Lecture 3: Statistical Learning (Textbook 2.2)

Nayel Bettache

Department of Statistical Science, Cornell University

Assess Model Accuracy

- **Objective**: Introduce you to a wide range of statistical learning methods

Assess Model Accuracy

- **Objective**: Introduce you to a wide range of statistical learning methods
- Why don't we just present the best performing method and study it extensively ?

Assess Model Accuracy

- **Objective**: Introduce you to a wide range of statistical learning methods
- Why don't we just present the best performing method and study it extensively ?
  - There is no free lunch in statistics

Assess Model Accuracy

- **Objective**: Introduce you to a wide range of statistical learning methods
- Why don't we just present the best performing method and study it extensively ?
  - There is no free lunch in statistics
  - No one method dominates all others over all possible data sets.

Assess Model Accuracy

- **Objective**: Introduce you to a wide range of statistical learning methods
- Why don't we just present the best performing method and study it extensively ?
  - There is no free lunch in statistics
  - No one method dominates all others over all possible data sets.
  - On a particular data set, one specifc method may work best, but some other method may work better on a similar but different data set.

Assess Model Accuracy

- **Objective**: Introduce you to a wide range of statistical learning methods
- Why don't we just present the best performing method and study it extensively ?
  - There is no free lunch in statistics
  - No one method dominates all others over all possible data sets.
  - On a particular data set, one specifc method may work best, but some other method may work better on a similar but different data set.
- It is an important task to decide for any given dataset which method produces the best results.

Assess Model Accuracy

- **Objective**: Introduce you to a wide range of statistical learning methods
- Why don't we just present the best performing method and study it extensively ?
  - There is no free lunch in statistics
  - No one method dominates all others over all possible data sets.
  - On a particular data set, one specifc method may work best, but some other method may work better on a similar but different data set.
- It is an important task to decide for any given dataset which method produces the best results.
- Selecting the best approach can be one of the most challenging parts of performing statistical learning in practice.

Measuring the Quality of Fit

We fit a model $\hat{f}$ with some training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

Measuring the Quality of Fit

We fit a model $\hat{f}$ with some training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

- **Objective**: Evaluate the performance of a statistical learning method on a given data set.

Measuring the Quality of Fit

We fit a model $\hat{f}$ with some training data $\{(x_1, y_1), \dots, (x_n, y_n)\}$.

- **Objective**: Evaluate the performance of a statistical learning method on a given data set.
- Why do we need to measure the performance?

Measuring the Quality of Fit

We fit a model $\hat{f}$ with some training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

- **Objective**: Evaluate the performance of a statistical learning method on a given data set.
- Why do we need to measure the performance?
  - Quantify how well the predicted response matches the observed data.

Measuring the Quality of Fit

We fit a model $\hat{f}$ with some training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

- **Objective**: Evaluate the performance of a statistical learning method on a given data set.
- Why do we need to measure the performance?
  - Quantify how well the predicted response matches the observed data.
  - In regression, the most commonly-used measure is the MSE.

Measuring the Quality of Fit

We fit a model $\hat{f}$ with some training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

- **Objective**: Evaluate the performance of a statistical learning method on a given data set.
- Why do we need to measure the performance?
  - Quantify how well the predicted response matches the observed data.
  - In regression, the most commonly-used measure is the MSE.
- Mean Squared Error (MSE): $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$.

Measuring the Quality of Fit

We fit a model $\hat{f}$ with some training data $\{(x_1, y_1), \dots, (x_n, y_n)\}$.

- **Objective**: Evaluate the performance of a statistical learning method on a given data set.
- Why do we need to measure the performance?
  - Quantify how well the predicted response matches the observed data.
  - In regression, the most commonly-used measure is the MSE.
- Mean Squared Error (MSE): $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$.
  - MSE is calculated as the average of the squared differences between the predicted and true response values on the TRAINING data $\{(x_1, y_1), \dots, (x_n, y_n)\}$.

Measuring the Quality of Fit

We fit a model $\hat{f}$ with some training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

- **Objective**: Evaluate the performance of a statistical learning method on a given data set.
- Why do we need to measure the performance?
    - Quantify how well the predicted response matches the observed data.
    - In regression, the most commonly-used measure is the MSE.
- Mean Squared Error (MSE): $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$.
    - MSE is calculated as the average of the squared differences between the predicted and true response values on the TRAINING data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.
    - Smaller MSE $\implies$ the predicted responses are closer to the true responses.

Measuring the Quality of Fit

We fit a model $\hat{f}$ with some training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

- **Objective**: Evaluate the performance of a statistical learning method on a given data set.
- Why do we need to measure the performance?
  - Quantify how well the predicted response matches the observed data.
  - In regression, the most commonly-used measure is the MSE.
- Mean Squared Error (MSE): $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$.
  - MSE is calculated as the average of the squared differences between the predicted and true response values on the TRAINING data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.
  - Smaller MSE $\implies$ the predicted responses are closer to the true responses.
- Training MSE vs Test MSE

Measuring the Quality of Fit

We fit a model $\hat{f}$ with some training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

- **Objective**: Evaluate the performance of a statistical learning method on a given data set.
- Why do we need to measure the performance?
    - Quantify how well the predicted response matches the observed data.
    - In regression, the most commonly-used measure is the MSE.
- Mean Squared Error (MSE): $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$.
    - MSE is calculated as the average of the squared differences between the predicted and true response values on the TRAINING data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.
    - Smaller MSE $\implies$ the predicted responses are closer to the true responses.
- Training MSE vs Test MSE
    - Test data refers to the data which are not used to train the statistical model (i.e., not used to calculate $\hat{f}$).

Measuring the Quality of Fit

We fit a model $\hat{f}$ with some training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

- **Objective**: Evaluate the performance of a statistical learning method on a given data set.
- Why do we need to measure the performance?
  - Quantify how well the predicted response matches the observed data.
  - In regression, the most commonly-used measure is the MSE.
- Mean Squared Error (MSE): $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$.
  - MSE is calculated as the average of the squared differences between the predicted and true response values on the TRAINING data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.
  - Smaller MSE $\implies$ the predicted responses are closer to the true responses.
- Training MSE vs Test MSE
  - Test data refers to the data which are not used to train the statistical model (i.e., not used to calculate $\hat{f}$).
  - Generally, we do not really care how well the method works on the training data (overfitting is possible).

Measuring the Quality of Fit

We fit a model $\hat{f}$ with some training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

- **Objective**: Evaluate the performance of a statistical learning method on a given data set.
- Why do we need to measure the performance?
  - Quantify how well the predicted response matches the observed data.
  - In regression, the most commonly-used measure is the MSE.
- Mean Squared Error (MSE): $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$.
  - MSE is calculated as the average of the squared differences between the predicted and true response values on the TRAINING data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.
  - Smaller MSE $\implies$ the predicted responses are closer to the true responses.
- Training MSE vs Test MSE
  - Test data refers to the data which are not used to train the statistical model (i.e., not used to calculate $\hat{f}$).
  - Generally, we do not really care how well the method works on the training data (overfitting is possible).
  - We prefer the accuracy of the predictions on unseen test data.

Measuring the Quality of Fit: overview

- We have at hand a dataset $\{(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$.

Measuring the Quality of Fit: overview

- We have at hand a dataset $\{(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$.
- We fit a model $\hat{f}$ with the training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

Measuring the Quality of Fit: overview

- We have at hand a dataset $\{(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$.
- We fit a model $\hat{f}$ with the training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- We compute the **Training MSE**.

Measuring the Quality of Fit: overview

- We have at hand a dataset $\{(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$.
- We fit a model $\hat{f}$ with the training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- We compute the **Training MSE**.
    - $MSE_{Tr} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$.

Measuring the Quality of Fit: overview

- We have at hand a dataset $\{(x_1, y_1), \dots, (x_n, y_n), (x_{n+1}, y_{n+1}), \dots, (x_{n+T}, y_{n+T})\}$.
- We fit a model $\hat{f}$ with the training data $\{(x_1, y_1), \dots, (x_n, y_n)\}$.
- We compute the **Training MSE**.
  - $MSE_{Tr} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$.
  - This quantifies how well the model performs on data used for training.

Measuring the Quality of Fit: overview

- We have at hand a dataset $\{(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$.
- We fit a model $\hat{f}$ with the training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- We compute the **Training MSE**.
  - $MSE_{Tr} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$.
  - This quantifies how well the model performs on data used for training.
  - Not a valid measure of the model fit because it can be an overfitting model.

Measuring the Quality of Fit: overview

- We have at hand a dataset $\{(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$.
- We fit a model $\hat{f}$ with the training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- We compute the **Training MSE**.
  - $MSE_{Tr} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$.
  - This quantifies how well the model performs on data used for training.
  - Not a valid measure of the model fit because it can be an overfitting model.
- We compute the **Test MSE**.

Measuring the Quality of Fit: overview

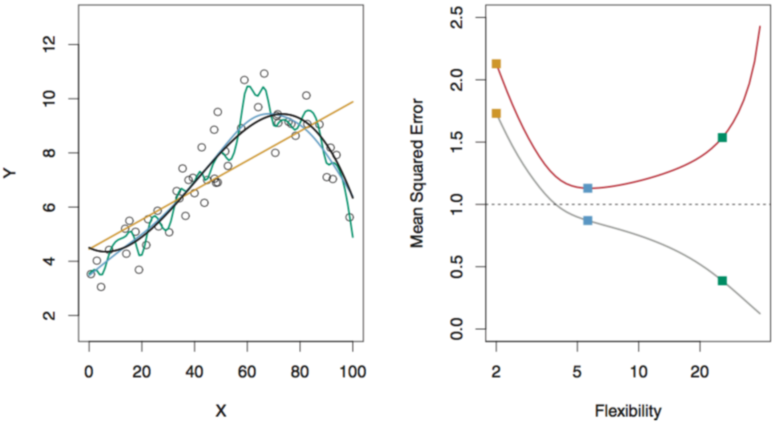
- We have at hand a dataset $\{(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$.
- We fit a model $\hat{f}$ with the training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- We compute the **Training MSE**.
  - $MSE_{Tr} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$.
  - This quantifies how well the model performs on data used for training.
  - Not a valid measure of the model fit because it can be an overfitting model.
- We compute the **Test MSE**.
  - $MSE_{Te} = \frac{1}{T} \sum_{t=1}^{T} (y_{n+t} - \hat{f}(x_{n+t}))^2$.

Measuring the Quality of Fit: overview

- We have at hand a dataset $\{(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$.
- We fit a model $\hat{f}$ with the training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- We compute the **Training MSE**.
  - $MSE_{Tr} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$.
  - This quantifies how well the model performs on data used for training.
  - Not a valid measure of the model fit because it can be an overfitting model.
- We compute the **Test MSE**.
  - $MSE_{Te} = \frac{1}{T} \sum_{t=1}^{T} (y_{n+t} - \hat{f}(x_{n+t}))^2$.
  - We'd like to select the model for which the test MSE is as small as possible.

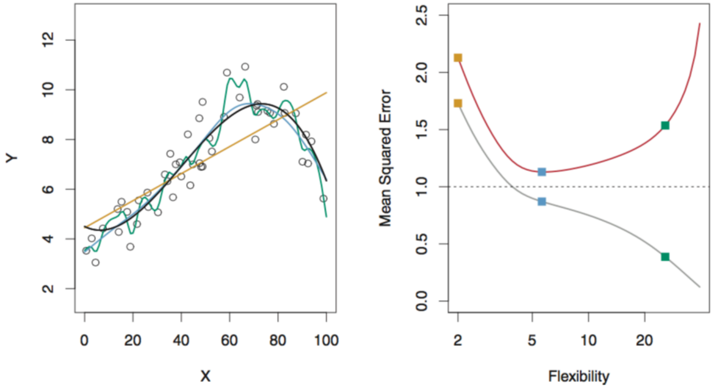
Training MSE vs Test MSE

Left: Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two nonparametric fits (blue and green curves). Right: Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.



Training MSE vs Test MSE: explanations

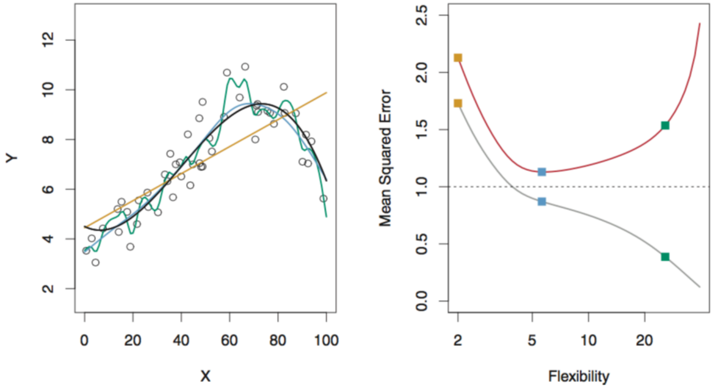
Left Panel:



Training MSE vs Test MSE: explanations

Left Panel:

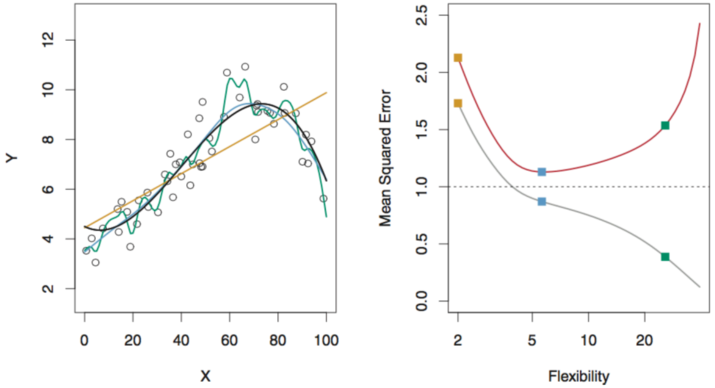
- True function f is represented by the black curve.



Training MSE vs Test MSE: explanations

Left Panel:

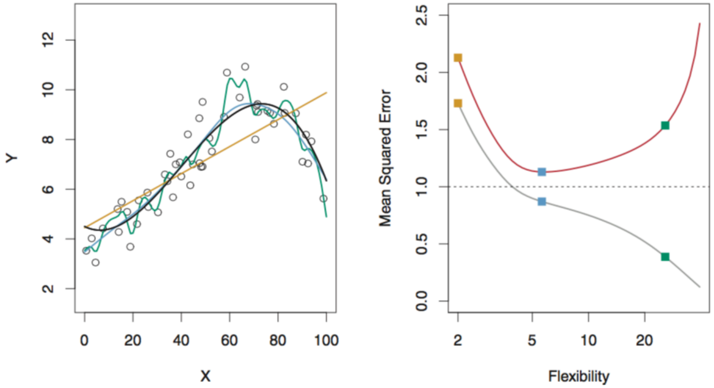
- True function f is represented by the black curve.
- Orange, blue, and green curves represent estimates of f using methods with increasing flexibility.



Training MSE vs Test MSE: explanations

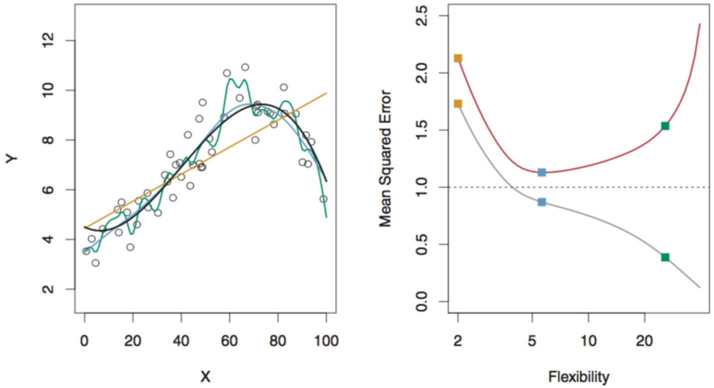
Left Panel:

- True function f is represented by the black curve.
- Orange, blue, and green curves represent estimates of f using methods with increasing flexibility.
- The green curve, the most flexible, fits the observed data closely but poorly estimates the true f because it is too wiggly.



Training MSE vs Test MSE: explanations

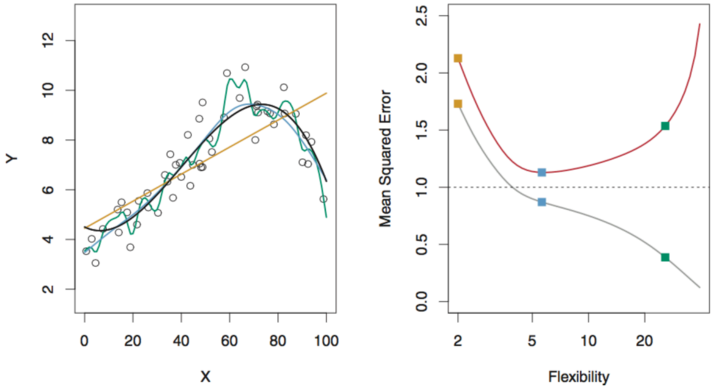
Right Panel:



Training MSE vs Test MSE: explanations

Right Panel:

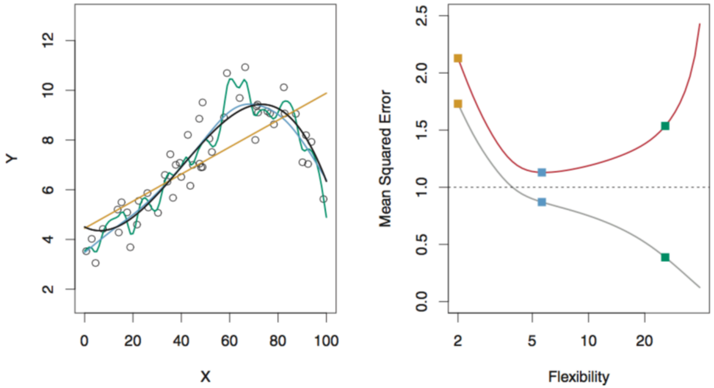
- Training MSE decreases as flexibility increases.



Training MSE vs Test MSE: explanations

Right Panel:

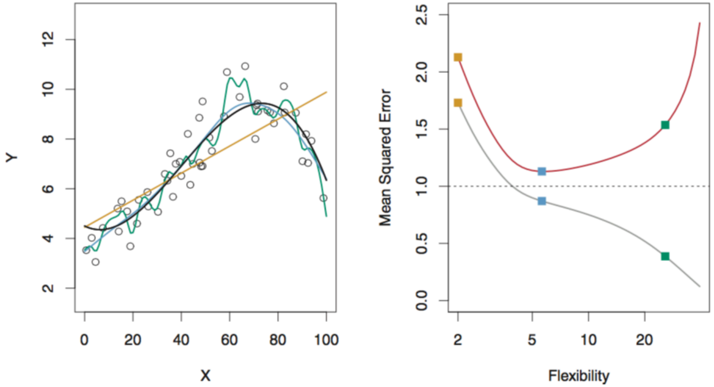
- Training MSE decreases as flexibility increases.
- Test MSE initially decreases, but eventually increases, showing a U-shape.



Training MSE vs Test MSE: explanations

Right Panel:

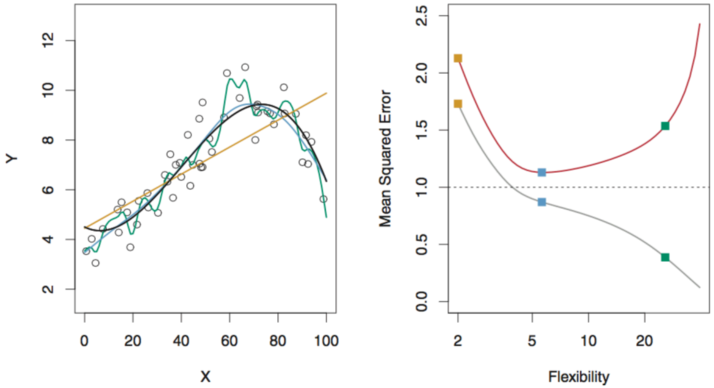
- Training MSE decreases as flexibility increases.
- Test MSE initially decreases, but eventually increases, showing a U-shape.
- The blue curve minimizes the test MSE, which visually appears to estimate f the best.



Training MSE vs Test MSE: explanations

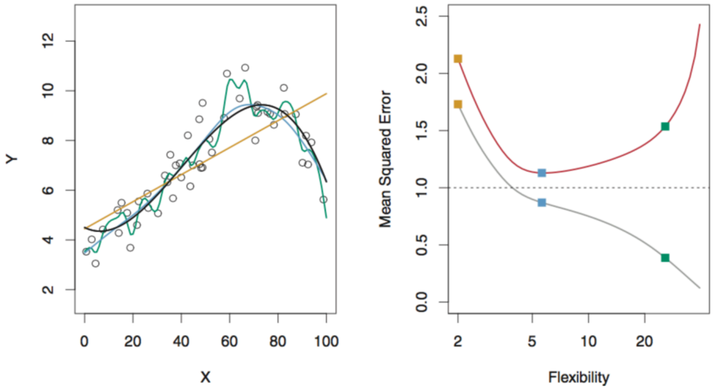
Right Panel:

- Training MSE decreases as flexibility increases.
- Test MSE initially decreases, but eventually increases, showing a U-shape.
- The blue curve minimizes the test MSE, which visually appears to estimate f the best.
- Horizontal dashed line represents the irreducible error, Var(ϵ), the lowest achievable test MSE.



Training MSE vs Test MSE: explanations

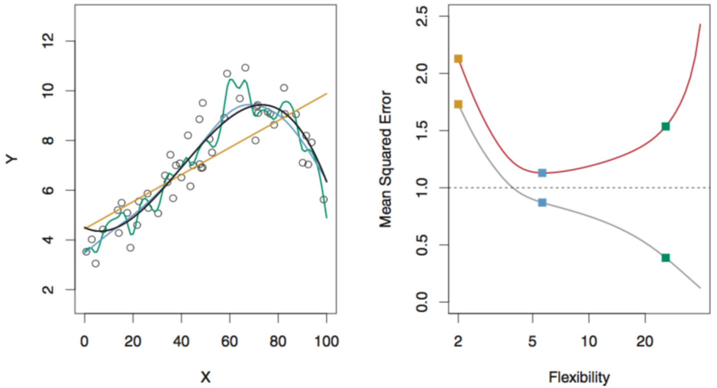
Key Insight:



Training MSE vs Test MSE: explanations

Key Insight:

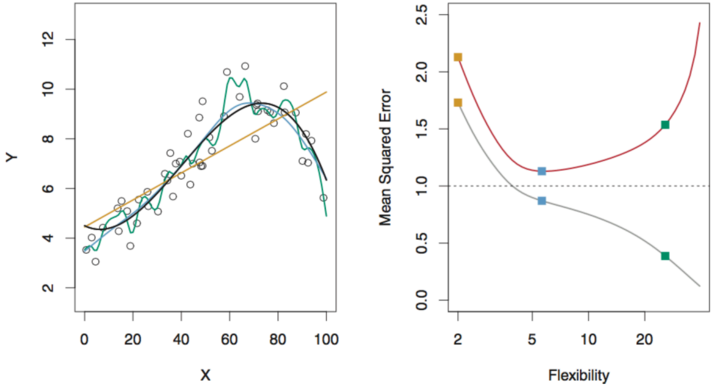
- As model flexibility increases, training MSE decreases, but test MSE may increase, leading to overfitting.



Training MSE vs Test MSE: explanations

Key Insight:

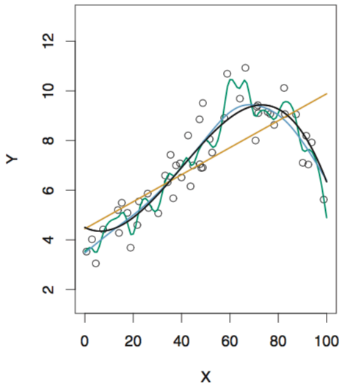
- As model flexibility increases, training MSE decreases, but test MSE may increase, leading to overfitting.
- Overfitting occurs when the method finds patterns in the training data that are due to random chance, leading to a high test MSE.

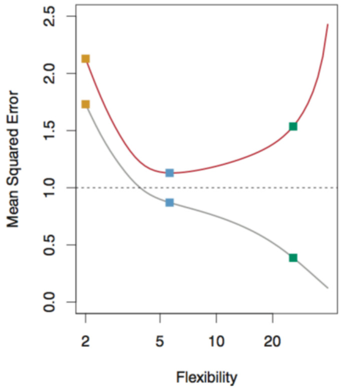


Training MSE vs Test MSE: explanations

Key Insight:

- As model flexibility increases, training MSE decreases, but test MSE may increase, leading to overfitting.
- Overfitting occurs when the method finds patterns in the training data that are due to random chance, leading to a high test MSE.
- Even without overfitting, training MSE is usually smaller than test MSE because most methods aim to minimize the training MSE.





Bias-Variance Trade-off

Suppose we have an estimator $\hat{f}(x)$ from the training data.

Bias-Variance Trade-off

Suppose we have an estimator $\hat{f}(x)$ from the training data.
Let (x_0, y_0) be a test observation drawn from the population.

Bias-Variance Trade-off

Suppose we have an estimator $\hat{f}(x)$ from the training data.
Let (x_0, y_0) be a test observation drawn from the population.
True model is $Y = f(X) + \epsilon$ (with $f(x) = \mathbb{E}(Y|X = x)$).

Bias-Variance Trade-off

Suppose we have an estimator $\hat{f}(x)$ from the training data.
Let (x_0, y_0) be a test observation drawn from the population.
True model is $Y = f(X) + \epsilon$ (with $f(x) = \mathbb{E}(Y|X = x)$).
Then the **expected test MSE** at x_0 is:

$$\mathbb{E}[(y_0 - \hat{f}(x_0))^2] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{\textit{Variance}} + \underbrace{[\mathbb{E}[\hat{f}(x_0)] - f(x_0)]^2}_{\textit{Bias}} + \underbrace{\mathbb{V}(\epsilon)}_{\textit{Irreducible error}} .$$

Bias-Variance Trade-off

Suppose we have an estimator $\hat{f}(x)$ from the training data.
Let (x_0, y_0) be a test observation drawn from the population.
True model is $Y = f(X) + \epsilon$ (with $f(x) = \mathbb{E}(Y|X = x)$).
Then the **expected test MSE** at x_0 is:

$$\mathbb{E}[(y_0 - \hat{f}(x_0))^2] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{\textit{Variance}} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{\textit{Bias}}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{\textit{Irreducible error}} .$$

In order to minimize the expected test error, we need to select a statistical learning method that simultaneously achieves low variance and low bias.

Bias-Variance Trade-off

Suppose we have an estimator $\hat{f}(x)$ from the training data.
Let (x_0, y_0) be a test observation drawn from the population.
True model is $Y = f(X) + \epsilon$ (with $f(x) = \mathbb{E}(Y|X = x)$).
Then the **expected test MSE** at x_0 is:

$$\mathbb{E}[(y_0 - \hat{f}(x_0))^2] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{\textit{Variance}} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{\textit{Bias}}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{\textit{Irreducible error}} .$$

In order to minimize the expected test error, we need to select a statistical learning method that simultaneously achieves low variance and low bias.
Note: the expected test MSE can never lie below $\mathbb{V}(\epsilon)$.

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{\textit{Variance}} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{\textit{Bias}}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{\textit{Irreducible error}} .$$

Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set.

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{\textit{Variance}} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{\textit{Bias}}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{\textit{Irreducible error}} .$$

Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set.
Different training data sets will result in a different $\hat{f}$.

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{Variance} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{Bias}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{Irreducible\ error} .$$

Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set.
Different training data sets will result in a different $\hat{f}$.
If a method has high variance then small changes in the training data can result in large changes in $\hat{f}$.

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{\textit{Variance}} + \underbrace{[\mathbb{E}[\hat{f}(x_0)] - f(x_0)]^2}_{\textit{Bias}} + \underbrace{\mathbb{V}(\epsilon)}_{\textit{Irreducible error}} .$$

Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set.
Different training data sets will result in a different $\hat{f}$.
If a method has high variance then small changes in the training data can result in large changes in $\hat{f}$.
Generally, more flexible methods have higher variance.

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{\textit{Variance}} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{\textit{Bias}}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{\textit{Irreducible error}} .$$

Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set.
Different training data sets will result in a different $\hat{f}$.
If a method has high variance then small changes in the training data can result in large changes in $\hat{f}$.
Generally, more flexible methods have higher variance.
Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model.

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{Variance} + \underbrace{[\mathbb{E}[\hat{f}(x_0)] - f(x_0)]^2}_{Bias} + \underbrace{\mathbb{V}(\epsilon)}_{Irreducible\ error} .$$

Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set.
Different training data sets will result in a different $\hat{f}$.
If a method has high variance then small changes in the training data can result in large changes in $\hat{f}$.
Generally, more flexible methods have higher variance.
Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model.
Example: linear regression assumes that there is a linear relationship between Y and $X_1, \ldots, X_p$.

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{\textit{Variance}} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{\textit{Bias}}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{\textit{Irreducible error}} .$$

Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set.
Different training data sets will result in a different $\hat{f}$.
If a method has high variance then small changes in the training data can result in large changes in $\hat{f}$.
Generally, more flexible methods have higher variance.
Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model.
Example: linear regression assumes that there is a linear relationship between Y and $X_1, \ldots, X_p$.
It is unlikely that any real-life problem truly has such a simple linear relationship, and so performing linear regression will undoubtedly result in some bias in the estimate of f.

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{Variance} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{Bias}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{Irreducible\ error} .$$

Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set.
Different training data sets will result in a different $\hat{f}$.
If a method has high variance then small changes in the training data can result in large changes in $\hat{f}$.
Generally, more flexible methods have higher variance.
Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model.
Example: linear regression assumes that there is a linear relationship between Y and $X_1, \ldots, X_p$.
It is unlikely that any real-life problem truly has such a simple linear relationship, and so performing linear regression will undoubtedly result in some bias in the estimate of f.
Generally, more flexible methods have lower bias.

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{Variance} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{Bias}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{Irreducible\ error} .$$

There is bias-variance trade-off when choosing a method !
Generally, more flexible methods have higher variance.
Generally, more flexible methods have lower bias.

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{Variance} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{Bias}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{Irreducible\ error} .$$

There is bias-variance trade-off when choosing a method !
This is referred to as a trade-off because:

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{\textit{Variance}} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{\textit{Bias}}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{\textit{Irreducible error}} .$$

There is bias-variance trade-off when choosing a method !
This is referred to as a trade-off because:

- It is easy to obtain a method with extremely low bias but high variance (for instance, by drawing a curve that passes through every single training observation).

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{\textit{Variance}} + \underbrace{[\mathbb{E}[\hat{f}(x_0)] - f(x_0)]^2}_{\textit{Bias}} + \underbrace{\mathbb{V}(\epsilon)}_{\textit{Irreducible error}} .$$

There is bias-variance trade-off when choosing a method !
This is referred to as a trade-off because:

- It is easy to obtain a method with extremely low bias but high variance (for instance, by drawing a curve that passes through every single training observation).
- It is easy to obtain a method with very low variance but high bias (by fitting a horizontal line to the data).

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{\textit{Variance}} + \underbrace{[\mathbb{E}[\hat{f}(x_0)] - f(x_0)]^2}_{\textit{Bias}} + \underbrace{\mathbb{V}(\epsilon)}_{\textit{Irreducible error}} .$$

There is bias-variance trade-off when choosing a method !
This is referred to as a trade-off because:

- It is easy to obtain a method with extremely low bias but high variance (for instance, by drawing a curve that passes through every single training observation).
- It is easy to obtain a method with very low variance but high bias (by fitting a horizontal line to the data).
- The challenge lies in finding a method for which both the variance and the squared bias are low.

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{Variance} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{Bias}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{Irreducible\ error} .$$

There is bias-variance trade-off when choosing a method !

In a real-life situation in which f is unobserved, it is generally NOT possible to explicitly compute the test MSE, bias, or variance for a statistical learning method.

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{Variance} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{Bias}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{Irreducible\ error} .$$

There is bias-variance trade-off when choosing a method !
In a real-life situation in which f is unobserved, it is generally NOT possible to explicitly compute the test MSE, bias, or variance for a statistical learning method.
One should ALWAYS keep the bias-variance trade-off in mind.

Bias-Variance Trade-off

$$\mathbb{E}[(Y_0 - \hat{f}(X))^2 | X = x_0] = \underbrace{\mathbb{V}(\hat{f}(x_0))}_{Variance} + [\underbrace{\mathbb{E}[\hat{f}(x_0)] - f(x_0)}_{Bias}]^2 + \underbrace{\mathbb{V}(\epsilon)}_{Irreducible\ error} .$$

There is bias-variance trade-off when choosing a method !
In a real-life situation in which f is unobserved, it is generally NOT possible to explicitly compute the test MSE, bias, or variance for a statistical learning method.
One should ALWAYS keep the bias-variance trade-off in mind.
We will explore very flexible methods that can eliminate bias. This does not guarantee that they will outperform a much simpler method such as linear regression.

Classification setting

Setup: Observe $\{(x_1, y_1), \ldots, (x_n, y_n)(x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$ where for $i = 1, \ldots, n$: $y_i = f(x_i) + \epsilon_i$.

Classification setting

Setup: Observe $\{(x_1, y_1), \ldots, (x_n, y_n)(x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$ where for $i = 1, \ldots, n$: $y_i = f(x_i) + \epsilon_i$.
Here the output (y_i) are qualitative.

Classification setting

Setup: Observe $\{(x_1, y_1), \ldots, (x_n, y_n)(x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$ where for $i = 1, \ldots, n$: $y_i = f(x_i) + \epsilon_i$.
Here the output (y_i) are qualitative.
Remember: f is not available and (ϵ_i) are random noises.

Classification setting

Setup: Observe $\{(x_1, y_1), \dots, (x_n, y_n)(x_{n+1}, y_{n+1}), \dots, (x_{n+T}, y_{n+T})\}$ where for $i = 1, \dots, n$: $y_i = f(x_i) + \epsilon_i$.
Here the output (y_i) are qualitative.
Remember: f is not available and (ϵ_i) are random noises.
Objective: Estimate f on the basis of training observations $\{(x_1, y_1), \dots, (x_n, y_n)\}$

Classification setting

Setup: Observe $\{(x_1, y_1), \ldots, (x_n, y_n)(x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$ where for $i = 1, \ldots, n$: $y_i = f(x_i) + \epsilon_i$.
Here the output (y_i) are qualitative.
Remember: f is not available and (ϵ_i) are random noises.
Objective: Estimate f on the basis of training observations $\{(x_1, y_1), \ldots, (x_n, y_n)\}$
Common Approach: Quantify the accuracy of our estimate $\hat{f}$ with the training error rate:

$$TER = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(y_i \neq \hat{y}_i).$$

Classification setting

Setup: Observe $\{(x_1, y_1), \ldots, (x_n, y_n)(x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$ where for $i = 1, \ldots, n$: $y_i = f(x_i) + \epsilon_i$.
Here the output (y_i) are qualitative.
Remember: f is not available and (ϵ_i) are random noises.
Objective: Estimate f on the basis of training observations $\{(x_1, y_1), \ldots, (x_n, y_n)\}$
Common Approach: Quantify the accuracy of our estimate $\hat{f}$ with the training error rate:

$$TER = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(y_i \neq \hat{y}_i).$$

$\hat{y}_i$ is the predicted class label for the i^{th} observation using $\hat{f}$.

Classification setting

Setup: Observe $\{(x_1, y_1), \dots, (x_n, y_n)(x_{n+1}, y_{n+1}), \dots, (x_{n+T}, y_{n+T})\}$ where for $i = 1, \dots, n$: $y_i = f(x_i) + \epsilon_i$.
Here the output (y_i) are qualitative.
Remember: f is not available and (ϵ_i) are random noises.
Objective: Estimate f on the basis of training observations $\{(x_1, y_1), \dots, (x_n, y_n)\}$
Common Approach: Quantify the accuracy of our estimate $\hat{f}$ with the training error rate:

$$TER = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(y_i \neq \hat{y}_i).$$

$\hat{y}_i$ is the predicted class label for the i^{th} observation using $\hat{f}$.
TER computes the fraction of incorrect classifications.

Classification setting

Setup: Observe $\{(x_1, y_1), \ldots, (x_n, y_n)(x_{n+1}, y_{n+1}), \ldots, (x_{n+T}, y_{n+T})\}$ where for $i = 1, \ldots, n$: $y_i = f(x_i) + \epsilon_i$.
Here the output (y_i) are qualitative.
Remember: f is not available and (ϵ_i) are random noises.
Objective: Estimate f on the basis of training observations $\{(x_1, y_1), \ldots, (x_n, y_n)\}$
Common Approach: Quantify the accuracy of our estimate $\hat{f}$ with the training error rate:

$$TER = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(y_i \neq \hat{y}_i).$$

$\hat{y}_i$ is the predicted class label for the i^{th} observation using $\hat{f}$.
TER computes the fraction of incorrect classifications.
As in the regression setting, we are most interested in the error rates that result from applying our classifier to test observations that were not used in training.

Bayes Classifier

Important: It is possible to show that the test error rate is minimized, on average, by a very simple classifier that assigns each observation to the most likely class, given its predictor values.

Bayes Classifier

Important: It is possible to show that the test error rate is minimized, on average, by a very simple classifier that assigns each observation to the most likely class, given its predictor values.
We should simply assign a test observation with predictor vector x_0 to the class j for which $\mathbb{P}[Y = j | X = x_0]$ is the largest.

Bayes Classifier

Important: It is possible to show that the test error rate is minimized, on average, by a very simple classifier that assigns each observation to the most likely class, given its predictor values.
We should simply assign a test observation with predictor vector x_0 to the class j for which $\mathbb{P}[Y = j|X = x_0]$ is the largest.
This very simple classifier is called the **Bayes classifier**.

Bayes Classifier

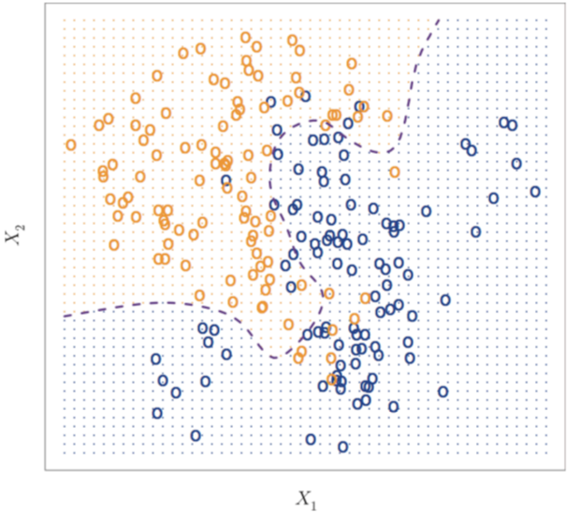
Important: It is possible to show that the test error rate is minimized, on average, by a very simple classifier that assigns each observation to the most likely class, given its predictor values.
We should simply assign a test observation with predictor vector x_0 to the class j for which $\mathbb{P}[Y = j | X = x_0]$ is the largest.
This very simple classifier is called the **Bayes classifier**.
Example: (y_i) are only distributed between two classes. I have a new unobserved x_{n+1}. What's the best guess for y_{n+1} ?

Bayes Classifier

Important: It is possible to show that the test error rate is minimized, on average, by a very simple classifier that assigns each observation to the most likely class, given its predictor values.
We should simply assign a test observation with predictor vector x_0 to the class j for which $\mathbb{P}[Y = j|X = x_0]$ is the largest.
This very simple classifier is called the **Bayes classifier**.
Example: (y_i) are only distributed between two classes. I have a new unobserved x_{n+1}. What's the best guess for y_{n+1} ?
In this case, the Bayes classifier corresponds to predicting class one if $\mathbb{P}[Y = 1|X = x_{n+1}] > 0.5$.

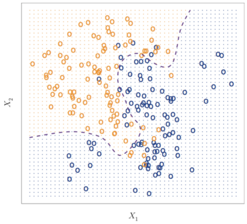
Bayes Classifier

Numerical Example: Simulated data set in a two-dimensional space consisting of predictors X_1 and X_2.



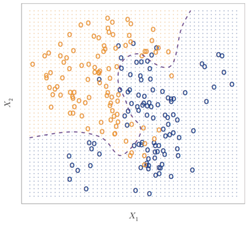
Bayes Classifier

Numerical Example: Simulated data set in a two-dimensional space consisting of predictors X_1 and X_2.
The orange and blue circles correspond to training observations that belong to two different classes.



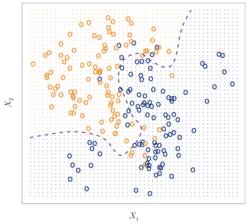
Bayes Classifier

Numerical Example: Simulated data set in a two-dimensional space consisting of predictors X_1 and X_2.
The orange and blue circles correspond to training observations that belong to two different classes.
For each value of X_1 and X_2, there is a different probability of the response being orange or blue.



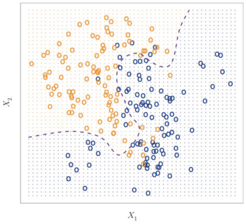
Bayes Classifier

Numerical Example: Simulated data set in a two-dimensional space consisting of predictors X_1 and X_2.
The orange and blue circles correspond to training observations that belong to two different classes.
For each value of X_1 and X_2, there is a different probability of the response being orange or blue.
Since this is simulated data, we know how the data were generated and we can calculate the conditional probabilities for each value of X_1 and X_2.



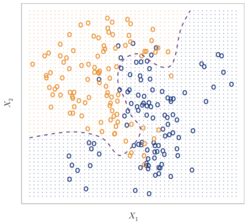
Bayes Classifier

Numerical Example: Simulated data set in a two-dimensional space consisting of predictors X_1 and X_2.
The orange and blue circles correspond to training observations that belong to two different classes.
For each value of X_1 and X_2, there is a different probability of the response being orange or blue.
Since this is simulated data, we know how the data were generated and we can calculate the conditional probabilities for each value of X_1 and X_2.
Orange region = points for which $\mathbb{P}[Y = \mathit{orange}|X] > 0.5$. Blue region= points for which $\mathbb{P}[Y = \mathit{orange}|X] < 0.5$.



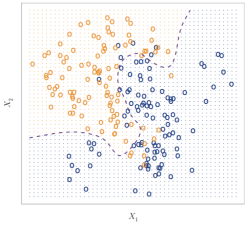
Bayes Classifier

Numerical Example: Simulated data set in a two-dimensional space consisting of predictors X_1 and X_2.
The orange and blue circles correspond to training observations that belong to two different classes.
For each value of X_1 and X_2, there is a different probability of the response being orange or blue.
Since this is simulated data, we know how the data were generated and we can calculate the conditional probabilities for each value of X_1 and X_2.
Orange region = points for which $\mathbb{P}[Y = orange|X] > 0.5$. Blue region= points for which $\mathbb{P}[Y = orange|X] < 0.5$.
The purple dashed line represents the points where the $\mathbb{P}$ is exactly 50%.



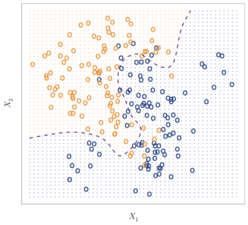
Bayes Classifier

Numerical Example: Simulated data set in a two-dimensional space consisting of predictors X_1 and X_2.
The orange and blue circles correspond to training observations that belong to two different classes.
For each value of X_1 and X_2, there is a different probability of the response being orange or blue.
Since this is simulated data, we know how the data were generated and we can calculate the conditional probabilities for each value of X_1 and X_2.
Orange region = points for which $\mathbb{P}[Y = orange|X] > 0.5$. Blue region= points for which $\mathbb{P}[Y = orange|X] < 0.5$.
The purple dashed line represents the points where the $\mathbb{P}$ is exactly 50%.
This is called the **Bayes decision boundary**.



Bayes Classifier

Numerical Example: Simulated data set in a two-dimensional space consisting of predictors X_1 and X_2.
Orange region = points for which $\mathbb{P}[Y = orange|X] > 0.5$. Blue region= points for which $\mathbb{P}[Y = orange|X] < 0.5$.
The purple dashed line represents the points where the $\mathbb{P}$ is exactly 50%.
This is called the **Bayes decision boundary**.
A new observation that falls on the orange side of the boundary will be assigned to the orange class, and similarly an observation on the blue side of the boundary will be assigned to the blue class.



KNN

Theory: Use the Bayes classifier.

KNN

Theory: Use the Bayes classifier.
Problem: With real data, we do not know the conditional distribution of Y given $X \implies$ computing the Bayes classifier is impossible.

KNN

Theory: Use the Bayes classifier.
Problem: With real data, we do not know the conditional distribution of Y given $X \implies$ computing the Bayes classifier is impossible.
Benchmark: Bayes classifier serves as an unattainable gold standard against which to compare other methods.

KNN

Theory: Use the Bayes classifier.
Problem: With real data, we do not know the conditional distribution of Y given $X \implies$ computing the Bayes classifier is impossible.
Benchmark: Bayes classifier serves as an unattainable gold standard against which to compare other methods.
What to do now ?: Given a real dataset, how can you estimate f ?

KNN

Theory: Use the Bayes classifier.
Problem: With real data, we do not know the conditional distribution of Y given $X \implies$ computing the Bayes classifier is impossible.
Benchmark: Bayes classifier serves as an unattainable gold standard against which to compare other methods.
What to do now ?: Given a real dataset, how can you estimate f ?
Idea: Estimate the conditional distribution of Y given X, and classify a given observation to the class with highest *estimated* probability.

KNN

Theory: Use the Bayes classifier.
Problem: With real data, we do not know the conditional distribution of Y given $X \implies$ computing the Bayes classifier is impossible.
Benchmark: Bayes classifier serves as an unattainable gold standard against which to compare other methods.
What to do now ?: Given a real dataset, how can you estimate f ?
Idea: Estimate the conditional distribution of Y given X, and classify a given observation to the class with highest *estimated* probability.
How ?: K-nearest neighbors is ONE method that do that.

KNN

Theory: Use the Bayes classifier.
Problem: With real data, we do not know the conditional distribution of Y given $X \implies$ computing the Bayes classifier is impossible.
Benchmark: Bayes classifier serves as an unattainable gold standard against which to compare other methods.
What to do now ?: Given a real dataset, how can you estimate f ?
Idea: Estimate the conditional distribution of Y given X, and classify a given observation to the class with highest *estimated* probability.
How ?: K-nearest neighbors is ONE method that do that.
KNN: Given a positive integer K and a test observation x_{n+1}, the KNN classifier first identifies the K points in the training data that are closest to x_{n+1}, represented by $\mathcal{N}_0$. It then estimates the conditional probability for class j as the fraction of points in $\mathcal{N}_0$ whose response values equal j:

$$\hat{\mathbb{P}}[Y = j | X = x_{n+1}] = \frac{1}{K} \sum_{i \in \mathcal{N}_0} \mathbb{1}(y_i = j).$$

KNN

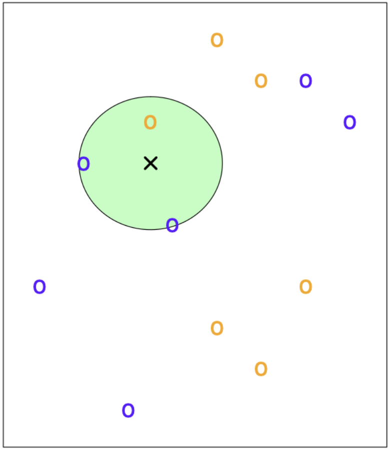
Theory: Use the Bayes classifier.
Problem: With real data, we do not know the conditional distribution of Y given $X \implies$ computing the Bayes classifier is impossible.
Benchmark: Bayes classifier serves as an unattainable gold standard against which to compare other methods.
What to do now ?: Given a real dataset, how can you estimate f ?
Idea: Estimate the conditional distribution of Y given X, and classify a given observation to the class with highest *estimated* probability.
How ?: K-nearest neighbors is ONE method that do that.
KNN: Given a positive integer K and a test observation x_{n+1}, the KNN classifier first identifies the K points in the training data that are closest to x_{n+1}, represented by $\mathcal{N}_0$. It then estimates the conditional probability for class j as the fraction of points in $\mathcal{N}_0$ whose response values equal j:

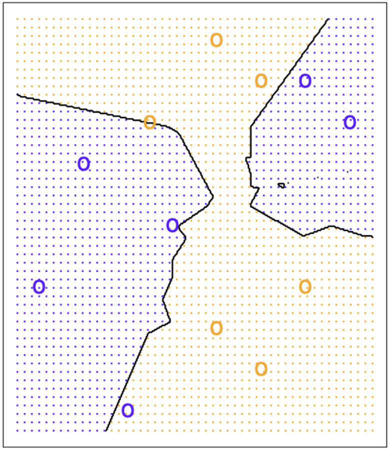
$$\hat{\mathbb{P}}[Y = j | X = x_{n+1}] = \frac{1}{K} \sum_{i \in \mathcal{N}_0} \mathbb{1}(y_i = j).$$

Finally, KNN classifies the test observation x_{n+1} to the class with the largest probability.

KNN

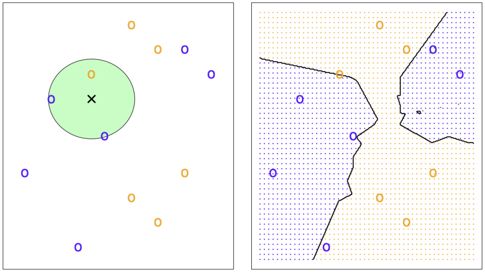
The KNN approach, using K = 3, is illustrated in a simple situation with six blue observations and six orange observations.





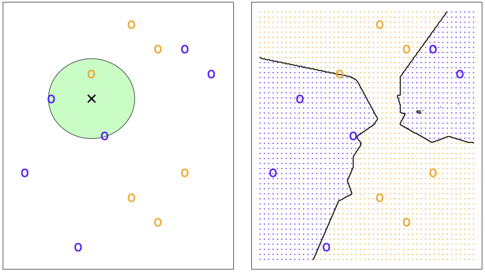
KNN

The KNN approach, using K = 3, is illustrated in a simple situation with six blue observations and six orange observations.
Left: A test observation at which a predicted class label is desired is shown as a black cross. The three closest points to the test observation are identified, and it is predicted that the test observation belongs to the most commonly-occurring class, in this case blue.



KNN

The KNN approach, using K = 3, is illustrated in a simple situation with six blue observations and six orange observations.
Left: A test observation at which a predicted class label is desired is shown as a black cross. The three closest points to the test observation are identified, and it is predicted that the test observation belongs to the most commonly-occurring class, in this case blue.
Right: The KNN decision boundary for this example is shown in black. The blue grid indicates the region in which a test observation will be assigned to the blue class, and the orange grid indicates the region in which it will be assigned to the orange class.



KNN

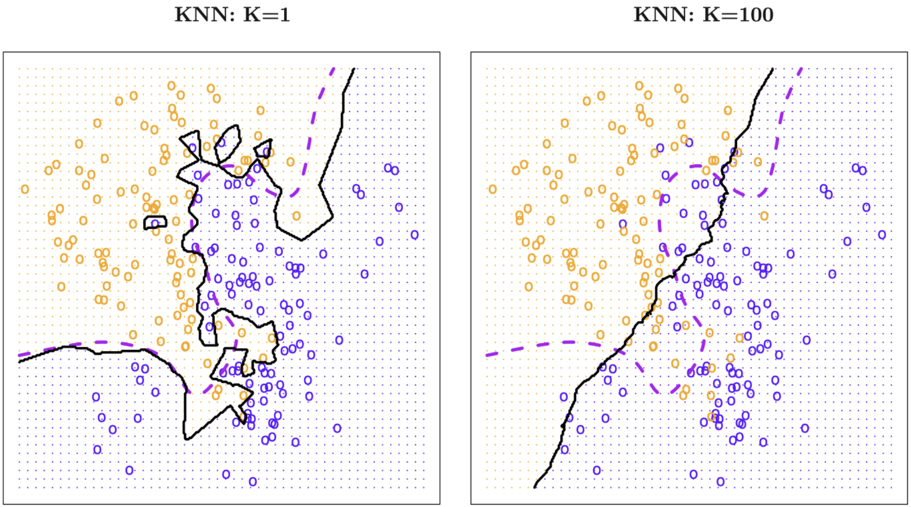
How to choose K ?

KNN

Bias-variance trade-off: As K diminishes, the method becomes more flexible !

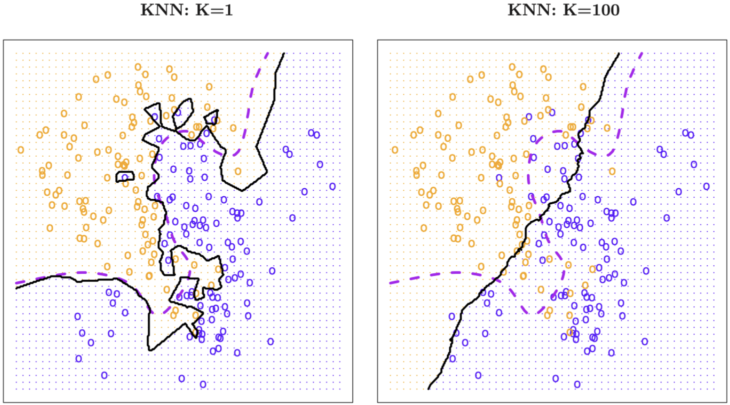
KNN

Bias-variance trade-off: As K diminishes, the method becomes more flexible !



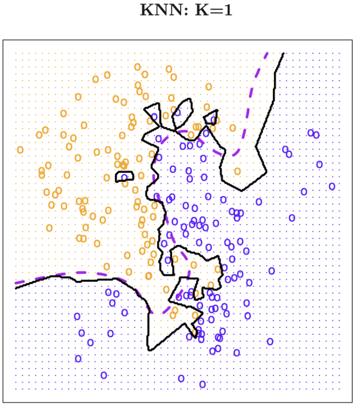
KNN

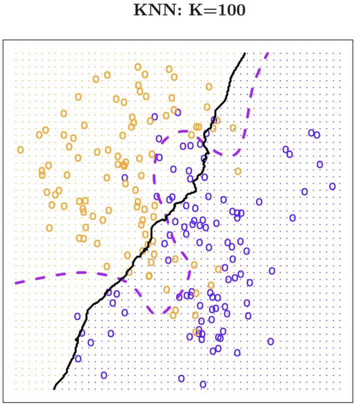
Bias-variance trade-off: As K diminishes, the method becomes more flexible ! When K $= 1$, the decision boundary is overly flexible and finds patterns in the data that don't correspond to the Bayes decision boundary. This corresponds to a classifier that has **low bias** but very **high variance**.



KNN

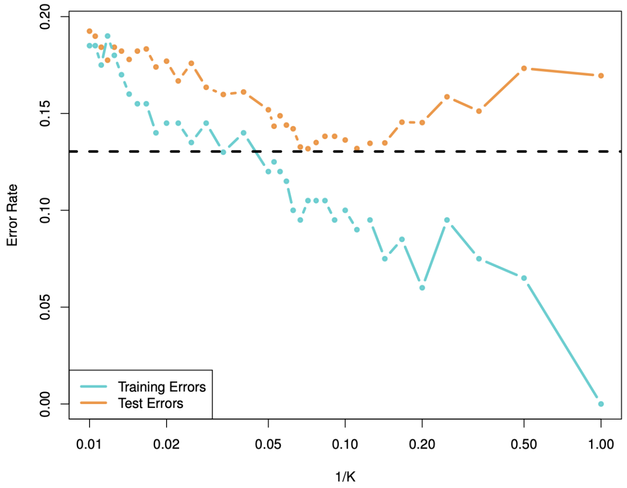
Bias-variance trade-off: As *K* diminishes, the method becomes more flexible ! When K = 100, the method becomes less flexible and produces a decision boundary that is close to linear. This corresponds to a **low-variance** but **high-bias classifier**.





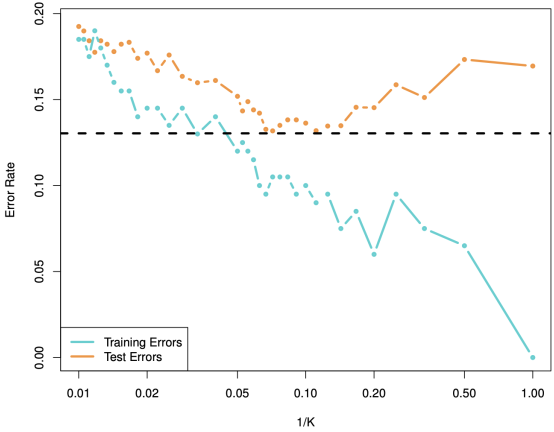
KNN

The KNN training error rate (blue, 200 observations) and test error rate (orange, 5,000 observations) on the data set, as the level of flexibility (assessed using 1/K on the log scale) increases.



KNN

As 1/K increases, the method becomes more flexible. As in the regression setting, the training error rate consistently declines as the flexibility increases. However, the test error exhibits a characteristic U-shape, declining at first (minimum at $\approx K = 10$) before increasing again when the method becomes excessively flexible and overfits.



Curse of dimensionality

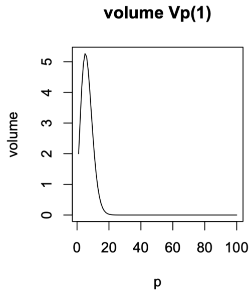
- It looks like the more features you collect, the more accurate your learning method will be.

Curse of dimensionality

- It looks like the more features you collect, the more accurate your learning method will be.
- As the number of features (dimensions) increases, the volume of the data space grows exponentially. The volume $V_p(r)$ of a p-dimensional ball of radius $r > 0$ is equal to

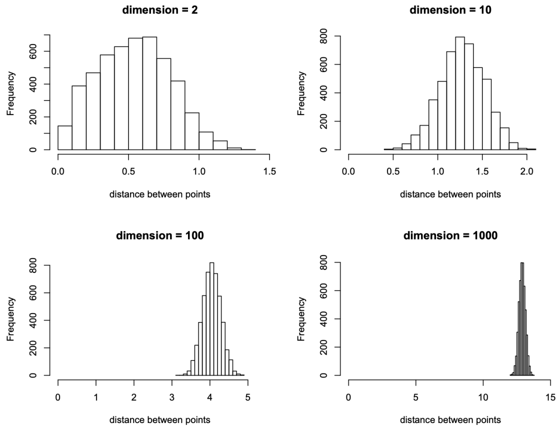
$$V_p(r) = r^p \frac{\pi^{p/2}}{\Gamma(p/2+1)} \sim_{p\to\infty} \left(\frac{2e\pi r^2}{p}\right)^{p/2} (p\pi){-}1/2).$$

The volume $V_p(r)$ of a ball of radius r goes to zero more than exponentially fast with the dimension p !!



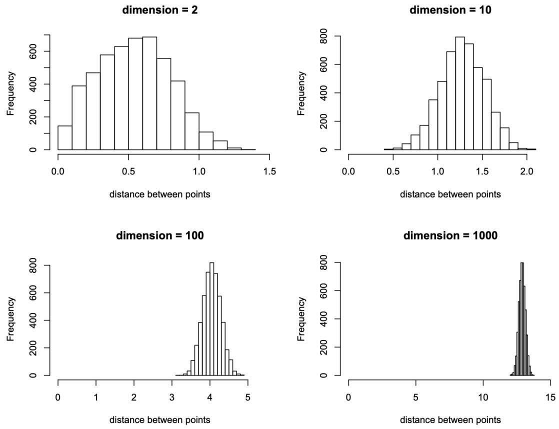
Curse of dimensionality

- Histograms of the pairwise-distances between $n = 100$ points sampled uniformly in the hypercube $[0, 1]^p$ for $p = 2,10,100,$ and 1000.



Curse of dimensionality

- Histograms of the pairwise-distances between $n = 100$ points sampled uniformly in the hypercube $[0, 1]^p$ for $p = 2{,}10{,}100$, and 1000.
- We observe that, when the dimension p increases, the minimal distance between two points increases and all the points are at a similar distance from the others, so the notion of "nearest points" vanishes.



Curse of dimensionality

- Many algorithms that work well in low-dimensional spaces become less effective or even fail in high-dimensional spaces.

Curse of dimensionality

- Many algorithms that work well in low-dimensional spaces become less effective or even fail in high-dimensional spaces.
- In low-dimensional spaces (e.g. 2D), KNN works well because the data is densely packed and the concept of nearness is meaningful.

Curse of dimensionality

- Many algorithms that work well in low-dimensional spaces become less effective or even fail in high-dimensional spaces.
- In low-dimensional spaces (e.g. 2D), KNN works well because the data is densely packed and the concept of nearness is meaningful.
- The algorithm can effectively find the k nearest neighbors and make accurate predictions.

Curse of dimensionality

- Many algorithms that work well in low-dimensional spaces become less effective or even fail in high-dimensional spaces.
- In low-dimensional spaces (e.g. 2D), KNN works well because the data is densely packed and the concept of nearness is meaningful.
- The algorithm can effectively find the k nearest neighbors and make accurate predictions.
- In high-dimensional spaces (e.g. 100D), KNN becomes less effective because the data is sparse and the concept of nearness becomes less meaningful.

Curse of dimensionality

- Many algorithms that work well in low-dimensional spaces become less effective or even fail in high-dimensional spaces.
- In low-dimensional spaces (e.g. 2D), KNN works well because the data is densely packed and the concept of nearness is meaningful.
- The algorithm can effectively find the k nearest neighbors and make accurate predictions.
- In high-dimensional spaces (e.g. 100D), KNN becomes less effective because the data is sparse and the concept of nearness becomes less meaningful.
- The algorithm struggles to find the k nearest neighbors and makes less accurate predictions.

Curse of dimensionality

- Many algorithms that work well in low-dimensional spaces become less effective or even fail in high-dimensional spaces.
- In low-dimensional spaces (e.g. 2D), KNN works well because the data is densely packed and the concept of nearness is meaningful.
- The algorithm can effectively find the k nearest neighbors and make accurate predictions.
- In high-dimensional spaces (e.g. 100D), KNN becomes less effective because the data is sparse and the concept of nearness becomes less meaningful.
- The algorithm struggles to find the k nearest neighbors and makes less accurate predictions.
- As the number of dimensions increases, the distance between data points increases exponentially.

Curse of dimensionality

- Many algorithms that work well in low-dimensional spaces become less effective or even fail in high-dimensional spaces.
- In low-dimensional spaces (e.g. 2D), KNN works well because the data is densely packed and the concept of nearness is meaningful.
- The algorithm can effectively find the k nearest neighbors and make accurate predictions.
- In high-dimensional spaces (e.g. 100D), KNN becomes less effective because the data is sparse and the concept of nearness becomes less meaningful.
- The algorithm struggles to find the k nearest neighbors and makes less accurate predictions.
- As the number of dimensions increases, the distance between data points increases exponentially.
- This makes it harder for KNN to find the k nearest neighbors, leading to decreased accuracy.

References

- Introduction to Statistical Learning with applications in R, Gareth James, Daniela Witten, Trevor Hastie & Robert Tibshirani, Springer
- Introduction to high-dimensional statistics, Christophe Giraud, Chapman and Hall/CRC.