Fundamentals of Machine Learning

Yiqiao Yin

Back to top.

Content

Chapter 1 - Introduction

Chapter 2 - Basics in Statistical Learning

Chapter 3 - Linear Regression

Chapter 4 - Classification

Chapter 5 - Sampling and Bootstrap

Chapter 6 - Model Selection & Regularization

Chapter 7 - Going Beyond Linearity

Chapter 8 - Tree-based Method

Chapter 9 - Support Vector Machine

Chapter 10 - Deep Learning

Chapter 11 - Unsupervised Learning

Chapter 12 - Classification Metrics

Back to top.

Chapter 1 - Introduction

Go back to main content, click here

Back to top.

https://www.youtube.com/YiqiaoYin

A brief history:

- Beginning of the 19th century: least square was developed, this is the earliest form of statistical modeling and predictive analysis of what we now call linear regression.

A brief history:

- Beginning of the 19th century: least square was developed, this is the earliest form of statistical modeling and predictive analysis of what we now call linear regression.
- In 1936, linear discriminant analysis was proposed.

Back to top.

A brief history:

- Beginning of the 19th century: least square was developed, this is the earliest form of statistical modeling and predictive analysis of what we now call linear regression.
- In 1936, linear discriminant analysis was proposed.
- In 1940s, logistic regression was implemented.

Back to top.

A brief history:

- Beginning of the 19th century: least square was developed, this is the earliest form of statistical modeling and predictive analysis of what we now call linear regression.
- In 1936, linear discriminant analysis was proposed.
- In 1940s, logistic regression was implemented.
- In the early 1970s, generalized linear model is developed to generate an entire class of statistical learning models of which includes both linear regression and logistic regression models.

Back to top.

A brief history:

- Beginning of the 19th century: least square was developed, this is the earliest form of statistical modeling and predictive analysis of what we now call linear regression.
- In 1936, linear discriminant analysis was proposed.
- In 1940s, logistic regression was implemented.
- In the early 1970s, generalized linear model is developed to generate an entire class of statistical learning models of which includes both linear regression and logistic regression models.
- By 1980s, computing technology had finally improved sufficiently that non-linear methods were no longer computationally prohibitive.

Back to top.

A brief history:

- Beginning of the 19th century: least square was developed, this is the earliest form of statistical modeling and predictive analysis of what we now call linear regression.
- In 1936, linear discriminant analysis was proposed.
- In 1940s, logistic regression was implemented.
- In the early 1970s, generalized linear model is developed to generate an entire class of statistical learning models of which includes both linear regression and logistic regression models.
- By 1980s, computing technology had finally improved sufficiently that non-linear methods were no longer computationally prohibitive.
- In the mid 1980s, classification and regression trees were developed, followed shortly by linear additive models.

Back to top.

A brief history:

- Beginning of the 19th century: least square was developed, this is the earliest form of statistical modeling and predictive analysis of what we now call linear regression.
- In 1936, linear discriminant analysis was proposed.
- In 1940s, logistic regression was implemented.
- In the early 1970s, generalized linear model is developed to generate an entire class of statistical learning models of which includes both linear regression and logistic regression models.
- By 1980s, computing technology had finally improved sufficiently that non-linear methods were no longer computationally prohibitive.
- In the mid 1980s, classification and regression trees were developed, followed shortly by linear additive models.
- In the late 1980s, neural networks started to gain popularity.

Back to top.

The teaching materials has the following premises:

1. Many statistical learning models and machine learning methods are relevant and useful in a wide range of academic and non-academic disciplines. We believe that many of the contemporary learning methods will become widely available. Hence, we contracted our focuses in presenting the methods that we believe is the most widely accepted.

Back to top.

The teaching materials has the following premises:

1. Many statistical learning models and machine learning methods are relevant and useful in a wide range of academic and non-academic disciplines. We believe that many of the contemporary learning methods will become widely available. Hence, we contracted our focuses in presenting the methods that we believe is the most widely accepted.
2. Statistical learning models should be considered as “black-box” methods. We intend to diagnose the each and every building blocks in each statistical model we introduce in this teaching material.

Back to top.

The teaching materials has the following premises:

1. Many statistical learning models and machine learning methods are relevant and useful in a wide range of academic and non-academic disciplines. We believe that many of the contemporary learning methods will become widely available. Hence, we contracted our focuses in presenting the methods that we believe is the most widely accepted.
2. Statistical learning models should be considered as “black-box” methods. We intend to diagnose the each and every building blocks in each statistical model we introduce in this teaching material.
3. We believe it is a crime to only teach statistical machine learning to mathematicians. This should not be the case. We believe quality teaching can be carried and intuition of the technical work can be processed for the audience without the knowledge required mathematical derivation. That’s right. We will make each and every step clear to each and every one of our audience.

Back to top.

The teaching materials has the following premises:

1. Many statistical learning models and machine learning methods are relevant and useful in a wide range of academic and non-academic disciplines. We believe that many of the contemporary learning methods will become widely available. Hence, we contracted our focuses in presenting the methods that we believe is the most widely accepted.
2. Statistical learning models should be considered as “black-box” methods. We intend to diagnose the each and every building blocks in each statistical model we introduce in this teaching material.
3. We believe it is a crime to only teach statistical machine learning to mathematicians. This should not be the case. We believe quality teaching can be carried and intuition of the technical work can be processed for the audience without the knowledge required mathematical derivation. That’s right. We will make each and every step clear to each and every one of our audience without unnecessary technical jargons.
4. We assume our audience is also interested in learning the coding perspective of the statistical modeling. We provide basic tutorials in two languages: R and Python.

Back to top.

Who is the intended audience?

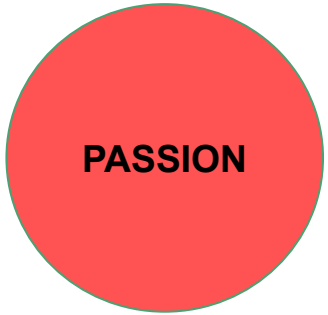
I would really recommend this course to you if you are in the demographic groups that are interested in learning the beginning building blocks and all the fundamentals of statistical machine learning.

This means our audience includes scientists, data scientists, data analysts, quants, and so on.

Some of these materials have major overlap with standard undergraduate or master level courses in machine learning or statistical machine learning. However, the entire materials is prepared by the writer of the course and we use our own teaching philosophy and methods to prepare these topics. We believe our approach can satisfy the premises listed before to the maximum amount of extent.

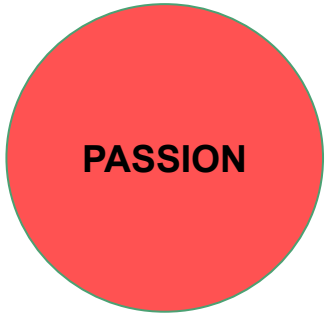
Back to top.

Teaching Philosophy: The Trinity Set



Back to top.

Teaching Philosophy: The Trinity Set



I presume everyone who attended this course is somewhat passionate about the topics. That's great!

Back to top.

Teaching Philosophy: The Trinity Set

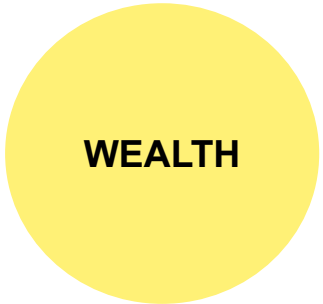
Back to top.

Teaching Philosophy: The Trinity Set

With time and effort, you will be well trained in the ways of statistical learning. You'll be able to tell which news is fake, what's wrong with your Tesla, and why can Alexa tell what you want. You can be taught and you will be talented!

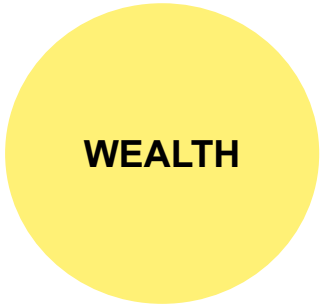
Back to top.

Teaching Philosophy: The Trinity Set



Back to top.

Teaching Philosophy: The Trinity Set

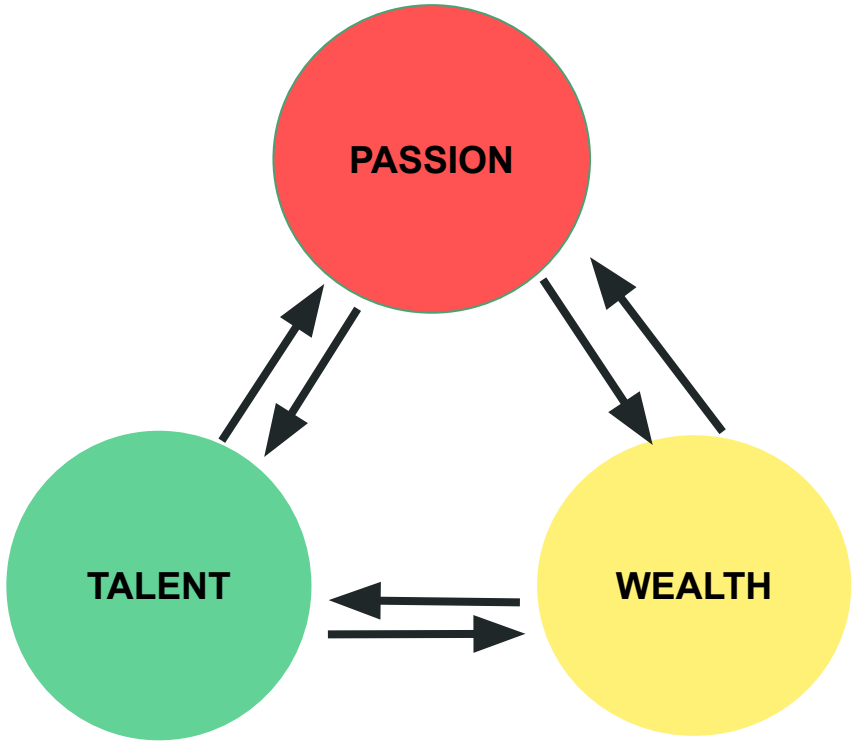


We also have a coaching session to guide you to land your dream job in data science. That will bring your wealth. Data Scientists get $120k starting salary and in some cities the starting salaries go from $150k. These are not small numbers!

Back to top.

https://www.youtube.com/YiqiaoYin

Teaching Philosophy: The Trinity Set



I presume everyone who attended this course is somewhat passionate about the topics. That's great!

With time and effort, you will be well trained in the ways of statistical learning. You'll be able to tell which news is fake, what's wrong with your Tesla, and why can Alexa tell what you want. You can be taught and you will be talented!

We also have a coaching session to guide you to land your dream job in data science. That will bring your wealth. Data Scientists get $120k starting salary and in some cities the starting salaries go from $150k. These are not small numbers!

Back to top.

Chapter 2 - Basics in Statistical Learning

Go back to main content, click here

Back to top.

https://www.youtube.com/YiqiaoYin

Some basic notations used in this course:

x_{ij}

This represents the value of the *j*th variable for the *i*th observation where *i* can take values 1, 2, 3, ..., n and *j* can take values 1, 2, ..., p.

Back to top.

We let **X** denote an n by p matrix whose (*i*, *j*)th element is x_{ij}

This matrix is

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}.$$

Back to top.

We can also write the data in vector form, which is another form of stating matrix.

For example, $\mathbf{X}_j$ is the vector matrix of the *j*th column in the data. Since the data has n observations, this vector matrix has running index 1j, 2j, ..., nj.

$$\mathbf{x}_j = \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{nj} \end{pmatrix}.$$

Back to top.

Denote the explanatory data matrix to be X;
Denote the response or target to be Y.

We can describe the relationship between X and Y by using the following formula.

$$Y = f(X) + \epsilon.$$

Back to top.

Denote the explanatory data matrix to be X;
Denote the response or target to be Y.

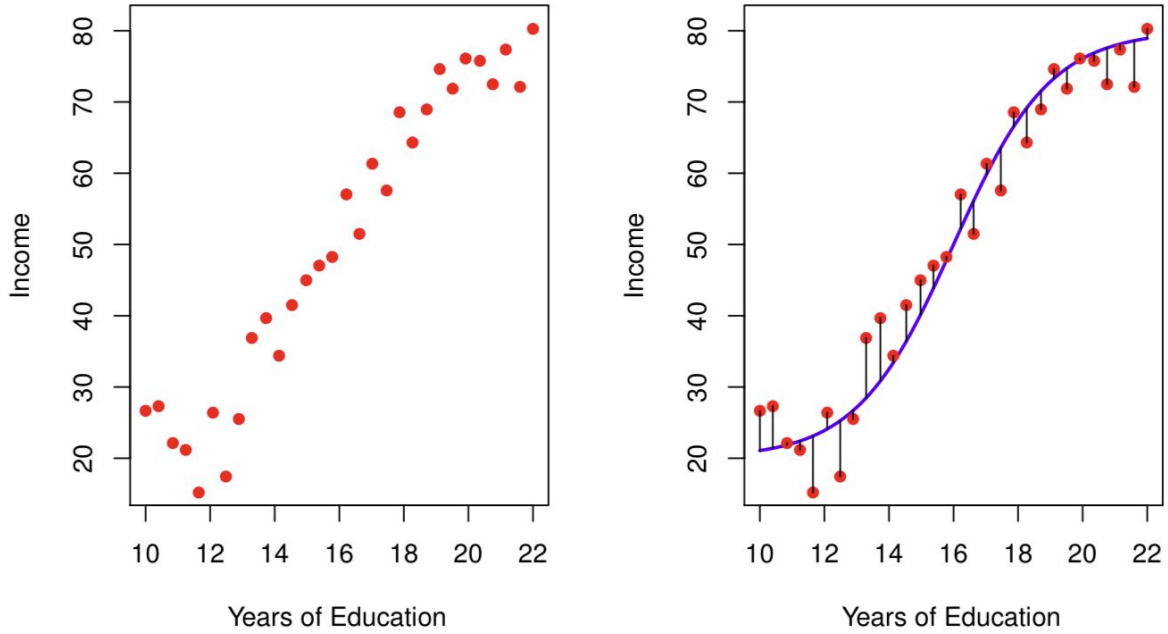
We can describe the relationship between X and Y by using the following formula.

$$Y = f(X) + \epsilon.$$

The formula says that the target “Y” is a function of “X” and we can use “f()” to denote the function or the relationship that we are interested in describing. The “epsilon” term added in the end of the formula is the error.

Back to top.

For example, we can try to predict income using years of education. In the following graphical representation, the years of education is the horizontal axis while the income is the vertical axis. The blue curve represents the underlying relationship between years of education and income. The black lines represent the error. Overall these errors sum to zero.



Back to top.

We want to make estimate the underlying relationship between X and Y, because of two major reasons: **prediction** and **inference**.

In practice, we usually observe X. However, Y cannot be easily observed or sometimes even impossible to obtain. In this case, we can try to predict Y by using an estimated f. We use the symbol “hat” denoted below:

Back to top.

We want to make estimate the underlying relationship between X and Y, because of two major reasons: **prediction** and **inference**.

In practice, we usually observe X. However, Y cannot be easily observed or sometimes even impossible to obtain. In this case, we can try to predict Y by using an estimated f. We use the symbol “hat” denoted below,

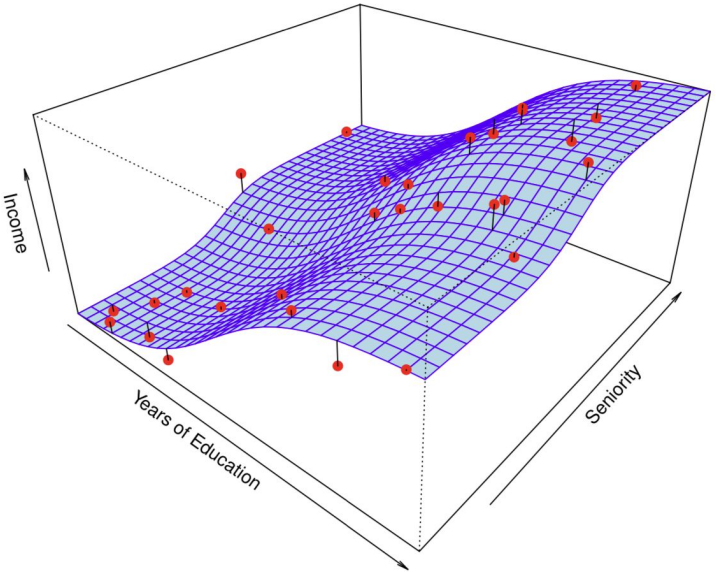
$$\hat{Y} = \hat{f}(X),$$

The above equation is read “y hat equals f hat with the input of x”.

Back to top.

For example, the income-education example we saw above can be extended in the following way. We can include an additional variable called “seniority”. Perhaps this new variable “seniority” can also help us out.

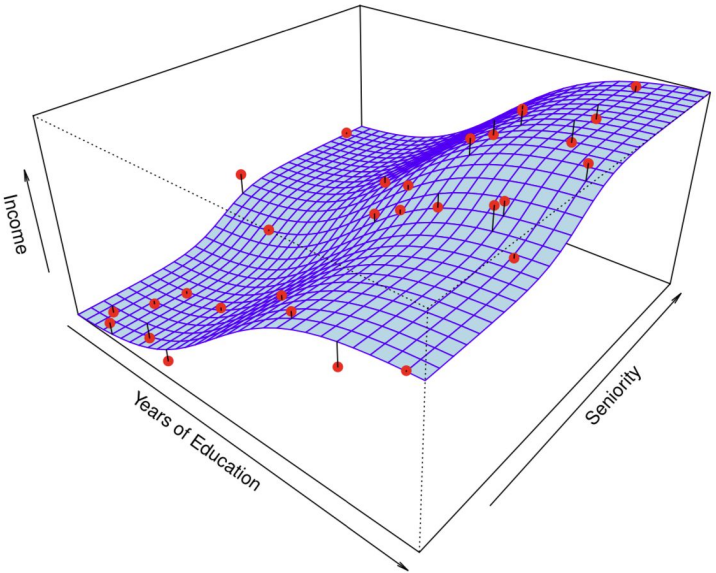
Putting everything together, we have ourselves a 3D graphical representation of income on the vertical axis while the years of income and seniority form the two horizontal axis that are independent to each other, hence form a horizontal plane.



Back to top.

For example, the income-education example we saw above can be extended in the following way. We can include an additional variable called “seniority”. Perhaps this new variable “seniority” can also help us out.

Putting everything together, we have ourselves a 3D graphical representation of income on the vertical axis while the years of income and seniority form the two horizontal axis that are independent to each other, hence form a horizontal plane.



In this case, we can think of the blue 3D plane to represent the true relationship while the red dots indicate the observed value.

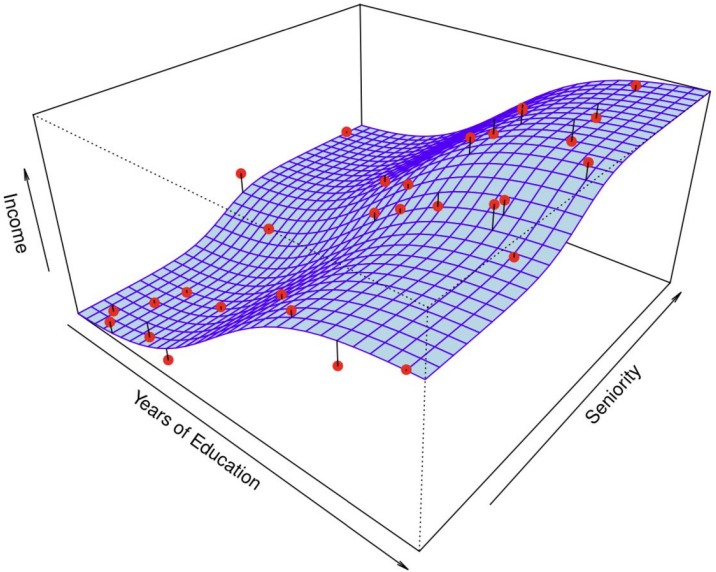
Back to top.

For example, the income-education example we saw above can be extended in the following way. We can include an additional variable called “seniority”. Perhaps this new variable “seniority” can also help us out.

Putting everything together, we have ourselves a 3D graphical representation of income on the vertical axis while the years of income and seniority form the two horizontal axis that are independent to each other, hence form a horizontal plane.

The goal for us can be to try to estimate the blue plane, hence we are trying to guess the real relationship. This guess or estimate is what we discussed:

$$\hat{Y} = \hat{f}(X)$$



In this case, we can think of the blue 3D plane to represent the true relationship while the red dots indicate the observed value.

Back to top.

We have introduced:

$\hat{f}$ This is read as “f hat”, and it means the estimate of the relationship f

$\hat{f}(X)$ This is read as “f hat of X”, and it means the estimate of the relationship f with the input X

$\hat{Y} = \hat{f}(X)$ The output of “f hat of X” is read as “Y hat”, which is the estimate of the target Y.

$\mathbb{E}()$ This symbol means the expectation of something. In practice, you can think of as the mean or average of something.

Back to top.

We have introduced:

$\hat{f}$ This is read as “f hat”, and it means the estimate of the relationship f

$\hat{f}(X)$ This is read as “f hat of X”, and it means the estimate of the relationship f with the input X

$\hat{Y} = \hat{f}(X)$ The output of “f hat of X” is read as “Y hat”, which is the estimate of the target Y.

$\mathbb{E}()$ This symbol means the expectation of something. In practice, you can think of as the mean or average of something.

We can derive

$$\begin{aligned}\mathbb{E}(Y - \hat{Y}) &= \mathbb{E}(f(X) - (\hat{f}(X) + \epsilon))^2 \\ &= (f(X) - f(\hat{X}))^2 + \mathrm{var}(\epsilon)\end{aligned}$$

Back to top.

We have introduced:

$\hat{f}$ This is read as “f hat”, and it means the estimate of the relationship f

$\hat{f}(X)$ This is read as “f hat of X”, and it means the estimate of the relationship f with the input X

$\hat{Y} = \hat{f}(X)$ The output of “f hat of X” is read as “Y hat”, which is the estimate of the target Y.

$\mathbb{E}()$ This symbol means the expectation of something. In practice, you can think of as the mean or average of something.

We can derive

$$\begin{aligned}\mathbb{E}(Y - \hat{Y}) &= \mathbb{E}(f(X) - (\hat{f}(X) + \epsilon))^2 \\ &= \underbrace{(f(X) - f(\hat{X}))^2}_{\text{Reducible}} + \underbrace{\text{var}(\epsilon)}_{\text{Irreducible}}\end{aligned}$$

Back to top.

We have introduced:

$\hat{f}$ This is read as “f hat”, and it means the estimate of the relationship f

$\hat{f}(X)$ This is read as “f hat of X”, and it means the estimate of the relationship f with the input X

$\hat{Y} = \hat{f}(X)$ The output of “f hat of X” is read as “Y hat”, which is the estimate of the target Y.

$\mathbb{E}()$ This symbol means the expectation of something. In practice, you can think of as the mean or average of something.

We can derive

$$\begin{aligned}\mathbb{E}(Y - \hat{Y}) &= \mathbb{E}(f(X) - (\hat{f}(X) + \epsilon))^2 \\ &= \underbrace{(f(X) - \hat{f}(X))^2}_{\text{Reducible}} + \underbrace{\text{var}(\epsilon)}_{\text{Irreducible}}\end{aligned}$$

The focus of this course is to give you a whole library of tools to estimate the real model with the aim of minimizing reducible error.

Back to top.

Inference:

We are also interested about making inference our statistical model. This means we are interested in answering the following questions:

1. Which predictors are associated with the response?

Back to top.

Inference:

We are also interested about making inference our statistical model. This means we are interested in answering the following questions:

1. Which predictors are associated with the response? Often times it is only a small number of predictors (aka explanatory variables) that has a large impact on target variable.

Back to top.

Inference:

We are also interested about making inference our statistical model. This means we are interested in answering the following questions:

1. Which predictors are associated with the response? Often times it is only a small number of predictors (aka explanatory variables) that has a large impact on target variable.
2. What is the relationship between response variable and each predictor? Some of predictors may have positive impact to the response variable. Some may have negative impact. Do not confuse this with significance. There can be a predictor that has a large impact in terms of magnitude, but it is not always important.

Back to top.

Inference:

We are also interested about making inference our statistical model. This means we are interested in answering the following questions:

1. Which predictors are associated with the response? Often times it is only a small number of predictors (aka explanatory variables) that has a large impact on target variable.
2. What is the relationship between response variable and each predictor? Some of predictors may have positive impact to the response variable. Some may have negative impact. Do not confuse this with significance. There can be a predictor that has a large impact in terms of magnitude, but it is not always important.
3. Can the relationship between Y and each predictor be adequately summarized using linear relationships?

Back to top.

Inference:

We are also interested about making inference our statistical model. This means we are interested in answering the following questions:

1. Which predictors are associated with the response? Often times it is only a small number of predictors (aka explanatory variables) that has a large impact on target variable.
2. What is the relationship between response variable and each predictor? Some of predictors may have positive impact to the response variable. Some may have negative impact. Do not confuse this with significance. There can be a predictor that has a large impact in terms of magnitude, but it is not always important.
3. Can the relationship between Y and each predictor be adequately summarized using linear relationships? It is sometimes not enough to use. It is true that linear model is easy to build and interpret. However, in reality, the real relationship is often times more complicated than linear relationship.

Back to top.

Now the question is how to search for the linear relationship, that is, if we assume the real relationship is linear.

Back to top.

Now the question is how to search for the linear relationship, that is, if we assume the real relationship is linear.

Remark: The assumption of linear relationship is imposed by us (or any end-users design this machine learning project). There can be human bias in the design. It is often important to address this before the model is built.

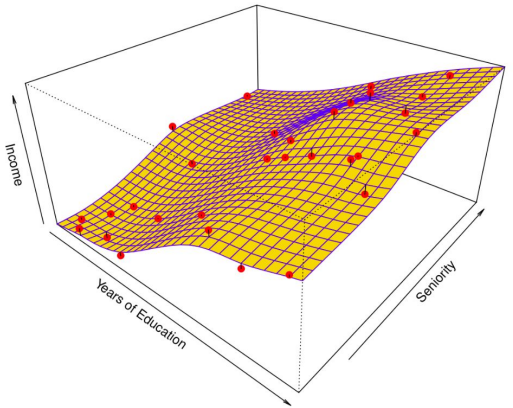
Back to top.

Now the question is how to search for the linear relationship, that is, if we assume the real relationship is linear.

Remark: The assumption of linear relationship is imposed by us (or any end-users design this machine learning project). There can be human bias in the design. It is often important to address this before the model is built.

We can assume that the income is in a linear relationship with education and seniority. This makes sense because it is a 3D plane.

$$\texttt{income} \approx \beta_0 + \beta_1 \times \texttt{education} + \beta_2 \times \texttt{seniority}.$$



Back to top.

There is a tradeoff between model flexibility and interpretability.

Back to top.

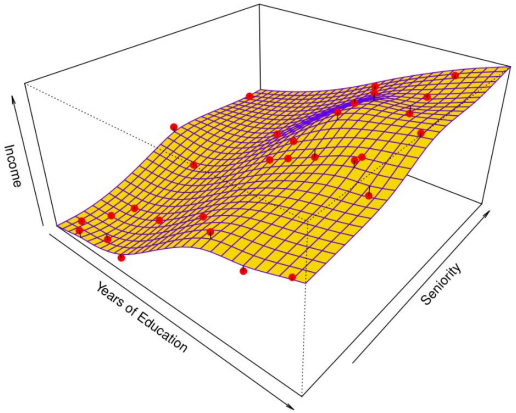
There is a tradeoff between model flexibility and interpretability.

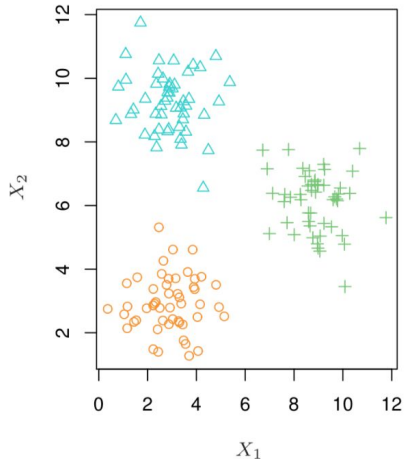
Back to top.

Parametric vs. Nonparametric.

Back to top.

Parametric vs. Nonparametric.

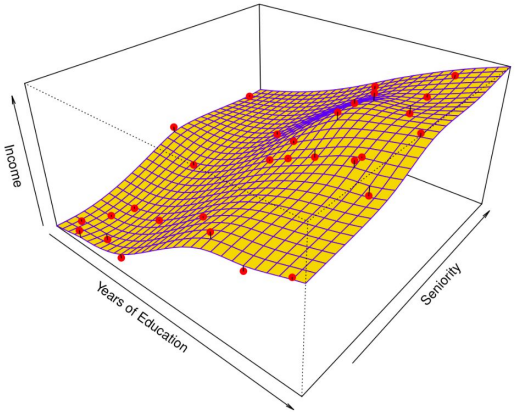


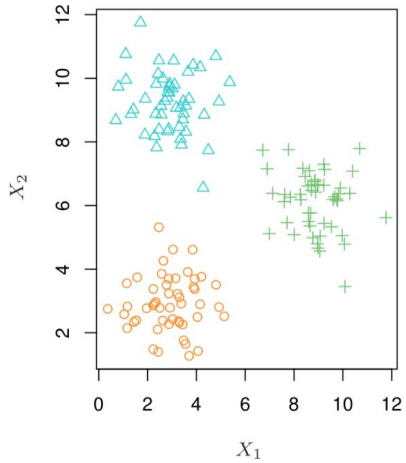


Left: We use a 3D plane. This means we use a model. In this case, it is a linear model.

Back to top.

Parametric vs. Nonparametric.





Left: We use a 3D plane. This means we use a model. In this case, it is a linear model. This is supervised learning.

This data is pretty well separated using the two axis, X1 and X2. We do not need any model to separate them into three different classes. This is unsupervised learning.

Back to top.

In the regression setting, the most commonly-used measure is the mean squared error (MSE), given by

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

Back to top.

In the regression setting, the most commonly-used measure is the mean squared error (MSE), given by

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

The capital sigma symbol means “summation”. The symbol has running index “i” starting from 1 to n. This is because we assume the samples in training data has n samples. The content inside the square parenthesis is the difference between target and estimate of real relationship with the input to be x, which is the mistake or error we make at *i*th observation. We take the square of each of them and we add all of them together and then divided by sample size n.

Back to top.

In the regression setting, the most commonly-used measure is the mean squared error (MSE), given by

Intuitively, the MSE states the average squared distance between target and estimate.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

The capital sigma symbol means “summation”. The symbol has running index “i” starting from 1 to n. This is because we assume the samples in training data has n samples. The content inside the square parenthesis is the difference between target and estimate of real relationship with the input to be x, which is the mistake or error we make at *i*th observation. We take the square of each of them and we add all of them together and then divided by sample size n.

Back to top.

The Mean Squared Error (MSE) defined above can be broken into the following:

$$\mathbb{E}(y_0 - \hat{f}(x_0))^2 = \mathrm{Var}(\hat{f}(x_0)) + (\mathrm{Bias}(\hat{f}(x_0)))^2 + \mathrm{Var}(\epsilon)$$

Back to top.

The Mean Squared Error (MSE) defined above can be broken into the following:

$$\mathbb{E}(y_0 - \hat{f}(x_0))^2 = \mathrm{Var}(\hat{f}(x_0)) + (\mathrm{Bias}(\hat{f}(x_0)))^2 + \mathrm{Var}(\epsilon)$$

This is the famous “bias-variance” tradeoff. This equation means that the Mean Squared Error (MSE) can be broken apart into variance of the estimator and the bias of the estimator for any given x_0 (pronounced as “x null” which is simply an instance of given value of x)

Back to top.

The Mean Squared Error (MSE) defined above can be broken into the following:

$$\mathbb{E}(y_0 - \hat{f}(x_0))^2 = \mathrm{Var}(\hat{f}(x_0)) + (\mathrm{Bias}(\hat{f}(x_0)))^2 + \mathrm{Var}(\epsilon)$$

This is the famous “bias-variance” tradeoff. This equation means that the Mean Squared Error (MSE) can be broken apart into **variance** of the estimator and the **bias** of the estimator for any given x_0 (pronounced as “x null” which is simply an instance of given value of x).

Since both variance and bias are non-negative, an important observation from the above equation is that the Mean Squared Error (MSE) can **never fall below** variance of epsilon, which the irreducible error of the model.

Back to top.

The previous slides discussed Mean Squared Error (MSE), which is a loss function for regression problems. What do we do when we are in a classification setting?

Back to top.

The previous slides discussed Mean Squared Error (MSE), which is a loss function for regression problems. What do we do when we are in a classification setting?

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{I}(y_i \neq \hat{y}_i)$$

Back to top.

The previous slides discussed Mean Squared Error (MSE), which is a loss function for regression problems. What do we do when we are in a classification setting?

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{I}(y_i \neq \hat{y}_i)$$

The boldface letter “I” means indicator function. The indicator function is “1” if the content inside the parenthesis satisfies and it is “0” if not. In this case, we want to record error rate, so we input target not equals estimate in the parenthesis of the indicator function.

Back to top.

The previous slides discussed Mean Squared Error (MSE), which is a loss function for regression problems. What do we do when we are in a classification setting?

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{I}(y_i \neq \hat{y}_i)$$

The boldface letter "I" means indicator function. The indicator function is "1" if the content inside the parenthesis satisfies and it is "0" if not. In this case, we want to record error rate, so we input target not equals estimate in the parenthesis of the indicator function.

Overall, the equation says we record "1" if the target does not equal to our estimate. Then we sum all the "1"s together and then divided by the total sample size n.

Back to top.

The previous slides discussed Mean Squared Error (MSE), which is a loss function for regression problems. What do we do when we are in a classification setting?

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{I}(y_i \neq \hat{y}_i)$$

The boldface letter “I” means indicator function. The indicator function is “1” if the content inside the parenthesis satisfies and it is “0” if not. In this case, we want to record error rate, so we input target not equals estimate in the parenthesis of the indicator function.

Overall, the equation says we record “1” if the target does not equal to our estimate. Then we sum all the “1”s together and then divided by the total sample size n.

For example, if there are 3 mistakes and the sample size n is 10, then we add 1 + 1 + 1 = 3 and then divide 3 by 10 to obtain 30%. This is the error rate in a classification setting.

Back to top.

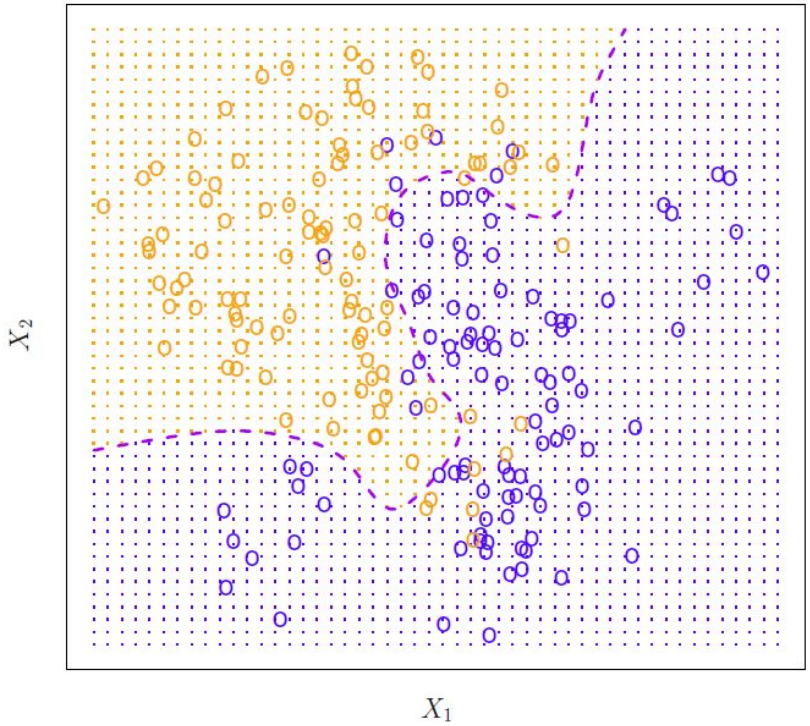
Bayes' Classifier

The Bayes' Classifier states the probability an observation has outcome in j conditional on the predictors X. We can formally write as the follows

$$\Pr(Y = j|X = x_0)$$

where j refers to certain class and the "|" means conditioning on something. In this case, the formula states the probability that the outcome Y is class j given or conditioning the predictors X.

Back to top.



Back to top.

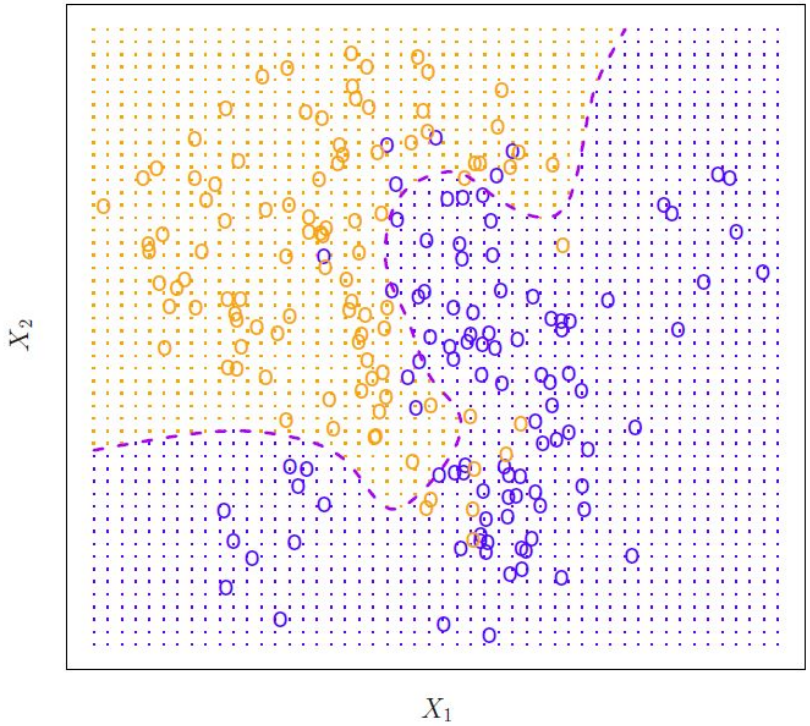


FIGURE 2.13. *A simulated data set consisting of* 100 *observations in each of two groups, indicated in blue and in orange. The purple dashed line represents the Bayes decision boundary. The orange background grid indicates the region in which a test observation will be assigned to the orange class, and the blue background grid indicates the region in which a test observation will be assigned to the blue class.*

Here we can say a few words about the Bayes error rate. Since the Bayes classifier will always choose the class for which the Bayes' classifier is the largest, the error rate will hence be

$$1 - E\left(\max_j \Pr(Y = j|X)\right),$$

since the Bayes' classifier aims to maximize the conditional probability the error rate will be the remaining value if we subtract the correct prediction rate from 1.

This builds up the fundamentals of the **K-Nearest Neighbors algorithm**.

Back to top.

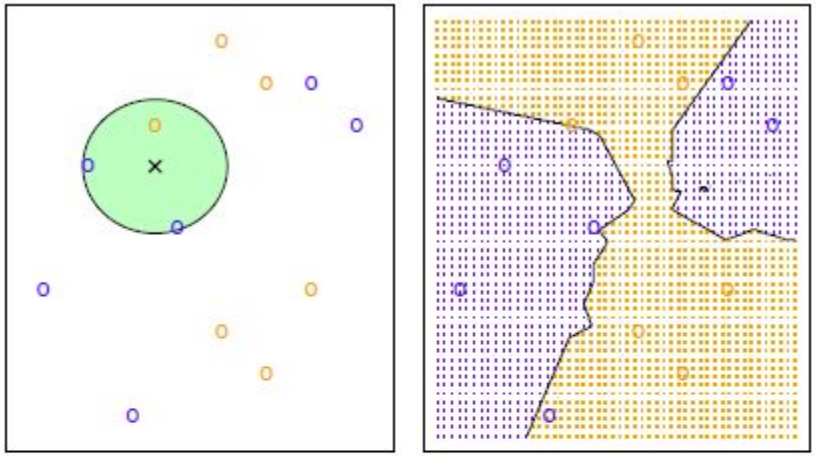
For real data, we never know the true conditional distribution of the outcome given the predictors. Hence, we never know the true Bayes' classifier. Hence, Bayes' classifier serves as the unattainable golden standard.

This is where K-nearest neighbors (KNN) can be a good candidate. Given a positive integer K and a test observation x0, the KNN classifier first identifies the K points in the training data that are closest to x0, represented by an entire neighbor, $\mathcal{N}_0$

It then estimates the conditional probability for class j as the fraction of points in the entire neighbor whose response values equal to j:

$$\Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j).$$

Back to top.



Back to top.

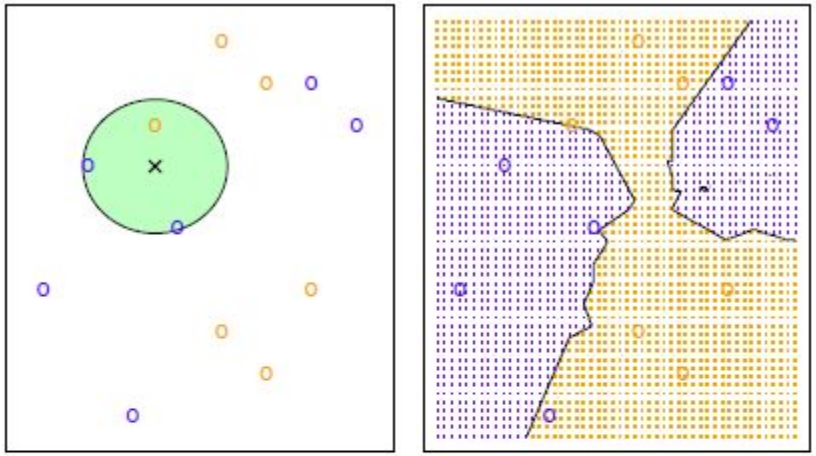


FIGURE 2.14. *The KNN approach, using $K = 3$, is illustrated in a simple situation with six blue observations and six orange observations.* Left: *a test observation at which a predicted class label is desired is shown as a black cross. The three closest points to the test observation are identified, 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.*

Back to top.

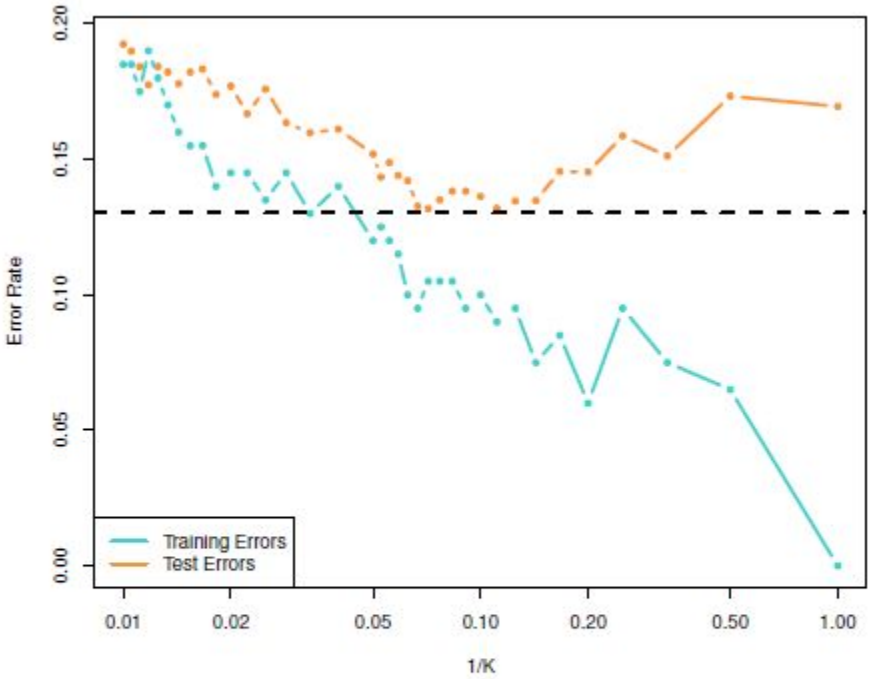


FIGURE 2.17. *The KNN training error rate (blue, 200 observations) and test error rate (orange, 5,000 observations) on the data from Figure 2.13, as the level of flexibility (assessed using $1/K$ on the log scale) increases, or equivalently as the number of neighbors K decreases. The black dashed line indicates the Bayes error rate. The jumpiness of the curves is due to the small size of the training data set.*

Back to top.

Chapter 3 - Linear Regression

Go back to main content, click here

Back to top.

https://www.youtube.com/YiqiaoYin

This chapter we introduce linear regression, a very fundamental approach for supervised learning. Linear regression is also a useful tool for predicting a quantitative response.

Recall the Advertising data. The outcome variable is sales (in thousands of units) can be a function of advertising budgets (in thousands of dollars) for TA, radio, and newspaper media.

Suppose that in our role as data scientist we are asked to suggest on the basis of this data, marketing plan for next year that will result in high product sales. What information would be useful in order to provide such a recommendation?

Questions:

- Is there a relationship between advertising budget and sales?
- How strong is the relationship between advertising budget and sales?
- Which media are associated with sales?
- How large is the association between each medium and sales?
- How accurately can we predict future sales?
- Is the relationship linear?
- Is there synergy among the advertising media?

Back to top.

Simple linear regression is a quantitative model that measures the most basic form of supervised learning: the model aims to predict response Y on the basis of a single predictor variable X. Further, we assume a linear relationship between X and Y.

Back to top.

Simple linear regression is a quantitative model that measures the most basic form of supervised learning: the model aims to predict response Y on the basis of a single predictor variable X. Further, we assume a linear relationship between X and Y.

Mathematically, we write

$$Y \approx \beta_0 + \beta_1 X$$

and it is read as Y is approximately modeled as beta-zero plus beta-one times X. Another way is saying this mathematical statement is “we regress Y on X”. This means that we are modeling the outcome variable Y by building a linear model on X.

Back to top.

Simple linear regression is a quantitative model that measures the most basic form of supervised learning: the model aims to predict response Y on the basis of a single predictor variable X. Further, we assume a linear relationship between X and Y.

Mathematically, we write

$$Y \approx \beta_0 + \beta_1 X$$

and it is read as Y is approximately modeled as beta-zero plus beta-one times X. Another way is saying this mathematical statement is “we regress Y on X”. This means that we are modeling the outcome variable Y by building a linear model on X.

For example, suppose in the Advertising data, we denote Y to be the sales and X to be the TV advertising. In this case, a linear model can be build as the following

$$\text{sales} \approx \beta_0 + \beta_1 \times \text{TV}$$

Back to top.

For example, suppose in the Advertising data, we denote Y to be the sales and X to be the TV advertising. In this case, a linear model can be build as the following

$$\text{sales} \approx \beta_0 + \beta_1 \times \text{TV}$$

The betas are known as the model coefficients or parameters. Given training data, we are able to fit the model to the training data and obtain estimated betas. Notice that these estimated betas are not exactly the same as the real ones. Hence, we use “hat” on their symbols to denote that they are the estimates. In other words, we write

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

The goal, then, would be to estimate the beta hats.

Back to top.

For example, suppose in the Advertising data, we denote Y to be the sales and X to be the TV advertising. In this case, a linear model can be build as the following

$$\text{sales} \approx \beta_0 + \beta_1 \times \text{TV}$$

The betas are known as the model coefficients or parameters. Given training data, we are able to fit the model to the training data and obtain estimated betas. Notice that these estimated betas are not exactly the same as the real ones. Hence, we use “hat” on their symbols to denote that they are the estimates. In other words, we write

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

The goal, then, would be to estimate the beta hats.

Before we fit the model, we need a way of telling us how many mistakes we have made.

Back to top.

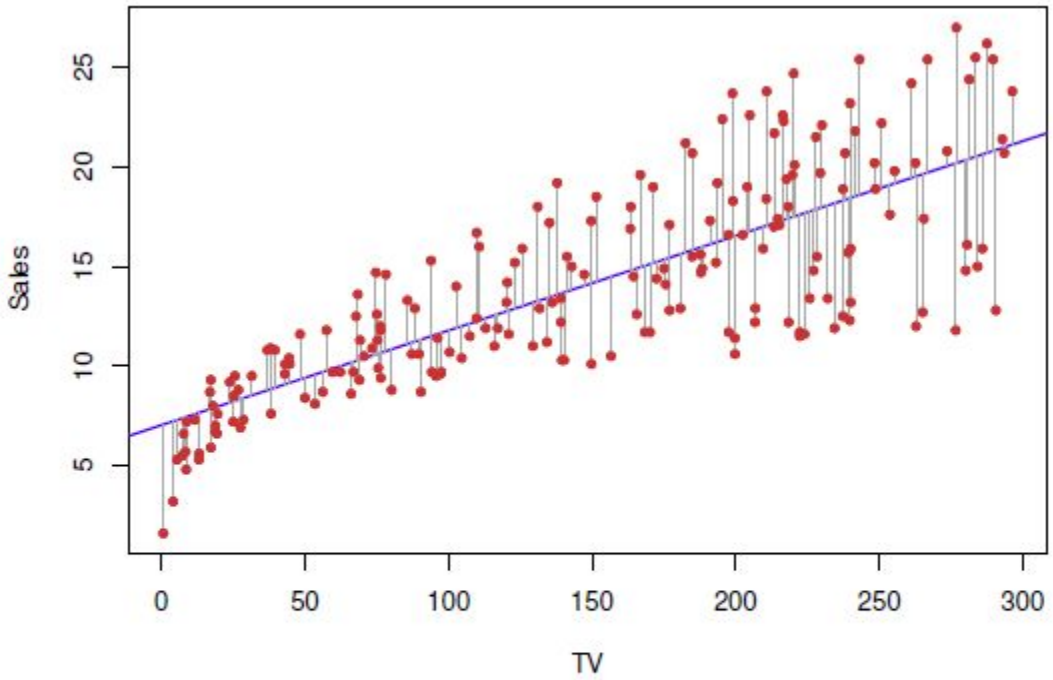


FIGURE 3.1. *For the* `Advertising` *data, the least squares fit for the regression of* `sales` *onto* `TV` *is shown. The fit is found by minimizing the residual sum of squares. Each grey line segment represents a residual. In this case a linear fit captures the essence of the relationship, although it overestimates the trend in the left of the plot.*

Back to top.

Let the linear model be the prediction for Y based on the ith value of X. Then we can compute “e” as the difference of y and y-hat. This “e” represents residual, which is the error or the mistake of our model.

$$\text{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + \cdots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$$

This residual sum of squares (RSS) written above is the error (or sometimes known as mistakes or loss) of the model. It is also RSS what we want to minimize. We do so by controlling the beta hats. Hence, we can write the objective function below

$$\min_{\beta_0, \beta_1} \text{RSS}$$

Back to top.

We can use least square to solve this problem. This least square approach chooses the beta hats (the estimated value of the linear coefficients) by minimizing the RSS. Afterwards, we can produce the mathematical equations

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$

Back to top.

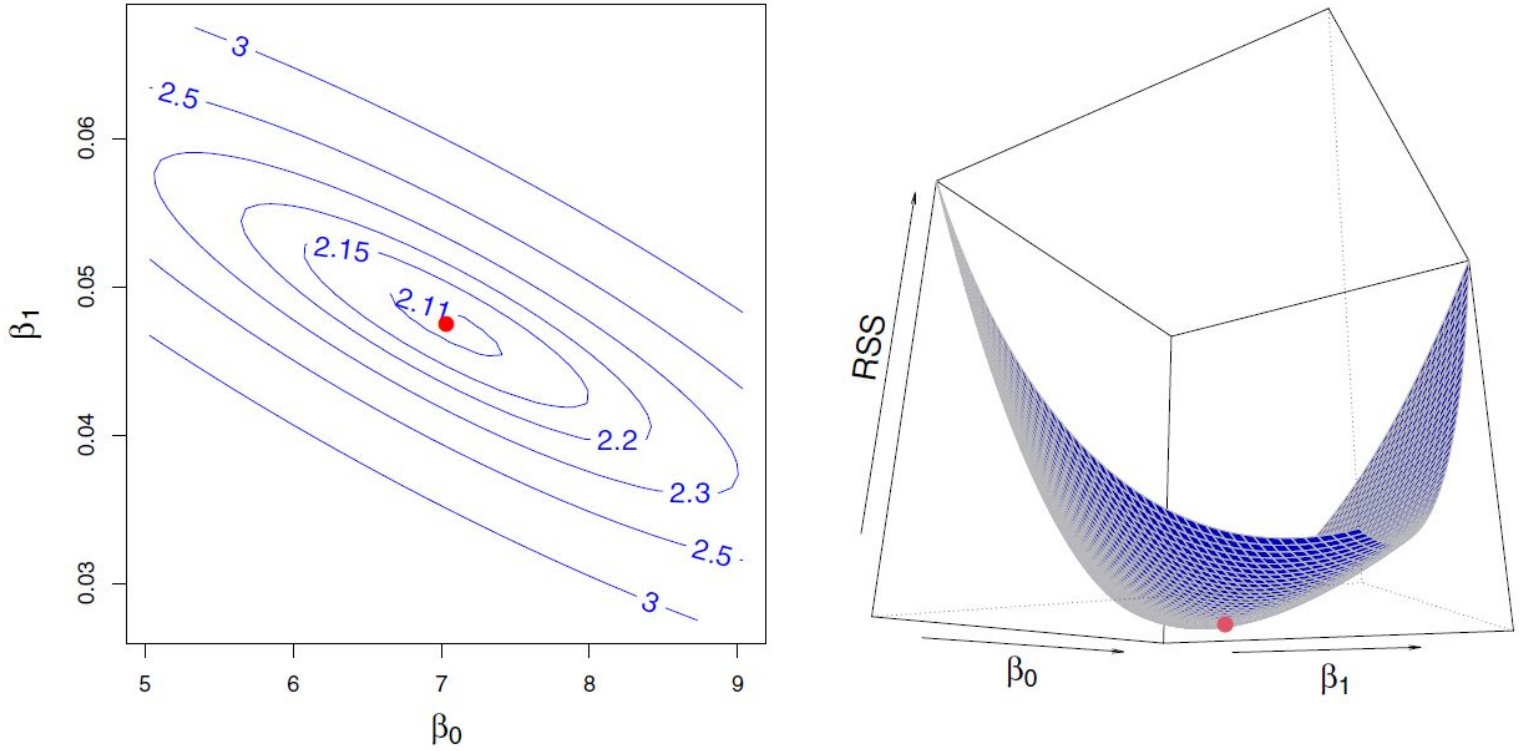


FIGURE 3.2. *Contour and three-dimensional plots of the RSS on the* Advertising *data, using* sales *as the response and* TV *as the predictor. The red dots correspond to the least squares estimates* $\hat{\beta}_0$ *and* $\hat{\beta}_1$*, given by (3.4).*

Back to top.

How accurate is this model?

Back to top.

How accurate is this model?

Well, first we need to define exactly what it is we want are interested in measuring. Since the outcome is continuous, the first fundamental intuition to straighten out is the average of the outcome Y for the population. However, we do not observe the entire population. Hence, we use the average of the sample Y as what we are interested in. In this case, the sample average of the outcome would be mu and the estimate would be mu-hat, which we can write in the following and derive its formulation,

$$\mathrm{Var}(\hat{\mu}) = \mathrm{SE}(\hat{\mu})^2 = \frac{\sigma^2}{n},$$

where sigma is the standard deviation of the outcome variable Y.

Back to top.

Sample average of the outcome would be mu and the estimate would be mu-hat, which we can write in the following and derive its formulation,

$$\mathrm{Var}(\hat{\mu}) = \mathrm{SE}(\hat{\mu})^2 = \frac{\sigma^2}{n},$$

where sigma is the standard deviation of the outcome variable Y.

We can do the same with the linear coefficients:

$$\mathrm{SE}(\hat{\beta}_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \right], \quad \mathrm{SE}(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

These formulas help us to essentially “understand” from statistical perspective how “accurate” the learned coefficients are.

What do we mean by this?

Sample average of the outcome would be mu and the estimate would be mu-hat, which we can write in the following and derive its formulation,

$$\mathrm{Var}(\hat{\mu}) = \mathrm{SE}(\hat{\mu})^2 = \frac{\sigma^2}{n},$$

where sigma is the standard deviation of the outcome variable Y.

We can do the same with the linear coefficients:

$$\mathrm{SE}(\hat{\beta}_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \right], \quad \mathrm{SE}(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

These formulas help us to essentially “understand” from statistical perspective how “accurate” the learned coefficients are.

What do we mean by this? Suppose we have a simple model y = 2 + 3x. We find the estimated beta-zero to be 1.99 while the standard error for beta-zero to be 0.1. We can “sort of” conclude this 1.99 is a “good guess”.

Back to top.

We can formally introduce the above intuition below using beta-one.

For the 95% confidence interval for beta-one, it approximately takes the form of the following range

$$\hat{\beta}_1 \pm 2 \cdot \mathrm{SE}(\hat{\beta}_1).$$

That is, there is approximately a 95% chance that the interval

$$\left[\hat{\beta}_1 - 2 \cdot \mathrm{SE}(\hat{\beta}_1),\ \hat{\beta}_1 + 2 \cdot \mathrm{SE}(\hat{\beta}_1)\right]$$

will contain the true value of beta-one. Similarly, a confidence interval for beta-one can be written in the following.

$$\hat{\beta}_0 \pm 2 \cdot \mathrm{SE}(\hat{\beta}_0).$$

The argument or debate takes a form of hypothesis testing. The null hypothesis can be: there is no relationship between X and Y. The alternative hypothesis can be: There is some relationship between X and Y. The linear coefficients and their standard errors help us establish quantitative and statistical reasoning of the model.

Mathematically, this corresponds to testing

$$H_0 : \beta_1 = 0$$

versus

$$H_a : \beta_1 \neq 0,$$

Back to top.

The argument or debate takes a form of hypothesis testing. The null hypothesis can be: there is no relationship between X and Y. The alternative hypothesis can be: There is some relationship between X and Y. The linear coefficients and their standard errors help us establish quantitative and statistical reasoning of the model.

Mathematically, this corresponds to testing

$$H_0 : \beta_1 = 0$$

versus

$$H_a : \beta_1 \neq 0,$$

Then we can compute the test statistic to be the ratio of the difference between the estimate of beta-one and null hypothesis over the standard error of the estimate of beta-one.

$$t = \frac{\hat{\beta}_1 - 0}{\mathrm{SE}(\hat{\beta}_1)},$$

The larger the value of test statistic the better. Here “better” means statistically significant.

Back to top.

	Coefficient	Std. error	t-statistic	p-value
Intercept	7.0325	0.4578	15.36	< 0.0001
TV	0.0475	0.0027	17.67	< 0.0001

TABLE 3.1. *For the* Advertising *data, coefficients of the least squares model for the regression of number of units sold on TV advertising budget. An increase of* \$1,000 *in the TV advertising budget is associated with an increase in sales by around 50 units. (Recall that the* sales *variable is in thousands of units, and the* TV *variable is in thousands of dollars.)*

Back to top.

We can expand our understanding from simple linear regression regression to **multiple linear regression** which takes the following form

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

The estimated outcome Y can be written as

$$\texttt{sales} = \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times \texttt{newspaper} + \epsilon.$$

or

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p.$$

The RSS can be written as

$$\begin{aligned} \text{RSS} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2. \end{aligned}$$

Back to top.

How to test this model?

We can conduct hypothesis testing. Instead of testing one variable, we can test all variables at once.

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0$$

$$H_a : \text{ at least one } \beta_j \text{ is non-zero.}$$

Then we can compute the F test statistic

$$F = \frac{(\text{TSS} - \text{RSS})/p}{\text{RSS}/(n-p-1)}$$

The same reasoning can be carried out: we generally want the F test statistic to be high value, which may lead us to conclude the the model is statistically significant.

Back to top.

Chapter 4 - Classification

Go back to main content, click here

Back to top.

The linear regression model discussed in previous chapter assumes that the response variable Y is quantitative or continuous. However, in many other situation, the response variable Y can be qualitative or categorical.

For example, the ratings of a movie can be 1 star, 2 stars, 3 stars, 4 stars, or 5 stars. In this case, given the information of a movie, a model needs to be able to produce a guess of a particular class and the 5 stars rating does not necessarily mean five times the likeliness of 1 star. In this case, the ratings of movies can be considered categorical (or discrete). This will be an example we prefer other models than linear regression model.

Back to top.

The linear regression model discussed in previous chapter assumes that the response variable Y is quantitative or continuous. However, in many other situation, the response variable Y can be qualitative or categorical.

For example, the ratings of a movie can be 1 star, 2 stars, 3 stars, 4 stars, or 5 stars. In this case, given the information of a movie, a model needs to be able to produce a guess of a particular class and the 5 stars rating does not necessarily mean five times the likeliness of 1 star. In this case, the ratings of movies can be considered categorical (or discrete). This will be an example we prefer other models than linear regression model.

The family of models that produce educated guesses that certain observation falls in a class is called **classifiers**. As opposed to what we learned before, a linear regression is a **regressor**. Hence, a classifier is used for classification problems where a regressor is used for regression problems.

The linear regression model discussed in previous chapter assumes that the response variable Y is quantitative or continuous. However, in many other situation, the response variable Y can be qualitative or categorical.

For example, the ratings of a movie can be 1 star, 2 stars, 3 stars, 4 stars, or 5 stars. In this case, given the information of a movie, a model needs to be able to produce a guess of a particular class and the 5 stars rating does not necessarily mean five times the likeliness of 1 star. In this case, the ratings of movies can be considered categorical (or discrete). This will be an example we prefer other models than linear regression model.

The family of models that produce educated guesses that certain observation falls in a class is called **classifiers**. As opposed to what we learned before, a linear regression is a **regressor**. Hence, a classifier is used for classification problems where a regressor is used for regression problems.

What are some famous classification problems?

The family of models that produce educated guesses that certain observation falls in a class is called **classifiers**. As opposed to what we learned before, a linear regression is a **regressor**. Hence, a classifier is used for classification problems where a regressor is used for regression problems.

What are some famous classification problems?

1. A person arrives at the emergency room with a set of symptoms that could possibly be attributed to one of the three medical conditions. Which of the three medical conditions does the individual have?
2. An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, on the basis of the IP address, past transaction, historical credit score, and so on.
3. On the basis of DNA sequence data for a member of patients with or without a given disease, a biologist would like to figure out which DNA mutations are deleterious (disease-causing) and which are not.

Why not linear regression?

Why not linear regression? Suppose we have a simple example. There are three medical conditions. We can write this outcome variable formally below based on these three medical conditions.

$$Y = \begin{cases} 1 & \text{if stroke;} \\ 2 & \text{if drug overdose;} \\ 3 & \text{if epileptic seizure.} \end{cases}$$

Mathematically speaking, linear regression can map the predictors to a range of educated guesses to real line. The real line is the set of all real numbers which can be from negative infinity to positive infinity. This means random numbers such as -5, 0.5, 2.7, 18 can all possibly appear in the predicted values. We cannot interpret any of these values in reality to human end-users.

So what do we do?

Why not linear regression? Suppose we have a simple example. There are three medical conditions. We can write this outcome variable formally below based on these three medical conditions.

$$Y = \begin{cases} 1 & \text{if stroke;} \\ 2 & \text{if drug overdose;} \\ 3 & \text{if epileptic seizure.} \end{cases}$$

What do we do? We convert this outcome variable to binary form with regarding to one particular class. For example, why don't we try to convert 1, 2, 3 into 1 or 0 by defining the outcome to be stroke versus drug overdose and epileptic seizure. In other words, the proposal is to write the following

$$Y = \begin{cases} 0 & \text{if stroke;} \\ 1 & \text{if drug overdose.} \end{cases}$$

Back to top.

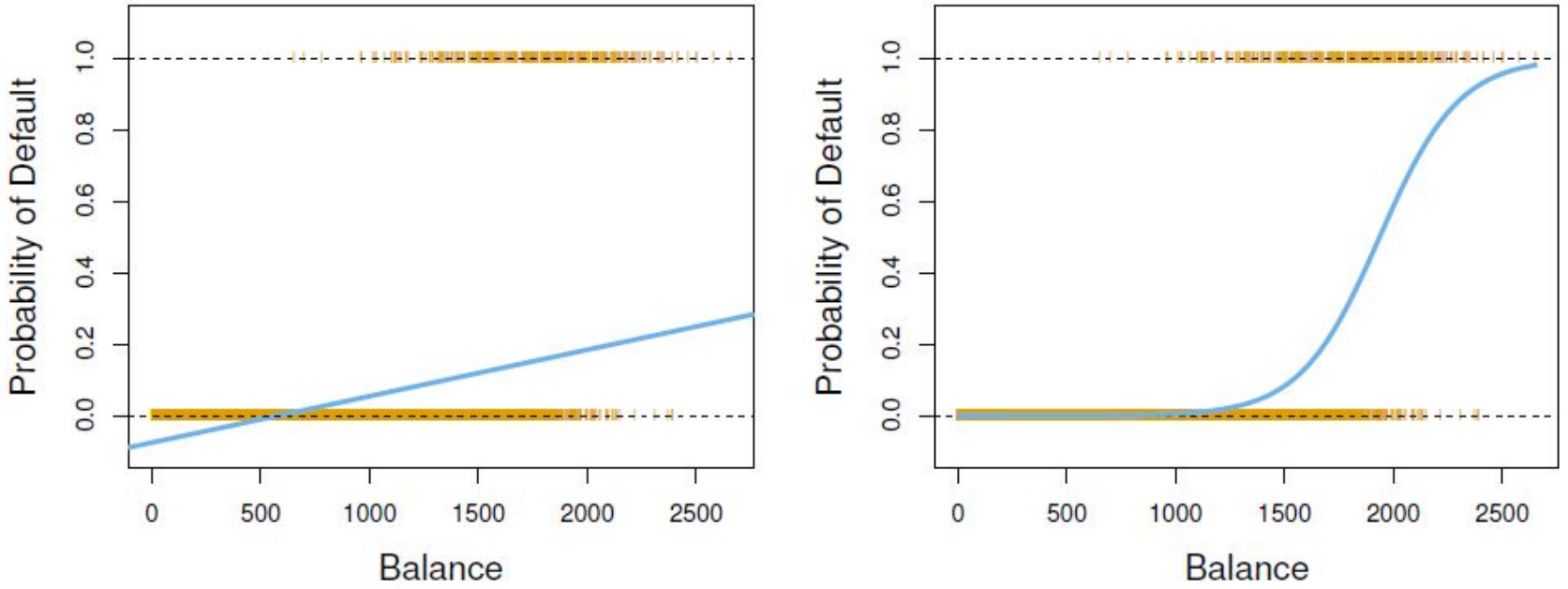


FIGURE 4.2. *Classification using the* `Default` *data.* Left: *Estimated probability of* `default` *using linear regression. Some estimated probabilities are negative! The orange ticks indicate the 0/1 values coded for* `default` *(*`No` *or* `Yes`*).* Right: *Predicted probabilities of* `default` *using logistic regression. All probabilities lie between* 0 *and* 1.

Back to top.

Consider again the `Default` data set, where the response `default` falls into one of two categories, `Yes` or `No`. Rather than modeling this response Y directly, logistic regression models the *probability* that Y belongs to a particular category.

For the `Default` data, logistic regression models the probability of default. For example, the probability of default given `balance` can be written as

$$\Pr(\texttt{default} = \texttt{Yes}|\texttt{balance}).$$

Back to top.

From the figure two slides before, the linear regression model produces educated guesses to the entire real line while in reality we want a model that can produce educated guess that falls in between 0 and 1, which can translate into a “probability”.

The proposed model is to throw the linear component into a link function. In this case, linear component is $\beta_0 + \beta_1 X$ while the link function is called sigmoid and it takes the form $\text{sigmoid}(a) = \frac{e^a}{1+e^a}$

By replacing “a” with the linear component, we obtain the following form

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

which is called “logistic function”. Hence, this model gained its name “logistic classifier”.

Back to top.

With a little modification of the mathematical form, we obtain

$$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X}$$

where the ratio on the left hand side is called odds ratio. Then we can further modify the formula to obtain

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

where the left hand side is now logarithm of odds ratio.

How do we estimate the model? (How to fit the model?)

How do we estimate the model? (How to fit the model?)

The method we introduce here is called **maximum likelihood**. The intuition here is that we are looking for estimates of the coefficients such that the predicted probability for an observation matches as closely as possible to the default status.

To describe the intuition mathematically, we can formally write the following

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

Hence, the objective function would be

$$\max_{\beta_0,\beta_1} l(\beta_0, \beta_1)$$

Back to top.

We can fit the model and generate the following results.

	Coefficient	Std. error	z-statistic	p-value
Intercept	−10.6513	0.3612	−29.5	<0.0001
balance	0.0055	0.0002	24.9	<0.0001

TABLE 4.1. *For the* Default *data, estimated coefficients of the logistic regression model that predicts the probability of* default *using* balance*. A one-unit increase in* balance *is associated with an increase in the log odds of* default *by* 0.0055 *units.*

Back to top.

We can fit the model and generate the following results.

	Coefficient	Std. error	z-statistic	p-value
Intercept	−10.6513	0.3612	−29.5	<0.0001
balance	0.0055	0.0002	24.9	<0.0001

TABLE 4.1. *For the* Default *data, estimated coefficients of the logistic regression model that predicts the probability of* default *using* balance*. A one-unit increase in* balance *is associated with an increase in the log odds of* default *by* 0.0055 *units.*

With the above table, we obtain the coefficients and hence the following model. Given an observation, say a person's balance is $1,000, the probability of default is computed as the follow

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1{,}000}}{1+e^{-10.6513+0.0055\times 1{,}000}} = 0.00576$$

Back to top.

We can fit the model and generate the following results.

	Coefficient	Std. error	z-statistic	p-value
Intercept	−10.6513	0.3612	−29.5	<0.0001
balance	0.0055	0.0002	24.9	<0.0001

TABLE 4.1. *For the* Default *data, estimated coefficients of the logistic regression model that predicts the probability of* default *using* balance*. A one-unit increase in* balance *is associated with an increase in the log odds of* default *by* 0.0055 *units.*

With the above table, we obtain the coefficients and hence the following model. Given an observation, say a person's balance is $1,000, the probability of default is computed as the follow

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1{,}000}}{1+e^{-10.6513+0.0055\times 1{,}000}} = 0.00576$$

Back to top.

Recall before, we discussed that sometimes an outcome variable Y may have more than one levels.

$$Y = \begin{cases} 1 & \text{if stroke;} \\ 2 & \text{if drug overdose;} \\ 3 & \text{if epileptic seizure.} \end{cases}$$

We can use multinomial logistic regression. To do this, we first select a single class called **baseline**. Without loss of generality, we select the kth class for this and we can write the model formally in the following

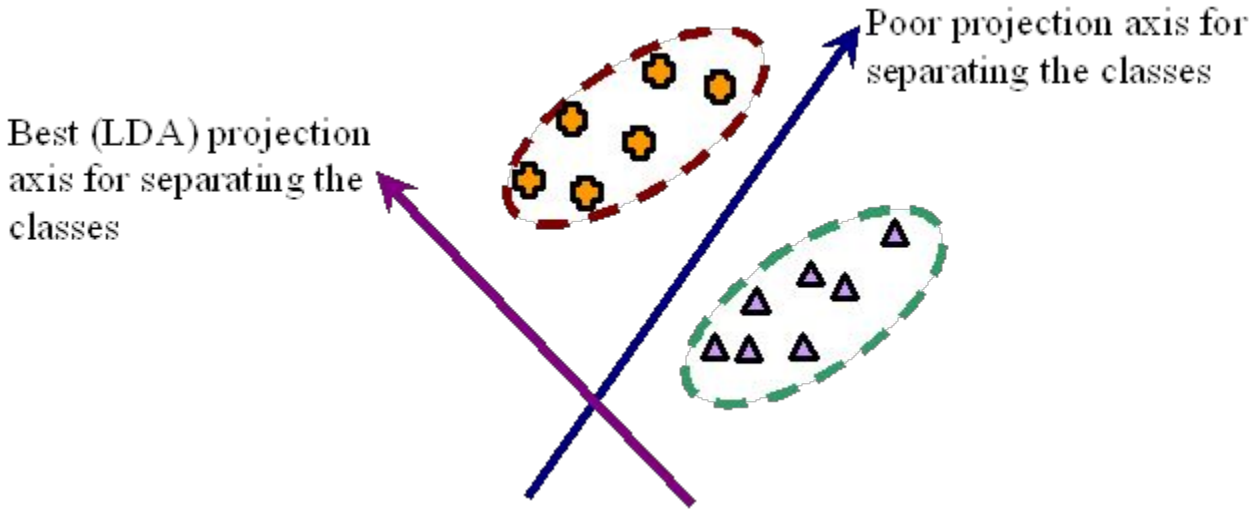
$$\Pr(Y = k|X = x) = \frac{e^{\beta_{k0}+\beta_{k1}x_1+\cdots+\beta_{kp}x_p}}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0}+\beta_{l1}x_1+\cdots+\beta_{lp}x_p}}$$

which we can translate to the following

$$\log\left(\frac{\Pr(Y = k|X = x)}{\Pr(Y = K|X = x)}\right) = \beta_{k0} + \beta_{k1}x_1 + \cdots + \beta_{kp}x_p$$

Back to top.

Linear Discriminant Analysis is a dimensionality reduction technique used as a preprocessing step in Machine Learning and pattern classification applications.



Back to top.

Linear Discriminant Analysis is a dimensionality reduction technique used as a preprocessing step in Machine Learning and pattern classification applications.

The first step is to calculate the separability between different classes(i.e the distance between the mean of different classes) also called as between-class variance

$$S_b = \sum_{i=1}^{g} N_i (\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T$$

Back to top.

Linear Discriminant Analysis is a dimensionality reduction technique used as a preprocessing step in Machine Learning and pattern classification applications.

The first step is to calculate the separability between different classes(i.e the distance between the mean of different classes) also called as between-class variance

$$S_b = \sum_{i=1}^{g} N_i (\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T$$

Second Step is to calculate the distance between the mean and sample of each class,which is called the within class variance

$$S_w = \sum_{i=1}^{g} (N_i - 1) S_i = \sum_{i=1}^{g} \sum_{j=1}^{N_i} (x_{i,j} - \bar{x}_i)(x_{i,j} - \bar{x}_i)^T$$

Back to top.

Linear Discriminant Analysis is a dimensionality reduction technique used as a preprocessing step in Machine Learning and pattern classification applications.

The first step is to calculate the separability between different classes(i.e the distance between the mean of different classes) also called as between-class variance

$$S_b = \sum_{i=1}^{g} N_i (\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T$$

Second Step is to calculate the distance between the mean and sample of each class,which is called the within class variance

$$S_w = \sum_{i=1}^{g} (N_i - 1) S_i = \sum_{i=1}^{g} \sum_{j=1}^{N_i} (x_{i,j} - \bar{x}_i)(x_{i,j} - \bar{x}_i)^T$$

The third step is to construct the lower dimensional space which maximizes the between class variance and minimizes the within class variance.Let P be the lower dimensional space projection,which is called Fisher's criterion

$$P_{lda} = \arg\max_{P} \frac{\left| P^T S_b P \right|}{\left| P^T S_w P \right|}$$

Back to top.

Extension to LDA :

Linear Discriminant Analysis is a simple and effective method for classification. Because it is simple and so well understood, there are many extensions and variations to the method. Some popular extensions include:

- **Quadratic Discriminant Analysis (QDA)**: Each class uses its own estimate of variance (or covariance when there are multiple input variables).
- **Flexible Discriminant Analysis (FDA)**: Where non-linear combinations of inputs is used such as splines.
- **Regularized Discriminant Analysis (RDA)**: Introduces regularization into the estimate of the variance (actually covariance), moderating the influence of different variables on LDA.

Chapter 5 - Sampling and Bootstrap

Go back to main content, click here

Back to top.

https://www.youtube.com/YiqiaoYin

Resampling methods are indispensable tool in machine learning. They involve repeatedly drawing samples from a training and refitting a model of interest on each sample in order obtain additional information about fitted model.

Two most famous common procedure we are introducing in this chapter are **cross-validation** and **bootstrap**.

Why are we interested in them?

Back to top.

Resampling methods are indispensable tool in machine learning. They involve repeatedly drawing samples from a training and refitting a model of interest on each sample in order obtain additional information about fitted model.

Two most famous common procedure we are introducing in this chapter are **cross-validation** and **bootstrap**.

Why are we interested in them?

Cross-validation can be used to estimate the test error associated with a given statistical learning method in order to evaluate the performance. The process of evaluating a model's performance is model assessment, where as the process of selecting the proper level of flexibility is model selection.

Bootstrap is used in a wide range of topics. The most common one is the measure of accuracy of a parameter estimate or of a given statistical learning method.

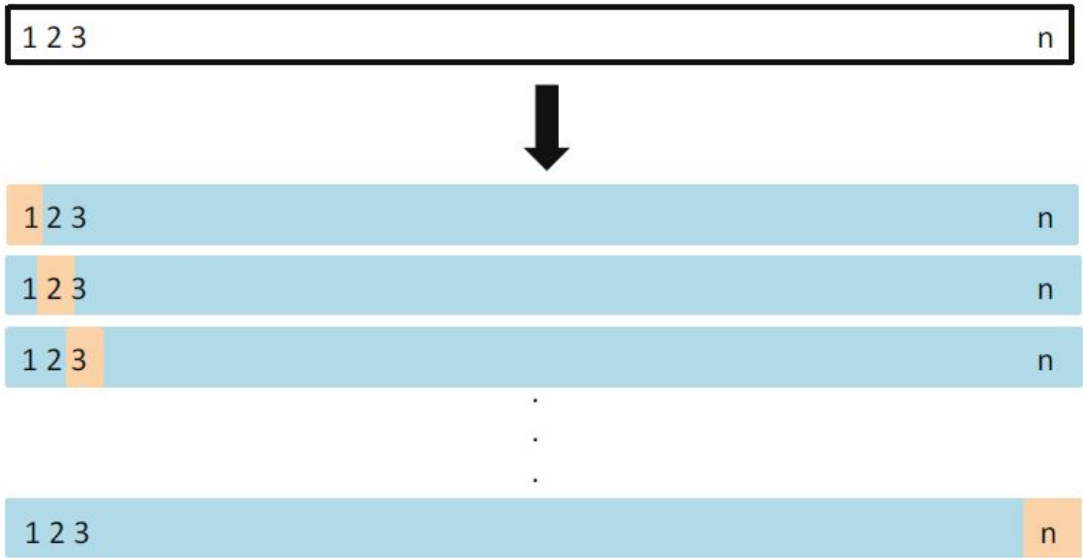


FIGURE 5.3. *A schematic display of LOOCV. A set of n data points is repeatedly split into a training set (shown in blue) containing all but one observation, and a validation set that contains only that observation (shown in beige). The test error is then estimated by averaging the n resulting MSE's. The first training set contains all but observation 1, the second training set contains all but observation 2, and so forth.*

Back to top.

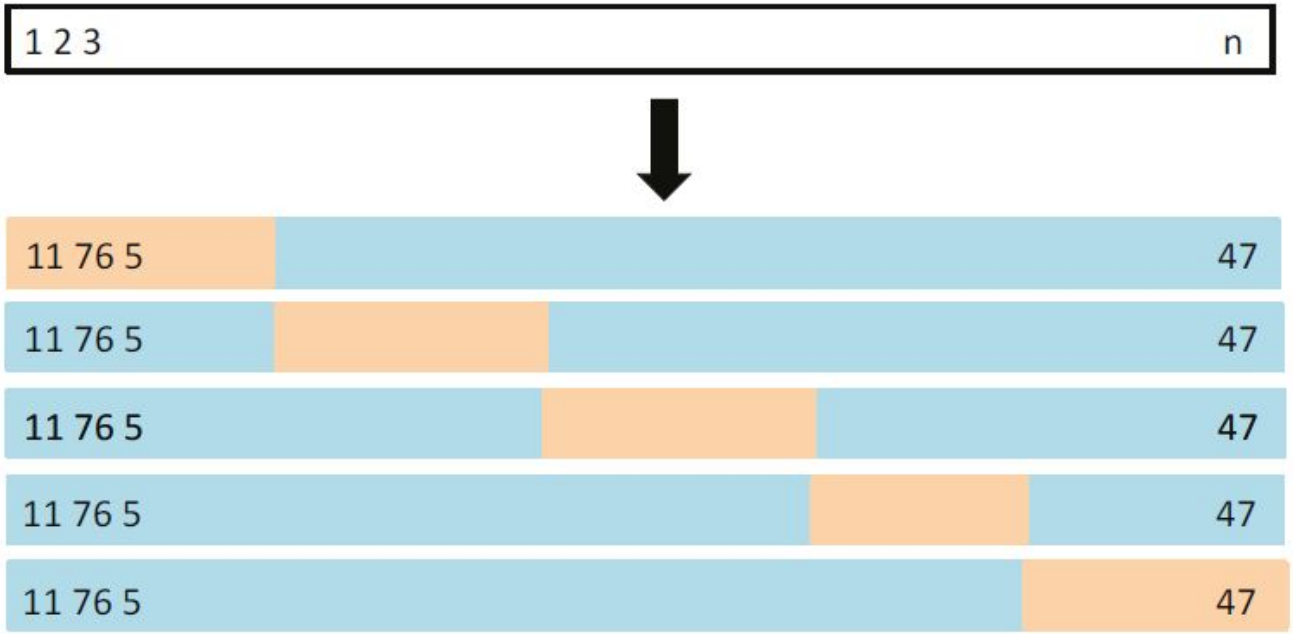


FIGURE 5.5. *A schematic display of 5-fold CV. A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE estimates.*

Back to top.

Suppose there are two financial assets that yield returns X and Y. Suppose we invest a fraction α in the first asset and returns X. Then that means we have only $1-\alpha$ to invest in the second asset and obtain Y.

The risk of the portfolio consisting of X and Y would then be

$$\text{var}(\alpha X + (1-\alpha)Y)$$

and hence we can rewrite, by minimizing the risk, the following

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$$

In reality, the variance of Y, the variance of X, and their covariance are all unknown. This leaves us to estimate

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$$

Back to top.

In reality, the variance of Y, the variance of X, and their covariance are all unknown. This leaves us to estimate

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$$

How to estimate them?

Back to top.

In reality, the variance of Y, the variance of X, and their covariance are all unknown. This leaves us to estimate

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$$

How to estimate them?

We can simulate 100 pairs of returns of X and Y. Then we can use these returns to generate the estimate of variance of X, the estimate of the variance of Y, and the estimate of the covariance of X and Y.

How do we know this way is better?

Back to top.

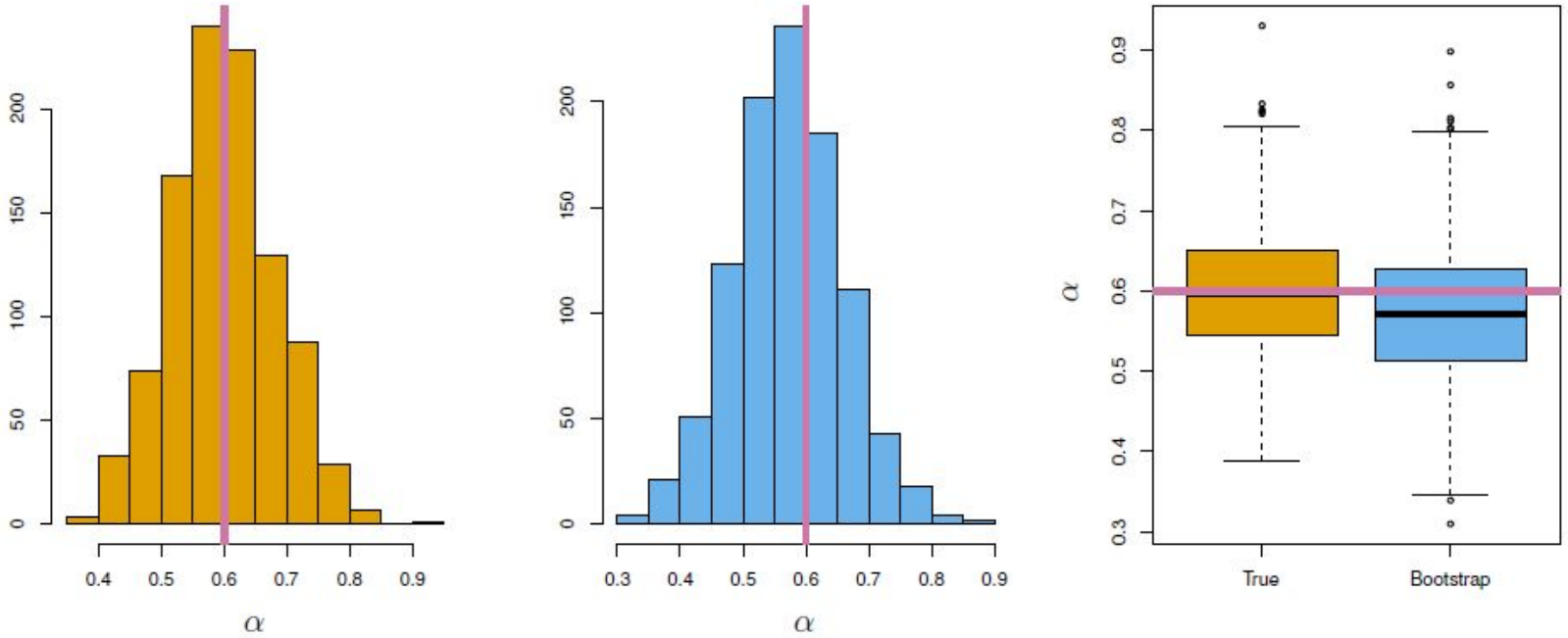


FIGURE 5.10. Left: *A histogram of the estimates of α obtained by generating 1,000 simulated data sets from the true population.* Center: *A histogram of the estimates of α obtained from 1,000 bootstrap samples from a single data set.* Right: *The estimates of α displayed in the left and center panels are shown as boxplots. In each panel, the pink line indicates the true value of α.*

Back to top.

Chapter 6 - Model Selection & Regularization

Go back to main content, click here

Back to top.

This chapter and the next chapter further explore the concepts of linear model framework. Here are some motivations for this chapter.

I. **Prediction Accuracy**: Provided that the true relationship between the response and the predictors is approximately linear, the least squares estimates will have low bias. If the sample size (n) is larger than the number of parameters (p), then the least squares estimates tend to have low variance. However, the where n is somewhat larger than p will lead to a lot of variability in the least squares fit. If p is greater than n, then there is no unique solution to the problem and the variance will be infinite. By constraining or shrinking the estimated coefficients, we can often substantially reduce the variance at the cost of a negligible increase in bias.

II. **Model Interpretability**: It is often the case some or many variables used in the linear model are not associated with the response. Including the unnecessary, noisy, or redundant variables in the model leads to unnecessary complexity of model development. By removing these variables, we can obtain a model that is more interpretable. Here we introduce some feature selection technique.

To perform variable set selection, we fit a separate least squares regression for each possible combination of the p predictors. That is, we fit all p models that contain exactly one predictor, all “p choose 2” models that contain exactly two predictors. We then look at what is the best of them. Here “p choose 2” means

$$\binom{p}{2} = \frac{p!}{2!(p-2)!} = \frac{p(p-1)(p-2)!}{2(p-2)!} = \frac{p(p-1)}{2}$$

To do this, we introduce the following algorithm:

Algorithm 6.1 *Best subset selection*

1. Let $\mathcal{M}_0$ denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.

2. For $k = 1, 2, \ldots p$:

   (a) Fit all $\binom{p}{k}$ models that contain exactly k predictors.

   (b) Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. Here *best* is defined as having the smallest RSS, or equivalently largest R^2.

3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Back to top.

Forward stepwise selection is a computationally efficient alternative to the best subset selection. While the best subset selection procedure considers all 2^p possible models, forward stepwise considers a much smaller set of models.

This approach begins with a model containing no predictors, and then adds predictors to the model, one at a time, until all of the predictors are in the model.

Why is it better?

> Unlike best subset selection, which involved fitting 2^p models, forward stepwise selection involves fitting one null model, along with $p-k$ models in the kth iteration, for $k = 0, \ldots, p-1$. This amounts to a total of $1 + \sum_{k=0}^{p-1}(p-k) = 1+p(p+1)/2$ models. This is a substantial difference: when $p = 20$, best subset selection requires fitting 1,048,576 models, whereas forward stepwise selection requires fitting only 211 models.[1]

Back to top.

Algorithm 6.2 *Forward stepwise selection*

1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
   (a) Consider all $p - k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
   (b) Choose the *best* among these $p - k$ models, and call it $\mathcal{M}_{k+1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Back to top.

Like the forward stepwise selection, **backward stepwise selection** provides efficient alternative to best subset selection. However, unlike the forward stepwise selection, the backward stepwise selection begins with all p predictors, and then iteratively removes the least useful predictor, one at a time.

Back to top.

Like the forward stepwise selection, **backward stepwise selection** provides efficient alternative to best subset selection. However, unlike the forward stepwise selection, the backward stepwise selection begins with all p predictors, and then iteratively removes the least useful predictor, one at a time.

Algorithm 6.3 *Backward stepwise selection*

1. Let $\mathcal{M}_p$ denote the *full* model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   (a) Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   (b) Choose the *best* among these k models, and call it $\mathcal{M}_{k-1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Back to top.

Recall all algorithms discussed above have certain procedures of testing model using some sort of loss function or error function. The basic loss function we discussed before in linear regression is the squared loss. We can also have residual sum of square (RSS). In addition, we have **Cp**, **AIC**, and **BIC**. We introduce these approaches here:

I. **Cp**. For fitting least square model containing d predictors, the Cp estimate the test MSE is computed using

$$C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$$

where the symbol sigma hat square is an estimate of the variance of the error epsilon in the regression model. The key here is that the value of Cp adds penalty of $2d\hat{\sigma}^2$ to the training RSS in order to justify the fact that the training error tends to underestimate the test error.

Back to top.

Recall all algorithms discussed above have certain procedures of testing model using some sort of loss function or error function. The basic loss function we discussed before in linear regression is the squared loss. We can also have residual sum of square (RSS). In addition, we have **Cp**, **AIC**, and **BIC**. We introduce these approaches here:

I. **Cp**. $C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$

II. **AIC or Akaike Information Criterion**. Suppose we have a statistical model with d predictors and estimate of the likelihood function. Then AIC is defined as

$$\text{AIC} = 2d - 2\ln \hat{L}$$

so the value of AIC rewards the goodness of fit, but it also penalizes the model when there is a large number of predictors. When we assume gaussian errors, the maximum likelihood and the least squares are essentially equivalent. Hence, we can also write

$$\text{AIC} = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$$

which you can see is the same as Cp.

Back to top.

Recall all algorithms discussed above have certain procedures of testing model using some sort of loss function or error function. The basic loss function we discussed before in linear regression is the squared loss. We can also have residual sum of square (RSS). In addition, we have **Cp**, **AIC**, and **BIC**. We introduce these approaches here:

I. **Cp**. $C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$
II. **AIC**. $\text{AIC} = 2d - 2\ln \hat{L}$ $\text{AIC} = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$
III. **BIC or Bayesian Information Criterion**. BIC is derived from a Bayesian point of view, but ends up looking similar to Cp and AIC. For least squares with d predictors, the BIC is formally written as

$$\text{BIC} = \frac{1}{n}(\text{RSS} + \log(n)d\hat{\sigma}^2)$$

where it penalizes the least square model by a similar penalty as that in AIC except that we have logarithm of n (sample size) instead of 2d (2 times the number of predictors).

Back to top.

Recall all algorithms discussed above have certain procedures of testing model using some sort of loss function or error function. The basic loss function we discussed before in linear regression is the squared loss. We can also have residual sum of square (RSS). In addition, we have **Cp**, **AIC**, and **BIC**. We introduce these approaches here:

I. **Cp**. $C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$
II. **AIC**. $\text{AIC} = 2d - 2\ln \hat{L}$ $\text{AIC} = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$
III. **BIC**. $\text{BIC} = \frac{1}{n}(\text{RSS} + \log(n)d\hat{\sigma}^2)$
IV. **Adjusted R-square**. Another popular metric is the adjusted R-square. When we evaluate the linear regression model, we can use R-square, which is defined as 1 - RSS/TSS. The component of RSS/TSS is the residual sum of square over the total sum of square, which means the amount of variance in the error of the model. The complement of this ratio would, therefore, be the the amount of variance contributed from the model. We can extend this idea to the adjusted R-square, which is formally given in the following

$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

which can be interpreted the following way. Once all the of the correct variables have been included in the model, adding additional noisy variable would reduce adjusted R-square by a little.

Back to top.

Recall all algorithms discussed above have certain procedures of testing model using some sort of loss function or error function. The basic loss function we discussed before in linear regression is the squared loss. We can also have residual sum of square (RSS). In addition, we have **Cp**, **AIC**, and **BIC**. We introduce these approaches here:

I. **Cp**. $C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$
II. **AIC**. $\text{AIC} = 2d - 2\ln\hat{L}$ $\text{AIC} = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$
III. **BIC**. $\text{BIC} = \frac{1}{n}(\text{RSS} + \log(n)d\hat{\sigma}^2)$
IV. **Adjusted R-square**. $\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$

Next topic coming up: shrinkage!

Recall all algorithms discussed above have certain procedures of testing model using some sort of loss function or error function. The basic loss function we discussed before in linear regression is the squared loss. We can also have residual sum of square (RSS). In addition, we have **Cp**, **AIC**, and **BIC**. We introduce these approaches here:

I. **Cp**. $C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$
II. **AIC**. $\text{AIC} = 2d - 2\ln \hat{L}$ $\text{AIC} = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$
III. **BIC**. $\text{BIC} = \frac{1}{n}(\text{RSS} + \log(n)d\hat{\sigma}^2)$
IV. **Adjusted R-square**. $\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$
V. **Ridge Regression**. Recall that fitting least square is to minimize

$$\text{RSS} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2$$

and the ridge regression minimize

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \text{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2$$

By using cross-validation, we can select the model that achieves the least ridge regression error (the summation of RSS and the penalization).

Back to top.

Recall all algorithms discussed above have certain procedures of testing model using some sort of loss function or error function. The basic loss function we discussed before in linear regression is the squared loss. We can also have residual sum of square (RSS). In addition, we have **Cp**, **AIC**, and **BIC**. We introduce these approaches here:

I. **Cp**. $C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$
II. **AIC**. $\text{AIC} = 2d - 2\ln \hat{L}$ $\text{AIC} = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$
III. **BIC**. $\text{BIC} = \frac{1}{n}(\text{RSS} + \log(n)d\hat{\sigma}^2)$
IV. **Adjusted R-square**. $\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$
V. **Ridge Regression**. $\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \text{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2$
VI. **Lasso Regression**. A problem in ridge regression is that the lambda term will shrink all of the coefficients towards zero. This may not be a problem for prediction accuracy, but it may prevent us to interpret the model correctly. Hence, we make adjustment and we hereby introduce lasso regression, which aims to minimize the following

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \text{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|$$

Back to top.

Recall all algorithms discussed above have certain procedures of testing model using some sort of loss function or error function. The basic loss function we discussed before in linear regression is the squared loss. We can also have residual sum of square (RSS). In addition, we have **Cp**, **AIC**, and **BIC**. We introduce these approaches here:

I. **Cp**. $C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$
II. **AIC**. $\text{AIC} = 2d - 2\ln\hat{L}$ $\text{AIC} = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$
III. **BIC**. $\text{BIC} = \frac{1}{n}(\text{RSS} + \log(n)d\hat{\sigma}^2)$
IV. **Adjusted R-square**. $\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$
V. **Ridge Regression**. $\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \text{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2$
VI. **Lasso Regression**. $\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \text{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|$

We can instead summarize ridge and lasso in a different way.

Recall all algorithms discussed above have certain procedures of testing model using some sort of loss function or error function. The basic loss function we discussed before in linear regression is the squared loss. We can also have residual sum of square (RSS). In addition, we have **Cp**, **AIC**, and **BIC**. We introduce these approaches here:

I. **Cp**. $C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$

II. **AIC**. $\text{AIC} = 2d - 2\ln \hat{L}$ $\text{AIC} = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$

III. **BIC**. $\text{BIC} = \frac{1}{n}(\text{RSS} + \log(n)d\hat{\sigma}^2)$

IV. **Adjusted R-square**. $\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$

V. **Ridge Regression**. $\min_{\beta} \left\{ \sum_{i=1}^{p} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right) \right\} \text{ subject to } \sum_{j=1}^{p} \beta_j^2 \leq s$

VI. **Lasso Regression**. $\min_{\beta} \left\{ \sum_{i=1}^{p} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right) \right\} \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq s$

Back to top.

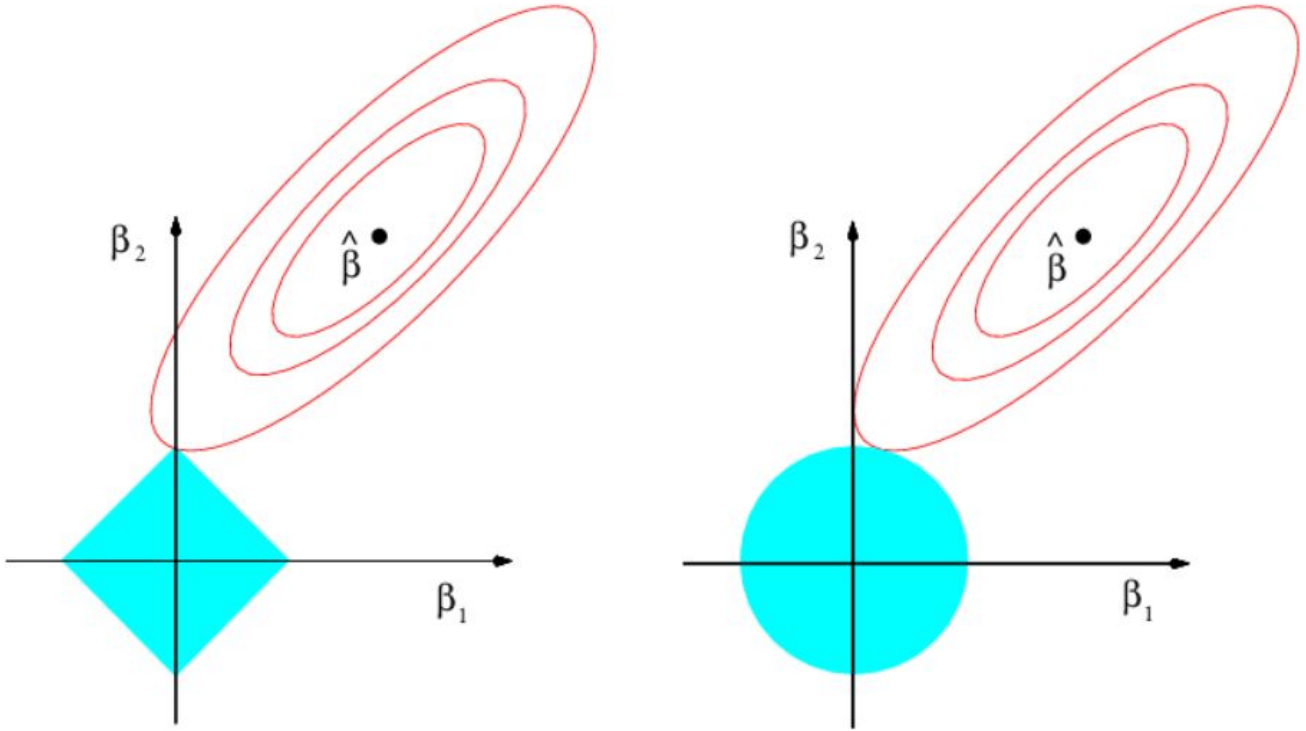
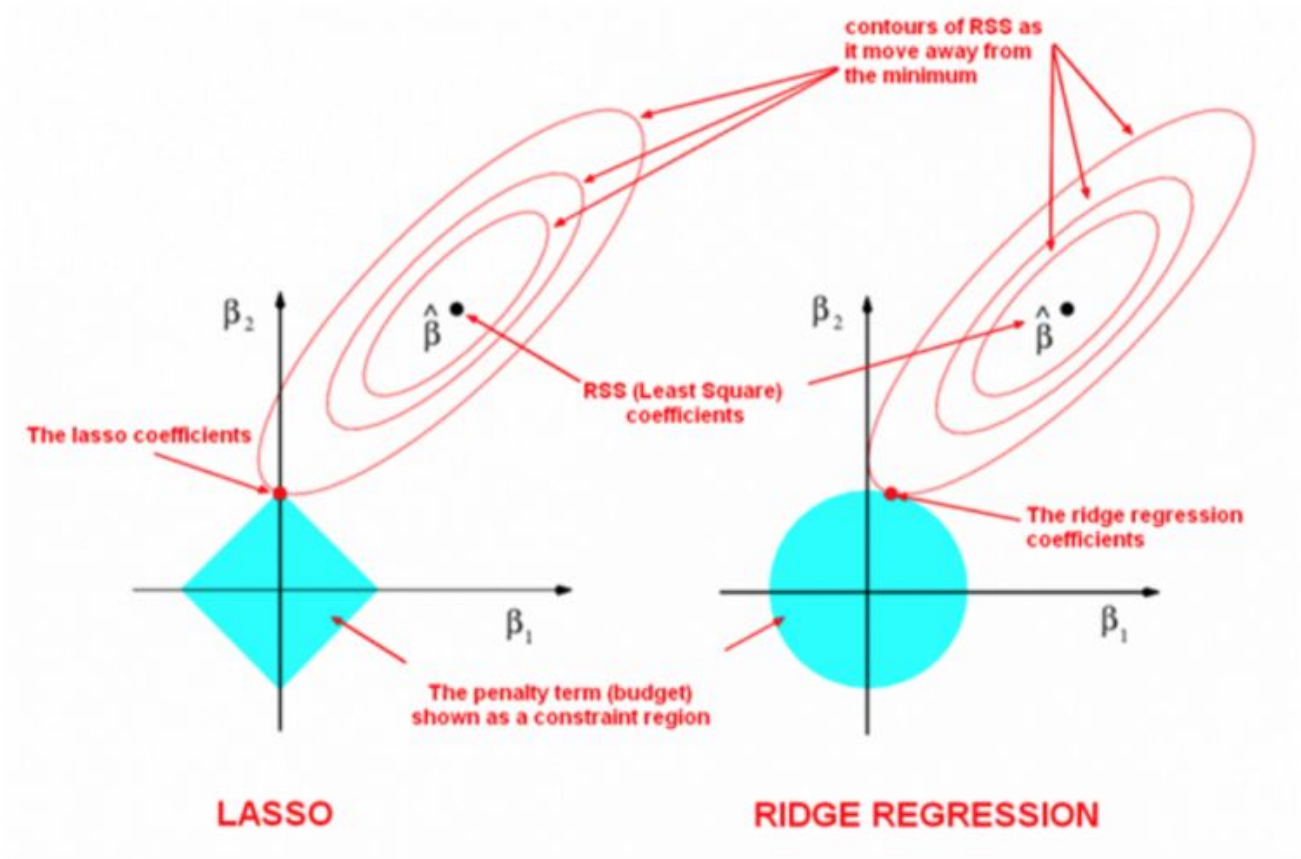


FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

Back to top.



Back to top.

Chapter 7 - Going Beyond Linearity

Go back to main content, click here

Back to top.

https://www.youtube.com/YiqiaoYin

A standard way to extend linear regression is to replace

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

with polynomial function

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d + \epsilon_i$$

where epsilon is the error term. This is called **polynomial regression**.

Back to top.

A standard way to extend linear regression is to replace

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

with polynomial function

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d + \epsilon_i$$

where epsilon is the error term. This is called **polynomial regression**.

Notice that this way we are able to extend the shape of linearity into higher-order dimensions. In other words, we are transforming from linearity to nonlinear functions.

Question: How do we pick d?

Back to top.

A standard way to extend linear regression is to replace

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

with polynomial function

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d + \epsilon_i$$

where epsilon is the error term. This is called **polynomial regression**.

Notice that this way we are able to extend the shape of linearity into higher-order dimensions. In other words, we are transforming from linearity to nonlinear functions.

Question: How do we pick d?

Answer: We can always go back to the variable selection or the model selection procedures we introduced in Chapter 6. We can start with an arbitrary number of d predictors. We can either add or subtract depending on the shrinkage procedures or the variable selection criterions.

Back to top.

Another form of nonlinear transformation is to use **step function**. The purpose of introducing step function is to break the variable X into ranges. For example, we can redefine our predictors using indicator functions such as the following

$$
\begin{aligned}
C_0(X) &= I(X < c_1), \\
C_1(X) &= I(c_1 \leq X < c_2), \\
C_2(X) &= I(c_2 \leq X < c_3), \\
&\vdots \\
C_{K-1}(X) &= I(c_{K-1} \leq X < c_K), \\
C_K(X) &= I(c_K \leq X),
\end{aligned}
$$

where the I() is the indicator function, i.e.

$$\mathbf{I}(X > a) = 1 \text{ if } X > a$$

$$\mathbf{I}(X > a) = 0 \text{ if } X \leq a$$

Back to top.

Another form of nonlinear transformation is to use **step function**. The purpose of introducing step function is to break the variable X into ranges. For example, we can redefine our predictors using indicator functions such as the following

$$
\begin{aligned}
C_0(X) &= I(X < c_1), \\
C_1(X) &= I(c_1 \leq X < c_2), \\
C_2(X) &= I(c_2 \leq X < c_3), \\
&\vdots \\
C_{K-1}(X) &= I(c_{K-1} \leq X < c_K), \\
C_K(X) &= I(c_K \leq X),
\end{aligned}
$$

where the I() is the indicator function. This way we can build a new linear model as the follows

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \cdots + \beta_K C_K(x_i) + \epsilon_i$$

Back to top.

Chapter 8 - Tree-based Method

Go back to main content, click here

Back to top.

https://www.youtube.com/YiqiaoYin

This chapter discusses tree-based methods. These methods involve stratifying or segmenting the predictor space into a number of simple regions. In order to make a prediction for a given observation, we typically use the mean or the mode response value for the training observations in the region to which it belongs. Since the summarized training can be drawn into trees, we call these methods tree-based methods.

Amongst the tree-based methods, a famous fundamental method is **decision tree**. In addition, these topics can also be extended combined with bagging and boosting, which leads to **random forests (RF)** and **bayesian additive regression tree (BART)**.

I. **Decision Tree**.

Back to top.

This chapter discusses tree-based methods. These methods involve stratifying or segmenting the predictor space into a number of simple regions. In order to make a prediction for a given observation, we typically use the mean or the mode response value for the training observations in the region to which it belongs. Since the summarized training can be drawn into trees, we call these methods tree-based methods.

Amongst the tree-based methods, a famous fundamental method is **decision tree**. In addition, these topics can also be extended combined with bagging and boosting, which leads to **random forests (RF)** and **bayesian additive regression tree (BART)**.

I. **Decision Tree**. Say we want to predict baseball players' salaries using regression tree.

Back to top.

I. **Decision Tree**. Say we want to predict baseball players' salaries using regression tree.

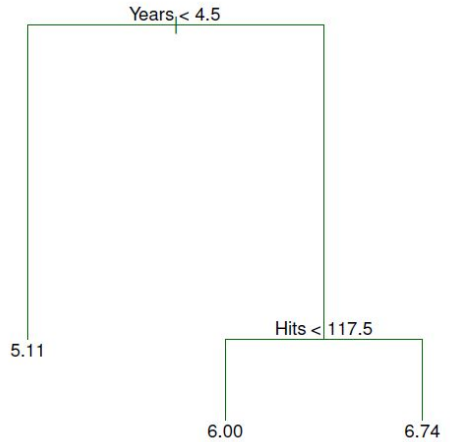


FIGURE 8.1. *For the* `Hitters` *data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year. At a given internal node, the label (of the form* $X_j < t_k$*) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to* $X_j \geq t_k$*. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to* `Years<4.5`*, and the right-hand branch corresponds to* `Years>=4.5`*. The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.*

Back to top.

I. **Decision Tree**. Say we want to predict baseball players' salaries using regression tree. How to model this?

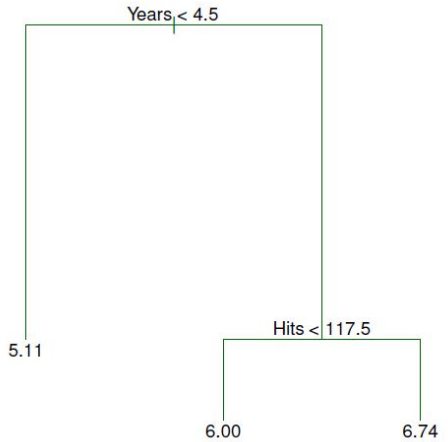
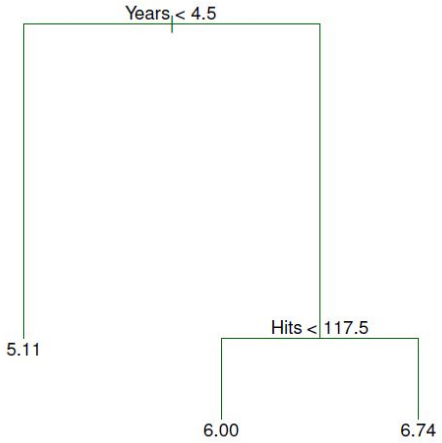


FIGURE 8.1. *For the* `Hitters` *data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year. At a given internal node, the label (of the form* $X_j < t_k$*) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to* $X_j \geq t_k$*. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to* `Years<4.5`*, and the right-hand branch corresponds to* `Years>=4.5`*. The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.*

Back to top.

I. **Decision Tree**. Say we want to predict baseball players' salaries using regression tree. How to model this?



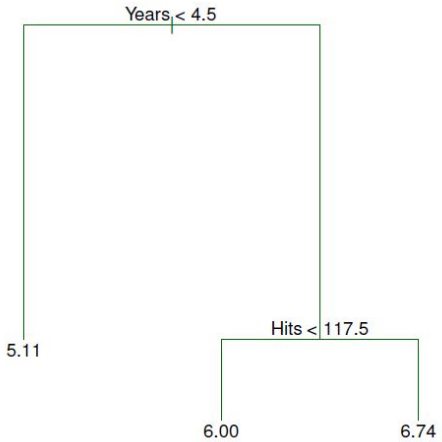
Back to top.

I. **Decision Tree**. Say we want to predict baseball players' salaries using regression tree. How to model this?

We now discuss the process of building a regression tree. Roughly speaking, there are two steps.

1. We divide the predictor space — that is, the set of possible values for $X_1, X_2, \ldots, X_p$ — into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.

2. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.



Back to top.

I. **Decision Tree**. Say we want to predict baseball players' salaries using regression tree. How to model this?

> We now discuss the process of building a regression tree. Roughly speaking, there are two steps.
>
> 1. We divide the predictor space — that is, the set of possible values for $X_1, X_2, \ldots, X_p$ — into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
>
> 2. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

How to construct the regions described in step 1? In theory, these regions can be any shape you like! The end game is to achieve the goal: minimize the following RSS, formally given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

where the y-hat term is the mean response for the training observations in the jth box. Since it's computationally infeasible to consider every possible partition of the feature space in to J boxes, we take top-down, greedy approach known as **recursive binary splitting**.

Back to top.

I. **Decision Tree**. Say we want to predict baseball players' salaries using regression tree. How to model this?

How to construct the regions described in step 1? In theory, these regions can be any shape you like! The end game is to achieve the goal: minimize the following RSS, formally given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

where the y-hat term is the mean response for the training observations in the jth box. Since it's computationally infeasible to consider every possible partition of the feature space in to J boxes, we take top-down, greedy approach known as **recursive binary splitting**.

This means from top we make splits of a variable. For example, we start with the predictor Xj and split Xj into regions that leads to the greatest possible reduction of RSS. In other words, we search for s such that $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction of RSS.

Back to top.

I. **Decision Tree**. Say we want to predict baseball players' salaries using regression tree. How to model this?

In other words, we formally write the mathematical expression as: for any j and s, we define the regions

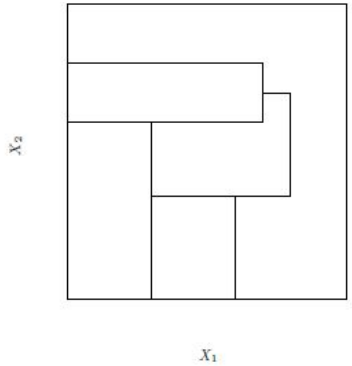
$$R_1(j,s) = \{X|X_j < s\} \text{ and } R_2(j,s) = \{X|X_j \geq s\}$$

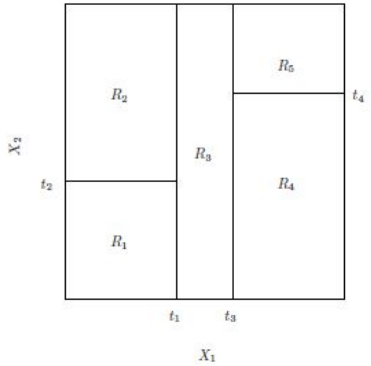
such that these regions generate predictions to allow us to minimize

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

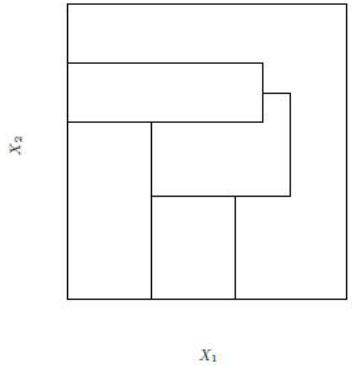
by controlling j and s. Here the term y-hat-R-one is the mean response for the training observations in region R1. Same goes with y-hat-R-two, which is the mean response for the training observations in region R2.

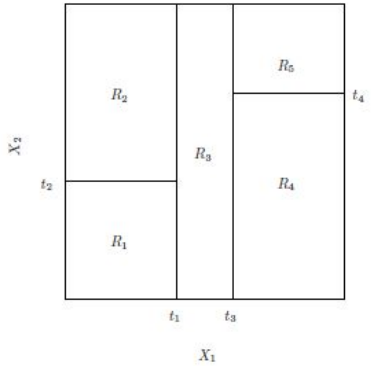
Back to top.

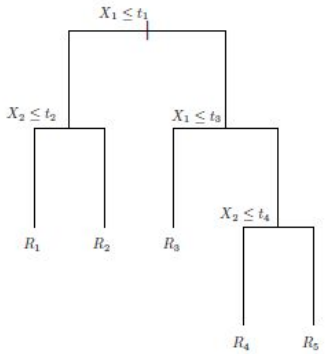


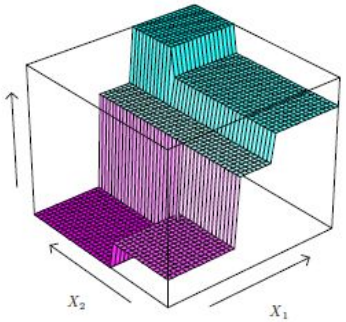


Back to top.









Back to top.

Cost complexity pruning - also known as weakest link pruning - gives us a way to use cross-validation consider the best model for decision tree. Consider a term alpha, we want to search for this alpha term such that the following term is minimized

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

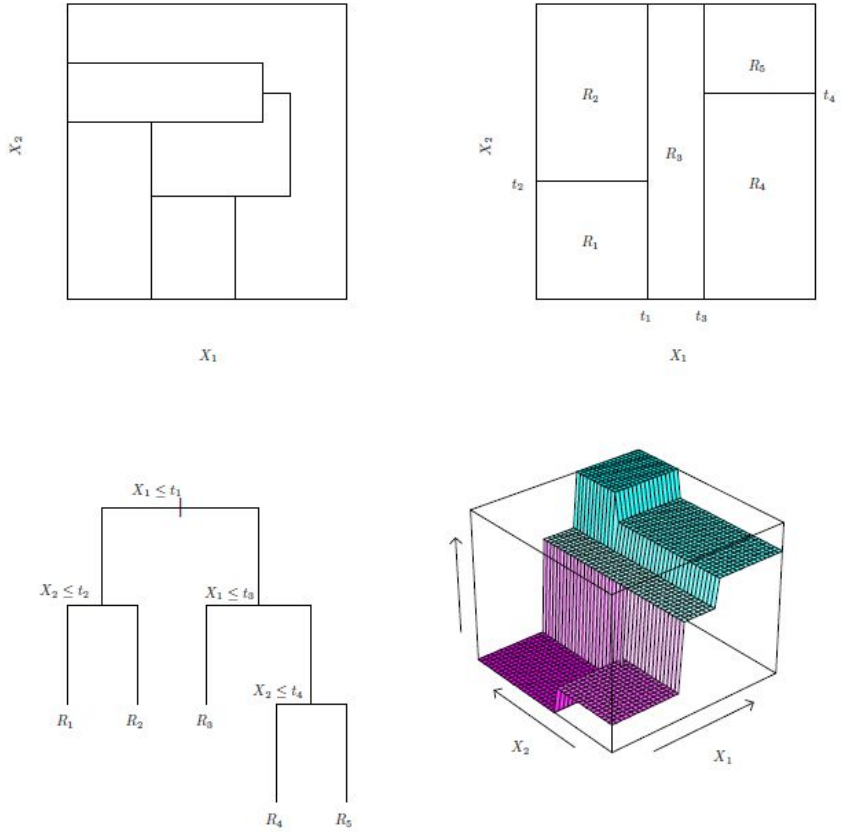


FIGURE 8.3. Top Left: *A partition of two-dimensional feature space that could not result from recursive binary splitting.* Top Right: *The output of recursive binary splitting on a two-dimensional example.* Bottom Left: *A tree corresponding to the partition in the top right panel.* Bottom Right: *A perspective plot of the prediction surface corresponding to that tree.*

Back to top.

Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Back to top.

Recall in linear regression model, we predict a continuous or quantitative variable. In logistic regression model, we predict the chance that an observation is in a certain class. Hence, we also call logistic regression model logistic classifier.

In decision tree, there is also the separation of **decision tree regressor** and **decision tree classifier**. For decision tree regressors, we want to minimize RSS. For decision tree classifiers, we want to use an alternative to RSS, which is classification error rate. If we denote $\hat{p}_{mk}$ to be the proportion of training observations in the mth region that are from the kth class, then the classification error rate is formally written as

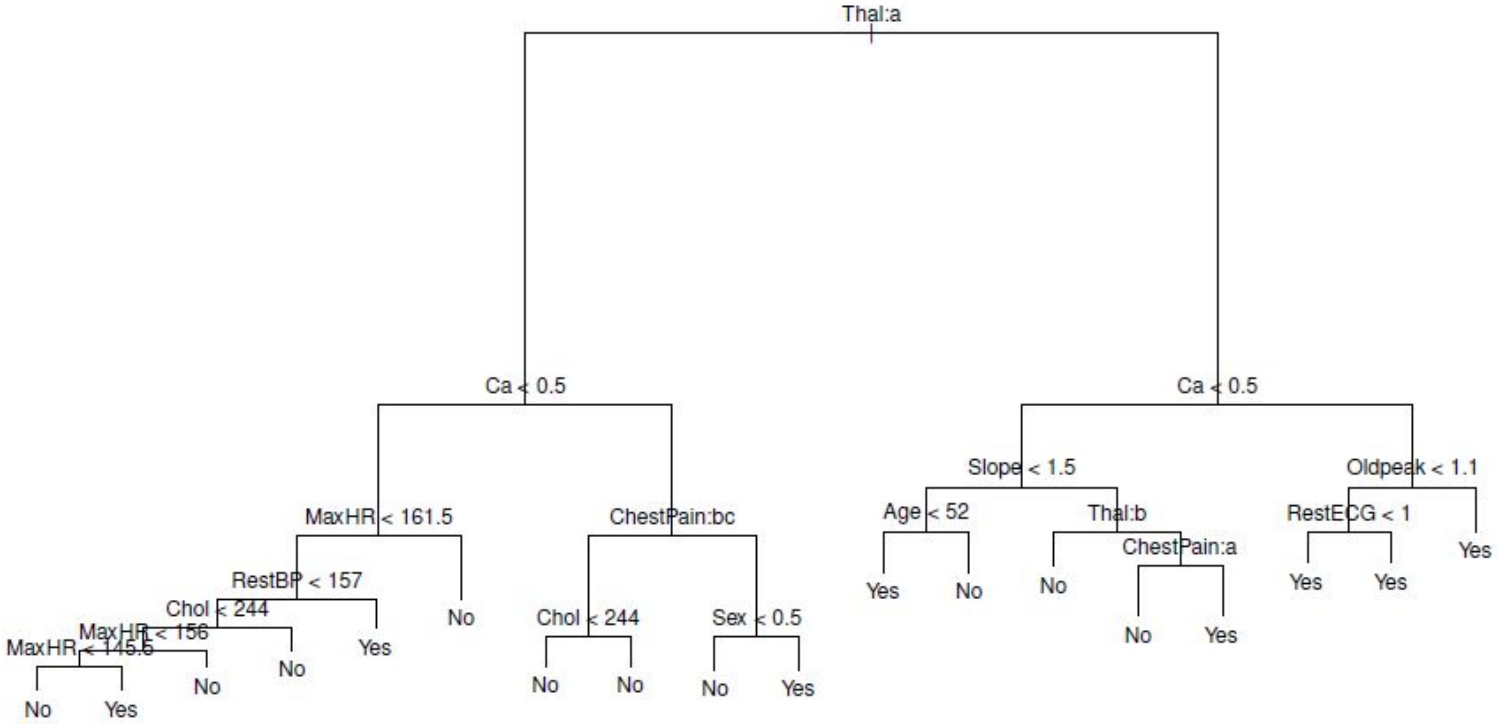
$$1 - \max_k(\hat{p}_{mk})$$

However, since the classification error rate is not sensitive enough to grow the trees, we introduce another measure called the **Gini index**

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

An alternative of Gini index is called **entropy** $D = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$

Back to top.



Back to top.

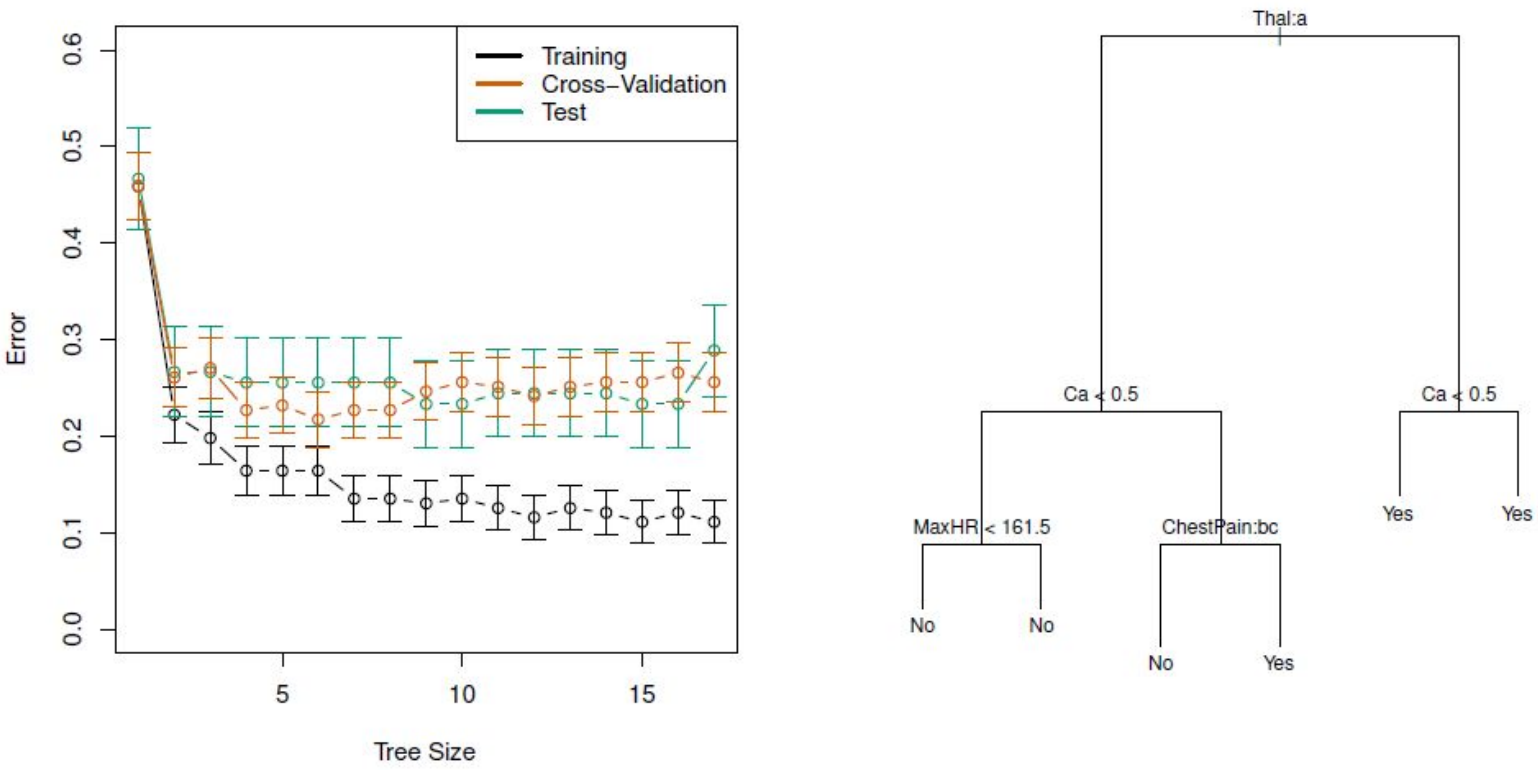


FIGURE 8.6. `Heart` *data.* Top: *The unpruned tree.* Bottom Left: *Cross-validation error, training, and test error, for different sizes of the pruned tree.* Bottom Right: *The pruned tree corresponding to the minimal cross-validation error.*

Back to top.

We discussed Bootstrap in Chapter 5, a power technique to understand the performance and sometimes accuracy of our model. We learned earlier in this chapter that decision tree makes splits of the variables in the data. This means we get less observation every split. The lack of observations lead to high variance. This is why we introduce algorithms such as **bootstrap aggregation** or **bagging**. This is a general purpose procedure for reducing the variance of a statistical learning method.

Let us briefly recall a statistical phenomenon:

Suppose we are given random samples $x_1, x_2, ..., x_n$ mean μ and variance σ^2. Suppose we denote x-bar to be the sample mean. Then, we can derive the following

$$\mathbb{E}(\bar{X}) = \mathbb{E}(\frac{1}{n}\sum_{i=1}^{n} x_i) = \frac{1}{n}\mathbb{E}(x_1 + \cdots + x_n) = \frac{n}{n}\mathbb{E}x_1 = \mu$$

$$\text{var}(\bar{X}) = \text{var}(\frac{1}{n}\sum_{i=1}^{n} x_i) = \frac{1}{n^2}\text{var}(x_1 + \dots) = \frac{n}{n^2}\text{var}(x_1) = \frac{\sigma^2}{n}$$

In other words, an important statistical inference here is average a set of observations reduce variance! This is the core idea in **bagging**!

Back to top.

Suppose we a bunch of models

$$\hat{f}^1(x), \hat{f}^2(x), ..., \hat{f}^B(x)$$

using B separate training sets. Then average them generate low-variance statistical learning models, and we can formally write the following

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B}(\hat{f}^1(x) + \hat{f}^2(x) + \cdots + \hat{f}^B(x)) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

However, under the scenario that we do not have access to multiple training sets with sufficient data, we can use bagging instead. Bagging is bootstrap aggregation, which means we do B rounds bootstrap and aggregate the models. We formally write it below

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}(\hat{f}^{*1}(x) + \hat{f}^{*2}(x) + \cdots + \hat{f}^{*B}(x)) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

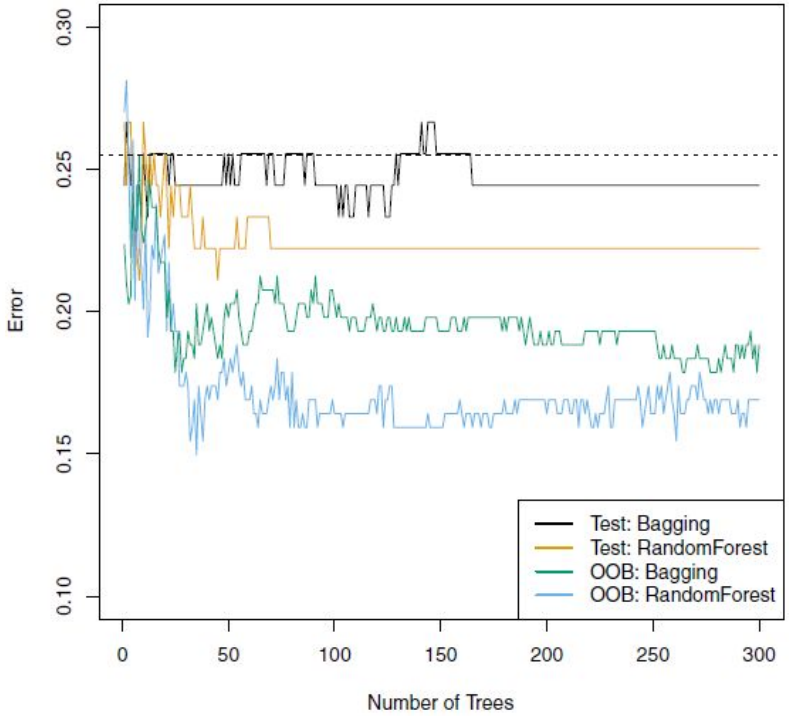


FIGURE 8.8. *Bagging and random forest results for the* `Heart` *data. The test error (black and orange) is shown as a function of* B*, the number of bootstrapped training sets used. Random forests were applied with* $m = \sqrt{p}$*. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which in this case is — by chance — considerably lower.*

Back to top.

II. **Random Forests**.

Random forests (RF) provide an improvement over bagged trees by way of a tweak that decorrelates the trees. When we build decision trees, each split we can randomly sample m predictors from the full set of p predictors. Typically, m is chosen as square root of p.

An interesting phenomenon:

In RF, we build many decision trees with each tree to select randomly sampled predictors. We also mentioned that a number of m predictors are sampled each tree while m is typically given as the square root of p. This may sound crazy because each tree does not even consider all the predictors!

This is true. But it has a clever rationale.

It turns out that it is often times a set of variables together form an important “information cluster” or “information hub”. We call this group of variables that have joint effect on predicting the outcome a **variable module**. RF has the potential power to select an important variable module, because RF only considers a variable set with size m.

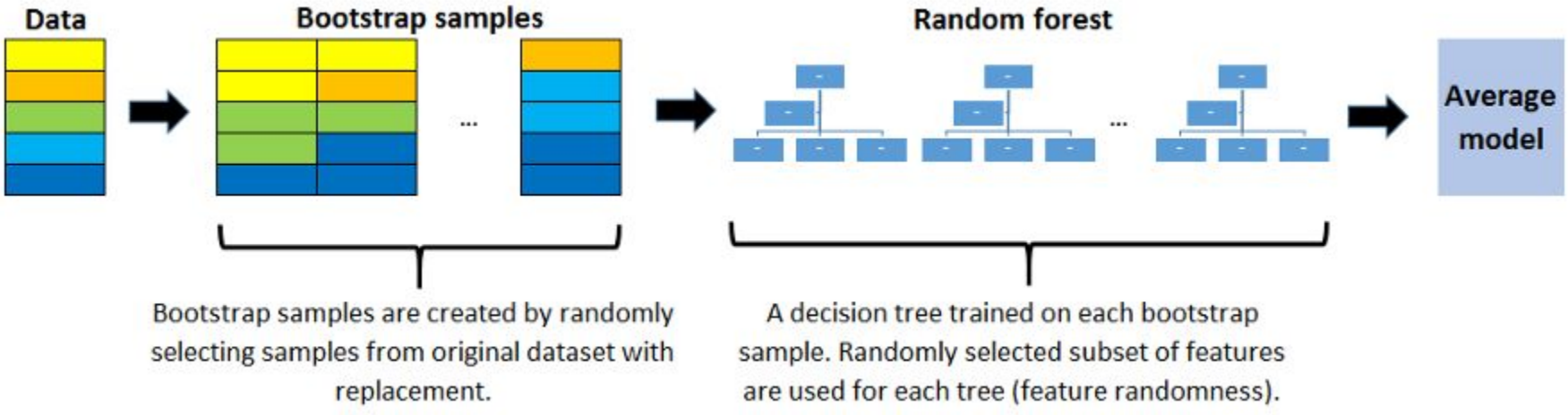
Random Forests can be understood using this diagram next.

Back to top.

II. **Random Forests**.

Random forests (RF) provide an improvement over bagged trees by way of a tweak that decorrelates the trees. When we build decision trees, each split we can randomly sample m predictors from the full set of p predictors. Typically, m is chosen as square root of p.

Random Forests can be understood using this diagram:



Back to top.

II. **Random Forests**.

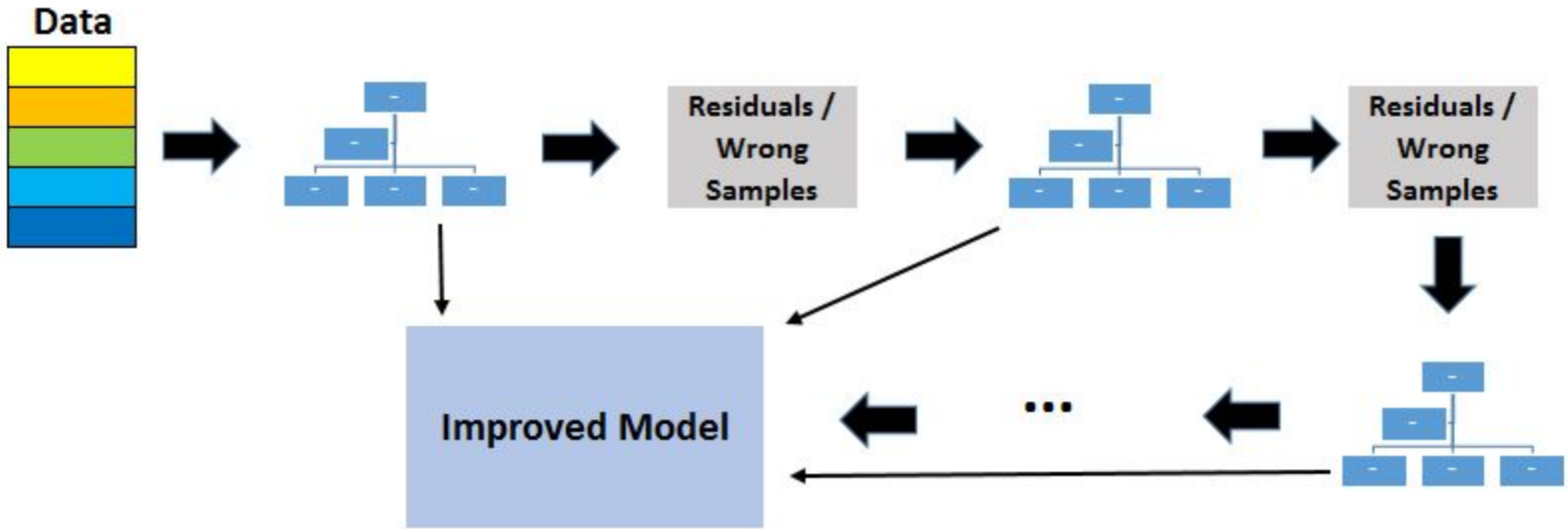
Random forests (RF) provide an improvement over bagged trees by way of a tweak that decorrelates the trees. When we build decision trees, each split we can randomly sample m predictors from the full set of p predictors. Typically, m is chosen as square root of p.

We introduce the algorithm next.

Back to top.

III. **Boosting Tree or Gradient Boosting Machine (GBM)**.

Instead of the taking the average model in the end (such as Random Forests), Gradient Boosting Machine or Gradient Boosting Trees suggests to using the incorrect or wrong samples and learn from them.



Back to top.

III. **Boosting Tree or Gradient Boosting Machine (GBM).**

Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.

2. For $b = 1, 2, \ldots, B$, repeat:

   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).

   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$

   (c) Update the residuals,

   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

Back to top.

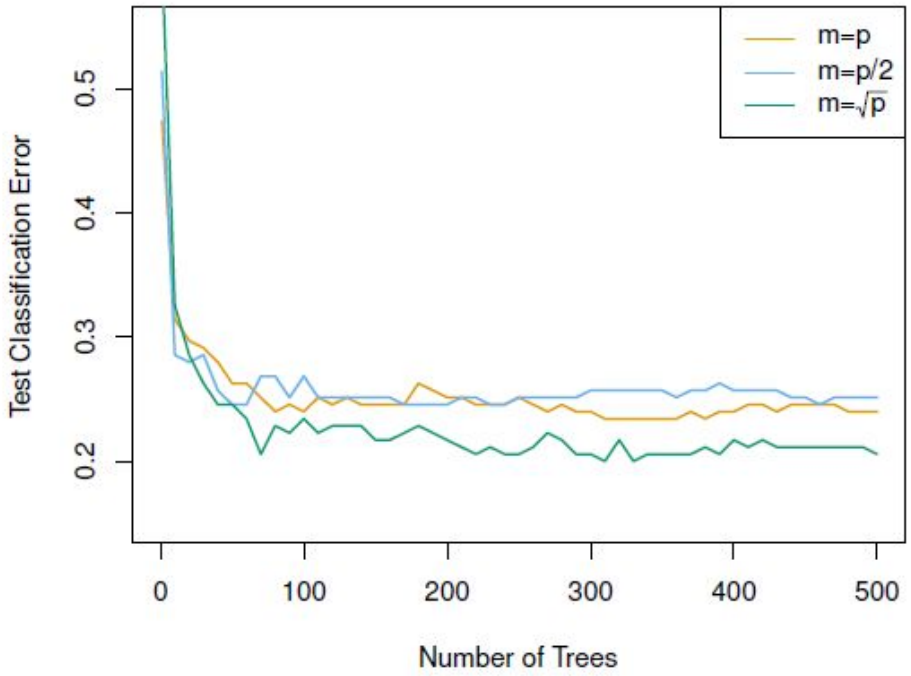


FIGURE 8.10. *Results from random forests for the 15-class gene expression data set with $p = 500$ predictors. The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of m, the number of predictors available for splitting at each interior tree node. Random forests ($m < p$) lead to a slight improvement over bagging ($m = p$). A single classification tree has an error rate of 45.7%.*

Back to top.

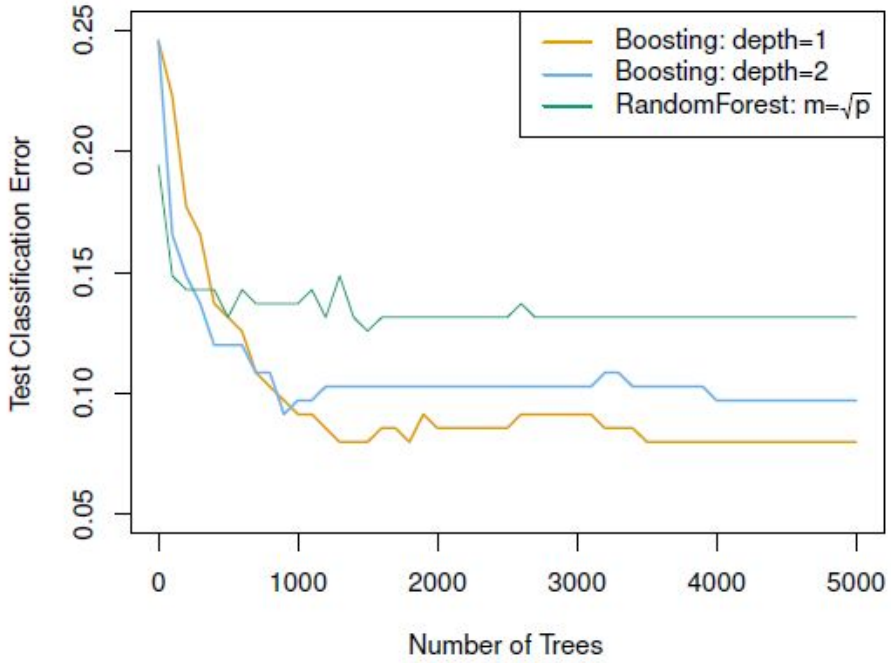


FIGURE 8.11. *Results from performing boosting and random forests on the 15-class gene expression data set in order to predict* cancer *versus* normal. *The test error is displayed as a function of the number of trees. For the two boosted models,* $\lambda = 0.01$. *Depth-1 trees slightly outperform depth-2 trees, and both outperform the random forest, although the standard errors are around 0.02, making none of these differences significant. The test error rate for a single tree is 24 %.*

Back to top.

IV. **Bayesian Additive Regression Tree (BART)**.

The last topic in this chapter is a more advanced version of random forests (RF) called **Bayesian Additive Regression Tree (BART)**. For simplicity, we present the regressor version of BART, which means this version of BART is for predicting continuous variables.

Recall that bagging and random forests make predictions from an average of regression trees, each of which is built using a random sample of data and/or predictors. Each tree is built separately from the others. By contrast, boosting uses a weighted sum of trees, each of which is constructed by fitting a tree to the residual of the current fit. Thus, each new tree attempts to capture signal that is not yet accounted for by the current set of trees. BART is related to both approaches: each tree is constructed in a random manner as in bagging and random forests, and each tree tries to capture signal not yet accounted for by the current model, as in boosting. The main novelty in BART is the way in which new trees are generated.

Suppose there are K number of regression trees. There are B iterations that BART executes on. The notation $\hat{f}_k^b(x)$ the prediction at x for which the kth regression tree used in bth iteration. At the end of each iteration, the K trees will be summed

$$\hat{f}^b(x) = \sum_{k=1}^{K} \hat{f}_k^b(x)$$

Back to top.

IV. **Bayesian Additive Regression Tree (BART)**.

Bayesian Additive Regression Tree (BART). Suppose there are K number of regression trees. There are B iterations that BART executes on. The notation $\hat{f}_k^b(x)$ the prediction at x for which the kth regression tree used in b^{th} iteration. At the end of each iteration, the K trees will be summed

$$\hat{f}^b(x) = \sum_{k=1}^{K} \hat{f}_k^b(x)$$

In the first iteration, all trees are initialized to have a single root node, with

$$\hat{f}_k^1(x) = \frac{1}{nK} \sum_{i=1}^{n} y_i$$

which is the mean response values divided by the total number of trees. Of course, these y values are the true outcomes in the training data.

In subsequent iterations, BART updates each of the K trees, one at a time. In the bth iteration, to update the kth tree, we subtract from each response value the predictions from all but the kth tree, in order to obtain partial residual

$$r_i = y_i - \sum_{k'<k} \hat{f}_{k'}^b(x_i) - \sum_{k'>k} \hat{f}_{k'}^{b-1}(x_i)$$

Back to top.

Algorithm 8.3 *Bayesian Additive Regression Trees*

1. Let $\hat{f}_1^1(x) = \hat{f}_2^1(x) = \cdots = \hat{f}_K^1(x) = \frac{1}{nK}\sum_{i=1}^{n} y_i$.
2. Compute $\hat{f}^1(x) = \sum_{k=1}^{K} \hat{f}_k^1(x) = \frac{1}{n}\sum_{i=1}^{n} y_i$.
3. For $b = 2, \ldots, B$:
   (a) For $k = 1, 2, \ldots, K$:
      i. For $i = 1, \ldots, n$, compute the current partial residual
      $$r_i = y_i - \sum_{k'<k} \hat{f}_{k'}^b(x_i) - \sum_{k'>k} \hat{f}_{k'}^{b-1}(x_i).$$
      ii. Fit a new tree, $\hat{f}_k^b(x)$, to r_i, by randomly perturbing the kth tree from the previous iteration, $\hat{f}_k^{b-1}(x)$. Perturbations that improve the fit are favored.
   (b) Compute $\hat{f}^b(x) = \sum_{k=1}^{K} \hat{f}_k^b(x)$.
4. Compute the mean after L burn-in samples,
$$\hat{f}(x) = \frac{1}{B-L}\sum_{b=L+1}^{B} \hat{f}^b(x).$$

Back to top.

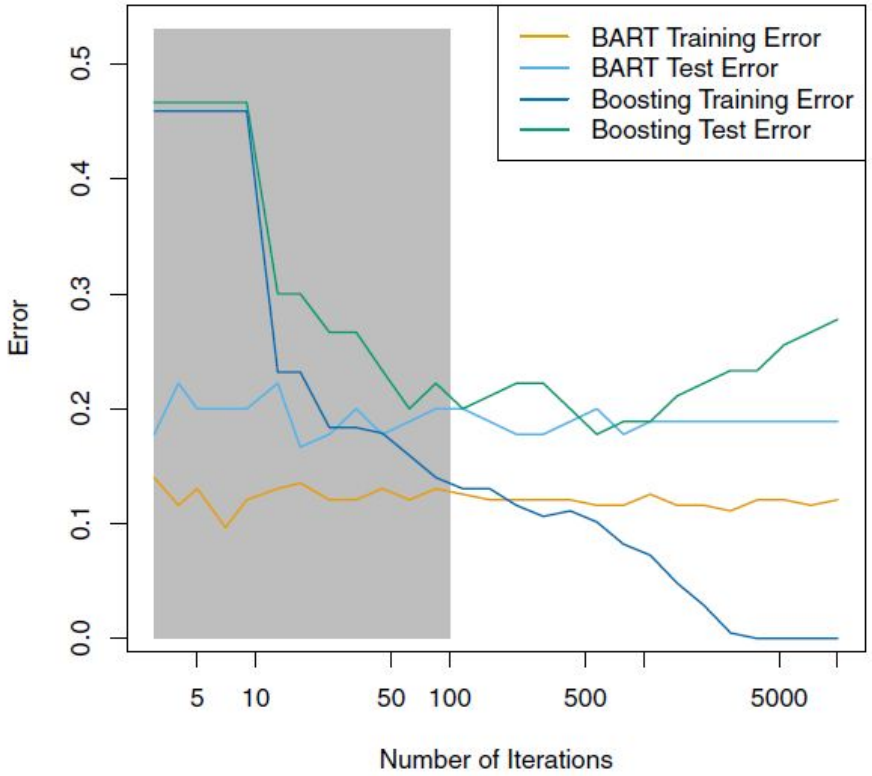


FIGURE 8.13. *BART and boosting results for the* `Heart` *data. Both training and test errors are displayed. After a burn-in period of* 100 *iterations (shown in gray), the error rates for BART settle down. Boosting begins to overfit after a few hundred iterations.*

Back to top.

Summary:

Trees are an attractive choice of weak learner for an ensemble method for a number of reasons, including their flexibility and ability to handle predictors of mixed types (i.e. qualitative as well as quantitative). We have now seen four approaches for fitting an ensemble of trees: bagging, random forests, boosting, and BART.

- In *bagging*, the trees are grown independently on random samples of the observations. Consequently, the trees tend to be quite similar to each other. Thus, bagging can get caught in local optima and can fail to thoroughly explore the model space.

- In *random forests*, the trees are once again grown independently on random samples of the observations. However, each split on each tree is performed using a random subset of the features, thereby decorrelating the trees, and leading to a more thorough exploration of model space relative to bagging.

- In *boosting*, we only use the original data, and do not draw any random samples. The trees are grown successively, using a "slow" learning approach: each new tree is fit to the signal that is left over from the earlier trees, and shrunken down before it is used.

- In *BART*, we once again only make use of the original data, and we grow the trees successively. However, each tree is perturbed in order to avoid local minima and achieve a more thorough exploration of the model space.

Back to top.

Chapter 9 - Support Vector Machine

Go back to main content, click here

Back to top.

https://www.youtube.com/YiqiaoYin

This chapter introduces **Support Vector Machine (SVM)**. It gained popularity since the 1990s and has been well known by the literature ever since. The SVM is actually a generalized form of a simple and intuitive model called **maximal margin classifier**. Hence, we will first start by introducing the basic idea of maximal margin classifier.

Back to top.

This chapter introduces **Support Vector Machine (SVM)**. It gained popularity since the 1990s and has been well known by the literature ever since. The SVM is actually a generalized form of a simple and intuitive model called **maximal margin classifier**. Hence, we will first start by introducing the basic idea of maximal margin classifier.

First thing to introduce: hyperplane.

Hyperplane is a flat affine subspace of dimension p-1 in a p-dimensional space. For example, in a 3-dimensional space, a p-1 (which is 3-1=2) dimensional affine subspace would be a plane.

Back to top.

This chapter introduces **Support Vector Machine (SVM)**. It gained popularity since the 1990s and has been well known by the literature ever since. The SVM is actually a generalized form of a simple and intuitive model called **maximal margin classifier**. Hence, we will first start by introducing the basic idea of maximal margin classifier.

First thing to introduce: hyperplane. In p-dimensional setup, we can write

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

Hyperplane is a flat affine subspace of dimension p-1 in a p-dimensional space. For example, in a 3-dimensional space, a p-1 (which is 3-1=2) dimensional affine subspace would be a plane.

For example, we can write a simple formula $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$ and we say that the pair (X1, X2) holds a point in this hyperplane.

What does it look like?

Back to top.

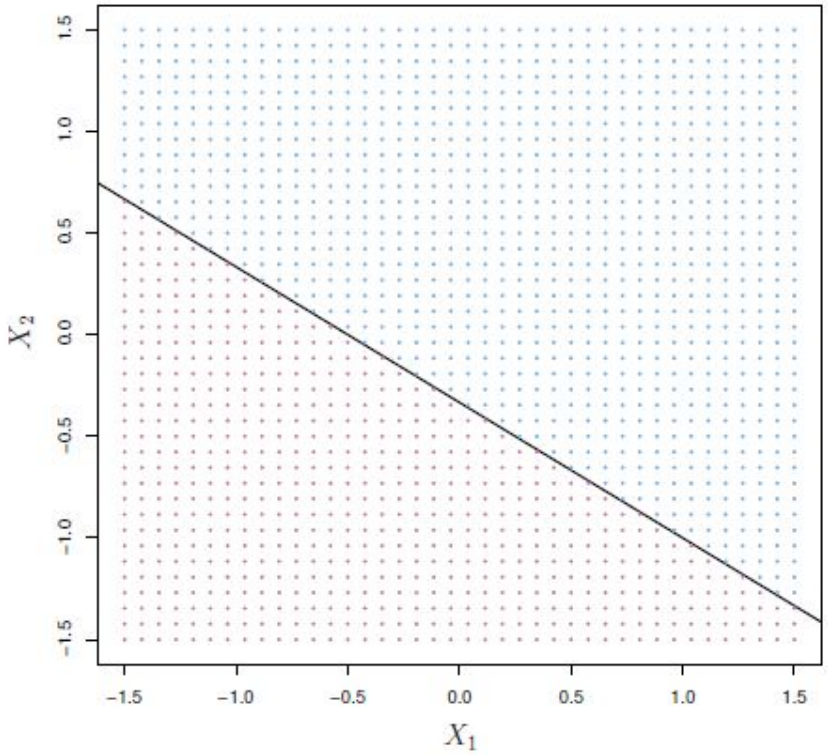


FIGURE 9.1. *The hyperplane* $1 + 2X_1 + 3X_2 = 0$ *is shown. The blue region is the set of points for which* $1 + 2X_1 + 3X_2 > 0$, *and the purple region is the set of points for which* $1 + 2X_1 + 3X_2 < 0$.

Back to top.

In p-dimensional setup, we can write

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

This means that this hyperplane can help us separate the plane into two segments. For the points that (if plugged into the equation) produce results greater than 0, these points can be in one class.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0$$

For the points produce results less than 0, these points can be another class.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0$$

This is the basic idea behind maximal margin classifier!

Back to top.

Now suppose we have data

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

and suppose that all data falls into one of the two classes $y_1, y_2, ..., y_n \in \{-1, 1\}$

To construct maximal margin classifier based on $x_1, x_2, ..., x_n \in \mathbb{R}^p$

We essentially want to solve the following optimization problem

$$\underset{\beta_0, \beta_1, \ldots, \beta_p, M}{\text{maximize}} \; M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \;\; \forall \, i = 1, \ldots, n.$$

Idea: the optimization problem seeks to maximize M (the distance between data and the classifier) and we essentially control all the betas in order to maximize this distance.

Back to top.

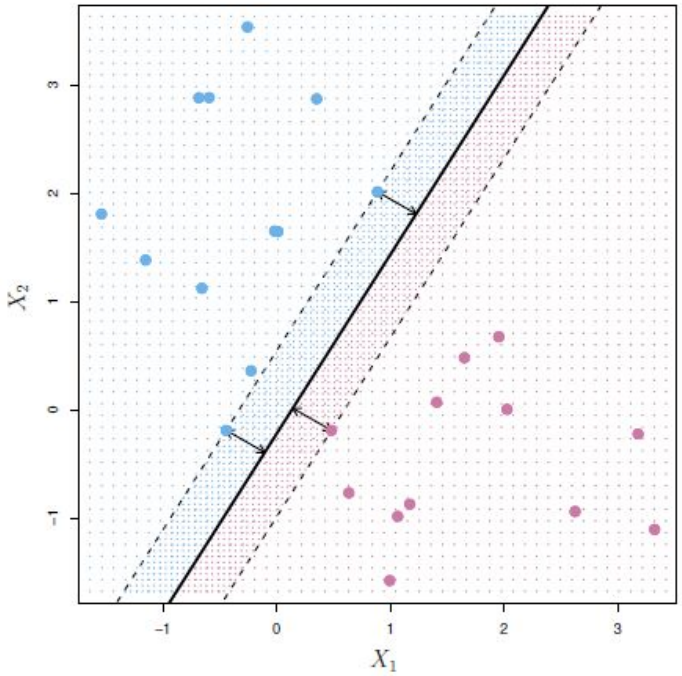


FIGURE 9.3. *There are two classes of observations, shown in blue and in purple. The maximal margin hyperplane is shown as a solid line. The margin is the distance from the solid line to either of the dashed lines. The two blue points and the purple point that lie on the dashed lines are the support vectors, and the distance from those points to the hyperplane is indicated by arrows. The purple and blue grid indicates the decision rule made by a classifier based on this separating hyperplane.*

Back to top.

With the basic understanding of maximal margin classifier, we can formally introduce SVM below

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,$$

Back to top.

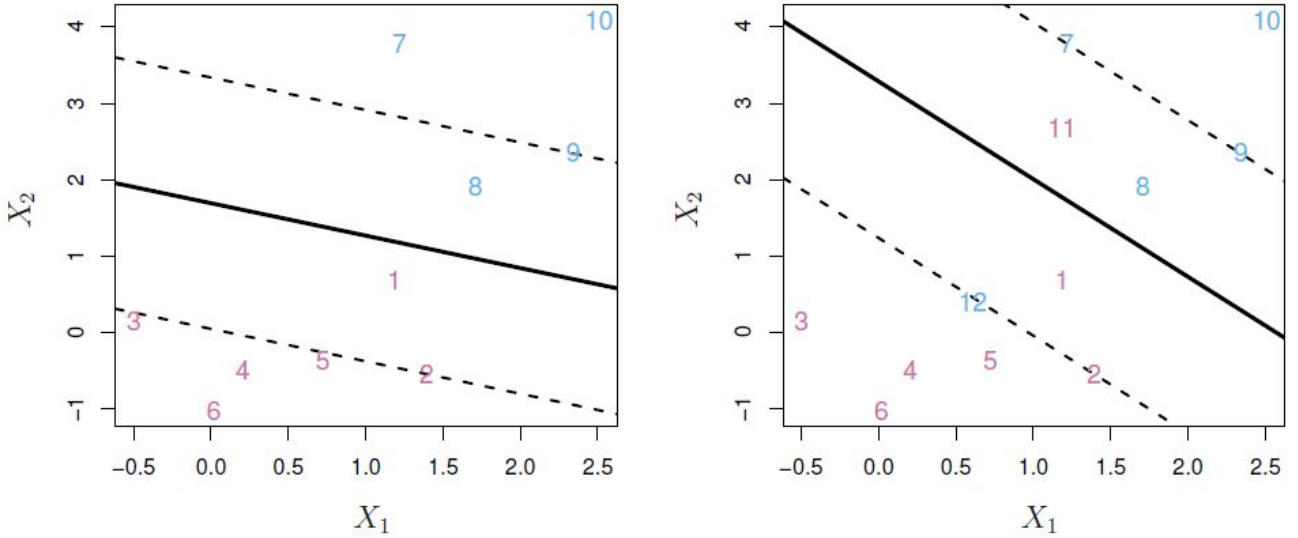


FIGURE 9.6. Left: *A support vector classifier was fit to a small data set. The hyperplane is shown as a solid line and the margins are shown as dashed lines.* Purple observations: *Observations* $3, 4, 5$, *and* 6 *are on the correct side of the margin, observation* 2 *is on the margin, and observation* 1 *is on the wrong side of the margin.* Blue observations: *Observations* 7 *and* 10 *are on the correct side of the margin, observation* 9 *is on the margin, and observation* 8 *is on the wrong side of the margin. No observations are on the wrong side of the hyperplane.* Right: *Same as left panel with two additional points,* 11 *and* 12. *These two observations are on the wrong side of the hyperplane and the wrong side of the margin.*

Back to top.

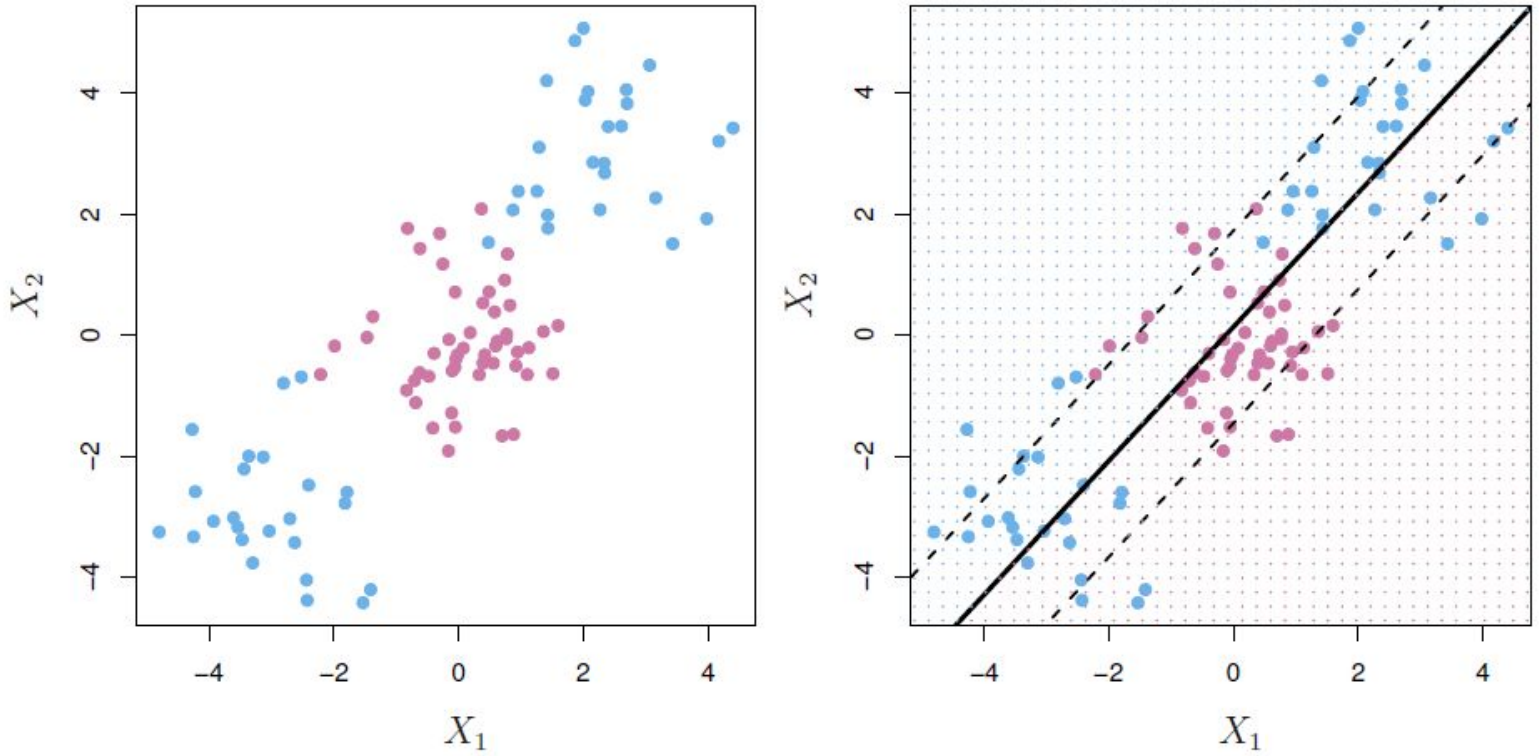


FIGURE 9.8. Left: *The observations fall into two classes, with a non-linear boundary between them.* Right: *The support vector classifier seeks a linear boundary, and consequently performs very poorly.*

Back to top.

To solve the above nonlinear boundary problem, one way is to use higher order predictors. Instead of using

$$X_1, X_2, ..., X_p$$

we can use

$$X_1, X_1^2, X_2, X_2^2 ..., X_p, X_p^2$$

Then, we can update our optimization problem to be the following

$$\underset{\beta_0,\beta_{11},\beta_{12},...,\beta_{p1},\beta_{p2},\epsilon_1,...,\epsilon_n,M}{\text{maximize}} \quad M$$

$$\text{subject to } y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i)$$

$$\sum_{i=1}^{n} \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1.$$

Additionally, there are also other tuning parameters such as changing kernels (polynomial kernel, radial based kernels, etc.) These methods can help us improve the performance of a SVM.

Back to top.

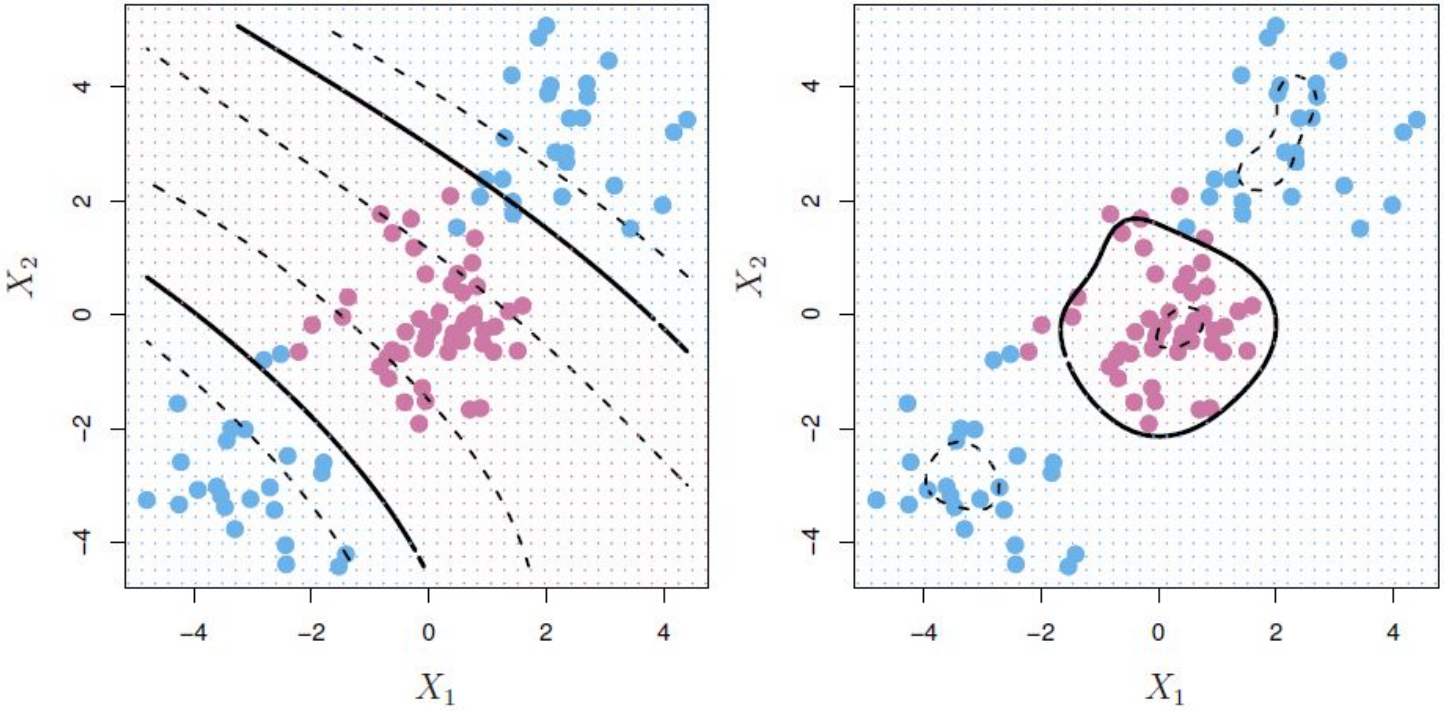


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

Back to top.

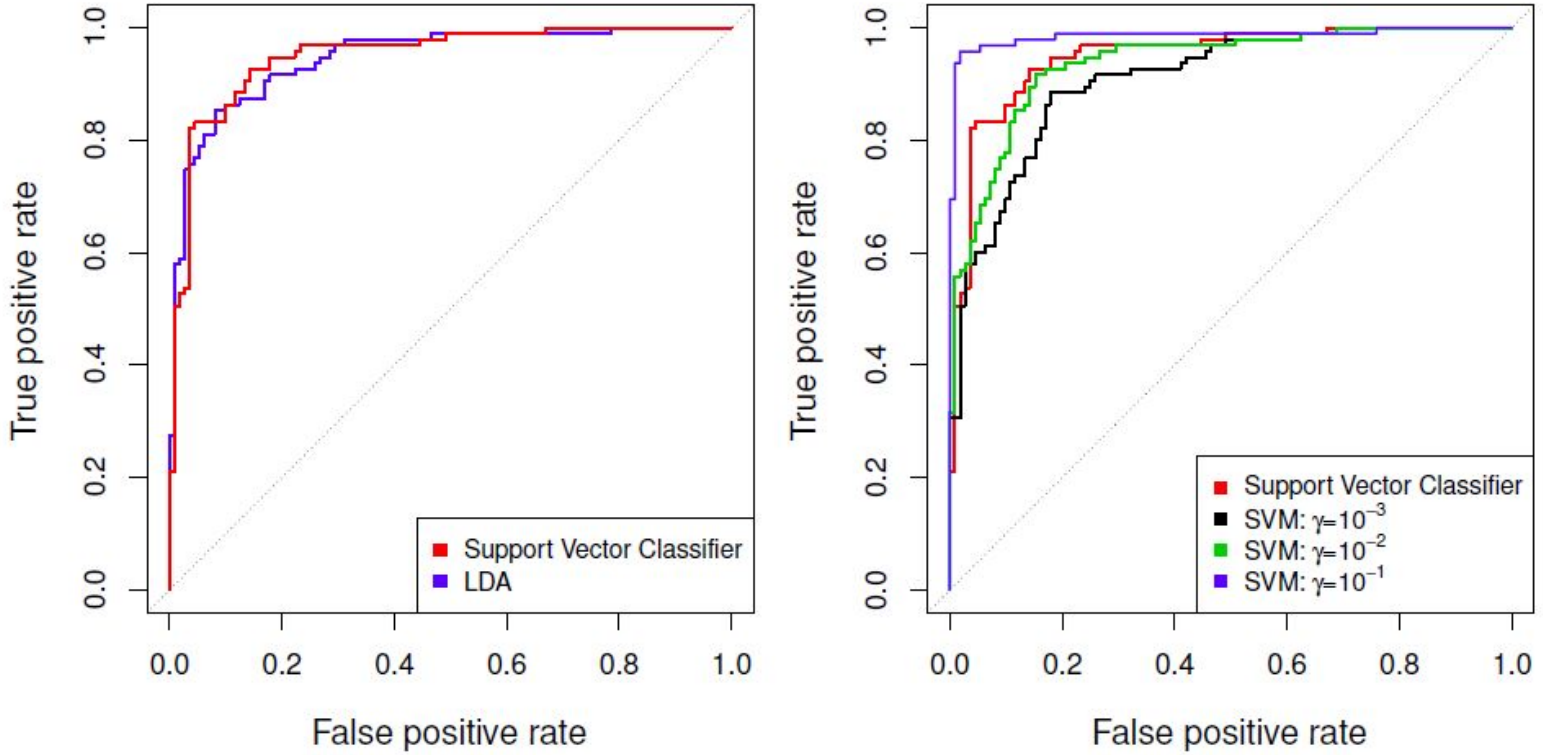


FIGURE 9.10. *ROC curves for the* Heart *data training set.* Left: *The support vector classifier and LDA are compared.* Right: *The support vector classifier is compared to an SVM using a radial basis kernel with* $\gamma = 10^{-3}$, 10^{-2}, *and* 10^{-1}.

Back to top.

Relationship to Logistic Regression

SVM made a big impact in early 1990s due to its mysterious, novelty, and good prediction performance. However, it is not completely irrelevant from linear or logistic regression models.

To see this deeper connection, let us rewrite the optimization problem in SVM in the following way

$$\min_{\beta_0,\beta_1,\ldots,\beta_p} = \left\{ \sum_{i=1}^{n} \max[0, 1-y_i f(x_i)] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

Recall linear regression model with regularization that takes the form of “loss + penalty”, we can write the following generation form

$$\min_{\beta_0,\beta_1,\ldots,\beta_p} = \left\{ L(\mathbf{X}, \mathbf{y}, \beta) + \lambda P(\beta) \right\}$$

where sometimes the loss term is written as

$$L(\mathbf{X}, \mathbf{y}, \beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{n} \beta_j x_{ij} \right)^2$$

which is essentially the form of what we used for ridge or lasso regression. However, the difference here in SVM is that loss is rewritten as

$$L(\mathbf{X}, \mathbf{y}, \beta) = \sum_{i=1}^{n} \max[0, 1 - y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})]$$

Back to top.

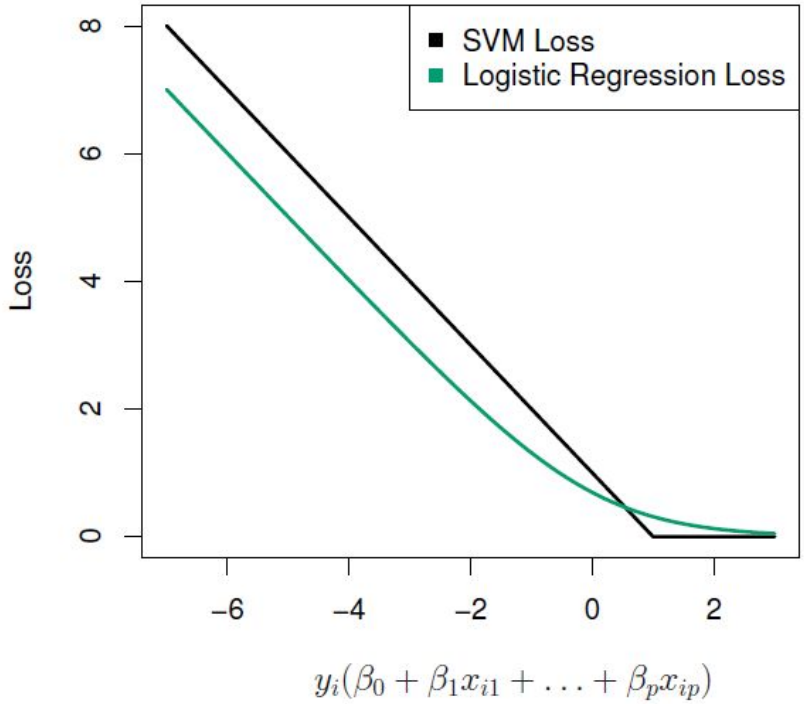


FIGURE 9.12. *The SVM and logistic regression loss functions are compared, as a function of* $y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})$. *When* $y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})$ *is greater than 1, then the SVM loss is zero, since this corresponds to an observation that is on the correct side of the margin. Overall, the two loss functions have quite similar behavior.*

Back to top.

Chapter 10 - Deep Learning

Go back to main content, click here

Back to top.

One of the most important milestone in late 80s is deep learning! I created this lecture series in 2021 Christmas and I have already been in the field for 6+ years. That is nothing comparing with the entire development of the field since the 1980s.

Famous topic we are introducing here are: **Artificial Neural Network (ANN)**, **Convolutional Neural Network (CNN)**, and **Recurrent Neural Network (RNN)**. Famous deep learning libraries are Keras (from Google Brain) and PyTorch (from Facebook AI Research, FAIR). We introduce these topics in the following order

- **Artificial Neural Network (ANN)**
- **Convolutional Neural Network (CNN)**
- **Recurrent Neural Network**

A neural network takes an input vector of p variables $X = (X_1, \ldots, X_p)$ and builds a nonlinear function $f(X)$ to predict response variable Y. A simple feed-forward neural network is shown below.

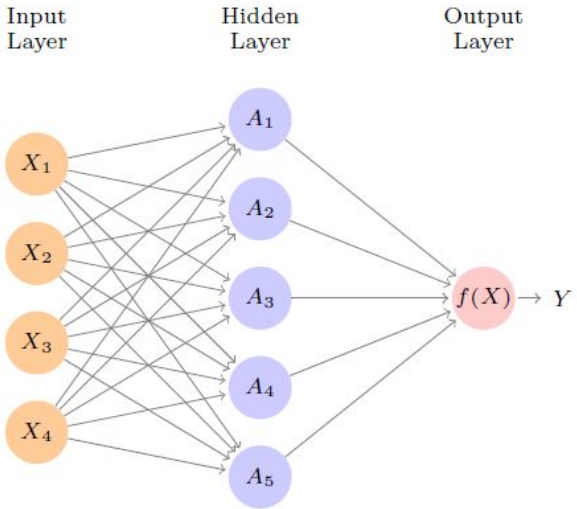


FIGURE 10.1. *Neural network with a single hidden layer. The hidden layer computes activations $A_k = h_k(X)$ that are nonlinear transformations of linear combinations of the inputs $X_1, X_2, \ldots, X_p$. Hence these A_k are not directly observed. The functions $h_k(\cdot)$ are not fixed in advance, but are learned during the training of the network. The output layer is a linear model that uses these activations A_k as inputs, resulting in a function $f(X)$.*

Back to top.

How do we describe the above figure, a simple **feed-forward neural network**, mathematically?

In the graph, there are 4 input variables. Hence, we set p to be 4 and there are 4 features: X1, X2, X3, and X4. These 4 features make up the **input layer**. The **arrows** indicate that each of the inputs from the input layer feeds into each of the K **hidden units**. This is called **fully-connected**. The model takes the following form

$$\begin{aligned} f(X) &= \beta_0 + \textstyle\sum_{k=1}^{K} \beta_k h_k(X) \\ &= \beta_0 + \textstyle\sum_{k=1}^{K} \beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j). \end{aligned}$$

To build the model, there are two steps. First, we take K activations in the hidden layer and it is computed as functions of the input features X1, ..., Xp. The activation function is

$$A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j)$$

where g(z) is nonlinear **activation function**. With K activations, we can then model the function f(X) and the resulting formula is

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k X_k$$

where all the parameters $\beta_0, \ldots, \beta_p, w_{10}, \ldots, w_{Kp}$ need to be estimated and trained.

A famous activation function is **sigmoid function** and it takes the following form

$$g(z) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}$$

which is the same function used in logistic regression model. The purpose is to convert the output of the unit or neuron from real line into a probability (a numerical number between 0 and 1).

Another famous activation function is **ReLU** or **Rectified Linear Unit**. The ReLU takes the following form

$$g(z) = (z)_+ = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{otherwise.} \end{cases}$$

which is a more efficient function than sigmoid function.

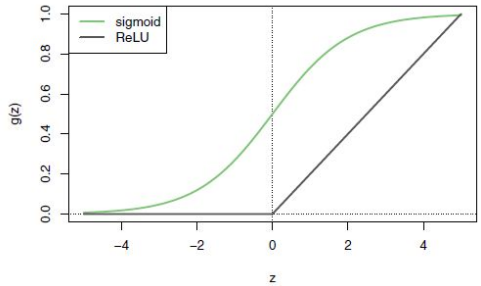


FIGURE 10.2. *Activation functions. The piecewise-linear* `ReLU` *function is popular for its efficiency and computability. We have scaled it down by a factor of five for ease of comparison.*

Back to top.

A famous activation function is **sigmoid function** and it takes the following form

$$g(z) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}$$

which is the same function used in logistic regression model. The purpose is to convert the output of the unit or neuron from real line into a probability (a numerical number between 0 and 1).

Another famous activation function is **ReLU** or **Rectified Linear Unit**. The ReLU takes the following form

$$g(z) = (z)_+ = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{otherwise.} \end{cases}$$

which is a more efficient function than sigmoid function.

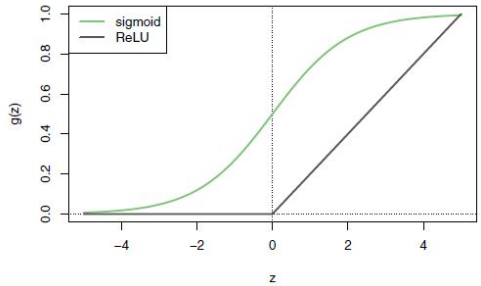


FIGURE 10.2. *Activation functions. The piecewise-linear* ReLU *function is popular for its efficiency and computability. We have scaled it down by a factor of five for ease of comparison.*

In simple words, the model depicted in the above figure derives five new features by computing five different linear combinations of X, and then squashes each through an activation function g(·) to transform it. The final model is linear in these derived variables.

Back to top.

To train this neural network, the last piece of puzzle is the loss function. We can always stick to the original version: least square. The squared-error or squared-loss takes the following form

$$\sum_{i=1}^{n}(y_i - f(x_i))^2$$

Back to top.

What happens when the data is more complicated? For example, instead of tabular data, we could have image data.

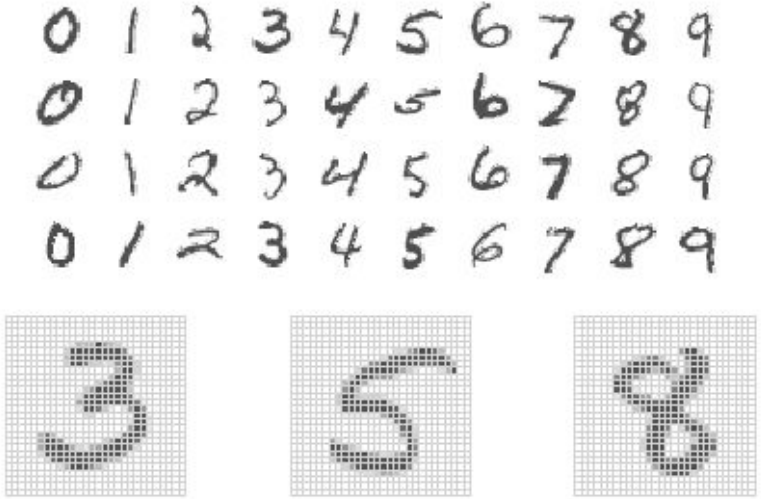


FIGURE 10.3. *Examples of handwritten digits from the* MNIST *corpus. Each grayscale image has* 28×28 *pixels, each of which is an eight-bit number (0–255) which represents how dark that pixel is. The first 3, 5, and 8 are enlarged to show their 784 individual pixel values.*

Back to top.

To adapt neural network to fit for the above image data, we need to pay attention to a few things:

- The input variables are not tabular. The input variables or input features are pixels from the images. A pixel takes greyscale from value 0 to 255 while 0 means no brightness and 255 means the highest level of brightness. Each image is sized 28 by 28, and this comes out to be 784 pixels. Hence, the input layer would have to have 784 variables.
- The output layer requires to have the same shape as the design of the experiment. This means that we need to design the output layer dependent on the requirement of the goal of this classification task. The goal is to read in an image and be able to train the model to recognize the digit. There are 10 digits: 0, 1, 2, ..., 9. This means that there are 10 classes and hence we are dealing with a 10-class classification problem. After **one-hot encoding**, there are 10 units in the output layer. We provide a small example of one-hot encoding below,

$$X = (1, 2, 3) \rightarrow X_1 = (1, 0, 0), X_2 = (0, 1, 0), X_3 = (0, 0, 1)$$

We can adapt to the new data, MNIST data set, to create the following diagram for an upgraded version of feed-forward neural network.

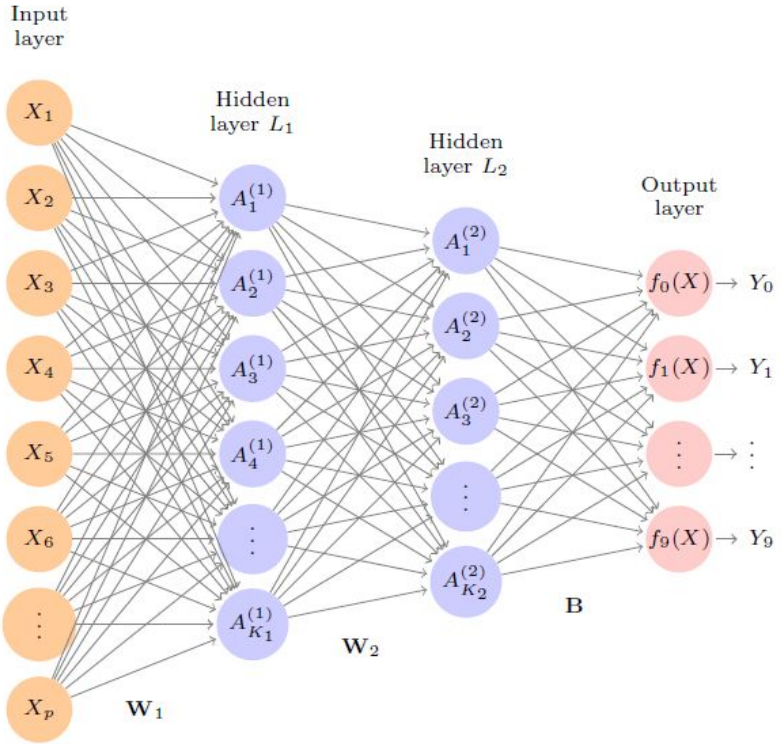


FIGURE 10.4. *Neural network diagram with two hidden layers and multiple outputs, suitable for the* MNIST *handwritten-digit problem. The input layer has* $p = 784$ *units, the two hidden layers* $K_1 = 256$ *and* $K_2 = 128$ *units respectively, and the output layer* 10 *units. Along with intercepts (referred to as* biases *in the deep-learning community) this network has 235,146 parameters (referred to as weights).*

Back to top.

Recall before

To train this neural network, the last piece of puzzle is the loss function. We can always stick to the original version: least square. The squared-error or squared-loss takes the following form

$$\sum_{i=1}^{n}(y_i - f(x_i))^2$$

Now since we have 10-class classification task, we need to upgrade our loss function to **multinomial log-likelihood** or **cross-entropy**. This loss function takes the following form

$$-\sum_{i=1}^{n}\sum_{m=0}^{9} y_{im}\log(f_m(x_i))$$

Back to top.

How do fit neural network?

Suppose we have $\beta = (\beta_0, \beta_1, ..., \beta_K)$ and $w_k = (w_{k0}, ..., w_{kp})$. These are the parameters required to design a simple feed-forward neural network introduced above. Now we need to figure out a way to estimate these parameters. Given training set observations (x_i, y_i), we would fit

$$\min_{\{w_k\}_1^K, \beta} \frac{1}{2} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

where the model f() takes the following form

$$f(x_i) = \beta_0 + \sum_{k=1}^{K} \beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj} x_{ij})$$

Back to top.

Method	Test Error
Neural Network + Ridge Regularization	2.3%
Neural Network + Dropout Regularization	1.8%
Multinomial Logistic Regression	7.2%
Linear Discriminant Analysis	12.7%

TABLE 10.1. *Test error rate on the* `MNIST` *data, for neural networks with two forms of regularization, as well as multinomial logistic regression and linear discriminant analysis. In this example, the extra complexity of the neural network leads to a marked improvement in test error.*

Back to top.

One of the most important milestone in late 80s is deep learning! I created this lecture series in 2021 Christmas and I have already been in the field for 6+ years. That is nothing comparing with the entire development of the field since the 1980s.

Famous topic we are introducing here are: **Artificial Neural Network (ANN)**, **Convolutional Neural Network (CNN)**, and **Recurrent Neural Network (RNN)**. Famous deep learning libraries are Keras (from Google Brain) and PyTorch (from Facebook AI Research, FAIR). We introduce these topics in the following order

- **Artificial Neural Network (ANN) - finished!**
- **Convolutional Neural Network (CNN) - in progress ...**
- **Recurrent Neural Network**

What if we have a data set like CIFAR100?

FIGURE 10.5. *A sample of images from the* CIFAR100 *database: a collection of natural images from everyday life, with 100 different classes represented.*

Back to top.

What if we have a data set like CIFAR100? The dataset has images size 32 by 32 by 3. The 3 means 3 channels of colors: Red, Green, Blue or short for RGB. There are 100 classes.

FIGURE 10.5. *A sample of images from the* CIFAR100 *database: a collection of natural images from everyday life, with 100 different classes represented.*

Back to top.

A special types of neural networks called **Convolutional Neural Networks (CNNs)** evolve from investigating images.

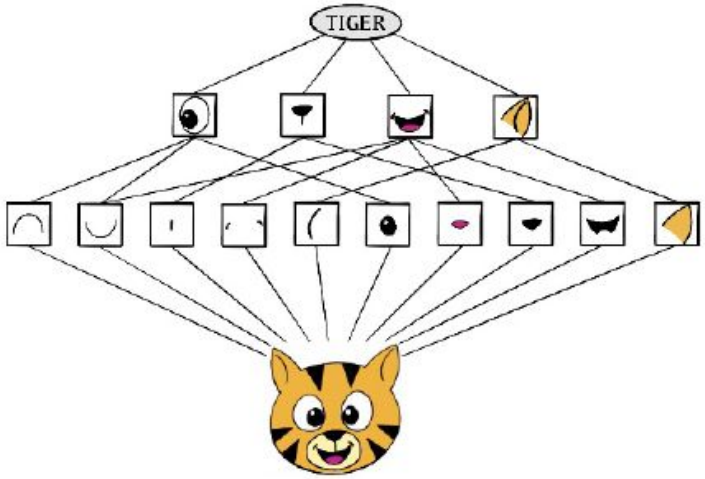


FIGURE 10.6. *Schematic showing how a convolutional neural network classifies an image of a tiger. The network takes in the image and identifies local features. It then combines the local features in order to create compound features, which in this example include eyes and ears. These compound features are used to output the label "tiger".*

Back to top.

A special types of neural networks called **Convolutional Neural Networks (CNNs)** evolve from investigating images.

The ANN relies on hidden layer to learn the internal representations of the dataset. When we have images, we need to create functions to self detect features from the images. This is where “convolution” comes in.

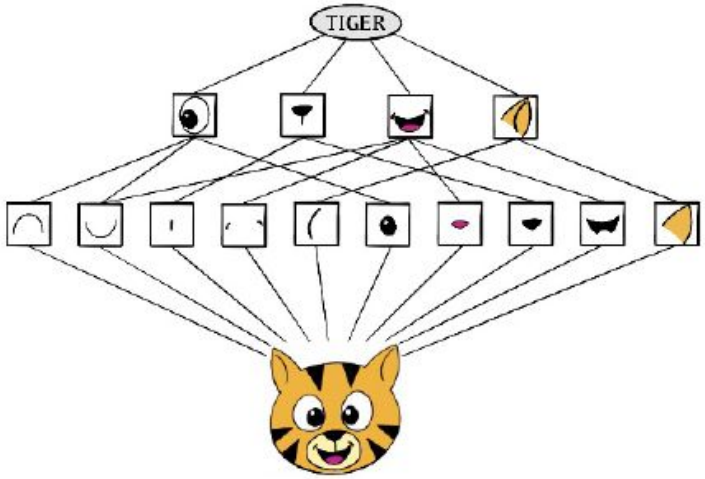


FIGURE 10.6. *Schematic showing how a convolutional neural network classifies an image of a tiger. The network takes in the image and identifies local features. It then combines the local features in order to create compound features, which in this example include eyes and ears. These compound features are used to output the label “tiger”.*

Back to top.

A convolution layer is made up of a large number of convolution filters, each of which is a template that determines whether a particular local feature is present in an image. A convolution filter relies on a very simple operation, called a convolution, which basically amounts to repeatedly multiplying matrix elements and then adding the results.

To understand how a convolution filter works, consider a very simple example of a 4×3 image:

$$\text{Original Image} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{bmatrix}.$$

Now consider a 2×2 filter of the form

$$\text{Convolution Filter} = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}.$$

When we *convolve* the image with the filter, we get the result[8]

$$\text{Convolved Image} = \begin{bmatrix} a\alpha + b\beta + d\gamma + e\delta & b\alpha + c\beta + e\gamma + f\delta \\ d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \\ g\alpha + h\beta + j\gamma + k\delta & h\alpha + i\beta + k\gamma + l\delta \end{bmatrix}.$$

Back to top.

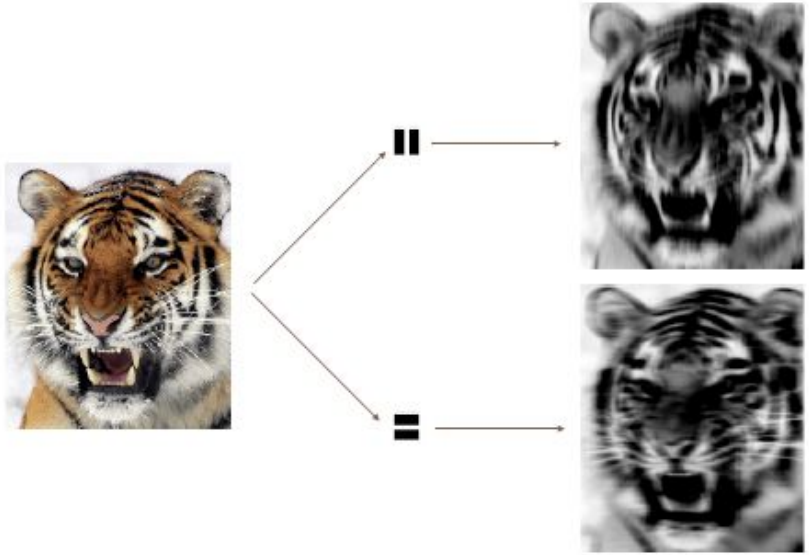


FIGURE 10.7. *Convolution filters find local features in an image, such as edges and small shapes. We begin with the image of the tiger shown on the left, and apply the two small convolution filters in the middle. The convolved images highlight areas in the original image where details similar to the filters are found. Specifically, the top convolved image highlights the tiger's vertical stripes, whereas the bottom convolved image highlights the tiger's horizontal stripes. We can think of the original image as the input layer in a convolutional neural network, and the convolved images as the units in the first hidden layer.*

Back to top.

A *pooling* layer provides a way to condense a large image into a smaller summary image. While there are a number of possible ways to perform pooling, the *max pooling* operation summarizes each non-overlapping 2×2 block of pixels in an image using the maximum value in the block. This reduces the size of the image by a factor of two in each direction, and it also provides some *location invariance*: i.e. as long as there is a large value in one of the four pixels in the block, the whole block registers as a large value in the reduced image.

Here is a simple example of max pooling:

$$\text{Max pool} \begin{bmatrix} 1 & 2 & 5 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 1 & 3 & 4 \\ 1 & 1 & 2 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix}.$$

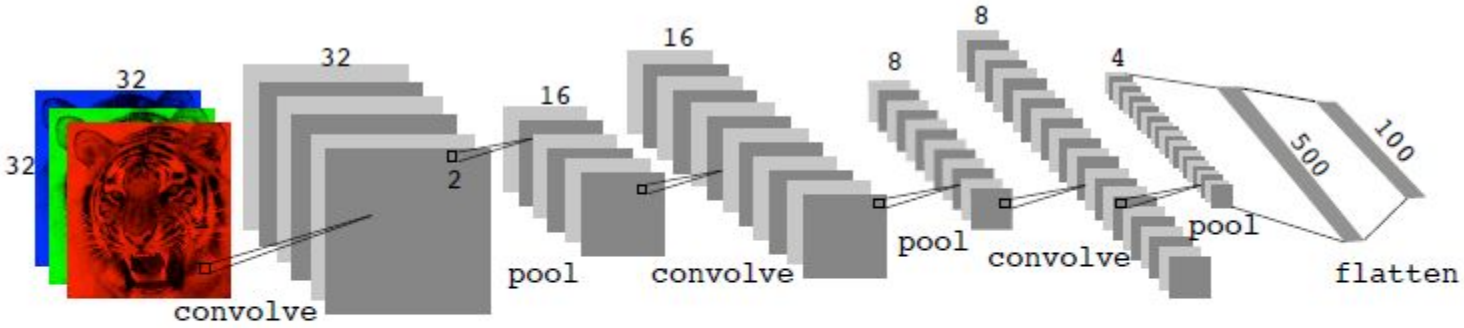


FIGURE 10.8. *Architecture of a deep CNN for the* `CIFAR100` *classification task. Convolution layers are interspersed with* 2×2 *max-pool layers, which reduce the size by a factor of* 2 *in both dimensions.*

Back to top.

This convolve-then-pool sequence is now repeated for the next two layers. Some details are as follows:

- Each subsequent convolve layer is similar to the first. It takes as input the three-dimensional feature map from the previous layer and treats it like a single multi-channel image. Each convolution filter learned has as many channels as this feature map.
- Since the channel feature maps are reduced in size after each pool layer, we usually increase the number of filters in the next convolve layer to compensate.
- Sometimes we repeat several convolve layers before a pool layer. This effectively increases the dimension of the filter.

This convolve-then-pool sequence is now repeated for the next two layers. Some details are as follows:

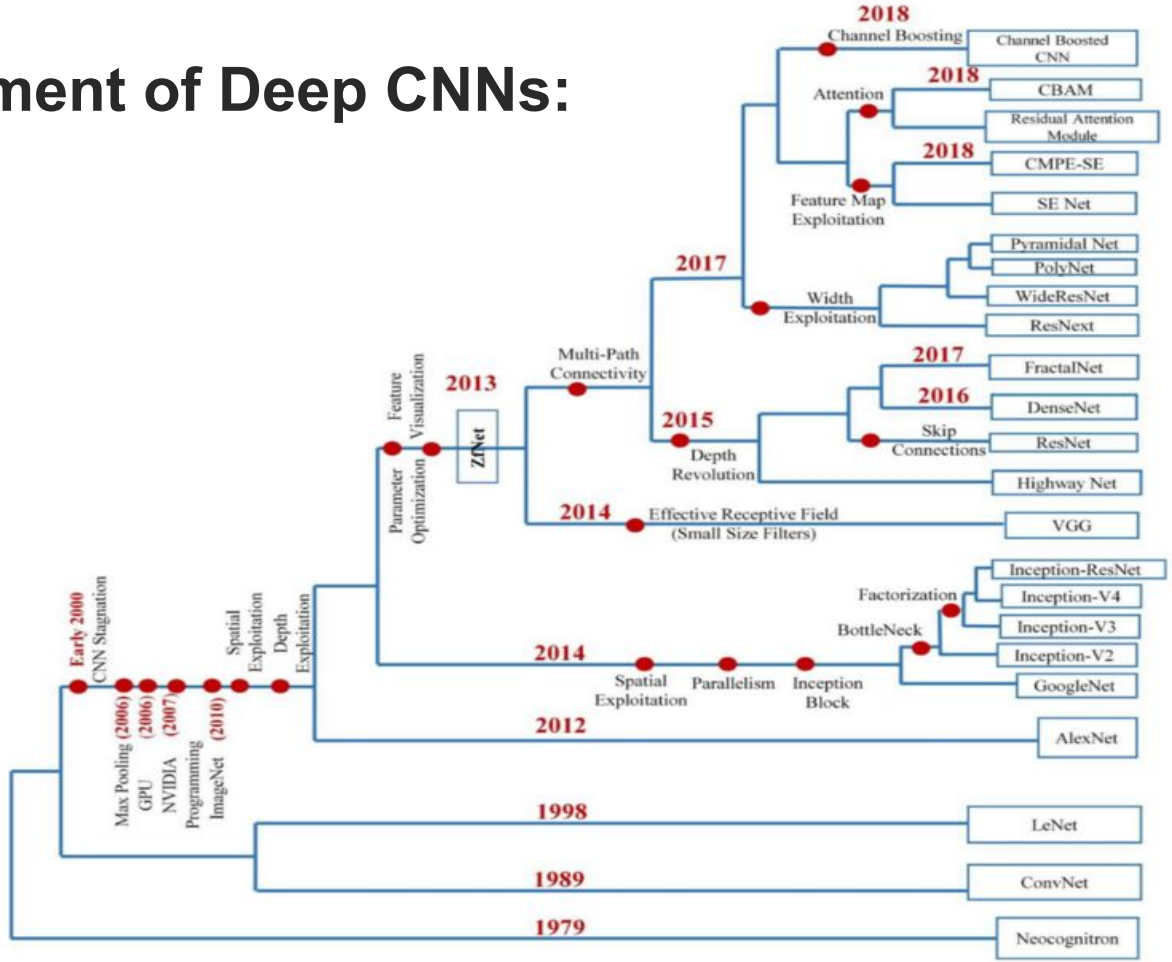
- Each subsequent convolve layer is similar to the first. It takes as input the three-dimensional feature map from the previous layer and treats it like a single multi-channel image. Each convolution filter learned has as many channels as this feature map.
- Since the channel feature maps are reduced in size after each pool layer, we usually increase the number of filters in the next convolve layer to compensate.
- Sometimes we repeat several convolve layers before a pool layer. This effectively increases the dimension of the filter.

These operations are repeated until the pooling has reduced each channel feature map down to just a few pixels in each dimension. At this point the three-dimensional feature maps are *flattened* — the pixels are treated as separate units — and fed into one or more fully-connected layers before reaching the output layer, which is a *softmax activation* for the 100 classes

FIGURE 10.9. *Data augmentation. The original image (leftmost) is distorted in natural ways to produce different images with the same class label. These distortions do not fool humans, and act as a form of regularization when fitting the CNN.*

Back to top.

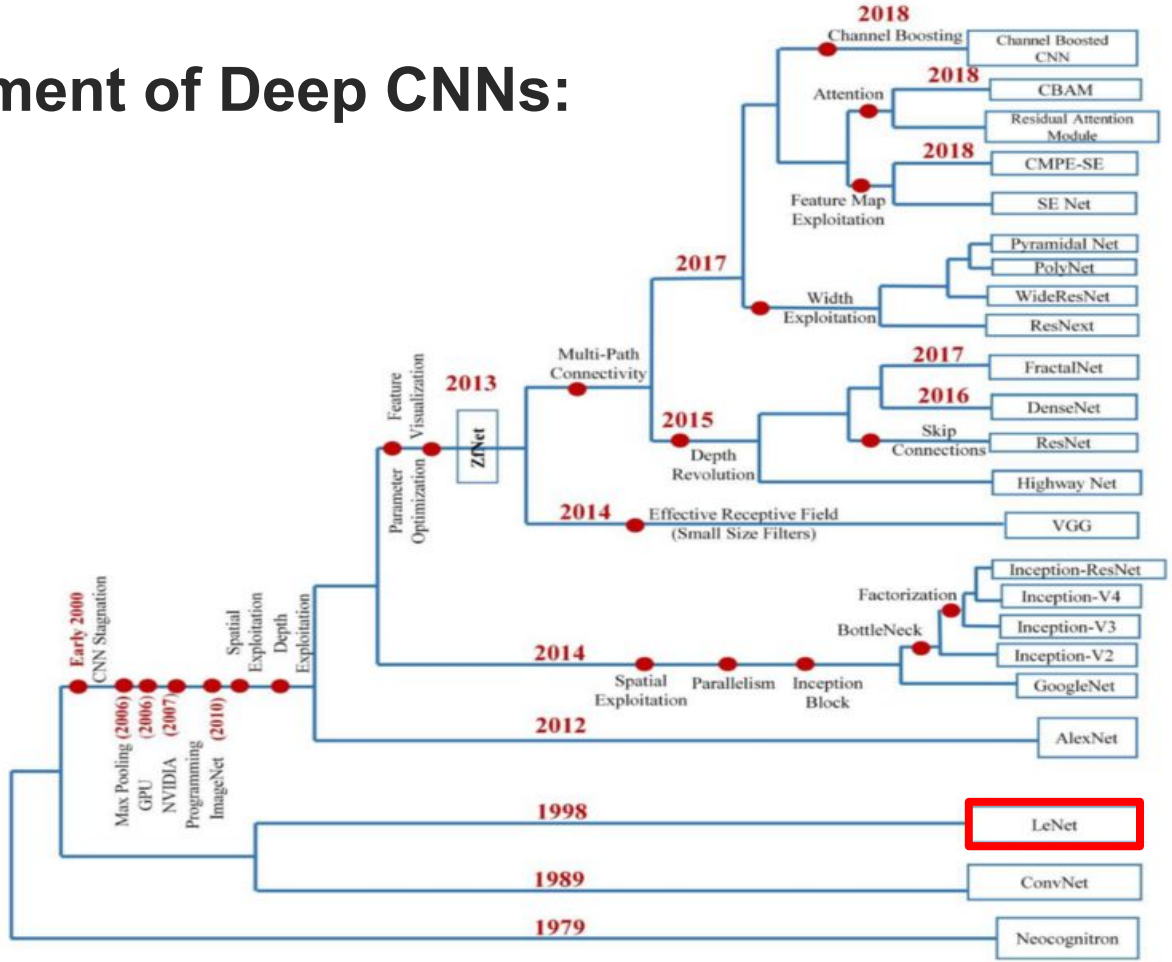
Development of Deep CNNs:



Back to top.

Development of Deep CNNs:

LeNet

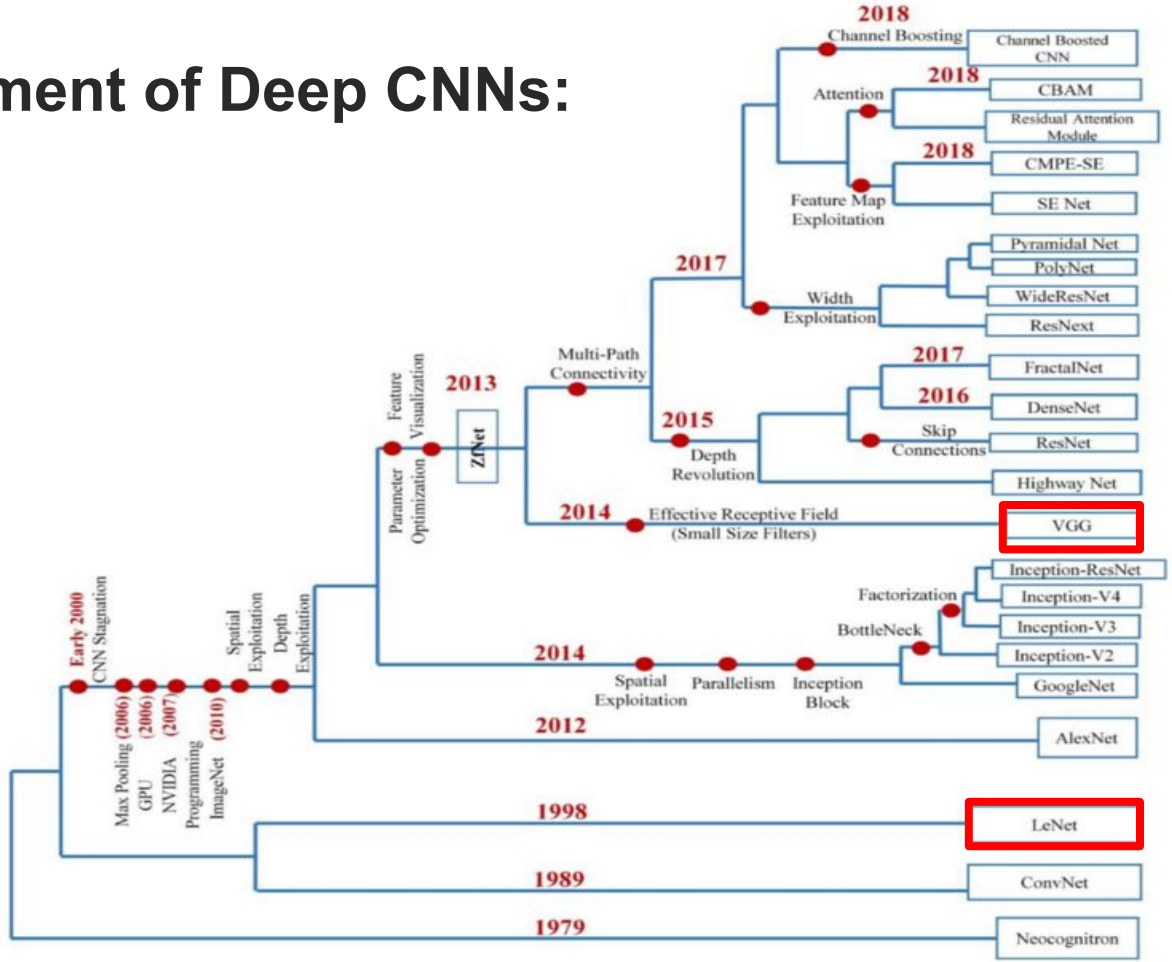


Back to top.

Development of Deep CNNs:

LeNet

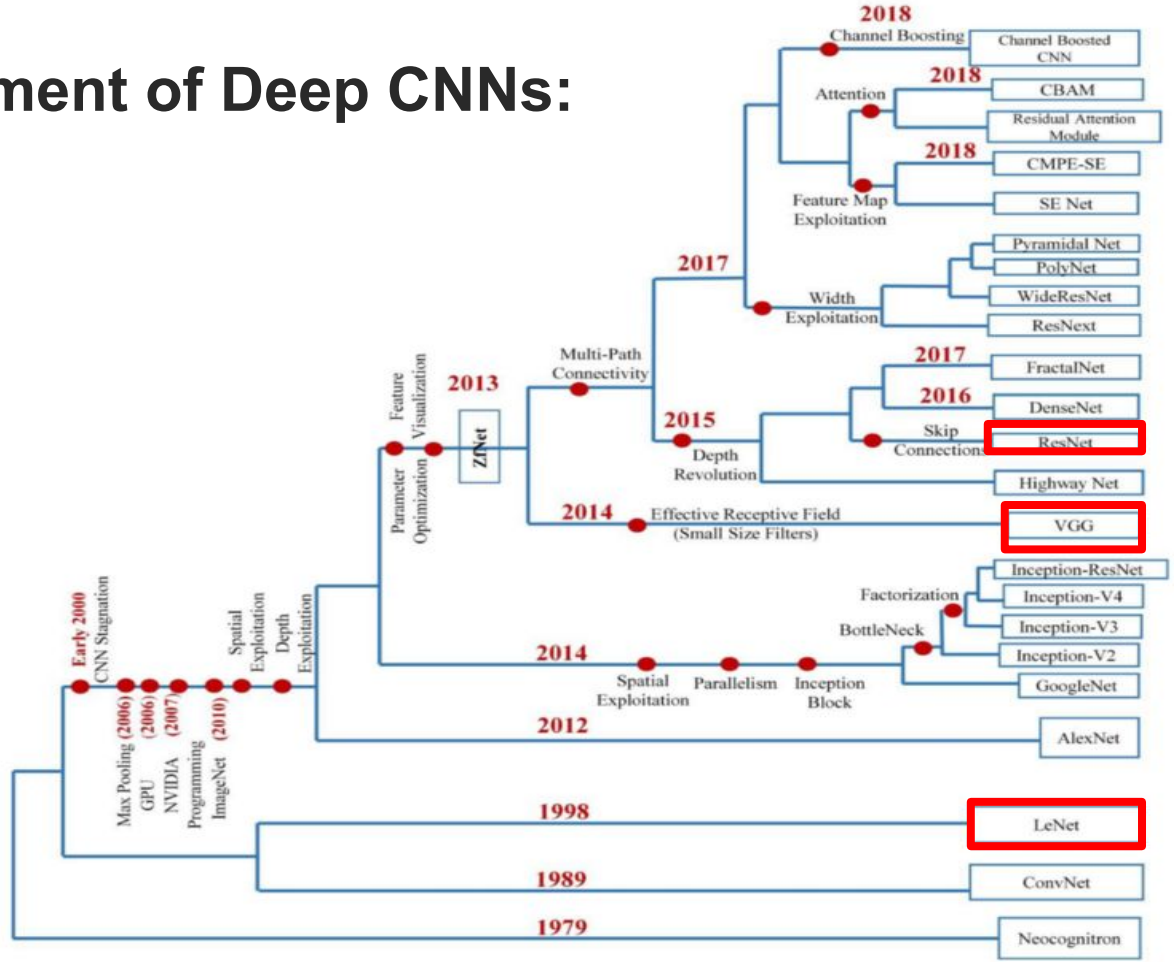
VGG



Back to top.

Development of Deep CNNs:

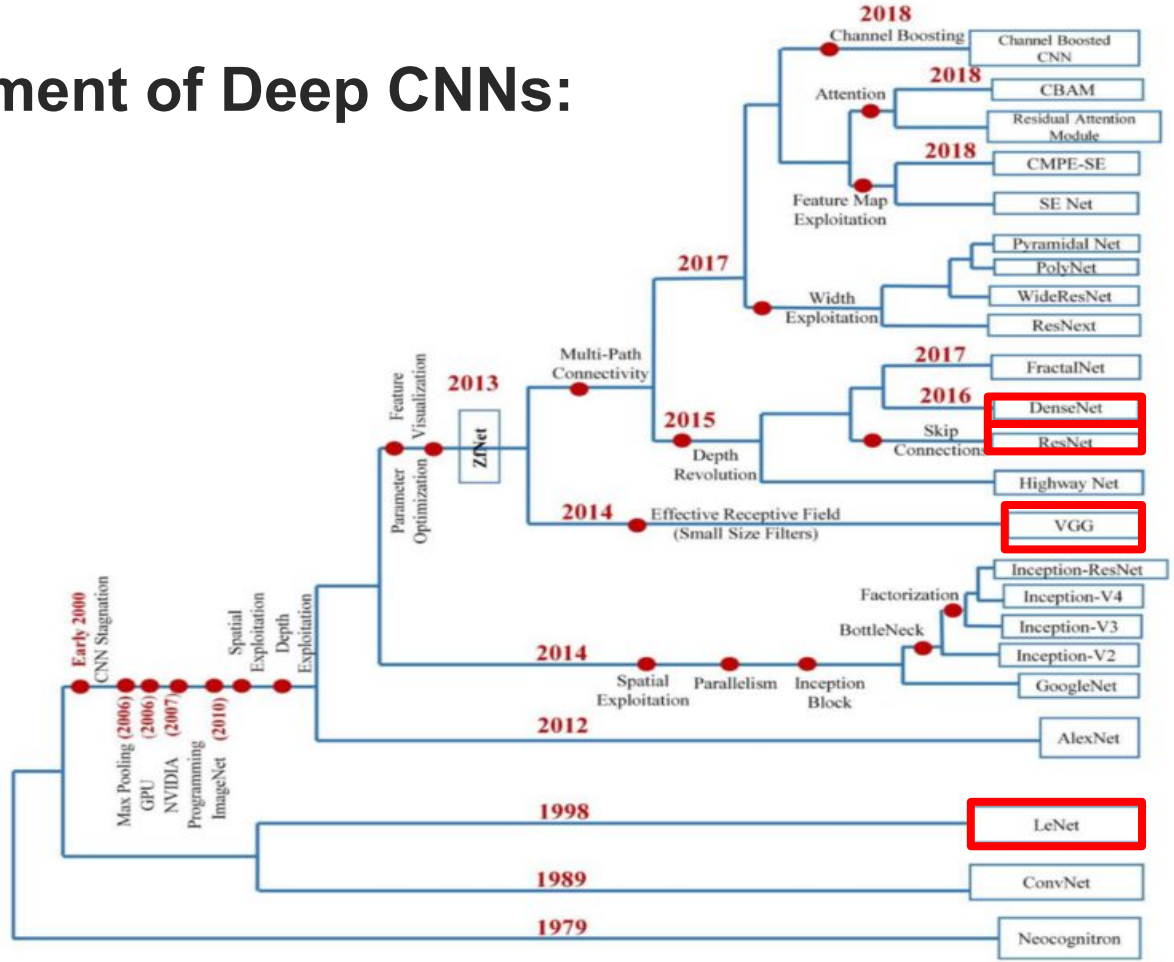
- LeNet
- VGG
- ResNet



Back to top.

Development of Deep CNNs:

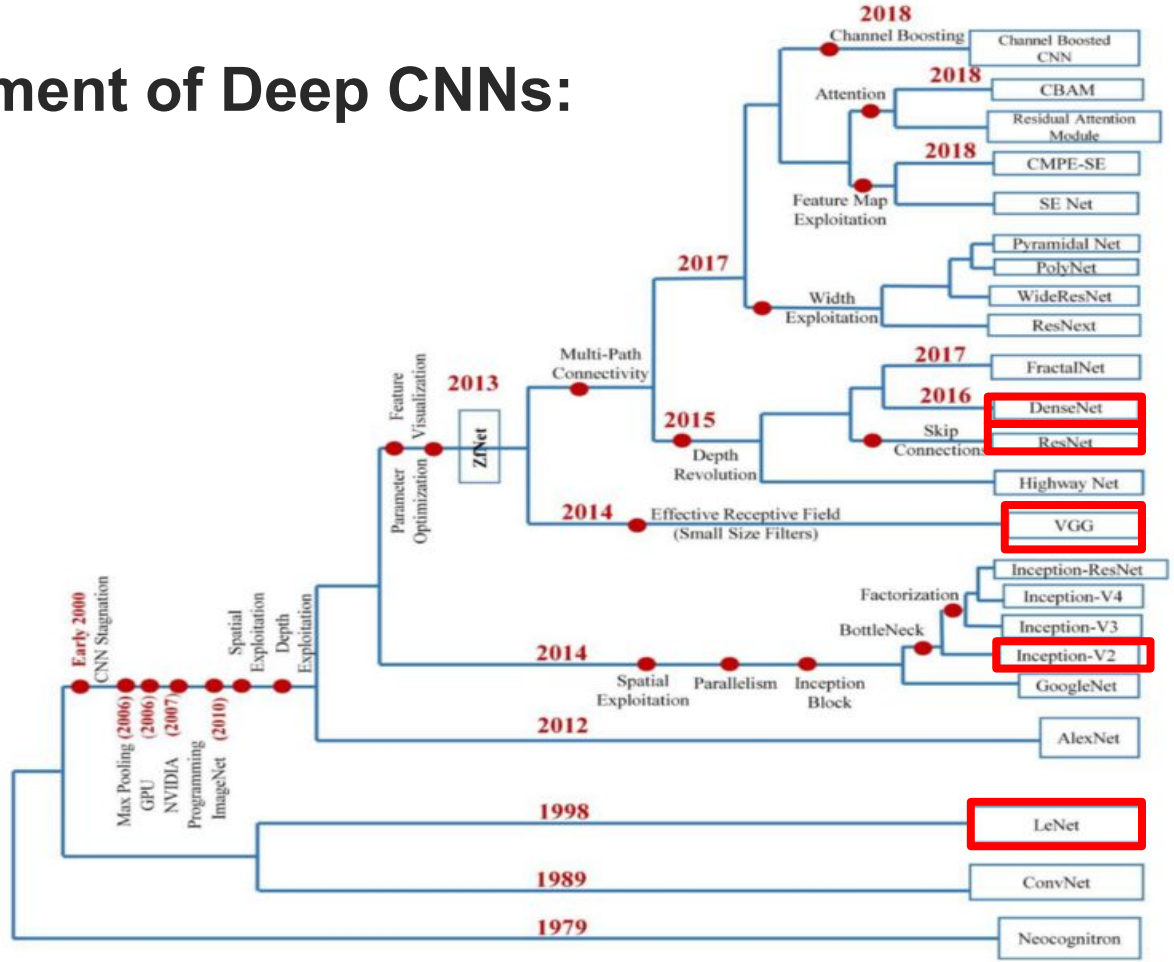
- LeNet
- VGG
- ResNet
- DenseNet



Back to top.

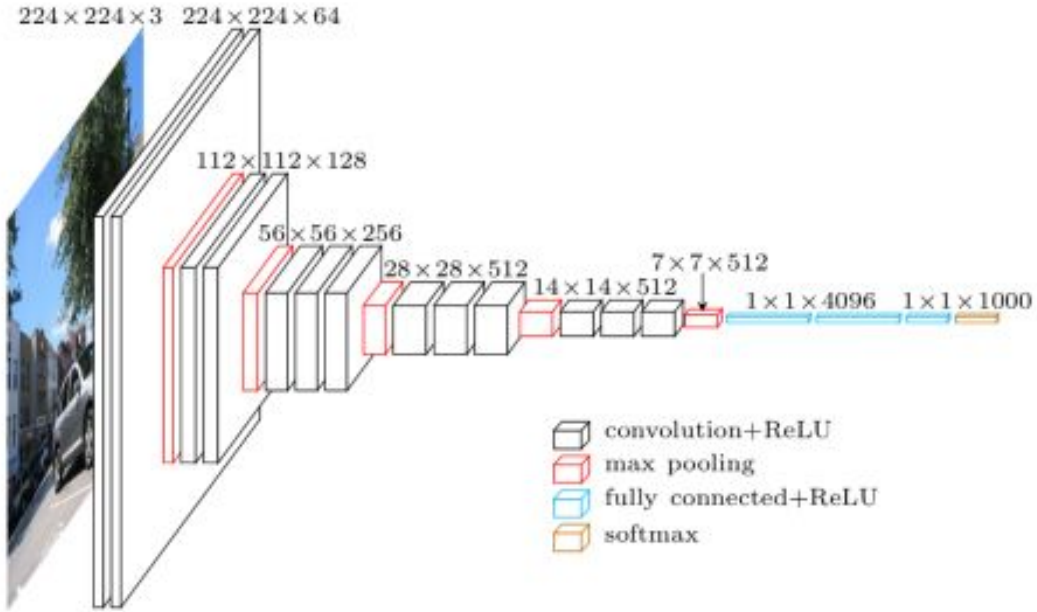
Development of Deep CNNs:

- LeNet
- VGG
- ResNet
- DenseNet
- Inception



Back to top.

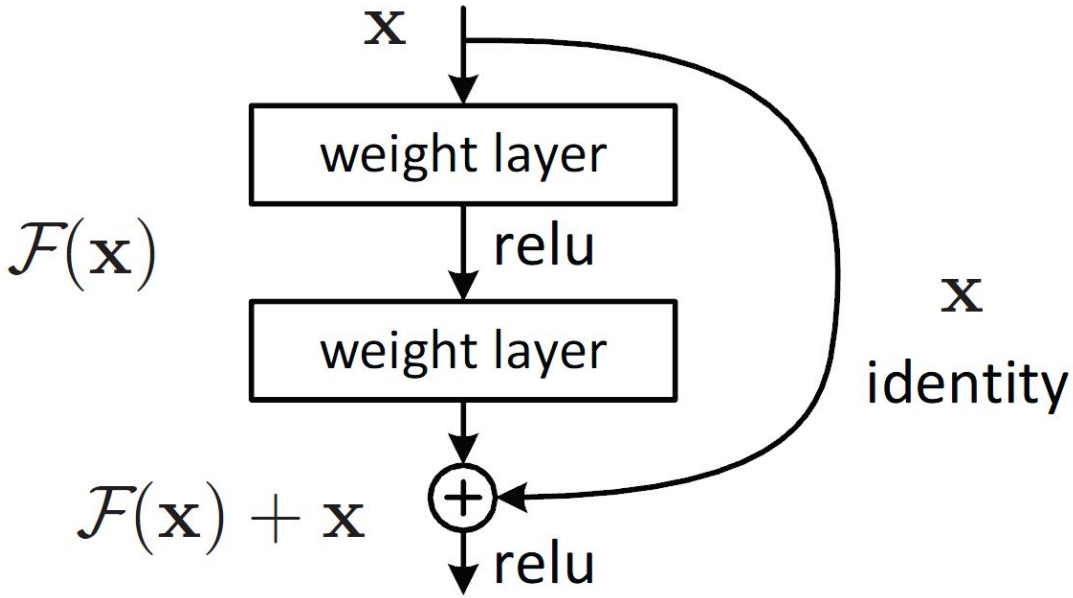
Famous deep CNNs: VGG16



VGG16 takes a color picture that has size 224 by 224 and reduce to size 14 by 14 by using 5 convolutional blocks. Then it sends condensed information through a dense layer with 4096 neurons.

Back to top.

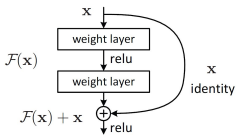
Famous deep CNNs: ResNet

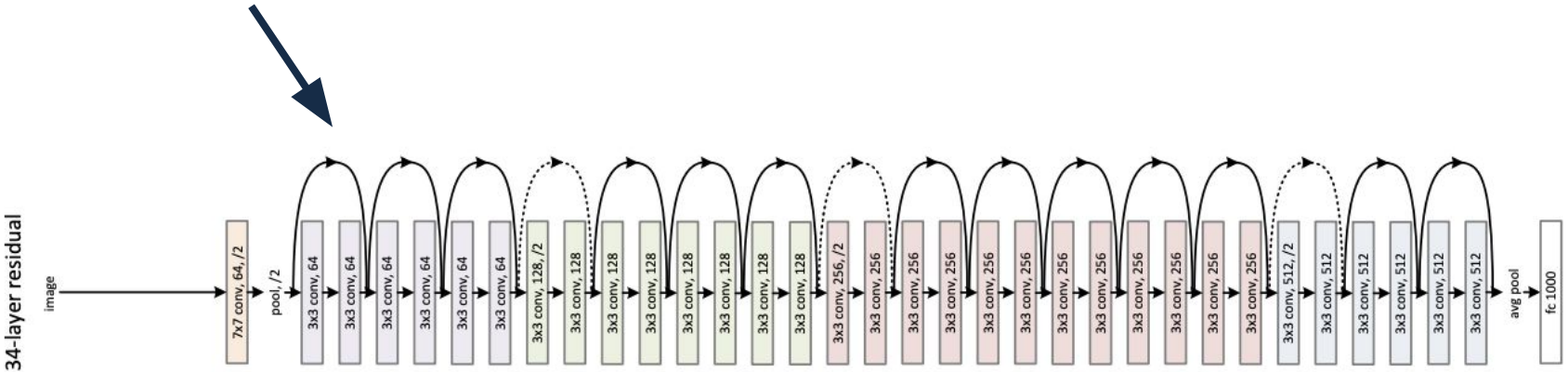


ResNet introduces a separate path called the "identity path", which has a name "Residual Block". Tale has been told that the identity map provides more information and prevents overfitting.

Back to top.

Famous deep CNNs: ResNet



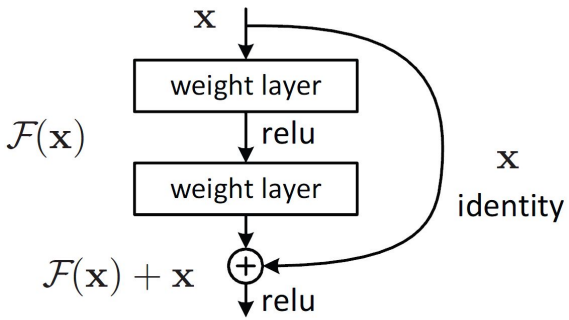


ResNet then builds many many, I mean many, "residual blocks" to form a much longer and condensed network architecture. For regular CNN, the deeper the worse the performance. That is not the case for ResNet!

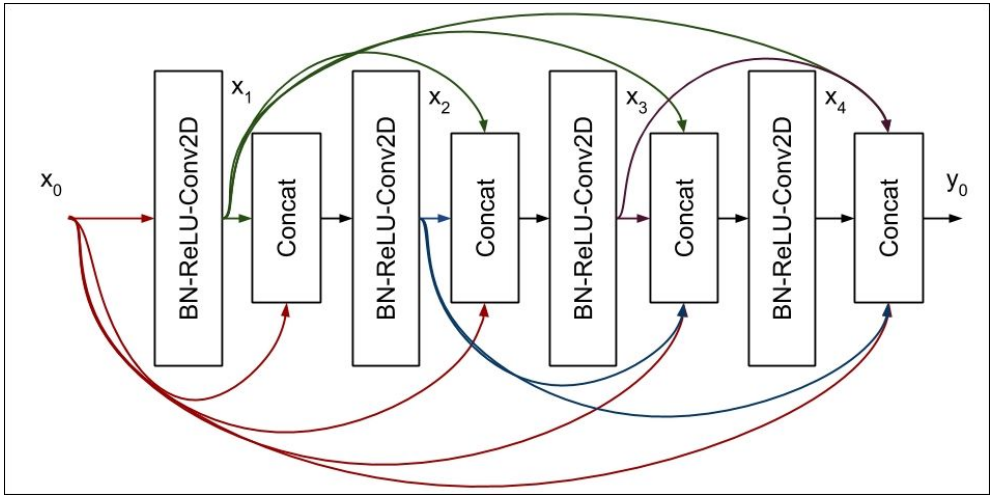
Back to top.

Famous deep CNNs: DenseNet

If this is possibly better, ….



Why don't we go crazy with the identity map?

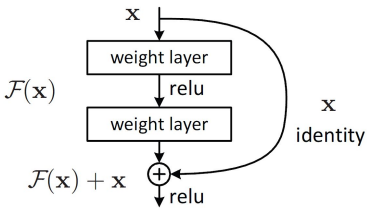


If building one identity map is better, why don't we build many of them? This is exactly what DenseNet is doing. It skips not just one but any number of layers to build many many, I mean many, identity maps. Hence, this model earned its name: DenseNet.

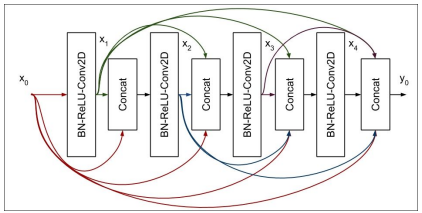
Back to top.

Famous deep CNNs: Inception Network

If this is a good start, ….

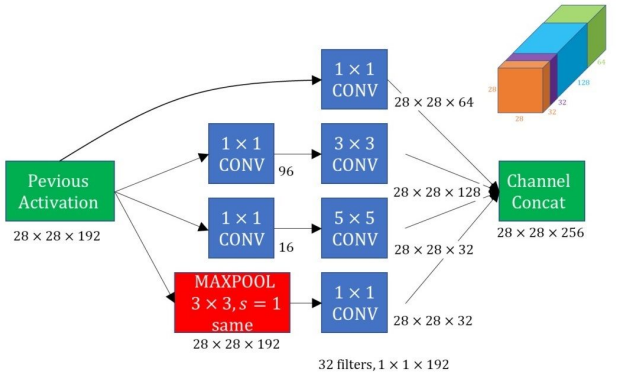


and this is better, …



Why don't we split on the identity maps?

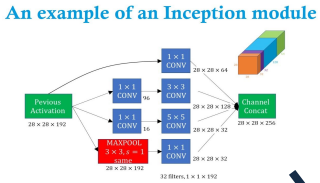
An example of an Inception module



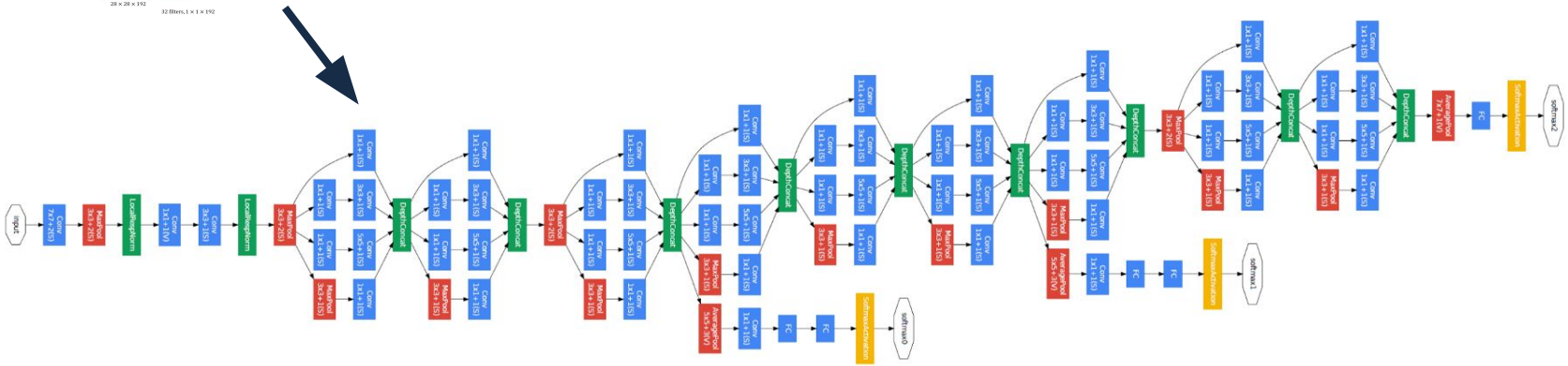
ResNet and DenseNet both take advantage on identity map. ResNet skips one block where DenseNet skips any random number of blocks. Inception network introduces an inception module where there are many diverge identity maps splitting different directions in the beginning and then we converge them in the end.

Back to top.

Famous deep CNNs: Inception Network



Inception Network has many many inception modules.



ResNet and DenseNet both have their own modules and the algorithm repeats the modules. Inception Network is no different. A regular Inception Network can sometimes repeat an inception module for more than 10 times.

One of the most important milestone in late 80s is deep learning! I created this lecture series in 2021 Christmas and I have already been in the field for 6+ years. That is nothing comparing with the entire development of the field since the 1980s.

Famous topic we are introducing here are: **Artificial Neural Network (ANN)**, **Convolutional Neural Network (CNN)**, and **Recurrent Neural Network (RNN)**. Famous deep learning libraries are Keras (from Google Brain) and PyTorch (from Facebook AI Research, FAIR). We introduce these topics in the following order

- **Artificial Neural Network (ANN) - finished!**
- **Convolutional Neural Network (CNN) - finished!**
- **Recurrent Neural Network - in progress ...**

In addition to images, there are other data types that are from temperatures, languages, stock prices, product sales, and etc.. These data types have time stamps along with the variables and these types of data are called **sequential data** or **time-series data**.

This is where we introduce Recurrent Neural Network (RNN) which can be described using the following diagram.

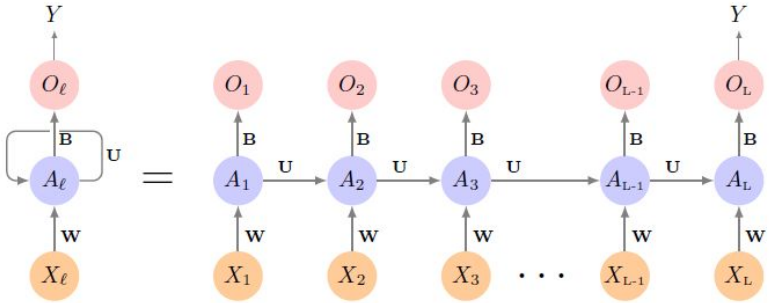


FIGURE 10.12. *Schematic of a simple recurrent neural network. The input is a sequence of vectors* $\{X_\ell\}_1^L$*, and here the target is a single response. The network processes the input sequence* X *sequentially; each* X_ℓ *feeds into the hidden layer, which also has as input the activation vector* $A_{\ell-1}$ *from the previous element in the sequence, and produces the current activation vector* A_ℓ*. The same collections of weights* $\mathbf{W}$, $\mathbf{U}$ *and* $\mathbf{B}$ *are used as each element of the sequence is processed. The output layer produces a sequence of predictions* O_ℓ *from the current activation* A_ℓ*, but typically only the last of these,* O_L*, is of relevance. To the left of the equal sign is a concise representation of the network, which is* unrolled *into a more explicit version on the right.*

Back to top.

In a recurrent neural network (RNN), the input object X is a sequence. Consider a corpus of a document such as the movie reviews from the IMDB dataset. Each document can be represented as a sequence of L words, so $X = \{X_1, X_2, ..., X_L\}$, where X_l represents a word. The output Y can be a sequence (if the task is machine translation). The output can also be a scalar, like the binary sentiment analysis of a review movie document. Suppose these L words are input features. Also suppose that we have hidden layer consists of K units, i.e. $A_l^T = (A_{l1}, A_{l2}, ..., A_{lK})$. We represent the collection of $K \times (p+1)$ shared weights w_{kj} for the input layer by matrix **W**. Similarly, **U** is a $K \times K$ matrix of the weights u_{ks} for the hidden-to-hidden layers, and **B** is a K+1 vector of weights β_k for the output layer. Then

$$A_{lk} = g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_{lj} + \sum_{s=1}^{K} u_{ks} A_{l-1,s})$$

and the output O_l is computed as

$$O_l = \beta_0 + \sum_{k=1}^{K} \beta_k A_{lk}$$

for a quantitative response, or with an additional sigmoid activation function for a binary response. Here the activation g() can be sigmoid or ReLU that we discussed before.

Back to top.

In a recurrent neural network (RNN), the input object X is a sequence. Consider a corpus of a document such as the movie reviews from the IMDB dataset. Each document can be represented as a sequence of L words, so $X = \{X_1, X_2, ..., X_L\}$, where X_l represents a word. The output Y can be a sequence (if the task is machine translation). The output can also be a scalar, like the binary sentiment analysis of a review movie document. Suppose these L words are input features. Also suppose that we have hidden layer consists of K units, i.e. $A_l^T = (A_{l1}, A_{l2}, ..., A_{lK})$. We represent the collection of $K \times (p+1)$ shared weights w_{kj} for the input layer by matrix **W**. Similarly, **U** is a $K \times K$ matrix of the weights u_{ks} for the hidden-to-hidden layers, and **B** is a K+1 vector of weights β_k for the output layer. Then

$$A_{lk} = g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_{lj} + \sum_{s=1}^{K} u_{ks} A_{l-1,s})$$

Notice that **W**, **U**, and **B** are shared, which means they take the same values in the sequence.

and the output O_l is computed as

$$O_l = \beta_0 + \sum_{k=1}^{K} \beta_k A_{lk}$$

for a quantitative response, or with an additional sigmoid activation function for a binary response. Here the activation g() can be sigmoid or ReLU that we discussed before.

Back to top.

How to train the RNN?

For regression problems, the loss function of (X,Y) is

$$(Y - O_L)^2$$

while the final output is

$$O_L = \beta_0 + \sum_{k=1}^{K} \beta_k A_{LK}$$

With n samples, we can formally write the following

$$\sum_{i=1}^{n}(y_i - o_{iL})^2 = \sum_{i=1}^{n}\left(y_i - (\beta_0 + \sum_{k=1}^{K}\beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj}x_{iLj} + \sum_{s=1}^{K} u_{ks}a_{i,L-1,s}))\right)^2$$

Back to top.

How to process words, sentences, or paragraphs to fit a RNN?

We introduce this concept of an **embedding layer**. We have, however, a dimensionality problem: each word in our document is represented by a one-hot-encoded vector (dummy variable) with 10,000 elements (one per word in the dictionary)!

An approach that has become popular is to represent each word in a much lower-dimensional embedding embedding space. This means that rather than representing each word by a binary vector with 9,999 zeros and a single one in some position, we will represent it instead by a set of m real numbers, none of which are typically zero. Here m is the embedding dimension, and can be in the low 100s, or even less. This means (in our case) that we need a matrix **E** of dimension $m \times 10,000$ where each column is indexed by one of the 10,000 words in our dictionary, and the values in that column give the m coordinates for that word in the embedding space.

This matrix **E** is visualized in the following.

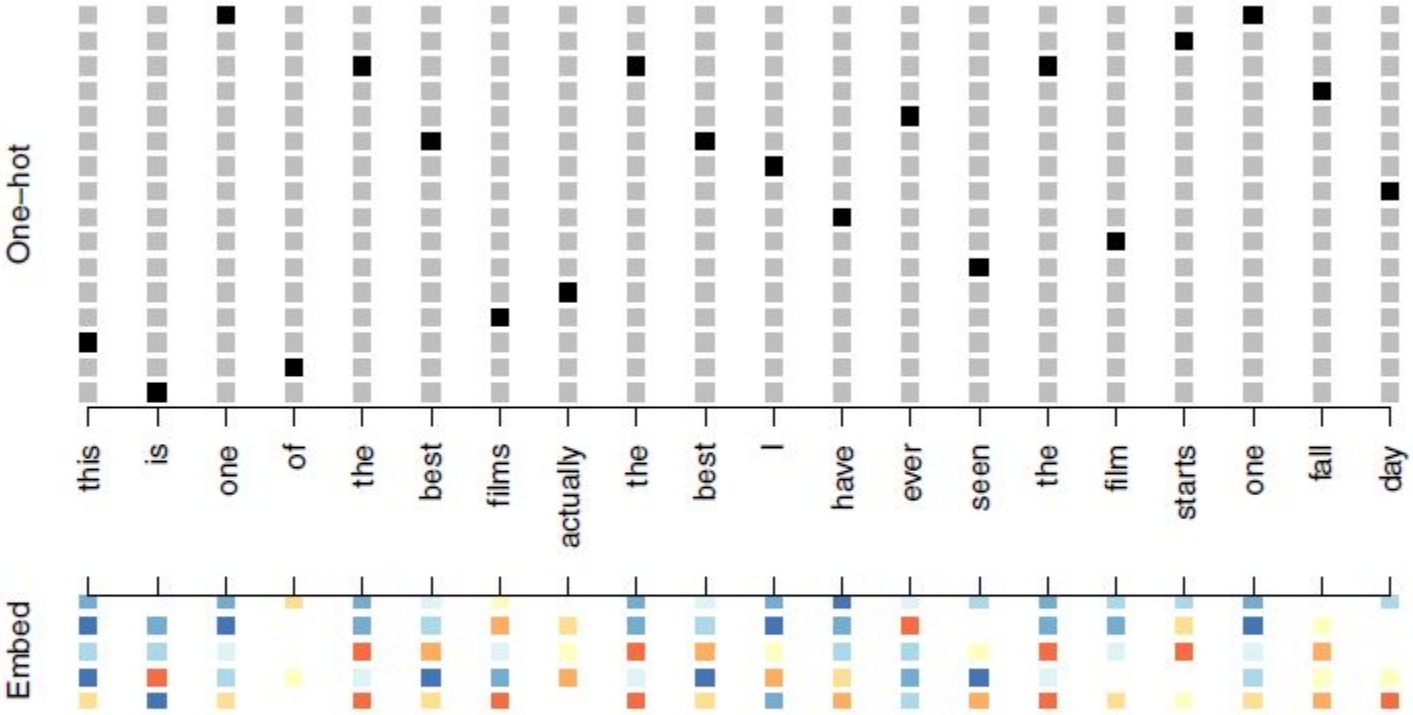


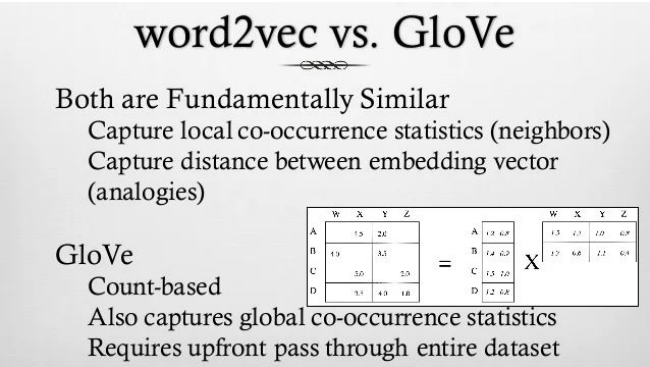
FIGURE 10.13. *Depiction of a sequence of* 20 *words representing a single document: one-hot encoded using a dictionary of* 16 *words (top panel) and embedded in an* m*-dimensional space with* $m = 5$ *(bottom panel).*

Back to top.

Where does E come from?

- If we have a large corpus of labeled documents, we can have the neural network learn **E** as part of the optimization. In this case **E** is referred to as an embedding layer, embedding layer and a specialized **E** is learned for the task at hand.

- Otherwise we can insert a precomputed matrix **E** in the embedding layer, a process known as weight freezing. Two pretrained embeddings, word2vec and GloVe, are weight freezing widely used. These are built from a very large corpus of documents by a variant of principal components analysis (Section 12.2). The idea is that the positions of words in the embedding space preserve semantic meaning; e.g. synonyms should appear near each other.



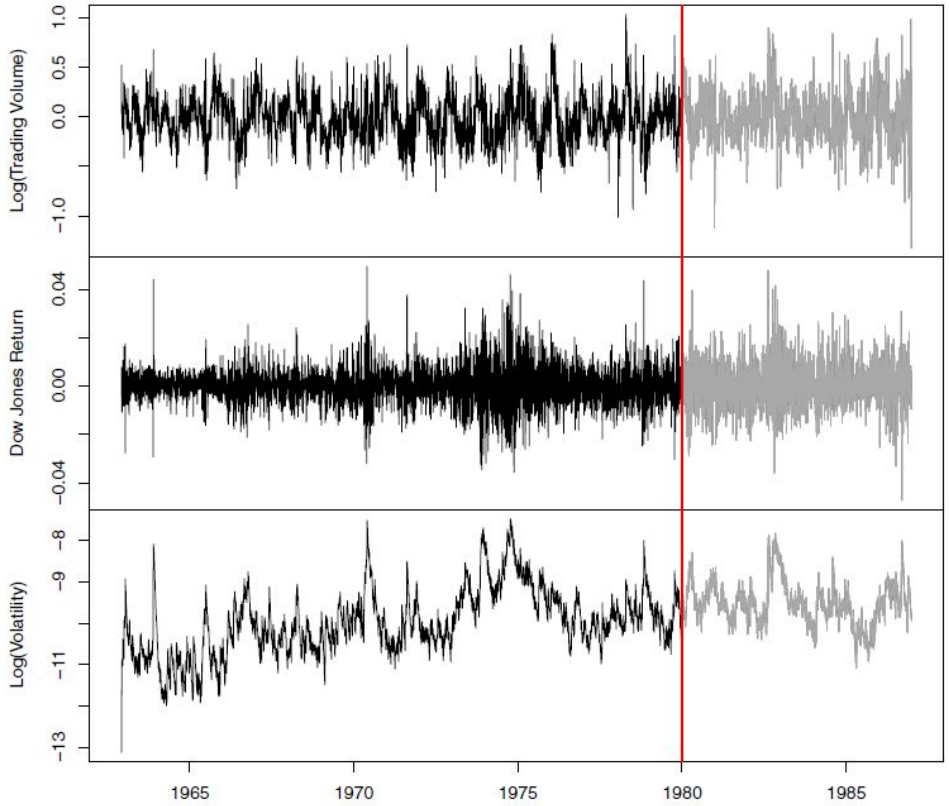


FIGURE 10.14. *Historical trading statistics from the New York Stock Exchange. Daily values of the normalized log trading volume, DJIA return, and log volatility are shown for a 24-year period from 1962–1986. We wish to predict trading volume on any day, given the history on all earlier days. To the left of the red bar (January 2, 1980) is training data, and to the right test data.*

Back to top.

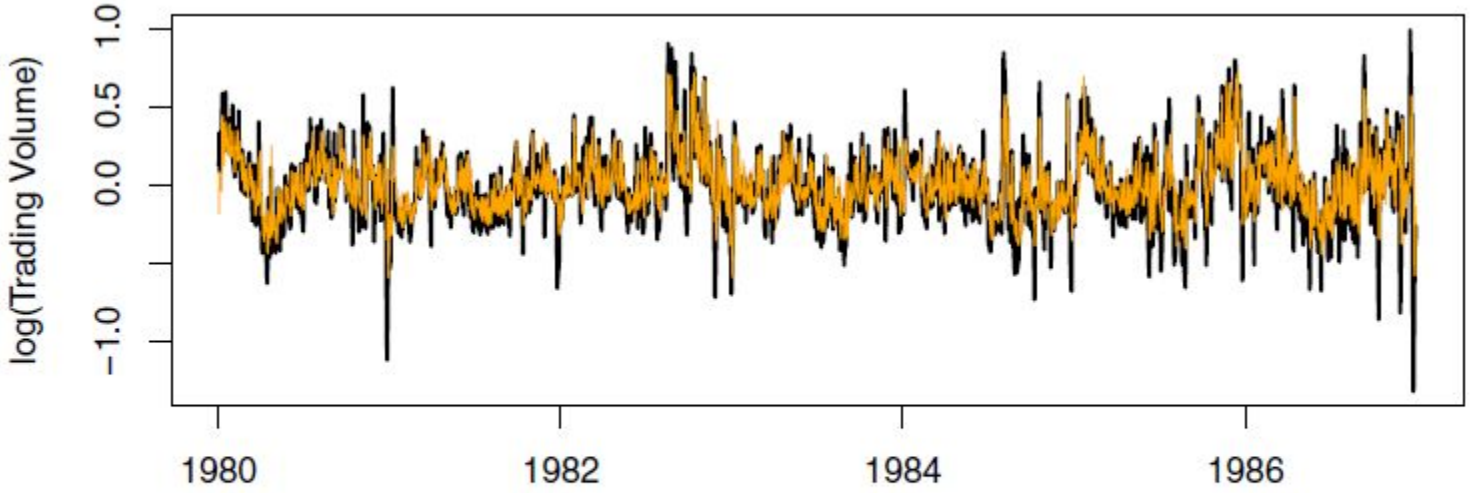


FIGURE 10.16. *RNN forecast of* `log_volume` *on the* `NYSE` *test data. The black lines are the true volumes, and the superimposed orange the forecasts. The forecasted series accounts for 42% of the variance of* `log_volume`.

Back to top.

The RNN we just fit has much in common with a traditional *autoregression* (AR) linear model, which we present now for comparison. We first consider the response sequence v_t alone, and construct a response vector $\mathbf{y}$ and a matrix $\mathbf{M}$ of predictors for least squares regression as follows:

$$\mathbf{y} = \begin{bmatrix} v_{L+1} \\ v_{L+2} \\ v_{L+3} \\ \vdots \\ v_T \end{bmatrix} \qquad \mathbf{M} = \begin{bmatrix} 1 & v_L & v_{L-1} & \cdots & v_1 \\ 1 & v_{L+1} & v_L & \cdots & v_2 \\ 1 & v_{L+2} & v_{L+1} & \cdots & v_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & v_{T-1} & v_{T-2} & \cdots & v_{T-L} \end{bmatrix}. \tag{10.21}$$

$\mathbf{M}$ and $\mathbf{y}$ each have $T - L$ rows, one per observation. We see that the predictors for any given response v_t on day t are the previous L values of the same series. Fitting a regression of $\mathbf{y}$ on $\mathbf{M}$ amounts to fitting the model

$$\hat{v}_t = \hat{\beta}_0 + \hat{\beta}_1 v_{t-1} + \hat{\beta}_2 v_{t-2} + \cdots + \hat{\beta}_L v_{t-L}, \tag{10.22}$$

and is called an order-L autoregressive model, or simply AR(L). For the `NYSE` data we can include lagged versions of `DJ_return` and `log_volatility`, r_t and z_t, in the predictor matrix $\mathbf{M}$, resulting in $3L + 1$ columns. An AR model with $L = 5$ achieves a test R^2 of 0.41, slightly inferior to the 0.42 achieved by the RNN.

One of the most important milestone in late 80s is deep learning! I created this lecture series in 2021 Christmas and I have already been in the field for 6+ years. That is nothing comparing with the entire development of the field since the 1980s.

Famous topic we are introducing here are: **Artificial Neural Network (ANN)**, **Convolutional Neural Network (CNN)**, and **Recurrent Neural Network (RNN)**. Famous deep learning libraries are Keras (from Google Brain) and PyTorch (from Facebook AI Research, FAIR). We introduce these topics in the following order

- **Artificial Neural Network (ANN) - finished!**
- **Convolutional Neural Network (CNN) - finished!**
- **Recurrent Neural Network - finished!**

For more information, please go to WYN Edu.

Chapter 11 - Unsupervised Learning

Go back to main content, click here

Back to top.

This chapter introduces **unsupervised learning** methods. From prior chapters, we always have an outcome variable that we are interested in. For example, linear regression model has an outcome Y that we want to learn to make educated guess of. It is the same with logistic regression, SVM, Neural Networks, and so on. They loss functions for **supervised learning** take the form of

Ground Truth - Educated Guess from a Model

This is not the guideline for **unsupervised learning**. In unsupervised learning, we are not interested in making predictions, because we do not have an associated outcome variable Y.

Is there an informative way to visualize the data? Can we discover subgroups among the variables or among the observations? Unsupervised learning refers to a diverse set of techniques for answering questions such as these. In this chapter, we will focus on two particular types of unsupervised learning: **principal components analysis (PCA)**, a tool used for data visualization or data pre-processing before supervised techniques are applied, and **clustering**, a broad class of methods for discovering unknown subgroups in data.

Principal components analysis (PCA) refers to the process by which principal components analysis are computed, and the subsequent use of these components in understanding the data.

The first principal component of a set of features X1, X2, . . . , Xp is the normalized linear combination of the features

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p$$

that has the largest variance. By normalized, we mean that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$ and we refer to all the phi values the loadings of the first principle component.

Back to top.

Principal components analysis (PCA) refers to the process by which principal components analysis are computed, and the subsequent use of these components in understanding the data.

The first principal component of a set of features X1, X2, . . . , Xp is the normalized linear combination of the features

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p$$

that has the largest variance. By normalized, we mean that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$ and we refer to all the phi values the loadings of the first principle component.

Given a $n \times p$ data set $\mathbf{X}$, how do we compute the first principal component? Since we are only interested in variance, we assume that each of the variables in $\mathbf{X}$ has been centered to have mean zero (that is, the column means of $\mathbf{X}$ are zero). We then look for the linear combination of the sample feature values of the form

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{p1}x_{ip}$$

Given a $n \times p$ data set $\mathbf{X}$, how do we compute the first principal component? Since we are only interested in variance, we assume that each of the variables in $\mathbf{X}$ has been centered to have mean zero (that is, the column means of $\mathbf{X}$ are zero). We then look for the linear combination of the sample feature values of the form

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{p1}x_{ip}$$

How to find these loadings? The first principal component loading vector solves the optimization problem

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximize}} \left\{ \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}x_{ij}\right)^2 \right\} \text{ subject to } \sum_{j=1}^{p}\phi_{j1}^2 = 1$$

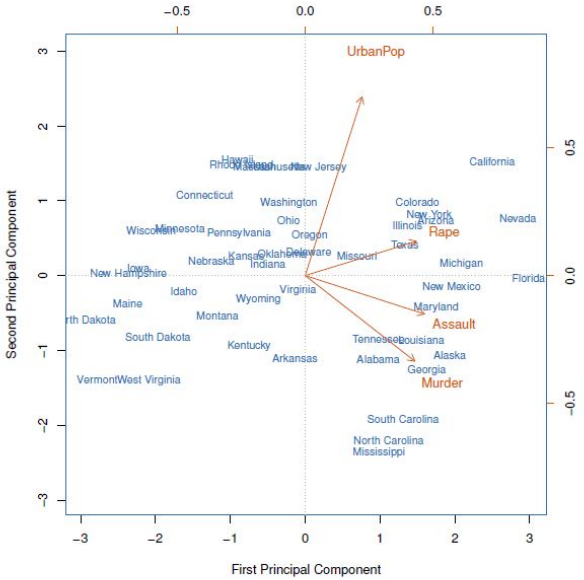
where we can rewrite the constraints as $\frac{1}{n}\sum_{i=1}^{n} z_{i1}^2$

Back to top.

After the first principal component Z1 of the features has been determined, we can find the second principal component Z2. The second principal component is the linear combination of X1, . . . ,Xp that has maximal variance out of all linear combinations that are uncorrelated with Z1. The second principal component scores $z_{12}, z_{22}, \ldots, z_{n2}$ take the form

$$z_{i2} = \phi_{12}x_{i1} + \phi_{22}x_{i2} + \cdots + \phi_{p2}x_{ip}$$

where these phi values are the loadings for the second principal components.



Back to top.

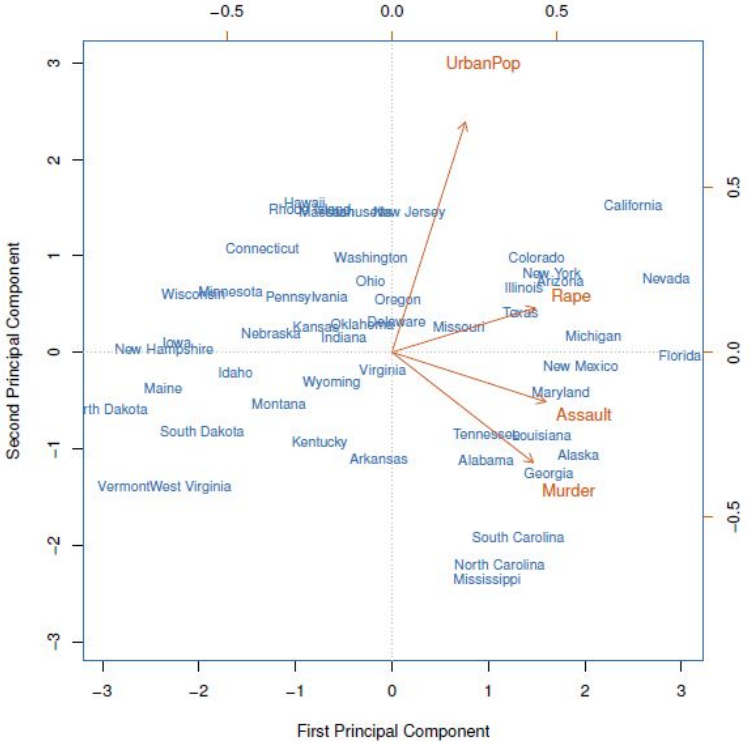


FIGURE 12.1. *The first two principal components for the* `USArrests` *data. The blue state names represent the scores for the first two principal components. The orange arrows indicate the first two principal component loading vectors (with axes on the top and right). For example, the loading for* `Rape` *on the first component is* 0.54*, and its loading on the second principal component* 0.17 *(the word* `Rape` *is centered at the point* $(0.54, 0.17)$*). This figure is known as a biplot, because it displays both the principal component scores and the principal component loadings.*

Back to top.

The first M principal component score vectors and the first M principal component loading vectors provide the best M-dimensional approximation (in terms of Euclidean distance) to the ith observation xij . This representation can be written as

$$x_{ij} \approx \sum_{m=1}^{M} z_{im}\phi_{jm}$$

We can state this more formally by writing down an optimization problem. Suppose the data matrix X is column-centered. Out of all approximations of the form

$$x_{ij} \approx \sum_{m=1}^{M} z_{im}\phi_{jm}$$

and we could ask for the one with the smallest residual sum of squares

$$\underset{\mathbf{A}\in\mathbb{R}^{n\times M},\mathbf{B}\in\mathbb{R}^{p\times M}}{\text{minimize}} \left\{ \sum_{j=1}^{p}\sum_{i=1}^{n} \left(x_{ij} - \sum_{m=1}^{M} a_{im}b_{jm} \right)^2 \right\}$$

Back to top.

We can now ask a natural question: how much of the information in a given data set is lost by projecting the observations onto the first few principal components? That is, how much of the variance in the data is *not* contained in the first few principal components? More generally, we are interested in knowing the *proportion of variance explained* (PVE) by each principal component. The *total variance* present in a data set (assuming that the variables have been centered to have mean zero) is defined as

Back to top.

We can now ask a natural question: how much of the information in a given data set is lost by projecting the observations onto the first few principal components? That is, how much of the variance in the data is *not* contained in the first few principal components? More generally, we are interested in knowing the *proportion of variance explained* (PVE) by each principal component. The *total variance* present in a data set (assuming that the variables have been centered to have mean zero) is defined as

$$\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2,$$

and the variance explained by the mth principal component is

$$\frac{1}{n} \sum_{i=1}^{n} z_{im}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right)^2.$$

We can now ask a natural question: how much of the information in a given data set is lost by projecting the observations onto the first few principal components? That is, how much of the variance in the data is *not* contained in the first few principal components? More generally, we are interested in knowing the *proportion of variance explained* (PVE) by each principal component. The *total variance* present in a data set (assuming that the variables have been centered to have mean zero) is defined as

$$\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2,$$

and the variance explained by the mth principal component is

$$\frac{1}{n} \sum_{i=1}^{n} z_{im}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right)^2.$$

Therefore, the PVE of the mth principal component is given by

$$\frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2} = \frac{\sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right)^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2}.$$

We showed that the first M principal component loading and score vectors can be interpreted as the best M-dimensional approximation to the data, in terms of residual sum of squares. It turns out that the variance of the data can be decomposed into the variance of the first M principal components plus the mean squared error of this M-dimensional approximation, as follows

$$\underbrace{\sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2}_{\text{Var. of data}} = \underbrace{\sum_{m=1}^{M} \frac{1}{n} \sum_{i=1}^{n} z_{im}^2}_{\text{Var. of first } M \text{ PCs}} + \underbrace{\frac{1}{n} \sum_{j=1}^{p} \sum_{i=1}^{n} \left(x_{ij} - \sum_{m=1}^{M} z_{im} \phi_{jm} \right)^2}_{\text{MSE of } M\text{-dimensional approximation}}$$

We can see that from certain arrangements the PVE defined above can be rewritten as

$$1 - \frac{\sum_{j=1}^{p} \sum_{i=1}^{n} \left(x_{ij} - \sum_{m=1}^{M} z_{im} \phi_{jm} \right)^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

Back to top.

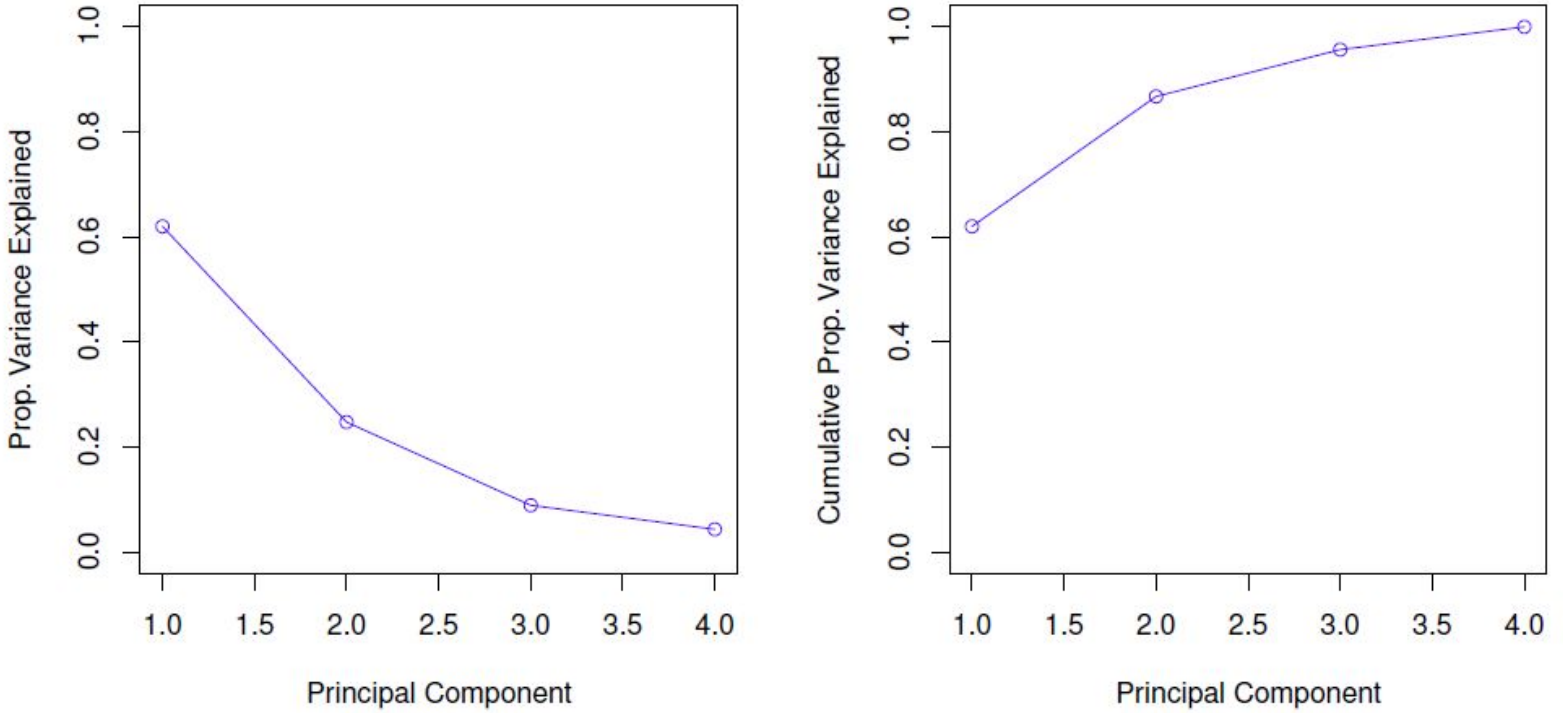


FIGURE 12.3. Left: *a scree plot depicting the proportion of variance explained by each of the four principal components in the* `USArrests` *data.* Right: *the cumulative proportion of variance explained by the four principal components in the* `USArrests` *data.*

Back to top.

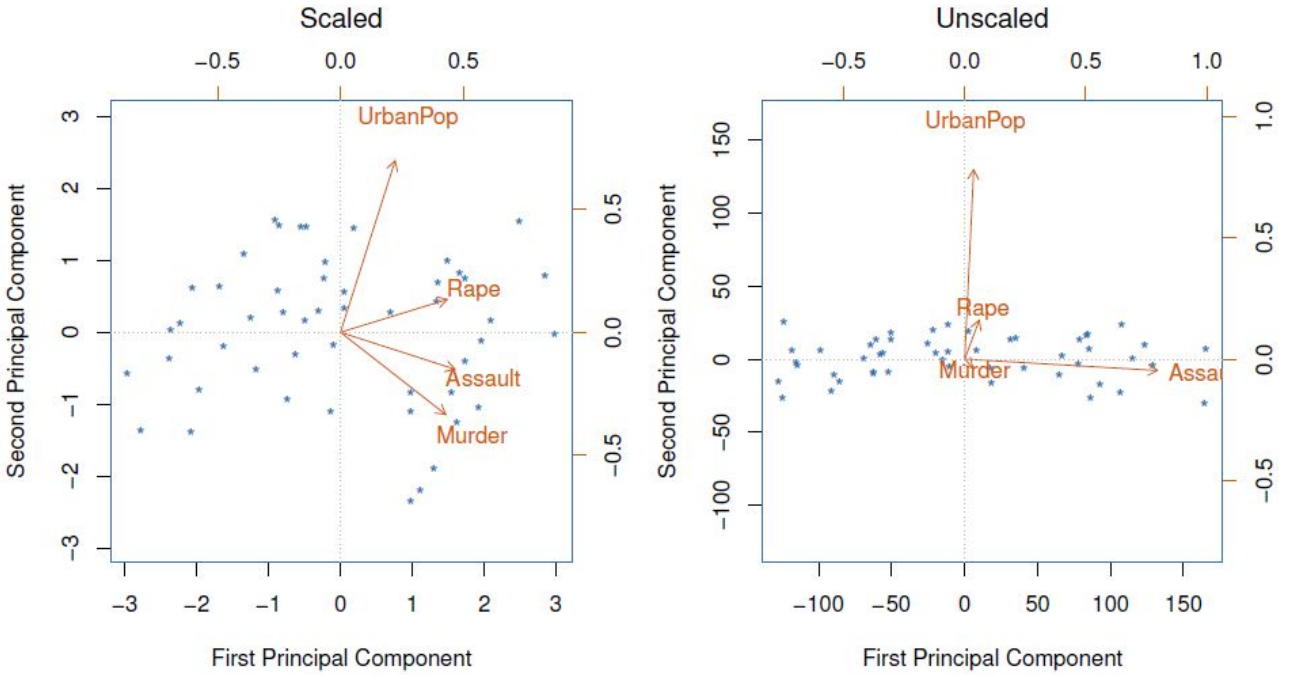


FIGURE 12.4. *Two principal component biplots for the* `USArrests` *data.* Left: *the same as Figure 12.1, with the variables scaled to have unit standard deviations.* Right: *principal components using unscaled data.* `Assault` *has by far the largest loading on the first principal component because it has the highest variance among the four variables. In general, scaling the variables to have standard deviation one is recommended.*

Clustering refers to a very broad set of techniques for finding subgroups, or clustering clusters, in a data set. When we cluster the observations of a data set, we seek to partition them into distinct groups so that the observations within each group are quite similar to each other, while observations in different groups are quite different from each other. Of course, to make this concrete, we must define what it means for two or more observations to be similar or different. Indeed, this is often a domain-specific consideration that must be made based on knowledge of the data being studied

Both clustering and PCA seek to simplify the data via a small number of summaries, but their mechanisms are different:

- **PCA** looks to find a low-dimensional representation of the observations that explain a good fraction of the variance;
- **Clustering** looks to find homogeneous subgroups among the observations.

Since clustering is popular in many fields, there exist a great number of clustering methods. In this section we focus on perhaps the two best-known clustering approaches: K-means clustering and hierarchical clustering. In K-means clustering, we seek to partition the observations into a pre-specified number of clusters.

The K-means clustering procedure results from a simple and intuitive mathematical problem. We begin by defining some notation. Let $C_1, \ldots, C_K$ denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:

1. $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$. In other words, each observation belongs to at least one of the K clusters.

2. $C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$. In other words, the clusters are non-overlapping: no observation belongs to more than one cluster.

The K-means clustering procedure results from a simple and intuitive mathematical problem. We begin by defining some notation. Let $C_1, \ldots, C_K$ denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:

1. $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$. In other words, each observation belongs to at least one of the K clusters.

2. $C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$. In other words, the clusters are non-overlapping: no observation belongs to more than one cluster.

For instance, if the ith observation is in the kth cluster, then $i \in C_k$. The idea behind K-means clustering is that a *good* clustering is one for which the *within-cluster variation* is as small as possible. The within-cluster variation for cluster C_k is a measure $W(C_k)$ of the amount by which the observations within a cluster differ from each other. Hence we want to solve the problem

The K-means clustering procedure results from a simple and intuitive mathematical problem. We begin by defining some notation. Let $C_1, \ldots, C_K$ denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:

1. $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$. In other words, each observation belongs to at least one of the K clusters.

2. $C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$. In other words, the clusters are non-overlapping: no observation belongs to more than one cluster.

For instance, if the ith observation is in the kth cluster, then $i \in C_k$. The idea behind K-means clustering is that a *good* clustering is one for which the *within-cluster variation* is as small as possible. The within-cluster variation for cluster C_k is a measure $W(C_k)$ of the amount by which the observations within a cluster differ from each other. Hence we want to solve the problem

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

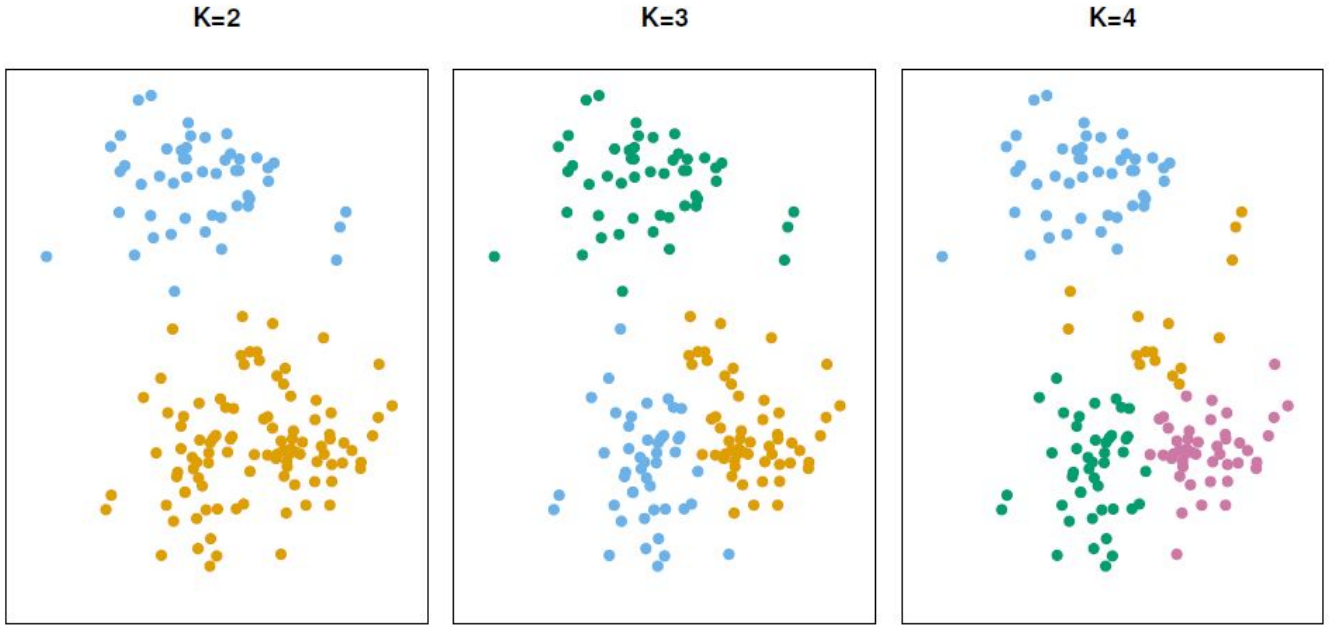


FIGURE 12.7. *A simulated data set with 150 observations in two-dimensional space. Panels show the results of applying K-means clustering with different values of K, the number of clusters. The color of each observation indicates the cluster to which it was assigned using the K-means clustering algorithm. Note that there is no ordering of the clusters, so the cluster coloring is arbitrary. These cluster labels were not used in clustering; instead, they are the outputs of the clustering procedure.*

Back to top.

In order to make it actionable we need to define the within-cluster variation. There are many possible ways to define this concept, but by far the most common choice involves squared Euclidean distance. That is, we define

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

where $|C_k|$ denotes the number of observations in the kth cluster.

In other words, the within-cluster variation for the kth cluster is the sum of all of the pairwise squared Euclidean distances between the observations in the kth cluster, divided by the total number of observations in the kth cluster. Combining the above equations gives the optimization problem that defines K-means clustering

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

Back to top.

Algorithm 12.2 *K-Means Clustering*

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.

2. Iterate until the cluster assignments stop changing:

   (a) For each of the K clusters, compute the cluster *centroid*. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.

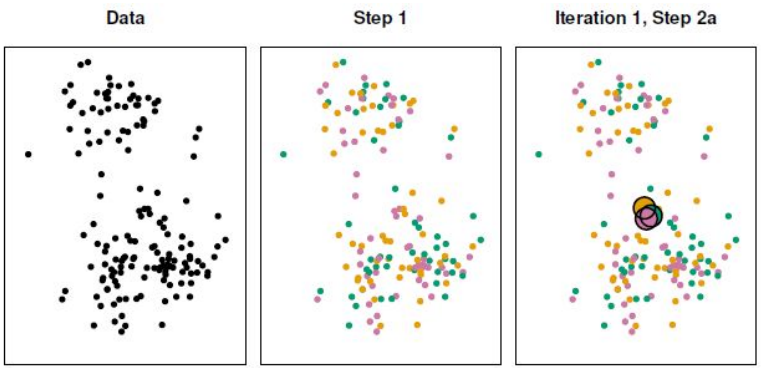
   (b) Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).

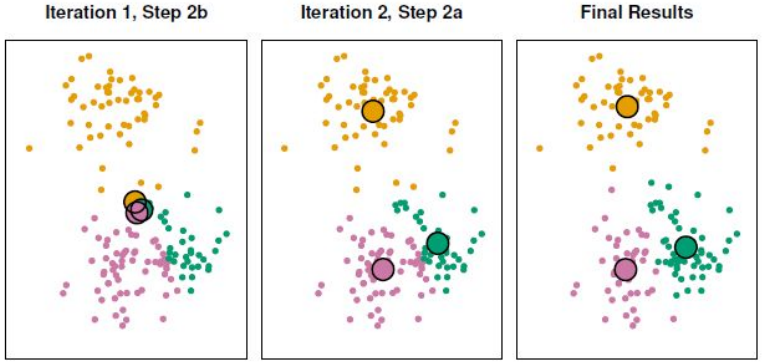
Back to top.

Algorithm 12.2 is guaranteed to decrease the value of the objective at each step. To understand why, the following identity is illuminating:

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 = 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2$$

where $\bar{x}_{kj} = \frac{1}{|C_k|}\sum_{i\in C_k} x_{ij}$ is the mean for feature j in cluster C_k





Back to top.

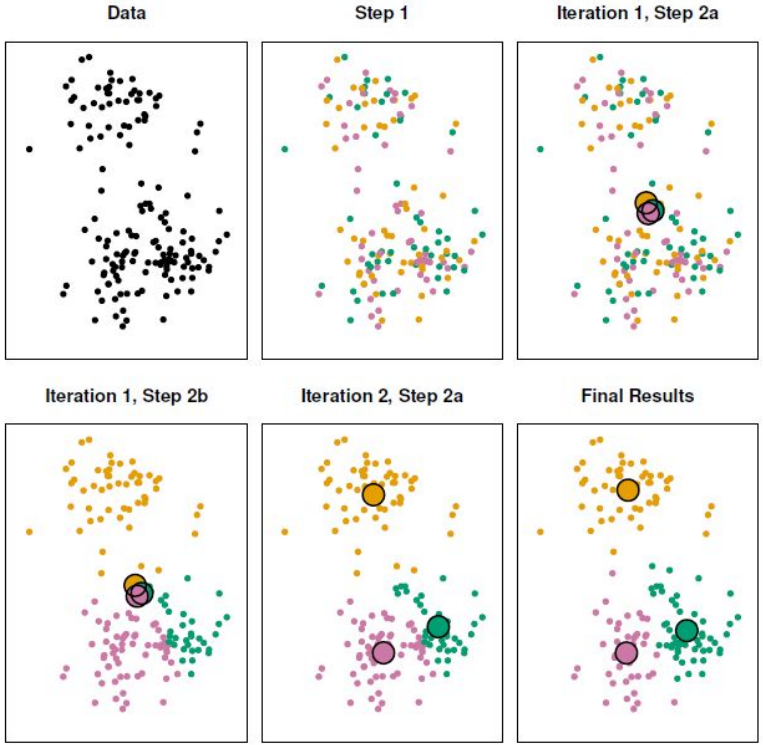


FIGURE 12.8. *The progress of the K-means algorithm on the example of Figure 12.7 with K=3.* Top left: *the observations are shown.* Top center: *in Step 1 of the algorithm, each observation is randomly assigned to a cluster.* Top right: *in Step 2(a), the cluster centroids are computed. These are shown as large colored disks. Initially the centroids are almost completely overlapping because the initial cluster assignments were chosen at random.* Bottom left: *in Step 2(b), each observation is assigned to the nearest centroid.* Bottom center: *Step 2(a) is once again performed, leading to new cluster centroids.* Bottom right: *the results obtained after ten iterations.*

Back to top.

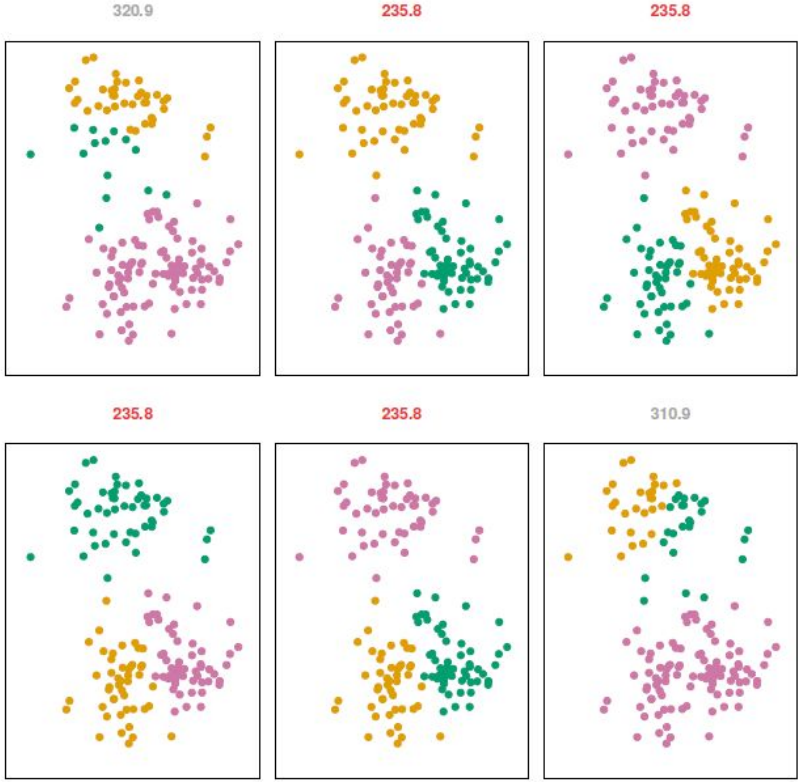


FIGURE 12.9. *K-means clustering performed six times on the data from Figure 12.7 with $K = 3$, each time with a different random assignment of the observations in Step 1 of the K-means algorithm. Above each plot is the value of the objective (12.17). Three different local optima were obtained, one of which resulted in a smaller value of the objective and provides better separation between the clusters. Those labeled in red all achieved the same best solution, with an objective value of 235.8.*

Back to top.

The hierarchical clustering dendrogram is obtained via an extremely simple algorithm.We begin by defining some sort of dissimilarity measure between each pair of observations. Most often, Euclidean distance is used; we will discuss the choice of dissimilarity measure later in this chapter. The algorithm proceeds iteratively.

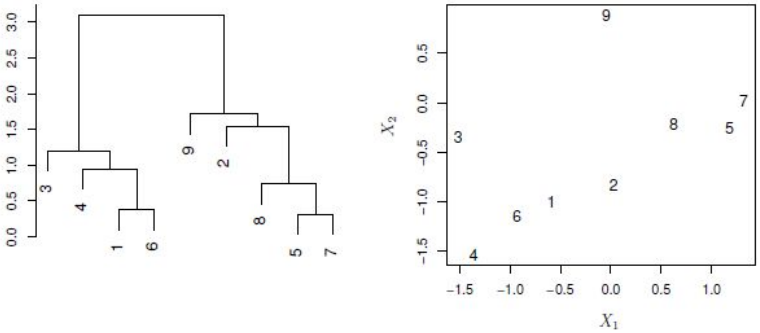


FIGURE 12.12. *An illustration of how to properly interpret a dendrogram with nine observations in two-dimensional space.* Left: *a dendrogram generated using Euclidean distance and complete linkage. Observations* 5 *and* 7 *are quite similar to each other, as are observations* 1 *and* 6. *However, observation* 9 *is* no more similar to *observation* 2 *than it is to observations* 8, 5, *and* 7, *even though observations* 9 *and* 2 *are close together in terms of horizontal distance. This is because observations* 2, 8, 5, *and* 7 *all fuse with observation* 9 *at the same height, approximately* 1.8. Right: *the raw data used to generate the dendrogram can be used to confirm that indeed, observation* 9 *is no more similar to observation* 2 *than it is to observations* 8, 5, *and* 7.

Back to top.

Algorithm 12.3 *Hierarchical Clustering*

1. Begin with n observations and a measure (such as Euclidean distance) of all the $\binom{n}{2} = n(n-1)/2$ pairwise dissimilarities. Treat each observation as its own cluster.

2. For $i = n, n-1, \ldots, 2$:

   (a) Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.

   (b) Compute the new pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.

Back to top.

Linkage	*Description*
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

TABLE 12.3. *A summary of the four most commonly-used types of linkage in hierarchical clustering.*

Variational Autoencoders (VAEs).

In a nutshell, a VAE is an autoencoder whose encodings distribution is regularised during the training in order to ensure that its latent space has good properties allowing us to generate some new data. Moreover, the term “variational” comes from the close relation there is between the regularisation and the variational inference method in statistics.

If the last two sentences summarise pretty well the notion of VAEs, they can also raise a lot of questions. What is an autoencoder? What is the latent space and why regularising it? How to generate new data from VAEs? What is the link between VAEs and variational inference? In order to describe VAEs as well as possible, we will try to answer all this questions (and many others!) and to provide the reader with as much insights as we can (ranging from basic intuitions to more advanced mathematical details). Thus, the purpose of this post is not only to discuss the fundamental notions Variational Autoencoders rely on but also to build step by step and starting from the very beginning the reasoning that leads to these notions.

Back to top.

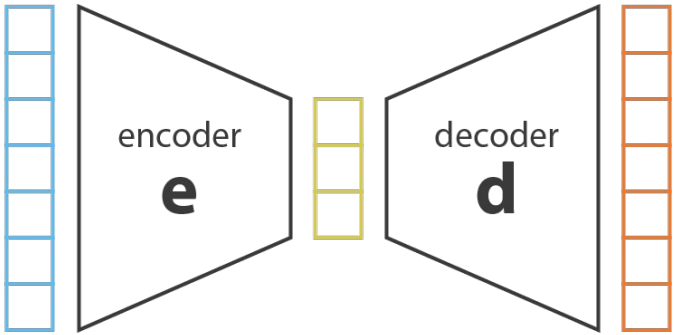
In this first section we will start by discussing some notions related to dimensionality reduction. In particular, we will review briefly principal component analysis (PCA) and autoencoders, showing how both ideas are related to each others.

What is dimensionality reduction?

In machine learning, **dimensionality reduction is the process of reducing the number of features that describe some data**. This reduction is done either by selection (only some existing features are conserved) or by extraction (a reduced number of new features are created based on the old features) and can be useful in many situations that require low dimensional data (data visualisation, data storage, heavy computation...).

Back to top.

First, let's call **encoder** the process that produce the "new features" representation from the "old features" representation (by selection or by extraction) and **decoder** the reverse process. Dimensionality reduction can then be interpreted as data compression where the encoder compress the data (from the initial space to the **encoded space**, also called **latent space**) whereas the decoder decompress them.



x — **initial data** in space R^n

e(x) — **encoded data** in latent space R^m (with m<n)

d(e(x)) — **encoded-decoded data** back in the initial space R^n

$x = d(e(x))$ ➡ **lossless encoding**
no information is lost when reducing the number of dimensions

$x \neq d(e(x))$ ➡ **lossy encoding**
some information is lost when reducing the number of dimensions and can't be recovered later

Back to top.

The main purpose of a dimensionality reduction method is to find the best encoder/decoder pair among a given family. In other words, for a given set of possible encoders and decoders, we are looking for the pair that **keeps the maximum of information when encoding** and, so, **has the minimum of reconstruction error when decoding**. If we denote respectively E and D the families of encoders and decoders we are considering, then the dimensionality reduction problem can be written

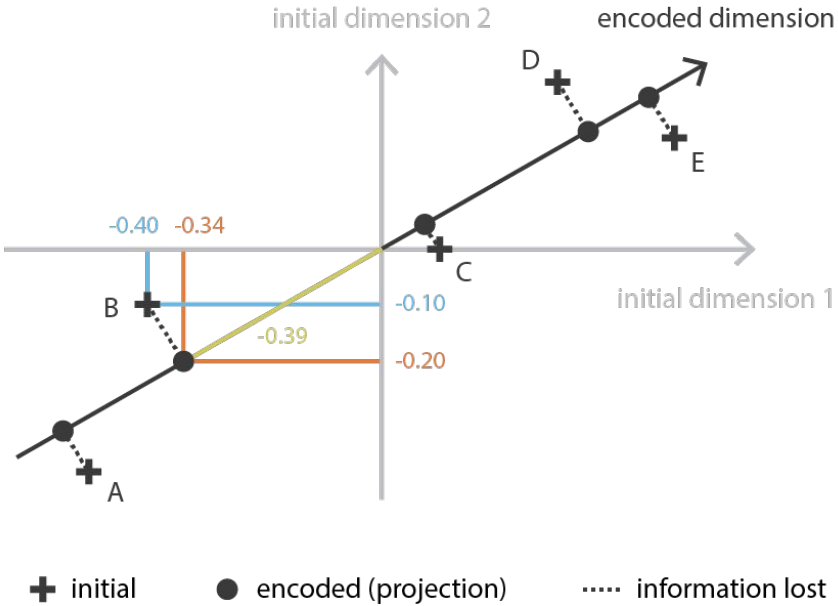
$$(e^*, d^*) = \underset{(e,d) \in E \times D}{\arg\min}\ \epsilon(x, d(e(x)))$$

where

$$\epsilon(x, d(e(x)))$$

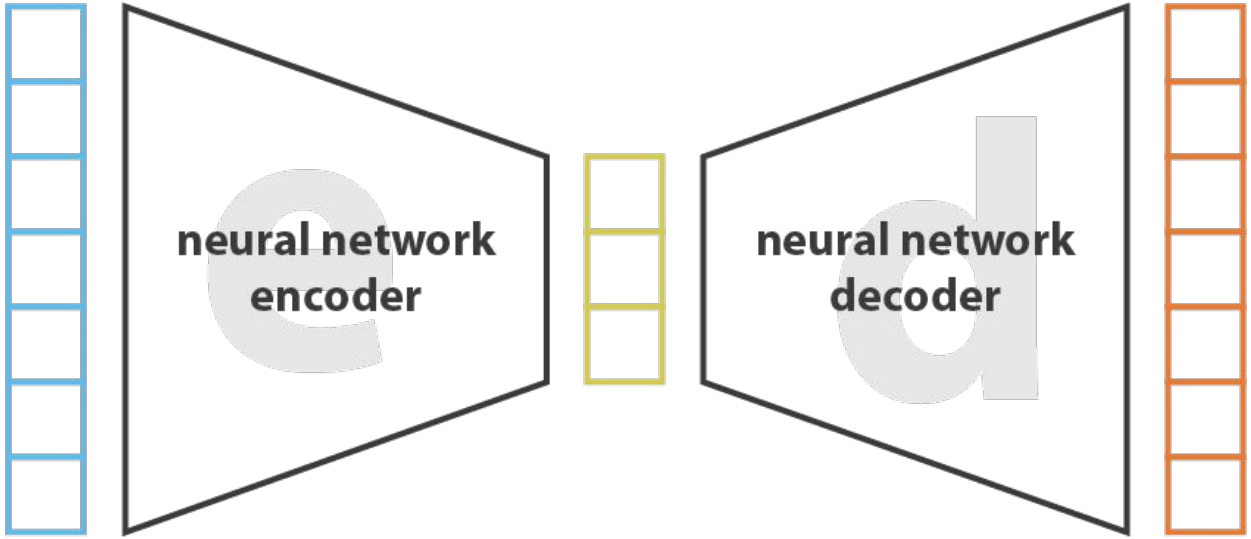
defines the reconstruction error measure between the input data x and the encoded-decoded data d(e(x)). Notice finally that in the following we will denote N the number of data, n_d the dimension of the initial (decoded) space and n_e the dimension of the reduced (encoded) space.

One of the first methods that we discussed when speaking about dimensionality reduction is **principal component analysis (PCA)**.



Point	Initial	Encoded	Decoded
A	(-0.50, -0.40)	-0.63	(-0.54, -0.33)
B	(-0.40, -0.10)	-0.39	(-0.34, -0.20)
C	(0.10, 0.00)	0.09	(0.07 0.04)
D	(0.30, 0.30)	0.41	(0.35, 0.21)
E	(0.50, 0.20)	0.53	(0.46, 0.27)

Back to top.



$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

Back to top.

Chapter 12 - Classification Metrics

Go back to main content, click here

Back to top.

https://www.youtube.com/YiqiaoYin

Loss functions are important for us to train our models. In both regressors and classifiers, we have discussed a number of loss functions: least square loss, mean square error, multinomial likelihood or cross-entropy, and so on.

In the end of the experiment, we also use performance metrics to report the final "accuracy" or how accurate our model is. For quantitative outcome variable or continuous outcome variable, there is not a whole lot of choices other than least square or similar form. For qualitative outcome variable or discrete (or categorical) outcome variable, this chapter covers the following terminologies:

- **Accuracy**
- **Specificity**
- **Sensitivity**
- **ROC**
- **ROC AUC**

Back to top.

This chapter adds some additional performance measure metrics for this course material specifically targeting on classification problems.

Let us imagine a basic situation: suppose we have actual condition in our data to be Positive (P) or Negative (N) and the predicted condition Positive (PP) or Negative (PN). We can draw the following 2 by 2 **confusion table** or **confusion matrix**. This table can also be interpreted as hypothesis testing where type I and type II errors are defined.

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection

Back to top.

Given this confusion table or confusion matrix below, we can define the first fundamental metric called **accuracy**.

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}$$

Back to top.

Given this confusion table or confusion matrix below, we can define the first fundamental metric called **false positive rate or 1 - specificity**.

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection

$$\mathrm{FPR} = 1 - \mathrm{Specificity} = 1 - \frac{TN}{N}$$

Back to top.

Given this confusion table or confusion matrix below, we can define the first fundamental metric called **sensitivity**.

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection

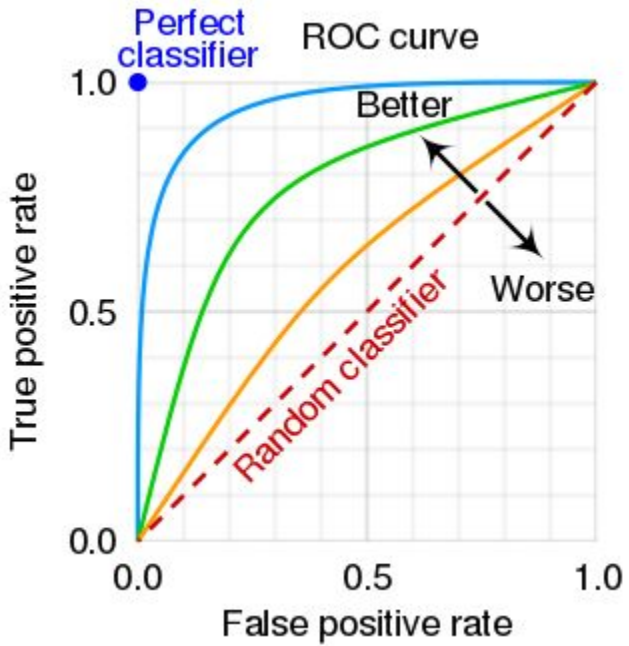
$$\text{Sensitivity} = \frac{\text{TP}}{\text{P}}$$

Back to top.

The **ROC curve** was first used during World War II for the analysis of radar signals before it was employed in signal detection theory. Following the attack on Pearl Harbor in 1941, the United States army began new research to increase the prediction of correctly detected Japanese aircraft from their radar signals. For these purposes they measured the ability of a radar receiver operator to make these important distinctions, which was called the Receiver Operating Characteristic.

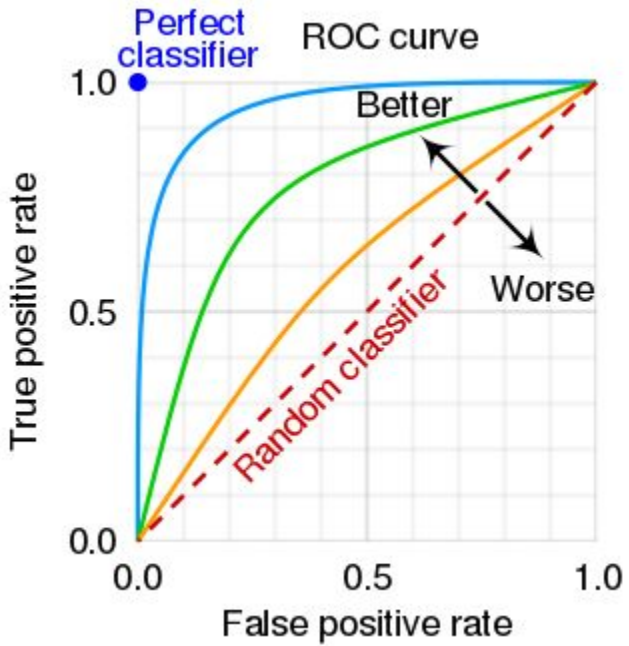
In the 1950s, ROC curves were employed in psychophysics to assess human (and occasionally non-human animal) detection of weak signals. In medicine, ROC analysis has been extensively used in the evaluation of diagnostic tests. ROC curves are also used extensively in epidemiology and medical research and are frequently mentioned in conjunction with evidence-based medicine. In radiology, ROC analysis is a common technique to evaluate new radiology techniques. In the social sciences, ROC analysis is often called the ROC Accuracy Ratio, a common technique for judging the accuracy of default probability models. ROC curves are widely used in laboratory medicine to assess the diagnostic accuracy of a test, to choose the optimal cut-off of a test and to compare diagnostic accuracy of several tests.

ROC curves also proved useful for the evaluation of machine learning techniques. The first application of ROC in machine learning was by Spackman who demonstrated the value of ROC curves in comparing and evaluating different classification algorithms.



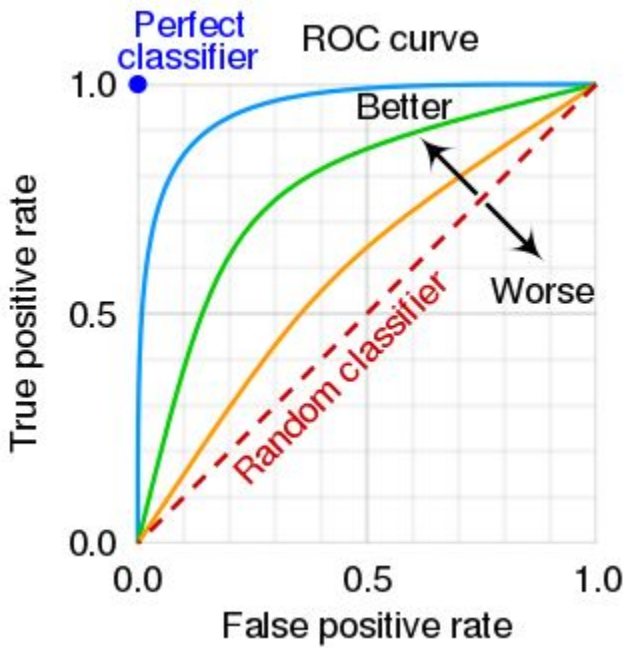
Back to top.

The vertical axis is true positive rate and it increases as true positive increase.



Back to top.

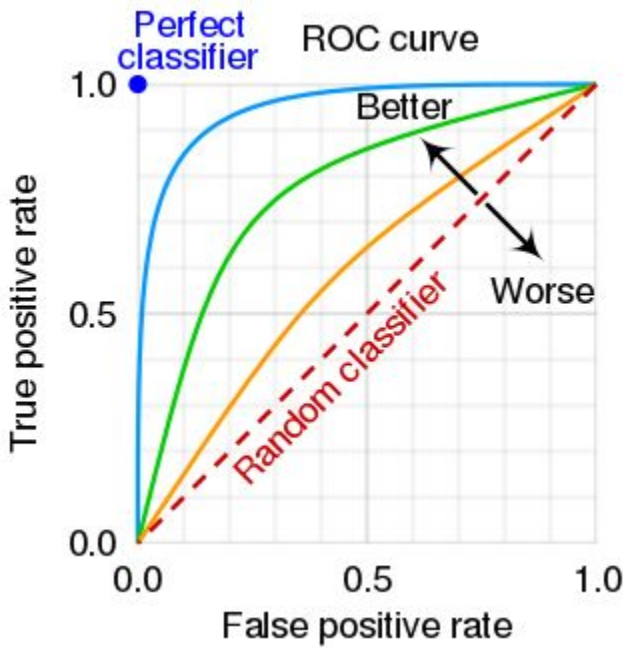
The vertical axis is true positive rate and it increases as true positive increases.



The horizontal axis is false positive rate which is 1 - specificity. Specificity increases as true negative increases. Hence, decrease of false positive rate would increase specificity.

Back to top.

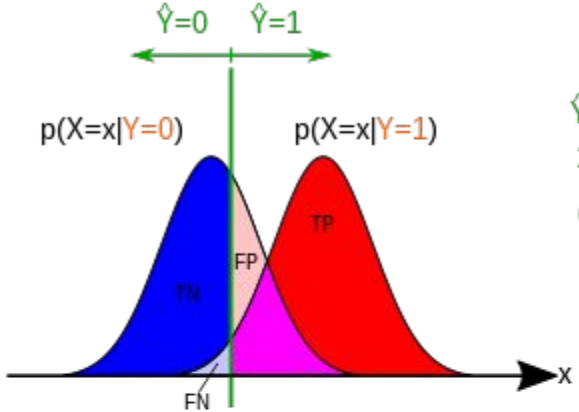
The vertical axis is true positive rate and it increases as true positive increases.



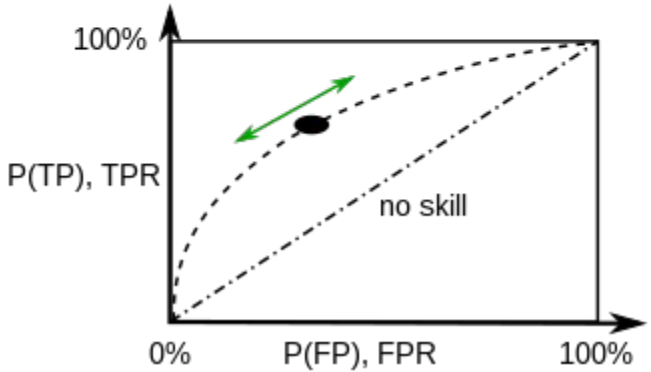
The perfect classifier would fall on the top left corner of the grid. This is a 100% all over the places, which means all guesses are correct (no type I or II errors). The red dash line is random classifier, which means all numbers are random. Everything else fall in between.

The horizontal axis is false positive rate which is 1 - specificity. Specificity increases as true negative increases. Hence, decrease of false positive rate would increase specificity.

Back to top.

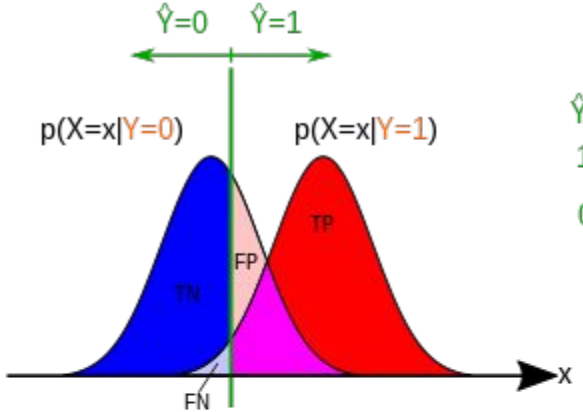


Ŷ \ Y	1	0
1	TP	FP
0	FN	TN

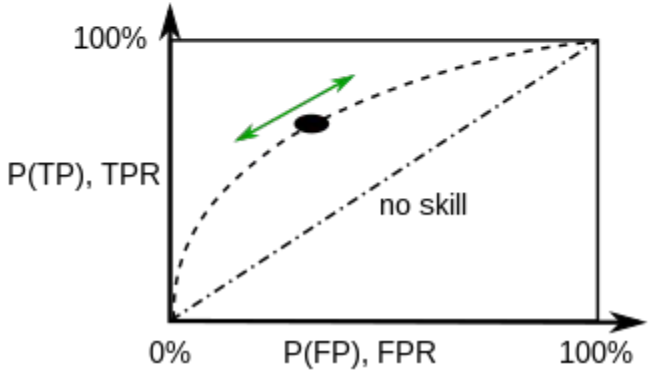


Back to top.

In hypothesis testing, we use the two distribution plot to investigate the power of the test.

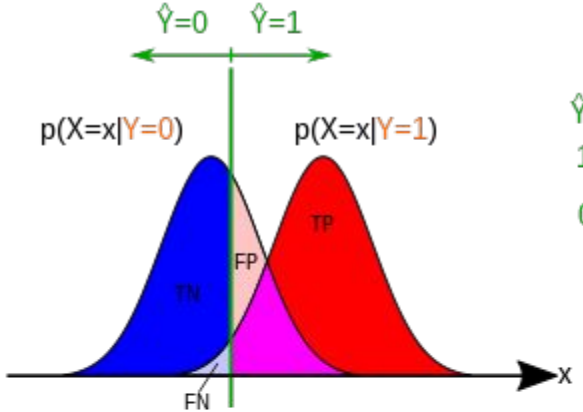


Ŷ \ Y	1	0
1	TP	FP
0	FN	TN



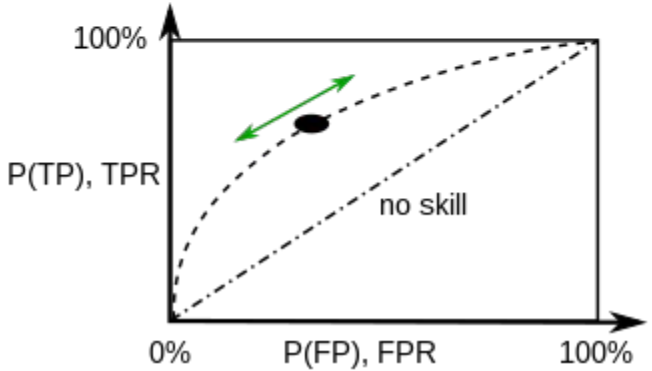
Back to top.

In hypothesis testing, we use the two distribution plot to investigate the power of the test.



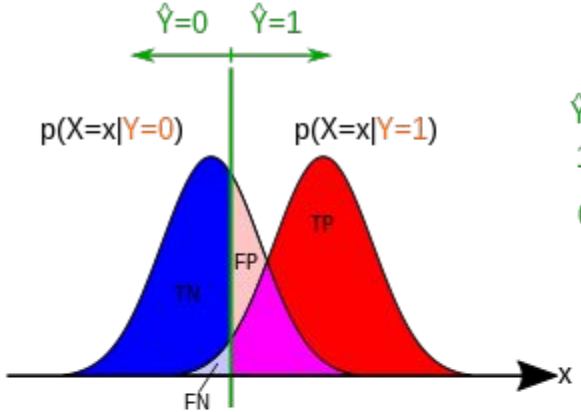
Ŷ \ Y	1	0
1	TP	FP
0	FN	TN

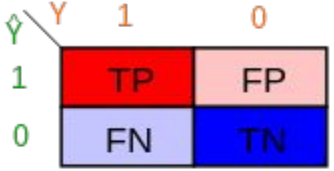
In classification tasks, we use confusion table or confusion matrix to understand accuracy, sensitivity, or specificity.



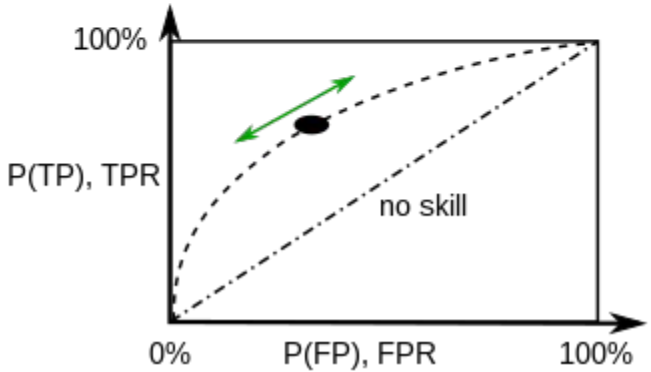
Back to top.

In hypothesis testing, we use the two distribution plot to investigate the power of the test.





In classification tasks, we use confusion table or confusion matrix to understand accuracy, sensitivity, or specificity.



When we have unbalanced data, guessing all 1s or 0s would give high accuracy though it doesn't imply good model. Hence, we use ROC AUC to accommodate this.

Back to top.

Last topic: F1-score

In statistical analysis of binary classification, the **F-score** or **F-measure** is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive. Precision is also known as positive predictive value, and recall is also known as sensitivity in diagnostic binary classification.

The $\mathbf{F_1}$ score is the harmonic mean of the precision and recall.

Back to top.

Last topic: F1-score

In statistical analysis of binary classification, the **F-score** or **F-measure** is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive. Precision is also known as positive predictive value, and recall is also known as sensitivity in diagnostic binary classification.

The $\mathbf{F_1}$ score is the harmonic mean of the precision and recall, which is $2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$

Last topic: F1-score

In statistical analysis of binary classification, the **F-score** or **F-measure** is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive. Precision is also known as positive predictive value, and recall is also known as sensitivity in diagnostic binary classification.

The $\mathbf{F_1}$ score is the harmonic mean of the precision and recall, which is $2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

But what exactly does this mean?

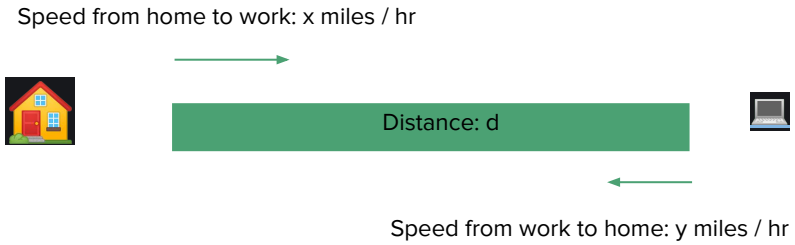
Back to top.

Last topic: F1-score

In statistical analysis of binary classification, the **F-score** or **F-measure** is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive. Precision is also known as positive predictive value, and recall is also known as sensitivity in diagnostic binary classification.

The $\mathbf{F_1}$ score is the harmonic mean of the precision and recall, which is $2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

But what exactly does this mean? Suppose we look at the following example. Say the goal is to compute <u>the average speed</u>.



To compute the average speed, we need to consider going to work and also coming back from work.

In other words, we are looking at the harmonic mean of both the speed of going to work and coming back from work.

Back to top.

Last topic: F1-score

In statistical analysis of binary classification, the **F-score** or **F-measure** is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive. Precision is also known as positive predictive value, and recall is also known as sensitivity in diagnostic binary classification.

The $\mathbf{F_1}$ score is the harmonic mean of the precision and recall, which is $2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

But what exactly does this mean? Suppose we look at the following example. Say the goal is to compute the average speed.

Speed from home to work: x miles / hr

Speed from work to home: y miles / hr

$$\begin{aligned}
\text{ave speed} &= \frac{\text{distance}}{\text{time}} \\
&= \frac{2d}{d/x+d/y} \\
&= \frac{2d}{(dy+dx)/xy} \\
&= \frac{2dxy}{d(y+x)} \\
&= \frac{2xy}{x+y}
\end{aligned}$$

Back to top.