

2018.0.0

기계학습

정유진

CONTENTS

1. 머신러닝 기본개념
2. Hypothesis/Cost Function/Gradient Descent
3. 모델 예시(지도 vs 비지도)
4. 모델 평가 방법
5. 코드

1. 머신러닝 기본개념

· 머신러닝(machine learning)이란???

개념 : 인공 지능을 구현하는 구체적 접근 방식

특징 : 1. AI로부터 파생
2. 기존 컴퓨터와 다른 새로운 능력을 포함

1. 머신러닝 기본개념

<전통적 컴퓨터>

VS

<머신러닝>

규칙기반(rule based)의 접근법

- 사람이 직접 컴퓨터에게 지능적으로 행동하는 방법을 알려주고, 빠르게 그 행동을 처리하도록 함.

사람이 직접 생각하는 방법을 알려주는 것이 아니라 기계가 스스로 배우도록 하는 것.

- 컴퓨터로 알고리즘을 학습하여 새로운 데이터가 들어왔을 때 데이터 결과 예측.
- 미리 프로그램되지 않은 부분에서도 예측과 결정을 내릴 수 있는 방식

1. 머신러닝 기본개념

- 활용 예

1. Database mining

ex) web click data, medical records

2. Applications can't program by hand

ex) 자동헬리콥터, 손글씨 자동인식, NLP

3. Self customizing system

ex) 아마존, 넷플릭스의 추천시스템

4. Human learning 이해

ex) brain, real AI

- 참조

딥러닝 : 머신러닝의 일종으로 인공신경망에서 발전한 형태의 인공지능. 뇌의 뉴런과 유사한 정보 입출력 계층을 활용해 데이터를 학습

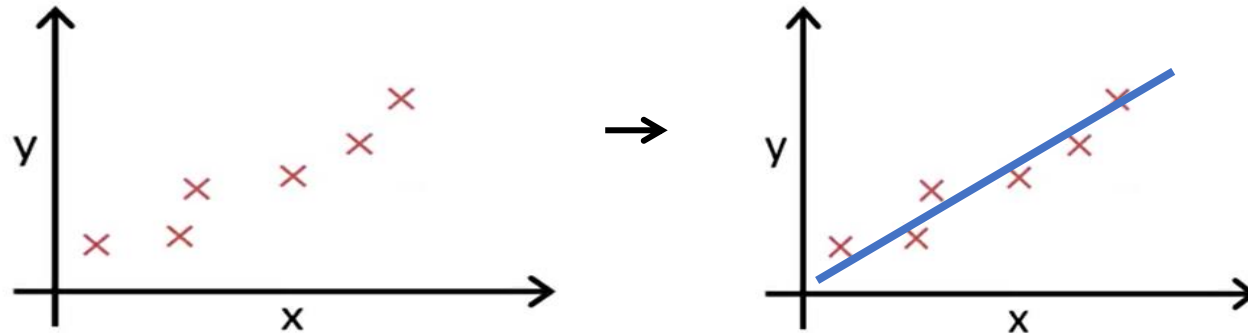
2. Hypothesis/Cost Function/Gradient Descent

- 가설(hypothesis)

Input(feature)과 output(target)의 관계를 나타내는 함수

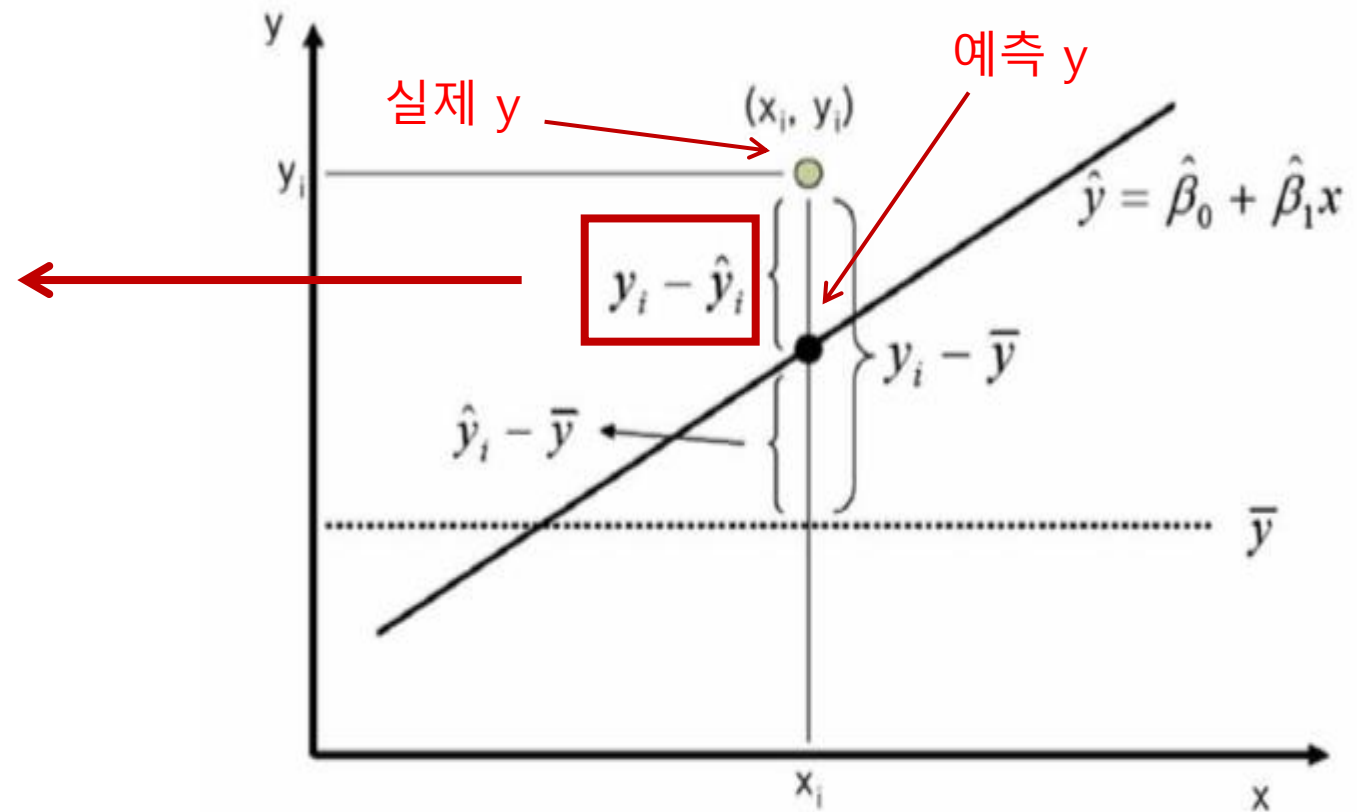
$$\text{예 : } h_{\theta}(x) = \theta_0 + \theta_1 x$$

 parameters



2. Hypothesis/Cost Function/Gradient Descent

실제 y 와 예측 y 와의 차이
($y - \hat{y}$)들을 줄여야 한다.



2. Hypothesis/Cost Function/Gradient Descent

• Cost Function

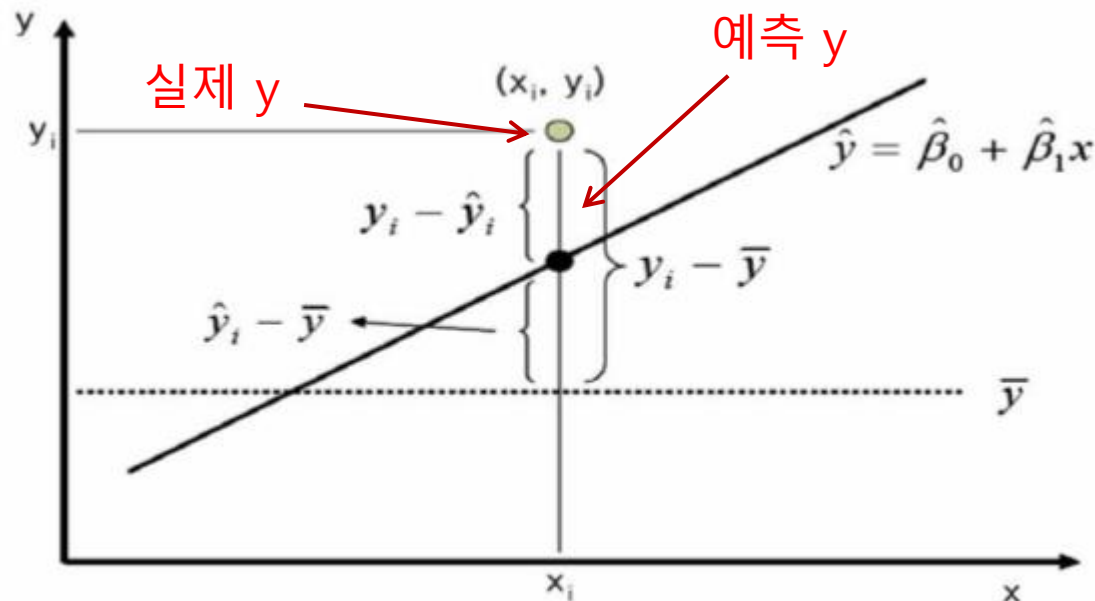
비용에 관련된 모든 변량에 대하여 어떤 관계를 나타내는 함수.
주어진 데이터에 가장 잘 '맞는' 직선을 선택하기 위한 일정 기준.
가설의 식의 정확성을 측정하기 위한 식.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m ((\underbrace{h_{\theta}(x^{(i)})}_{\text{가설}}) - \underbrace{y^{(i)}}_{\text{실제}}))^2 = \frac{1}{2m} \sum_{i=1}^m ((\hat{y}^{(i)}) - y^{(i)})^2$$

2. Hypothesis/Cost Function/Gradient Descent

- Cost Function

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m ((h_{\theta}(x^{(i)}) - y^{(i)}))^2 = \frac{1}{2m} \sum_{i=1}^m ((\hat{y}^{(i)}) - y^{(i)})^2$$



$$y_i - \bar{y} = (y_i - \hat{y}_i) + (\hat{y}_i - \bar{y})$$

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

총제곱합

SST

$n - 1$

잔차제곱합

SSE

$n - 2$

회귀제곱합

SSR

1 (자유도)


*SST (Total sum of squares)

*SSR (Regression sum of squares)

2. Hypothesis/Cost Function/Gradient Descent

- **Gradient Descent**

Hypothesis function의 최적의 parameter을 찾는 방법

$$h_{\theta}(x) = \boxed{\theta_0} + \boxed{\theta_1}x$$


parameters

목표 : Cost Function에서 $J(\theta_0, \theta_1)$ 을 최소화 $\rightarrow \min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

전략 : 1. 어떤 parameter에서든 시작 가능
2. 계속 이 parameter을 변화해가면서 최소의 J를 찾는 것

2. Hypothesis/Cost Function/Gradient Descent

- Gradient Descent

algorithm : $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$

Learning Rate
(학습율)

<derivative>

<Update Rules>

$$\theta_0 := \theta_0 - \alpha \frac{d}{d\theta_0} J(\theta_0, \theta_1)$$

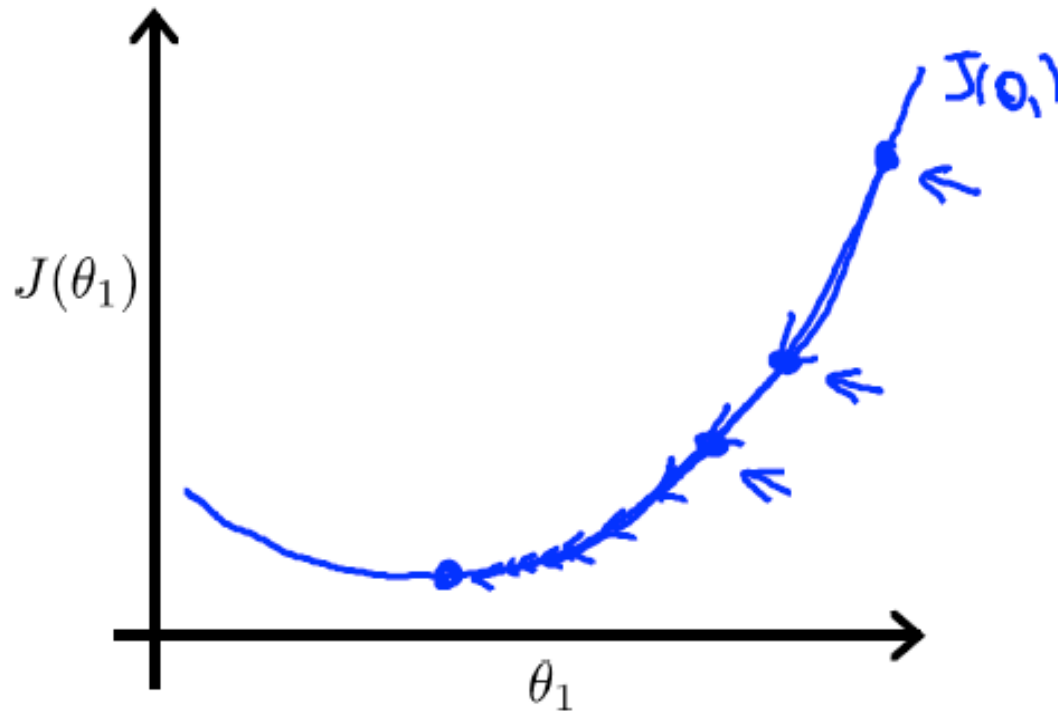
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_0, \theta_1)$$

$$\frac{d}{d\theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})$$

$$\frac{d}{d\theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

2. Hypothesis/Cost Function/Gradient Descent

• Gradient Descent

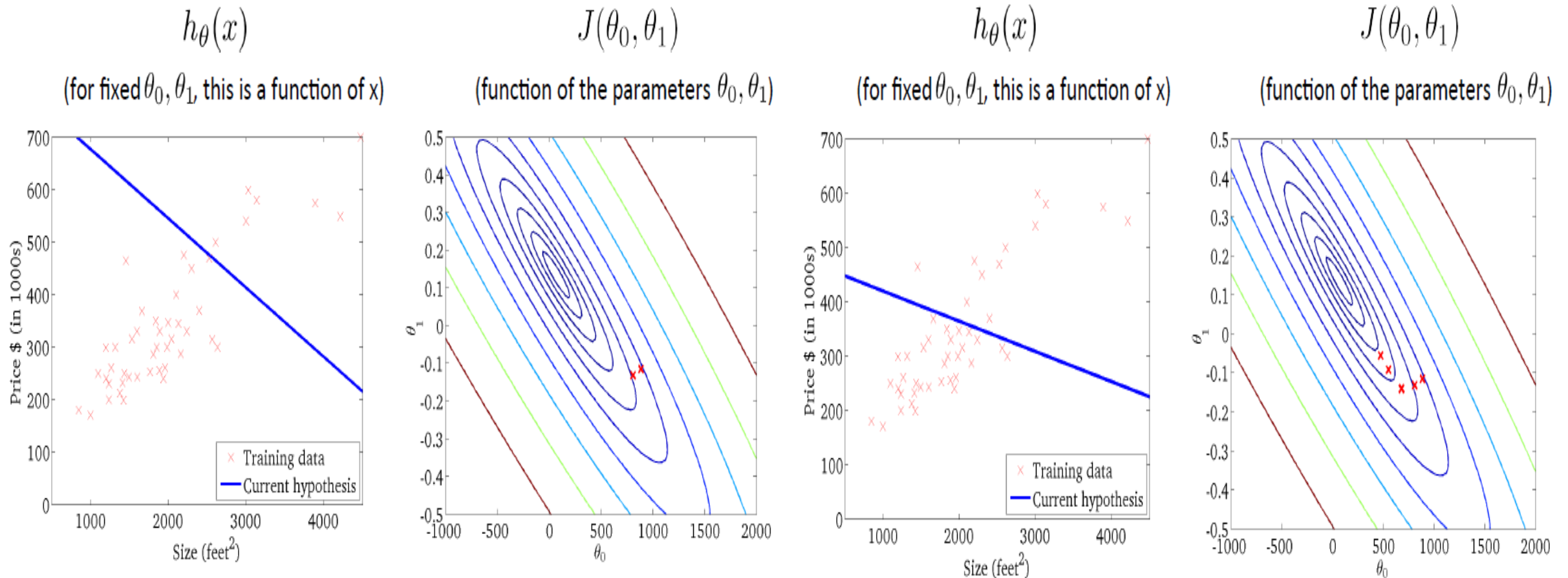


$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

학습률(learning rate)= α 는 항상 양수

- Minimum < θ 일 경우, 미분 값(>0)이 점 점 작아지며 θ 가 왼쪽으로 향한다.
- Minimum > θ 일 경우, 미분 값(<0)이 점 점 작아지며 θ 가 오른쪽으로 향한다.

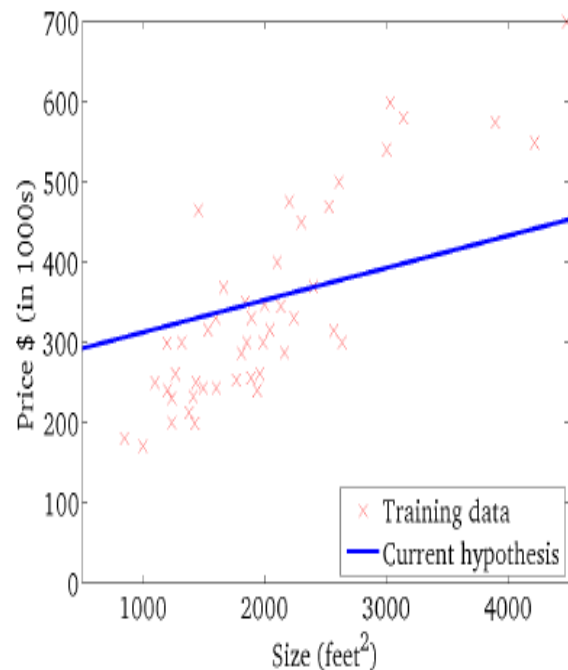
2. Hypothesis/Cost Function/Gradient Descent



2. Hypothesis/Cost Function/Gradient Descent

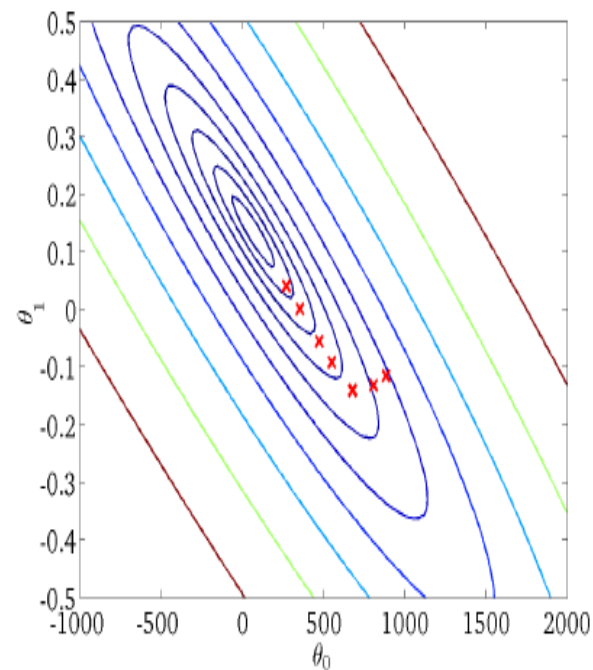
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



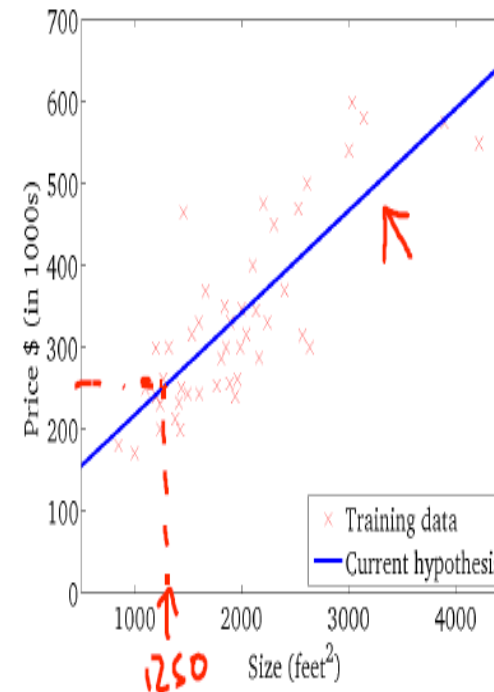
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



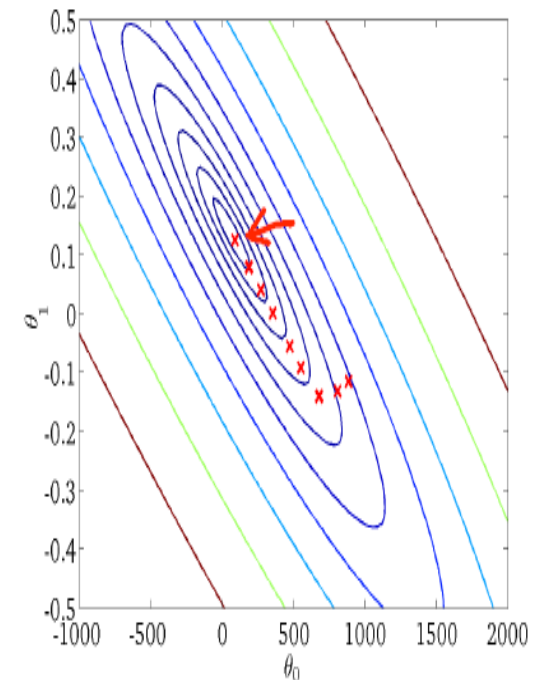
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



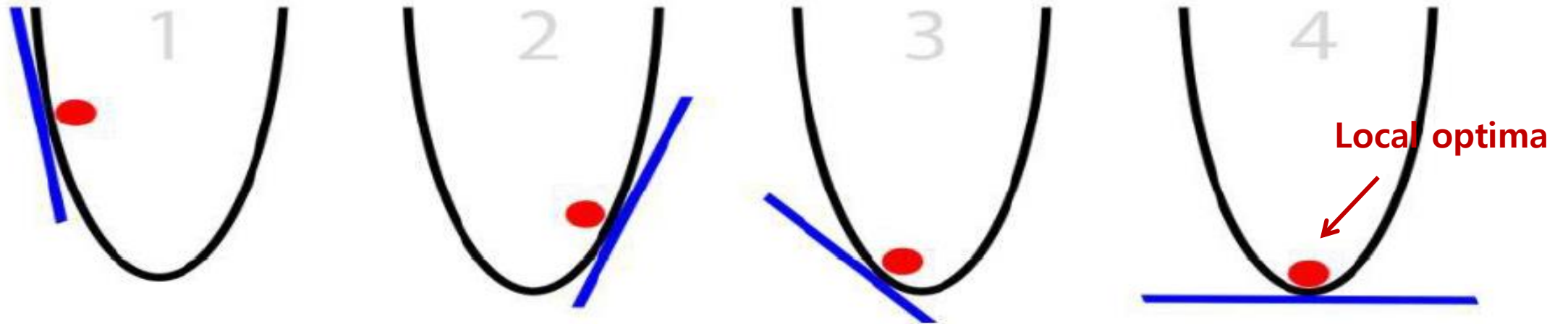
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



2. Hypothesis/Cost Function/Gradient Descent

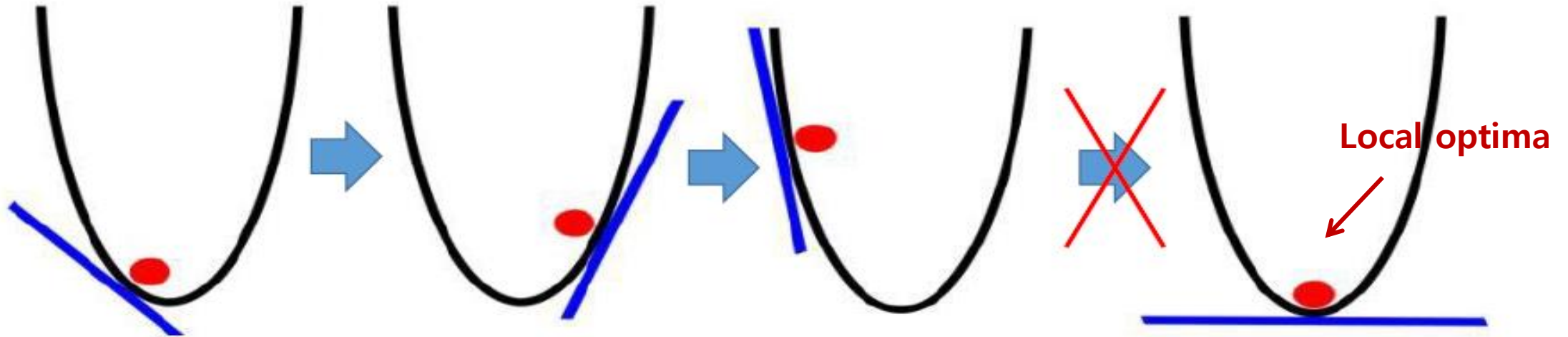
• Gradient Descent



- α 에 따라서 위치가 계속해서 바뀐다.
- Gradient descent가 local optima에 이르면 편미분항이 0이라 더이상 update되지 않는다.
- 최적값에 가까워질수록 편미분항의 크기와 gradient descent의 크기가 작아져 learning rate를 굳이 update하지 않아도 된다.

2. Hypothesis/Cost Function/Gradient Descent

• Gradient Descent



Q. 만약 learning rate(α)가 너무 크다면???

- 최적값에 가까워지지 않고 점점 최적값으로부터 멀어지게 update된다..
- 결국 최적값에 다다를 수 없게 된다.

2. Hypothesis/Cost Function/Gradient Descent

- Normal Equation

	Size (feet ²)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)
x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$
 $m \times (n+1)$

$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$
 m -dimensional vector

$\theta = (X^T X)^{-1} X^T y$

2. Hypothesis/Cost Function/Gradient Descent

- **Normal Equation**

Method to solve for θ analytically

$X\theta = y$ 를 완전히 만족하는 θ 가 optimal. $\rightarrow \therefore \theta = X^{-1}y$

But inverse of X 가 존재하지 않을 때가 있음.

$$\theta = (X^T X)^{-1} X^T y$$

2. Hypothesis/Cost Function/Gradient Descent

- Gradient Descent vs Normal Equation

	Gradient Descent	Normal Equation
Learning Rate	필요	불필요
N이 클 때(약 $n=10^6$)	잘 작동	매우 느리다
Iteration	많이 필요	X

3. 모델 예시(지도 vs 비지도)

• 모델이란?

다양한 변수 간의 수학적(or 확률적) 관계를 표현한 것
어떤 물리현상을 특정한 목적에 맞추어 이용하기 쉬운 형식으로 표현하는 일

• 지도학습

데이터에 대한 레이블(명시적인 정답)이 주어진 상태에서 컴퓨터를 학습시키는 방법

• 비지도학습

데이터에 대한 레이블(명시적인 정답)이 주어지지 않은 상태에서 컴퓨터를 학습시키는 방법론. 데이터 형태로 학습을 진행.

3. 모델 예시(지도 vs 비지도)

• 지도학습(supervised learning)

특정 타겟을 예측. 과거에 타겟 정보가 있는 데이터를 사용하여 모델 학습. 새로운 데이터를 활용하여 모델 평가

Regression	Classification
Output이 continuous일 경우	Output이 discrete일 경우

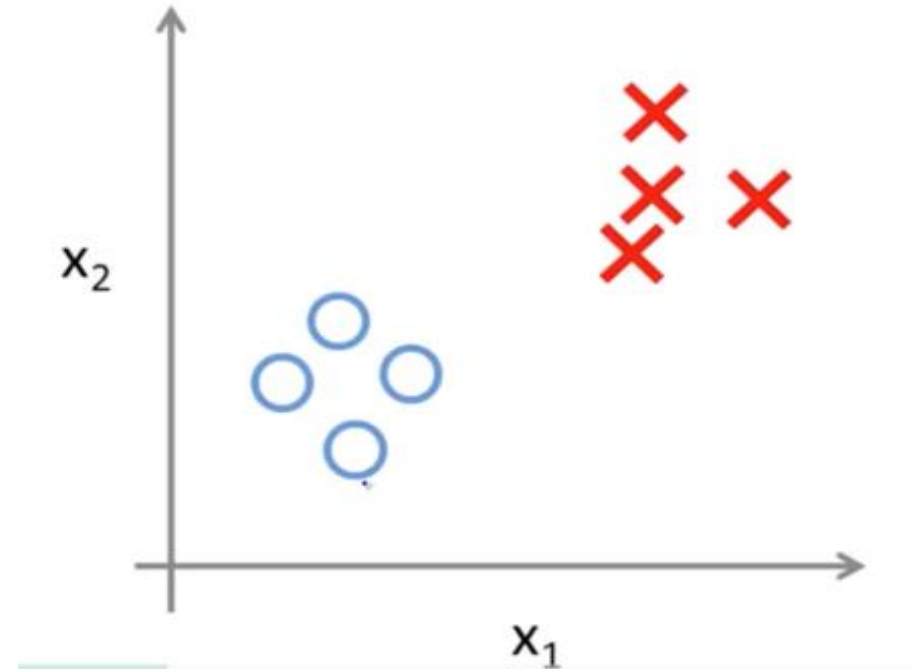
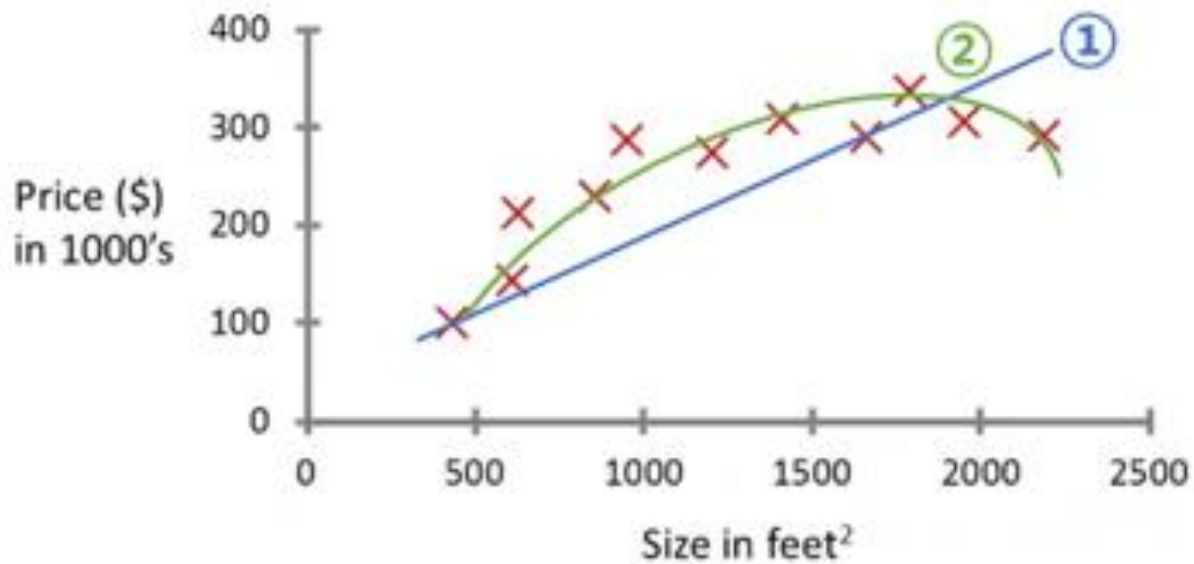
• 비지도학습(unsupervised learning)

예측하고자 하는 타겟 변수 존재하지 않음. 데이터에 내재된 특성을 분석(데이터 분포 추정, 고객 집단 구분, 연관 규칙 분석).

Clustering	Non-Clustering : 독립성분분석
유사한 데이터를 묶는 것	Cocktail party problem : 목소리 구분

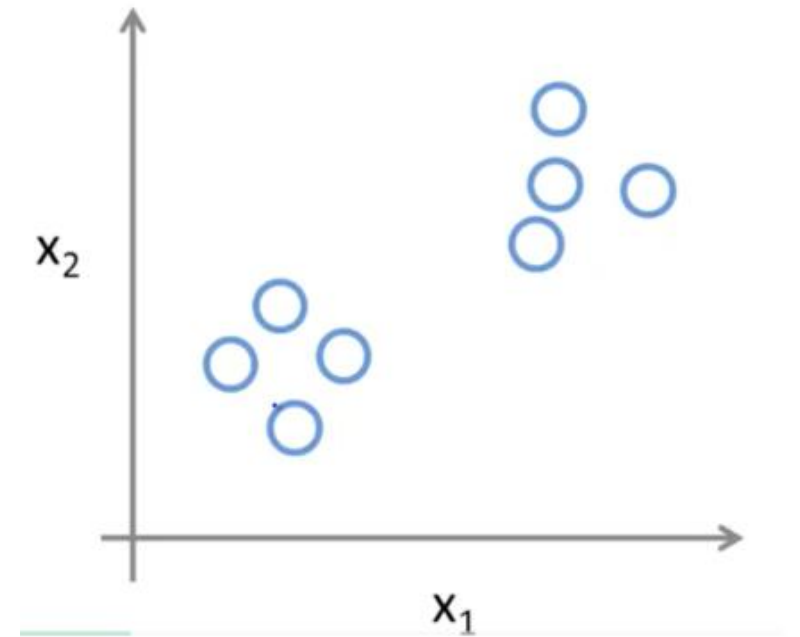
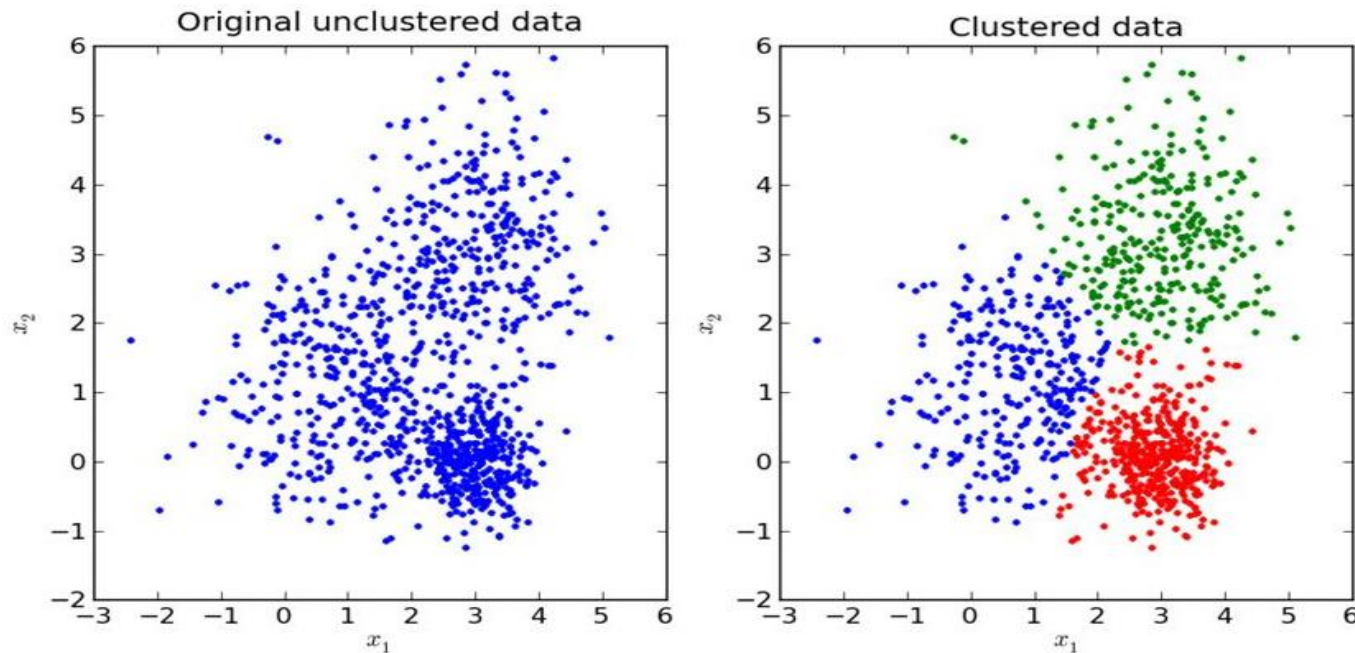
3. 모델 예시(지도 vs 비지도)

- 지도학습(supervised learning) 예시



3. 모델 예시(지도 vs 비지도)

- 비지도학습(unsupervised learning) 예시



<https://www.youtube.com/watch?v=T0HP9cxri0A>

3. 모델 예시(지도 vs 비지도)

지도학습	Classification	kNN
		Naïve Bayes
		Support Vector machine
		Decision Tree
	Regression	Linear regression
		Locally weighted linear regression
		Ridge
		Lasso
비지도학습		Clustering
		K means
		Density estimation
		Expectation maximization
		Pazen window
		DBSCAN

4. 모델 평가방법

- **test-train split**

모델의 정확성을 검증하기 위하여 training set, validation set, test set으로 나눈다. (비율은 임의대로지만 대부분 7:3 or 6:2:2로 나눈다.)

Training set으로 학습시키고 test set으로 평가한다. Validation set으로 모델이 여러 개일 때 최종 모델을 선정하기 위한 성능 평가도 한다.



4. 모델 평가방법

- test-train split

```
def split_data(data, prob):  
    results = [], []  
    for row in data:  
        results[0 if random.random() < prob else 1].append(row)  
    return results
```

```
def train_test_split(x, y, test_pct):  
    data = zip(x, y)          #Data를 두 종류로 나눔  
    train, test = split_data(data, 1 - test_pct) #데이터 셋을 나눔  
    x_train, y_train = zip(*train) #zip 풀기  
    x_test, y_test = zip(*test)  
    return x_train, x_test, y_train, y_test
```

```
model = SomeKindOfModel()  
x_train, x_test, y_train, y_test = train_test_split(xs, ys, 0.33)  
model.train(x_train, y_train)  
performance = model.test(x_test, y_test)
```

4. 모델 평가방법

- 혼동행렬

	실제 O	실제 X
분류 O	True Positive	False Positive
분류 X	False Negative	True Negative

예 : 스팸메일 분류

True Positive(TP) : 실제 스팸메일을 스팸메일로 분류

False Positive(FP) : 실제 스팸메일이 아니지만 스팸메일로 분류

False negative(FN) : 실제 스팸메일이지만 스팸이 아닌 것으로 분류

True negative(TN) : 실제 스팸메일이 아니고 스팸메일이 아니라 분류

4. 모델 평가방법

<정확성 지표>

1. 정확도

모델이 정확하게 양성 또는 음성으로 예측한 비율

$$(TP+TN)/Total$$

3. 재현율

실제 양성으로 예측한 것 중 실제 양성인 비율

$$TP/(TP+FN)$$

2. 정밀도(검정력)

모델이 양성으로 예측한 것 중 실제 양성인 비율

$$TP/(TP+FP)$$

4. F1점수

정밀도와 재현율의 조화평균, 항상 정밀도와 재현율 사이의 값을 가짐

$$2*p*r/(p+r)$$

4. 모델 평가방법

<정확성 지표 코드>

1. 정확도

```
def accuracy(tp, fp, fn, tn):  
    correct = tp + tn  
    total = tp + fp + fn + tn  
    return correct / total  
  
print(accuracy(70, 4930, 13930, 981070))
```

2. 정밀도(검정력)

```
def precision(tp, fp, fn, tn):  
    return tp / (tp + fp)  
  
print(precision(70, 4930, 13930, 981070))
```

3. 재현율

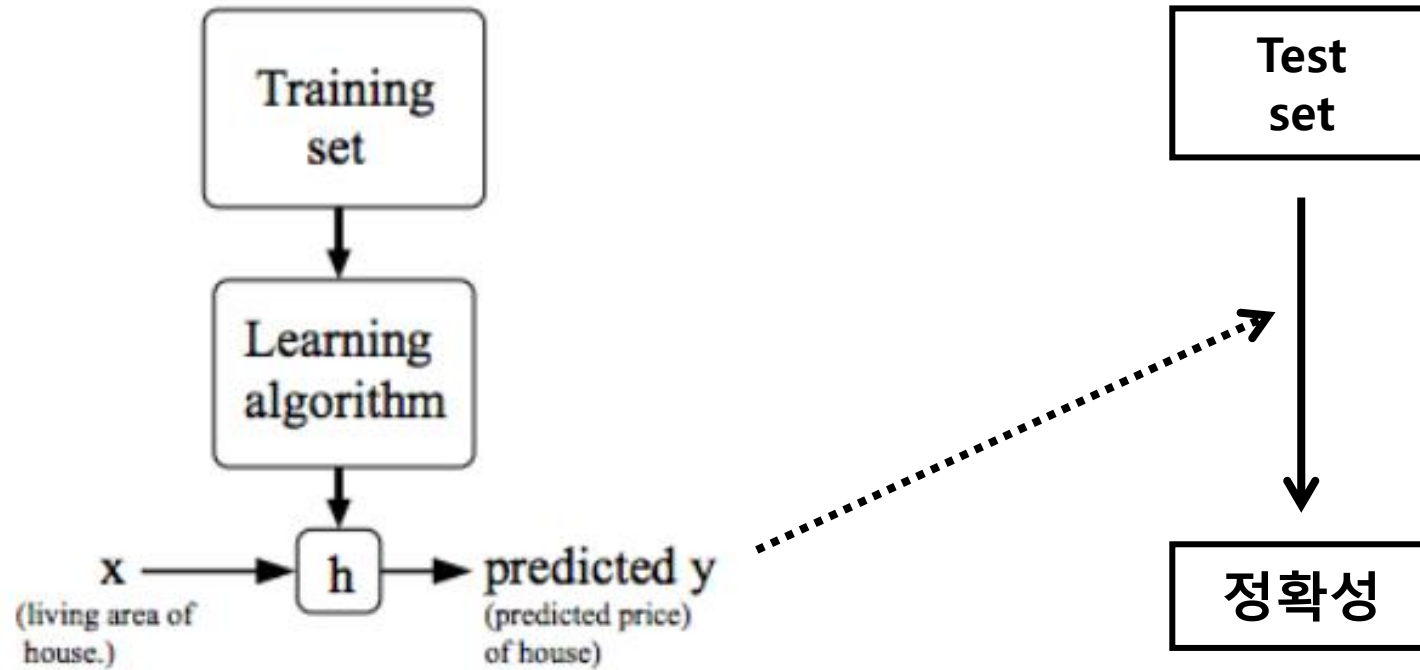
```
def recall(tp, fp, fn, tn):  
    return tp / (tp + fn)  
  
print(recall(80, 4930, 13930, 981070))
```

4. F1점수

```
def f1_score(tp, fp, fn, tn):  
    p = precision(tp, fp, fn, tn)  
    r = recall(tp, fp, fn, tn)  
  
    return 2 * p * r / (p + r)
```

4. 모델 평가방법

- test-train split



4. 모델 평가방법

• Overfitting vs Underfitting



Overfitting

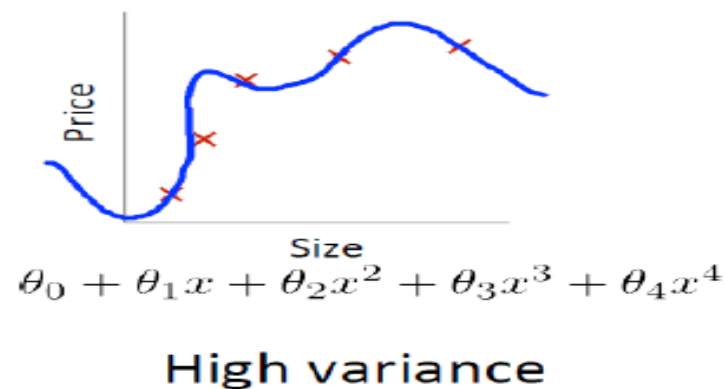
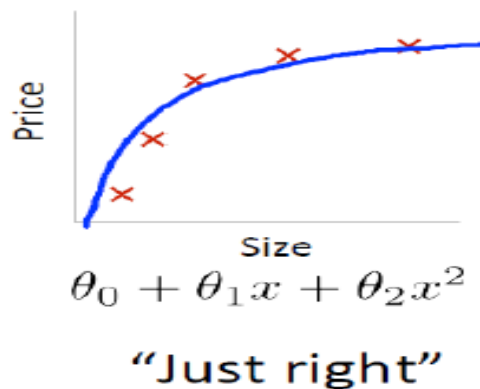
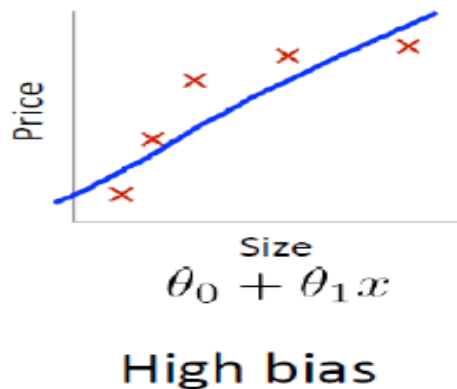
모델의 성능이 학습데이터에는 좋지만, 새로운 데이터에 대해서는 좋지 않은 경우

Underfitting

모델의 성능이 학습데이터에도 좋지 않은 경우

4. 모델 평가방법

- bias vs variance



- High bias

1. 새로운 feature 추가하기
2. polynomial feature 추가하기

- High variance

1. 학습데이터 양을 늘리기
2. 쓸데없는 변수 줄이기

5. 코드

• 필요 모듈

1. **Matplotlib** : matlab과 비슷한 인터페이스를 가진 라이브러리로 그래프를 그릴 수 있게 해준다.
2. **Numpy** : X변수 설정을 위해 필요하다

numpy.linspace

`numpy.linspace` (`start`, `stop`, `num=50`, `endpoint=True`, `retstep=False`, `dtype=None`) [\[source\]](#)

Return evenly spaced numbers over a specified interval.

Returns `num` evenly spaced samples, calculated over the interval `[start, stop]`.

The endpoint of the interval can optionally be excluded.

```
xx = np.linspace(-10, 10, 501)
```

임의의 xx변수에 x값 할당해준다.
(-10,10)범위에서 같은 간격으로 500개 뽑아 x에
입력값의 범위를 설정해준다.

3. **Scipy** : 최저점을 찾게 해주는 모듈

scipy.optimize.minimize

`scipy.optimize.minimize` (`fun`, `x0`, `args=()`, `method=None`, `jac=None`, ...)

5. 코드

• 1차원 함수-그래프 나타내기

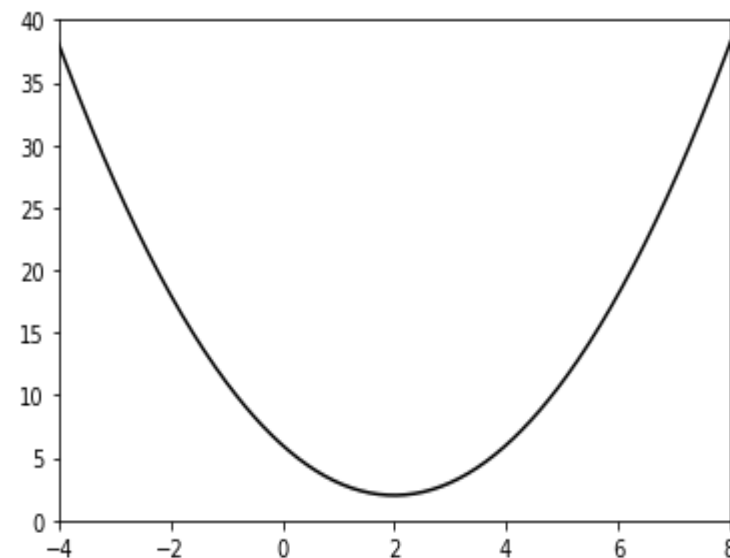
(x,y) 데이터 셋에서 $Y=aX+b$ 로 모델을 가정하였다.

Cost Function : $(x-2)^2+2$

```
from matplotlib import pyplot as plt
import numpy as np

#1변수 함수 정의
def f1(x):
    return (x-2)**2+2

xx=np.linspace(-10,10,501) #(-10,10)범위에서 같은 간격으로 500개 뽑아 점 찍기
plt.plot(xx,f1(xx),'k') #k는 선으로 잇는 것, xx값에 x를 할당해서 f1 그래프 그리기
plt.xlim(-4,8) #x범위 (-4,8)범위
plt.ylim(0,40) #y범위 (0,40)범위
plt.show() #그래프 보여주기
```



5. 코드

• 1차원 함수-Gradient Descent

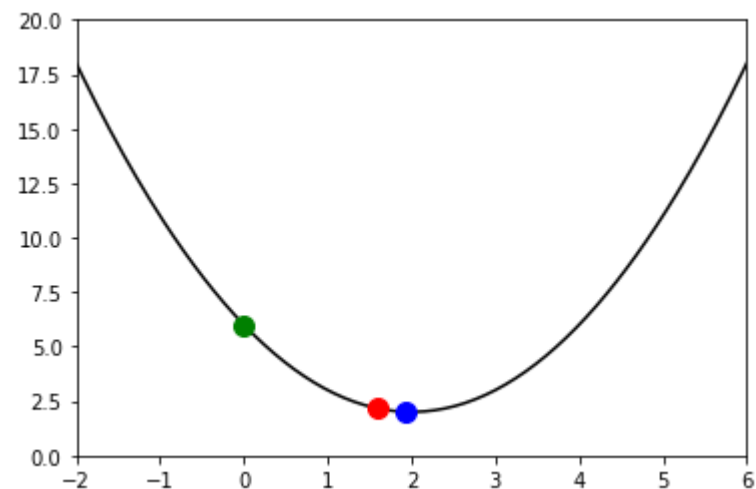
```
#f1을 미분한 함수 정의
def f1d(x):
    return 2*(x-2)

plt.plot(xx,f1(xx),'k')
alpha=0.4 #learning rate 설정
x=0 #초기값 설정
plt.plot(x,f1(x),'go',markersize=10) #go는 점을 찍으란 소리, 사이즈는 10

x=x-alpha*f1d(x) #gradient descent
plt.plot(x,f1(x),'go',markersize=10,color='r')#색깔은 red

x=x-alpha*f1d(x) #gradient descent
plt.plot(x,f1(x),'go',markersize=10,color='b')#색깔은 blue

plt.xlim(-2,6)
plt.ylim(0,20)
plt.show()
```



5. 코드

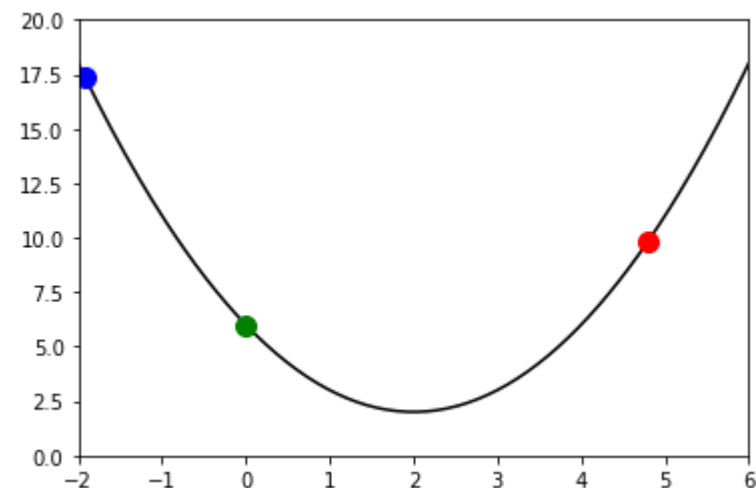
- 1차원 함수-Gradient Descent(learning rate가 크다면?)

```
plt.plot(xx,f1(xx),'k')
alpha=1.2 #learning rate 설정
x=0 #초기값 설정
plt.plot(x,f1(x),'go',markersize=10) #go는 점을 찍으란 소리, 사이즈는 10

x=x-alpha*f1d(x) #gradient descent
plt.plot(x,f1(x),'go',markersize=10,color='r') #색깔은 red

x=x-alpha*f1d(x) #gradient descent
plt.plot(x,f1(x),'go',markersize=10,color='b') #색깔은 blue

plt.xlim(-2,6)
plt.ylim(0,20)
plt.show()
```



5. 코드

• 1차원 함수-Gradient Descent(if, while문)

```
alpha=0.4
x=0

temp=x-alpha*f1d(x)

if (f1(temp)>f1(x)): #알파값이 너무 클 때 크다고 알려주는 메세지 나오게 하기
    print("learning rate is too large!")
else :
    while(True):
        temp=x-alpha*f1d(x)
        if ((temp-x)<0.005): #temp와 x의 차이가 매우 얼마나지 않을 때까지 실행
            break
        x=temp

print(x,f1(x))
```

1.9968 2.00001024

5. 코드

- 1차원 함수-scipy이용

```
from scipy import optimize as op #scipy모듈 불러오기
result=op.minimize(f1,1) #바로 알 수 있다.
print(result)
```

```
      fun: 2.0
hess_inv: array([[0.5]])
      jac: array([0.])
message: 'Optimization terminated successfully.'
      nfev: 9
       nit: 2
      njev: 3
  status: 0
success: True
       x: array([1.99999999])
```

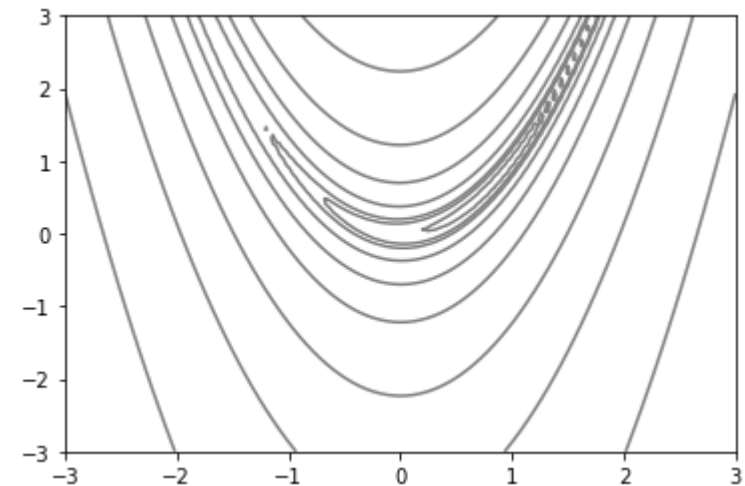
5. 코드

• 2차원 함수-그래프 그리기

(x,y,z) 데이터 셋에서 모델을 가정하였다.

Cost Function : $(1-x)^2 + 100 \cdot (y-x^2)^2$

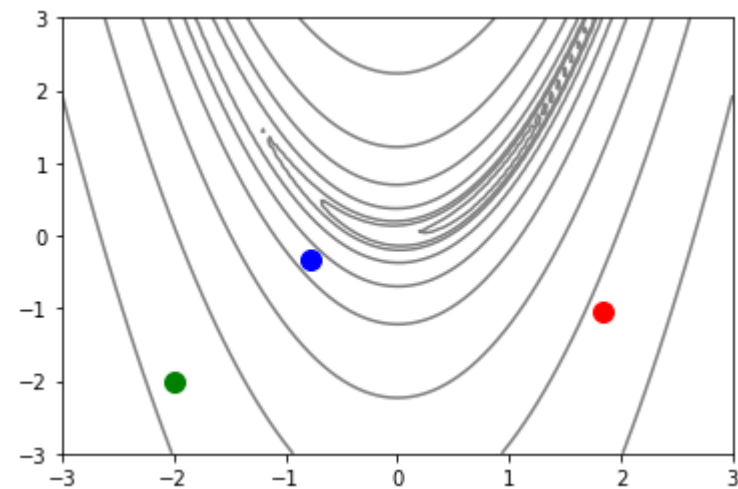
```
def f2(x,y):  
    return (1-x)**2 + 100*(y-x**2)**2  
  
xx=np.linspace(-3,3,101)  
yy=np.linspace(-3,3,101)  
X,Y=np.meshgrid(xx,yy) #2차원함수 그리기 쉽도록 meshgrid이용(참조바람)  
Z=f2(X,Y)  
  
plt.contour(X,Y,Z,colors='gray',levels=[0.7,3,5,15,50,150,500,1500,5000]) #등고선 그리기  
plt.show
```



5. 코드

• 2차원 함수-Gradient Descent

```
def f2d(x,y):  
    return np.array([2*x-2-400*x*(y-x**2),200*(y-x**2)])  
plt.contour(X,Y,Z,colors='gray',levels=[0.7,3,5,15,50,150,500,1500,5000])  
  
alpha=8e-04  
x=-2  
y=-2  
g=f2d(x,y)  
print(g)  
  
plt.plot(x,y,'go',markersize=10)  
  
x=x-alpha*g[0]  
y=y-alpha*g[1]  
g=f2d(x,y)  
plt.plot(x,y,'go',markersize=10,color='r')  
  
x=x-alpha*g[0]  
y=y-alpha*g[1]  
g=f2d(x,y)  
plt.plot(x,y,'go',markersize=10,color='b')  
  
plt.show()
```



5. 코드

- 2차원 함수-Gradient Descent(if, while문)

```
alpha=8e-04
x=-2
y=-2
g=f2d(x,y)

temp=(x-alpha*g[0],y-alpha*g[1])

if(f2(temp[0],temp[1])>f2(x,y)):
    print("learning rate is too large!")
else :
    while(True):
        g=f2d(x,y)
        temp=(x-alpha*g[0],y-alpha*g[1])
        if(f2(x,y)-f2(temp[0],temp[1])<0.000000000005):
            break
        x=temp[0]
        y=temp[1]

print(x,y,f2(x,y))
```

0.9997205483173297 0.9994400562865879 7.821833404464664e-08

5. 코드

- 2차원 함수-Gradient Descent(Scipy 이용)

```
def f3(x):  
    return (1-x[0])**2 +100*(x[1]-x[0]**2)**2  
  
result=op.minimize(f3,(2,2))  
print(result)
```

```
fun: 1.8932893809017893e-11  
hess_inv: array([[0.51675994, 1.03186494],  
                 [1.03186494, 2.0655726 ]])  
jac: array([ 5.27380711e-06, -2.50575298e-06])  
message: 'Optimization terminated successfully.'  
nfev: 140  
nit: 30  
njev: 35  
status: 0  
success: True  
x: array([0.99999565, 0.99999129])
```

Quest

**'caschool.csv' 데이터에서 str과 avginc 변수를 통해 read_scr을
예측할 수 있는 $y=a*x_1+b*x_2+c$ 모형으로 Cost function을
minimize해보세요.**

(Scipy.optimize.minimize이용 권장)