# 7. 머신 러닝 (Machine Learning) 개요

집현전 초급반 7조

김유진, 장호섭, 최희옥

머신 러닝은 기존에 해결할 수 없었던 수많은 문제들에 대한
최적의 해결방법을 제시해주고 있음

# 목차


01 머신 러닝이란(What is Machine Learning?)

02 머신 러닝 훑어보기

03 선형 회귀(Linear Regression)

04 자동 미분과 선형 회귀 실습

머신 러닝이란(What is Machine Learning?)

# 머신 러닝(Machine Learning)이 아닌 접근 방법의 한계

```
def prediction(이미지 as input):
    어떻게 코딩해야하지?
    return 결과
```


Ex) 주어진 사진으로부터 고양이 사진인지 강아지 사진인지 판별하는 일.

사진으로부터 공통된 명확한 특징을 잡아내는 것이 쉽지 않음

→ 머신러닝이 해결책이 될 수 있음

머신 러닝이란(What is Machine Learning?)

# 머신 러닝은 기존 프로그래밍의 한계에 대한 해결책이 될 수 있다

### 1. 기존의 프로그래밍 접근 방법


### 2. 머신 러닝의 접근 방법


### 머신러닝

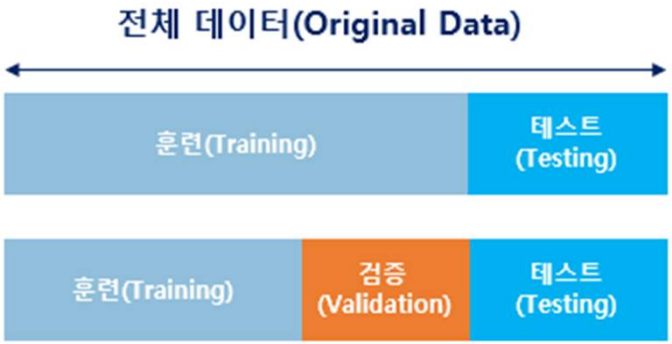
주어진 데이터로부터 규칙성을 찾는 것에 초점이 맞추어져 있음.

### 학습(training)

주어진 데이터로부터 규칙성을 찾는 과정

머신 러닝 훑어보기

# 머신 러닝 모델의 평가




| 훈련용 데이터 | 검증용 데이터 | 테스트 데이터 |
|---|---|---|
| 훈련 | 정확도 검증, 하이퍼파라미터 튜닝(tuning) | 모델의 성능 평가 |
| 문제지(공부 자료) | 모의고사 | 수능시험 |

## 검증용 데이터

모델의 성능을 조정하기 위한 용도과

과적합이 되고 있는지 판단하거나 하이퍼파라미터의 조정을 위한 용도

## 하이퍼파라미터(초매개변수)

값에 따라서 모델의 성능에 영향을 주는 매개변수들을 말함

사용자가 직접 정해줄 수 있는 변수

Ex) 선형 회귀 - 경사 하강법 학습률(learning rate)

딥 러닝 - 은닉층의 수, 뉴런의 수, 드롭아웃 비율 등

## 매개변수

가중치와 편향과 같은 학습을 통해 바뀌어져가는 변수

모델이 학습하는 과정에서 얻어지는 값

기계가 훈련을 통해서 바꾸는 변수

Classification & Regression

# 분류와 회귀

## 1. 이진 분류 문제(Binary Classification)

주어진 입력에 대해서 둘 중 하나의 답을 정하는 문제

Ex) 시험 성적 – 합격, 불합격 판단 / 메일 – 정상 메일, 스팸 메일 판단

## 2. 다중 클래스 분류(Multi-class Classification)

주어진 입력에 대해 두 개 이상의 정해진 선택지 중에서

답을 정하는 문제

선택지 : 카테고리 또는 범주 또는 클래스라고 함

## 3. 회귀 문제(Regression)

분류 문제처럼 0 또는 1이나 분리된(비연속적인) 답이 결과가 아닌

연속된 값을 결과로 가짐

Ex) 시계열 데이터를 이용한 주가 예측, 생산량 예측, 지수 예측 등

Supervised Learning & Unsupervised Learning

# 지도 학습과 비지도 학습

## 1. 지도학습

레이블(Lable)이라는 정답과 함께 학습

기계는 예측값과 실제값의 차이인 오차를 줄이는 방식으로 학습

레이블 : $y$, 실제값 등으로 부르기도 함

예측값 : $\hat{y}$ 과 같이 표현하기도 함

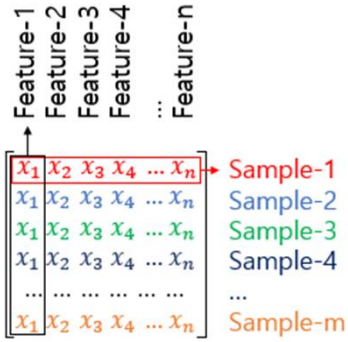
## 2. 비지도 학습

레이블 없이 학습

Ex) 토픽 모델링의 LDA, 워드투벡터(Word2Vec) - 비지도 학습

Sample & Feature

# 샘플과 특성




독립 변수 x의 행렬을 X라고 했을 때,
독립 변수의 개수가 n개이고
데이터의 개수가 m인 행렬 X

## 샘플(Sample)

머신 러닝에서는 하나의 데이터, 하나의 행을 샘플(Sample)이라고 부름

(데이터베이스에서는 레코드라고 부르는 단위)

## 특성(Feature)

종속 변수 y를 예측하기 위한 각각의 독립변수 x를 특성(Feature)이라고 부름

Confusion Matrix

# 혼동 행렬

## 혼동 행렬(Confusion Matrix)의 정의

- 머신 러닝에서는 맞춘 문제수를 전체 문제수로 나눈 값을 정확도(Accuracy)라고 함
- 정확도는 맞춘 결과와 틀린 결과에 대한 세부적인 내용을 알려주지는 않음. 이를 위해서 사용하는 것이 혼동 행렬(Confusion Matrix)

| - | 참 | 거짓 |
|---|---|---|
| 참 | TP | FN |
| 거짓 | FP | TN |

Ex

양성(Positive)과 음성(Negative)을 구분하는 이진 분류가 있다고 했을 때 혼동 행렬 (각 열은 예측값, 각 행은 실제값을 나타냄)

- 각각 TP(True Positive), TN(True Negative), FP(False Postivie), FN(False Negative)라고 하는데 True는 정답을 맞춘 경우고 False는 정답을 맞추지 못한 경우

- Positive와 Negative는 각각 제시했던 정답.

  TP : 양성(Postive)이라고 대답, 실제로 양성이라서 정답을 맞춘 경우

  TN : 음성(Negative)이라고 대답, 실제로 음성이라서 정답을 맞춘 경우

  FP : 양성이라고 대답, 음성이라서 정답을 틀린 경우

  FN : 음성이라고 대답, 양성이라서 정답을 틀린 경우

  그리고 이 개념을 사용하면 또 새로운 개념인 정밀도(Precision)과 재현율(Recall)이 됨

Confusion Matrix

# 혼동 행렬

## 1. 정밀도(Precision)

정밀도는 양성이라고 대답한 전체 케이스에 대한 TP의 비율

$$정밀도 = \frac{TP}{TP + FP}$$

## 2. 재현율(Recall)

재현율은 실제값이 양성인 데이터의 전체 개수에 대해서 TP의 비율

양성인 데이터 중에서 얼마나 양성인지를 예측(재현)했는지를 나타냄

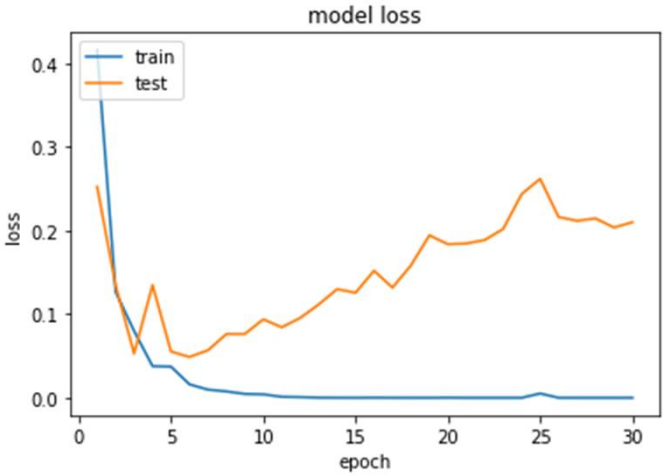
$$재현율 = \frac{TP}{TP + FN}$$

Overfitting & Underfitting

# 과적합과 과소 적합

## 과적합(Overfitting)의 정의

- 훈련 데이터를 과하게 학습한 경우
- 훈련 데이터에 대해서만 과하게 학습하면 테스트 데이터나 실제 서비스에서의 데이터에 대해서는 정확도가 좋지 않은 현상이 발생
- 과적합 상황에서는 훈련 데이터에 대해서는 오차가 낮지만, 테스트 데이터에 대해서는 오차가 높아지는 상황이 발생




## 과소 적합(Underfitting)의 정의

- 테스트 데이터의 성능이 올라갈 여지가 있음에도 훈련을 덜 한 상태
- 훈련 자체가 부족한 상태이므로 과대 적합과는 달리 훈련 데이터에 대해서도 보통 정확도가 낮음

## 과적합과 과소적합이라고 부르는 이유

- 머신 러닝에서 학습 또는 훈련이라고 하는 과정을 적합(fitting)이라고도 부를 수 있기 때문
- 모델이 주어진 데이터에 대해서 적합해져가는 과정이기 때문 (이러한 이유로 케라스에서는 기계를 학습시키는 도구의 이름을 fit()이라고 명명함)

Linear Regression

# 회귀분석

| 1차 구분요인 | 2차 구분요인 | 구분 |
|---|---|---|
| 독립변수의 수 | 1개 | 단순 회귀분석 |
| | 2개 이상 | 다중 회귀분석 |
| 독립변수의 척도 | 명목척도, 서열척도 | 더미변수 회귀분석 |
| | 등간척도, 비율척도 | 일반 회귀분석 |
| 독립변수와 종속변수 관계 | 선형 | 선형 회귀분석 |
| | 비선형 | 비선형 회귀분석 |

| 척도 | 순서 | 균등한 간격 | 영(0)점 존재 |
|---|---|---|---|
| 명목척도 (ex.사는 곳) | X | X | X |
| 서열척도 (ex.사회계층) | O | X | X |
| 등간척도 (ex.온도) | O | O | X |
| 비율척도 (ex.자녀수) | O | O | O |

## 정의

독립변수(원인)과 종속변수(결과)간의 상호 연관성 정도를

파악하는 분석 방법

## 목적

두 변수(종속변수, 독립변수) 간의 인과관계를 분석할 때 주로 사용

## 예시

아파트 매매가(종속변수)는 평수, 교통수단, 층수 등 다양한

요인(독립변수)들로 인해 가격이 결정된다.

Simply/Multiple Linear Regression Analysis

# 선형 회귀분석의 종류

## 1. 단순 선형 회귀분석 (Simple Linear Regression Analysis)

단순 선형 회귀분석은 종속 변수가 한가지의 독립 변수에 영향을 받는 경우

$$y = 'Wx + b$$

- W는 가중치(weight), b는 편향(bias)
- 직선의 방정식에서는, 가중치는 선의 기울기, b는 절편을 의미
- 가중치와 절편은 데이터를 학습한 후 결정

**예시**

집의 가격(종속 변수)은 집의 평수(독립 변수)에 의해 결정

집의 가격(y)=W*집의 평수(x)+b

## 2. 다중 선형 회귀분석 (Multiple Linear Regression Analysis)

다중 선형 회귀분석은 종속 변수가 여러 개의 독립 변수에 영향을 받는 경우

$$y = W_1x_1 + W_2x_2 + \ldots W_nx_n + b$$

- 단순 회귀분석에 비해, 연산이 복잡해지지만 더욱 정교한 회귀분석 결과를 산출할 수 있는 장점 존재

**예시**

집의 가격(종속 변수)은 평수뿐만 아니라 교통 수단, 학군, 지역, 층수 등 다양한 독립변수로 선형회귀 모델을 만들면, 이는 다중 선형 회귀분석이라고 합니다.

Build hypothesis

# 가설 세우기

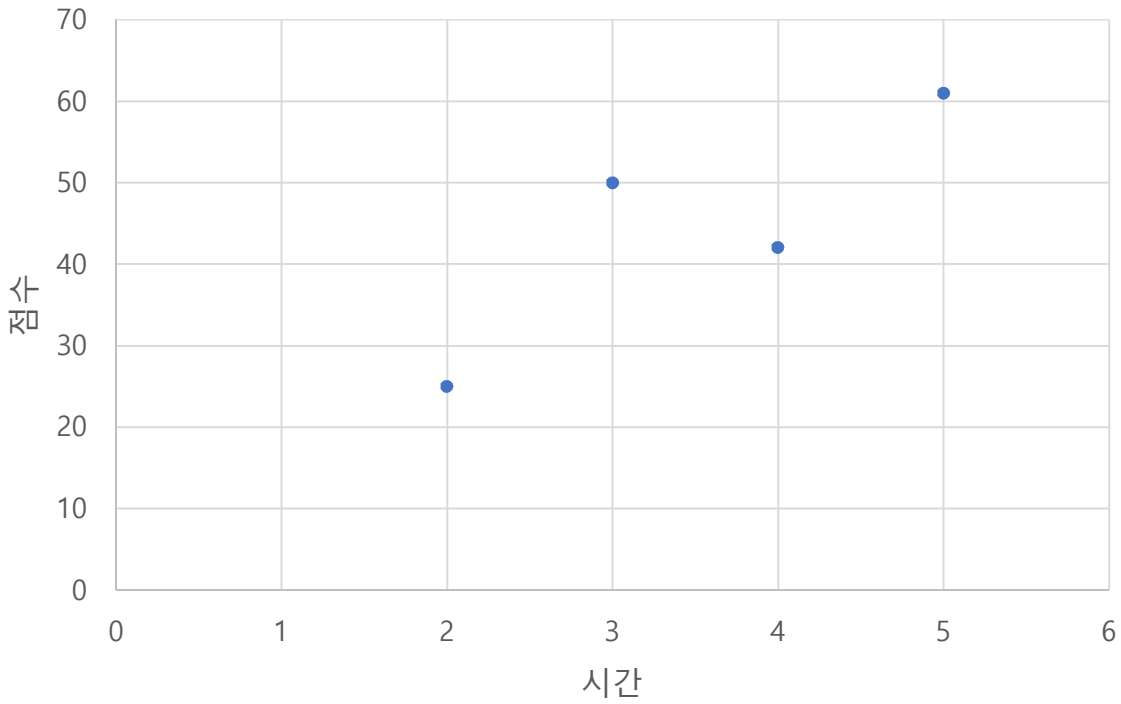
| 시간 (x) | 점수 (y) |
|---|---|
| 2 | 25 |
| 3 | 50 |
| 4 | 42 |
| 5 | 61 |
| 8 | ? |

## 데이터 설명

- 학생들이 공부 시간에 따라서 얻은 점수의 분포

## 가설 (Hypothesis)

- x와 y의 관계를 유추하기 위해서, 수학적으로 식을 세워보는 과정
- 데이터를 기반으로 x와 y의 관계를 유추하고, 학생이 8시간 공부했을 때의 공부했을 때의 성적을 예측



Build hypothesis

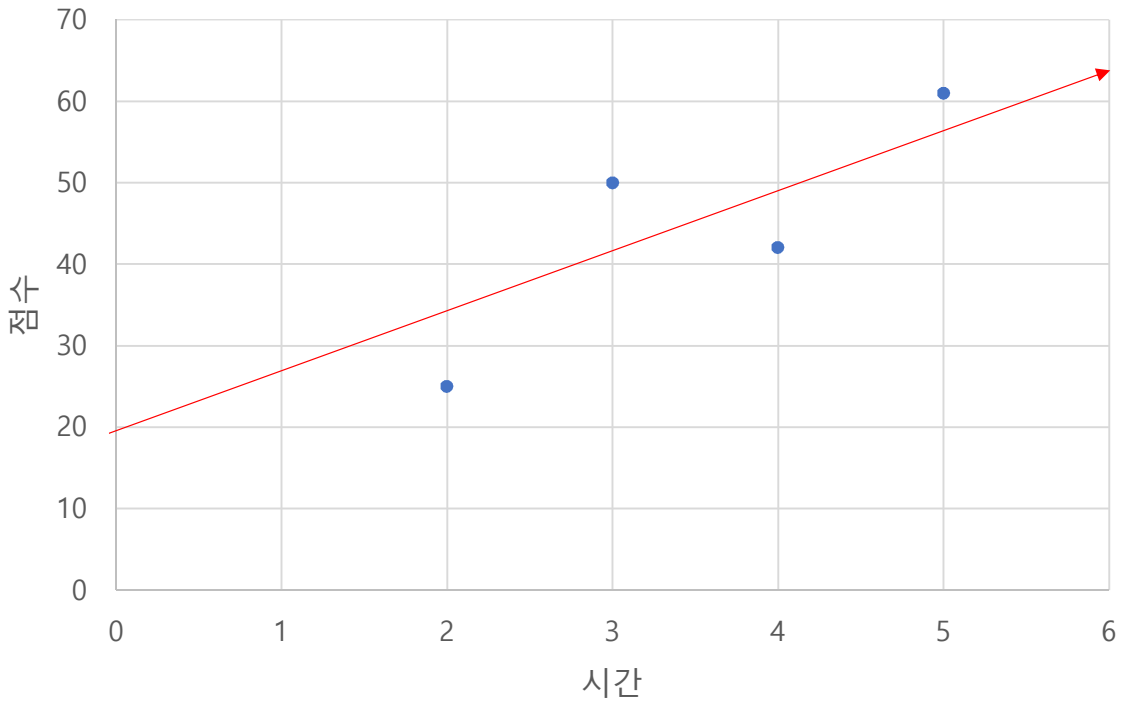
# 가설 세우기

| 시간 (x) | 점수 (y) |
|---|---|
| 2 | 25 |
| 3 | 50 |
| 4 | 42 |
| 5 | 61 |
| 6 | ? |

## 가설 세우기

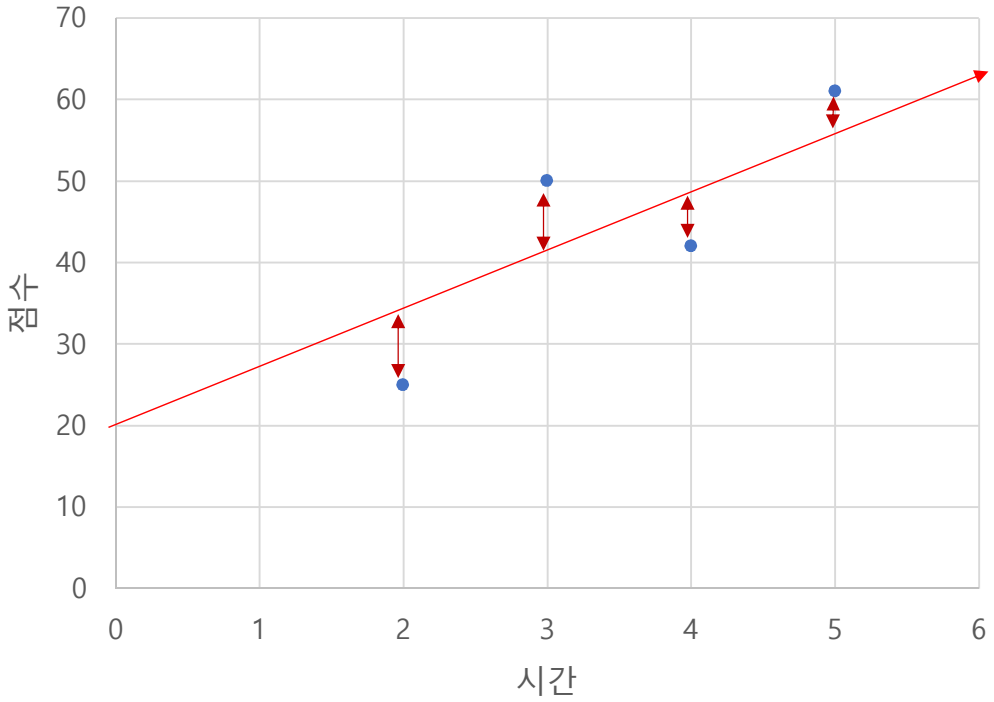
- **H(x)= Wx + b 공식을 기반으로 데이터를 수학공식에 대입해서 W와 b를 산출**

  **(새로운 데이터가 추가될 시, W와 b 값은 지속적으로 업데이트)**



Cost function: MSE (Mean Squared Error)

# 비용 함수 : 평균 제곱 오차




## 비용 함수

- 실제값과 가설로부터 얻은 예측값의 오차를 계산하는 식
- 비용 함수는 단순히 실제값과 예측값에 대한 오차를 표현하면 되는 것이 아니라, 예측값의 오차를 줄이는 일에 최적화 된 식이어야 함
- 각 문제들에는 적합한 비용 함수들이 있으며, 회귀 문제의 경우에는 주로 평균 제곱 오차(Mean Squared Error, MSE)가 사용되고 있음

Cost function: MSE (Mean Squared Error)

# 비용 함수 : 평균 제곱 오차

## 오차(Error) 정의

- 오차는 주어진 데이터에서 각 에서의 실제값 와 위의 직선에서 예측하고 있는 H(x)값의 차이
- 오차를 줄이면서, W와 b의 값을 구하기 위해서는 전체 오차의 크기를 구해야됨 (예측값은 y=13x+1으로 가정)

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

- 수식적으로 단순히 '오차 = 실제값 - 예측값' 이라고 정의한 후에 모든 오차를 더하면 음수 오차도 있고, 양수 오차도 있으므로 오차의 절대적인 크기를 구할 수가 없음
- 이를 해결하기 위해, 모든 오차를 제곱하여 더하는 방법을 사용

$$\sum_{i=1}^{n} \left[ y^{(i)} - H(x^{(i)}) \right]^2 = (-2)^2 + 10^2 + (-7)^2 + (-5)^2 = 178$$

- 이때 데이터의 개수로 나누면, 오차의 제곱합에 대한 평균을 구할 수 있는데 이를 평균 제곱 오차(Mean Squered Error, MSE)라고 함

$$\frac{1}{n} \sum_{i=1}^{n} \left[ y^{(i)} - H(x^{(i)}) \right]^2 = 178/4 = 44.5$$

Cost function: MSE (Mean Squared Error)

# 비용 함수 : 평균 제곱 오차

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

- 평균 제곱 오차의 값을 최소값으로 만드는 W와 b를 찾아내는 것이 정답인 직선을 찾아내는 일
- 평균 제곱 오차를 와 에 의한 비용 함수(Cost function)로 재정의 (하단 수식)

$$cost(W,b) = \frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)} - H(x^{(i)})\right]^2$$

- 모든 점들과의 오차가 클 수록 평균 제곱 오차는 커지며, 오차가 작아질 수록 평균 제곱 오차는 작아지므로 Cost(W,b)를 최소로 만드는 W와 b를 구하면 결과적으로 y와 x의 관계를 가장 잘 나타내는 직선을 그릴 수 있음

$$W, b \rightarrow minimize\ cost(W,b)$$

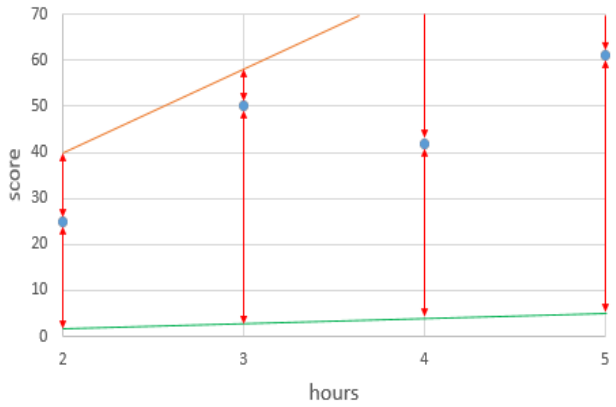
Optimizer : Gradient Descent

# 옵티마이저 : 경사하강법

## 옵티마이저 (최적화 알고리즘) 정의

- 옵티마이저(Optimizer)는 선형 회귀를 포함한 수많은 머신 러닝, 딥 러닝의 학습은 결국 비용 함수를 최소화하는 매개 변수인 W와 b를 찾기 위한 작업을 수행하는데 사용
- 머신 러닝에서 학습(training)은 옵티마이저를 통해 적절한 W와 b를 찾아내는 과정

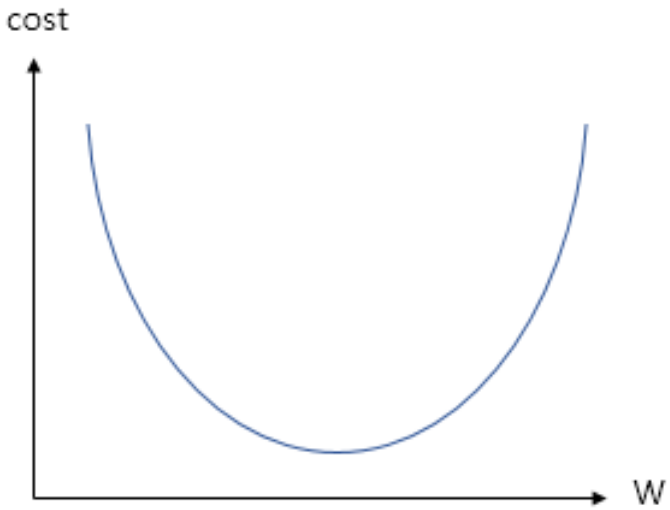
## 경사 하강법(Gradient Descent)의 정의




- 가장 기본적인 옵티마이저 알고리즘
- W는 머신 러닝 용어로는 가중치라고 불리지만, 직선의 방정식 관점에서 보면 직선의 기울기를 의미
- 좌측 그래프는 기울기 W가 지나치게 높거나, 낮을 때 어떻게 오차가 커지는지 보여주고 있음

Optimizer : Gradient Descent

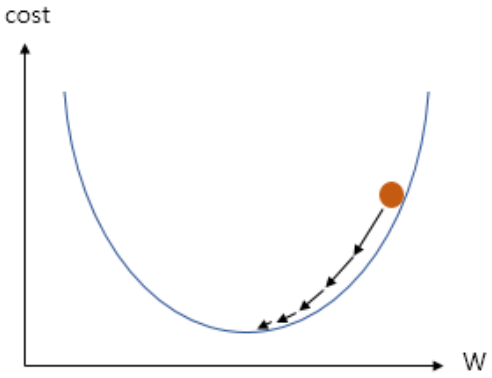
# 옵티마이저 : 경사하강법

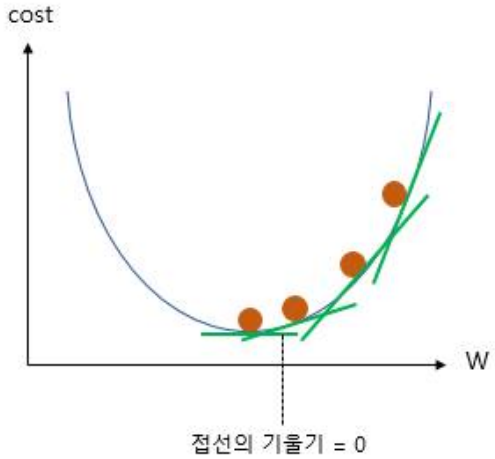



- 설명의 편의를 위해 편향 b 없이 단순히 가중치 W만을 사용한 y=Wx라는 가설 H(x)를 가지고, 경사 하강법을 수행한다고 가정
- 비용 함수의 cost(W)값 는 cost라고 줄여서 표현했을 시, W와 cost의 관계를 그래프로 표현하면 좌측 그래프와 같음

Optimizer : Gradient Descent

# 옵티마이저 : 경사하강법




- **기울기 W가 무한대로 커지면 커질 수록 cost의 값 또한 무한대로 커지고, 반대로 기울기 W가 무한대로 작아져도 cost의 값은 무한대로 커짐**
- **위의 그래프에서 cost가 가장 작을 때는 볼록한 부분의 맨 아래 부분**
- **해야할 일은 cost가 가장 최소값을 가지게 하는 W를 찾는 일**이므로, 볼록한 부분의 맨 아래 부분의 의 값을 찾아야 함

- **초기에는 랜덤 값(임의의 수)을 정한 뒤에, 맨 아래의 볼록한 부분을 향해 점차 W의 값을 수정**
- **각 기울기의 미분을 취해서, 접선의 기울기를 계산**
- **점차 기울기가 낮아지면서, 접선의 기울기가 0이 되는 시점이 cost가 최소가 되는 시점**

Linear regression with Keras

# 케라스로 구현하는 선형 회귀

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras import optimizers
```

```
X=[1,2,3,4,5,6,7,8,9] # 공부하는 시간
y=[11,22,33,44,53,66,77,87,95] # 각 공부하는 시간에 맵핑되는 성적

model = Sequential()

# 입력 x의 차원은 1, 출력 y의 차원도 1. 선형 회귀이므로 activation은 'linear'
model.add(Dense(1, input_dim=1, activation='linear'))

# sgd는 경사 하강법을 의미. 학습률(learning rate, lr)은 0.01.
sgd = optimizers.SGD(lr=0.01)

# 손실 함수(Loss function)은 평균제곱오차 mse를 사용합니다.
model.compile(optimizer=sgd ,loss='mse',metrics=['mse'])

# 주어진 X와 y데이터에 대해서 오차를 최소화하는 작업을 300번 시도합니다.
model.fit(X,y, batch_size=1, epochs=300, shuffle=False)
```

1. 필요 패키지 호출 (keras, numpy)
2. 공부한 시간 x, 성적 y인 가상 데이터 구성
3. Sequential 모델 생성
4. 선형 회귀 (linear) 모델, x와 y의 차원을 1로 세팅한 차원 추가
5. sgd(경사 하강법) 옵티마이저 사용, 학습률= 0.01로 세팅
6. 비용 함수는 mse(평균 제곱 오차) 방법을 사용
7. 데이터의 훈련 횟수(epoch)은 300번으로 세팅한 후 실행

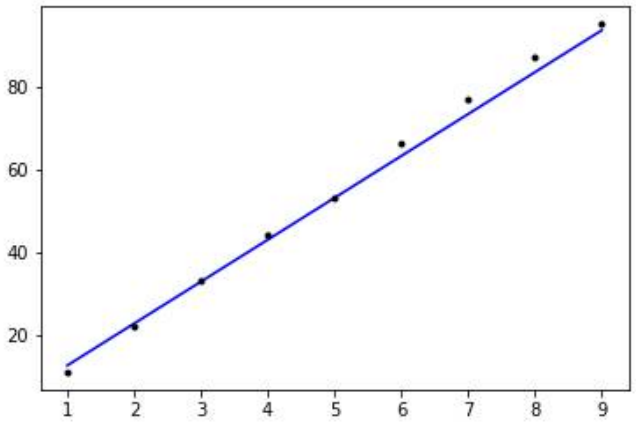
Linear regression with Keras

# 케라스로 구현하는 선형 회귀

```
Epoch 1/300
9/9 [==============================] - 0s 37ms/step - loss: 294.9226 - mean_squared_error: 294.9226
... 중략 ...
Epoch 167/300
9/9 [==============================] - 0s 1ms/step - loss: 2.1460 - mean_squared_error: 2.1460
... 중략 ...
Epoch 300/300
9/9 [==============================] - 0s 1ms/step - loss: 2.1460 - mean_squared_error: 2.1460
```

훈련 횟수를 거듭할수록, 오차 (mean_squared_error)이 감소하다가, 167번째 epoch(훈련) 이후 더 이상 떨어지지 않음

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.plot(X, model.predict(X), 'b', X,y, 'k.')
```




오차를 최소화하는 모델 생성 후, 그래프로 표현

```
print(model.predict([9.5]))
```

```
[[98.556465]]
```

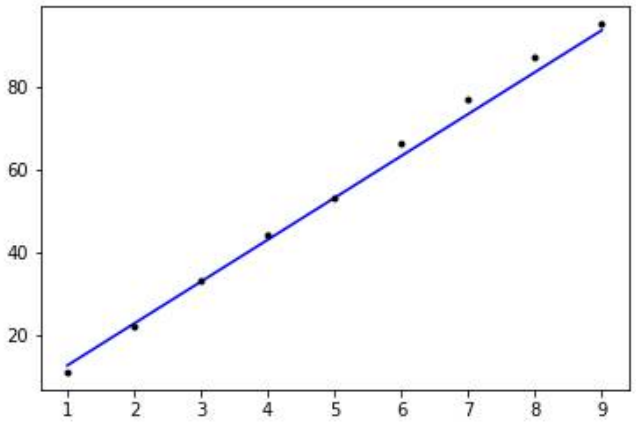
학습 된 모델로 9.5시간 학습했을 시, 나오는 성적을 예측하면 98.5점을 얻는다고 예측

Linear regression with Keras

# 자동 미분과 선형 회귀

```
Epoch 1/300
9/9 [==============================] - 0s 37ms/step - loss: 294.9226 - mean_squared_error: 294.9226
... 중략 ...
Epoch 167/300
9/9 [==============================] - 0s 1ms/step - loss: 2.1460 - mean_squared_error: 2.1460
... 중략 ...
Epoch 300/300
9/9 [==============================] - 0s 1ms/step - loss: 2.1460 - mean_squared_error: 2.1460
```

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.plot(X, model.predict(X), 'b', X,y, 'k.')
```




오차를 최소화하는 모델 생성 후,

그래프로 표현

미분을 통해 작은 2계층(2-layer) 신경망에서는 역전파 단계를 직접 구현하는 것이 큰일이 아니지만, 복잡한 대규모 신경망에서는 매우 복잡한 일일 것임.

다행히도, torch나 keras같은 딥러닝 툴을 쓰면 툴에 내장되어 있는 자동 미분을 사용하여 신경망의 역전파 단계 연산을 자동으로 계산.

학습 된 모델로 9.5시간 학습했을 시, 나오는 성적을 예측하면 98.5점을 얻는다고 예측

```
print(model.predict([9.5]))
```

```
[[98.556465]]
```


감사합니다