강원대학교
AI 소프트웨어학과

머신러닝2

- 지도학습 –

기계학습의 회귀분석

KNU
강원대학교

통계적 방법과 기계학습 방법의 차이

기계학습의 방법(지도학습)

- 다중 회귀분석(Regression)
- 로지스틱 회귀분석(Logistic Regression)
- 신경망(Atificial Neural Network)
- 서포트 백터 머신(Support Vector Machine)
- 의사결정나무(Decision Tree)
- 앙상블(Ensemble)
- K-근접 이웃기법(k-Nearest Neighbor)

통계적 방법과 기계학습 방법의 차이

통계의 방법(회귀분석)

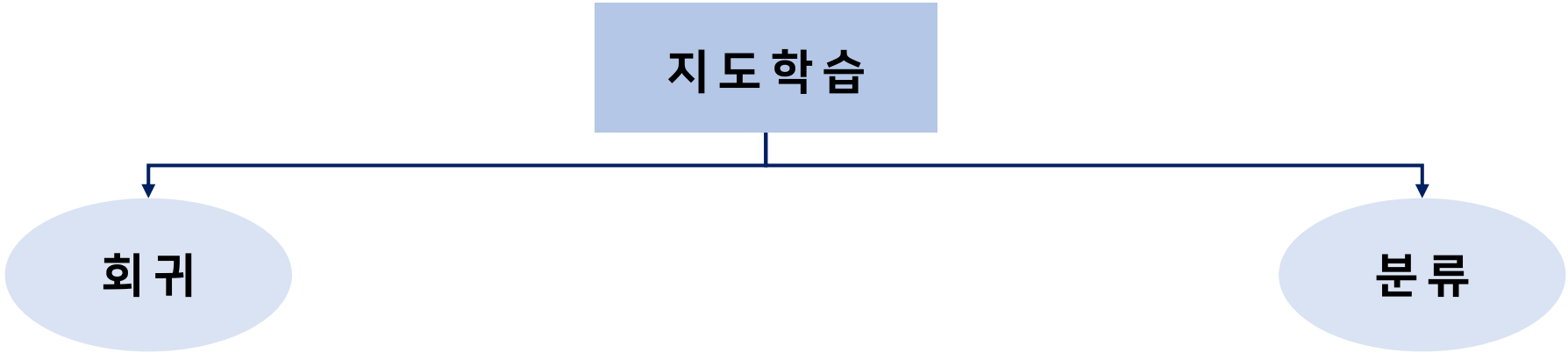
- 데이터 : 표본 데이터(소수)
- 가정 : 통계적 가정 필요, 모수통계(정규분포), 비모수통계, 회귀분석 가정(선형성, 등분산성, 독립성 등)
- 방법 : 가설검정(귀무가설, 대립가설)
- 검정 : 유의수준(p-value)
- 학습 방법 : 모든 데이터를 사용

기계학습의 방법

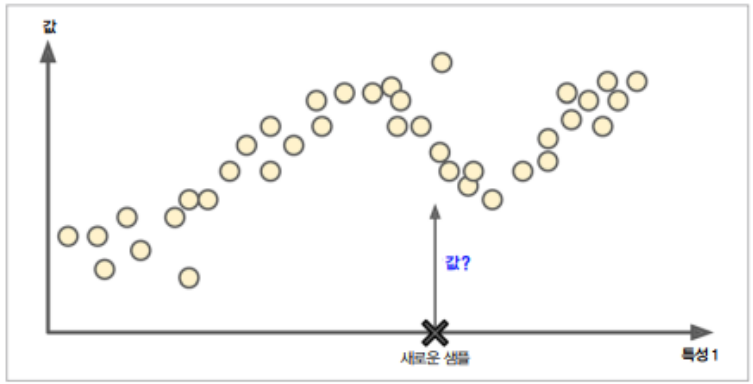
- 데이터 : 대용량 데이터(빅데이터)
- 가정 : 없음
- 방법 : 데이터의 특징을 찾아 스스로 학습
- 검정 : 예측의 정확도(Accuracy)
- 학습 방법 : Training, Validation, Test로 구분해 사용

01 지도학습

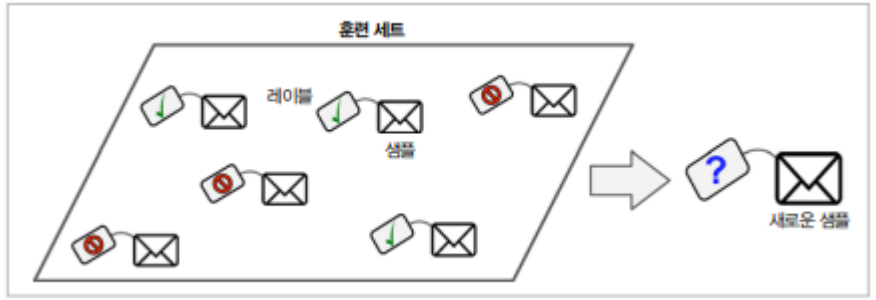
통계적 방법과 기계학습 방법의 차이



- 연속적인 값 예측
- 회귀분석



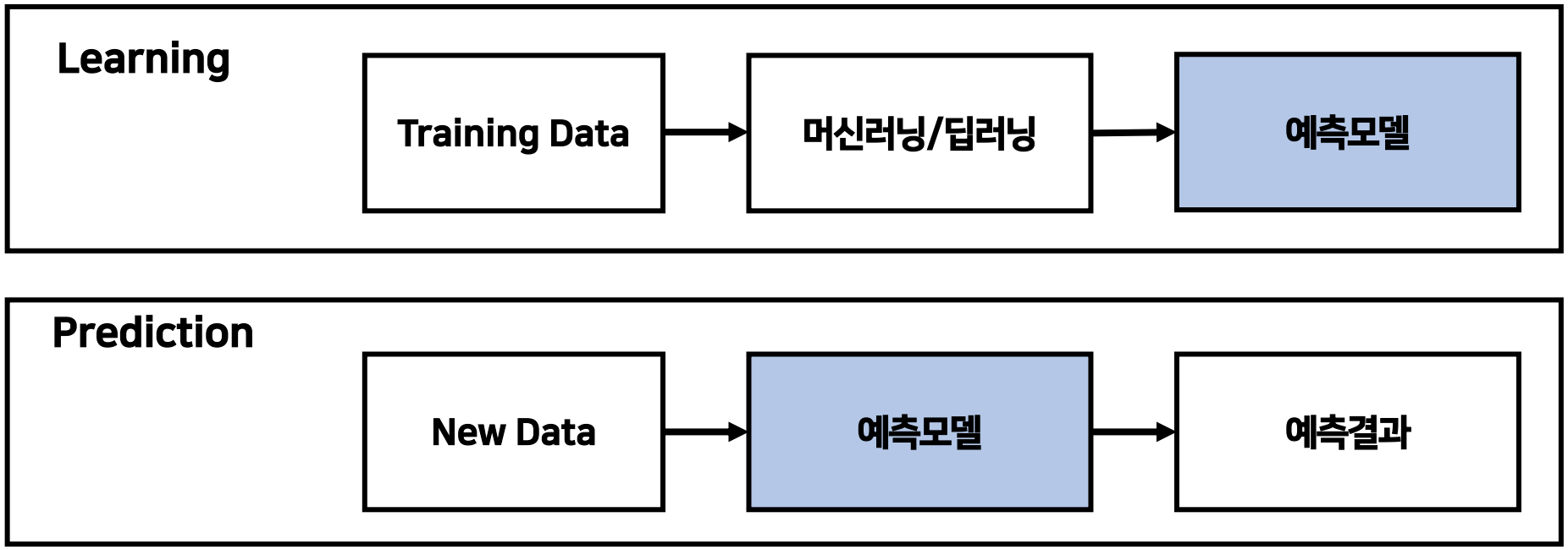
- 클래스 예측
- 확률적 경사 하강법, 결정 트리, 랜덤 포레스트, KNN, 나이브 베이즈, SVM 등



기계학습의 학습 방법

데이터의 수에 따라서 다름

- 회귀분석에서 우리가 이 모델이 얼마나 정확한지 판단할 수 없고 가정만 가능
- 기계학습의 경우 테스트를 통해 정확도를 검증을 할 수 있음



다중회귀 회귀(Multiple Linear Regression)

<단순선형회귀모형>

모형 : $y_i = \beta_0 + \beta_1 x_{i1} + \varepsilon_i$

가정 : $\varepsilon_i \sim iidN(0, \quad \sigma^2)$

<다중 선형회귀모형>

모형: $y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i$

가정: $\varepsilon_i \sim iidN(0, \quad \sigma^2)$

최소제곱법은 회귀모형의

오차 제곱의 합 $SSE = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$ **을 최소로**

하는 회귀계수를 이들의 추정치로 하는 것

다중회귀 회귀(Multiple Linear Regression)

데이터 수에 따른 유의수준의 변화

	n= 200			n = 690			n= 1300		
	표준 계수	t	유의수준	표준 계수	t	유의수준	표준 계수	t	유의수준
(상수)		-7.732	.000		-14.775	.000		-18.620	.000
연면적	.099	2.951	.004	.103	5.041	.000	.112	7.390	.000
품질	.271	7.365	.000	.271	12.332	.000	.263	15.974	.000
상태	.109	4.141	.000	.142	8.245	.000	.133	10.356	.000
건축년도	.286	8.153	.000	.273	11.356	.000	.264	15.064	.000
리모델링년도	.054	1.726	.086	.065	3.252	.001	.059	3.997	.000
지하면적	.129	2.879	.004	.144	5.744	.000	.152	8.542	.000
차고면적	.120	3.973	.000	.094	5.180	.000	.105	7.813	.000
면적_1층	.280	6.144	.000	.304	11.435	.000	.295	15.421	.000
면적_2층	.346	11.533	.000	.340	18.391	.000	.339	24.493	.000
주거_2가구변경	-.014	-.596	.552	-.022	-1.470	.142	-.025	-2.286	.022
주거_듀플렉스	-.045	-1.844	.067	-.072	-4.828	.000	-.074	-6.585	.000
주거_타운젠트 바깥쪽	-.045	-1.697	.091	-.026	-1.469	.142	-.009	-.670	.503
주거_타운젠트 안쪽	-.078	-2.814	.005	-.032	-1.913	.056	-.038	-3.083	.002

다중회귀 회귀(Multiple Linear Regression)

다중 회귀분석

- 다수의 요소를 가지고 y를 예측하고 싶을 때, 이를 다중 선형 회귀분석 이라고 함

$$y = W_1 x_1 + W_2 x_2 + \cdots W_n x_n + b$$

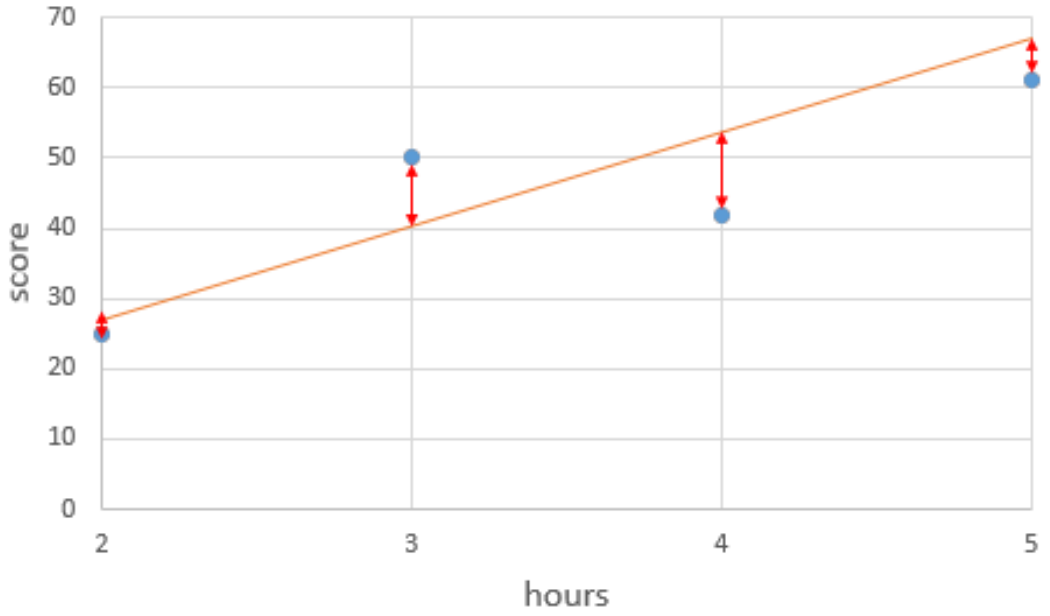
- 가설(Hypothesis) 공부 시간과 시험점수 간의 회귀 모델
- Y=가설(Hypothesis)를 가장 잘 표현 할 수 있는 임의의 선을 그려 가장 적절한 값을 찾음

$$H(W,b) = Wx + b \longrightarrow score = W(hours) + b$$

다중회귀 회귀(Multiple Linear Regression)

다중 회귀분석

- 가설(Hypothesis)에 대한 최적의 W와 b를 찾아
- 실제값과 예측값에 대한 오차 식을 비용함수(Cost function)이라고 함
- 회귀 문제의 경우에는 주로 평균 제곱 오차(Mean Squared Error, MSE)사용



다중회귀 회귀(Multiple Linear Regression)

<단순선형회귀모형>

모형 : $y_i = \beta_0 + \beta_1 x_{i1} + \varepsilon_i$

가정 : $\varepsilon_i \sim iidN(0, \quad \sigma^2)$

<다중 선형회귀모형>

모형: $y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i$

가정: $\varepsilon_i \sim iidN(0, \quad \sigma^2)$

경사하강법은 회귀모형의(최소제곱법 (Mean Squared Error) - 구체적인 가정들이 필요 없음)

비용 함수 $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ **최소화로**

하는 반복적 접근 방식으로 회귀계수를 이들의 추정치로 하는 것

SSE는 데이터 세트의 모든 관측치에 대한 전체 오류를 제공
MSE는 관측치 수에 걸쳐 총 오류를 평균화

다중회귀 회귀(Multiple Linear Regression)

비용 함수(Cost function) - 평균 제곱 오차(MSE)

- 가설(Hypothesis)에서 세워진 식을 통해 예측값을 도출함
- 오차=실제값 - 예측값 →음수, 양수 모두를 포함하고 있으므로 제곱해 더함
- MSE를 최소로 만드는 W와 b를 찾아서 회귀분석 식을 도출함

hours(x)	2	3	4	5
실제값	25	50	42	61
예측값	27	40	53	66
오차	-2	10	-7	-5

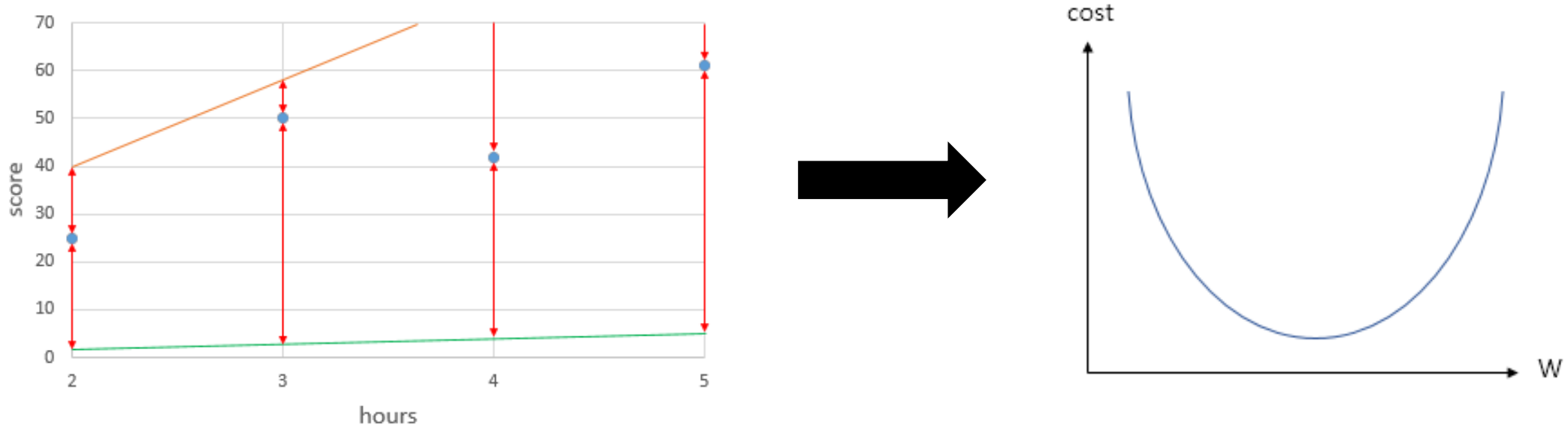
$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = (-2)^2 + 10^2 + (-7^2) + (-5^2) = 178$$

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{178}{4} = 44.5$$

다중회귀 회귀(Multiple Linear Regression)

Optimizer - backward

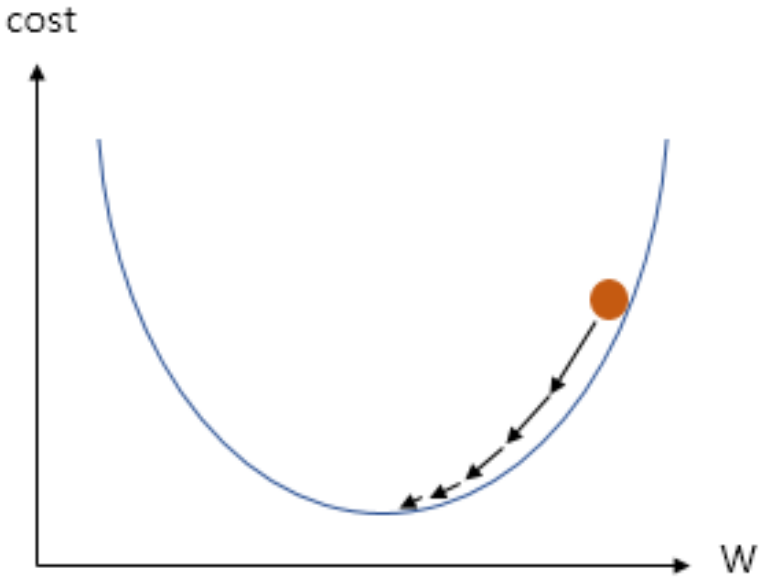
- Cost function을 최소화하는 매개 변수인 W와 b를 찾기 위한 작업을 할 때 사용되는 알고리즘을 Optimizer이라고 함
- 이를 머신러닝에서 학습이라고 부름
- 가장 기본적인 Optimizer 방법이 경사 하강법(Gradient Descent)
- Y=Wx+b에서 W의 크기가 지나치게 높거나 낮을 때, 오차가 커지는 것을 알 수 있음

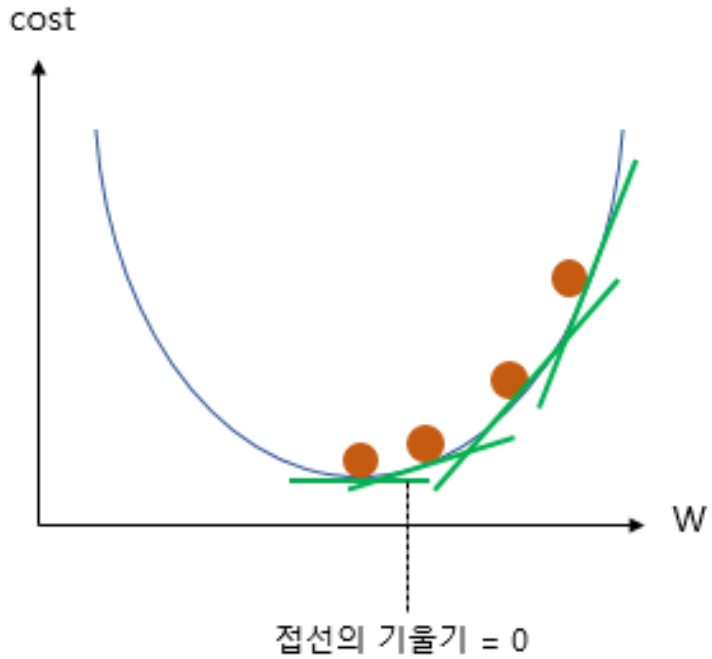


다중회귀 회귀(Multiple Linear Regression)

Optimizer – backward

- 기울기 W가 무한대로 커지면 커질수록 cost값 또한 무한대로 커지고, W가 무한대로 작아져도 cost값이 무한대로 커짐
- Cost가 가장 최소값을 가지게 하는 W를 찾아야 하는 것이 목적이므로 볼록한 맨 아래 부분의 W값을 찾아야 함
- 임의의 랜덤값 W를 정하고 볼록한 부분으로 향해 점차 W 값을 수정해감
- 접선의 기울기가 이 되는 지점이 Cost가 최소화가 되는 지점





다중회귀 회귀(Multiple Linear Regression)

Optimizer - 학습률(learning rate)

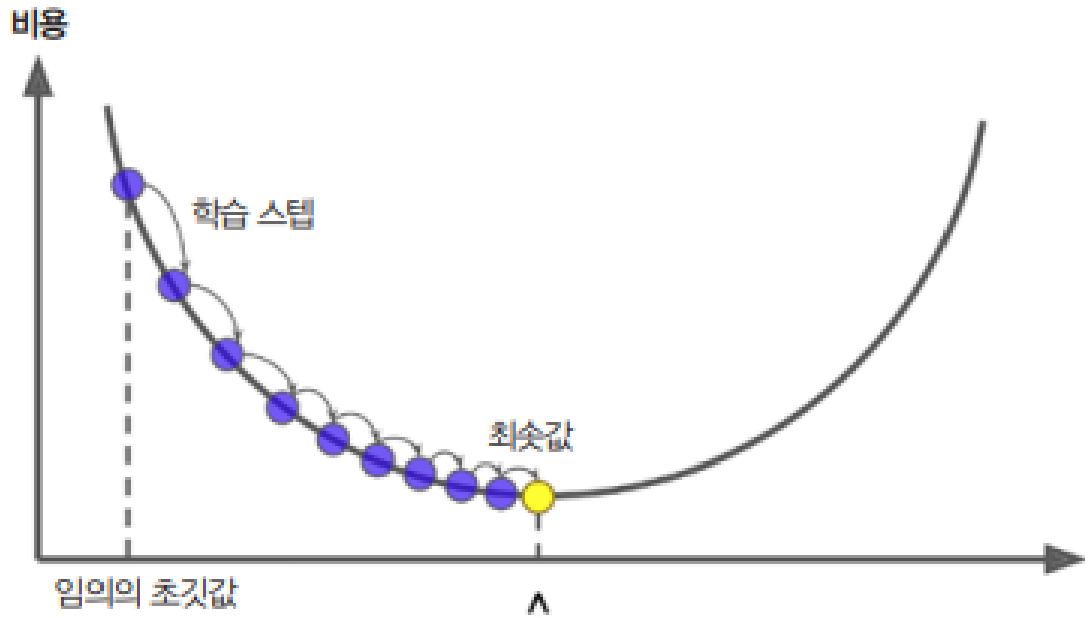
- W의 값을 변경할 때, 얼마나 크게 변경할지를 결정하는 값
- 접점의 기울기가 0인 지점을 찾는 것에 있어 어떤 크기의 폭으로 이동할지를 결정함

$$W := W - \alpha \frac{\partial}{\partial W} Cost(W) \qquad b := b - \alpha \frac{\partial}{\partial b} Cost(b)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}$$

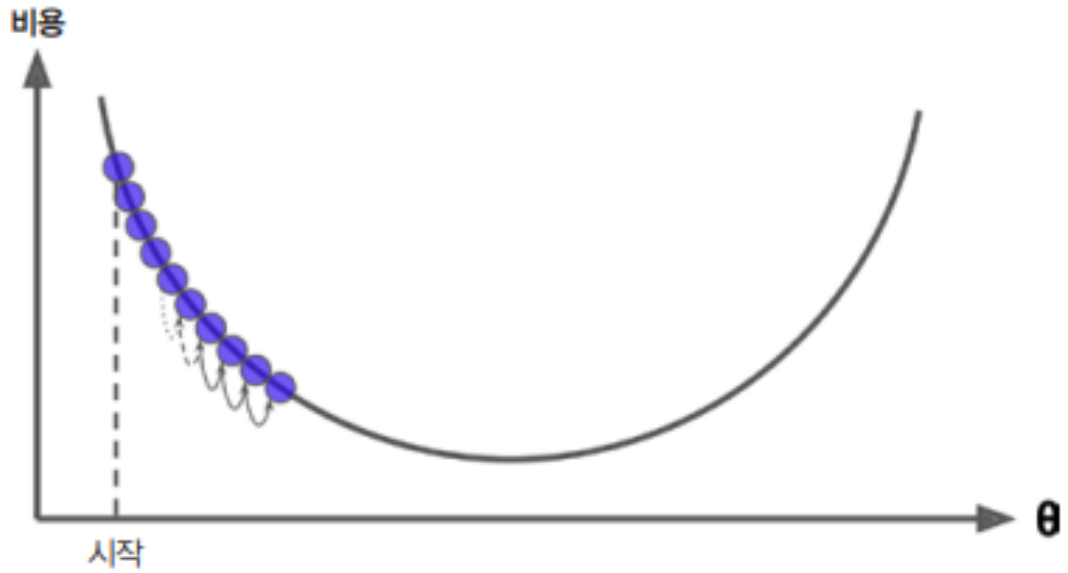
$$\frac{\partial MSE}{\partial \beta_j} = \frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)x_{ij}$$

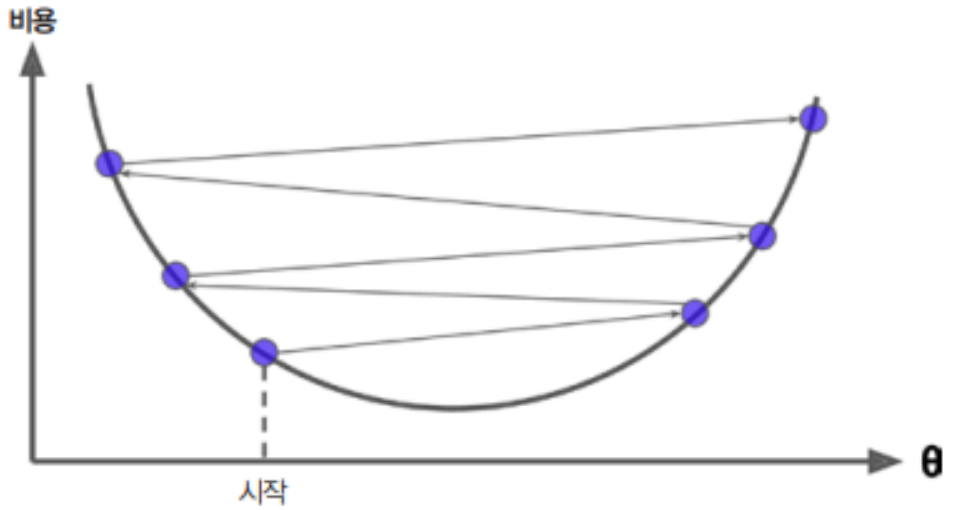


다중회귀 회귀(Multiple Linear Regression)

Optimizer - 학습률(learning rate)

- 학습률이 너무 작으면 알고리즘이 수렴하기 위해 반복을 많이 진행하므로 시간이 오래걸림
- 학습률이 너무 크면 골짜기를 가로질러 반대편으로 건너뛰게 되어 접선이 0이 되는 지점을 찾지 못함
- 학습 횟수(Epoch)를 통해 최적의 Low Cost를 찾는 것이 중요함

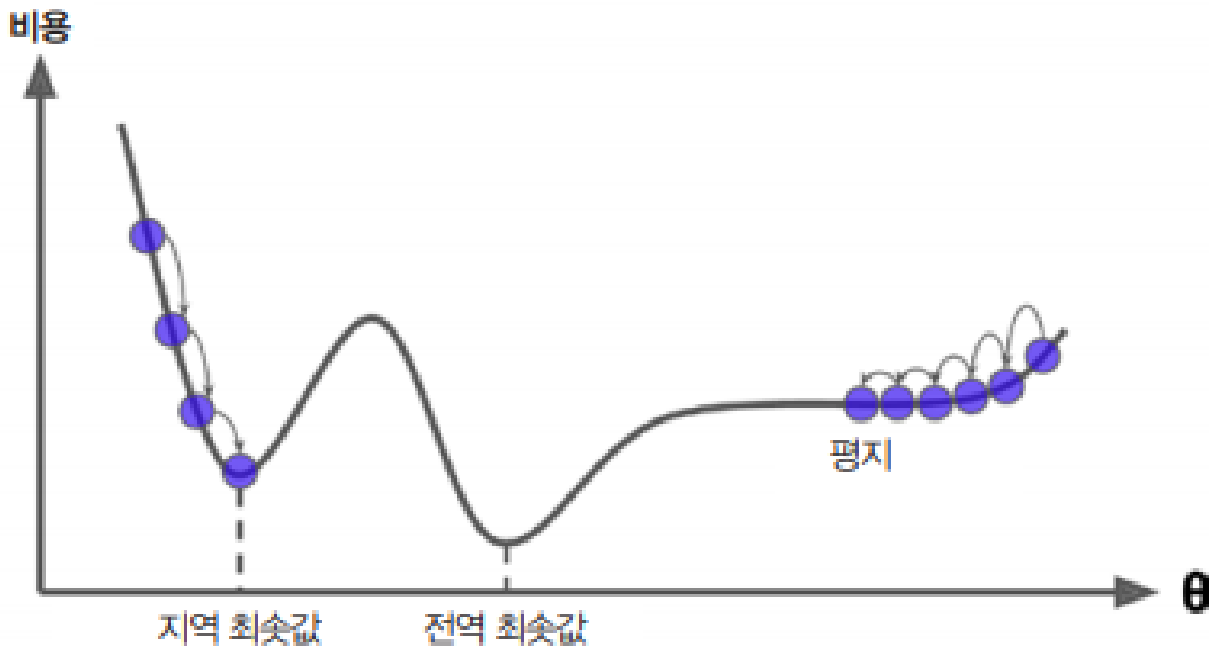




다중회귀 회귀(Multiple Linear Regression)

Optimizer - 최적의 Cost 결정의 문제

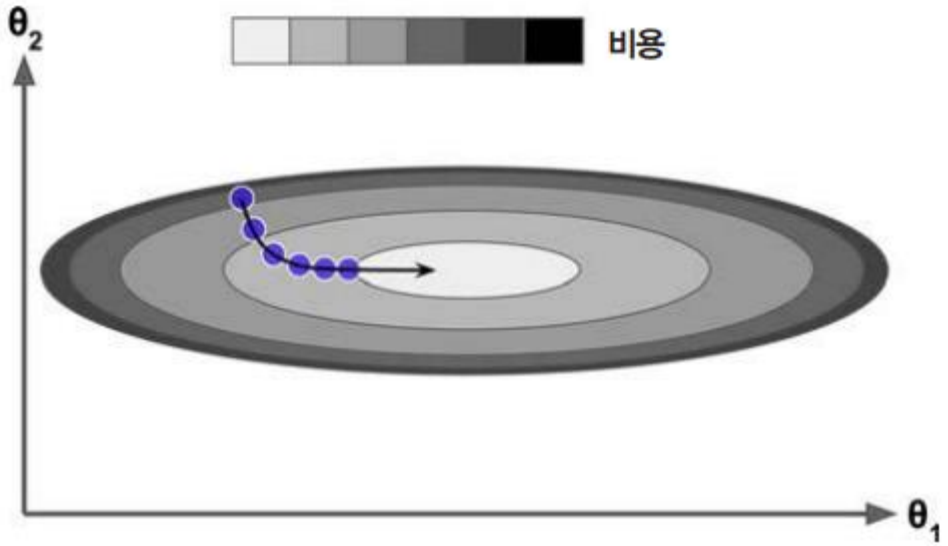
- 알고리즘이 왼쪽에서 시작하면, 전역 최소값보다 덜 좋은 지역 최소값에 수렴
- 오른쪽에 시작하면, 평탄한 지역을 지나기 위해 오랜 시간이 걸리고 일찍 멈추게 되어 전역 최솟값에 도달하지 못함



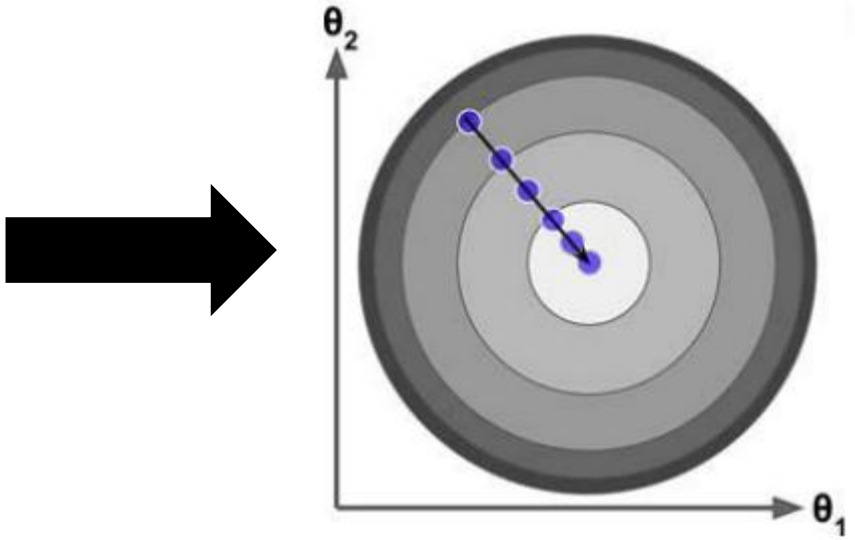
다중회귀 회귀(Multiple Linear Regression)

Optimizer - 변수 스케일의 크기의 문제

- 변수 1이 변수 2보다 스케일이 작은 훈련 데이터인 경우
- 표준화 진행을 통해 변수의 수준을 같게 변환해 학습의 시간을 줄여줌



표준화를 적용한 경사 하강법

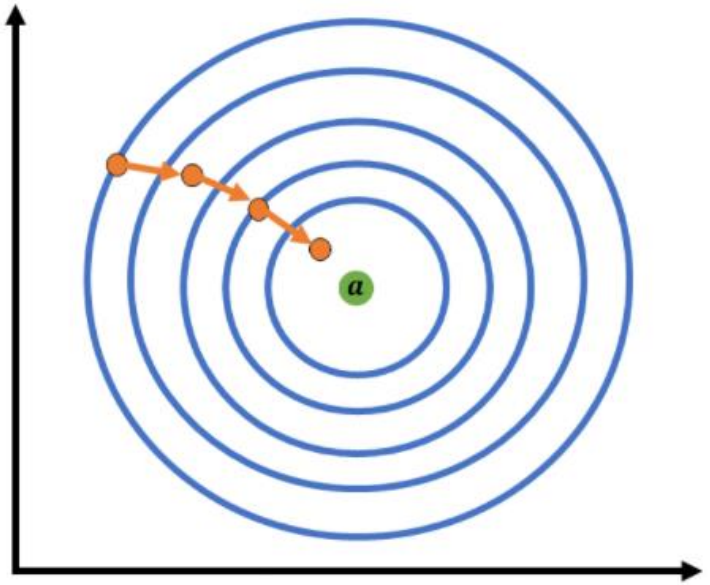


표준화를 적용한 경사 하강법

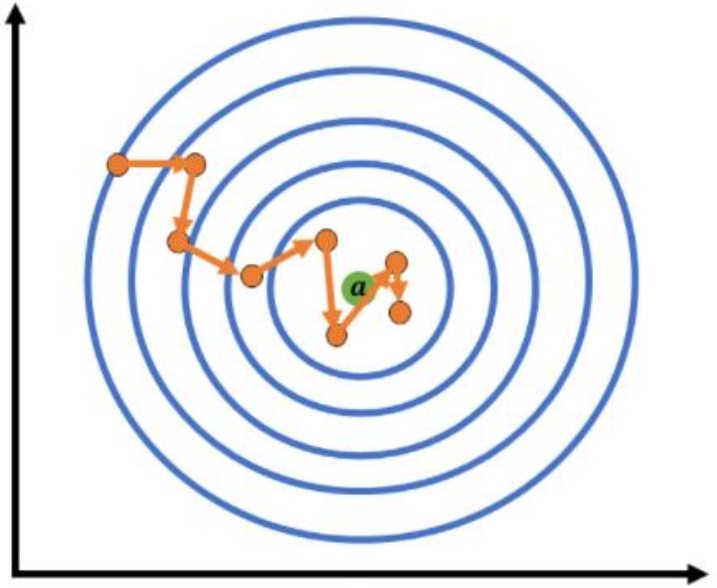
다중회귀 회귀(Multiple Linear Regression)

Optimizer - 변수 스케일의 크기의 문제

- 미니배치 경사 하강법 : 전체 데이터셋을 통해 W값을 업데이트 하는 방법
- 확률적 경사 하강법 : 하나의 데이터를 통해 W값을 구한 뒤 다음 데이터셋을 보며 W를 업데이트



미니배치 경사 하강법
(Batch Gradient Descent, BGD)

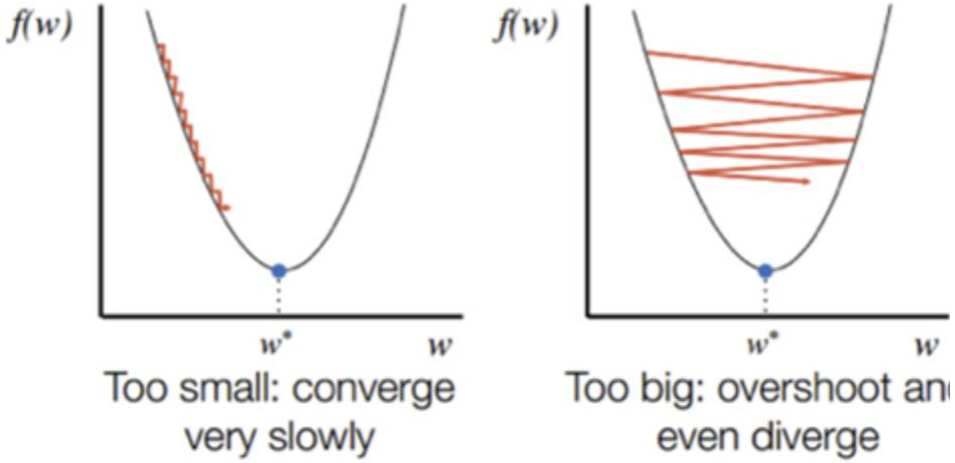


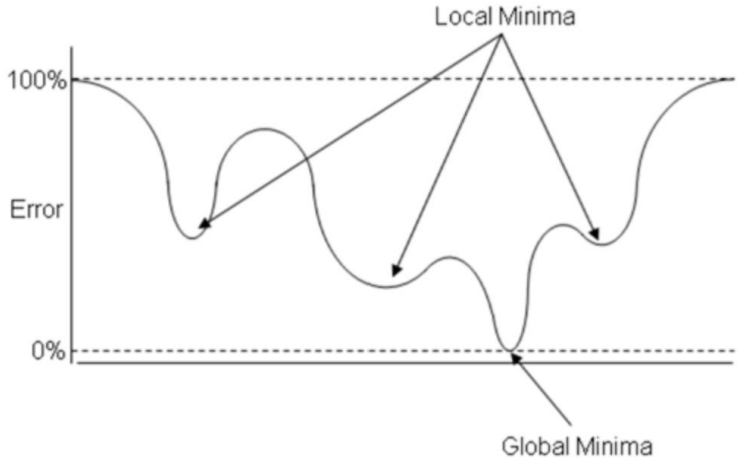
확률적 경사 하강법
(Stochastic Gradient Descent, SGD)

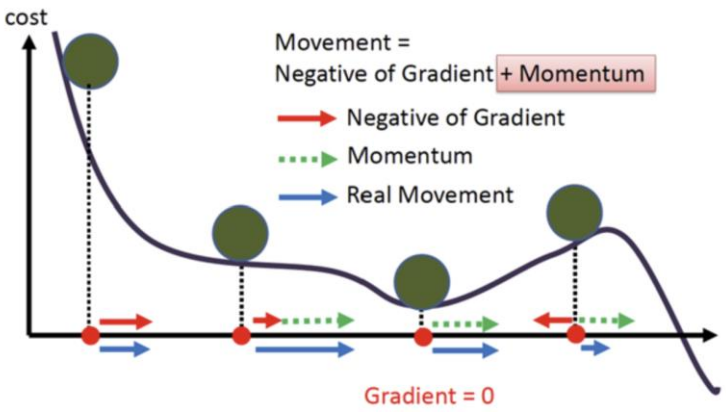
기계학습의 학습 방법

Cost function - 경사하강법(Gradient descent)

- 미분을 이용하여 기울기가 최소인 지점을 찾음
- 학습률(Learning Rate)
  - 새로운 정보를 얼마나 반영할지를 조절
  - 학습률(Learning Rate)을 이용해 점차적으로 변화량을 조절함
- 모멘텀(Momentum) : 어느 정도 기존의 방향을 유지할 것인지 조정







다중회귀 회귀(Multiple Linear Regression)

Cost function - 경사하강법의 프로세스(Gradient Descent)

1) 초기화: 최적화해야 하는 매개변수의 초기 값을 선택하는 것부터 시작함

→무작위로 설정되거나 일부 경험적 방법을 기반으로 설정함

2) 기울기 계산: 각 매개변수에 대한 비용 함수의 기울기를 계산

3) 매개변수 업데이트: 그라데이션 반대 방향으로 매개변수를 조정

4) 비용 함수가 최소값에 수렴할 때까지 2단계와 3단계를 반복하며 업데이트

$$\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \varepsilon_i$$

$$\hat{y}_i = w_0 + w_1 x_{i1} + \varepsilon_i$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$Cost = J(\beta_0, \beta_1) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2 \rightarrow \frac{\partial MSE}{\partial \beta_1} = -\frac{2}{n}(y_i - \hat{y}_i)\cdot x_{i1}$$

다중회귀 회귀(Multiple Linear Regression)

Cost function - 경사하강법의 프로세스(Gradient Descent)

비용함수 : 예측 오류 또는 손실에 대한 정량적 측정을 제공하여 모델 성능을 수치적으로 평가

→ 최적의 매개변수를 찾는 것을 목표

각 매개변수 β_j 가 비용 함수 J에 어떻게 영향을 미치는지 이해하는 것이 중요함

$$J(\beta_0, \beta_1, \dots \beta_k) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik}))^2$$

Step 1) 비용함수 정의

$$\frac{\partial J}{\partial \beta_0} = -\frac{2}{n}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik}))$$

Step 2) 매개변수 편미분

$$\frac{\partial J}{\partial \beta_j} = -\frac{2}{n}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik})) \cdot x_{ij}$$

$$\frac{\partial MSE}{\partial \beta_1} = -\frac{2}{n}(y_i - \hat{y}_i) \cdot x_{i1}$$

다중회귀 회귀(Multiple Linear Regression)

Cost function - 경사하강법의 프로세스(Gradient Descent)

비용함수 : 예측 오류 또는 손실에 대한 정량적 측정을 제공하여 모델 성능을 수치적으로 평가

→ 최적의 매개변수를 찾는 것을 목표

각 매개변수 β_j 가 비용 함수 J에 어떻게 영향을 미치는지 이해하는 것이 중요함

$$w_j^{(new)} = w_j^{(old)} - \alpha\left(\frac{\partial MSE}{\partial w_j}\right)$$

Step 3) 매개변수 업데이트

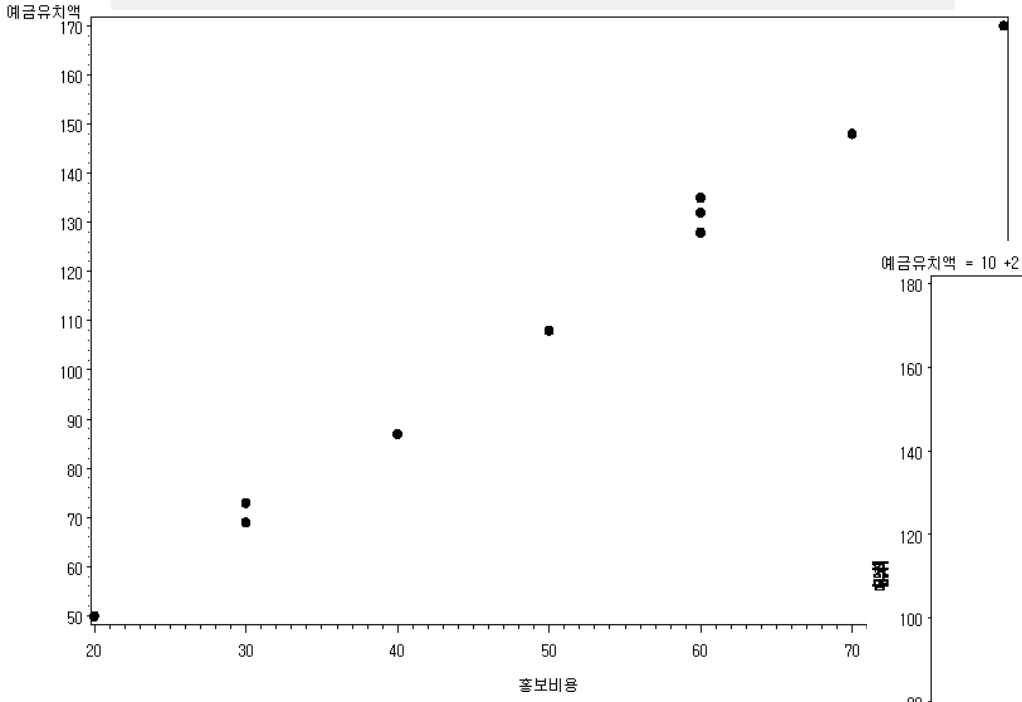
$$\frac{\partial MSE}{\partial w_1} = -\frac{2}{n}(y_i - \hat{y}_i) \cdot x_{i1}$$

Step 4) Step 2~3반복

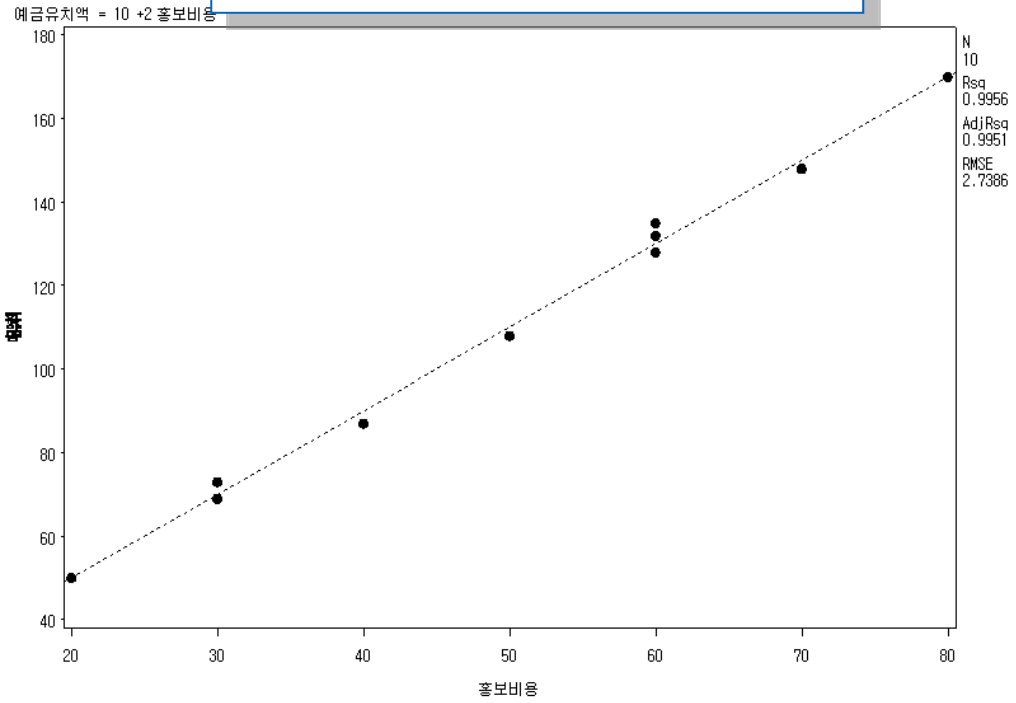
다중회귀 회귀(Multiple Linear Regression)

$$y = \beta_0 + \beta_1 x_1 + \varepsilon, \quad y = 예금유치액, x_1 = 홍보비용$$

홍보비용과 예금유치액에 대한 산점도



$$\hat{y} = 10 + 2 \times 홍보비용$$



다중회귀 회귀(Multiple Linear Regression)

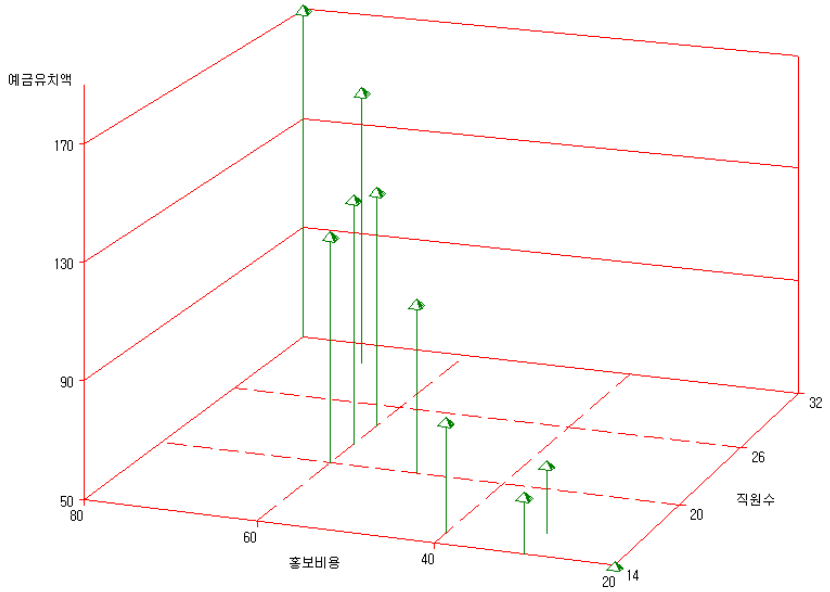
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$$

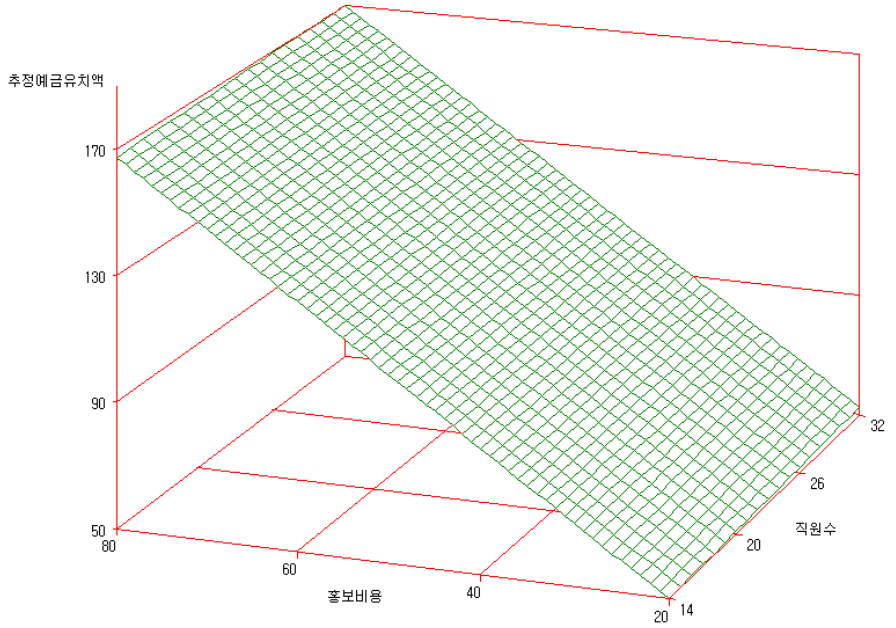
y = 예금유치액, x_1=홍보비용, x_2=직원수

홍보비용, 직원수 그리고 예금유치액에 대한 산점도

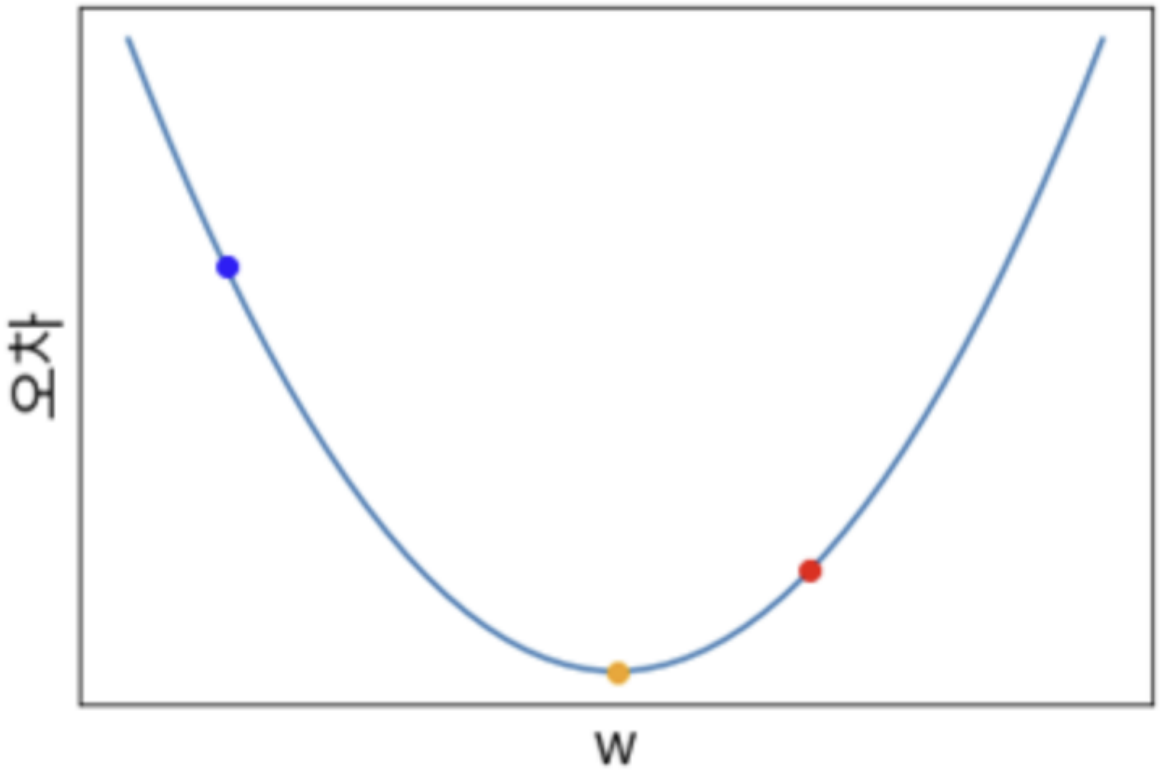
추정된 다중회귀모형

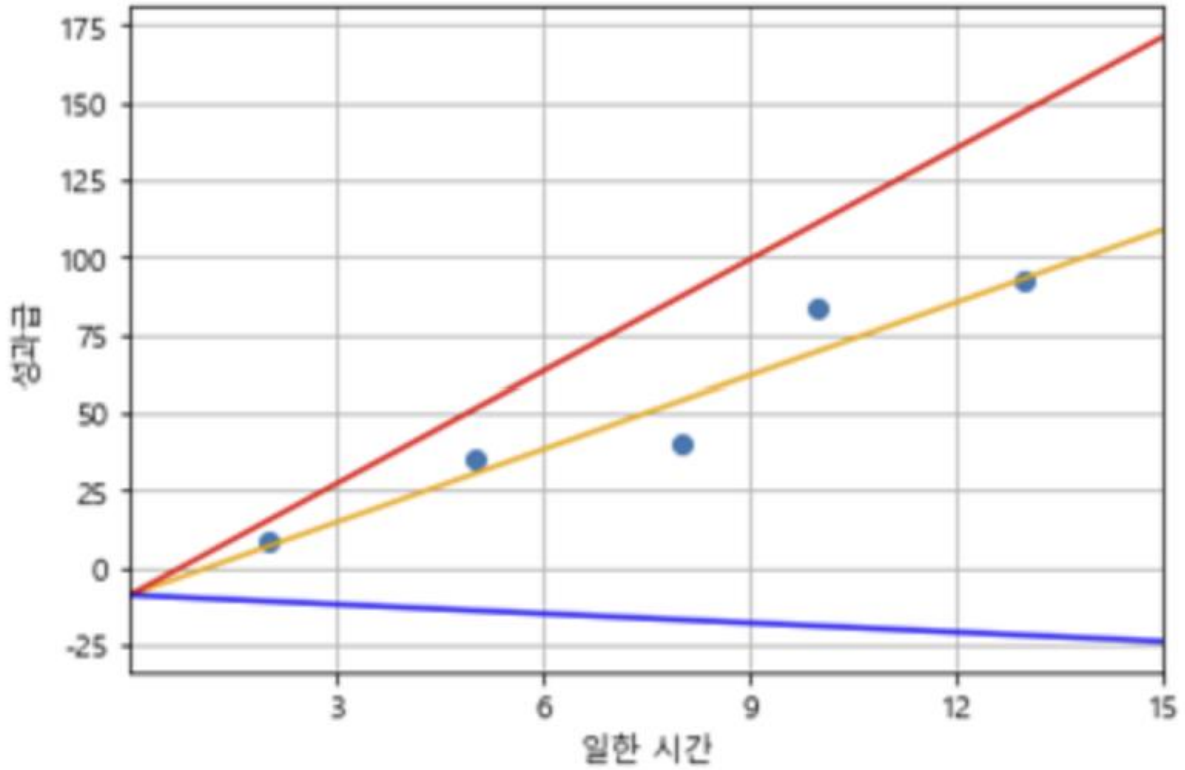
$$\hat{y} = 9.23 + 1.95 \times \text{홍보비용} + 0.15 \times \text{직원수}$$





다중회귀 회귀(Multiple Linear Regression)





다중회귀 회귀(Multiple Linear Regression)

```
library(MASS)
library(car)
library(skimr)

data("Boston")
df <- Boston
skim(df)

initial_predictors <- c("lstat", "rm", "age", "dis", "rad", "tax", "ptratio", "indus", "nox",
"crim")

result <- lm(df$medv ~ ., data = df[,initial_predictors])
vif_values <- vif(result)

print(vif_values)

selected_predictors <- names(vif_values[vif_values < 7])
print(selected_predictors)
```

다중회귀 회귀(Multiple Linear Regression)

```
library(caret)
set.seed(123)
train_indices <- createDataPartition(df$medv, p = 0.8, list = FALSE)
df_train <- df[train_indices, ]
df_test <- df[-train_indices, ]

X_train <- df_train[, selected_predictors]
y_train <- df_train$medv

X_test <- df_test[, selected_predictors]
y_test <- df_test$medv

X_train <- cbind(1, as.matrix(X_train)) #절편 항 추가
y_train <- as.numeric(y_train)

X_test <- cbind(1, as.matrix(X_test)) #절편 항 추가
y_test <- as.numeric(y_test)

# 모든 변수의 베타값을 0으로 초기화
beta <- rep(0, ncol(X_train))
```

다중회귀 회귀(Multiple Linear Regression)

```
gradient_descent <- function(X, y, beta, learning_rate, num_iterations) {
    m <- nrow(X)
    cost_history <- numeric(num_iterations)
    for (i in 1:num_iterations) {
    # 예측값 계산
    y_pred <- X %*% beta
    residuals <- y_pred - y
    gradient <- t(X) %*% residuals / m
    beta <- beta - learning_rate * gradient
    # 비용 함수 계산 및 저장
    cost <- sum(residuals^2) / (2 * m)
    cost_history[i] <- cost
        if (i %% 1000 == 0) {
      cat("Iteration", i, " | Cost:", cost, "\n")
    }
  }
  return(list(beta = beta, cost_history = cost_history))
}
```

다중회귀 회귀(Multiple Linear Regression)

```
mse_function <- function(beta, X, y) {
  # Calculate the predictions based on the current coefficients
  y_pred <- X %*% beta

  # Calculate the residuals (errors)
  residuals <- y_pred - y

  # Calculate the Mean Squared Error (MSE)
  mse <- sum(residuals^2) / nrow(X)

  # Return the MSE value
  return(mse)
}
```

다중회귀 회귀(Multiple Linear Regression)

```
# Hyper Parameters 정의
learning_rate <- 0.0001
num_iterations <- 10000

# 경사 하강법으로 도출한 Parameters 도출
gd_result <- gradient_descent(X_train, y_train, beta, learning_rate, num_iterations)
final_beta <- gd_result$beta
cost_history <- gd_result$cost_history

# 경사 하강법 그래프
plot(1:num_iterations, cost_history, type = "l", col = "blue", xlab = "Iteration", ylab =
"Cost", main = "Cost Function over Iterations")
```

다중회귀 회귀(Multiple Linear Regression)

```
# 행렬의 곱셈
y_pred <- X_test %*% final_beta

# 잔차 제곱합
ss_res <- sum((y_test - y_pred)^2)

# 총 제곱합
ss_total <- sum((y_test - mean(y_test))^2)

# 결정 계수
r2_test <- 1 - (ss_res / ss_total)

# MSE 및 RMSE 계산
mse_test <- mse_function(final_beta, X_test, y_test)
rmse_test <- sqrt(mse_test)

print(paste("Mean Squared Error(MSE) on Test Set:", round(mse_test, 3)))
print(paste("Root Mean Squared Error(RMSE) on Test Set:", round(rmse_test, 3)))
print(paste("R-squared (R²) on Test Set:", round(r2_test,3)))
```

다중회귀 회귀(Multiple Linear Regression)

```
# 학습 모델 저장
save(final_beta, selected_predictors, file = "final_model.RData")

# 모델 불러오기
load("final_model.RData")

# 새로운 데이터로 예측
df_test <- read.csv(“Boston_test.csv”, header=TRUE)
new_data <- df_test[, selected_predictors]

#모델의 입력값에 맞춰 형식변경
X_new <- cbind(1, as.matrix(new_data))
y_new_pred <- X_new %*% final_beta

# 예측 결과 출력
print(y_new_pred)
```

로지스틱 회귀(Logistic Regression)

연속형 종속변수가 아닌 범주형 종속변수일 경우?

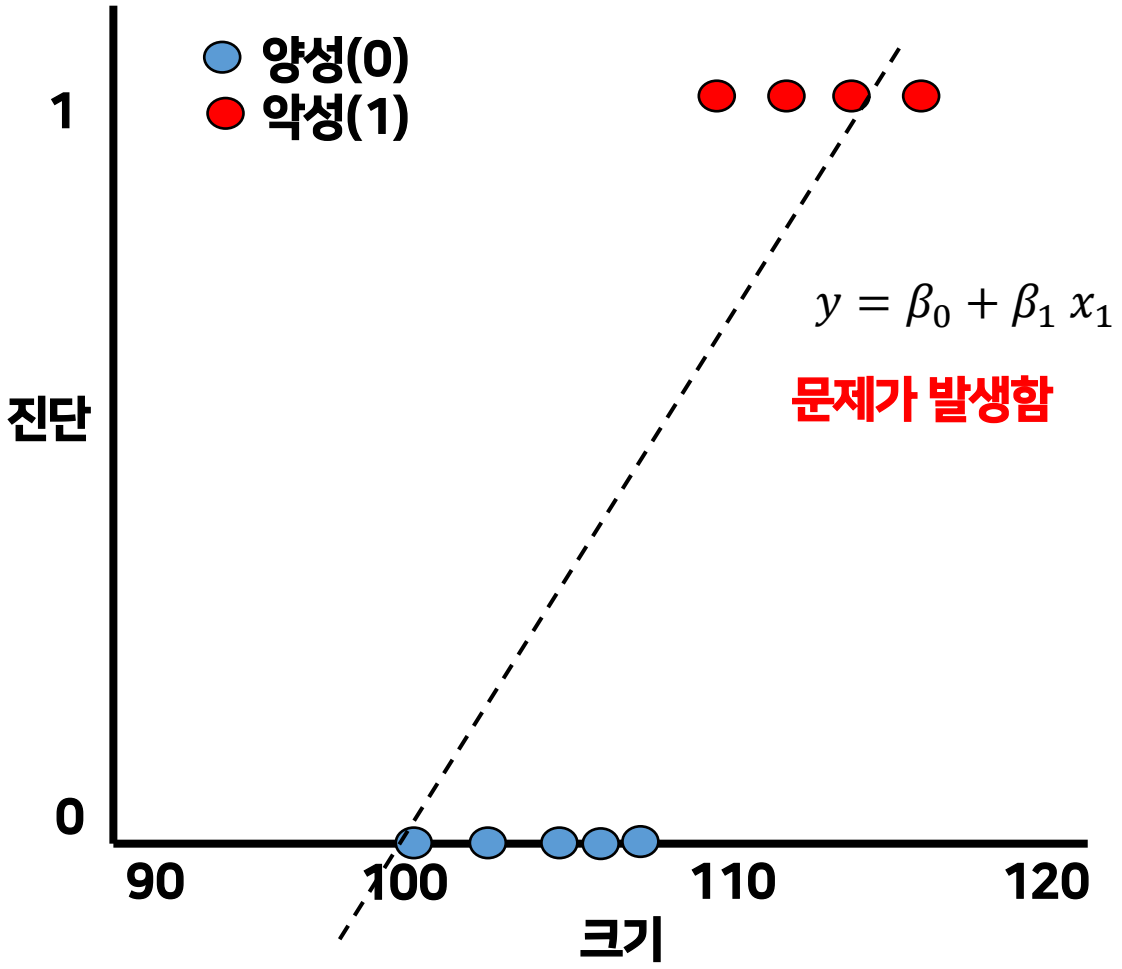
- **로지스틱 회귀분석(Logistic Regression) : 로지스틱 회귀 모델은 시그모이드 함수라고도 알려진 로지스틱 함수를 사용하여 예측값을 확률에 매핑해 0과 1로 도출함**
- **질병 진단(0 = 양성, 1 = 악성)**
- **강아지, 고양이 판단**

$$\sigma(z) = \frac{1}{1 + e^{-z}} \qquad z = \beta_0 + \beta_1 x_1 + \beta_n x_n + \cdots + \beta_n x_n$$

로지스틱 회귀(Logistic Regression)

로지스틱 회귀(변수가 1개)

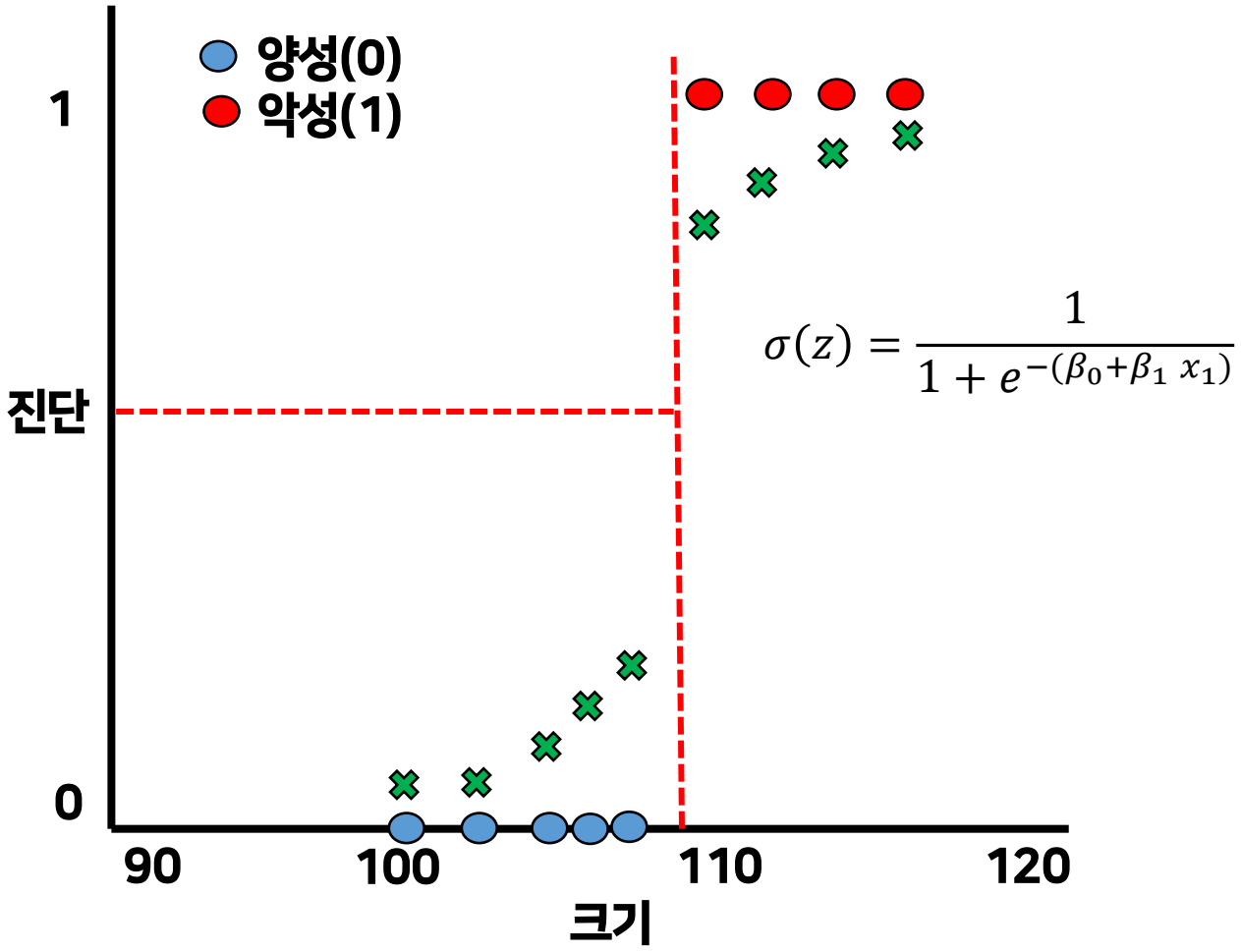
크기	진단
100	0
101	0
102	0
103	0
104	0
105	0
106	0
107	0
108	0
109	0
110	1
111	1
112	1
113	1
114	1



로지스틱 회귀(Logistic Regression)

로지스틱 회귀(변수가 1개)

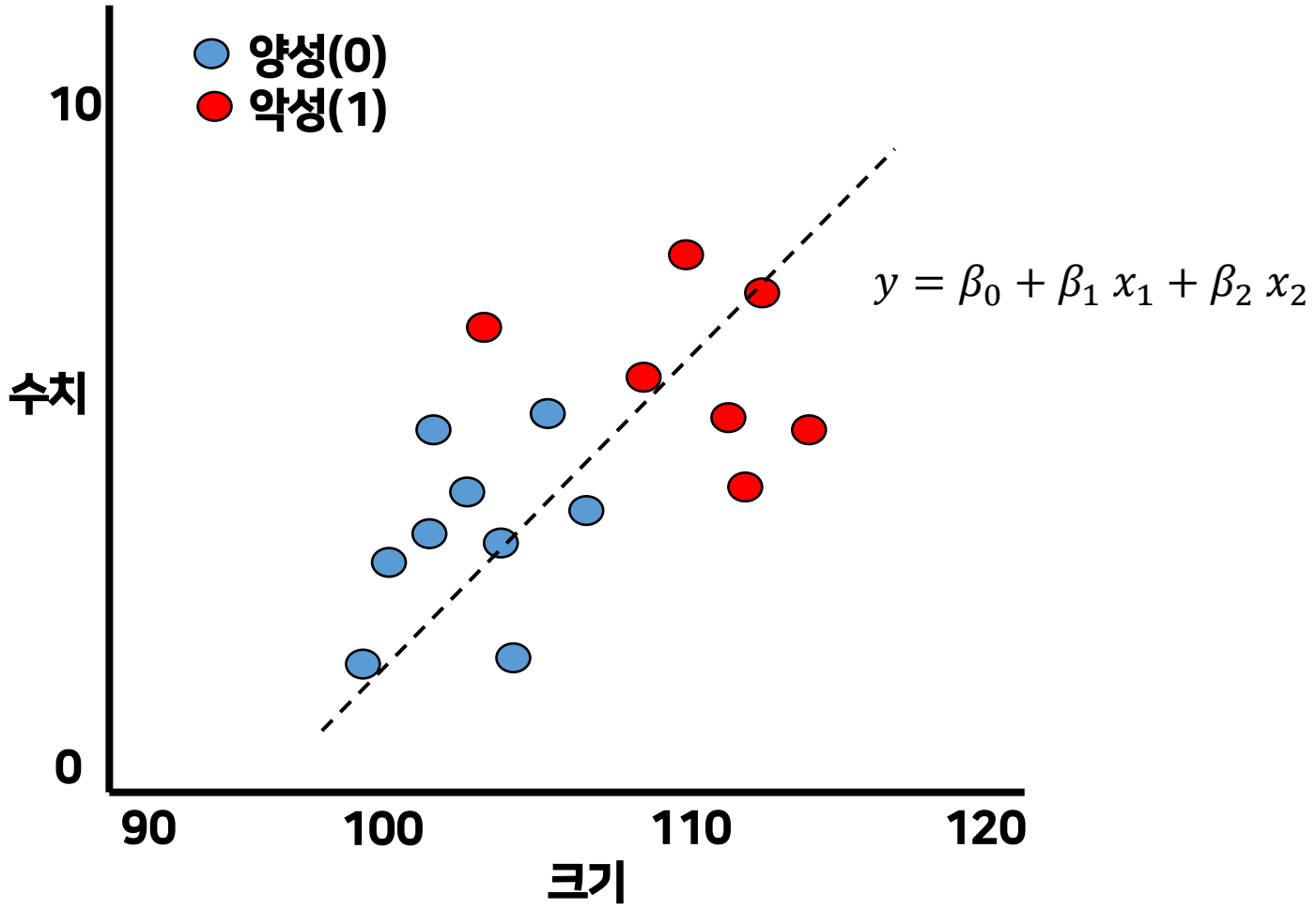
크기	예측확률	진단
100	0.005	0
101	0.007	0
102	0.009	0
103	0.01	0
104	0.02	0
105	0.05	0
106	0.1	0
107	0.4	0
108	0.5	0
109	0.6	0
110	0.65	1
111	0.7	1
112	0.75	1
113	0.8	1
114	0.9	1



로지스틱 회귀(Logistic Regression)

로지스틱 회귀(변수가 2개)

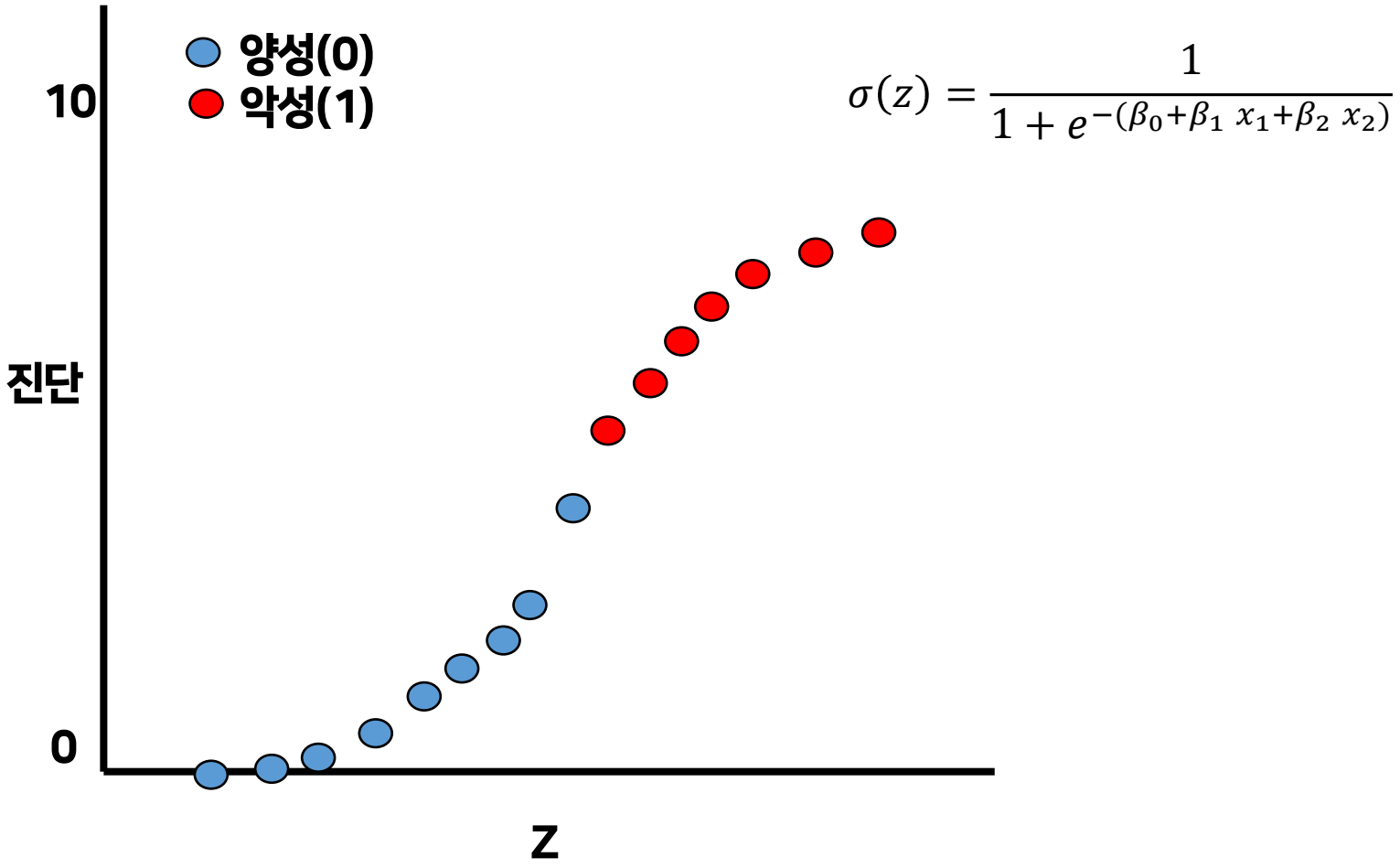
크기	수치	진단
100	5	0
101	6	0
102	7	0
103	4	0
104	5	0
105	7	0
106	4	0
107	6	0
108	8	0
109	5	0
110	8	1
111	5	1
112	7	1
113	9	1
114	8	1



로지스틱 회귀(Logistic Regression)

로지스틱 회귀(변수가 2개)

크기	수치	진단	예측확률
100	5	0	0.005
101	6	0	0.007
102	7	0	0.009
103	4	0	0.01
104	5	0	0.02
105	7	0	0.05
106	4	0	0.1
107	6	0	0.4
108	8	0	0.5
109	5	0	0.6
110	8	1	0.65
111	5	1	0.7
112	7	1	0.75
113	9	1	0.8
114	8	1	0.9



$$\sigma(z) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}}$$

로지스틱 회귀(Logistic Regression)

로지스틱 회귀분석

```
library(GGally)
library(skimr)
library(tidyverse)
library(tidymodels)
library(caret)

df <- read.csv("UniversalBank.csv", header = TRUE, na = ".")
skim(df)

df$Personal.Loan <- as.factor(df$Personal.Loan)
df$Personal.Loan <- factor(df$Personal.Loan, levels = c(0,1), labels = c("yes", "no"))

trainIndex <- createDataPartition(df$Personal.Loan, p = 0.8, list = FALSE)
train_data <- df[trainIndex,]
test_data <- df[-trainIndex,]
```

로지스틱 회귀(Logistic Regression)

로지스틱 회귀분석

```
result=glm(train_data$Personal.Loan ~., data= train_data, family = “binomial”)

tidy_result <- tidy(result)
tidy_result <- mutate_if(tidy_result, is.numeric, round, 3)
tidy_result$odds <- exp(tidy_result$estimate)

print(tidy_result)
```

로지스틱 회귀(Logistic Regression)

로지스틱 회귀분석

```
library(car)
잔차로 검증할 수 없음

#다중공선성(변수의 중복성) : 10이상 GVIF^(1/(2*DF))
vif(result)

#다중공선성 위배 변수제거 후 다시 분석
result=glm(train_data$Personal.Loan ~., data= train_data, family = “binomial”)

tidy_result <- tidy(result)
tidy_result <- mutate_if(tidy_result, is.numeric, round, 3)
tidy_result$odds <- exp(tidy_result$estimate)

print(tidy_result)
```

로지스틱 회귀(Logistic Regression)

로지스틱 회귀분석

```
#Test 데이터셋 결과
test_predictions <- predict(result, newdata = test_data, type = "response")
predicted_class <- ifelse(test_predictions > 0.5, "yes", "no")
print(predicted_class)

#모델 저장
save(result, file = "logistic_model.RData")

#모델 로드
load("logistic_model.RData")
```

로지스틱 회귀(Logistic Regression)

모델검정

- 수치형 : RMSE, R-Squared(R^2)
- 범주형 : Accuracy, ROC Curve
  - Precision(정밀도) : 모델에 의해 수행된 모든 긍정적인 예측 중 실제 긍정적인 예측의 비율
    - 장난감 상자에 10번 장난감을 꺼낼 경우, 10번 중에서 장난감 7개와 기타 물건 3개(책, 양말, 연필 등)를 선택함 → Precision는 잡은 물건 중 실제로 장난감이 몇 개나 되는지에 대한 것 따라서 이 경우 당신이 잡은 물건 중 7개가 장난감 이었기 때문에 Precision값은 7/10
  - Recall(재현율) : 모든 실제 양성 사례 중 참양성 예측의 비율(남아있는 장난감)
    - 상자에 10개의 장난감이 있다고 상상할 때(더 있을수도 있음), 10개의 장난감 중에서 7개를 찾음 → 리콜은 당신이 발견한 상자 속 장난감의 총 개수에 관한 것 → 이 경우 장난감 10개 중 7개를 찾았기 때문에 기억력은 10점 만점에 7점
  - Accuarcy(정확도) : 전체 예측 정답 중 얼마나 올바르게 예측했는가에 대한 비율
  - F1 Score : Precision와 Recall의 조화 평균 → Label의 성능이 불균형일 경우 정확하게 평가할 수 있음

로지스틱 회귀(Logistic Regression)

모델검정

- **Precision(정밀도) :** $\frac{TP}{TP+FP}$
- **Recall(재현율) :** $\frac{TP}{TP+FN}$
- **Accuracy(정확도) :** $\frac{TP+TN}{TP+FN+FP+TN}$
- **F1-Score :** $2 \times \frac{Precision \times Recall}{Precision+Recall}$

		실제 정답	
		True(환자)	False(정상)
분류결과	True(환자)	TP(20)	FP(40)
	False(정상)	FN(30)	TN(10)

Precision

Recall

20대의 자동차 중 10대는 택시 10대는 택시가 아닐 때 → 10대가 지나갈 경우
5대의 차량에 손을 흔드니 모두 택시 → 정밀도 100%
10대가 지나갔지만 5대에만 손을 흔듦 하지만 10대가 택시 → 재현율 50%

로지스틱 회귀(Logistic Regression)

모델검정

- 높은 Precision, 낮은 Recall → 5개 물건만 집어 들었고, 5개 모두 장난감이었다고 할 경우 정확하다는 것을 의미함(장난감이 아닌 것은 집어 들지 않았기 때문) → 하지만 상자 안에 장난감을 남겨두었을 수도 있음 → Precision는 높지만 Recall은 낮음

		실제 정답	
		True(장난감)	False(물건)
분류결과	True(장난감)	TP(5)	FP(0)
	False(물건)	FN(4)	TN(2)

- 높은 Recall, 낮은 Precision → 이제 6개의 물건을 집어 들었고, 4개의 장난감을 찾았지만 실수로 책 2권도 집을 경우 → 대부분의 장난감을 찾았지만(높은 Recall) 장난감이 아닌 물건도 집어 들었으므로 정확성이 완벽하지 않음

		실제 정답	
		True(장난감)	False(물건)
분류결과	True(장난감)	TP(4)	FP(2)
	False(물건)	FN(0)	TN(2)

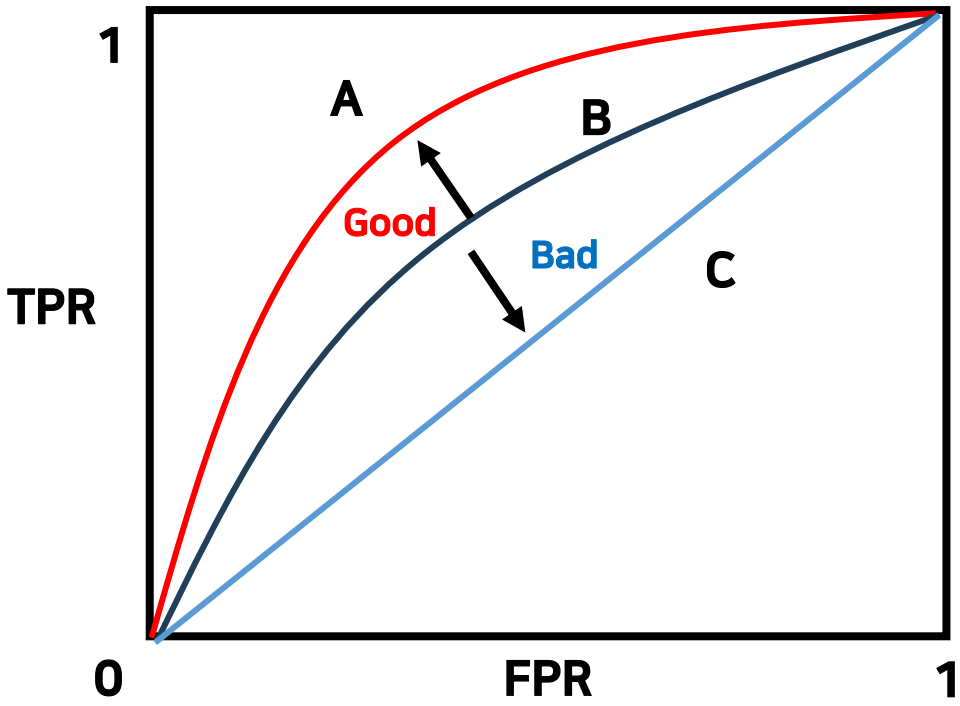
- 높은 Precision 및 높은 Recall:9개 물건을 집었고 9개 모두 장난감이었고 상자에 장난감이 1개만 남아 있었다면 높은 Precision(장난감만 집은 것)와 높은 Recall(거의 모든 장난감을 찾았음)을 모두 갖게 됨

로지스틱 회귀(Logistic Regression)

모델검정

- ROC(Receiver Operating Characteristic)
- TPR(True Positive Rate) : 실제 클래스 True중에 잘 맞춘 것 → 4/6
- FPR(False Positive Rate) : 실제 클래스 False중에 못 맞춘 것 → 2/4

		실제 정답	
		True(장난감)	False(물건)
분류결과	True(장난감)	TP(4)	FP(2)
	False(물건)	FN(2)	TN(2)



A : 좋은 모델

B : 보통 모델

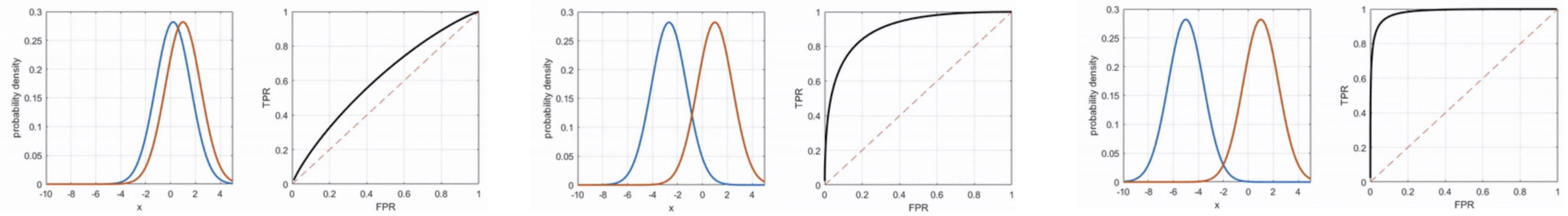
C : 나쁜모델

로지스틱 회귀(Logistic Regression)

모델검정

- ROC(Receiver Operating Characteristic)
- TPR(True Positive Rate) : 실제 클래스 True중에 잘 맞춘 것 → 4/6
- FPR(False Positive Rate) : 실제 클래스 False중에 못 맞춘 것 → 2/4

		실제 정답	
		True(장난감)	False(물건)
분류결과	True(장난감)	TP(4)	FP(2)
	False(물건)	FN(2)	TN(2)



로지스틱 회귀(Logistic Regression)

로지스틱 회귀분석(ROC Curve)

```
install.packages("pROC")
library(pROC)

# 모델의 Confusion matrix
confusion_matrix <- confusionMatrix(test_predictions, test_data$Personal.Loan, type =
'response')

print(confusion_matrix)

# ROC curve and AUC
roc_curve <- roc(test_data$Personal.Loan, test_predictions)
plot(roc_curve)
auc(roc_curve)
```

로지스틱 회귀(Logistic Regression)

로지스틱 회귀분석(ROC Curve)

```
#새로운 데이터를 활용한 모델 적용
df_test <- read.csv("UniversalBank_test.csv", header=TRUE)
```