Statistics 101C

Shirong Xu

University of California, Los Angeles

shirong@stat.ucla.edu

October 2, 2024

Two Main Problems - Recall

We observe a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^p$ is p-dimensional predictors. By the type of response Y, there are two **learning problems**:

- **Regression**: The response Y is quantitative. For example, people's income, the value of a house, blood pressure of patient.
- **Classification**: The response Y is qualitative: binary (gender, like or dislike a product), categorical (brand of a product), and ordinal (ratings given by users to movies or restaurant)

Regression - Recall

For a dataset $D = \{x_i, y_i\}_{i=1}^n$

- Training error:

$$\widehat{f} = \underbrace{\arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2}_{\text{Minimize training error}}$$

- Testing Error: Evaluate $\widehat{f}$ at a new point (x_0, y_0):

$$(\widehat{f}(x_0) - y_0)^2$$

- Consider a theoretical analysis of $(\widehat{f}(x_0) - y_0)^2$ when x_0 is given

$$\text{Theoretical Test Error} : \mathbb{E}_D \mathbb{E}_{y_0} (\widehat{f}(x_0) - y_0)^2)$$

- There exists bias-variance trade-off in **Theoretical Test Error**.

Contents

Binary Classification

- Let $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^p$ be p-dimensional covariates and $y \in \{-1, 1\}$ (or $y \in \{0, 1\}$) be binary response. Example:

	Status	Length	Left	Right	Bottom	Top	Diagonal
159	counterfeit	214.8	130.4	130.6	12.5	10.0	139.3
179	counterfeit	214.8	130.5	130.3	10.2	12.1	139.1
14	genuine	214.7	129.7	129.7	7.7	10.9	141.7
195	counterfeit	214.9	130.3	130.5	11.6	10.6	139.8
170	counterfeit	214.9	130.0	129.9	11.4	10.9	139.7
50	genuine	214.5	129.0	129.6	7.8	9.8	142.0
118	counterfeit	214.7	130.4	130.1	12.1	10.4	139.9
43	genuine	214.9	129.6	129.4	9.3	9.0	141.7
198	counterfeit	214.8	130.3	130.4	10.6	11.1	140.0
194	counterfeit	215.0	130.5	130.3	9.6	11.0	138.5
153	counterfeit	214.6	129.7	129.3	10.4	11.0	139.3

Binary Classification

- Let $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^p$ be p-dimensional covariates and $y \in \{-1, 1\}$ (or $y \in \{0, 1\}$) be binary response. Example:

	Status	Length	Left	Right	Bottom	Top	Diagonal
159	counterfeit	214.8	130.4	130.6	12.5	10.0	139.3
179	counterfeit	214.8	130.5	130.3	10.2	12.1	139.1
14	genuine	214.7	129.7	129.7	7.7	10.9	141.7
195	counterfeit	214.9	130.3	130.5	11.6	10.6	139.8
170	counterfeit	214.9	130.0	129.9	11.4	10.9	139.7
50	genuine	214.5	129.0	129.6	7.8	9.8	142.0
118	counterfeit	214.7	130.4	130.1	12.1	10.4	139.9
43	genuine	214.9	129.6	129.4	9.3	9.0	141.7
198	counterfeit	214.8	130.3	130.4	10.6	11.1	140.0
194	counterfeit	215.0	130.5	130.3	9.6	11.0	138.5
153	counterfeit	214.6	129.7	129.3	10.4	11.0	139.3

- **Objective**: estimate a classifier to accurately find counterfeit banknote.
- Let $f : \mathcal{X} \to \{-1, 1\}$ be a classifier.

Binary Classification

- **Question**: Given a sample $(\boldsymbol{x}, y)$, how to measure the performance of f on this sample?

Binary Classification

- **Question**: Given a sample $(\boldsymbol{x}, y)$, how to measure the performance of f on this sample?
- Binary Loss function (0-1 loss):

$$L(f(\boldsymbol{x}), y) = I(f(\boldsymbol{x}) \neq y) = \begin{cases} 1, & \text{if } f(\boldsymbol{x}) \neq y \\ 0, & \text{if } f(\boldsymbol{x}) = y \end{cases}$$

where $I(\cdot)$ is the indicator function.

Binary Classification

- **Question**: Given a sample $(\boldsymbol{x}, y)$, how to measure the performance of f on this sample?
- Binary Loss function (0-1 loss):

$$L(f(\boldsymbol{x}), y) = I(f(\boldsymbol{x}) \neq y) = \begin{cases} 1, & \text{if } f(\boldsymbol{x}) \neq y \\ 0, & \text{if } f(\boldsymbol{x}) = y \end{cases}$$

  where $I(\cdot)$ is the indicator function.
- **An implicit assumption** of binary loss is: two types of errors are equally treated.

Binary Classification

- **Question**: Given a sample $(\boldsymbol{x}, y)$, how to measure the performance of f on this sample?
- Binary Loss function (0-1 loss):

$$L(f(\boldsymbol{x}), y) = I(f(\boldsymbol{x}) \neq y) = \begin{cases} 1, & \text{if } f(\boldsymbol{x}) \neq y \\ 0, & \text{if } f(\boldsymbol{x}) = y \end{cases}$$

  where $I(\cdot)$ is the indicator function.
- **An implicit assumption** of binary loss is: two types of errors are equally treated.
- The risk of f (the averaged loss):

$$R(f) = \mathbb{E}\big[L(f(\boldsymbol{x}), y)\big] = \mathbb{P}\Big(f(\boldsymbol{x}) \neq y\Big),$$

  where the expectation is taken with respect to the distribution of $(\boldsymbol{x}, y)$.

Bayes Classifier

- A closer look at the risk $R(f)$:

$$\begin{aligned} R(f) =& \mathbb{E}_x\big(\mathbb{E}_y\big[L(f(\boldsymbol{x}), y)|\boldsymbol{x}\big]\big) \\ =& \int_{\mathcal{X}} \mathbb{P}(\boldsymbol{x})\Big(\eta(\boldsymbol{x})I(f(\boldsymbol{x}) \neq 1) + \big(1-\eta(\boldsymbol{x})\big)I(f(\boldsymbol{x}) \neq -1)\Big)d\boldsymbol{x} \end{aligned}$$

Bayes Classifier

- A closer look at the risk $R(f)$:

$$\begin{aligned} R(f) =& \mathbb{E}_x\big(\mathbb{E}_y\big[L(f(\boldsymbol{x}), y)|\boldsymbol{x}\big]\big) \\ =& \int_{\mathcal{X}} \mathbb{P}(\boldsymbol{x})\Big(\eta(\boldsymbol{x})I(f(\boldsymbol{x}) \neq 1) + \big(1-\eta(\boldsymbol{x})\big)I(f(\boldsymbol{x}) \neq -1)\Big)d\boldsymbol{x} \end{aligned}$$

- Given a covariate $\boldsymbol{x}$, we have

$$\begin{aligned} f(\boldsymbol{x}) = 1 &\Rightarrow \mathbb{E}_y\big[L(f(\boldsymbol{x}), y)|\boldsymbol{x}\big] = 1-\eta(\boldsymbol{x}) \\ f(\boldsymbol{x}) = -1 &\Rightarrow \mathbb{E}_y\big[L(f(\boldsymbol{x}), y)|\boldsymbol{x}\big] = \eta(\boldsymbol{x}) \end{aligned}$$

Bayes Classifier

- A closer look at the risk $R(f)$:

$$\begin{aligned} R(f) =& \mathbb{E}_x\big(\mathbb{E}_y\big[L(f(\boldsymbol{x}), y)|\boldsymbol{x}\big]\big) \\ =& \int_{\mathcal{X}} \mathbb{P}(\boldsymbol{x})\Big(\eta(\boldsymbol{x})I(f(\boldsymbol{x}) \neq 1) + \big(1-\eta(\boldsymbol{x})\big)I(f(\boldsymbol{x}) \neq -1)\Big)d\boldsymbol{x} \end{aligned}$$

- Given a covariate $\boldsymbol{x}$, we have

$$\begin{aligned} f(\boldsymbol{x}) = 1 &\Rightarrow \mathbb{E}_y\big[L(f(\boldsymbol{x}), y)|\boldsymbol{x}\big] = 1-\eta(\boldsymbol{x}) \\ f(\boldsymbol{x}) = -1 &\Rightarrow \mathbb{E}_y\big[L(f(\boldsymbol{x}), y)|\boldsymbol{x}\big] = \eta(\boldsymbol{x}) \end{aligned}$$

- The optimal classifier f^* (Bayes classifier) is defined as

$$f^*(\boldsymbol{x}) = sign(\eta(\boldsymbol{x}) - 1/2) = \begin{cases} 1, & \text{if } \eta(\boldsymbol{x}) > 1/2 \\ -1, & \text{if } \eta(\boldsymbol{x}) < 1/2 \end{cases}$$

Measure the quality of $\widehat{f}$

- Generally, we obtain a classifier $\widehat{f}$ based on a training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.
  - **Question**: Can we use $R(\widehat{f})$ to measure the quality of $\widehat{f}$ in reality?

Measure the quality of $\widehat{f}$

- Generally, we obtain a classifier $\widehat{f}$ based on a training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.
  - **Question**: Can we use $R(\widehat{f})$ to measure the quality of $\widehat{f}$ in reality?
  - **Answer**: No, since we do not have the distribution of $(\boldsymbol{x}, y)$ in reality.

Measure the quality of $\widehat{f}$

- Generally, we obtain a classifier $\widehat{f}$ based on a training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.
  - **Question**: Can we use $R(\widehat{f})$ to measure the quality of $\widehat{f}$ in reality?
  - **Answer**: No, since we do not have the distribution of $(\boldsymbol{x}, y)$ in reality.

- Usually, we have an another dataset $\mathcal{D}_{test} = \{(\widetilde{\boldsymbol{x}}_i, \widetilde{y}_i)\}_{i=1}^m$, therefore we use the following approximation:

$$\widehat{R}(\widehat{f}) = \frac{1}{m}\sum_{i=1}^{m} I(\widehat{f}(\widetilde{\boldsymbol{x}}_i) \neq \widetilde{y}_i) \rightarrow \text{ Testing Error}$$

Measure the quality of $\widehat{f}$

- Generally, we obtain a classifier $\widehat{f}$ based on a training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.
  - **Question**: Can we use $R(\widehat{f})$ to measure the quality of $\widehat{f}$ in reality?
  - **Answer**: No, since we do not have the distribution of $(\boldsymbol{x}, y)$ in reality.

- Usually, we have an another dataset $\mathcal{D}_{test} = \{(\widetilde{\boldsymbol{x}}_i, \widetilde{y}_i)\}_{i=1}^m$, therefore we use the following approximation:

$$\widehat{R}(\widehat{f}) = \frac{1}{m}\sum_{i=1}^{m} I(\widehat{f}(\widetilde{\boldsymbol{x}}_i) \neq \widetilde{y}_i) \rightarrow \text{ Testing Error}$$

- By Law of Large Number (LLN), we have $\widehat{R}(f)$ converges to $R(f)$ in probability for any f.

- **Question**: Can we use the training error to measure the performance of $\widehat{f}$?

$$\widehat{R}_{train}(\widehat{f}) = \frac{1}{n}\sum_{i=1}^{n} I(\widehat{f}(\boldsymbol{x}_i) \neq y_i) \rightarrow \text{ Training Error}$$

- **Question**: Can we use the training error to measure the performance of $\widehat{f}$?

$$\widehat{R}_{train}(\widehat{f}) = \frac{1}{n}\sum_{i=1}^{n} I(\widehat{f}(\boldsymbol{x}_i) \neq y_i) \rightarrow \text{ Training Error}$$

- **Answer**: No, since the training error $\widehat{R}_{train}(\widehat{f})$ is significantly biased in most cases, particularly in the domain of deep learning.

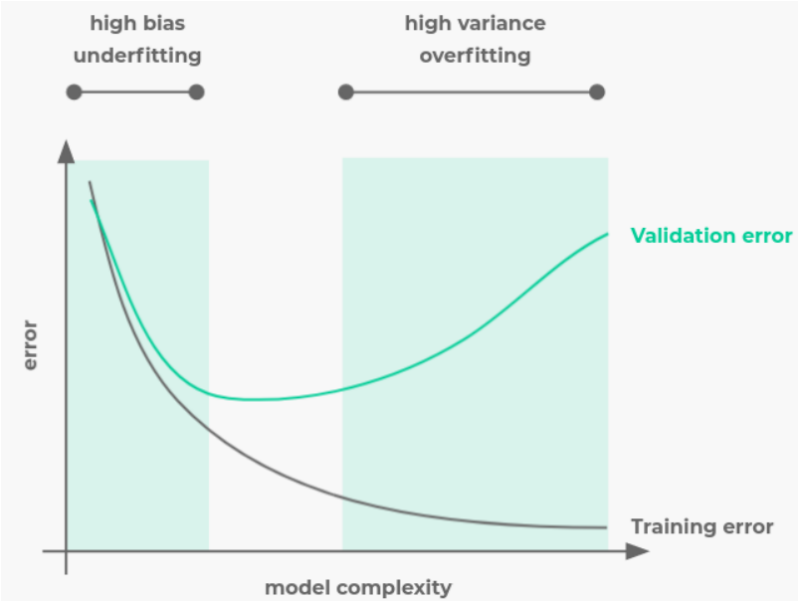


Figure: The figure is from https://quantdare.com/mitigating-overfitting-neural-networks/

- **Question**: Can we use the training error to measure the performance of $\widehat{f}$?

$$\widehat{R}_{train}(\widehat{f}) = \frac{1}{n}\sum_{i=1}^{n} I(\widehat{f}(\boldsymbol{x}_i) \neq y_i) \rightarrow \text{ Training Error}$$

- **Answer**: No, since the training error $\widehat{R}_{train}(\widehat{f})$ is significantly biased in most cases, particularly in the domain of deep learning.

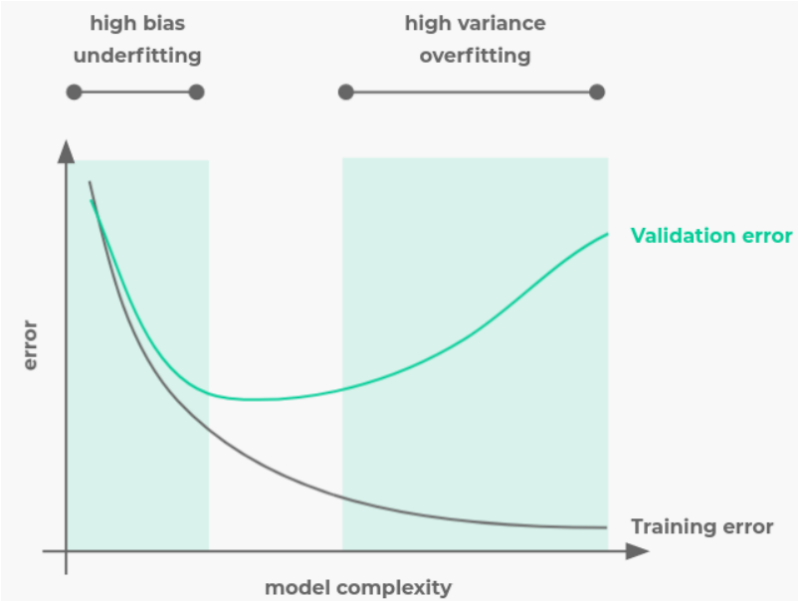


Figure: The figure is from https://quantdare.com/mitigating-overfitting-neural-networks/

- Notice that $I(\widehat{f}(\boldsymbol{x}_i) \neq y_i), i = 1, \ldots, n$ are not independent, so LLN cannot be applied here.

Other metrics in classification

- Confusion matrix:

	$\hat{f}(x) = 1$	$\hat{f}(x) = -1$
$Y = 1$	43 TP	11 FN
$Y = -1$	12 FP	34 TN

  - True Positive (TP): the truth is positive, and the prediction is positive.
  - True Negative (TN): the truth is negative, and the prediction is negative.
  - False Negative (FN): the truth is positive, and the prediction is negative.
  - False Positive (FP): the truth is negative, and the prediction is positive.

Other metrics in classification

- True Positive Rate (TPR): the percentage that positive samples are classified as positive.

$$TPR = \frac{TP}{TP + FN}$$

Other metrics in classification

- True Positive Rate (TPR): the percentage that positive samples are classified as positive.

$$TPR = \frac{TP}{TP + FN}$$

- False Positive Rate (FPR): the percentage that negative samples are classified as positive.

$$FPR = \frac{FP}{FP + TN}$$

Other metrics in classification

- True Positive Rate (TPR): the percentage that positive samples are classified as positive.

$$TPR = \frac{TP}{TP + FN}$$

- False Positive Rate (FPR): the percentage that negative samples are classified as positive.

$$FPR = \frac{FP}{FP + TN}$$

- True Negative Rate (TNR): the percentage that negative samples are classified as negative.

$$TNR = \frac{TN}{FP + TN}$$

Other metrics in classification

- True Positive Rate (TPR): the percentage that positive samples are classified as positive.

$$TPR = \frac{TP}{TP + FN}$$

- False Positive Rate (FPR): the percentage that negative samples are classified as positive.

$$FPR = \frac{FP}{FP + TN}$$

- True Negative Rate (TNR): the percentage that negative samples are classified as negative.

$$TNR = \frac{TN}{FP + TN}$$

- **Question**: What is the definition of False Negative Rate (FNR)?

Why do we need TNR and TPR?

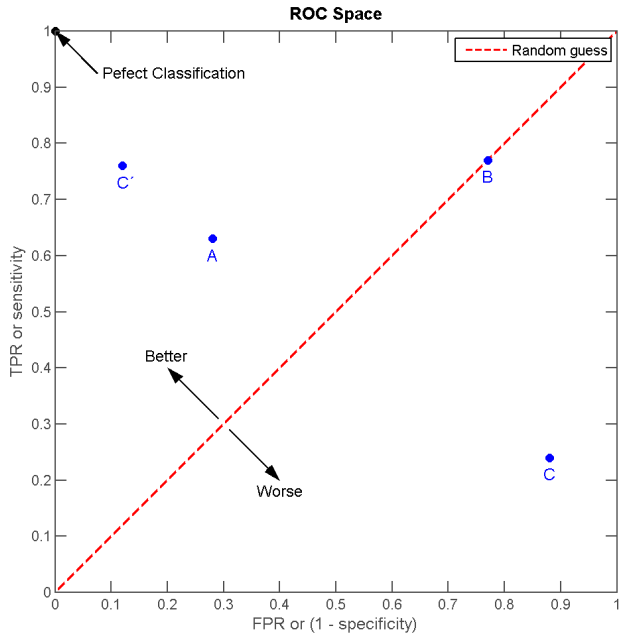
- Binary accuracy will lie to you, particularly for unbalanced datasets.
  - **Example**: Consider a dataset containing **90** positive samples and **10** negative samples.
  - If a classifier predicts all samples as positive, then the binary accuracy is 90%. But the TNR is 0, which shows that this classifier is not good.

Why do we need TNR and TPR?

- Binary accuracy will lie to you, particularly for unbalanced datasets.
  - **Example**: Consider a dataset containing **90** positive samples and **10** negative samples.
  - If a classifier predicts all samples as positive, then the binary accuracy is 90%. But the TNR is 0, which shows that this classifier is not good.
- Application: In medical tests, TPR is important. For example, a covid-positive person should be detected accurately such that he can receive instant treatment.

ROC

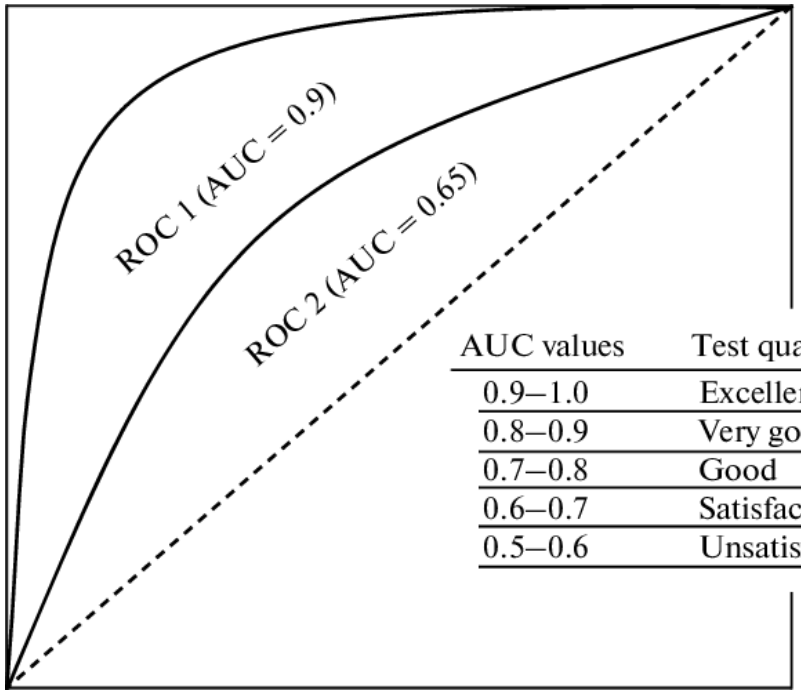
- Receiver operating characteristic curve (ROC)



- A ROC space is defined by FPR and TPR as x and y axes, respectively, which depicts relative trade-offs between **true positive** (benefits) and **false positive** (costs).
- The (0,1) point is also called a perfect classification

Area Under Curve (AUC)

- **Definition**: the area under the ROC curve.



AUC values	Test quality
0.9–1.0	Excellent
0.8–0.9	Very good
0.7–0.8	Good
0.6–0.7	Satisfactory
0.5–0.6	Unsatisfactory

K-nearest neighbors classification

Let $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ be the training dataset. Then K-nearest neighbor classification is as follows.

1. Specify an integer value for K (K is an odd value)
2. For $\boldsymbol{X} = \boldsymbol{x}_0$, find the K points in the training dataset that are closest to $\boldsymbol{x}_0$. Denote this subset of points as $\mathcal{N}_0$.
3. Then the probability $\mathbb{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x}_0)$is estimated as

$$\widehat{\mathbb{P}}(Y = 1|\boldsymbol{X} = \boldsymbol{x}_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = 1)$$

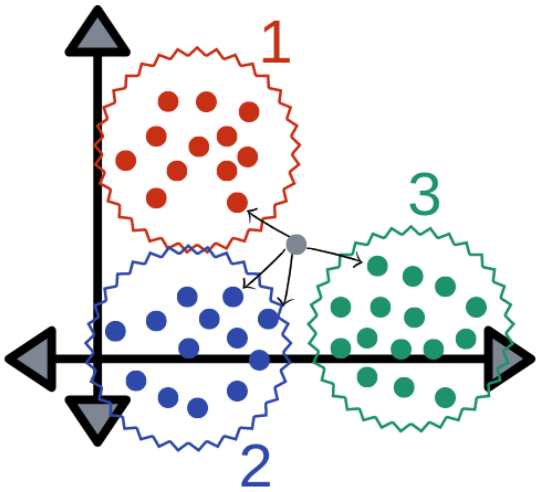
4. We then classify as follows:

$$\widehat{f}(\boldsymbol{x}) = \begin{cases} 1 \text{ if } \widehat{\mathbb{P}}(Y = 1|\boldsymbol{X} = \boldsymbol{x}_0) > 0.5 \\ -1 \text{ if } \widehat{\mathbb{P}}(Y = 1|\boldsymbol{X} = \boldsymbol{x}_0) \leq 0.5 \end{cases}$$

KNN: assumption

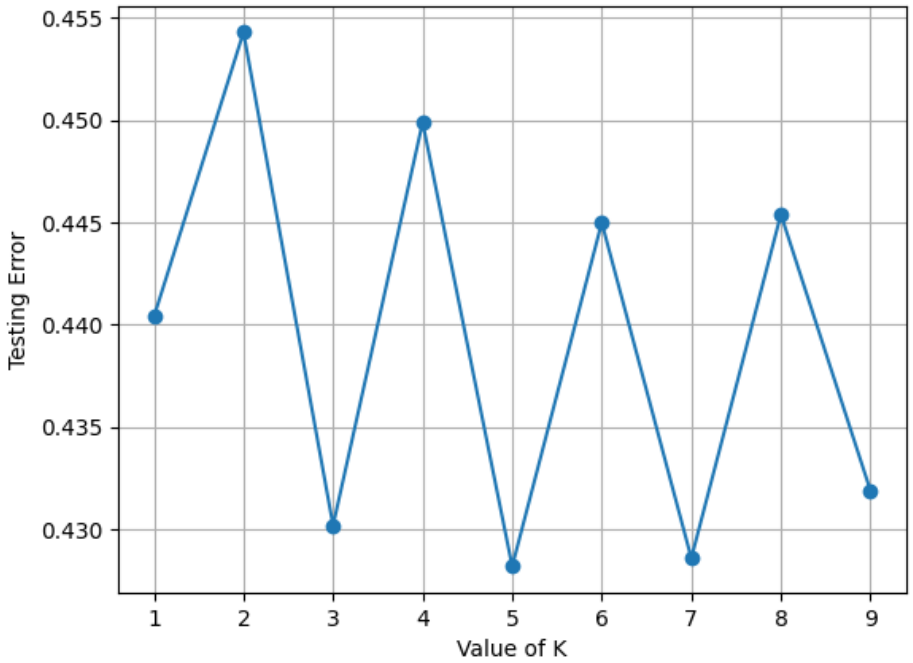
An implicit assumptions of KNN:

- Points closer in the feature space are more likely to have similar labels.



Insights about KNN

- **Question**: Why do we need to specify an odd value for K?
- **Answer**: To avoid the tie between two classes. For example, if we let K to be an even number, say $K = 4$. For a point $\boldsymbol{x}_0$, its 4 neighbors consists of 2 positive points and 2 negative points. The prediction will be randomly choosing a class.



The testing error of KNN will fluctuate as K increases.

Insights about KNN: determine K

- **Question**: How to decide the value of K?
- **Answer**: Usually, the choice of K depends on the size of training dataset, $\sqrt{n}$ is not-bad in practice.

Insights about KNN: determine K

- **Question**: How to decide the value of K?
- **Answer**: Usually, the choice of K depends on the size of training dataset, $\sqrt{n}$ is not-bad in practice.

- **Question**: **K** usually affects the flexibility of *KNN*. What is the relationship between K and the flexibility of KNN?
- **Answer**: As K increases, the model becomes less flexible. We can consider the example that if the number of neighbors is set as the size of training dataset, then the prediction is fixed for any testing sample.

Simulation: best K in practice

- Generate $\boldsymbol{x} = (x_1, x_2)$ from uniform distribution [-1,1].
- Set the probability as

$$\mathbb{P}(Y = 1) = \frac{1}{1 + \exp(-x_1 - x_2)}$$

- Generate 300 samples as training dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{300}$ as above
- Generate 10,000 samples as testing dataset as above.

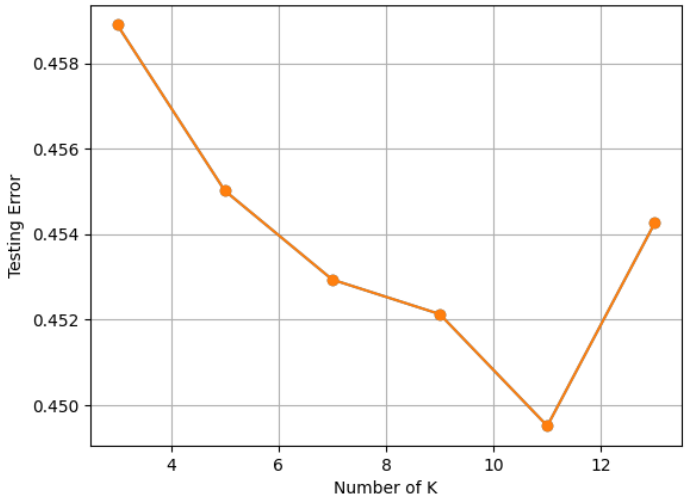


Figure: The best K in this case is 11.

Application: Banknote Dataset

- Dataset: consists of 200 genuine and counterfeit banknotes

	Status	Length	Left	Right	Bottom	Top	Diagonal
159	counterfeit	214.8	130.4	130.6	12.5	10.0	139.3
179	counterfeit	214.8	130.5	130.3	10.2	12.1	139.1
14	genuine	214.7	129.7	129.7	7.7	10.9	141.7
195	counterfeit	214.9	130.3	130.5	11.6	10.6	139.8
170	counterfeit	214.9	130.0	129.9	11.4	10.9	139.7
50	genuine	214.5	129.0	129.6	7.8	9.8	142.0
118	counterfeit	214.7	130.4	130.1	12.1	10.4	139.9
43	genuine	214.9	129.6	129.4	9.3	9.0	141.7
198	counterfeit	214.8	130.3	130.4	10.6	11.1	140.0
194	counterfeit	215.0	130.5	130.3	9.6	11.0	138.5
153	counterfeit	214.6	129.7	129.3	10.4	11.0	139.3

- Status: represent whether the banknote is true or false
- Length: Length of bill (mm)
- Left: Width of left edge (mm)
- Right: Width of right edge (mm)
- Bottom: Bottom margin width (mm)
- Top: Top margin width (mm)
- Diagonal: Length of diagonal (mm)

KNN function in Rstudio

- library(class)
- knn(train, test, cl, k = 1, l = 0)
  - **train**: dataframe of training features in shape $n \times p$, where n is the number of training samples and p is the number of dimensions
  - **test**: dataframe of testing features in shape $m \times p$, where m is the number of testing samples.
  - **cl**: an array of length n
  - k: is the number of neighbors you choose.
- the output of knn is an array of **predicted labels** of testing features.
- table(array 1,array 2): confusion matrix

Application in Rstudio

```
1  library(mclust) # import the package mclust
2  data(banknote) # import the banknote data
3  library(class) # import the package class
4  index <- 1:dim(banknote)[1] # Index number
5  index.train <- sample(index, 130, replace = F) # 130 samples used for
   training
6  bn.train <- banknote[index.train,-c(1,7)] # only use predictors 2-6 as
   predictors
7  bn.test <- banknote[-index.train, -c(1,7)]
8  Label.train <- banknote[index.train,]$Status
9  Label.test <- banknote[-index.train,]$Status
10 m.knn <- knn(bn.train, bn.test, Label.train, k = 1) # Output the predicted
   labels of testing samples
11 table.knn <- table(m.knn, Label.test)
```

Banknote: Result

Run the code: table.knn

```
> table.knn
                Label.test
m.knn            counterfeit genuine
  counterfeit             34       1
  genuine                  3      32
```

- The accuracy is (34+32)/70 $\approx$ 94.2%
- True Positive Rate = ?
- False Negative Rate = ?
- True Negative = ?
- False Positive Rate = ?

Banknote: Result

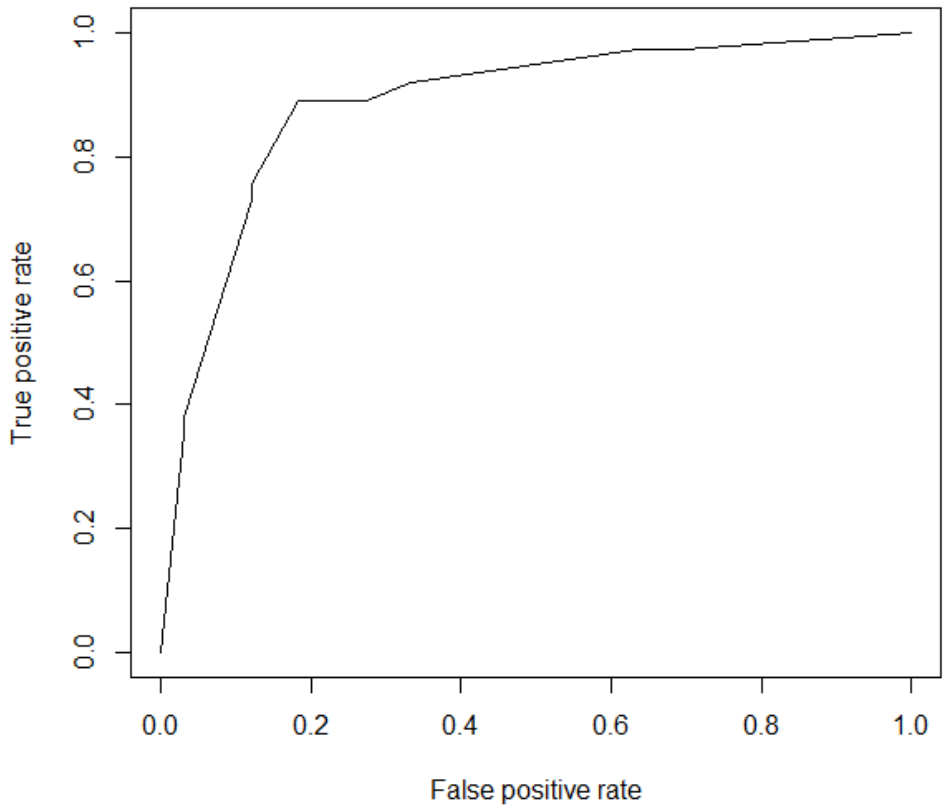
Run the code: table.knn

```
> table.knn
                Label.test
m.knn            counterfeit genuine
  counterfeit             34       1
  genuine                  3      32
```

- The accuracy is (34+32)/70 $\approx$ 94.2%
- True Positive Rate = 32/33 $\approx$ 96.9%
- False Negative Rate = 1/33 $\approx$ 3.1%
- True Negative Rate = 34/37 $\approx$ 91.9%
- False Positive Rate = 3/37 $\approx$ 8.1%

Banknote: ROC curve

- knnPredict <- predict(knnFit, newdata = bn.test, type='prob')
- pred <- prediction(knnPredict$counterfeit, bn.test$Status == 'counterfeit')
- perf <- performance(pred,"tpr","fpr")
- plot(perf)



What you should know

- The general procedure of KNN.
- KNN is a non-parametric model.
- Classification metrics and their motivation
- The corresponding implementations in R or Python.

Assignment 1 - Part 2

- Reproduce the result in Page 18 of KNN (testing performance decreases and then increases.) (Codes and results should be included in the PDF)