SESSION # 07

By Team 4
@MJ, Jay, Winnie, Elly
Date _ 2017.08.08

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 현상인데, 사람이 직접 모든 일을 알려주는 것으로 이런 목표를 이루기가 힘들다는 한계가 있음

기계학습(machine learning)이란?

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

01 모델링과 기계학습

기계학습(machine learning)이란?

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

01 모델링과 기계학습

(참고)딥러닝이란?

- 머신러닝의 일종
- 인간 두뇌에 있는 신경세포들의 연결방식에서 영감을 얻은 것
- 개별적인 특징 자체를 식별할 수 있다는 것이 장점

(다른 형태의 머신러닝은 미리 정의된 특징을 분석해서 예측의 기반으로 삼음)

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 사이의 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

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

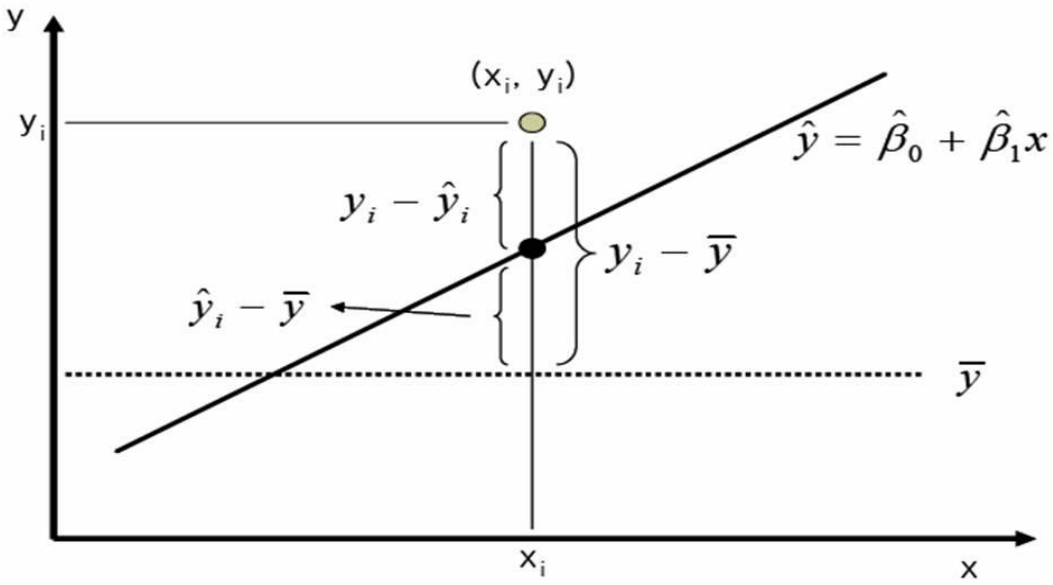
$$y_i - \bar{y} = (y_i - \hat{y_i}) + (\hat{y_i} - \bar{y})$$

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

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

*SST (Total sum of squares)

*SSR (Regression sum of squares)



01 모델링과 기계학습

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

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

01 모델링과 기계학습

Gradient Descent

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

Update rules:

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

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

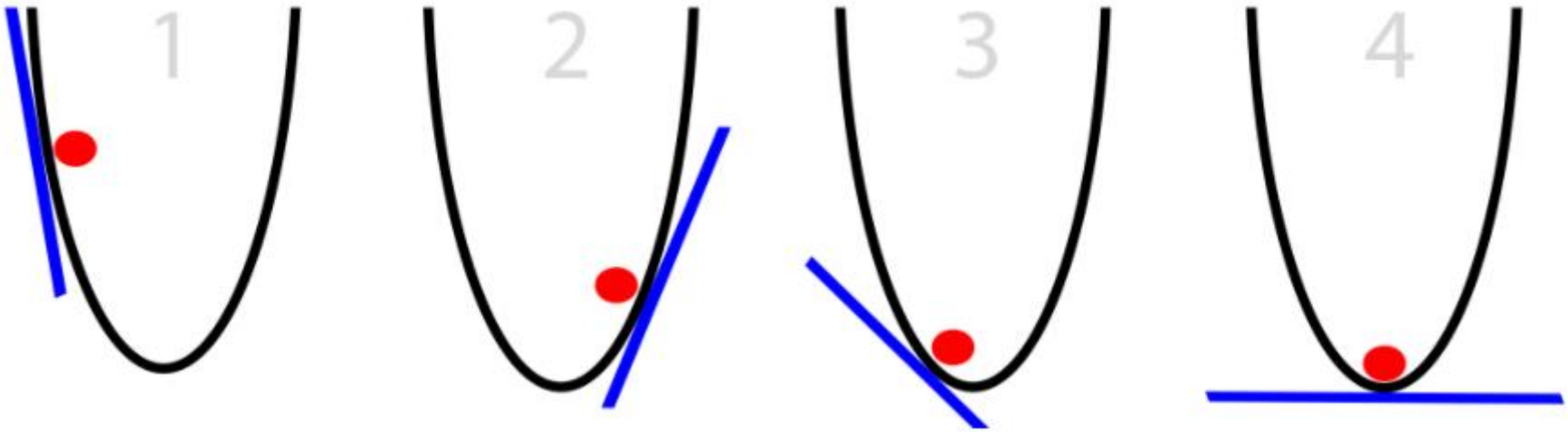
Derivatives:

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

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

01 모델링과 기계학습

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

02 오버피팅과 언더피팅

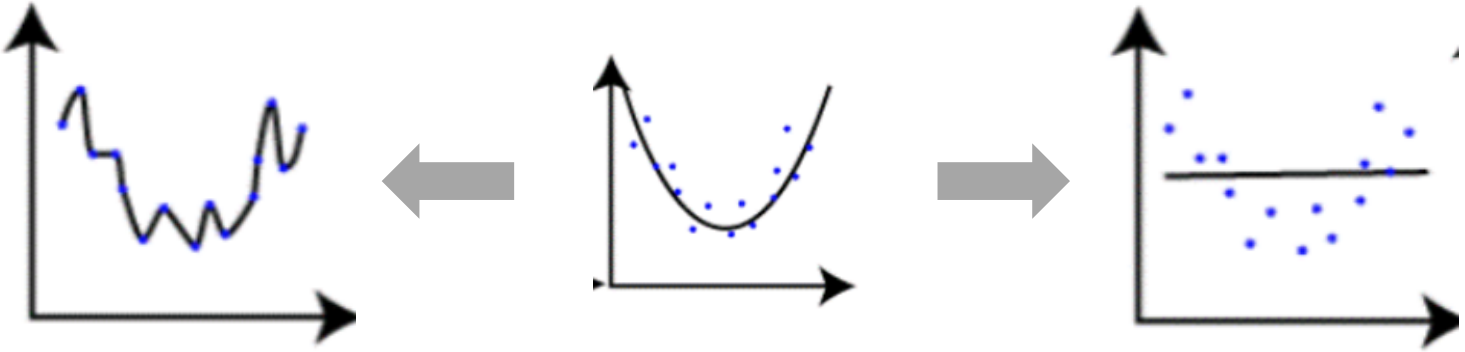
Overfitting and Underfitting

Overfitting

만들어진 모델의 성능이 학습 데이터에서는 좋지만, 기존에 관측한 적이 없는 새로운 데이터에서는 좋지 않은 경우

Underfitting

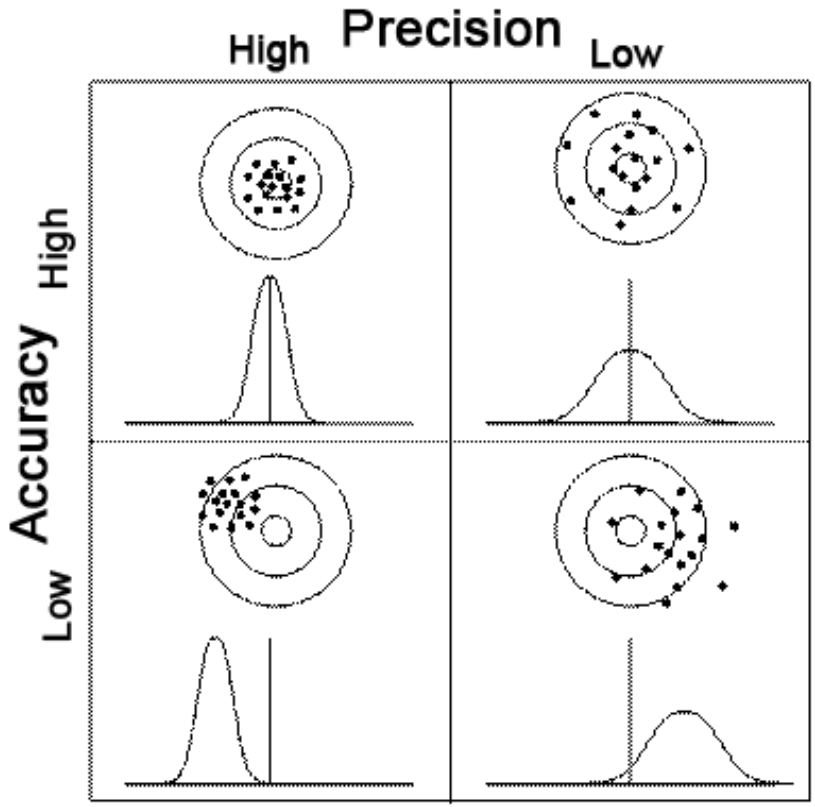
모델의 성능이 학습 데이터에서도 좋지 않은 경우



How to? 전체 데이터의 2/3로 모델을 학습시키고, 나머지 1/3로 모델의 성능을 평가

03 정확도와 정밀도

정확도(Accuracy)와 정밀도(Precision)



정확도(Accuracy)

추정량이 *참값에* 근접한 정도
편향(Bias)이 작을수록 정확도는 높다
편향이란 추정량의 기대값과 모수 참값의 차이
c.f. Test set의 bias가 작다고 항상 정확한 모델인 것은 아니다

정밀도(Precision)

추정량이 *서로* 근접한 정도
분산(Variance)이 작을수록 정밀도는 높다

03 정확도와 정밀도

검사의 오류(Prosecutor's Fallacy)

어느 범죄 현장에서 범인의 것으로 추정되는 담배꽁초 이외의 어떤 단서도 발견되지 않았다.
주변 지역의 전과자들을 대상으로 DNA 검사를 해보니 용의자가 한 명 나왔다. 이 용의자는 범죄가 발생한 시점에 홀로 집에 있었다고 주장하지만 알리바이를 입증해 줄 증인은 찾지 못했다.

검사는 다음과 같은 논리를 가지고 이 용의자를 범인이라고 기소했다.

1.범인이 아닌 경우에 DNA가 일치할 확률은 100만분의 1이다.
2.범죄 현장의 담배꽁초에서 발견된 DNA가 용의자의 DNA가 아닐 확률은 100만분의 1이다.
3.그러므로 용의자가 범인이 아닐 확률은 0.0001%다. 따라서 범인일 확률은 99.9999%이다.

검사의 논리에 대하여 변호사는 다음과 같이 반박했다.

1.우리나라에는 대략 5천만 명의 인구가 있다.
2.DNA가 일치할 확률이 100만분의 1이므로 범인의 DNA와 일치하는 용의자는 약 50명이다.
3.검사가 기소한 용의자는 50명 중의 1명이므로 범인일 확률은 2%이고 아닐 확률은 98%이다.

03 정확도와 정밀도

검사의 오류(Prosecutor's Fallacy)

	실제로 용의자가 범인	실제로 용의자가 범인이 아님(A)	합계
용의자의 DNA가 범인의 것과 불일치	0	49,999,950	49,999,950
용의자의 DNA가 범인의 것과 일치(B)	1	**49**	50
합계	1	49,999,999	50,000,000

A: 용의자가 범인이 아닌 사건

B: 용의자의 DNA가 범인의 것과 일치하는 사건

용의자의 DNA가 범인의 것과 일치할 때 용의자가 범인이 아닐 확률? → $P(A|B)$ → 알고싶은 것

용의자가 범인이 아닐 때 용의자의 DNA가 범인의 것과 일치할 확률? → $P(B|A)$ → 검사의 주장

$$P(B|A) = \frac{1}{1{,}000{,}000}$$

c.f. 통계적으로 p-value

$$P(A|B) = P(B|A) \cdot \frac{P(A)}{P(B)}$$

$$= \frac{1}{1{,}000{,}000} \cdot 980{,}392 \approx 0.98$$

검사의 오류: P(A)/P(B)를 1로 가정, P(B|A) = P(A|B)라고 결론 짓는 오류

03 정확도와 정밀도

혼동행렬(Confusion Matrix)

	실제로 테러리스트	실제로 테러리스트가 아님	합계
블랙리스트에 있음	10 True Positive	99,990 False Positive	100,000
블랙리스트에 없음	9,990 False Negative	199,890,010 True Negative	199,900,000
합계	10,000	199,990,000	200,000,000

정확도: 모델이 정확하게 양성 또는 음성으로 예측한 비율

a = (TP + TN) / Total

정밀도: 모델이 양성으로 예측한 것 중 실제 양성인 비율

p = TP / (TP + FP) c.f. 통계적으로 '검정력'이라고 함

재현율(Recall): 실제 양성 중 모델이 정확하게 양성으로 예측한 비율

r = TP / (TP + FN)

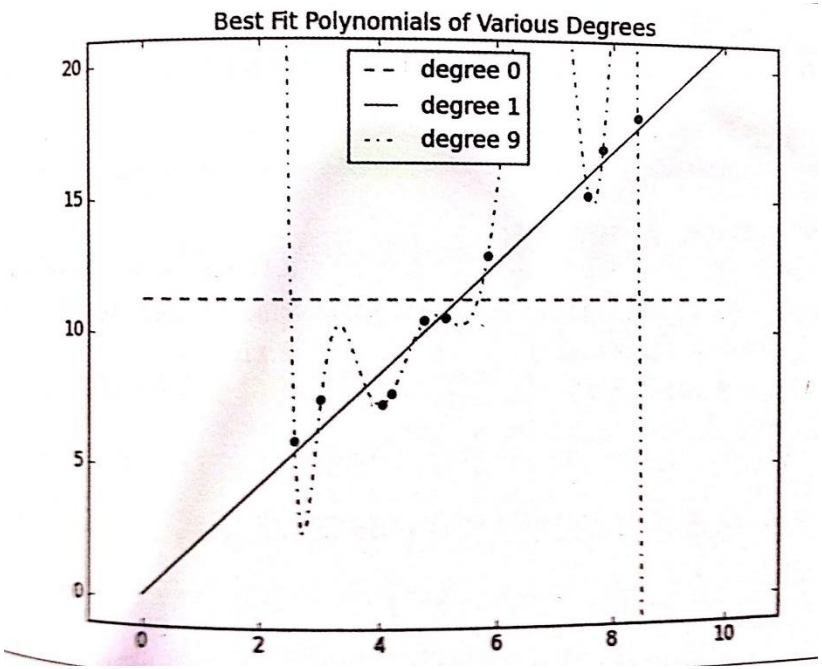
F1 점수: 정밀도와 재현율의 조화평균, 항상 정밀도와 재현율 사이의 값을 가짐

F1 = 2*p*r / (p + r)

a = 0.99945
p = 0.0001
r = 0.001
F1 = 0.0001818

04 Bias – Variance 트레이드오프

Bias vs Variance



밑바닥 책 graph 참조

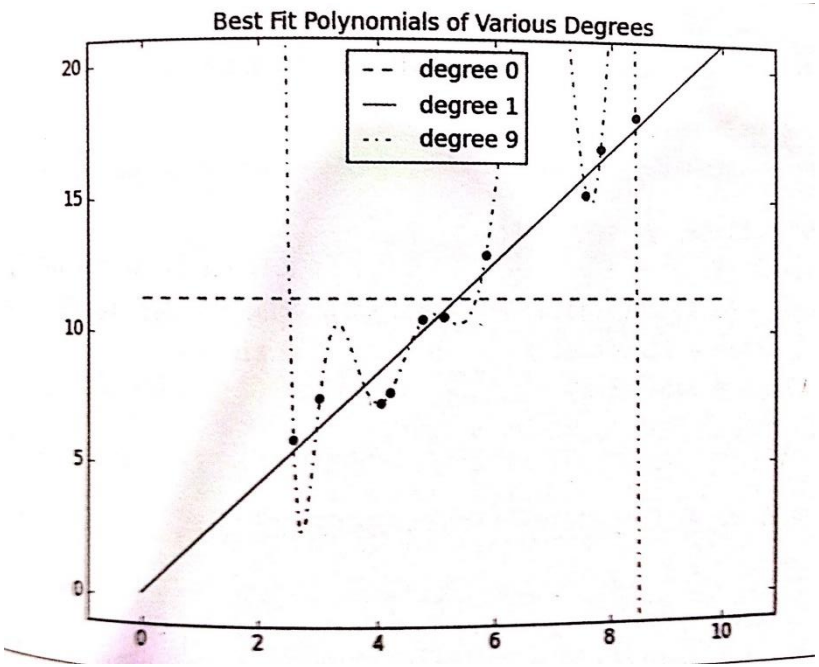
상수함수: 항상 같은 값 반환
그 값으로 치우쳐 있다!
Bias 높음, Variance 낮음 (Underfit)

9차 함수: 모든 train 데이터에 fit
그러나 꼬불꼬불~ 옳지 않음
Bias 낮음, Variance 높음 (Overfit)

04 Bias – Variance 트레이드오프

Bias vs Variance

따라서 Bias와 Variance는 한 극단의 값을 갖는것이 아니라
중도에서 균형을 이루어야!!



밑바닥 책 graph 참조

Bias가 높을 경우: (상수함수)

데이터(training)을
아무리 많이 추가해도 계속 오류!
Ex) 집값 예측에서
전 주인 머리카락 갯수를 변수로 할때

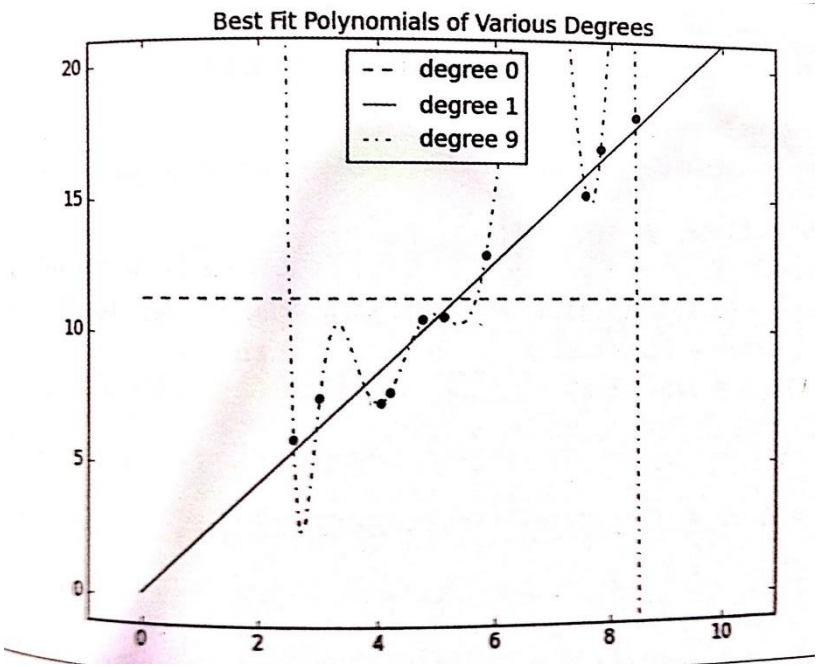
새로운 특성(feature)을 추가하기…
예를 들면, 집의 넓이나 위치

상수함수 Y=C 에서
일차함수Y=aX+b

04 Bias – Variance 트레이드오프

Bias vs Variance

따라서 Bias와 Variance는 한 극단의 값을 갖는것이 아니라
중도에서 균형을 이루어야!!



밑바닥 책 graph 참조

Variance가 높을 경우: (9차함수)

그렇다고 '특성(feature)'을 너무 많이 사용?

집 넓이, 위치, 살았던 사람수,
연식, 방 갯수, 중앙난방 여부, 리모델링….

오버피팅!

쓸데 없는 변수 줄이기
트레이닝 더 많이 하기…

모든 변수가 쓸모 있다고 생각할때?
Regularization!

04 Bias – Variance 트레이드오프

Regularization

$$h(x)\text{: 집값} = \theta_1 \times (\text{집 넓이}) + \theta_2 \times (\text{방 갯수}) + \ \ldots$$

또는

$$h(x)\text{: 집값} = \theta_1 \times (\text{집 넓이}) + \theta_2 \times (\text{집 넓이})^2 + \cdots$$

Cost Function

$$J(\theta) = \frac{1}{2m}\sum (h(x) - y)^2$$

$$J(\theta) = \frac{1}{2m}\sum (h(x) - y)^2 + \lambda \sum \theta^2$$

쓸데없는 Theta들은 J를 증가시키기에 자연히 줄어듦

04 Bias – Variance 트레이드오프

MATLAB을 통해 보기

집값과 집 넓이 사이의 상관관계!

1. 데이터 시각화, 추세선 예측해보기

2. 1차 함수 그래프로 fitting => underfitting 확인

3. 추론, 변수 추가하기

$$h(x)\text{: 집값} = \theta_1 \times (\text{집 넓이}) + \theta_2 \times (\text{집 넓이})^2 + \cdots$$

4. 다차 함수 그래프로 fitting => overfitting 확인

5. Regularization

04 Bias – Variance 트레이드오프

~~MATLAB을 통해 보기~~ → "PYTHON!"

집값과 집 넓이 사이의 상관관계!

1. 데이터 시각화, 추세선 예측해보기

2. 1차 함수 그래프로 fitting => underfitting 확인

3. 추론, 변수 추가하기

$$h(x)\text{: 집값} = \theta_1 \times (\text{집 넓이}) + \theta_2 \times (\text{집 넓이})^2 + \cdots$$

4. 다차 함수 그래프로 fitting => overfitting 확인

5. Regularization

05 PYTHON Practice for Cost Function

비용함수 Cost Function 이란?

*"**비용**에 관련된 모든 변량에 대하여 어떤 관계를 나타내는 함수.*
*즉, **최적화**를 위해 사용되는 복잡한 조건의 스칼라 측정을 말한다."*

$$\text{모수} \xrightarrow{f} \text{예측 오차}$$

타겟 값과 가장 유사한 값을 출력하는 예측 오차를
최소화 하는 모형을 찾는 것이 목표

05 PYTHON Practice for Cost Function

GD로 최저점 찾기

GD(Gradient Descent)

- 최적화 알고리즘 중 하나로, 함수의 최저점을 찾는데 사용됨
- 현재 위치에서 기울기 값 $g(x_k)$을 이용하여 다음에 시도할 위치를 알아내는 방법

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

- 목표 : $\min_{\theta_0,\theta_1} J(\theta_0,\theta_1)$

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

05 PYTHON Practice for Cost Function

GD로 최저점 찾기

GD(Gradient Descent)

- 만약 현재 위치 x_k에서 기울기가 음수라면

  $g(x_k) < 0$ ▶ 앞으로 진행

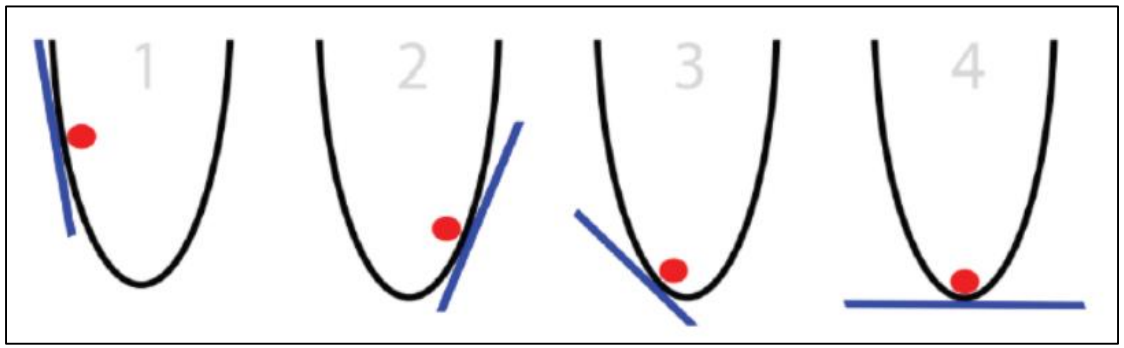
- 만약 현재 위치 x_k에서 기울기가 양수라면

  $g(x_k) > 0$ ▶ 뒤로 진행

  - 최적점에 도달하였을 때

  $g(x_k) = 0$ ▶ 더 이상 이동하지 않음

- $\alpha = \mu$ = Step size
- 최적화를 하는 사람이 임의로 결정
- 너무 작으면 -> 시간이 오래 걸림
- 너무 크면 -> 최저점을 찾지 못할 수 있음



01 모델링과 기계학습

Gradient Descent

$$h_\theta(x) = \theta_0 + \theta_1 x$$

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

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:

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

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

05 PYTHON Practice for Cost Function

1차원 함수 그래프로 그리기

05 PYTHON Practice for Cost Function

1차원 함수 그래프로 그리기

1) Matplotlib 라이브러리 불러오기

- Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms
  - matlab과 비슷한 인터페이스를 가지는 라이브러리
  - Pyplot이라는 sub-package를 통해서 그래프 그림

05 PYTHON Practice for Cost Function

1차원 함수 그래프로 그리기

1) Matplotlib 라이브러리 불러오기

- Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms
  - matlab과 비슷한 인터페이스를 가지는 라이브러리
  - Pyplot이라는 sub-package를 통해서 그래프 그림

05 PYTHON Practice for Cost Function

GD로 1차원 함수 최저점 찾기

2) 1차원 함수 정의

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

05 PYTHON Practice for Cost Function

GD로 1차원 함수 최저점 찾기

2) X값 할당

- 임의의 'xx' 변수에 x값 할당
- X에 따른 y의 변화를 알아보기 위해서 x에 입력 값의 범위를 설정해야 함

05 PYTHON Practice for Cost Function

GD로 1차원 함수 최저점 찾기

```
Out[374]: array([-4.        , -3.91919192, -3.83838384, -3.75757576, -3.67676768,
                 -3.5959596 , -3.51515152, -3.43434343, -3.35353535, -3.27272727,
                 -3.19191919, -3.11111111, -3.03030303, -2.94949495, -2.86868687,
                 -2.78787879, -2.70707071, -2.62626263, -2.54545455, -2.46464646,
                 -2.38383838, -2.3030303 , -2.22222222, -2.14141414, -2.06060606,
                 -1.97979798, -1.8989899 , -1.81818182, -1.73737374, -1.65656566,
                 -1.57575758, -1.49494949, -1.41414141, -1.33333333, -1.25252525,
                 -1.17171717, -1.09090909, -1.01010101, -0.92929293, -0.84848485,
                 -0.76767677, -0.68686869, -0.60606061, -0.52525253, -0.44444444,
                 -0.36363636, -0.28282828, -0.2020202 , -0.12121212, -0.04040404,
                  0.04040404,  0.12121212,  0.2020202 ,  0.28282828,  0.36363636,
                  0.44444444,  0.52525253,  0.60606061,  0.68686869,  0.76767677,
                  0.84848485,  0.92929293,  1.01010101,  1.09090909,  1.17171717,
                  1.25252525,  1.33333333,  1.41414141,  1.49494949,  1.57575758,
                  1.65656566,  1.73737374,  1.81818182,  1.8989899 ,  1.97979798,
                  2.06060606,  2.14141414,  2.22222222,  2.3030303 ,  2.38383838,
                  2.46464646,  2.54545455,  2.62626263,  2.70707071,  2.78787879,
                  2.86868687,  2.94949495,  3.03030303,  3.11111111,  3.19191919,
                  3.27272727,  3.35353535,  3.43434343,  3.51515152,  3.5959596 ,
                  3.67676768,  3.75757576,  3.83838384,  3.91919192,  4.        ])
```

05 PYTHON Practice for Cost Function

GD로 1차원 함수 최저점 찾기

3) 그래프 그리기

- x값에 따른 y의 변화를 나타내는 f1(x) 그래프 그리기

- **plt.plot()**
  - 그래프 그리기
- **plt.xlim()**
  - x범위 지정
- **plt.ylim()**
  - y범위 지정
- **plt.show()**
  - 그래프 출력

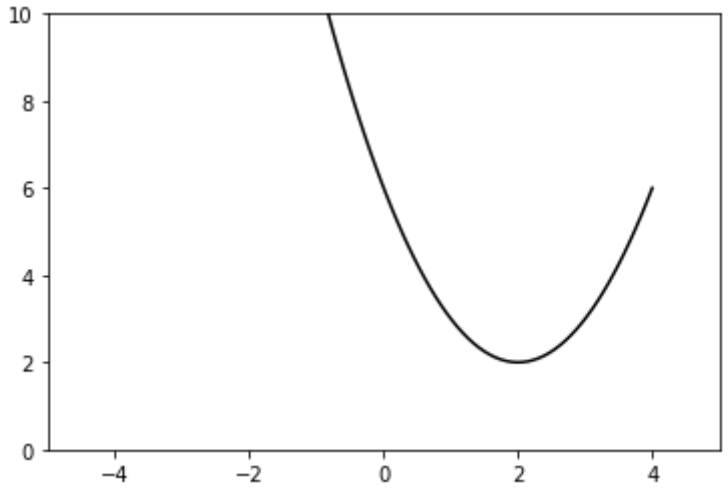
05 PYTHON Practice for Cost Function

GD로 1차원 함수 최저점 찾기

3) 그래프 그리기

- x값에 따른 y의 변화를 나타내는 f1(x) 그래프 그리기

- **plt.plot()**
  - 그래프 그리기
- **plt.xlim()**
  - x범위 지정
- **plt.ylim()**
  - y범위 지정
- **plt.show()**
  - 그래프 출력



05 PYTHON Practice for Cost Function

GD로 1차원 함수 최저점 찾기

4) 도함수 정의

- f(x) = (x-2)^2 + 2 의 도함수 정의하기

05 PYTHON Practice for Cost Function

GD로 1차원 함수 최저점 찾기

5) 도함수 그래프 그리기

f1(x)

---

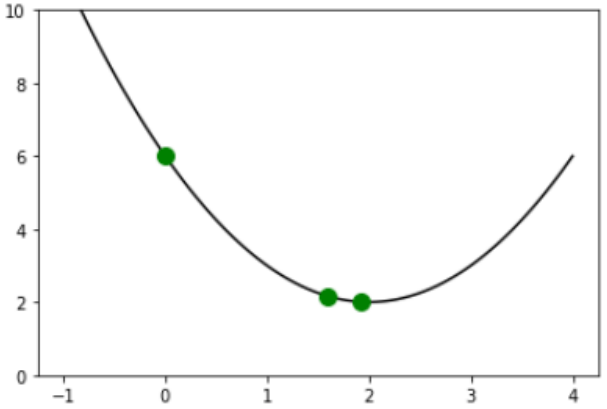
mu

X = 0

X = x - mu * f1d(x)

X = x - mu * f1d(x)

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




```
x=0, g=-4.0
x = 1.6, g = -0.7999999999999998
x = 1.92, g = -0.16000000000000014
```

05 PYTHON Practice for Cost Function

GD로 1차원 함수 최저점 찾기

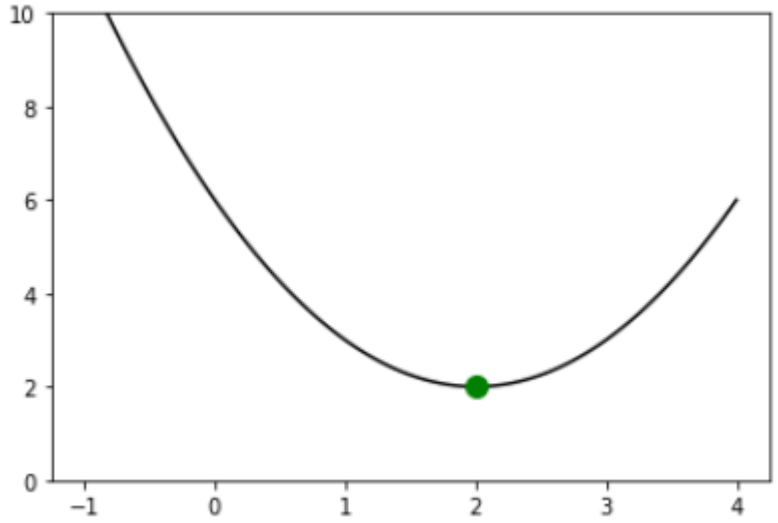
6) Scipy를 이용한 최적화

```
import scipy.optimize as op

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

```
array([ 1.99999999])
```



05 PYTHON Practice for Cost Function

GD로 2차원 함수 최저점 찾기

05 PYTHON Practice for Cost Function

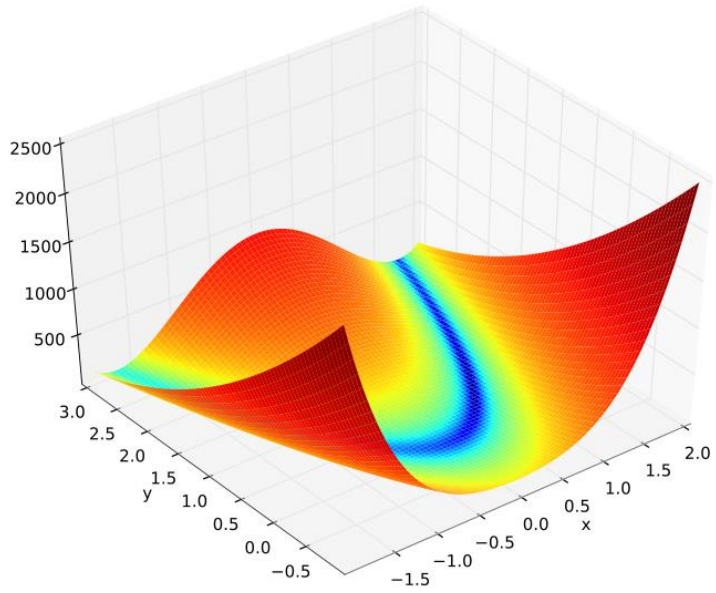
GD로 2차원 함수 최저점 찾기

1) 2차원 함수 정의

- 2차원 Rosenbrock 함수 정의

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

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



05 PYTHON Practice for Cost Function

GD로 2차원 함수 최저점 찾기

1) 2차원 함수 정의

- 2차원으로 나타내기 위해 x와 y값의 범위를 지정해줌

```
xx = np.linspace(-3,3,100)
yy = np.linspace(-3,3,100)
X, Y = np.meshgrid(xx,yy)
```

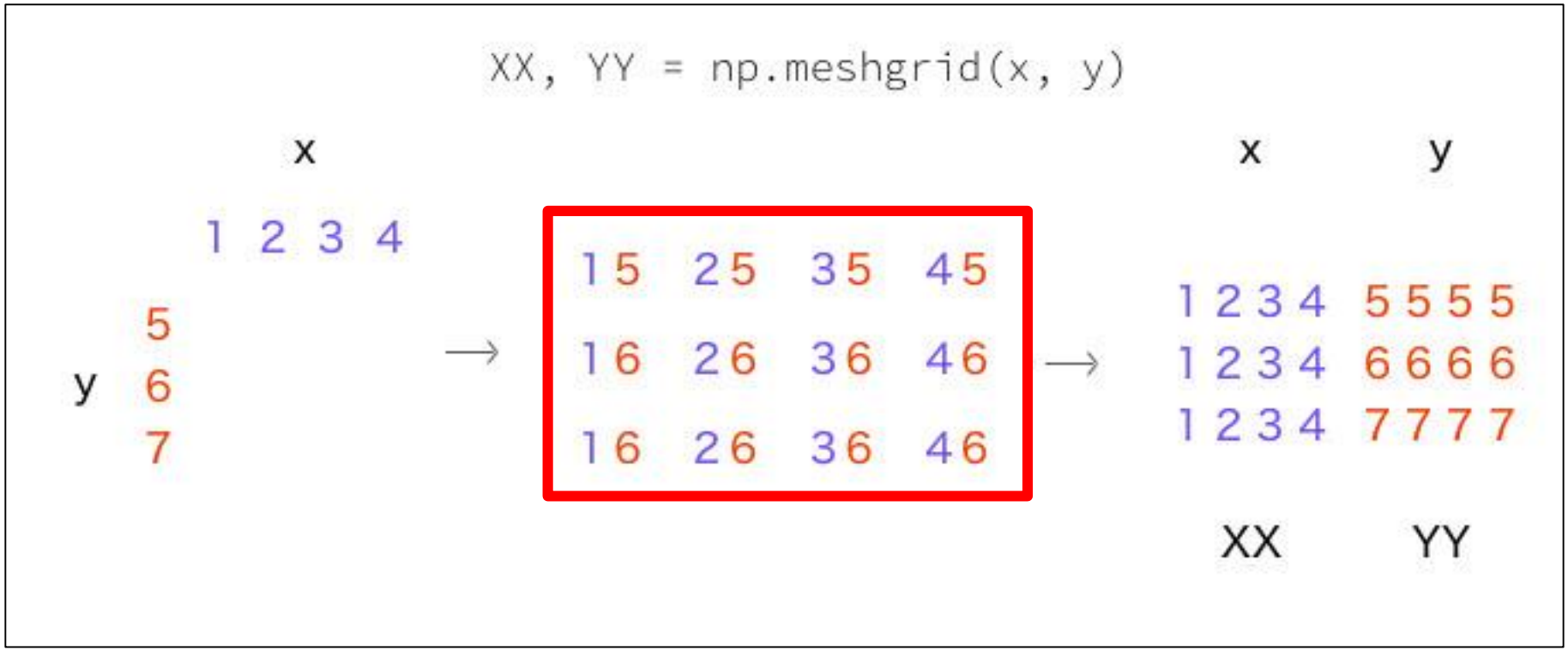
* -3과 3 사이의 100개의 동일한 간격의 수를 추출

* Meshgrid 명령을 통해 행단위와 열단위로 각각 해당 배열을 정방행렬로 선언

05 PYTHON Practice for Cost Function

GD로 2차원 함수 최저점 찾기

1) 2차원 함수 정의



05 PYTHON Practice for Cost Function

GD로 2차원 함수 최저점 찾기

```
xx = np.linspace(-3,3,100)
yy = np.linspace(-3,3,100)
X, Y = np.meshgrid(xx,yy)
```

```
[array([[-3.        , -2.93939394, -2.87878788, ...,  2.87878788,
         2.93939394,  3.        ],
       [-3.        , -2.93939394, -2.87878788, ...,  2.87878788,
         2.93939394,  3.        ],
       [-3.        , -2.93939394, -2.87878788, ...,  2.87878788,
         2.93939394,  3.        ],
       ...,
       [-3.        , -2.93939394, -2.87878788, ...,  2.87878788,
         2.93939394,  3.        ],
       [-3.        , -2.93939394, -2.87878788, ...,  2.87878788,
         2.93939394,  3.        ],
       [-3.        , -2.93939394, -2.87878788, ...,  2.87878788,
         2.93939394,  3.        ]]), array([[-3.        , -3.        , -3.        , ..., -3.        ,
        -3.        , -3.        ],
       [-2.93939394, -2.93939394, -2.93939394, ..., -2.93939394,
        -2.93939394, -2.93939394],
       [-2.87878788, -2.87878788, -2.87878788, ..., -2.87878788,
        -2.87878788, -2.87878788],
       ...,
       [ 2.87878788,  2.87878788,  2.87878788, ...,  2.87878788,
         2.87878788,  2.87878788],
       [ 2.93939394,  2.93939394,  2.93939394, ...,  2.93939394,
         2.93939394,  2.93939394],
       [ 3.        ,  3.        ,  3.        , ...,  3.        ,
         3.        ,  3.        ]])]
```

xx

yy

05 PYTHON Practice for Cost Function

GD로 2차원 함수 최저점 찾기

2) 2차원 그래프 그리기

좌표에서 그려진 surface에서
같은 값들을 연결한 등가선 표현하기

- plt.contour()
- plt.xlim()
- plt.ylim()
- plt.show()

05 PYTHON Practice for Cost Function

GD로 2차원 함수 최저점 찾기

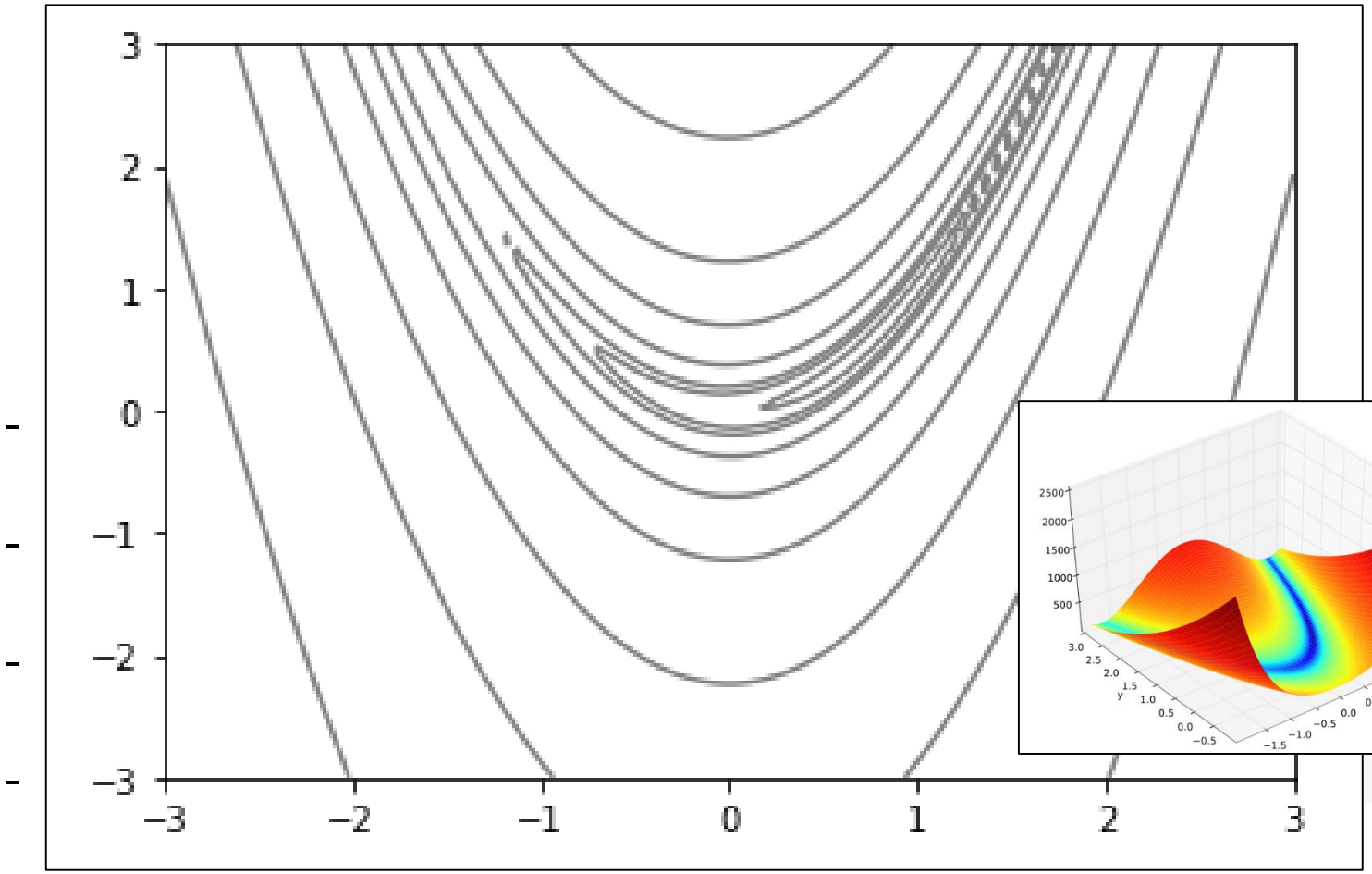
2) 2차원 그래프 그리기

좌표에서 그려진 surface에서
같은 값들을 연결한 등고선 표현하기

- plt.contour(X, Y, Z)
  >> Make a contour plot of an array Z.
  The levels are chose automatically.
  >> X, Y specify the (x,y) coordinates of the surface.
- plt.xlim()
- plt.ylim()
- plt.show()

05 PYTHON Practice for Cost Function

GD로 2차원 함수 최저점 찾기



05 PYTHON Practice for Cost Function

GD로 2차원 함수 최저점 찾기

3) 도함수 정의하기

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

05 PYTHON Practice for Cost Function

GD로 2차원 함수 최저점 찾기

4) 도함수 그래프 그리기

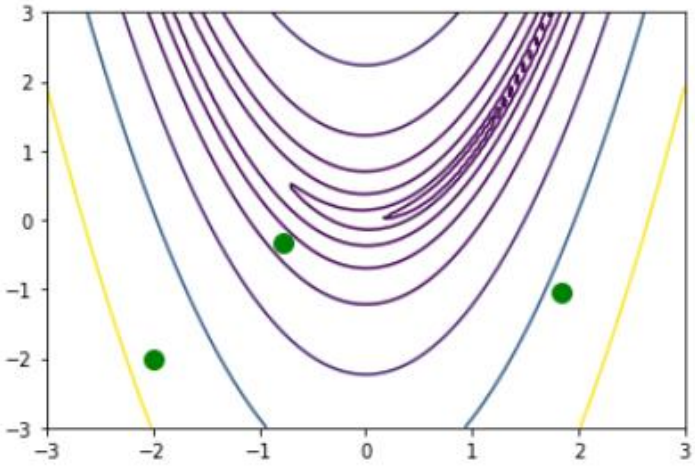
```
f2(x)
```

```
mu
```

```
X = -2
Y = -2
```

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

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



05 PYTHON Practice for Cost Function

GD로 2차원 함수 최저점 찾기

6) Scipy를 이용한 최적화

```
def f2(x):
    return (1 - x[0])**2 + 100.0 * (x[1] - x[0]**2)**2

result = op.minimize(f2, (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])
```

```
def f2(x, y)
    return (1-x)**2 + 100.0 * (y - x**2)**2
```

>> minimize 명령으로 다변수 함수를 최적화하는 경우에는 목적 함수가 벡터 인수를 가져야 한다.

scipy.optimize. **minimize** (*fun, x0, args=(), method=None, jac=None, hess=None, hessp=None, bounds=None, constraints=(), tol=None, callback=None, options=None*)¶ [source]

Minimization of scalar function of one or more variables.

In general, the optimization problems are of the form:

05 PYTHON Practice for Cost Function

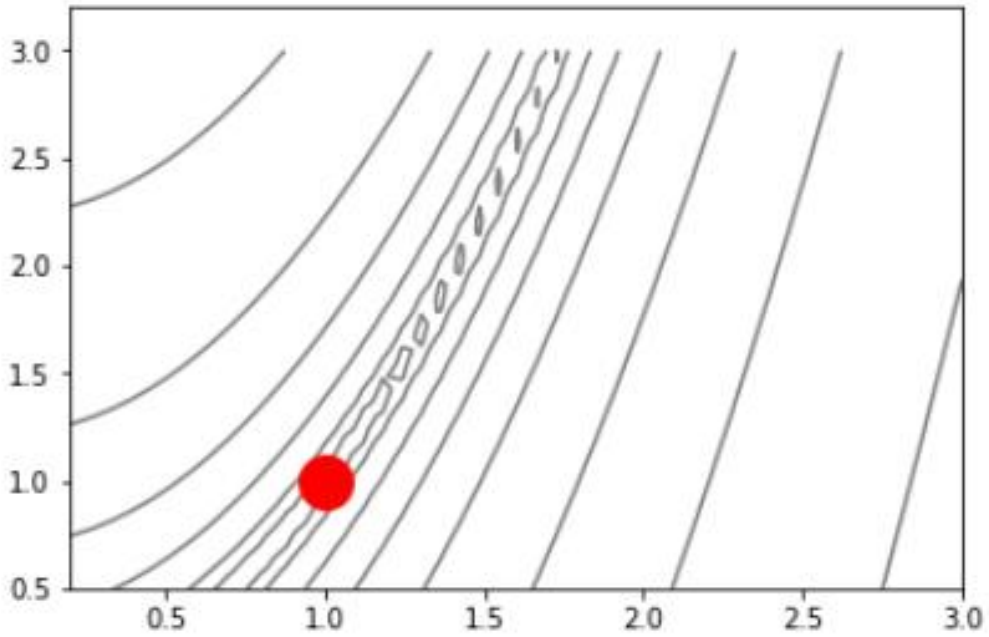
GD로 2차원 함수 최저점 찾기

6) Scipy를 이용한 최적화

```
def f2(x):
    return (1 - x[0])**2 + 100.0 * (x[1] - x[0]**2)**2

result = op.minimize(f2, (2,2))
print(result)
```

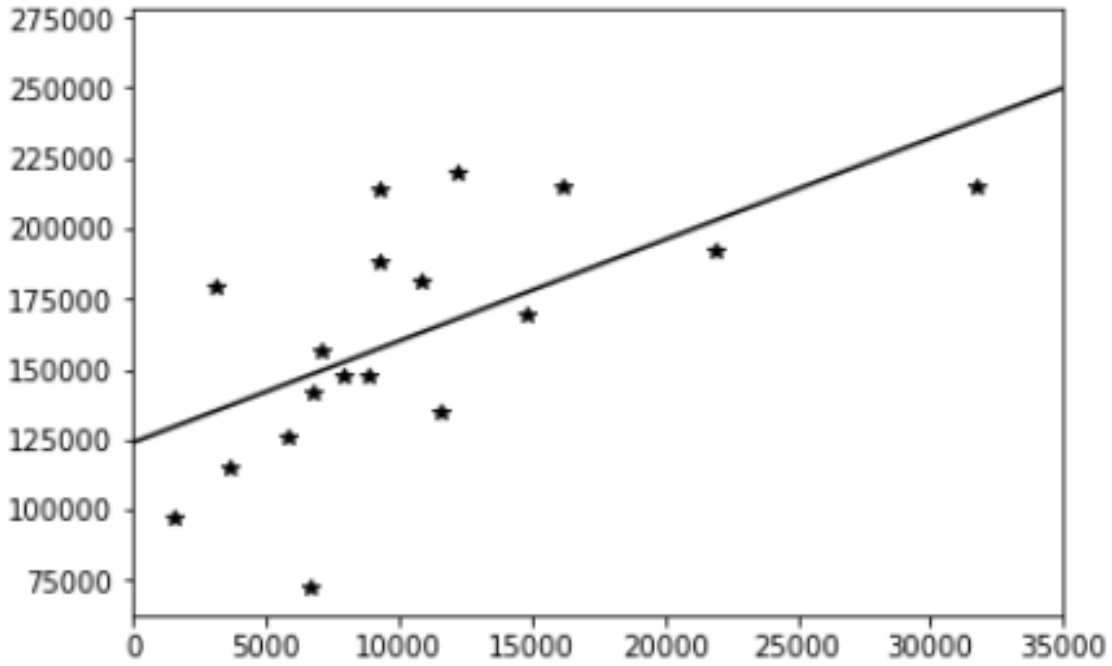
```
      fun: 1.8932893809017893e-11
 hess_inv: array([[ 0.51675994,  1.03186494],
       [ 1.03186494,  2.0655726 ]])
      jac: array([  5.27380711e-06,  -2.50575298e-06])
  message: 'Optimization terminated successfully.'
     nfev: 140
      nit: 30
     njev: 35
   status: 0
  success: True
  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



05 PYTHON Practice for Cost Function

QUEST

'newtrain.csv' 데이터를 기반으로 Scipy.optimize.minimize를 이용하여 집값을 예측할 수 있는 모형을 만들고 그래프를 그려보세요!



참고자료

- http://learning-image.com/220769047794
- https://wikidocs.net/4209
- https://www.coursera.org/learn/machine-learning/supplement/cmjlc/advanced-optimization
- http://www.ciokorea.com/news/35035
- http://blog.naver.com/miroku0820/220695617076
- http://blog.naver.com/2011topcit/220401457200
- http://blog.naver.com/960125_hds/221026064994
- http://enook.jbnu.ac.kr/contents/45/#!/p/39
- [네이버 지식백과] Machine Learning - 기계 학습
- 수식없이 이해하는 지도학습(Supervised Learning), 비지도학습(Unsupervised Learning) 그리고 강화학습(Reinforcement Learning)
- 지형 공간정보체계 용어사전
- https://shapeofdata.wordpress.com/2013/03/26/general-regression-and-over-fitting/

Thank you !