SESSION # 03

By Team 3
남정우 이영준 차유경

Date _ 2018.04.10

CONTENTS

01 모델링과 기계학습

모델이란?

다양한 변수 간의 수학적(혹은 확률적) 관계를 표현한 것

어떤 물리현상을 특정한 목적에 맞추어 이용하기 쉬운 형식으로 표현하는 일

01 모델링과 기계학습

인공지능이란?

규칙기반(rule based)의 접근법 :
사람이 직접 컴퓨터에게 지능적으로 행동하는 방법을 알려줌

–〉 인간이 원하는 것은 스스로 사고하고 행동하는 지능적인 현상.
But, 사람이 직접 모든 일을 알려주는 것으로 이런 목표를 이루기가 힘들다는 한계가 있음.

01 모델링과 기계학습

기계학습(machine learning)이란?

머신 러닝: 인공 지능을 구현하는 구체적 접근 방식

사람이 직접 생각하는 방법을 알려주는 것이 아니라 기계가 스스로 배우도록 하는 것
–〉 컴퓨터로 하여금 알고리즘을 기반으로 학습하게 한 뒤, 새로운 데이터가 들어왔을 때 데이터의 결과를 예측하도록 만드는 것

–〉 대량의 데이터를 처리하고 이를 통해 학습할 수 있는 알고리즘을 구현함으로써 미리 프로그램되지 않은 부분에 대해서도 예측과 결정을 내릴 수 있게 하는 방식

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Tom Mitchell

(참고)딥러닝이란?

완전한 머신 러닝을 실현하는 기술

머신러닝의 일종

인간 두뇌에 있는 신경세포들의 연결방식에서 영감을 얻은 것

딥 러닝은 인공신경망에서 발전한 형태의 인공 지능으로, 뇌의 뉴런과 유사한 정보 입출력 계층을 활용해 데이터를 학습

Example) 사진에 있는 사람들 얼굴 식별

코, 눈동자 등과 같은 개별적인 특징을 제공할 필요X

사진 전체를 제공하면 이를 검토해 여러 특징을 이해함으로써 사진 내용을 독자적으로 예측

01 모델링과 기계학습

기계 학습의 학습방법

지도학습(supervised learning)	비지도학습(unsupervised learning)
• 특정한 타겟을 예측하는 것 • 독립변수 x와 종속변수 y의 상관관계를 찾음. • 과거에 타겟 정보가 있는 데이터를 사용하여 모델 학습 • 새로운 데이터를 활용하여 모델 평가	• 데이터에 내재된 특성을 분석 • 데이터의 분포 추정, 고객 집단 구분, 연관 규칙 분석 등 • 예측하고자 하는 지정된 타켓 변수가 존재하지 않음

지도학습: 데이터에 대한 레이블(Label)–명시적인 정답이 주어진 상태에서 컴퓨터를 학습시키는 방법

비지도학습: 데이터에 대한 레이블(Label)–명시적인 정답이 주어지지 않은 상태에서 컴퓨터를 학습시키는 방법론. (데이터(data)) 형태로 학습을 진행

01 모델링과 기계학습

기계 학습의 학습방법

지도학습(supervised learning)	비지도학습(unsupervised learning)
• 특정한 타겟을 예측하는 것 • 독립변수 x와 종속변수 y의 상관관계를 찾음. • 과거에 타겟 정보가 있는 데이터를 사용하여 모델 학습 • 새로운 데이터를 활용하여 모델 평가	• 데이터에 내재된 특성을 분석 • 데이터의 분포 추정, 고객 집단 구분, 연관 규칙 분석 등 • 예측하고자 하는 지정된 타켓 변수가 존재하지 않음

준지도 학습(Semi-Supervised Learning)

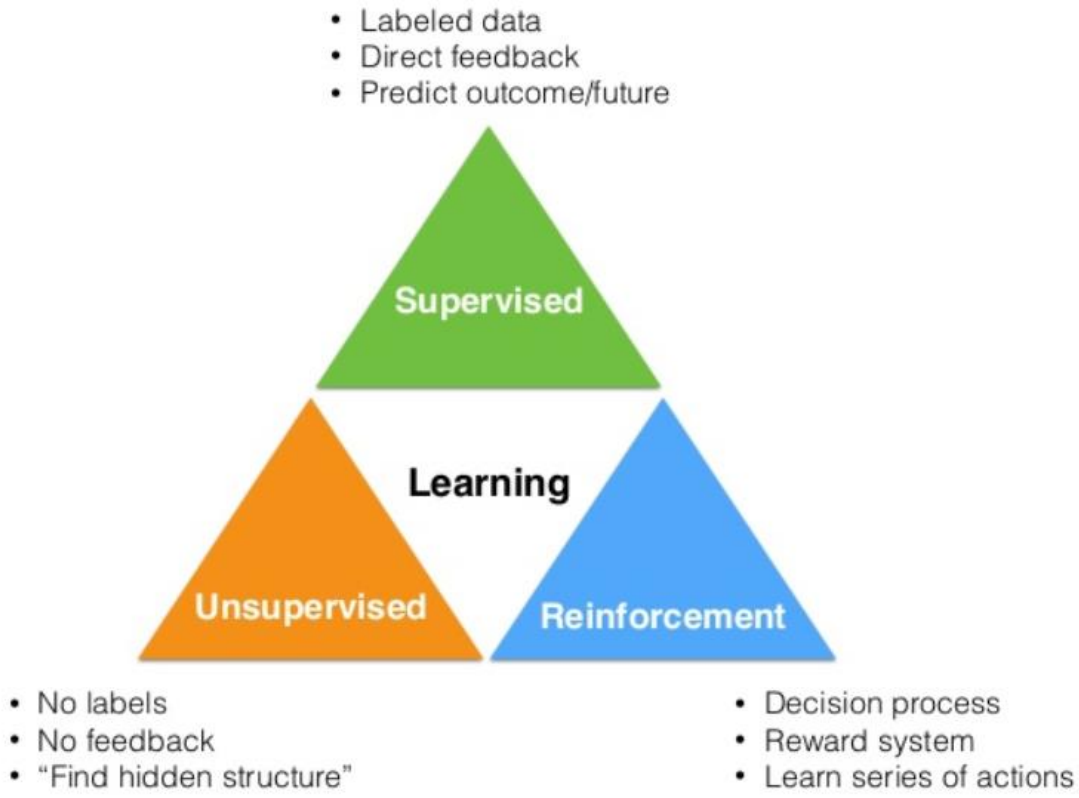
- 지도 학습과 비지도 학습을 섞는 방법
- 우선 기계가 지도 학습으로 특징값을 산출하게 하고 그 이후 비지도 학습으로 방대한 훈련 데이터를 제공해 자동으로 특징값을 산출하게 하여 반복 학습을 하는 방법

강화학습(Reinforcement Learning)

- 주어진 상황에서 보상(reward)을 최대로 만드는 액션을 찾는 문제

01 모델링과 기계학습

기계 학습의 학습방법



기계 학습의 알고리즘

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

01 모델링과 기계학습

지도학습(supervised learning)

Regression	Classification
Output이 continuous 한 값을 가진 경우 = 어떤 연속 함수를 찾는 과정	Output이 discrete 한 값을 가진 경우

비지도학습(supervised learning)

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

01 모델링과 기계학습

가설(hypothesis)

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

$$h_\theta(x) = \theta_0 + \theta_1 x$$ Univariate linear regression

Cost Function

비용에 관련된 모든 변량에 대하여 어떤 관계를 나타내는 함수. 즉, 최적화를 위해 사용되는 복잡한 조건의 스칼라 측정을 말함

주어진 데이터에 가장 잘 '맞는' 직선을 선택하기 위한 일정한 기준

(hypothesis function의 정확도를 측정하기 위해 cost function)

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

01 모델링과 기계학습

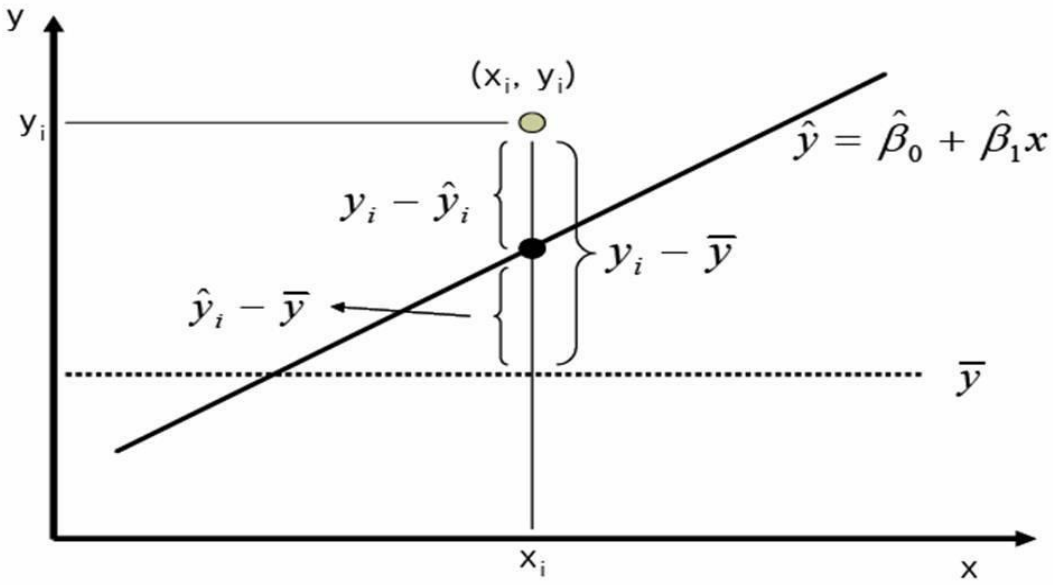
$$y_i - \bar{y} = (y_i - \hat{y_i}) + (\hat{y_i} - \bar{y})$$

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

총제곱합	잔차제곱합	회귀제곱합	
SST	SSE	SSR	
$n-1$	$n-2$	1	(자유도)

*SST (Total sum of squares)

*SSR (Regression sum of squares)



cf) Matlab 용어
SST = TSS

SSR (Regression Sum of Squares) = ESS (*Explained* Sum of Squares)

SSE (Error Sum of Squares) = RSS (*Residual* Sum of Squares)

Gradient Descent

- Hypothesis function의 최적의 parameter을 찾는 방법

$h_\theta(x) = \theta_0 + \theta_1 x$ 에서 $\theta_0\ \theta_1$ 값을 추정하는 방법

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

$= \frac{1}{2m}\sum_{i=1}^{m}((\theta_0 + \theta_1 x^{(i)}) - y^{(i)})^2$

$\min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$

전략

- 어떤 parameter에서든 시작 가능
- 계속 이 parameter를 변화해가면서 최소의 J를 찾는 것

Gradient Descent

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

$\alpha = Learning\ Rate$ (학습율)

Update rules:

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

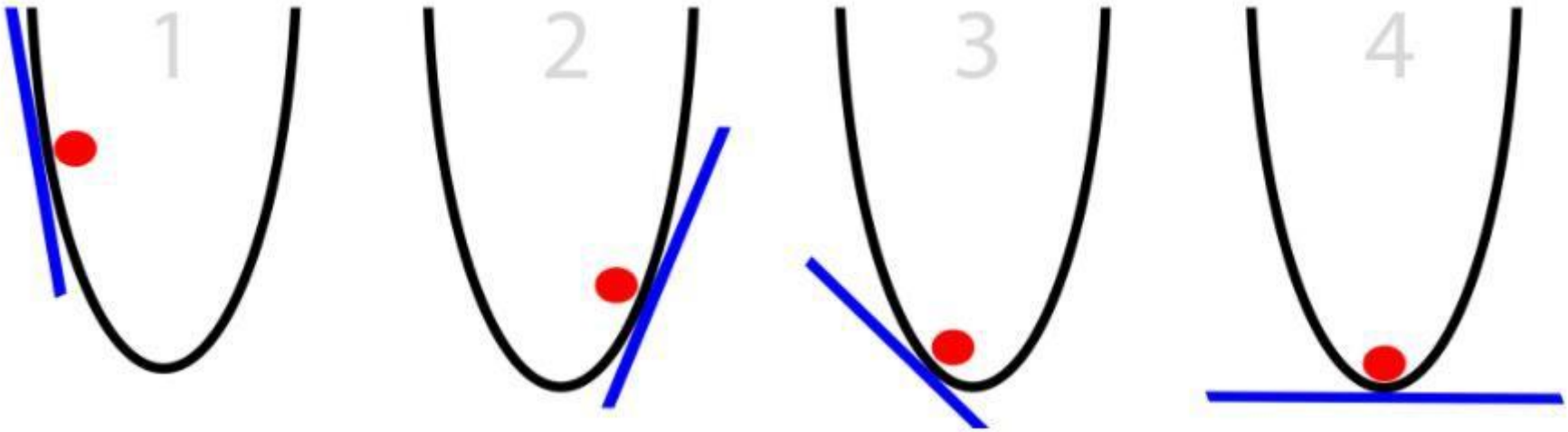
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_0, \theta_1)$$

Derivatives:

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

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

Gradient Descent



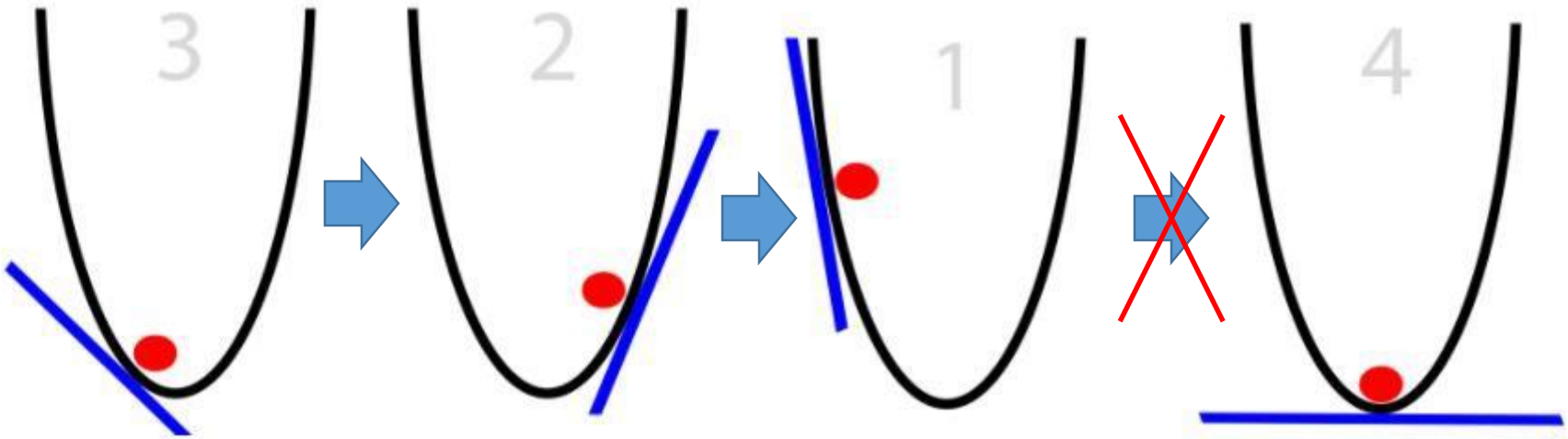
- Gradient descent 가 local optima 에 이르면 편미분항이 0

–〉 더 이상 update 되지 않음

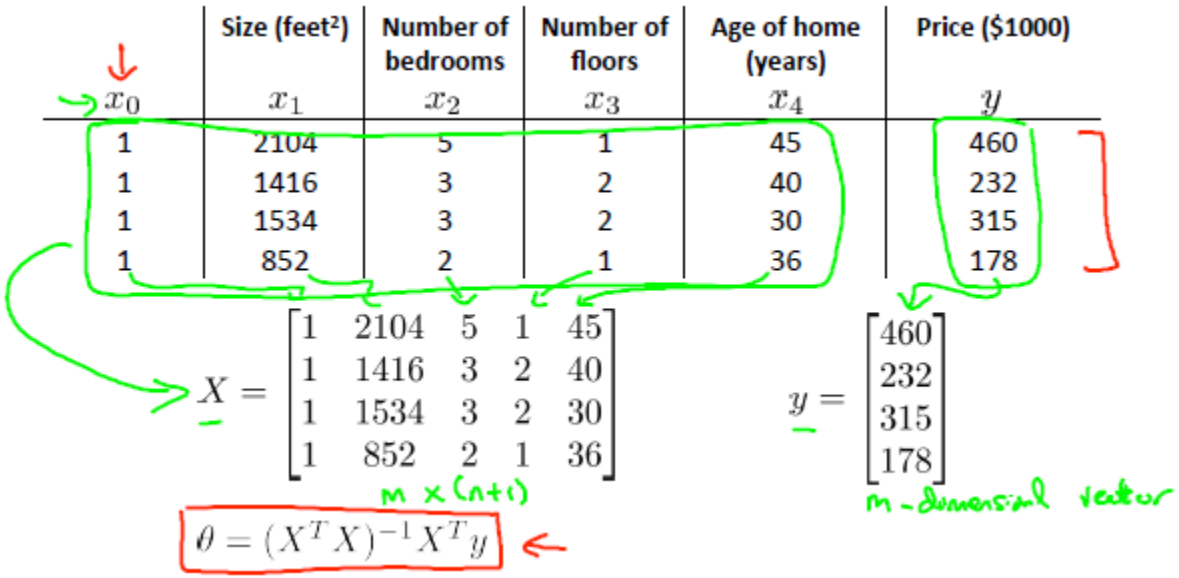
최적값에 가까워질수록 편미분항의 크기와 gradient descent의 크기가 작아지므로, learning rate를 따로 update하지 않아도 됨

Gradient Descent

If alpha is too big?



Normal Equation



$X\theta = y$ 를 완전히 만족하는 theta가 optimal.

$\therefore \theta = X^{-1} y$, but inverse of X가 존재하지 않으면??

Normal Equation

$X\theta = y$ 를 완전히 만족하는 theta가 optimal.

$\therefore \theta = X^{-1}y$, but inverse of X가 존재하지 않으면??

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

Cf) pinv in MATLAB/Octave

Gradient Descent vs Normal Equation

	Gradient Descent	Normal Equation
Learning Rate	필요.	불필요.
# of training ex. (m) (if Large)	X	O
# of features (n) (if Large)	O	X
Iteration	O	X

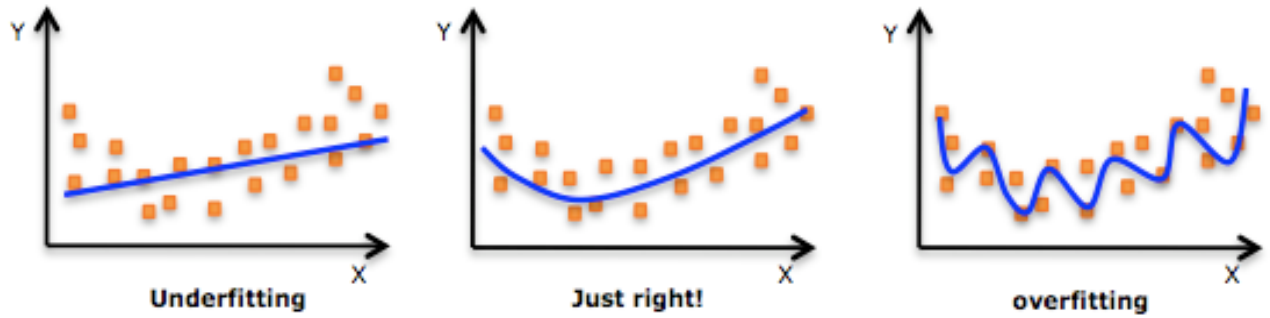
02 Overfitting / Underfitting, Bias / Variance

Overfitting

모델의 성능이 학습데이터에는 좋지만, 새로운 데이터에 대해서는 좋지 않은 경우

Underfitting

모델의 성능이 학습데이터에도 좋지 않은 경우



02 Overfitting / Underfitting, Bias / Variance

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

```python
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
```

```python
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)
```

복잡하지 않은 모델을 어떻게 만들까?

주어진 데이터를 학습과 평가라는
두 가지 목적으로 나누어 활용

평가에서 성능이 좋다면 오버피팅 X

여러 개의 모델을 평가할 때는 '검증'
데이터를 구분할 필요

02 Overfitting / Underfitting, Bias / Variance

Confusion Matrix 혼동행렬

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

이 메일은 과연 스팸메일인가?

True positive(TP) : 실제로 스팸메일이며 정확하게 스팸으로 분류

False positive(FN): 스팸 메일이 아니지만 스팸메일로 분류

False negative(FP): 실제로 스팸메일이지만 스팸이 아닌 것으로 분류

True negative(TN): 스팸 메일이 아니며 정확하게 스팸이 아닌 것으로 분류

02 Overfitting / Underfitting, Bias / Variance

```python
#정확도
def accuracy(tp, fp, fn, tn):
    correct = tp + tn
    total = tp + fp + fn + tn
    return correct / total

print(accuracy(70, 4930, 13930, 981070))
```

```python
#정밀도
def precision(tp, fp, fn, tn):
    return tp/ (tp + fp)

print(precision(70, 4930, 13930, 981070))
```

정확도: 모델이 정확하게 양성 또는 음성으로 예측한 비율= (TP + TN) / Total

정밀도: 모델이 양성으로 예측한 것 중 실제 양성인 비율= TP / (TP + FP)

=〉통계적으로 '검정력' 이라고도 함

재현율: 실제 양성 중 모델이 정확하게 양성으로 예측한 비율 =TP / (TP + FN)

F1 점수: 정밀도와 재현율의 조화평균, 항상 정밀도와 재현율 사이의 값을 가짐 = 2*p*r / (p + r)

02 Overfitting / Underfitting, Bias / Variance

```python
#재현율
def recall(tp, fp, fn, tn):
    return tp/ (tp + fn)

print(recall(80, 4930, 13930, 981070))
```

```python
#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)
```

정확도: 모델이 정확하게 양성 또는 음성으로 예측한 비율= (TP + TN) / Total

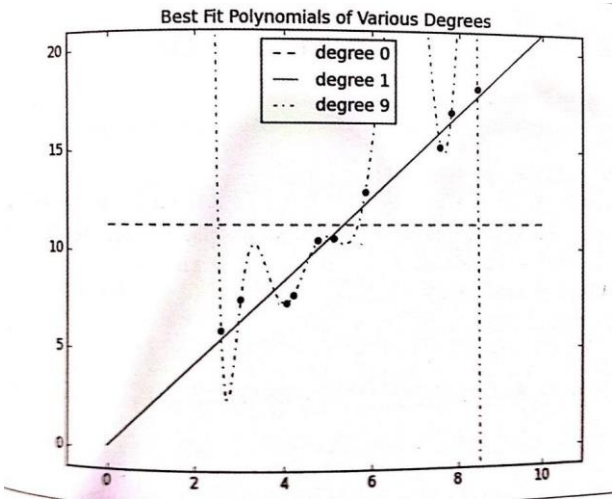
정밀도: 모델이 양성으로 예측한 것 중 실제 양성인 비율= TP / (TP + FP)

=〉통계적으로 '검정력' 이라고도 함

재현율: 실제 양성 중 모델이 정확하게 양성으로 예측한 비율 =TP / (TP + FN)

F1 점수: 정밀도와 재현율의 조화평균, 항상 정밀도와 재현율 사이의 값을 가짐 = 2*p*r / (p + r)

02 Overfitting / Underfitting, Bias / Variance



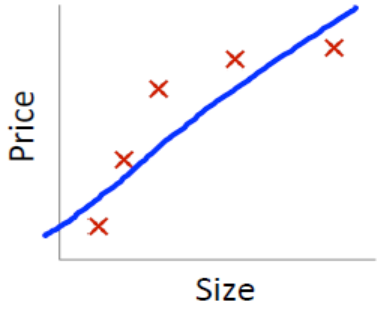
Bias / Variance

상수함수 : 항상 같은 값 반환
Bias가 높고, variance가 낮다
=〉 underfitting

9차함수 : 완벽하게 학습데이터 통과
Variance가 높고, bias가 낮다
=〉 overfitting

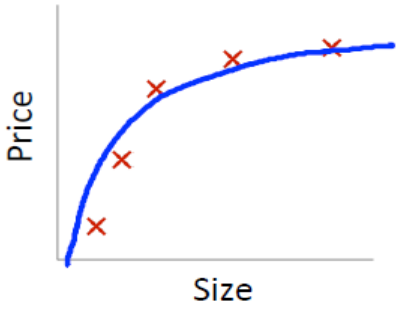
02 Overfitting / Underfitting, Bias / Variance

Bias Variance Trade off



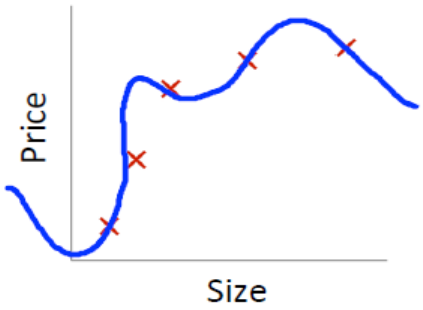
$\theta_0 + \theta_1 x$

High bias



$\theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High variance

02 Overfitting / Underfitting, Bias / Variance

Bias Variance Trade off

Bias가 높을 때:

새로운 feature추가하기
polynomial feature추가하기

Variance가 높을 때:

학습데이터 양을 늘리기
쓸데없는 변수 줄이기

03 Code로 적용하기

1변수함수 그래프로 그리기

```
1  from matplotlib import pyplot as plt
2  import numpy as np
3
4  def f1(x):
5      return (x-2)**2+2        # 1변수 함수 정의
```

Matplotlib, numpy 라이브러리 불러오기

- matlab과 비슷한 인터페이스를 가진 라이브러리
- pyplot이라는 sub-package를 통해 그래프 그림
- Numpy 라이브러리는 x 변수 설정을 위해 필요 (후술)

1차원 함수 정의

$$f(x) = (x-2)^2+2$$

03 Code로 적용하기

1변수함수 그래프로 그리기

```
 7  xx = np.linspace(-10,10,501)    # 점 찍기.
 8  plt.plot(xx,f1(xx),'k')      # 그래프 그리기.
 9  plt.xlim(-2,6)               # x범위 지정
10  plt.ylim(0,20)               # y범위 지정
11  plt.show()
12
```

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.

x값 할당

임의의 xx변수에 x값 할당.

x에 입력 값의 범위를 설정해야 함

–〉 (–10,10) 범위에서 같은 간격으로 500개 뽑음

–〉 첫값과 마지막값을 포함하므로 501 입력

그래프 그리기

X값에 따른 y값의 변화를 그래프로 확인

'k' 는 선으로 이은 그래프 그리는 법

plt.plot() 그래프 그리기

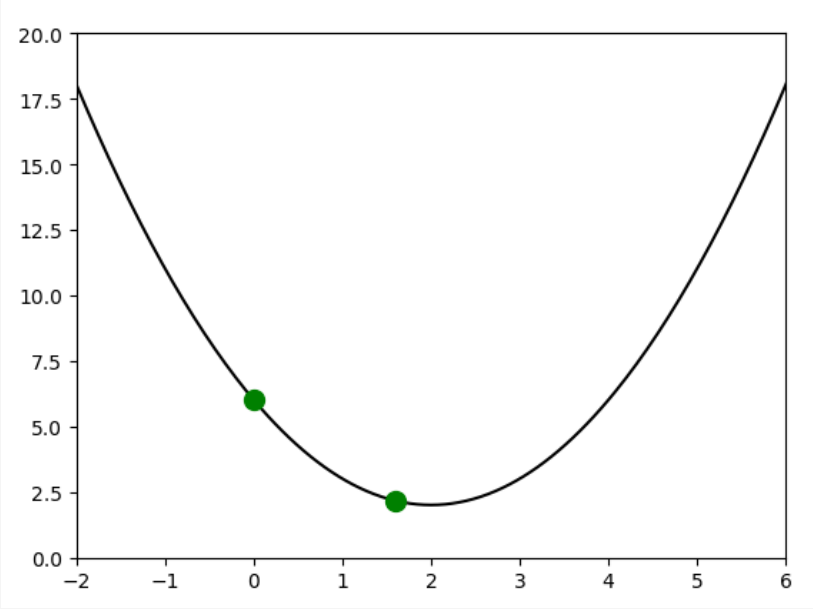
plt.xlim(), plt.ylim() x,y범위 지정

plt.show() 그래프출력

03 Code로 적용하기

GD로 1변수함수 최저점 찾기

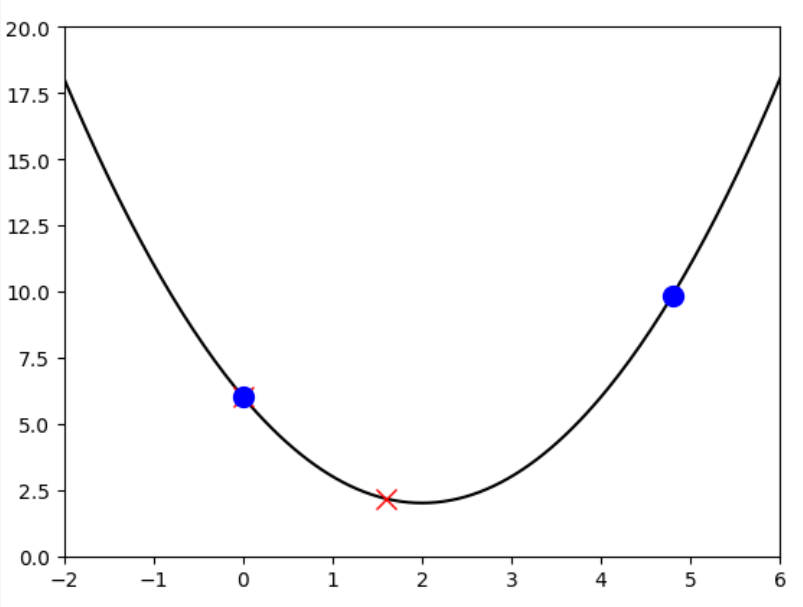
```
13  def f1d(x):
14      return 2*(x-2)          # f(x) 도함수 정의
15
16  plt.plot(xx,f1(xx),'k')
17  mu = 0.4                    # learning rate
18  x=0
19  plt.plot(x,f1(x),'go',markersize=10)
20  #go: 점을 찍어라.
21  x =x-mu*f1d(x)
22  plt.plot(x,f1(x),'go',markersize=10)
23
24  plt.xlim(-2,6)
25  plt.ylim(0,20)
26  plt.show()
```



03 Code로 적용하기

GD로 1변수함수 최저점 찾기
But, if mu is too large?

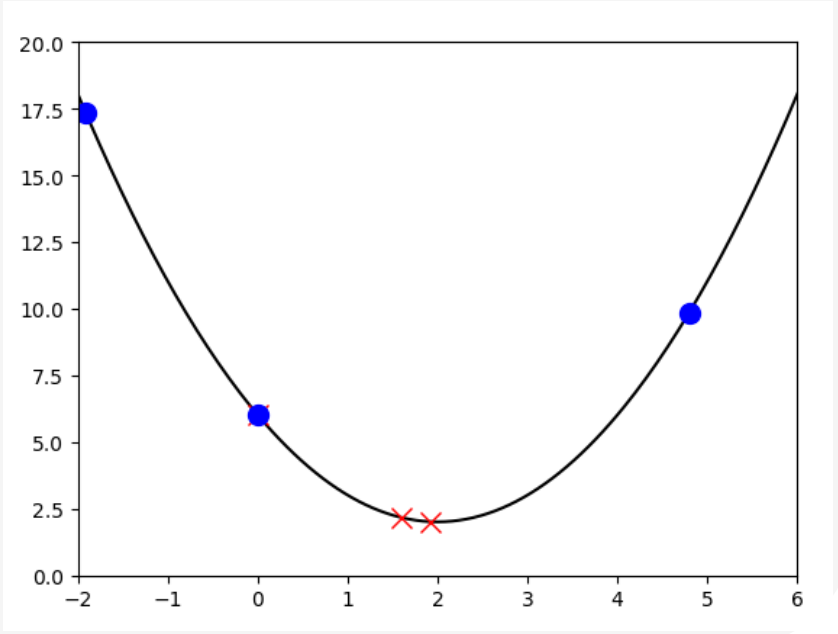
```
28  plt.plot(xx,f1(xx),'k')
29  mu1 = 0.4
30  mu2 = 1.2
31  x1=0
32  x2=0
33  plt.plot(x1,f1(x1),'x',markersize=10, color='r')
34  plt.plot(x2,f1(x2),'go',markersize=10, color='b')
35  x1=x1-mu1*f1d(x1)
36  x2=x2-mu2*f1d(x2)
37  plt.plot(x1,f1(x1),'x',markersize=10, color='r')
38  plt.plot(x2,f1(x2),'go',markersize=10, color='b')
39
40  plt.xlim(-2,6)
41  plt.ylim(0,20)
42  plt.show()
```



03 Code로 적용하기

GD로 1변수함수 최저점 찾기
But, if mu is too large?

```
44  plt.plot(xx,f1(xx),'k')
45  mu1 = 0.4
46  mu2 = 1.2
47  x1=0
48  x2=0
49  plt.plot(x1,f1(x1),'x',markersize=10, color='r')
50  plt.plot(x2,f1(x2),'go',markersize=10, color='b')
51  x1=x1-mu1*f1d(x1)
52  x2=x2-mu2*f1d(x2)
53  plt.plot(x1,f1(x1),'x',markersize=10, color='r')
54  plt.plot(x2,f1(x2),'go',markersize=10, color='b')
55  x1=x1-mu1*f1d(x1)
56  x2=x2-mu2*f1d(x2)
57  plt.plot(x1,f1(x1),'x',markersize=10, color='r')
58  plt.plot(x2,f1(x2),'go',markersize=10, color='b')
59
60  plt.xlim(-2,6)
61  plt.ylim(0,20)
62  plt.show()
```



03 Code로 적용하기

GD로 1변수함수 최저점 찾기
If, while문 이용

```
64  mu = 0.4
65  x = 0
66
67  temp = x-mu*f1d(x)
68  if(f1(temp)>f1(x)):
69      print("mu is too large!")
70  else:
71      while(True):
72          temp = x-mu*f1d(x)
73          if((temp-x)<0.005):
74              break
75          x = temp
76
77  print(x,f1(x))
```

```
C:₩Users₩JeongWoo₩Desktop₩학교₩Growth Hackers₩기계학습세션준비>onevarfunc.py
1.9968 2.00001024
```

03 Code로 적용하기

GD로 1변수함수 최저점 찾기
If, while문 이용

```
64  mu = 1.2
65  x = 0
66
67  temp = x-mu*f1d(x)
68  if(f1(temp)>f1(x)):
69      print("mu is too large!")
70  else:
71      while(True):
72          temp = x-mu*f1d(x)
73          if((temp-x)<0.005):
74              break
75          x = temp
76
77  print(x,f1(x))
```

```
C:₩Users₩JeongWoo₩Desktop₩학교₩Growth Hackers₩기계학습세션준비>onevarfunc.py
mu is too large!
0 6
```

03 Code로 적용하기

GD로 1차원함수 최저점 찾기
Scipy 이용하기

```
80  result = op.minimize(f1,1)
81  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])
```

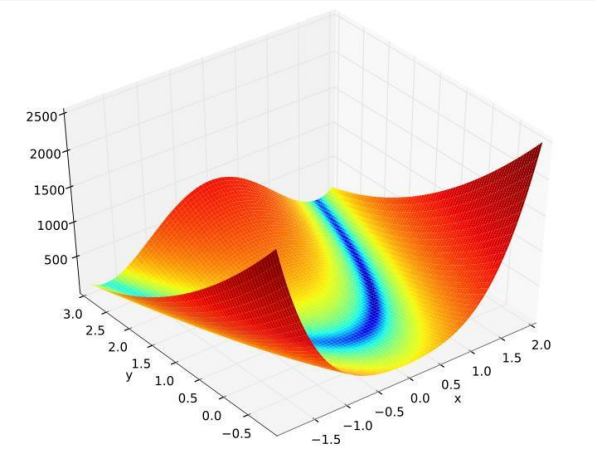
Scipy 이용하여 최저점 찾기

scipy.optimize.minimize

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

03 Code로 적용하기

2차원 함수 그래프로 그리기



2차원 Rosenbrock함수 정의

로젠브록 함수(Rosenbrock function)
수학적 최적화에서 최적화 알고리즘을
시험해볼 용도로 사용하는 비볼록함수

$$f(x,y) = (1-x)^2 + 100(y-x^2)^2$$

03 Code로 적용하기

2변수 함수 그래프로 그리기

```
1  from matplotlib import pyplot as plt
2  import numpy as np
3  from scipy import optimize as op
4
5  def f2(x,y):
6      return (1-x)**2 + 100*(y-x**2)**2
7
8  xx = np.linspace(-3,3,101)
9  yy = np.linspace(-3,3,101)
10 X,Y = np.meshgrid(xx,yy)
11 Z=f2(X,Y)
12
13 plt.contour(X,Y,Z, color='gray',
14     levels=[0.7,3,5,15,50,150,500,1500,5000])
15 plt.show()
```

2변수 함수 정의

f(x , y) = $(1-x)^2+100(y-x^2)^2$

2변수 그래프 그리기

그려진 surface에서 같은 값들을 연결한 등고선 표현

np.meshgrid

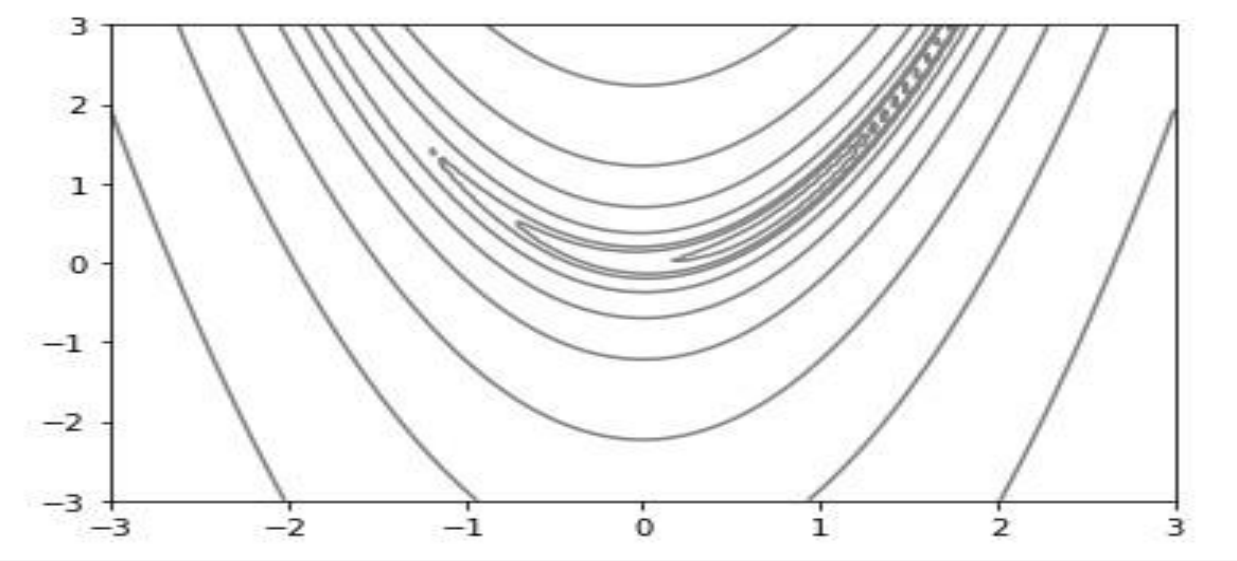
2차원 함수 그리기 편하도록

https://datascienceschool.net/view-notebook/17608f897087478bbeac096438c716f6/

참고

03 Code로 적용하기

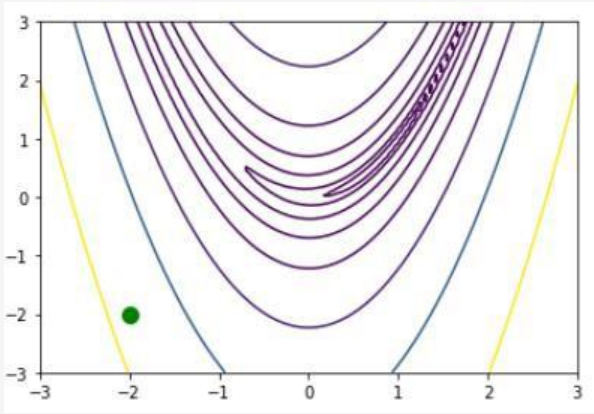
2변수 함수 그래프로 그리기



03 Code로 적용하기

GD로 2변수함수 최저점 찾기

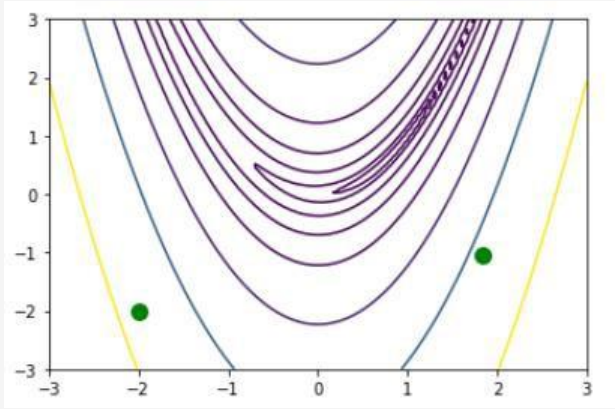
```
17  def f2d(x,y):
18      return np.array([2*x-2-400*x*(y-x**2), 200*(y-x**2)])
19  plt.contour(X,Y,Z, color='gray',
20      levels=[0.7,3,15,50,150,500,1500,5000])
21  mu = 8e-04
22  x = -2
23  y = -2
24  g = f2d(x,y)
25  print(g)
26
27  plt.plot(x, y, 'go', markersize=10)
28
29  plt.show()
```



03 Code로 적용하기

GD로 2변수함수 최저점 찾기

```
plt.contour(X,Y,Z, color='gray',
    levels=[0.7,3,15,50,150,500,1500,5000])
mu = 8e-04
x = -2
y = -2
g = f2d(x,y)
print(g)

plt.plot(x, y, 'go', markersize=10)

x = x-mu*g[0]
y = y-mu*g[1]

g = f2d(x,y)

plt.plot(x,y,'go',markersize=10)

plt.show()
```



03 Code로 적용하기

GD로 2변수함수 최저점 찾기
–If, While 문 이용

```
58  mu = 8e-04
59  x = -2
60  y = -2
61  g = f2d(x,y)
62
63  temp=(x-mu*g[0], y-mu*g[1])
64
65  if(f2(temp[0],temp[1])>f2(x,y)):
66      print("mu is too large!")
67  else:
68      while(True):
69          g = f2d(x,y)
70          temp=(x-mu*g[0], y-mu*g[1])
71          if(f2(x,y)-f2(temp[0],temp[1])<0.0000000005):
72              break
73          x = temp[0]
74          y = temp[1]
75
76  print(x,y,f2(x,y))
77
```

```
C:₩Users₩JeongWoo₩Desktop₩학교₩Growth Hackers₩기계학습세션준비>twovarfunc.py
0.999116848075 0.998230940211 7.81207578587e-07
```

03 Code로 적용하기

Scipy 이용하여 최저점 찾기

```
1  from scipy import optimize as op
2
3  def f2(x):
4      return (1-x[0])**2 + 100*(x[1]-x[0]**2)**2
5
6  result = op.minimize(f2,(2,2))
7  print(result)
```

op.minimize(f2,(2,2))
: minimize 명령으로 다변수 함수를 최적화하는 경우에는 목적 함수가 벡터 인수를 가져야한다.

scipy.optimize.minimize

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

```
C:₩Users₩JeongWoo₩Desktop₩학교₩Growth Hackers₩기계학습세션준비>twovarscipy.py
      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

'Tipping.csv' 데이터에서 SEX와 TOTBILL변수를 통해 Tip rate를 예측할 수 있는 모형을 만들고 그래프를 그려보세요. (Scipy.optimize.minimize를 이용)

THANK YOU