustering & PCA
- Week 10: Reinforcement Learning

Syllabus

- Week 1: 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: Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models
- Week 8: Bagging and Boosting
- Week 9: Unsupervised Learning - Clustering & PCA
- Week 10: Reinforcement Learning

Syllabus

- Week 1: 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: Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models
- Week 8: Bagging and Boosting
- Week 9: Unsupervised Learning - Clustering & PCA
- Week 10: Reinforcement Learning

Syllabus

- Week 1: 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: Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models
- Week 8: Bagging and Boosting
- Week 9: Unsupervised Learning - Clustering & PCA
- Week 10: Reinforcement Learning

Syllabus

- Week 1: 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: Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models
- Week 8: Bagging and Boosting
- Week 9: Unsupervised Learning - Clustering & PCA
- Week 10: Reinforcement Learning

Syllabus

- Week 1: 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: Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models
- Week 8: Bagging and Boosting
- Week 9: Unsupervised Learning - Clustering & PCA
- Week 10: Reinforcement Learning

Syllabus

- Week 1: 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: Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models
- Week 8: Bagging and Boosting
- Week 9: Unsupervised Learning - Clustering & PCA
- Week 10: Reinforcement Learning

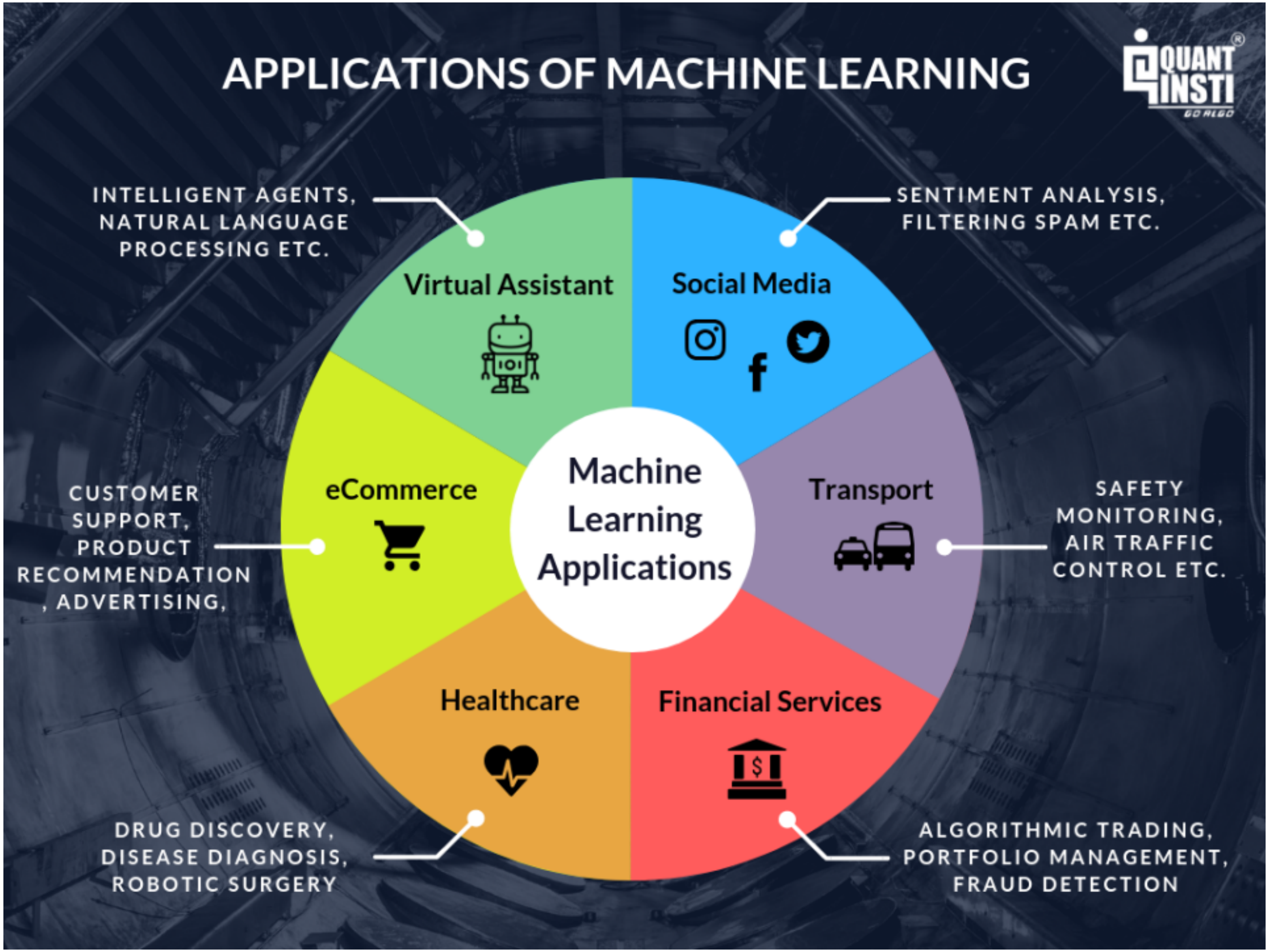
Syllabus

- Week 1: 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: Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models
- Week 8: Bagging and Boosting
- Week 9: Unsupervised Learning - Clustering & PCA
- Week 10: Reinforcement Learning

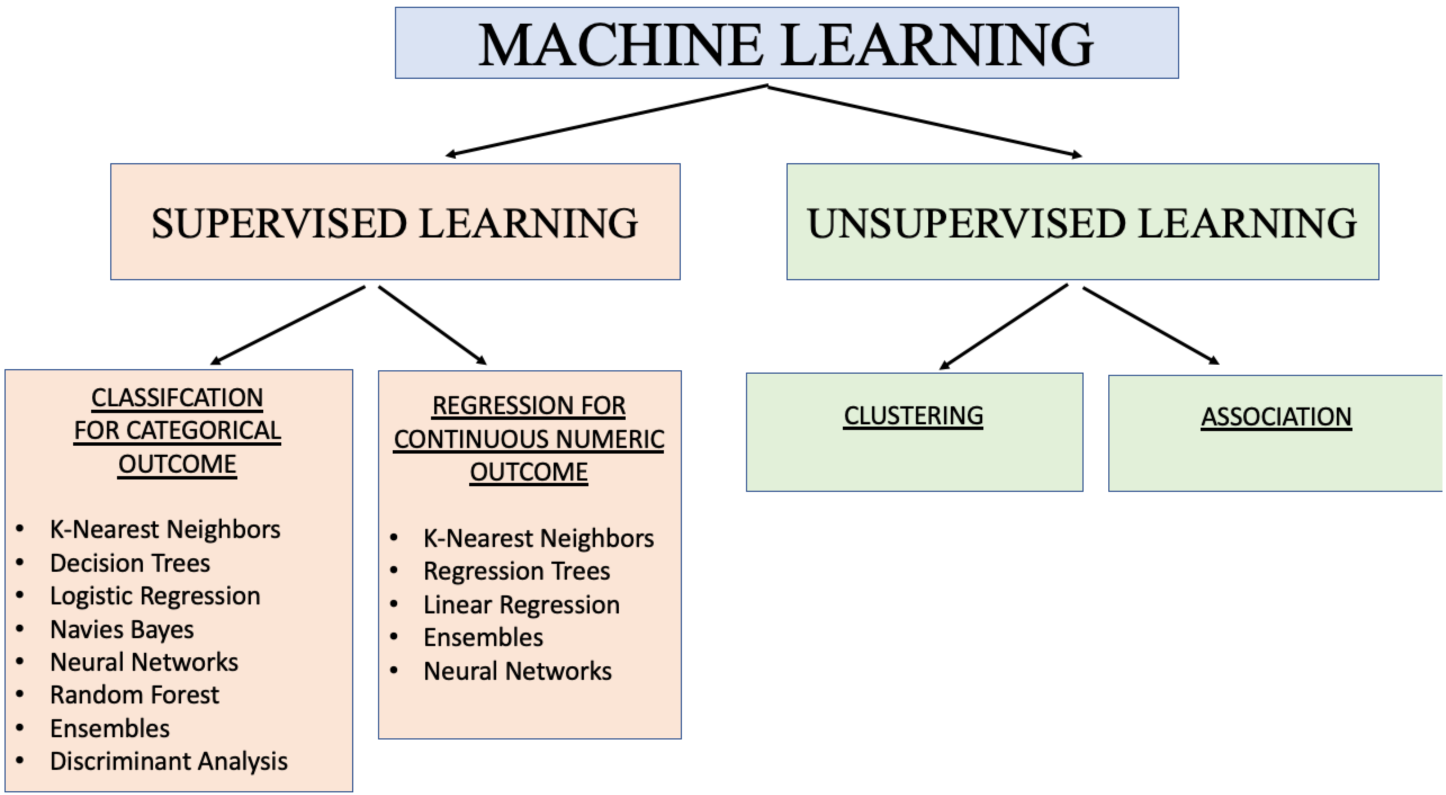
Syllabus

- Week 1: 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: Imbalanced Data
- Week 6: Neural Nets
- Week 7: Large Language Models
- Week 8: Bagging and Boosting
- Week 9: Unsupervised Learning - Clustering & PCA
- Week 10: Reinforcement Learning

Machine learning in the real world

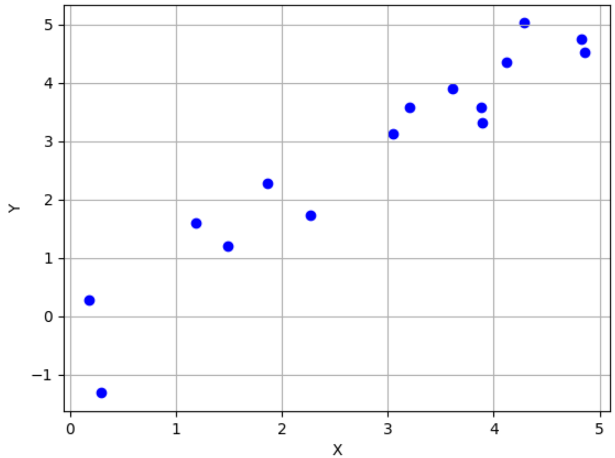


ML Quadrants



Machine Learning and Statistical Models

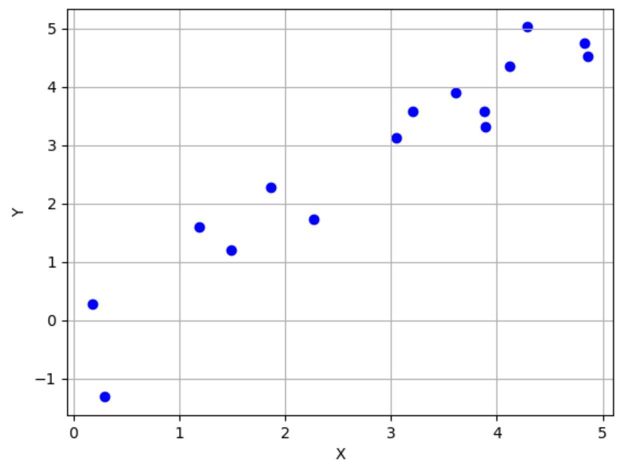
- Simple example. Observe a dataset:



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

Machine Learning and Statistical Models

- Simple example. Observe a dataset:

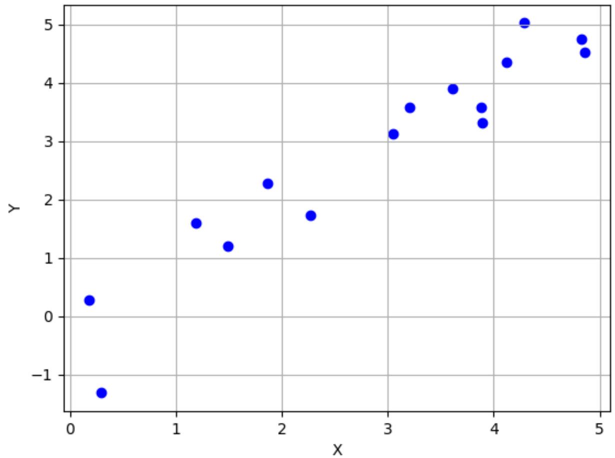


- What is machine learning?

- What is a statistical model?

Machine Learning and Statistical Models

- Simple example. Observe a dataset:

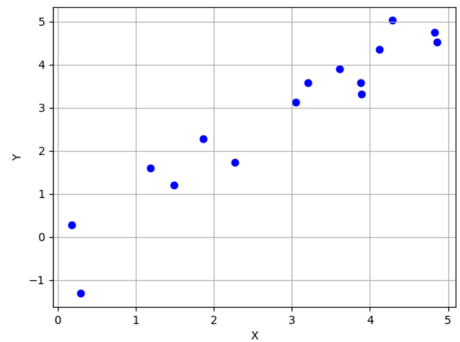


- What is machine learning?

- What is a statistical model?

Machine Learning and Statistical Models

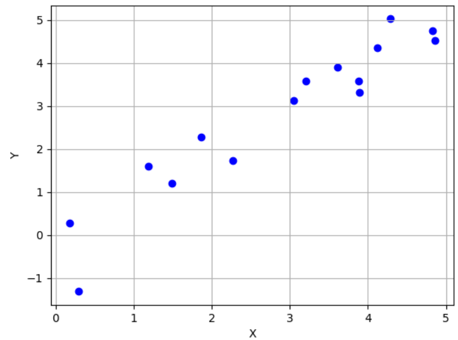
- Suppose we observe a dataset:



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

Machine Learning and Statistical Models

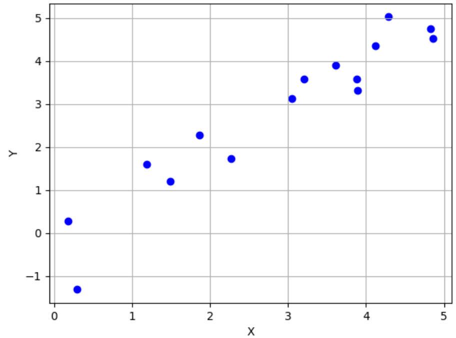
- Suppose we observe a dataset:



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

Machine Learning and Statistical Models

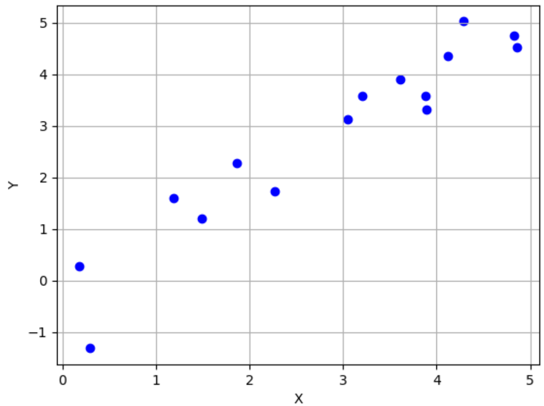
- Suppose we observe a dataset:



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

Statistical Model

Mathematical model (a set of statistical assumptions),
focus on estimation and inference
(hypothesis testing, uncertainty quantification).
Applies to scenarios that demand interpretability

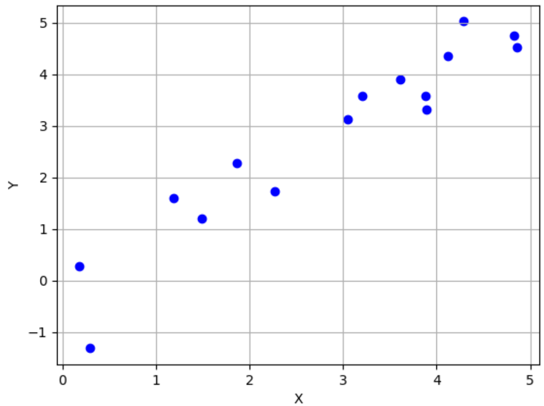


Assumptions:

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

Statistical Model

Mathematical model (a set of statistical assumptions), focus on estimation and inference (hypothesis testing, uncertainty quantification). Applies to scenarios that demand interpretability

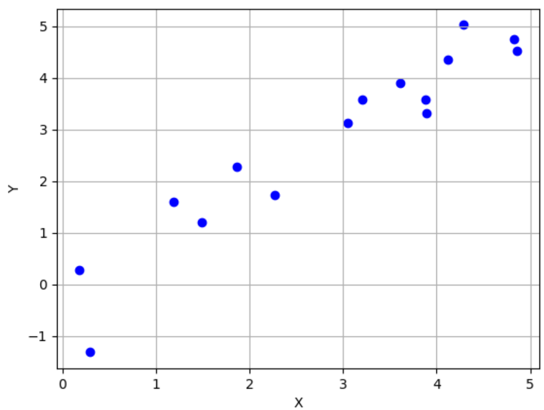


Assumptions:

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

Statistical Model

Mathematical model (a set of statistical assumptions), focus on estimation and inference (hypothesis testing, uncertainty quantification). Applies to scenarios that demand interpretability

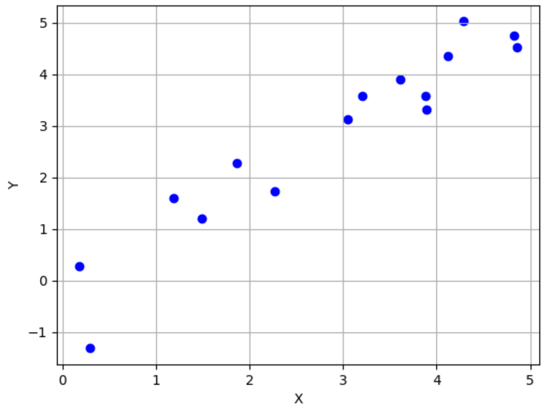


Assumptions:

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

Statistical Model

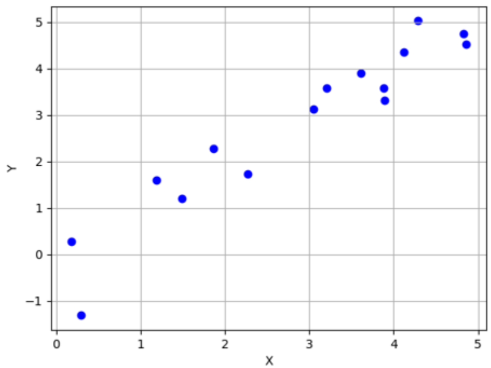
Mathematical model (a set of statistical assumptions), focus on estimation and inference (hypothesis testing, uncertainty quantification).
Applies to scenarios that demand interpretability

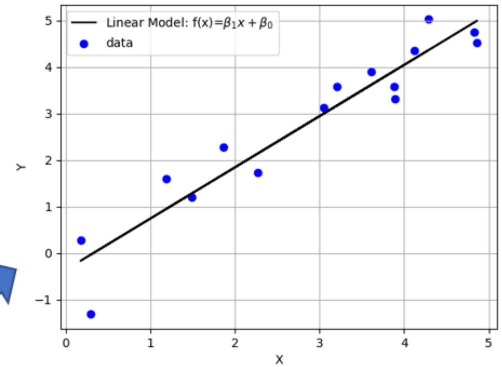


Assumptions:

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

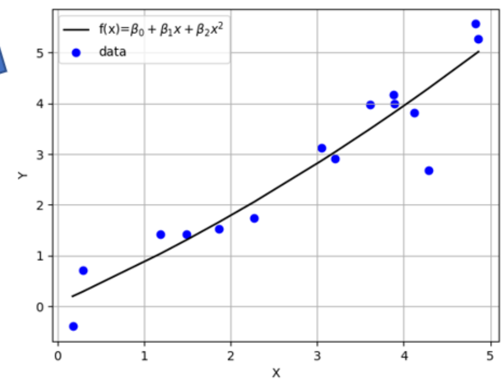
Example: form of f





Assumption: the relationship between X and Y is Linear!

Assumption: the relationship between X and Y is Quadratic!



Statistical machine learning

These days, the two cultures merge as "Statistical ML," which is the main focus of this course

In what follows, I will introduce basic concepts in stat ML:

- Parametric vs nonparametric models
- Training vs testing data
- Bias–variance tradeoff
- Model validation

Statistical machine learning

These days, the two cultures merge as "Statistical ML," which is the main focus of this course

In what follows, I will introduce basic concepts in stat ML:

- Parametric vs nonparametric models
- Training vs testing data
- Bias–variance tradeoff
- Model validation

Statistical machine learning

These days, the two cultures merge as “Statistical ML,” which is the main focus of this course

In what follows, I will introduce basic concepts in stat ML:

- Parametric vs nonparametric models
- Training vs testing data
- Bias–variance tradeoff
- Model validation

Statistical machine learning

These days, the two cultures merge as "Statistical ML," which is the main focus of this course

In what follows, I will introduce basic concepts in stat ML:

- Parametric vs nonparametric models
- Training vs testing data
- Bias–variance tradeoff
- Model validation

Statistical machine learning

These days, the two cultures merge as "Statistical ML," which is the main focus of this course

In what follows, I will introduce basic concepts in stat ML:

- Parametric vs nonparametric models
- Training vs testing data
- Bias–variance tradeoff
- Model validation

Statistical machine learning

These days, the two cultures merge as "Statistical ML," which is the main focus of this course

In what follows, I will introduce basic concepts in stat ML:

- Parametric vs nonparametric models
- Training vs testing data
- Bias–variance tradeoff
- Model validation

Parametric VS nonparametric models

- **Parametric models**: $f(x)$ depends on a *finite number of parameters*. Example: linear regression

$$f(x) = \beta_0 + \beta' x$$

- once we know assume the parametric form of f, the estimation of f reduces to estimating the parameters (β_0, β)

Parametric VS nonparametric models

- **Parametric models**: $f(x)$ depends on a *finite number of parameters*. Example: linear regression

$$f(x) = \beta_0 + \beta' x$$

- once we know assume the parametric form of f, the estimation of f reduces to estimating the parameters (β_0, β)

Parametric VS nonparametric models

- **Nonparametric models**: model that is not parametric. Many interpretations: growing/infinite number of parameters
- Example: K in K-nearest neighbor classifier that grows with sample size
- Other examples: depth of a decision tree, number of layers and width in deep neural networks
- In this course, a nonparametric model is one that does not make explicit assumptions about the form of f

Parametric VS nonparametric models

- **Nonparametric models**: model that is not parametric. Many interpretations: growing/infinite number of parameters
- Example: K in K-nearest neighbor classifier that grows with sample size
- Other examples: depth of a decision tree, number of layers and width in deep neural networks
- In this course, a nonparametric model is one that does not make explicit assumptions about the form of f

Parametric VS nonparametric models

- **Nonparametric models**: model that is not parametric. Many interpretations: growing/infinite number of parameters
- Example: K in K-nearest neighbor classifier that grows with sample size
- Other examples: depth of a decision tree, number of layers and width in deep neural networks
- In this course, a nonparametric model is one that does not make explicit assumptions about the form of f

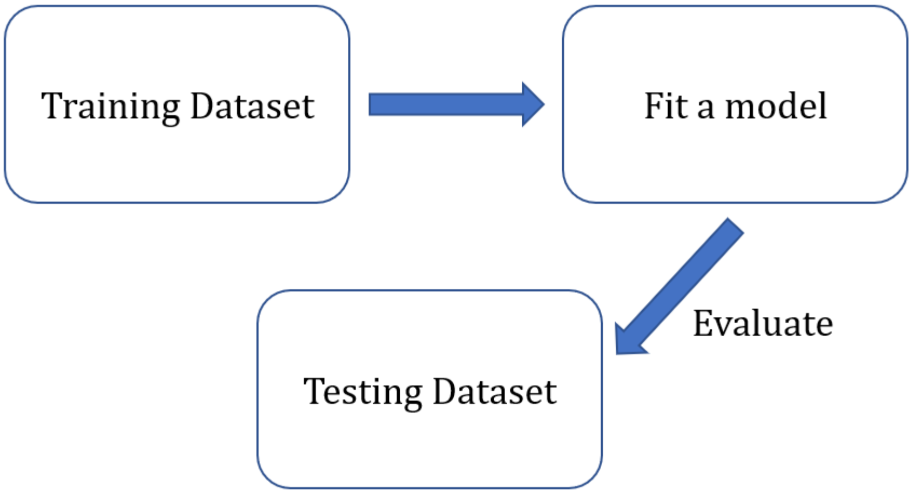
Parametric VS nonparametric models

- **Nonparametric models**: model that is not parametric. Many interpretations: growing/infinite number of parameters
- Example: K in K-nearest neighbor classifier that grows with sample size
- Other examples: depth of a decision tree, number of layers and width in deep neural networks
- In this course, a nonparametric model is one that does not make explicit assumptions about the form of f

Training VS testing data

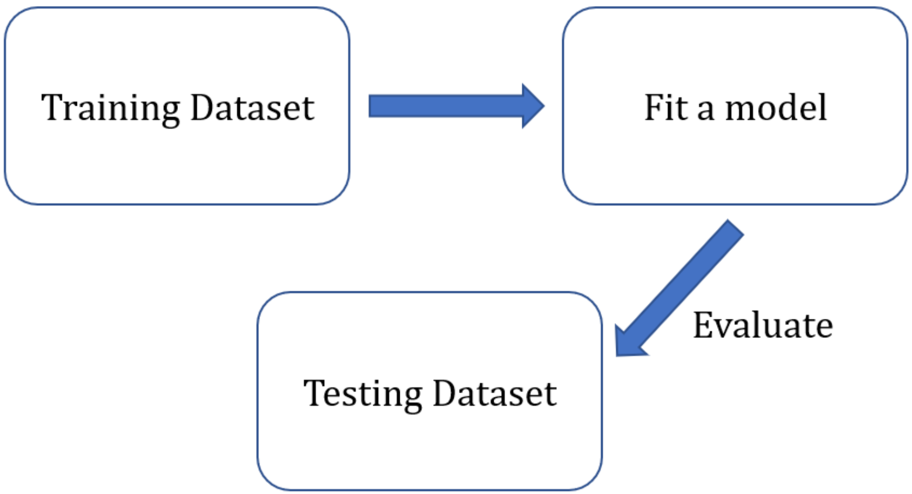
- **Training data**: data used to fit the model

- **Testing data**: data NOT used to fit the model, but used to test how well the model performs



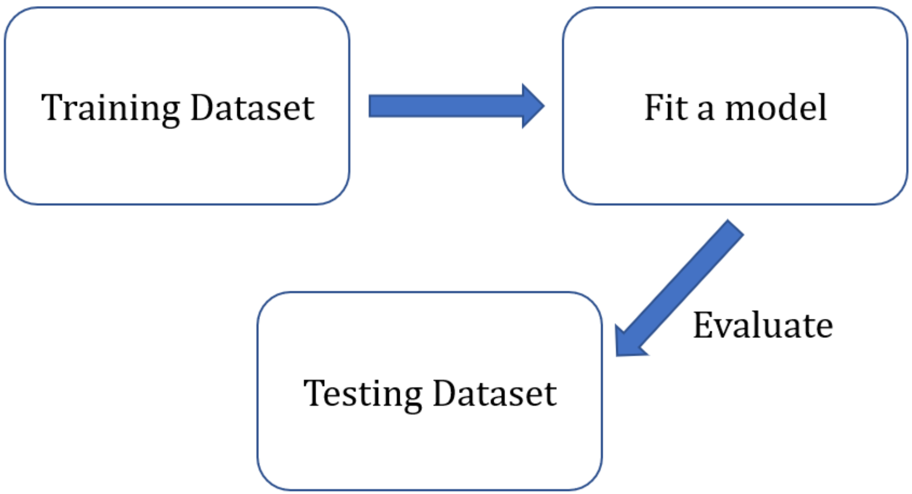
Training VS testing data

- **Training data**: data used to fit the model
- **Testing data**: data NOT used to fit the model, but used to test how well the model performs



Training VS testing data

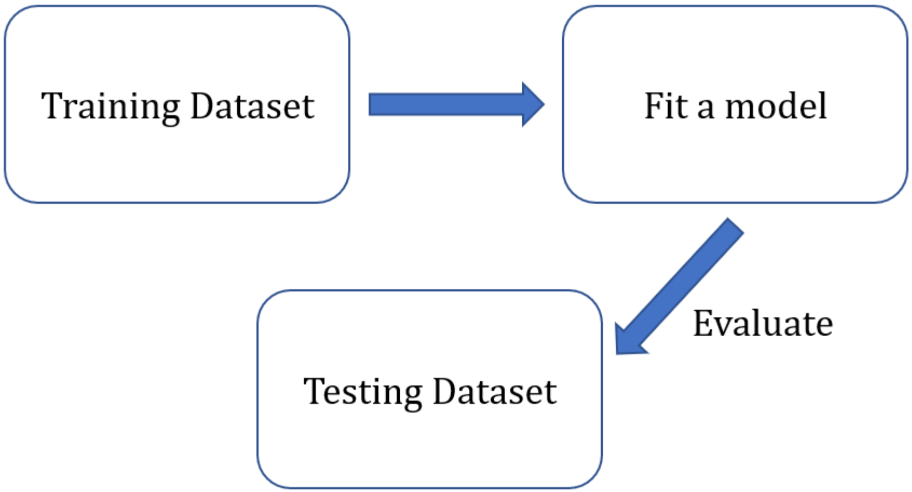
- **Training data**: data used to fit the model
- **Testing data**: data NOT used to fit the model, but used to test how well the model performs



Training VS testing data

- **Training data**: data used to fit the model

- **Testing data**: data NOT used to fit the model, but used to test how well the model performs



Example: regression

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

- **Response/label/target**: Y is any random variable. Generally, Y is something we want to predict, e.g. $Y \in \{cat, dog\}$ or Y is starting salary after graduation

- **Relationship between X and Y**:

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

where $\boldsymbol{X}$ and ϵ independent, $\mathbb{E}(\epsilon) = 0$, $Var(\epsilon) = \sigma^2$

Example: regression

- **Predictors/feature/covariates**: $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)$ is p-dimensional random variable
- **Response/label/target**: Y is any random variable. Generally, Y is something we want to predict, e.g. $Y \in \{cat, dog\}$ or Y is starting salary after graduation
- **Relationship between $\boldsymbol{X}$ and Y**:

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

where $\boldsymbol{X}$ and ϵ independent, $\mathbb{E}(\epsilon) = 0$, $Var(\epsilon) = \sigma^2$

Example: regression

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

- **Response/label/target**: Y is any random variable. Generally, Y is something we want to predict, e.g. $Y \in \{cat, dog\}$ or Y is starting salary after graduation

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

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

where $\boldsymbol{X}$ and ϵ independent, $\mathbb{E}(\epsilon) = 0$, $Var(\epsilon) = \sigma^2$

Statistical machine learning for regression

- **Goal**: Find function f for predicting Y (or approximate f^* well)

- **Loss function**: e.g. squared loss

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

- **Average loss** (expected error, risk) of f:

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

Statistical machine learning for regression

- **Goal**: Find function f for predicting Y (or approximate f^* well)

- **Loss function**: e.g. squared loss

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

- **Average loss** (expected error, risk) of f:

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

Closer look at risk function R

- The expected squared loss can be written as

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

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

- Decomposition:

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

- Therefore, $R(f)$ attains its minimum at

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

Closer look at risk function R

- The expected squared loss can be written as

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

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

- Decomposition:

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

- Therefore, $R(f)$ attains its minimum at

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

Closer look at risk function R

- The expected squared loss can be written as

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

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

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

- Therefore, $R(f)$ 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 form of $\mathbb{E}(Y|\boldsymbol{X})$
- **Question**: What do we usually do?
  - Impose structure on f, e.g.

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

  - Define a hypothesis space

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

.

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

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

How to estimate f in practice?

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

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

  - Define a hypothesis space

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

.

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

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

How to estimate f in practice?

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

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

  - Define a hypothesis space

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

.

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

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

How to estimate f in practice?

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

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

  - Define a hypothesis space

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

.

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

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

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

- Given training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, we obtain an estimator

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

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

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

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

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

- Given training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, we obtain an estimator

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

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

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

- Here, (x_0, Y) is the testing data

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

- Given training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, we obtain an estimator

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

用来evaluate模型的quality

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

xo是testing data

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

fundamental noise

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

Bias–variance tradeoff

- Reducible part can be decomposed into two components

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

  where the expectation is taken w.r.t. the training dataset D (that takes care of randomness in $\widehat{f}$)
  - **Variance**: represents variability of the predicted value. Randomness comes from the training data
  - **Squared Bias**: if $\mathcal{F}$ is flexible enough, then the mean across all training datasets is the truth, i.e. bias $=0$

Bias–variance tradeoff

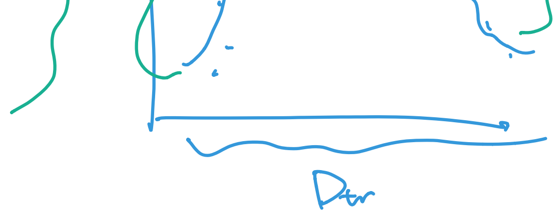
- Reducible part can be decomposed into two components

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

where the expectation is taken w.r.t. the training dataset D (that takes care of randomness in $\widehat{f}$)

- **Variance**: represents variability of the predicted value. Randomness comes from the training data
- **Squared Bias**: if $\mathcal{F}$ is flexible enough, then the mean across all training datasets is the truth, i.e. bias $=0$

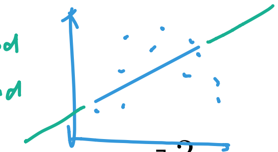
Bias–variance tradeoff



(bias)
fit good
var bad

- Reducible part can be decomposed into two components

var good
(bias) fit bad



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

where the expectation is taken w.r.t. the training dataset D (that takes care of randomness in $\widehat{f}$)

- **Variance**: represents variability of the predicted value. Randomness comes from the training data
- **Squared Bias**: if $\mathcal{F}$ is flexible enough, then the mean across all training datasets is the truth, i.e. bias $=0$

Training MSE VS 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 to use for assessing quality of $\widehat{f}$?

Training MSE VS 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 to use for assessing quality of $\widehat{f}$?

Training MSE VS 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

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

want to min it
⇒ bias low
var high

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

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

如果是 "all function f" 则 training error = 0

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

- **Question**: Which one to use for assessing quality of $\widehat{f}$? Testing error

if $\hat{f}$ becomes more flexible, traing error decrease. (or: not increase) (better fit)

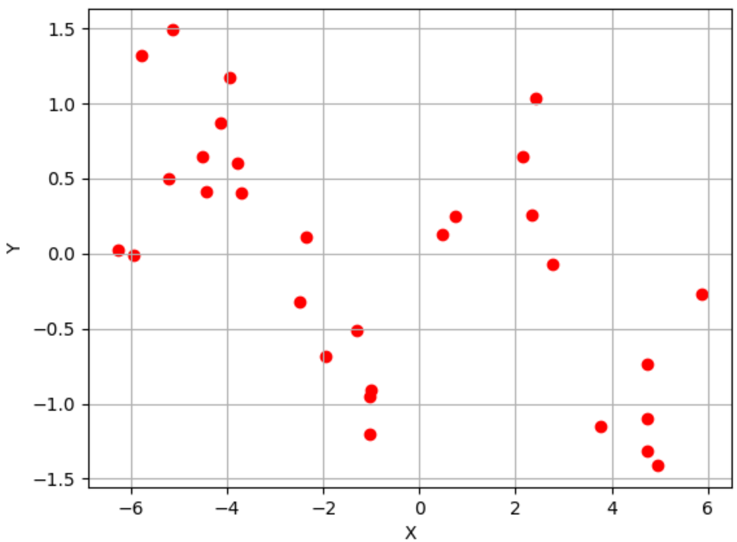
N 多大 or 加入更多参数 or "more complex"

Example

- Let $(x_i, y_i)_{i=1}^n$ satisfy

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

  - $x_i \sim \text{Unif}(-2\pi, 2\pi)$
  - $\epsilon_i \sim N(0, 0.5)$ indep. of x
  - sample size $n = 30$



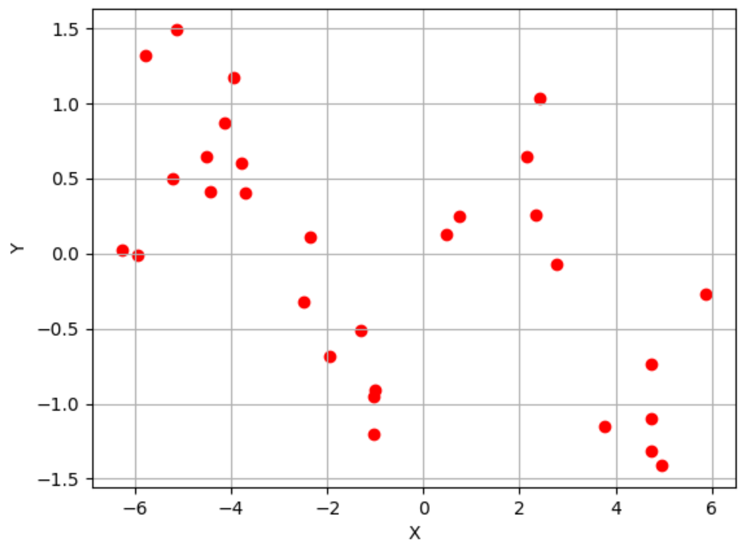
Example

"error" refers to square loss

- Let $(x_i, y_i)_{i=1}^n$ satisfy

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

  - $x_i \sim \text{Unif}(-2\pi, 2\pi)$
  - $\epsilon_i \sim N(0, 0.5)$ indep. of x
  - sample size $n = 30$

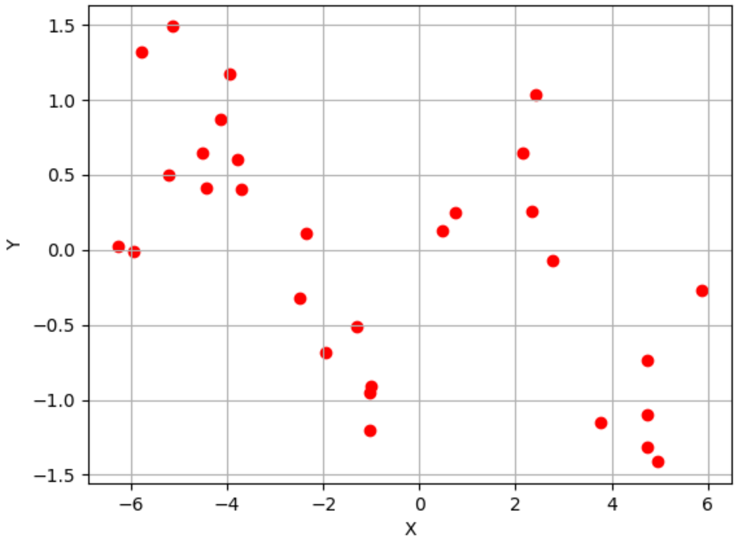


Example

- Let $(x_i, y_i)_{i=1}^n$ satisfy

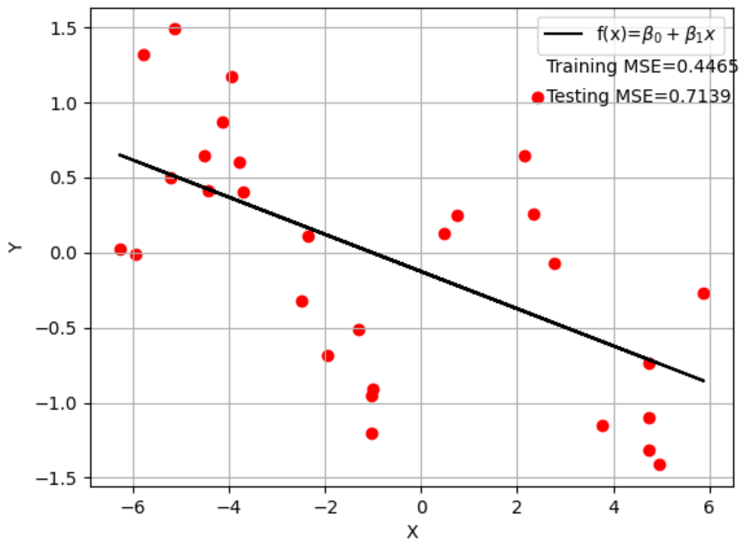
$$y_i = \sin(x_i) + \epsilon_i$$

  - $x_i \sim \text{Unif}(-2\pi, 2\pi)$
  - $\epsilon_i \sim N(0, 0.5)$ indep. of x
  - sample size $n = 30$



Example: linear regression

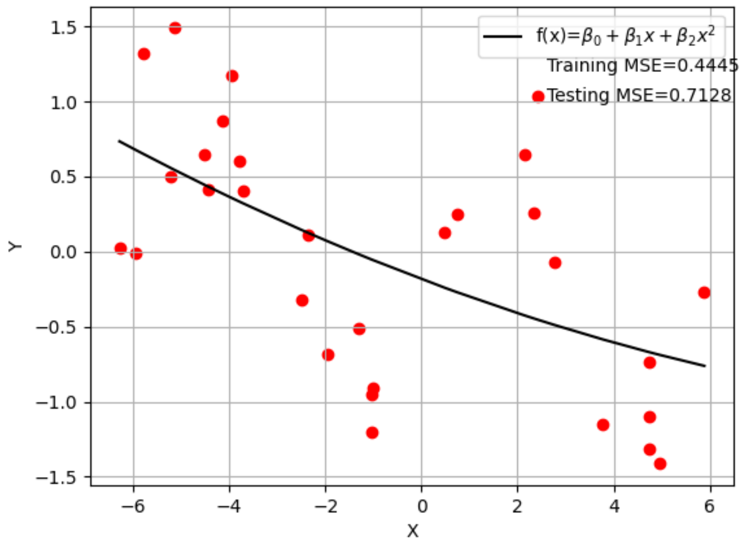
- Fit a linear model $f(x) = \beta_0 + \beta_1 x$



- Training MSE is 0.4465
- Testing MSE is 0.7139

Example: quadratic regression

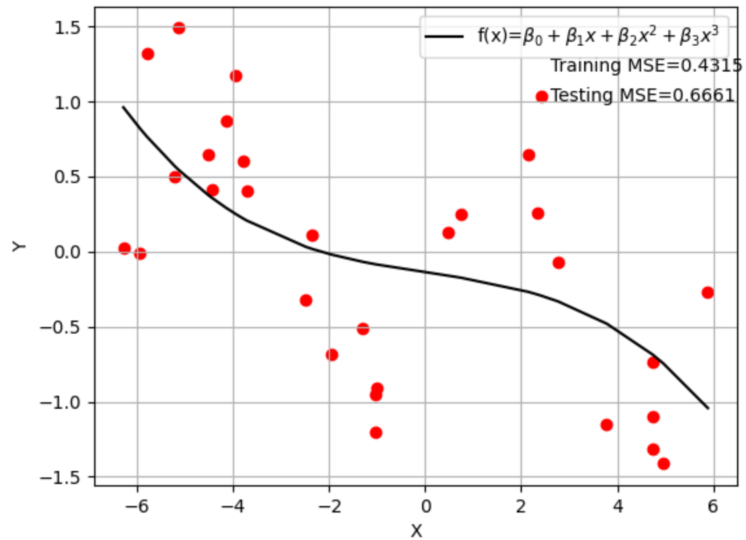
- Fit a linear model $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2$



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

Example: polynomial regression

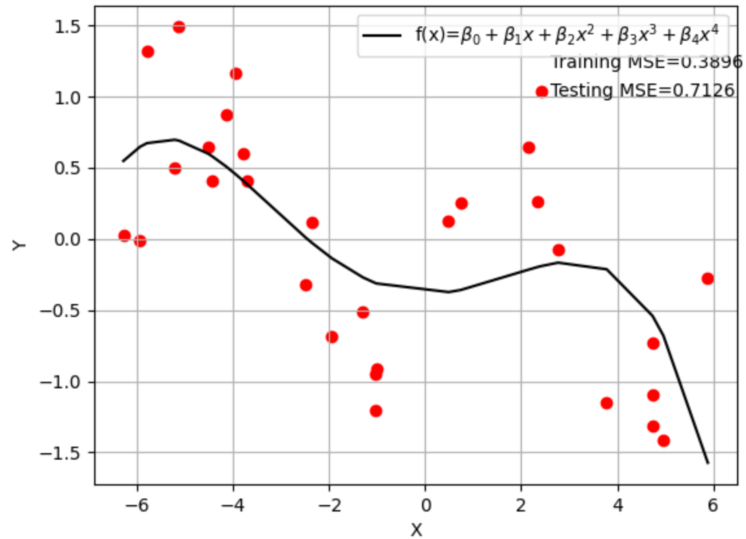
- Fit a linear model $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$



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

Example: higher order polynomial

- Fit a linear model $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$



- Training MSE is 0.3896 (decreased by 0.0419)
- Testing MSE is 0.7126 (increased by 0.0464): start fitting noise rather than signal...

Irrelavant from minimizing the testing error

Example: takeaways

should be the optimal one

more flexible. less bias
high var

Metrics	Model 1	Model 2	Model 3	Model 4
Train MSE	0.4465	0.4445	0.4315	0.3896
Test MSE	0.7139	0.7128	0.6661	0.7126

- **Train vs test MSE vs flexibility of the model**:
  (1) Train MSE is non-increasing
  (2) Test MSE first decreases and then increases (U-shape)
- **Bias-variance tradeoff for test MSE**:
  (1) More flexibility ⇒ small bias, high variance
  (2) Less flexibility ⇒ high bias, low variance

Selecting a model

- so far: choosing a model amounts to balancing bias vs variance
- details depend on exact modeling assumptions
- e.g. should we choose the function f linear OR quadratic OR cubic?
- more generally: how to select the best model from a set of candidate models?
- use model validation

Model validation

- **Training set**: used to train the ML model that learns patterns, relationships, and features from this set
- **Validation set**: a separate subset of data that is NOT used directly for training, but is used during the training phase to assess the model's performance on unseen data
- **Test (holdout) set**: yet another subset of data that is NOT used during training OR validation, but is reserved for the final evaluation of the model performance after training

Training/validation split

- **Def**. (True) testing error is mean loss on unseen data, e.g.

  有人会写作y

  $$\mathbb{E}_{x_0,D}\left(\hat{f}(x_0) - f(x_0)\right)^2,$$

  where x_0 is indep. of data D used to train $\hat{f}$
- Split the data into two parts: training and validation. The average error on validation data is an **estimate of testing error**
- in practice, training/validation split is usually 80:20 or 66:34

Simulation experiment:
how well does validation error estimate test error?

就是 observations

- The *diabetes* dataset has 768 samples
- Use 500 samples for training and 268 samples for testing
- The “true” test MSE is 0.2350 (calculated on the 268 testing samples)

Simulation experiment: how well does validation error estimate test error?

- Repeat the following experiment many times to see how well the validation error estimates the true testing error (0.2350)
- For each repetition, split the data into **training** (350 samples) and **validation** (150 samples) again
- **Fit** a logistic regression (to be discussed in Week 2) on the training set
- Compute the **validation error** (MSE on the validation set) and compare it with the true testing error

Simulation experiment: how well does validation error estimate test error?

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

Simulation experiment: how well does validation error estimate test error?

- Disadvantages of hold-out validation:

  (1) the **validation error** is **highly variable**, depending on the split

  (2) **only a subset of data is used to train the model** (350 out of 500) $\Rightarrow$ information loss, test error overestimated

Solution: cross-validation

- **Cross-validation**: repeating the training/validation split multiple times

- **Objective**: for a given ML method, estimate the test error to
  - evaluate performance with less variability (model assessment)
  - select an appropriate level of flexibility (model selection)

- **Algorithm**: hold out a subset of data from the training process and evaluate model fit on those held-out (unseen) observations

K-Fold Cross-Validation

- The dataset is divided into K subsets (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

- Similar to K-Fold, but ensures that each fold maintains the same class distribution as the original data. Particularly useful for imbalanced datasets (to be discussed in week 5)

Leave-One-Out Cross-Validation (LOOCV)

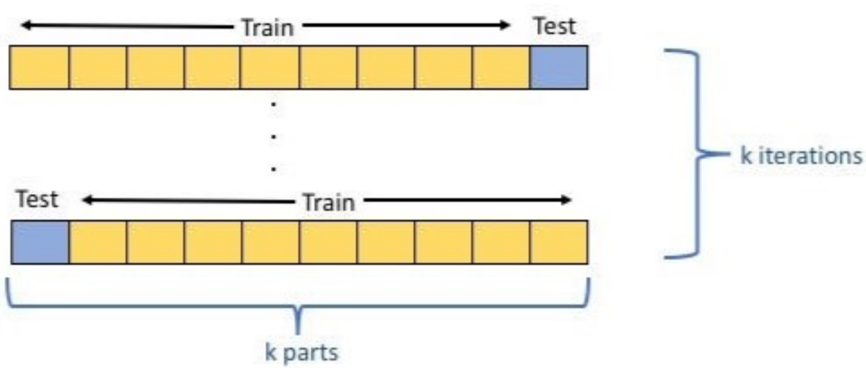
- Only one data point is used for validation, and the model is trained on the remaining data. This process is repeated for each data point in the dataset. E.g., if $n = 100$, then 100 training/validation iterations

- Computationally expensive (or even infeasible) when sample size is large (except linear regression where an explicit formula is available)

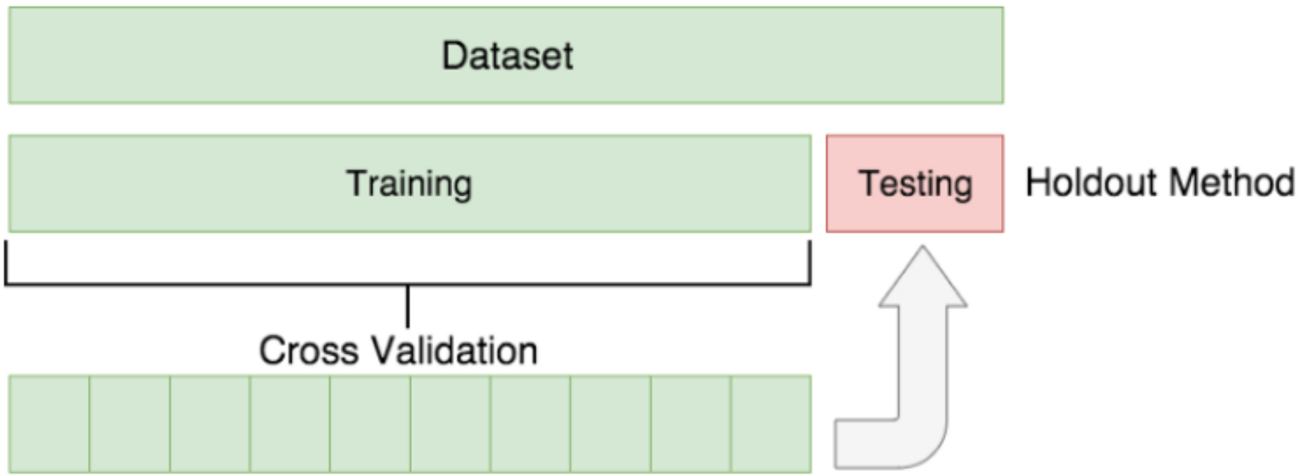
- The validation MSE from LOOCV is an average of n fold-specific error estimates. Each of these is based on almost the same data $\Rightarrow$ highly correlated $\Rightarrow$ high variance

K-fold Cross-validation: implementation

- Divide the 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
- Calculate a performance metric (e.g., mean squared error, misclassification error, prediction interval) as an average across folds



Training/validation/testing



Example: 6-fold cross-validation

1. Data $D = (z_i)_{i=1}^{6n} = (\boldsymbol{x}_i, y_i)_{i=1}^{6n}$. Split D into 6 parts:

$$D_1 = (z_i)_{i=1}^{n}, D_2 = (z_i)_{i=n+1}^{2n}, D_2 = (z_i)_{i=2n+1}^{3n}$$
$$D_4 = (z_i)_{i=3n+1}^{4n}, D_5 = (z_i)_{i=4n+1}^{5n}, D_6 = (z_i)_{i=5n+1}^{6n}$$

2. For $j = 1, \ldots, 6$:
   (1) Construct $D_{-j} = \cup_{i \neq j} D_i$
   (2) Train a function $\hat{f}$ as

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

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

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

Example: 6-fold cross-validation

4. Use the average (across folds) validation error as an estimate of testing error

$$VE = \frac{1}{6} \sum_{j=1}^{6} VE_j(\widehat{f}_{-j})$$