

# Linear Regression

Hyunjoong Kim

[soy.lovit@gmail.com](mailto:soy.lovit@gmail.com)

[github.com/lovit/python\\_ml\\_intro](https://github.com/lovit/python_ml_intro)

# Vector

---

- 데이터를 테이블 형식으로 표현하는 경우가 많습니다.
  - 하나의 데이터 (row, data, point, instance, record, ... ) 는 이를 설명하는 변수 (column, feature, column, variables) 로 구성되어 있습니다.

	datetime	temperature	humidity	wind speed	demand
row 1	2016. 06. 25	33	45	3.5	254
row 2	2016. 05. 21	23	21	4.3	761
row 3	2016. 05. 23	27	57	2.1	340
row 4	...	...	...	...	...

# Vector

- 숫자로 이뤄진 테이블은 그대로 벡터로 표현할 수 있습니다.

	datetime	temperature	humidity	wind speed	demand
row 1	2016. 06. 25	33	45	3.5	254
row 2	2016. 05. 21	23	21	4.3	761
row 3	2016. 05. 23	27	57	2.1	340
row 4	...	...	...	...	...



```
[[ 25, 33, 45, 3.5, 254 ],  
 [ -10, 23, 21, 4.3, 761 ],  
 [ -8, 27, 57, 2.1, 340 ],  
 ...  
]
```

# Vector

- 명목형 변수가 포함된 경우, 이를 dummy 로 표현할 수도 있습니다.

	Type	temperature	humidity	wind speed	demand
row 1	A	33	45	3.5	254
row 2	A	23	21	4.3	761
row 3	B	27	57	2.1	340
row 4	...	...	...	...	...



	Type A	Type B	...	humidity	wind speed	demand	temperature
row 1	1	0	0	45	3.5	254	33
row 2	1	0	0	21	4.3	761	23
row 3	0	1	0	57	2.1	340	27
row 4	...	...	...	...	...	...	...

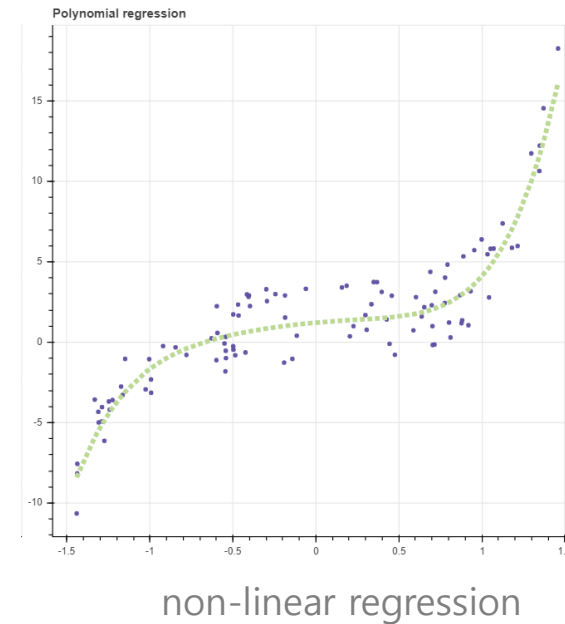
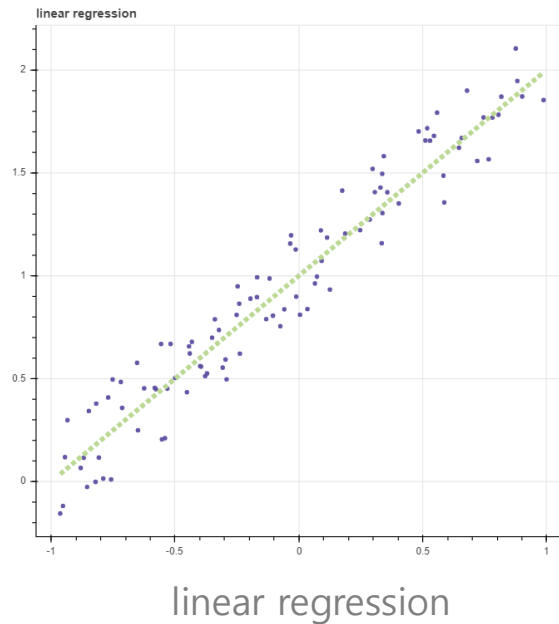
# Regression

- Regression 은 입력변수 (input variables) 를 이용하여 숫자형 출력변수 (numerical output variable) 의 값을 예측 (predict) 하는 문제입니다.

	(input variables)				(output variables)
	datetime	temperature	humidity	wind speed	demand
row 1	25	33	45	3.5	254
row 2	-10	23	21	4.3	761
row 3	-8	27	57	2.1	340
row 4	...	...	...	...	...

# Regression

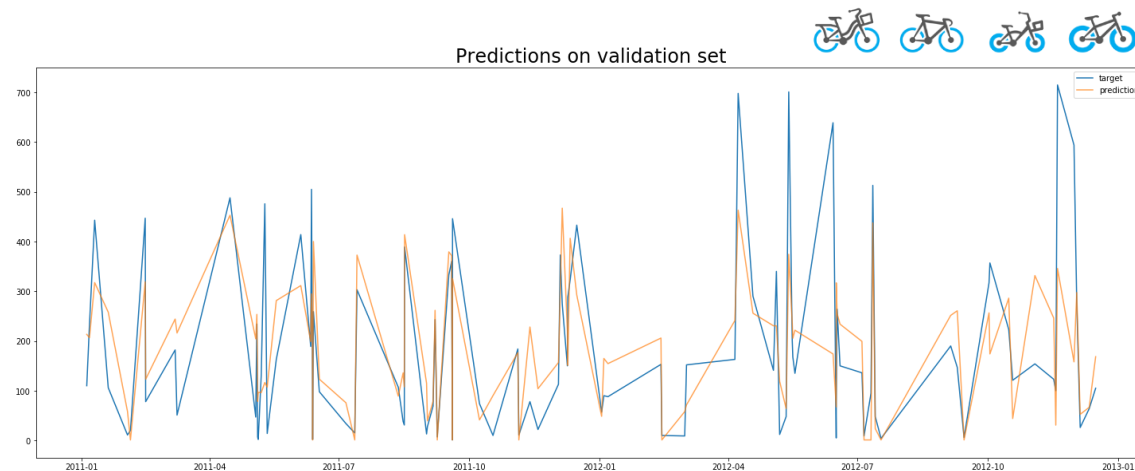
- 추세선의 종류 (직선, 곡선)에 따라 linear / non-linear regression 으로 분류합니다.
- 선형은  $x$  의 범위가 변하여도  $y$  와의 관계식이 일정하다는 가정이 필요합니다.



# Regression

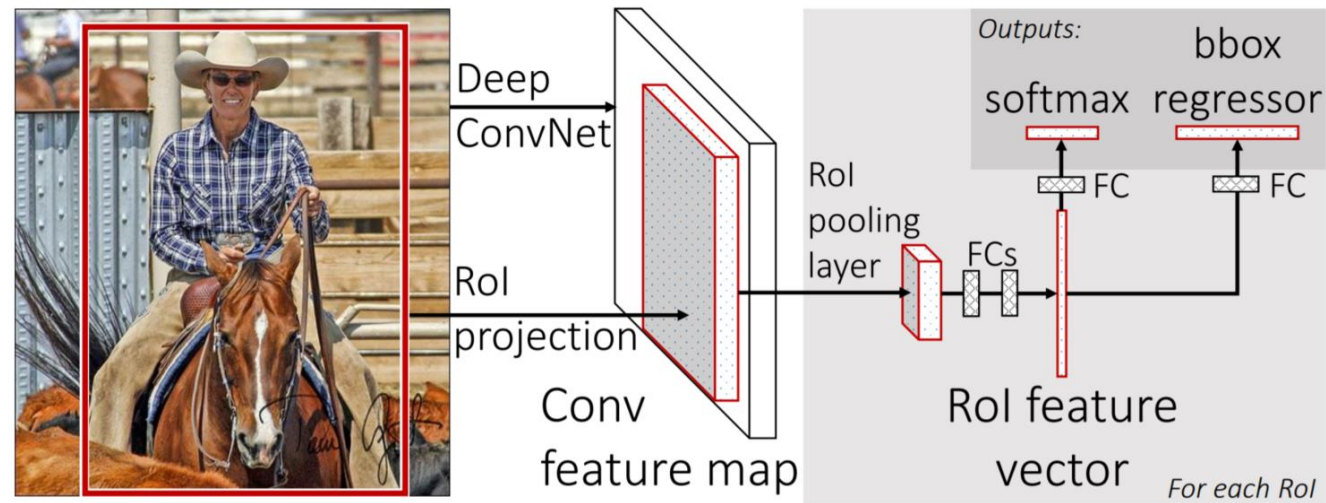
- Regression 은 수요 예측의 모델링에 이용될 수 있습니다.
  - 수요 (output) 와 이를 설명하는 변수 (input) 와의 관계를 탐색합니다.

$\text{count} = f(\text{datetime, season, holiday, workingday, weather, temp, humidity, windspeed, casual, registered})$



# Regression

- 사진 (input) 에서 물체의 위치 (output) 를 탐색하는데도 이용됩니다.
  - 사진 속 물체의 상대적인 위치를 box 로 나타낼 수 있습니다.
  - 설명 변수와의 관계가 선형이 아니기 때문에 복잡한 형태의 모델을 이용합니다.

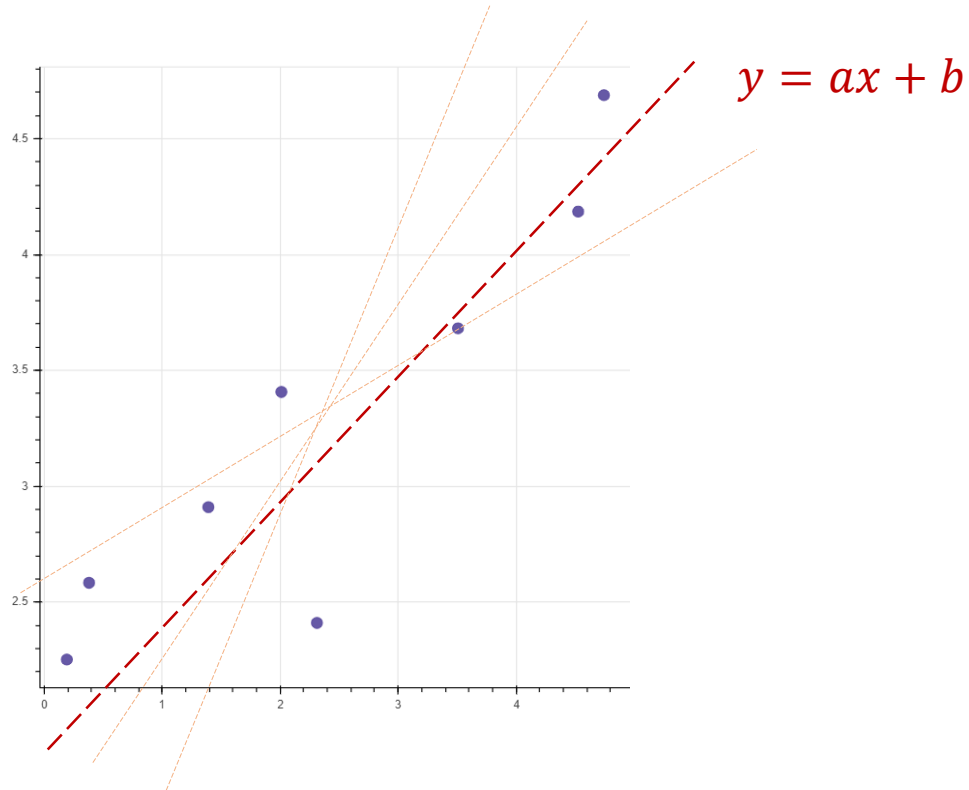


bounding box : (x\_min, x\_max, y\_min, y\_max)



# Univariate Linear Regression

- 두 변수  $(x, y)$  간에는  $y = ax + b$  라는 관계가 있다고 가정합니다.
  - 다양한 직선 중에서 데이터 간의 관계를 **가장 잘 설명**하는 직선을 선택합니다.
  - 설명력의 평가 기준이 필요합니다.



# Univariate Linear Regression

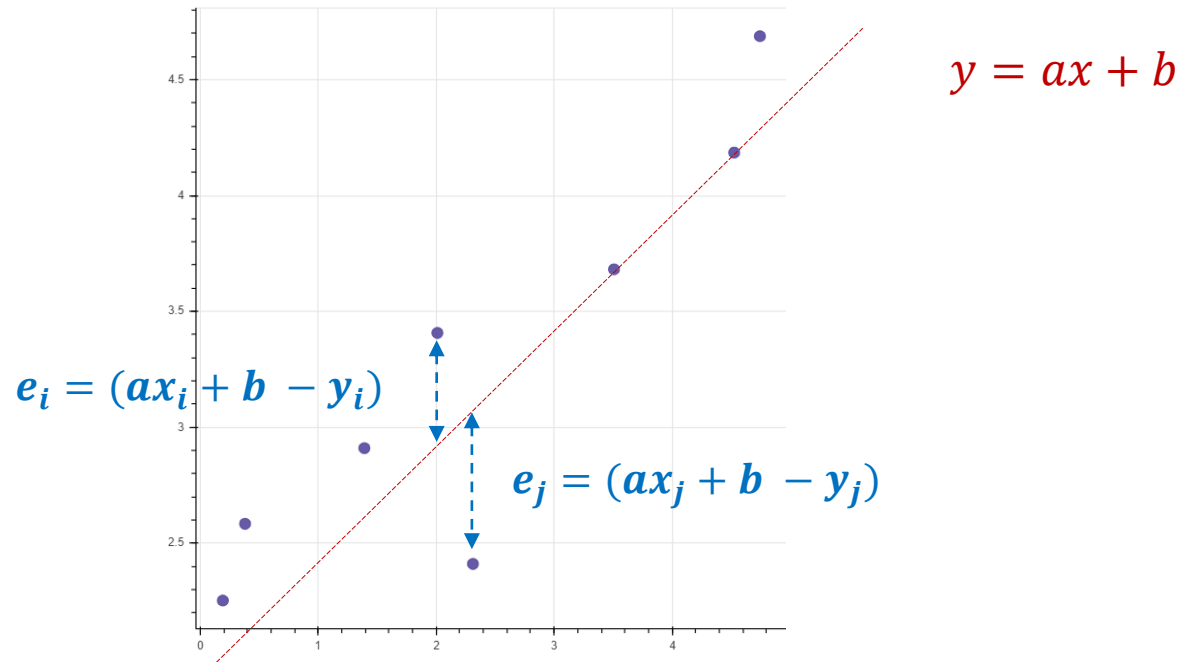
- 잔차 (residual,  $e_i$ ) 는 회귀선의 예측이 틀린 정도로 정의합니다.
  - 잔차는 음수가 될 수 있기 때문에 제곱의 합이 최소가 되는 직선을 선택합니다.
  - Loss 는 주어진 식이 얼마나 틀렸는지를 나타내는 척도입니다.

$$\text{loss} := \sum_i (ax_i + b - y_i)^2$$

minimize loss

$$\rightarrow \arg \min_{a,b} \sum_i (ax_i + b - y_i)^2$$

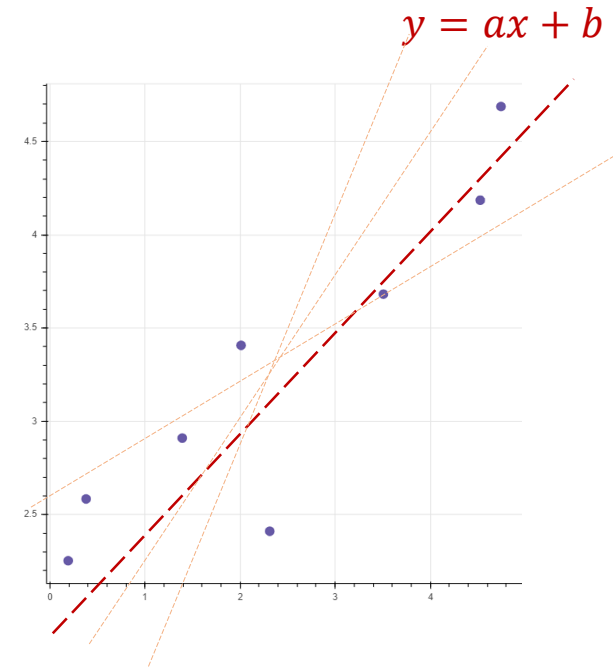
-----  
목적식 (objective)



# Univariate Linear Regression

- 잔차 제곱의 합으로 loss 를 정의하면 목적식은  $a, b$  의 2차 함수입니다.
  - 모델을 설명하는 변수  $a, b$  를 조절하여 loss 가 작은 직선을 탐색합니다.
  - 이 변수를 **패러미터 (parameters)** 라 합니다.

$$\begin{aligned} & \arg \min_{a,b} \sum_i (ax_i + b - y_i)^2 \\ \rightarrow & \sum_i (ax_i + b - y_i)^2 \\ = & \sum_i a^2 x_i^2 + b^2 + y_i^2 + 2(abx_i - ax_i y_i - by_i) \end{aligned}$$



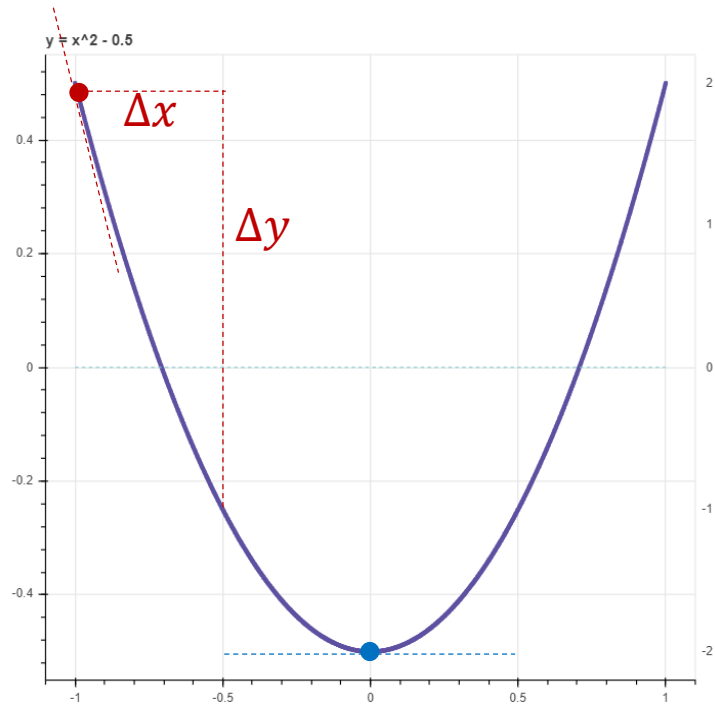
- $y = x^2 - 0.5$

- $\frac{dy}{dx} = 2x$

$\Delta y / \Delta x$  는  $(x, y)$  를 지나는 인접한 직선의 기울기입니다

$$\Delta x = 0.5, \Delta y = -0.75$$

$$\frac{\Delta y}{\Delta x} = -\frac{3}{2}$$



2 차 함수의 최저점에서의 인접한 직선의 기울기는 0 입니다.

# Least Squares (LS)

---

- 미분을 이용하면 ... 최저점에서 기울기가 0 이므로

$$\arg \min_{a,b} f = \sum_i (ax_i + b - y_i)^2$$

$$\frac{df}{db} = \sum_i 2(ax_i + b - y_i) = 0$$

$$\rightarrow nb = \sum_i y_i - \sum_i ax_i$$

$$\rightarrow b = \frac{\sum_i y_i}{n} - \frac{\sum_i ax_i}{n}$$

$$\rightarrow b = \bar{y} - a\bar{x}$$

$$\frac{df}{da} = \sum_i 2x_i(ax_i + b - y_i) = 0$$

$$= \sum_i 2x_i(ax_i - a\bar{x} + \bar{y} - y_i) = 0$$

$$\rightarrow \sum_i ax_i(x_i - \bar{x}) = \sum_i x_i(y_i - \bar{y})$$

$$\rightarrow a(\sum_i x_i(x_i - \bar{x}) - \bar{x} \sum_i (x_i - \bar{x})) = \sum_i x_i(y_i - \bar{y}) - \bar{x} \sum_i (y_i - \bar{y})$$

$$\rightarrow a = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

# Least Squares (LS)

---

- 입력변수가 2 개 이상이라면 ...

$$e_i = X_i\beta - y_i,$$

$$\arg \min_{\beta} \sum_i e_i^2 = (X\beta - y)^T(X\beta - y)$$

$$\rightarrow \frac{d}{d\beta} (X\beta - y)^T(X\beta - y) = -2X^T(X\beta - Y) = 0$$

$$\rightarrow (X^T X)\hat{\beta} = Xy$$

$$\rightarrow \hat{\beta} = (X^T X)^{-1}Xy$$

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}$$

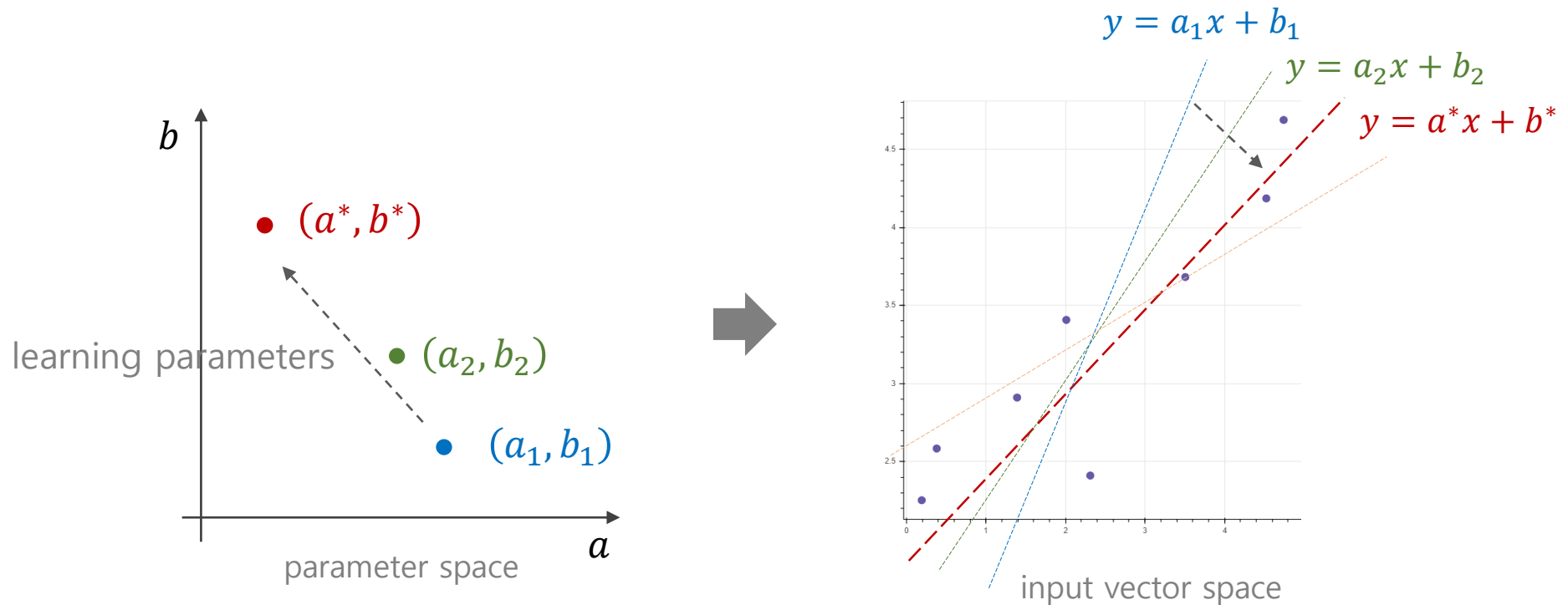
# Least Squares (LS)

---

- 수식으로 결정되는 최적해를 탐색하는 방법은 loss 식이 2차 함수처럼 간단한 모양일 때는 잘 작동합니다.
  - 하지만 모델이 복잡하면 loss 의 식을 수식으로 풀 수 없기도 합니다.
  - 대신 **경험적**으로 (empirically) **최적해**를 찾을 수도 있습니다.

# Gradient descent

- 경사하강법 (gradient descent) 는 현재의 직선식  $(a_1, b_1)$  보다 잔차제곱의 합이 더 작은 직선으로 조금씩 loss 를 줄여갑니다.





# Gradient descent

- $y = x^2 - 0.5$
- $\frac{dy}{dx} = 2x$

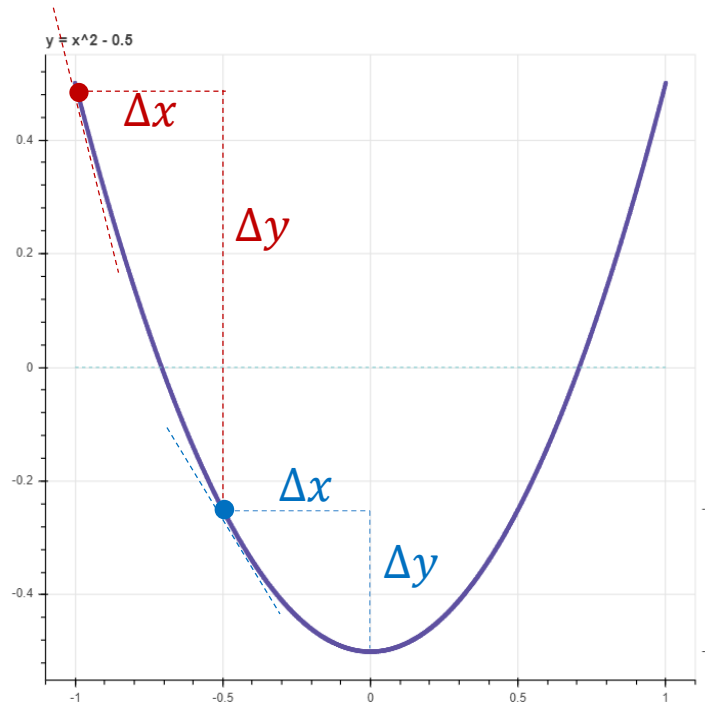
$\Delta y / \Delta x$  는  $(x, y)$  를 지나는 인접한 직선의 기울기입니다

$$\Delta x = 0.5, \Delta y = -0.75$$

$$\frac{\Delta y}{\Delta x} = -\frac{3}{2}$$

$$\Delta x = 0.5, \Delta y = -0.25$$

$$\frac{\Delta y}{\Delta x} = -\frac{1}{2}$$



$x$  를  $\Delta x$  만큼 이동하면  $y$  값이  $\Delta y$  만큼 증가합니다 (감소).

# Gradient descent

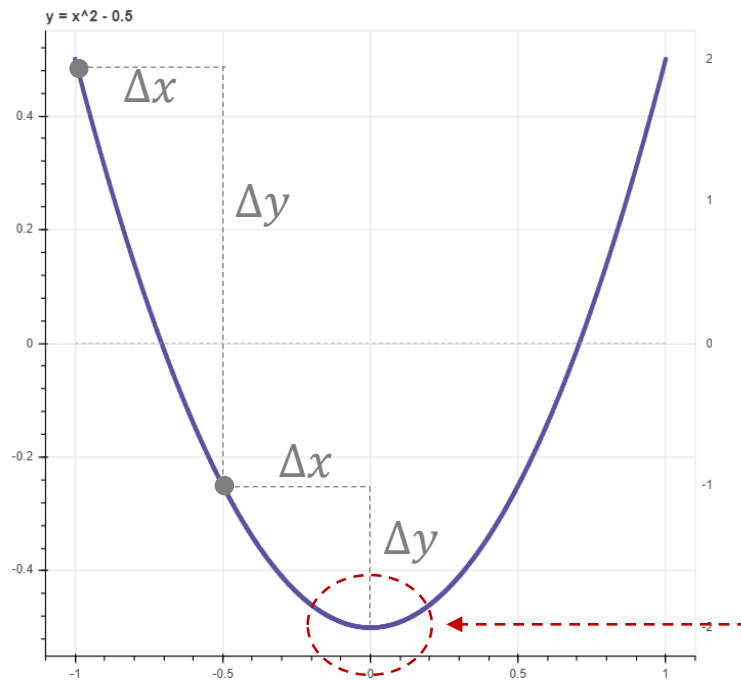
- $y = x^2 - 0.5$
- $\frac{dy}{dx} = 2x$

$$\Delta x = 0.5, \Delta y = -0.75$$

$$\frac{\Delta y}{\Delta x} = -\frac{3}{2}$$

$$\Delta x = 0.5, \Delta y = -0.25$$

$$\frac{\Delta y}{\Delta x} = -\frac{1}{2}$$



최저점 근처에서는 작은  $\Delta x$  에 대한  $\Delta y$  도 매우 작습니다.  
사실상 0 에 가깝습니다

# Gradient descent

최저점을 지나면  $\frac{\Delta y}{\Delta x}$  의 값이 양수입니다.

$x$  를 증가하면  $y$  도 증가합니다.  $x$  를 줄이는 방향으로 이동합니다.

- $y = x^2 - 0.5$

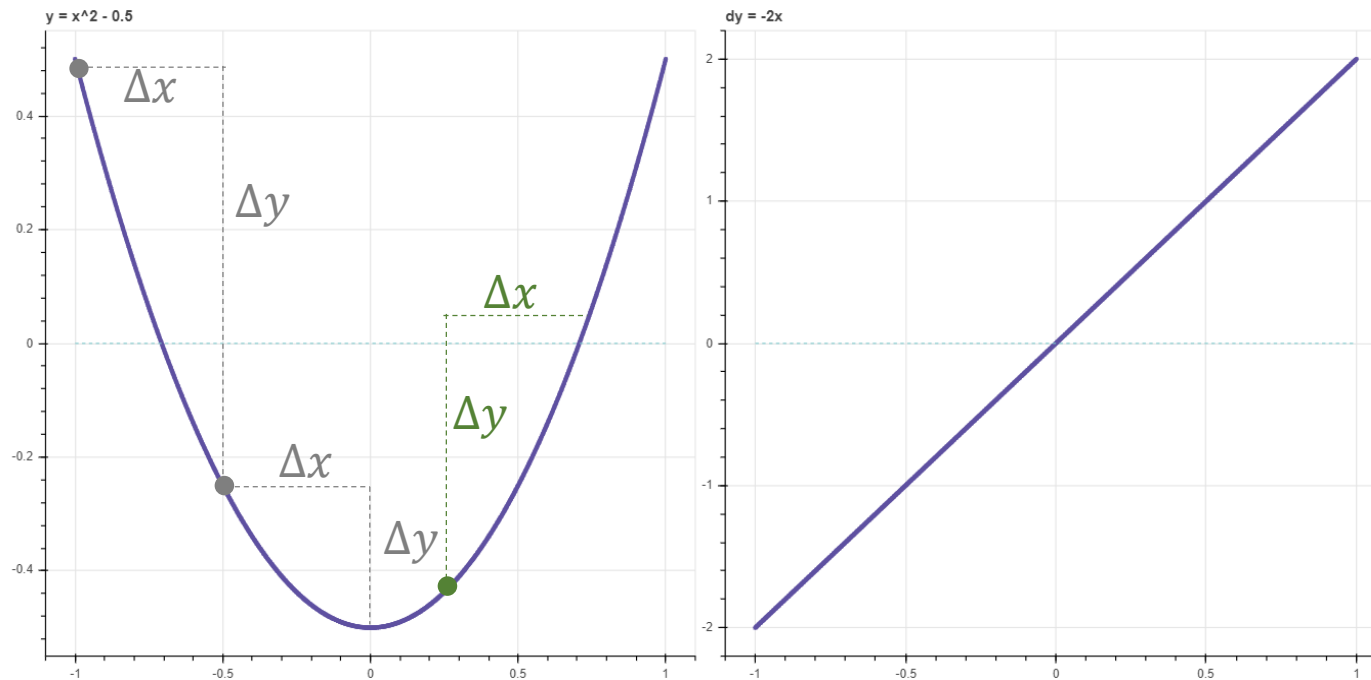
- $\frac{dy}{dx} = 2x$

$$\Delta x = 0.5, \Delta y = -0.75$$

$$\frac{\Delta y}{\Delta x} = -\frac{3}{2}$$

$$\Delta x = 0.5, \Delta y = -0.25$$

$$\frac{\Delta y}{\Delta x} = -\frac{1}{2}$$



도함수 ( $dy / dx$ ) 는 기울기의 관계를 표현한 함수입니다.

최저점에서 기울기의 값이 0 입니다.

# Gradient descent

- 경사하강법 (gradient descent) 은  $y$  가 줄어드는 방향으로  $x$  를 이동하여  $y$  의 최저점을 탐색합니다.
- $x$  는  $-dy/dx$  에 작은 수 (learning rate) 를 곱한 값만큼만 이동합니다.

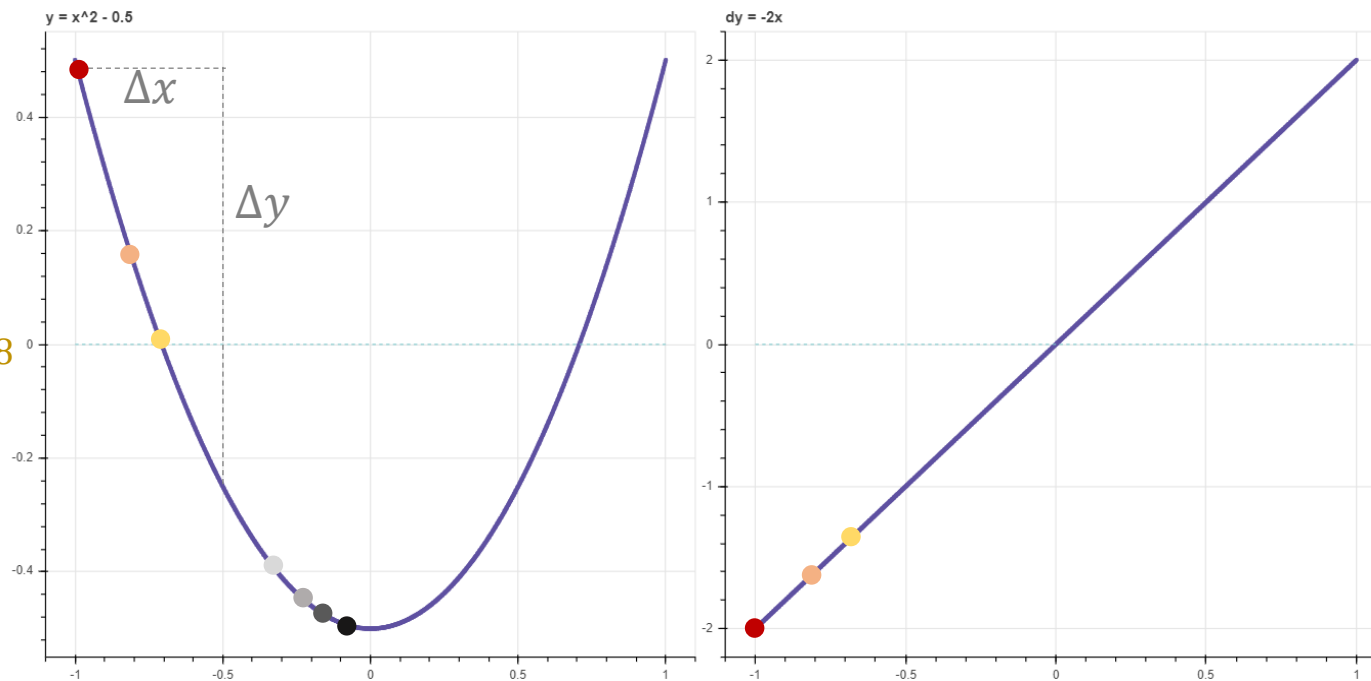
- $y = x^2 - 0.5$

- $\frac{dy}{dx} = 2x$

$$-\frac{dy}{dx}(x = -1) * 0.1 = -0.2$$

$$-\frac{dy}{dx}(x = -0.8) * 0.1 = -0.16$$

$$-\frac{dy}{dx}(x = -0.64) * 0.1 = -0.128$$

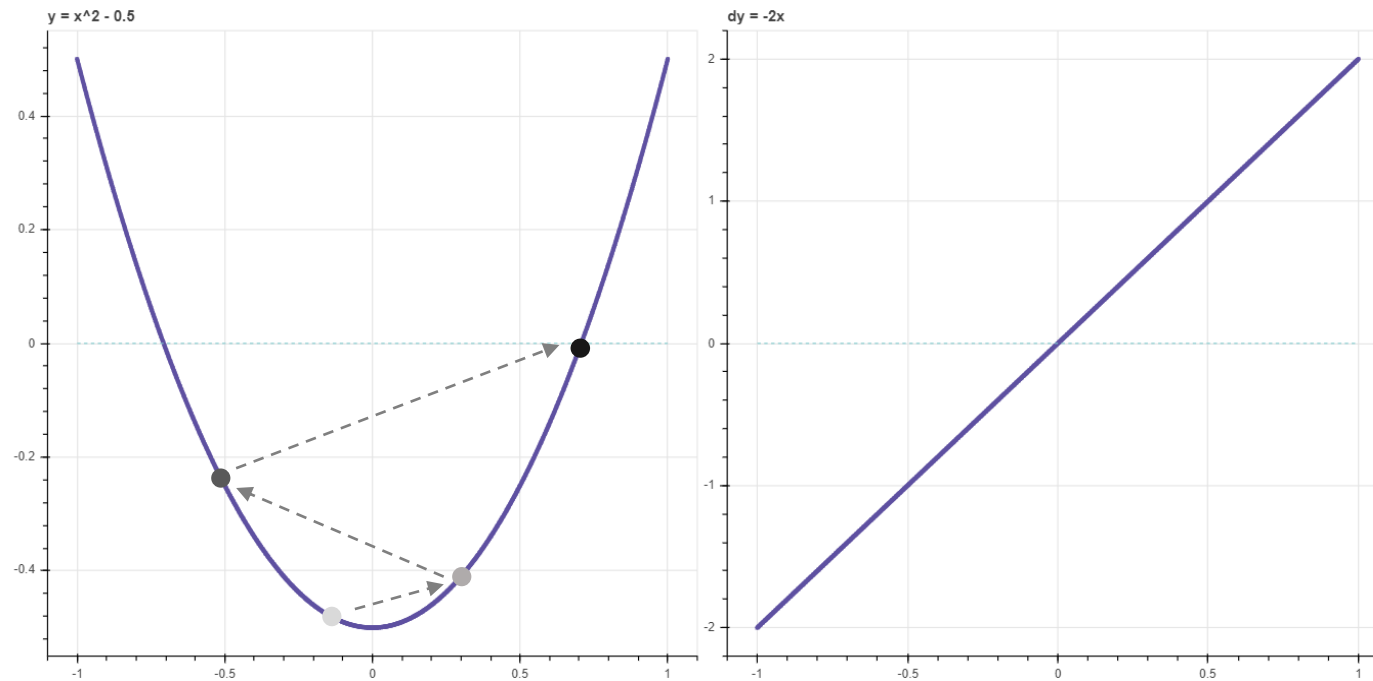


작은 수를 곱하기 때문에 최저점을 지나치지 않을 가능성이 높습니다.

# Gradient descent

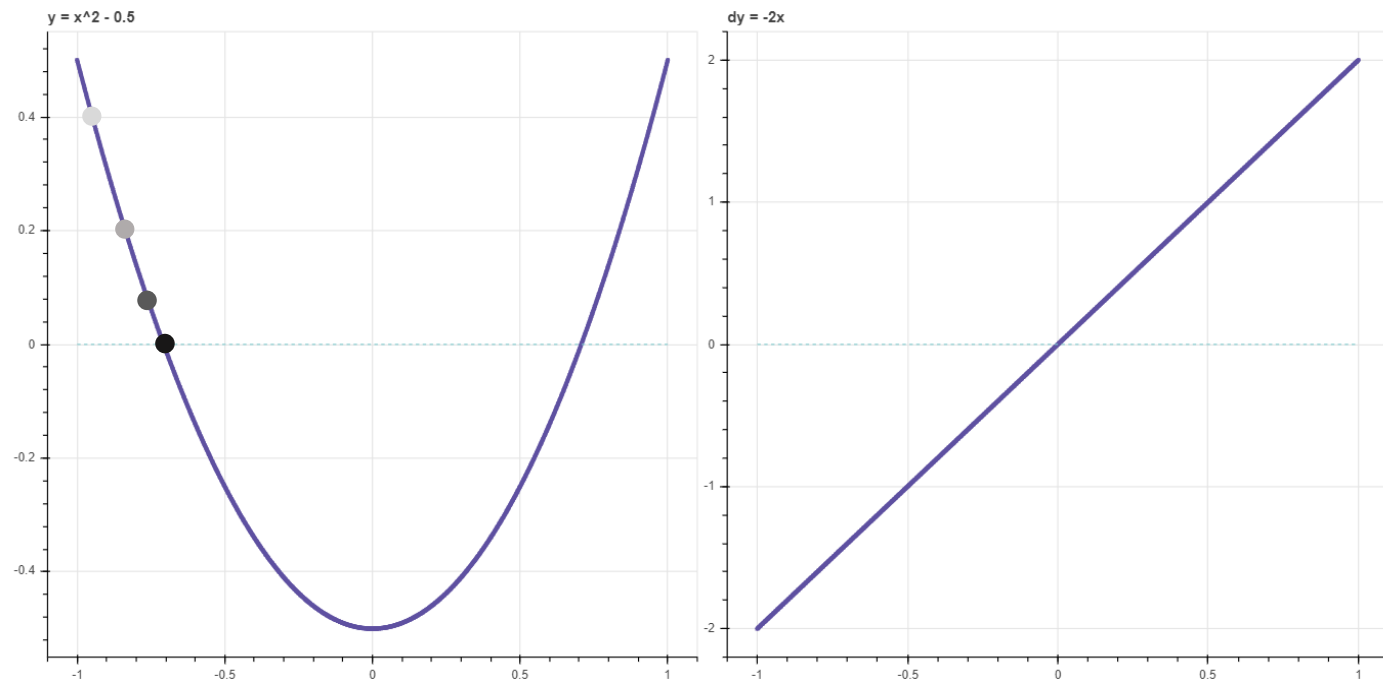
- Learning rate 로 큰 숫자를 이용하면 최저점을 지나 더 먼 곳으로 튕겨질 수 있습니다.  
이러한 현상을 "gradient explosion" 이라 합니다.
- Learning rate 로 작은 숫자를 이용하는 이유입니다.

- $y = x^2 - 0.5$
- $\frac{dy}{dx} = 2x$



# Gradient descent

- Explosion 을 방지하기 위하여 언제나 작은 learning 를 이용하면 최저점 탐색이 비효율적입니다
- 계속 같은 방향의 gradient 가 주어지는데도 불구하고 속도를 줄이면 최저점까지 이동하기 위한 steps 횟수가 많아집니다.



# Gradient descent

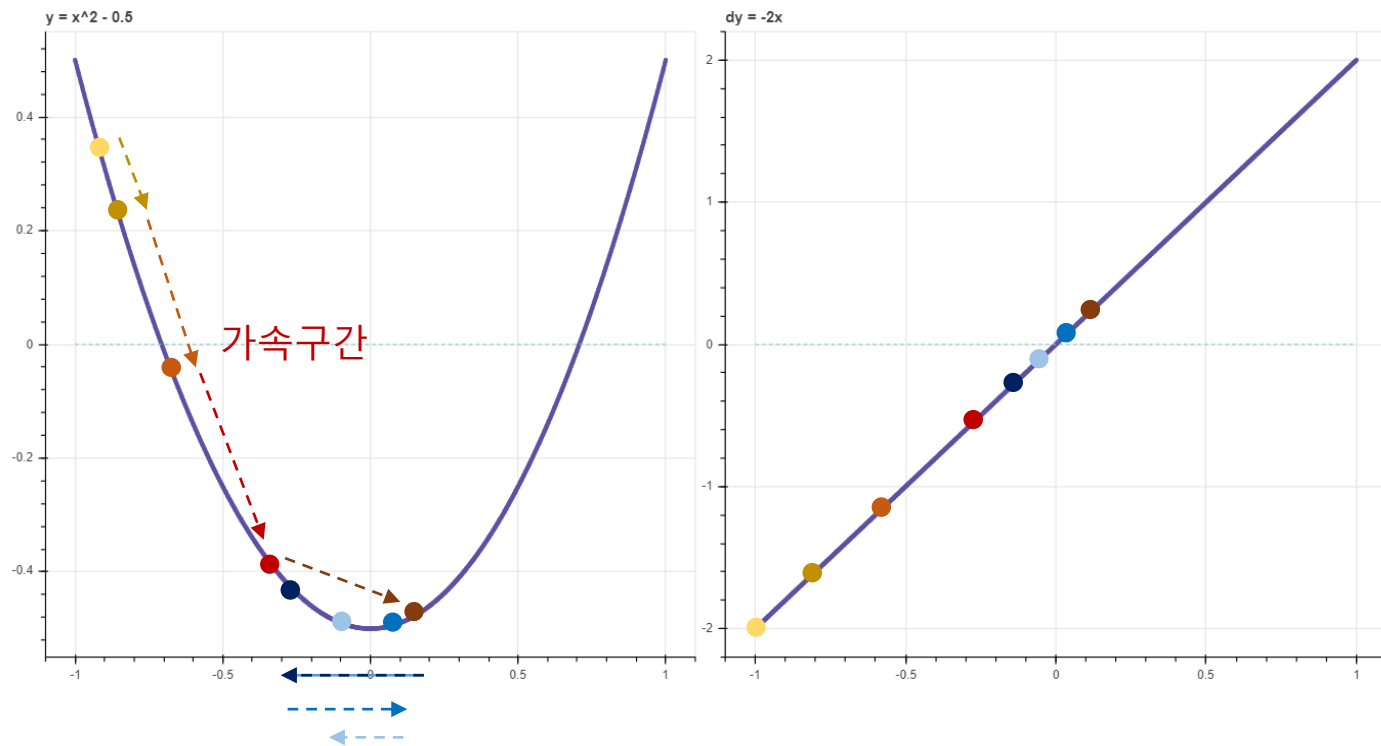
- Adaptive gradient (AdaGrad) 같은 방법은 이전  $dy/dx$  에 현재의  $dy/dx$  를 누적하여 방향이 동일하면 이동량을 가속합니다

$$\Delta x_1 = \frac{dy}{dx}(x_1)$$

$$\Delta x_2 = \alpha \frac{dy}{dx}(x_1) + \frac{dy}{dx}(x_2)$$

$$\Delta x_3 = \alpha^2 \frac{dy}{dx}(x_1) + \alpha \frac{dy}{dx}(x_2) + \frac{dy}{dx}(x_3)$$

$$\Delta x_n = \alpha \cdot \Delta x_{n-1} + \frac{dy}{dx}(x_n)$$



감속구간

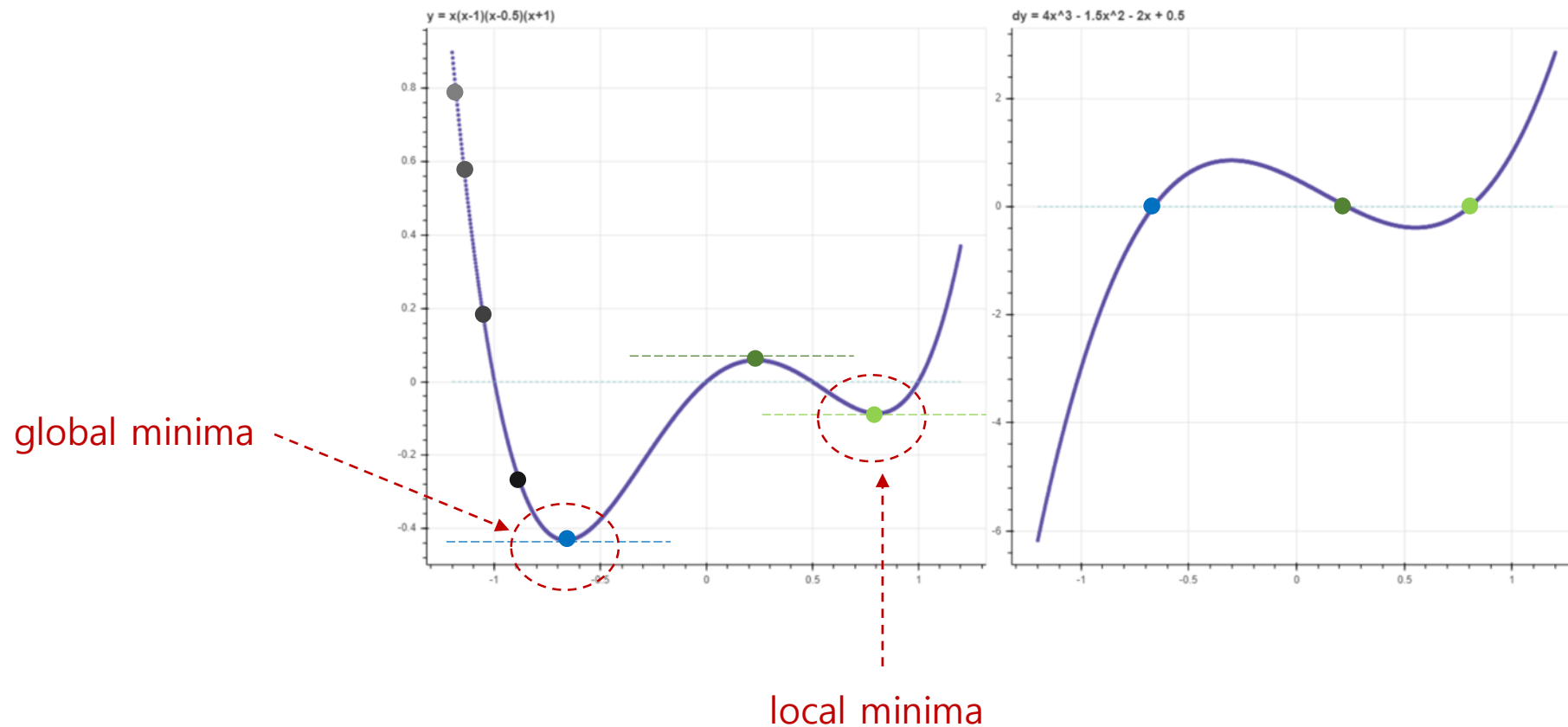
$\frac{dy}{dx}$  의 방향이 바뀌면 누적된 값이 줄어들어 감속됩니다

# Gradient descent

- 경사하강법의 접근은 복잡한 함수에도 적용할 수 있습니다
- 그러나 (지역적으로) 최저인 점이 여러 개 존재합니다.

- $y = x(x - 1)(x - 0.5)(x + 1)$

- $\frac{dy}{dx} = 4x^3 - 1.5x^2 - 2x + 0.5$





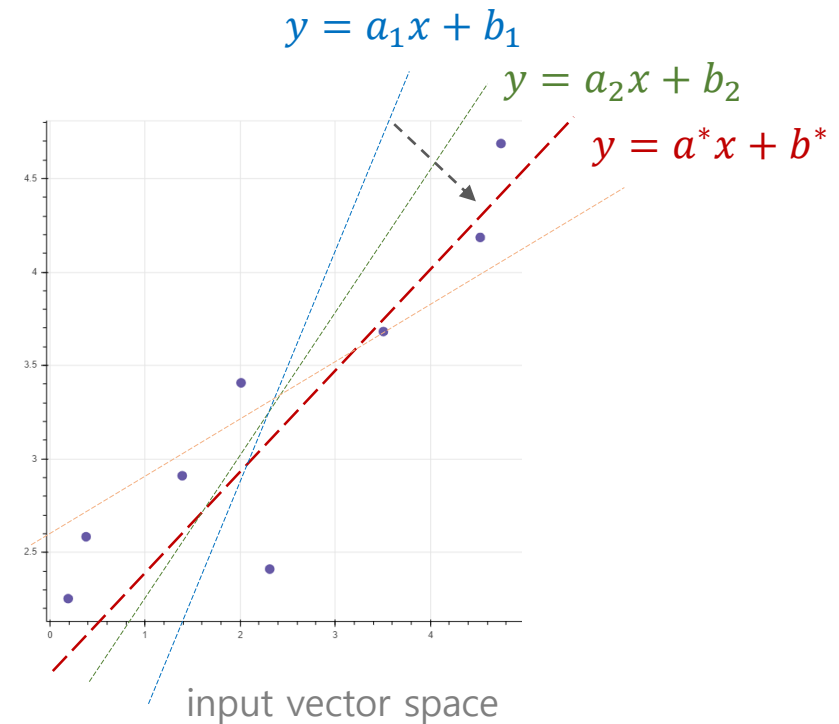
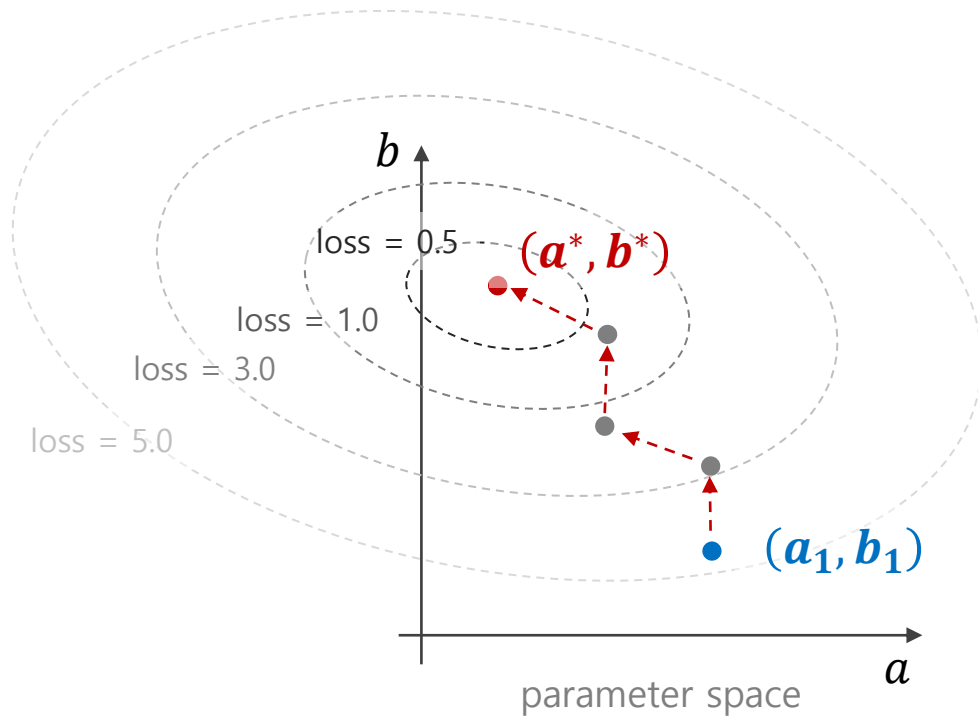
# Gradient descent

---

- 경사하강법 기반으로 패러미터 학습하기 위한 다양한 전략들이 있습니다.
  - 이들이 Adam, AdaGrad, RMSProp 등의 optimizers 입니다.
  - Local optima 를 방지하며, 효율적으로 최저점을 탐색하는 것이 공통된 목적입니다.

# Gradient descent

- 경사하강법 (gradient descent) 는 현재의 직선식  $(a_1, b_1)$  보다 잔차제곱의 합이 더 작은 직선으로 조금씩 loss 를 줄여갑니다.



# Gradient descent

---

- 정리하면,
  - 경사하강법은 모델의 품질을 측정하는 기준 (loss) 이 주어지면
  - 현재의 패러미터보다 **품질이 더 좋은 패러미터로** 조금씩 패러미터를 **수정**하는
  - 경험적인 패러미터 탐색법입니다.
- LS 처럼 수학적으로 해를 정의할 수 없을 때에도 이용가능합니다. 즉, 다양한 모델에 범용적으로 이용할 수 있다는 의미입니다.

# Maximum Likelihood Estimation (MLE)

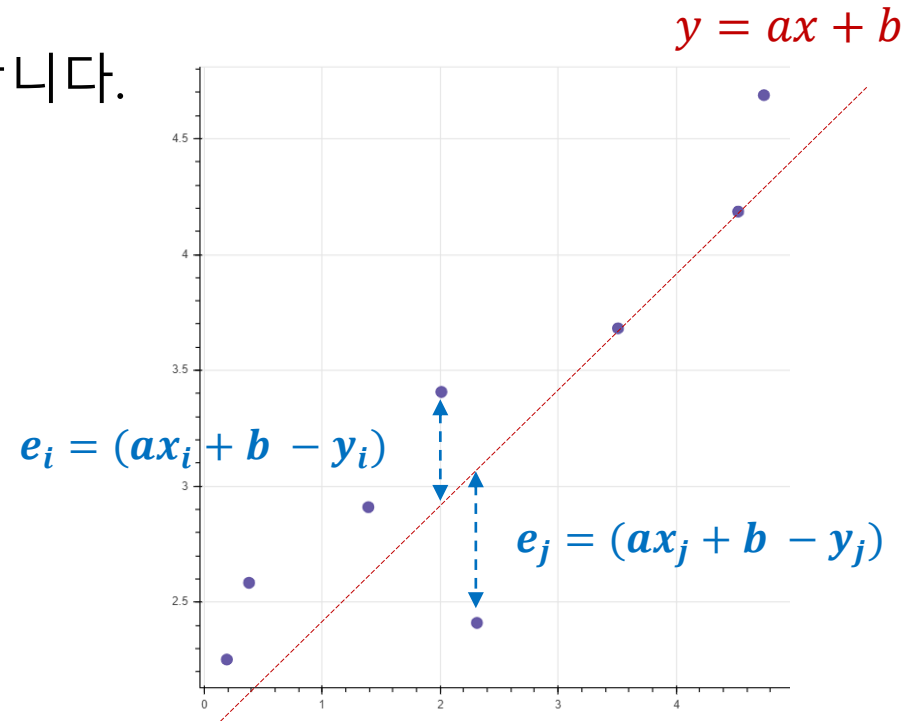
- 회귀직선의 품질을 다른 방식으로 정의할 수도 있습니다.
  - 주어진 식을 이용하여 예측한  $\hat{y} = ax_i + b$  의 값이  $y_i$  에 가깝게 만들고 싶습니다.
  - 잔차  $\hat{y} - y_i$  의 크기가 작을수록 확률이 높도록 확률식을 정의합니다.
  - 잔차는 정규분포  $N(0, \sigma)$  를 따른다고 가정합니다.

$$\text{Likelihood} := \prod_i P(e_i) = \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\left(\frac{e_i}{\sigma}\right)^2\right)$$

$$= \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\left(\frac{ax_i + b - y_i}{\sigma}\right)^2\right)$$

$$\arg \max_{a,b} \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\left(\frac{ax_i + b - y_i}{\sigma}\right)^2\right)$$

목적식 (objective)

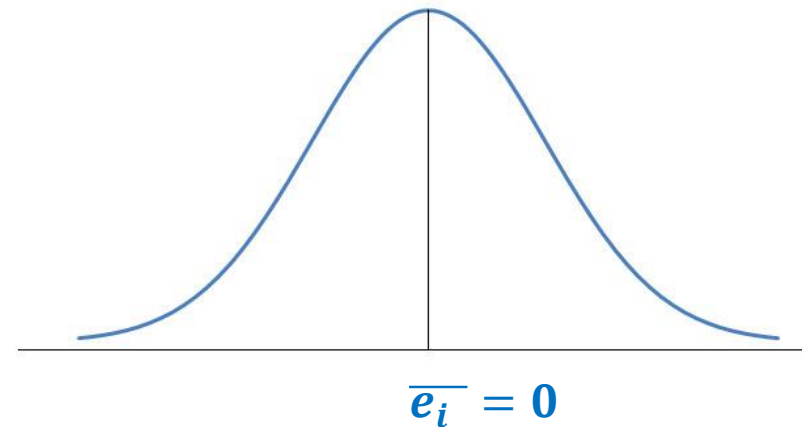


# Maximum Likelihood Estimation (MLE)

---

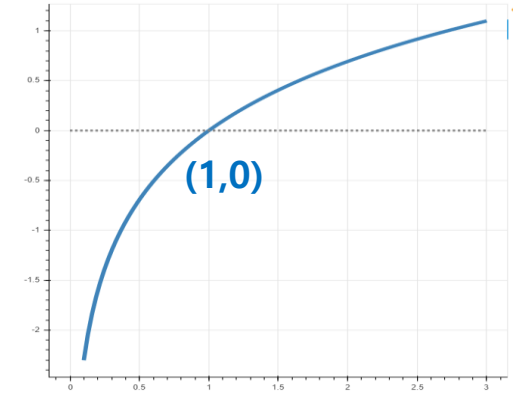
- MLE 는 확률을 최대화 하는 패러매터  $a, b$  를 탐색합니다.
  - 잔차가 작으면 확률이 큼니다.
  - 이번에도 경사하강법을 이용하여 패러매터를 탐색합니다.

$$\arg \max_{a,b} \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp \left( - \left( \frac{ax_i + b - y_i}{\sigma} \right)^2 \right)$$



# Negative Log Likelihood (NLL)

- log 함수는 단조증가 함수입니다.
  - 함수  $f(x)$  와  $\log(f(x))$  의 최소값의 위치가 같습니다.



$$\text{Likelihood} : \arg \max_{a,b} \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp \left( - \left( \frac{ax_i + b - y_i}{\sigma} \right)^2 \right)$$

$$\text{Log-likelihood: } \arg \max_{a,b} \sum_i \left( \log \left( \frac{1}{\sqrt{2\pi}\sigma} \right) + \log \exp \left( - \left( \frac{ax_i + b - y_i}{\sigma} \right)^2 \right) \right)$$

$$\rightarrow \arg \max_{a,b} \sum_i - \left( \frac{ax_i + b - y_i}{\sigma} \right)^2$$

로그를 덧씌우면 경사하강법을 적용하기 편리합니다.

$$\text{Negative LL} : \rightarrow \arg \min_{a,b} \sum_i \left( \frac{ax_i + b - y_i}{\sigma} \right)^2 = \arg \min_{a,b} \sum_i (ax_i + b - y_i)^2$$

부호를 바꾸면 최소값이 원함수의 최대값입니다.

# Negative Log Likelihood (NLL)

---

- 잔차 확률과 최소자승법으로 정의한 회귀직선의 품질 기준이 동일합니다.

$$\arg \max_{a,b} \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\left(\frac{ax_i + b - y_i}{\sigma}\right)^2\right) = \arg \min_{a,b} \sum_i (ax_i + b - y_i)^2$$

- NLL 의 접근방법은 임의의 확률기반 loss 에 이용할 수 있습니다.
  - NLL loss 는 확률이 작을수록 loss 가 크다는 의미입니다.

---

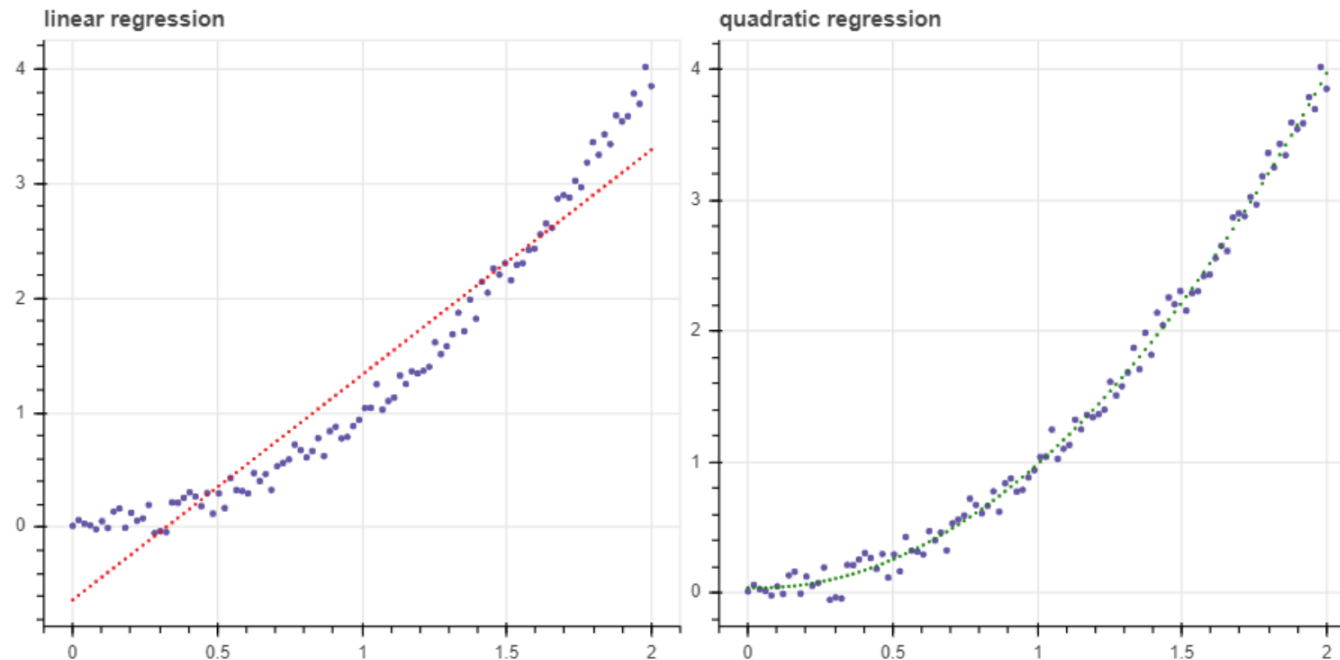
- 정리하면,

- 모델의 품질 기준을 설정합니다 (least square, log-likelihood)
- 경사하강법은 품질이 더 좋은 (loss 가 적은) 방향으로 매개변수를 이동합니다.
- 앞의 수식을 완벽히 이해하지는 않으셔도 됩니다.
- 매개변수를 이동할 때, explosion 이 일어나지 않도록 조금씩 움직입니다. 품질이 더 좋아지지 않으면 학습 (매개변수 이동)을 종료합니다.
- 수식을 통하여 최적해를 찾는 것이 아닌, 경험적으로 더 좋은 해로 개선합니다.



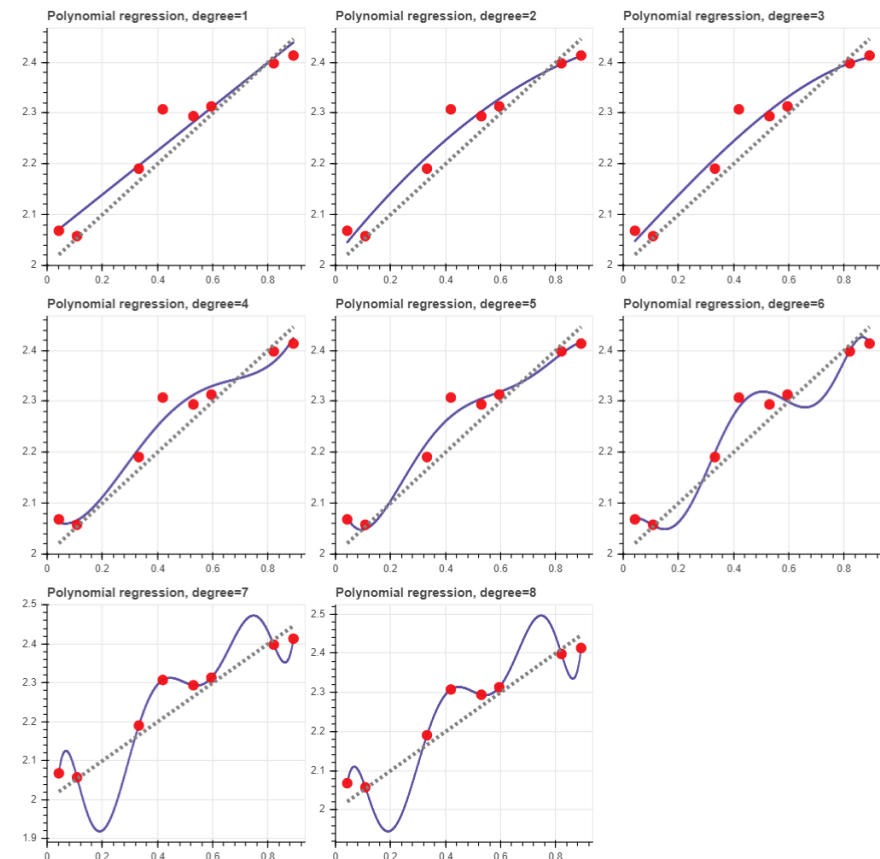
# Variable selection

- 주어진 변수의 설명력이 부족하면 변수를 추가할 수 있습니다.
  - 2차 함수곡선을 설명하기 위하여  $x \rightarrow (x, x^2)$  로 변수를 추가하면 예측이 잘됩니다.



# Variable selection

- 변수의 개수가 지나치게 많으면 과적합 (overfitting) 이 발생합니다.
  - 머신러닝 알고리즘은 어느 정도 "반복된" 데이터로부터 주요 패턴을 학습하는데 능숙합니다.
- LS 를 이용한 모델의 과적합 특징 중 하나는 매개변수의 값이 매우 큼니다.



# Variable selection

---

- 같은 예측 성능이라면 변수의 개수가 적은 모델이 효율적/안정적입니다.

*Entities are not to be multiplied without necessity*

*(Occam's razor)*

# Variable selection

---

- Exhaustive search 는 변수의 개수가  $p$  개일 때,  $2^p - 1$  개의 모든 조합에 대하여 모델링을 수행하는 것으로, 비현실적인 방법입니다.
  - Forward selection 은 유의미한 성능 향상이 있는 변수를 하나씩 추가하는 방법입니다.
  - Backward elimination 은 성능 하락이 없다면 변수를 하나씩 제거하는 방법입니다.
  - Stepwise selection 은 forward selection 과 backward elimination 을 동시에 진행합니다.
- 
- 성능 변화의 유의성은 통계 (ANOVA) 를 기반으로 검증합니다. 그러나 이러한 방법은 통계학에서 이용하는 방법이며, 머신러닝에서는 다른 방식으로 접근합니다.

# Regularization

---

- p-norm은 벡터의 크기를 정의하는 방법입니다

- $|X|_p = \sqrt[p]{|X_1|^p + \dots + |X_q|^p}$  로 정의되며,  $X_j$ 는 j번째 차원의 값

- $p = 2$  이면, L2 norm (Euclidean distance)

- $|(3, 0, 4)|_{p=2} = \sqrt{|3|^2 + 0^2 + |4|^2} = 5$

# Regularization

---

- p-norm은 벡터의 크기를 정의하는 방법입니다

- $|X|_p = \sqrt[p]{|X_1|^p + \dots + |X_q|^p}$  로 정의되며,  $X_j$ 는 j번째 차원의 값

- $p = 1$  이면, X의 각 차원의 값의 절대값의 합. (Manhattan distance)

- $|(3, 0, -4)|_{p=1} = \sqrt[1]{|3|^1 + |0|^1 + |-4|^1} = 7$

# Regularization

---

- p-norm은 벡터의 크기를 정의하는 방법입니다

- $|X|_p = \sqrt[p]{|X_1|^p + \dots + |X_q|^p}$  로 정의되며,  $X_j$ 는 j번째 차원의 값

- $p = 0$  이면,  $X$  에서 0이 아닌 차원의 갯수

- $|(3, 0, 4)|_{p=0} = |3|^0 + |4|^0 = 2$

# L2 regularization

- 목적식을 loss 와 regularization cost 의 합으로 정의합니다.
  - $\lambda$  는 loss 와 regularization 의 중요도를 조절하는 사용자 지정 변수입니다.
  - L2 regularization 은  $b_j$  중 유독 큰 값이 없도록 유도합니다. ( $\frac{d}{db_j} = 2\lambda b_j$  이므로)

$$\text{objective} : \arg \min_{a,b} \sum_i (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip} - y_i)^2 + \lambda \cdot (b_0^2 + b_1^2 + \dots + b_p^2)$$

↑  
loss

↑  
regularization cost

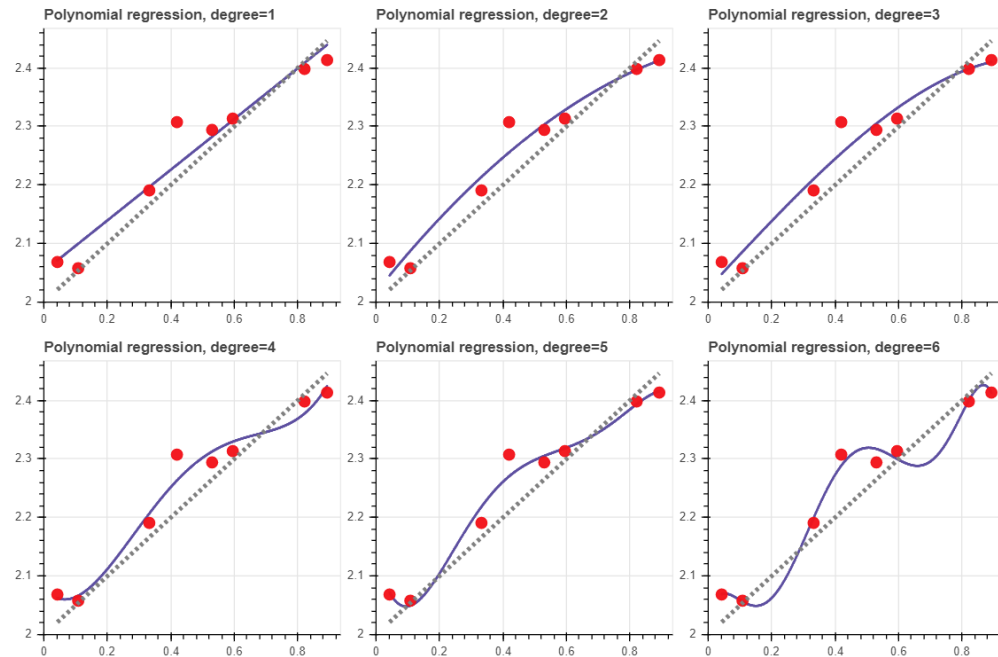
training cost = loss + regularization cost



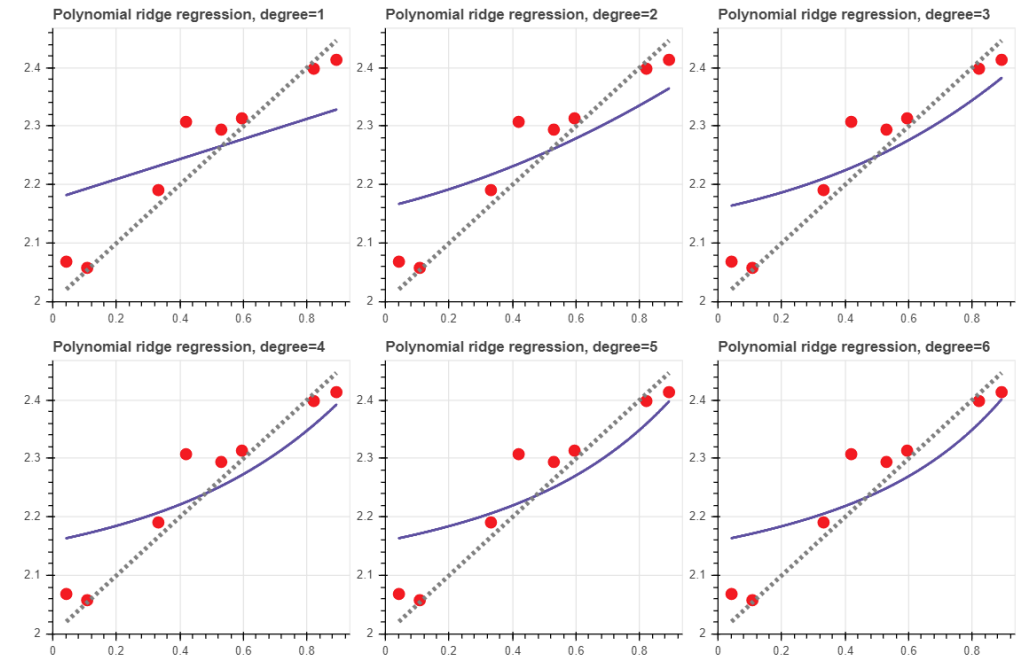
# Ridge regression

$$\arg \min_{\beta} \sum_i (X_i \beta - y_i)^2 + \lambda \cdot \|\beta\|_2^2$$

- L2 regularization 을 이용하는 선형회귀를 ridge regression 라 합니다.  
과적합 방지 능력이 있습니다.



polynomial linear regression



polynomial ridge regression


# L1 regularization

---

- L1 regularization 은 변수 선택의 효과를 얻을 수 있습니다.
  - $b_j = 0$  이면 해당 변수를 이용하지 않는다는 의미입니다.

$$\frac{d|b_j|}{db_j} = \begin{cases} 1 & \text{if } b_j > 0 \\ 0 & \text{if } b_j = 0 \\ -1 & \text{if } b_j < 0 \end{cases}$$

$$\text{objective} : \arg \min_{a,b} \sum_i (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip} - y_i)^2 + \lambda \cdot (|b_0| + |b_1| + \dots + |b_p|)$$

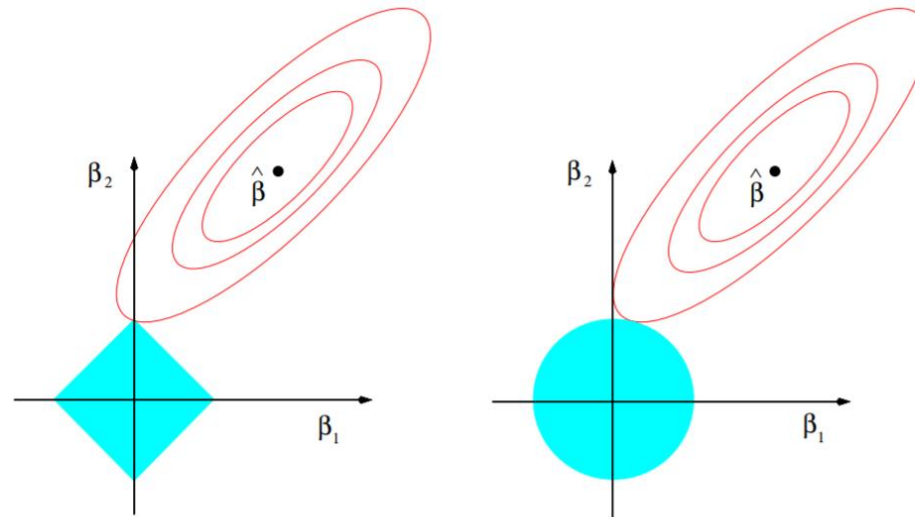


# L1 regularization

---

- 특히 변수가 데이터의 개수보다 많거나 (fat data  $n \leq p$ ) , 중복된 변수가 존재하는 경우에는 과적합 방지를 위하여 L1 regularization 이 필요합니다.
  - L1 regularization 은 logistic regression 에서 더 자세히 살펴봅니다.

- The Elements of Statistical Learning <sup>^1</sup> 에 L1 regularization 이 왜 sparse coefficient 를 학습하는지에 대한 수학적인 설명이 있습니다.



**FIGURE 3.11.** Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions  $|\beta_1| + |\beta_2| \leq t$  and  $\beta_1^2 + \beta_2^2 \leq t^2$ , respectively, while the red ellipses are the contours of the least squares error function.

# Evaluation

## $R^2$

---

- $y$  의 분산 대비 잔차의 비율이 작을수록 모델이 데이터를 잘 설명합니다.
- $R^2 \leq 1$  로, 1에 가까울수록 잔차가 없음을 의미합니다.

$$R^2 = \frac{\sum_i (\hat{y}_i - \bar{y})^2}{\sum_i (y_i - \bar{y})^2} = 1 - \frac{\sum_i (e_i)^2}{\sum_i (y_i - \bar{y})^2}$$

$$\begin{aligned} & \sum_i (y_i - \bar{y})^2 \\ &= \sum_i (y_i - \hat{y}_i + \hat{y}_i - \bar{y})^2 \\ &= \sum_i (y_i - \hat{y}_i)^2 + \sum_i (\hat{y}_i - \bar{y})^2 \\ &= \sum_i e_i^2 + \sum_i (\hat{y}_i - \bar{y})^2 \\ &\Rightarrow R^2 = 1 - \frac{\sum_i (e_i)^2}{\sum_i (y_i - \bar{y})^2} \end{aligned}$$

# Adjusted R<sup>2</sup>

---

- 변수의 개수를 고려한 R<sup>2</sup> 입니다.
  - 변수 개수에 penalty 를 부여합니다.

$$R_{adj}^2 = 1 - \left( \frac{n-1}{n-k-1} \right) (1 - R^2)$$

$n$  : number of data

$k$  : number of input variables

# Average Error

- 잔차의 크기가 크더라도 평균은 0 일 수 있습니다.

$$avg.error = \frac{1}{n} \sum_i (\hat{y}_i - y_i)$$

$y_i$	$\hat{y}_i$	residual
5.329	5.212	-0.117
1.812	1.792	-0.02
4.781	4.771	-0.01
6.53	6.44	-0.09
6.265	6.242	-0.023
5.334	5.572	0.238
3.362	3.402	0.04
2.759	2.74	-0.019

Average error = -0.000125



# Mean Absolute Error (MAE)

- $y_i$  의 크기에 관계없이 잔차의 영향력이 같습니다.
  - (1, 2) 에서의 0.1 과 (10, 20) 에서의 0.1 잔차는 다릅니다.

$$\text{MAE} = \frac{1}{n} \sum_i |\hat{y}_i - y_i|$$

$y_i$	$\hat{y}_i$	$ y_i - \hat{y}_i $
5.329	5.212	-0.117
1.812	1.792	-0.02
4.781	4.771	-0.01
6.53	6.44	-0.09
6.265	6.242	-0.023
5.334	5.572	0.238
3.362	3.402	0.04
2.759	2.74	-0.019

MAE = -0.069

# Mean Absolute Percentage Error (MAPE)

- $y_i$  의 크기를 고려한 잔차의 상대적 크기를 계산합니다.

$$\text{MAPE} = \frac{1}{n} \sum_i \frac{|\hat{y}_i - y_i|}{|y_i|}$$

$y_i$	$\hat{y}_i$	$\left  \frac{y_i - \hat{y}_i}{y_i} \right $
5.329	5.212	-0.117
1.812	1.792	-0.02
4.781	4.771	-0.01
6.53	6.44	-0.09
6.265	6.242	-0.023
5.334	5.572	0.238
3.362	3.402	0.04
2.759	2.74	-0.019

MAPE = 0.014

# Root Mean Squared Error (RMSE)

- $e_i$  의 크기가 클 경우 더 큰 패널티를 부여합니다.
- RMSE 는 잔차와  $y_i$  의 스케일을 동일하게 맞추습니다.

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

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_i (\hat{y}_i - y_i)^2}$$

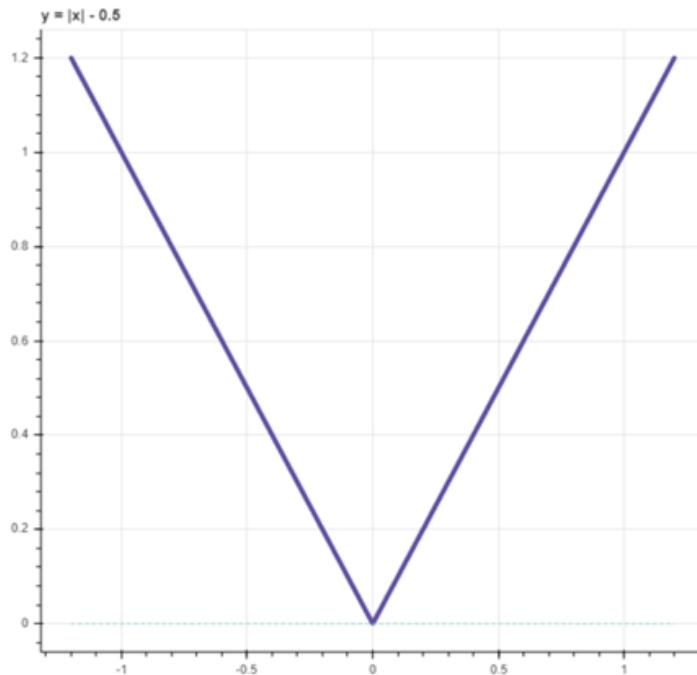
$y_i$	$\hat{y}_i$	$(y_i - \hat{y}_i)^2$
5.329	5.212	-0.117
1.812	1.792	-0.02
4.781	4.771	-0.01
6.53	6.44	-0.09
6.265	6.242	-0.023
5.334	5.572	0.238
3.362	3.402	0.04
2.759	2.74	-0.019

RMSE = 0.101

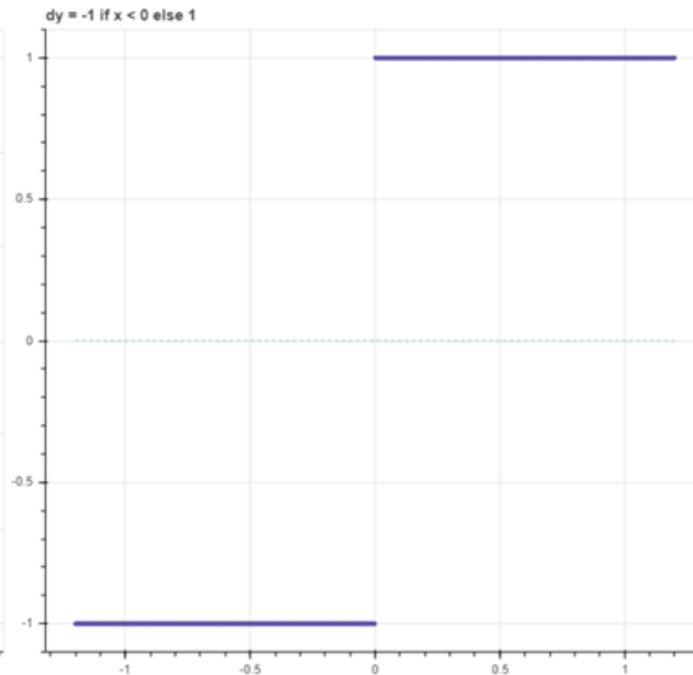
# Why we use RMSE loss instead MAP

- $|e_i|$  의 미분값은 -1, 1 입니다.
  - 최저점 근방에 도착하여도 속도가 줄어들지 않습니다.

$$y = |e| - 0.5$$



$$\frac{dy}{de} = \begin{cases} 1 & \text{if } e > 0 \\ -1 & \text{if } e < 0 \end{cases}$$



- 
- Evaluation metric 은 사후 품질 평가 수단으로도 이용되지만, 경사하강법에서는 패러미터의 이동 방향을 결정하는 지표로도 이용됩니다.
  - 손쉬운 미분이 가능한 평가 척도가 경사하강법의 loss 로 자주 이용됩니다.

# Seaborn

# scatter plot 을 그리는 두 가지 방법

```
import seaborn as sns
```

```
g = sns.scatterplot(x="total_bill", y="tip", data=tips)
```

```
# to save
```

```
g = g.get_figure()
```

```
g = savefig("filepath")
```

return type: matplotlib.axes.\_subplots.AxesSubplot

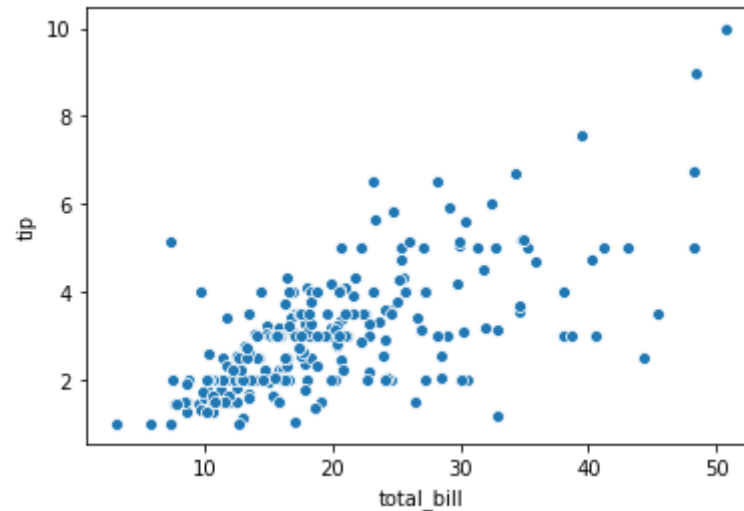
```
import seaborn as sns
```

```
g = sns.relplot(x="total_bill", y="tip", data=tips, kind="scatter")
```

```
# to save
```

```
g = savefig("filepath")
```

return type: seaborn.axisgrid.FacetGrid



# relplot() vs scatterplot()

---

seaborn.relplot(..., kind='scatter')

└─ **execute** matplotlib.pyplot.close()  
    **prepare** new FacetGrid  
    **execute** seaborn.scatterplot()  
    **return** the grid

← return type = AxesSubplot  
scatterplot 의 옵션은 모두 이용가능합니다 (hue, size, ...)

seaborn.relplot(..., kind='line')

└─ **execute** matplotlib.pyplot.close()  
    **prepare** new figure  
    **execute** seaborn.lineplot()  
    **return** the grid



# relplot() vs scatterplot()

---

```
seaborn.relplot(..., kind='scatter', col='variable', col_wrap=3)
```

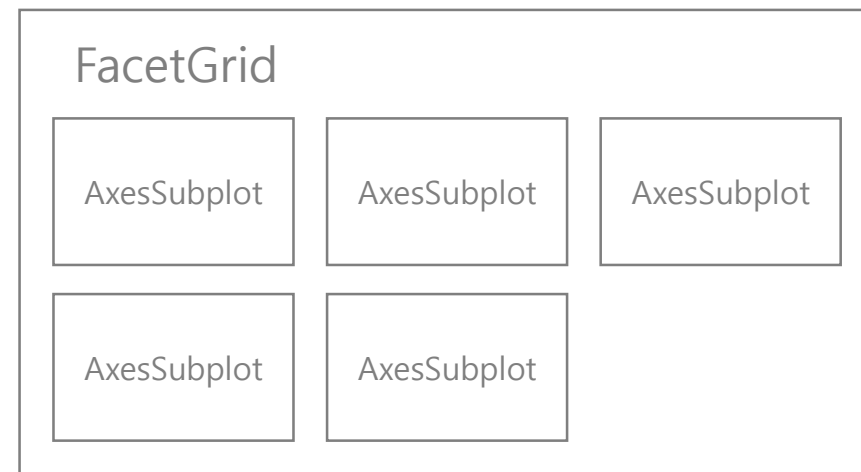
└─ **execute** `matplotlib.pyplot.close()`  
   **prepare** new `FacetGrid`

for var in variable:

**execute** `seaborn.scatterplot(x_var, y_var)`

**return** the grid

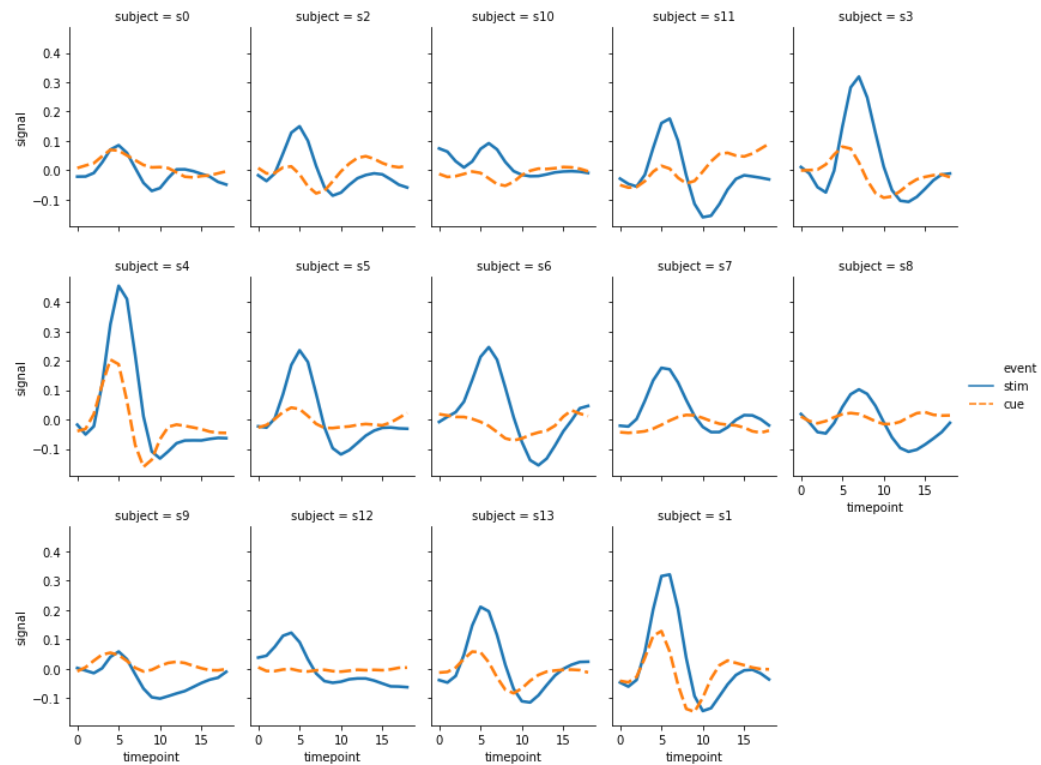
return type = `AxesSubplot`



# relplot() vs scatterplot()

```
import seaborn as sns
```

```
g = sns.relplot(..., col="subject", col_wrap=5)
```



---

`seaborn.relplot(..., kind='scatter')`

- 'scatter'
- 'line'

`seaborn.catplot(..., kind='box')`

- 'strip'
- 'swarm'
- 'box'
- 'violin'
- 'point'
- 'bar'
- 'count'

# Pairwise plot

- 변수 간의 관계에 대하여 모두 relplot, catplot 을 그려줍니다.
  - 변수의 개수  $p$  의  $p^2$  개의 그림을 그리기 때문에 필요한 변수들을 꼭 설정합니다.
  - Lower, upper, diagonal 에 모두 다른 플랏을 그릴 수 있습니다.

```
import seaborn as sns

g = sns.PairGrid(data, hue= ...)
g = sns.PairGrid(data, vars=["var1", "var2"], hue= ...)
g = g.map_upper(plt.scatter)
g = g.map_lower(sns.kdeplot)
g = g.map_diag(all_functions(x,y, ...))
```

