

Logistic Regression

Hyunjoong Kim

soy.lovit@gmail.com

github.com/lovit

Regression

- 회귀 (regression) 문제는 다양한 변수 (input variables) 를 이용하여 연속형 값 (output variable) 을 예측합니다.
 - 변수간의 관계를 함수 F 로 학습합니다.
 - 'fare' = $F(\text{survived, pclass, sex, age})$

	input variable				output variable
	Survived	Pclass	Sex	Age	Fare
PassengerId					
1	0	3	male	22.0	7.2500
2	1	1	female	38.0	71.2833
3	1	3	female	26.0	7.9250
4	1	1	female	35.0	53.1000
5	0	3	male	35.0	8.0500
6	0	3	male	NaN	8.4583
7	0	1	male	54.0	51.8625
8	0	3	male	2.0	21.0750
9	1	3	female	27.0	11.1333
10	1	2	female	14.0	30.0708

Classification

- 분류 문제는 입력변수로부터 이산형 출력 (categorical output) 을 예측합니다.
 - 출력값의 종류가 2 종류면 이진분류 (binary classification) 라 합니다.
 - 'survived' = $F(\text{pclass}, \text{sex}, \text{age}, \text{fare})$

	output variable	input variable			
	Survived	Pclass	Sex	Age	Fare
PassengerId					
1	0	3	male	22.0	7.2500
2	1	1	female	38.0	71.2833
3	1	3	female	26.0	7.9250
4	1	1	female	35.0	53.1000
5	0	3	male	35.0	8.0500
6	0	3	male	NaN	8.4583
7	0	1	male	54.0	51.8625
8	0	3	male	2.0	21.0750
9	1	3	female	27.0	11.1333
10	1	2	female	14.0	30.0708

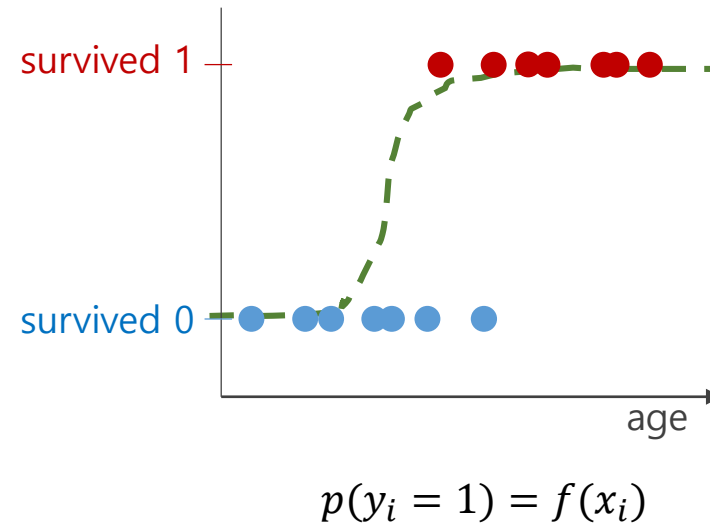
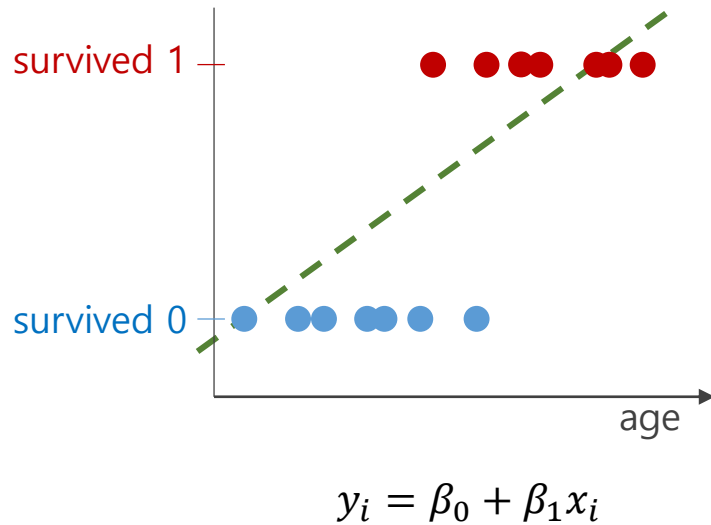
Classification

- 분류 문제는 입력변수로부터 이산형 출력 (categorical output) 을 예측합니다.
 - 출력값의 종류가 3 종류면 multiclass classification 이라 합니다.
 - 'pclass' = $F(\text{sex}, \text{age}, \text{fare})$

		output variable		input variable		
	Survived	Pclass	Sex	Age	Fare	
PassengerId						
1	0	3	male	22.0	7.2500	
2	1	1	female	38.0	71.2833	
3	1	3	female	26.0	7.9250	
4	1	1	female	35.0	53.1000	
5	0	3	male	35.0	8.0500	
6	0	3	male	NaN	8.4583	
7	0	1	male	54.0	51.8625	
8	0	3	male	2.0	21.0750	
9	1	3	female	27.0	11.1333	
10	1	2	female	14.0	30.0708	

Logistic Regression

- 회귀 모델로는 이산형 출력값을 그대로 모델링 할 수 없습니다.
 - "survived 1" 일 확률은 연속형 변수이므로, 이를 회귀 모델로 학습합니다.



Logistic Regression

- Odds 는 확률의 비율로, 승산의 의미를 지닙니다.

- $odds = \frac{p}{1-p}$

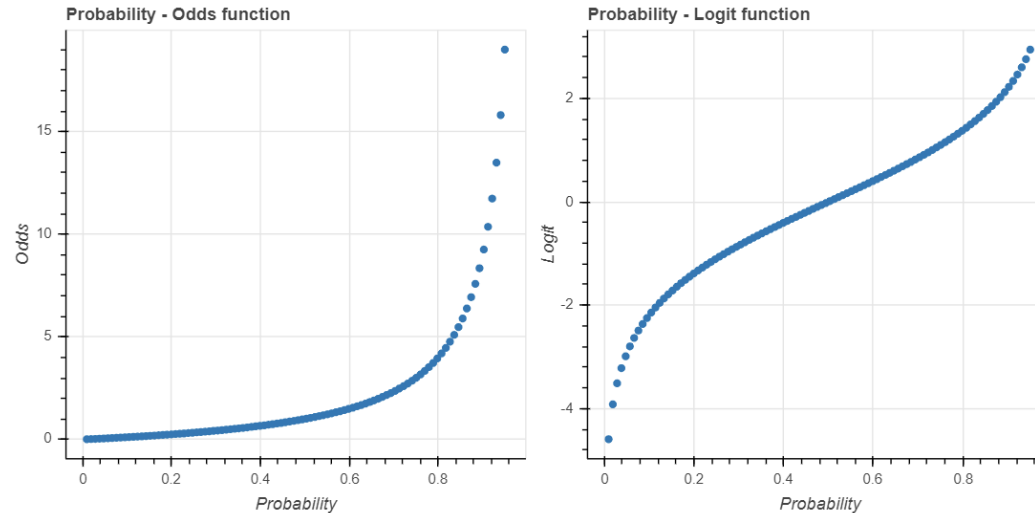
- $p = 0.2 \rightarrow odds = \frac{0.2}{0.8} = 0.25$

- $p = 0.8 \rightarrow odds = \frac{0.8}{0.2} = 4$ ←----- 성공확률이 80% 이면, 성공확률은 실패확률보다 4배 크다

- $p = 0.5 \rightarrow odds = \frac{0.5}{0.5} = 1$

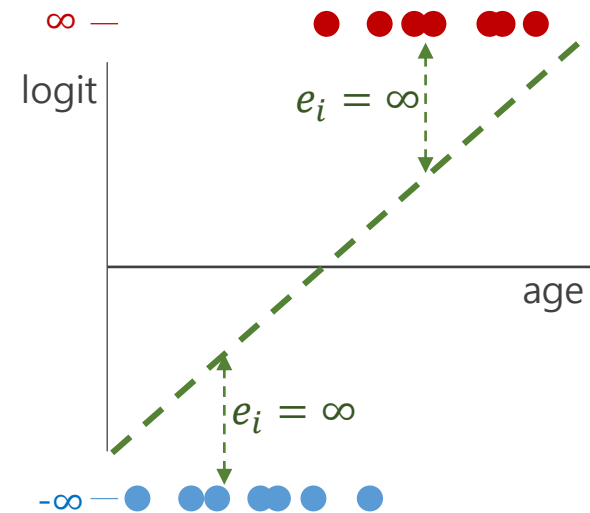
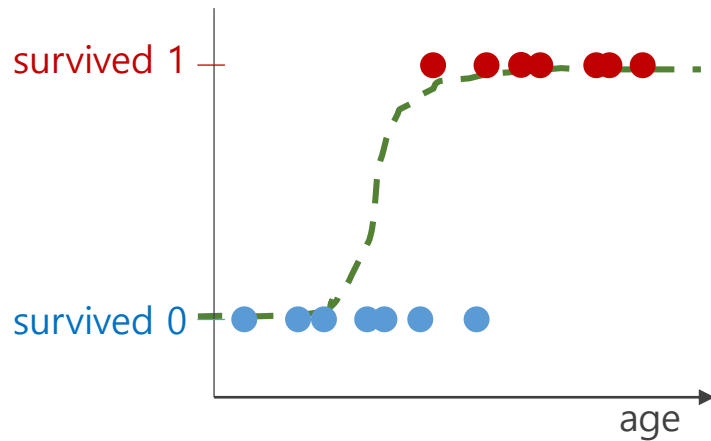
Logistic Regression

- 회귀식의 범위와 출력 변수와의 범위가 일치하지 않습니다.
 - 회귀식 출력의 범위는 $(-\infty, \infty)$ 이지만, 확률의 범위는 $[0, 1]$ 입니다.
 - odds ($\frac{p}{1-p}$) 의 범위도 $(0, \infty)$ 입니다.
 - Log odds (logit, $\ln \frac{p}{1-p}$) 의 범위는 $(-\infty, \infty)$ 입니다.



Logistic Regression

- $x^T \beta = \ln \frac{p}{1-p}$ 의 모델을 학습할 수 있도록 데이터를 변형합니다.
- 그러나 $p = 0, p = 1$ 일때의 logit 은 각각 $-\infty, \infty$ 입니다.
- 잔차 (residual) 가 정의되지 않습니다. Least square 를 이용할 수 없습니다.



Logistic Regression

- Least square 대신 maximum likelihood estimation 을 이용합니다.

$$x^T \beta = \ln \left(\frac{p}{1-p} \right)$$

$$\rightarrow e^{(x^T \beta)} = \frac{p}{1-p}$$

$$(e^a)^b = e^{ab}$$

$$\rightarrow e^{(-x^T \beta)} = \frac{1-p}{p} = \frac{1}{p} - 1$$

$$\rightarrow 1 + e^{(-x^T \beta)} = \frac{1}{p}$$

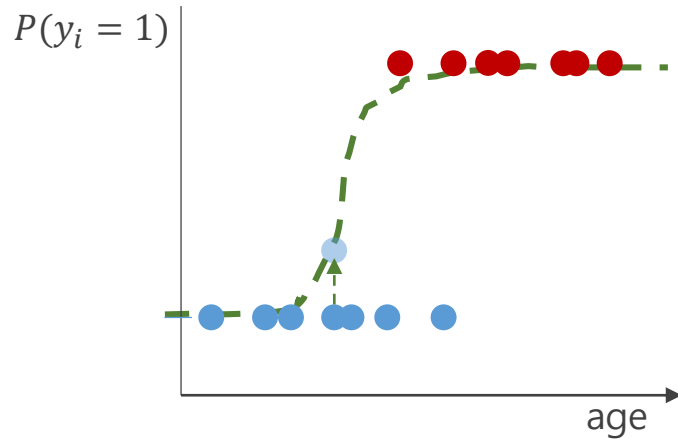
logistic regression function

$$\rightarrow \frac{1}{1 + e^{(-x^T \beta)}} = p$$

Logistic Regression

- Least square 대신 maximum likelihood estimation 을 이용합니다.

$$\arg \max_{\beta} \prod_{i=1}^n P(f(x_i) = y_i)$$



← $y_i = 1$ 인 점들은 $\frac{1}{1+e^{(-x_i^T \beta)}}$ 의 값이 1 에 가깝게

$y_i = 0$ 인 점들은 $\frac{1}{1+e^{(-x_i^T \beta)}}$ 의 값이 0 에 가깝게

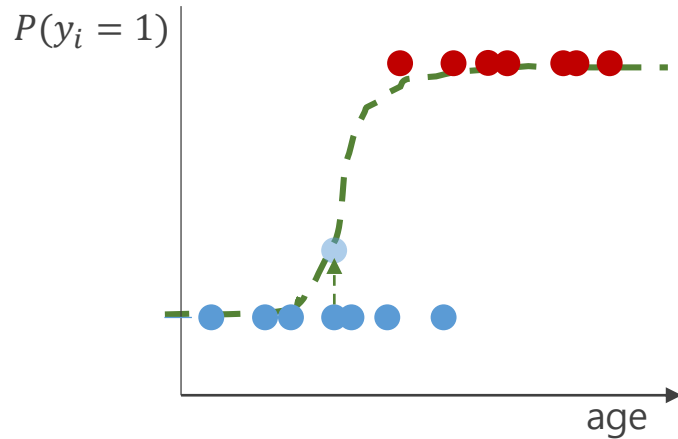
← 예측하는 **패러미터 β** 를 **탐색**해야 합니다.

탐색법은 뒤에서 자세히 살펴봅니다.

Logistic Regression

- Logistic regression 의 식은 선형 경계면을 의미합니다.

- $$\frac{1}{1+e^{-(\beta_0+\beta_1 x_{i1})}} = P(y_i = 1)$$



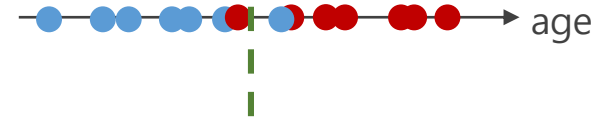
$$\frac{1}{1+e^{-(\beta_0+\beta_1 x_{i1})}} = 0.5$$

$$\rightarrow e^{-(\beta_0+\beta_1 x_{i1})} = 1$$

$$\rightarrow \beta_0 + \beta_1 x_{i1} = 0$$

$$\beta_0 + \beta_1 x = 0$$

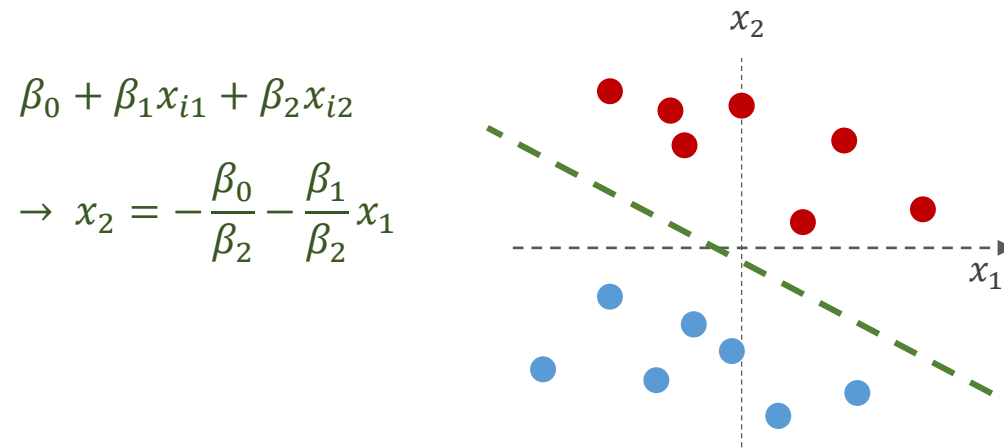
$$\rightarrow x = -\frac{\beta_0}{\beta_1}$$



Logistic Regression

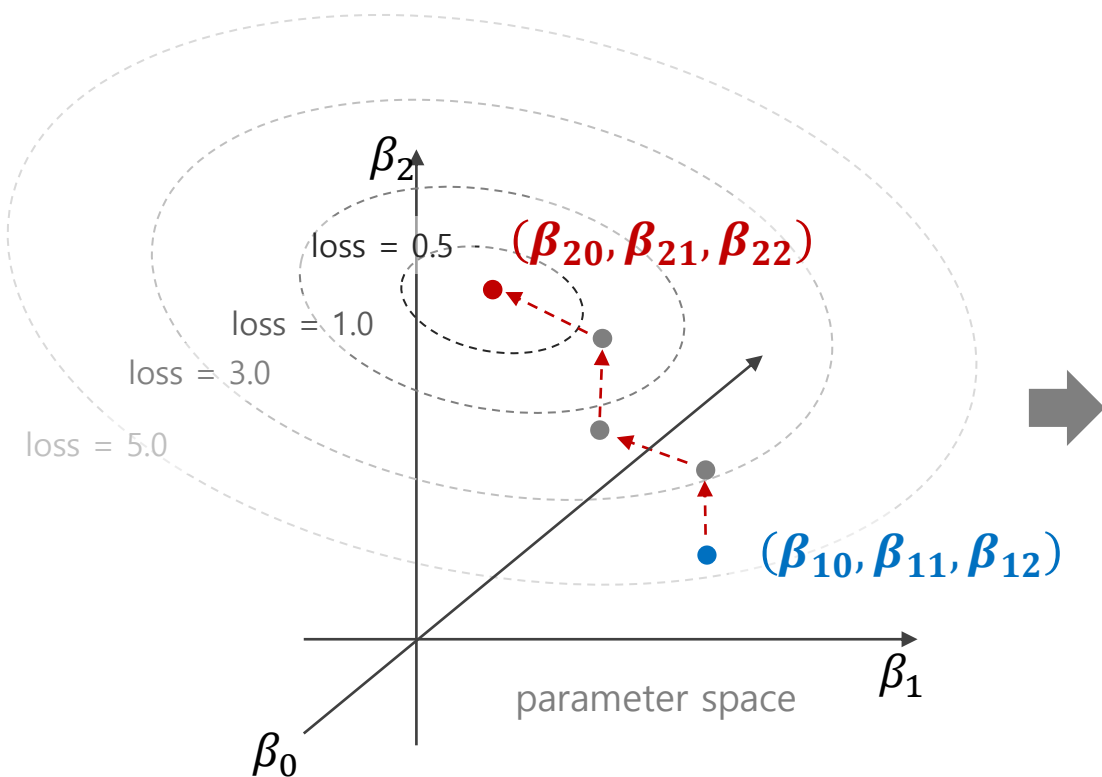
- Logistic regression 의 식은 선형 경계면을 의미합니다.

- $$\frac{1}{1+e^{-(\beta_0+\beta_1x_{i1}+\beta_2x_{i2})}} = P(y_i = y)$$

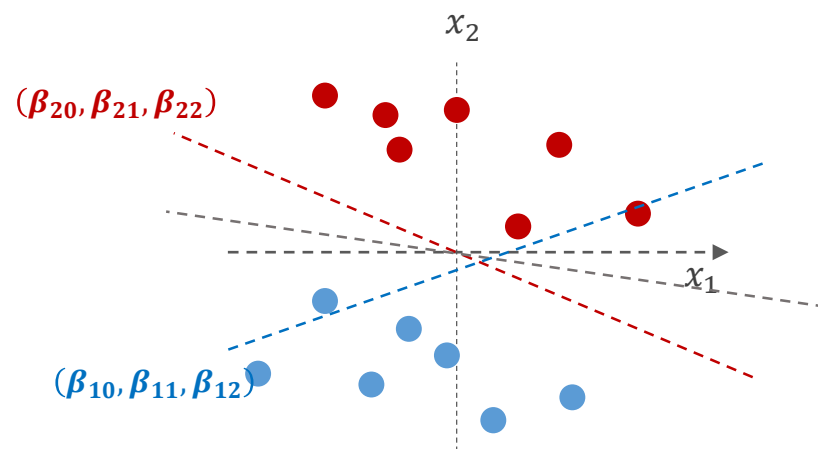


Logistic Regression

- 패러미터의 학습은, 두 클래스를 잘 구분하는 경계면의 탐색 과정입니다.



$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}$$



Logistic Regression

- Logistic regression 의 β 를 두 개로 분리할 수 있습니다.

p

$$\begin{aligned} & \frac{1}{1 + e^{(-x^T \beta)}} \\ &= \frac{e^{(x^T \beta)}}{e^{(x^T \beta)} + 1} \\ &= \frac{e^{(x^T \beta_1 - x^T \beta_2)}}{e^{(x^T \beta_1 - x^T \beta_2)} + 1} \\ &= \frac{e^{(x^T \beta_1)}}{e^{(x^T \beta_1)} + e^{(x^T \beta_2)}} \end{aligned}$$

$1 - p$

$$\begin{aligned} & \frac{e^{(-x^T \beta)}}{1 + e^{(-x^T \beta)}} \\ &= \frac{1}{e^{(x^T \beta)} + 1} \\ &= \frac{1}{e^{(x^T \beta_1 - x^T \beta_2)} + 1} \\ &= \frac{e^{(x^T \beta_2)}}{e^{(x^T \beta_1)} + e^{(x^T \beta_2)}} \end{aligned}$$

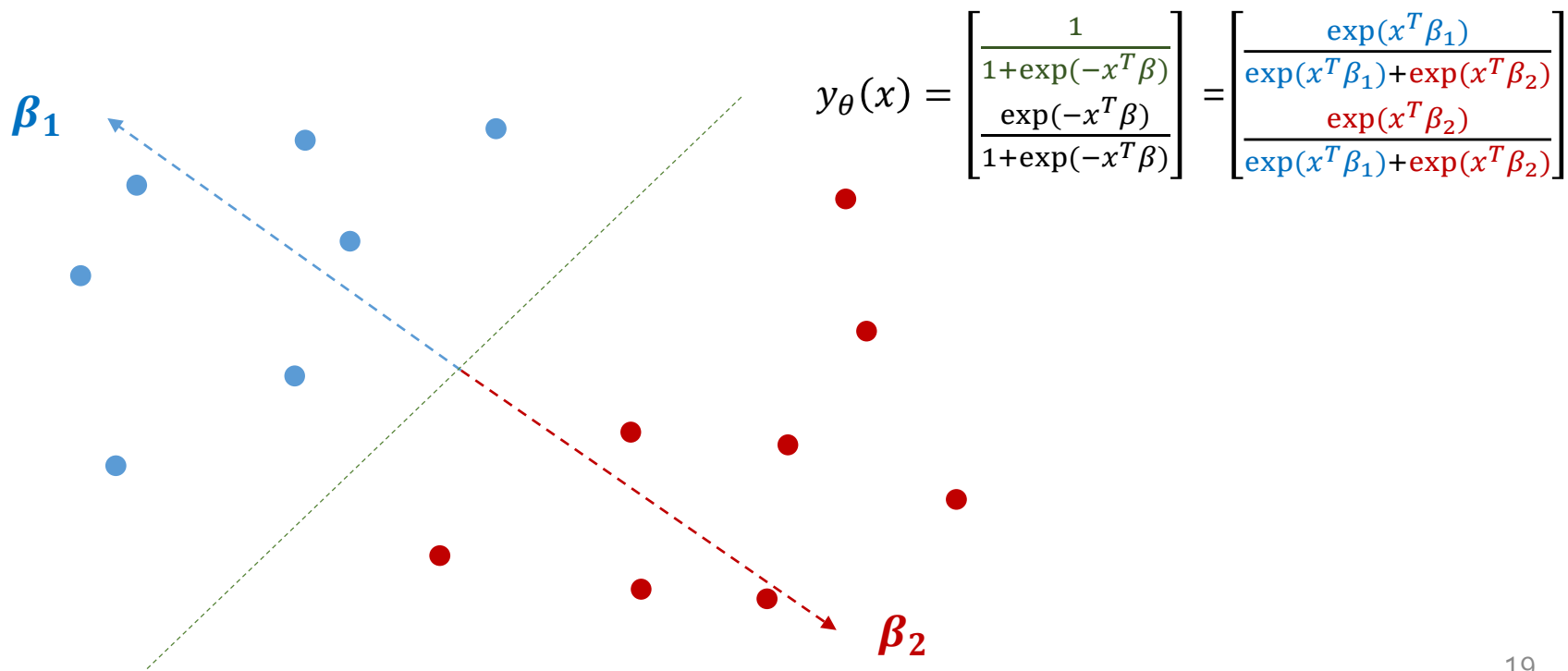
$$e^a \times e^b = e^{ab}$$

$$\beta := \beta_1 - \beta_2$$

$$\times e^{(x^T \beta_2)}$$

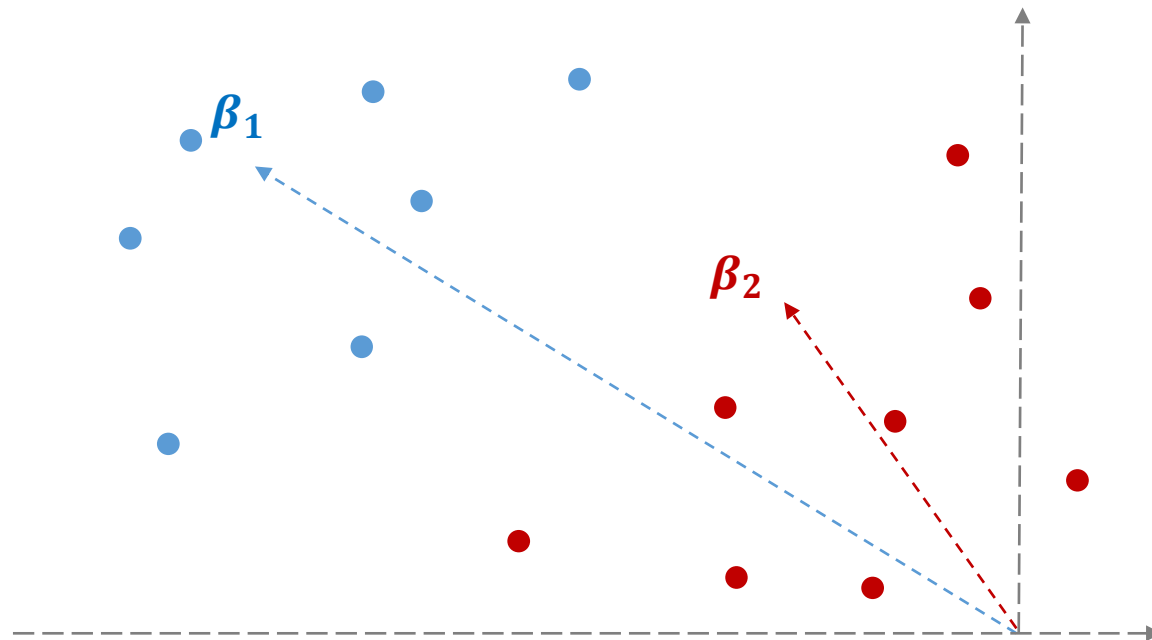
Logistic Regression

- Softmax regression 표현하면 $\beta = \beta_1 - \beta_2$ 이며, β_i 는 클래스 i 의 대표 벡터로 해석할 수 있습니다
 - Cosine measure 처럼 β_i 의 방향이 중요합니다



Logistic Regression

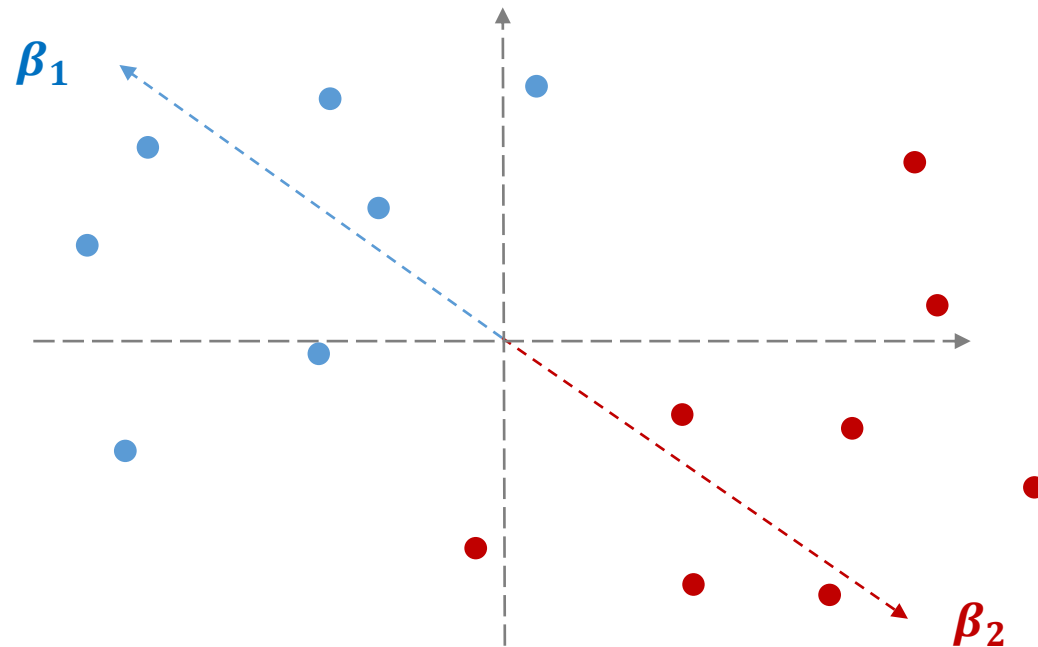
- 두 클래스의 데이터가 같은 방향에 있으면 β^i 만으로는 두 클래스를 구분하기 어려울 수도 있습니다



Logistic Regression

- Bias term은 클래스를 잘 구분하도록 x 를 평행이동 시킵니다

$$\exp(x^T \beta) = \exp(\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p) = \exp(\beta_1(x_1 - k_1) + \dots + \beta_p(x_p - k_p))$$



Logistic Regression

- Bias term 을 학습하는 것은 입력변수의 차원을 +1 하는 것과 같습니다.

$$\exp(x^T \beta) = \exp(\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p) = \exp(\beta_1(x_1 - k_1) + \dots + \beta_p(x_p - k_p))$$
$$(x_1, x_2, \dots, x_p) \rightarrow (1, x_1, x_2, \dots, x_p)$$

Softmax Regression

- Softmax regression 은 Logistic regression의 multi class 버전입니다.

$$h_{\beta}(x) = \begin{bmatrix} P(y = 1|x; \beta) \\ \vdots \\ P(y = K|x; \beta) \end{bmatrix} = \frac{1}{\sum_{j=1}^K \exp(x^T \beta_j)} \exp \left(\begin{bmatrix} x^T \beta_1 \\ \vdots \\ x^T \beta_K \end{bmatrix} \right)$$

$$P(y = 1 | x) = \frac{e^{(x^T \beta_1)}}{e^{(x^T \beta_1)} + e^{(x^T \beta_2)} + e^{(x^T \beta_3)}}$$

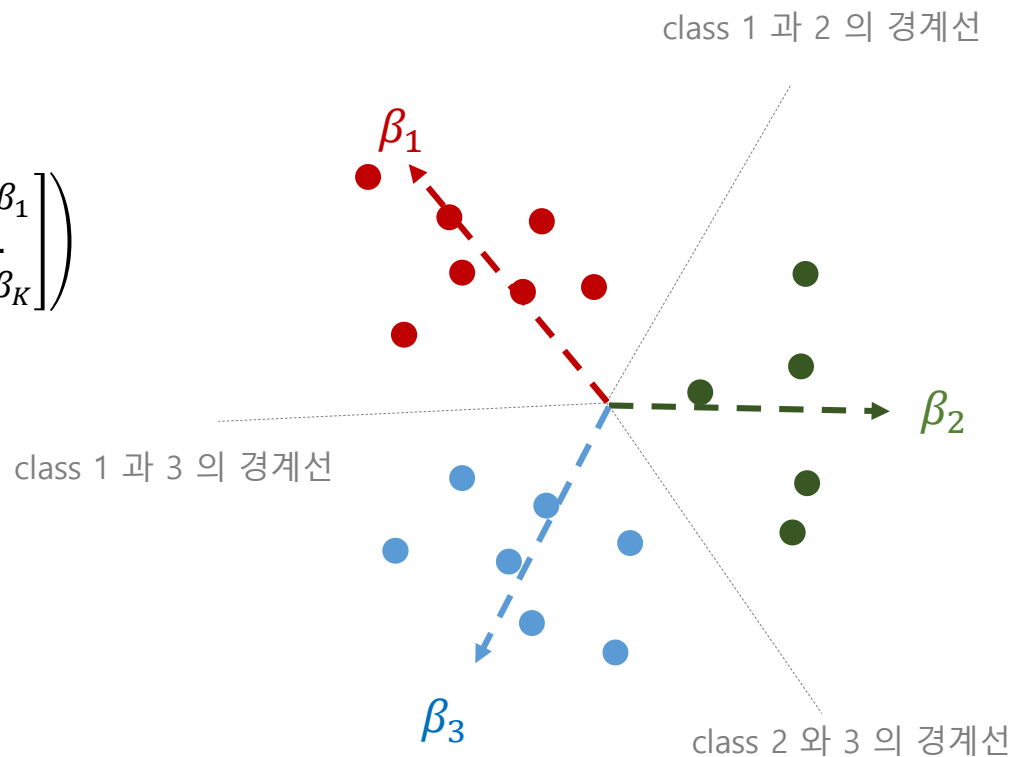
$$P(y = 2 | x) = \frac{e^{(x^T \beta_2)}}{e^{(x^T \beta_1)} + e^{(x^T \beta_2)} + e^{(x^T \beta_3)}}$$

$$P(y = 3 | x) = \frac{e^{(x^T \beta_3)}}{e^{(x^T \beta_1)} + e^{(x^T \beta_2)} + e^{(x^T \beta_3)}}$$

Softmax Regression

- Softmax 는 데이터 공간을 클래스 개수의 영역으로 나눕니다.

$$h_{\beta}(x) = \begin{bmatrix} P(y = 1|x; \beta) \\ \vdots \\ P(y = K|x; \beta) \end{bmatrix} = \frac{1}{\sum_{j=1}^K \exp(x^T \beta_j)} \exp \left(\begin{bmatrix} x^T \beta_1 \\ \vdots \\ x^T \beta_K \end{bmatrix} \right)$$



Parameter learning using MLE

- MLE 를 이용하여 $\prod_{i=1}^n P(y_i|x_i)$ 을 최대화하는 $(\beta_1, \dots, \beta_K)$ 를 탐색합니다.

$I(y_i = j)$: indicator function

$y_i = j$ 일 때에만 1, 아니면 0

$$\text{Likelihood} : \prod_i^n I(y_i = j) \boxed{\frac{\exp(x_i^T \beta_j)}{\sum_j^K \exp(x_i^T \beta_j)}} \leftarrow \pi_j \quad x_i \text{ 가 } y_i \text{ 일 확률의 누적값}$$

$$\text{Log-likelihood} : \sum_{i=1}^n \sum_{j=1}^K I(y_i = j) \log \frac{\exp(x_i^T \beta_j)}{\sum_j^K \exp(x_i^T \beta_j)}$$

NLL loss

$$\text{Negative LL} : - \sum_{i=1}^n \sum_{j=1}^K I(y_i = j) \left(x_i^T \beta_j - \log \sum_j^K \exp(x_i^T \beta_j) \right)$$

Parameter learning using MLE

- MLE 를 이용하여 $\prod_{i=1}^n P(y_i|x_i)$ 을 최대화하는 $(\beta_1, \dots, \beta_K)$ 를 탐색합니다.

$$\text{NLL} : \sum_{i=1}^n \sum_{j=1}^K I(y_i = j) \underbrace{\left(-x_i^T \beta_j + \log \sum_{j=1}^K \exp(x_i^T \beta_j) \right)}_{l(x_i, y_i)}$$

$$\frac{\partial l(x_i, y_i)}{\partial \beta_{jq}} = x_{iq} \frac{\exp(x_i^T \beta_j)}{\sum_{j=1}^K \exp(x_i^T \beta_j)} - x_{iq} = x_{iq}(\pi_i - 1)$$

Parameters

$\beta_{10}, \beta_{11}, \beta_{12}, \dots, \beta_{1p}$

$\beta_{20}, \beta_{21}, \beta_{22}, \dots, \beta_{2p}$

$\beta_{j0}, \beta_{j1}, \beta_{jq}, \dots, \beta_{jp}$

...

$\beta_{k0}, \beta_{k1}, \beta_{k2}, \dots, \beta_{kp}$

$$(ax)' = a$$

$$(\log x)' = \frac{1}{x}$$

$$(\log f(x))' = \frac{f(x)'}{f(x)}$$

$$(\log g(f(x)))' = \frac{f(x)' g(f(x))}{g(f(x))}$$

$\pi_i - 1 < 0$ 이므로 β_{jq} 가 x_{iq} 방향으로 움직일수록 loss 가 작아집니다.

($x_{iq} > 0$ 이면 β_{jq} 는 커지는 방향으로, $x_{iq} < 0$ 이면 β_{jq} 는 작아지는 방향으로)

Parameter learning using MLE

- MLE 를 이용하여 $\prod_{i=1}^n P(y_i|x_i)$ 을 최대화하는 $(\beta_1, \dots, \beta_K)$ 를 탐색합니다.

$$\text{NLL} : \sum_{i=1}^n \sum_{j=1}^K I(y_i = j) \underbrace{\left(-x_i^T \beta_j + \log \sum_{j'}^K \exp(x_i^T \beta_{j'}) \right)}_{l(x_i, y_i)}$$

$$\frac{\partial l(x_i, y_i)}{\partial \beta_{j'q}} = x_{iq} \frac{\exp(x_i^T \beta_j)}{\sum_{j'}^K \exp(x_i^T \beta_{j'})} - 0 = x_{iq} \pi_i$$

Parameters

$\beta_{10}, \beta_{11}, \beta_{12}, \dots, \beta_{1p}$

$\beta_{20}, \beta_{21}, \beta_{22}, \dots, \beta_{2p}$

$\beta_{j0}, \beta_{j1}, \beta_{jq}, \dots, \beta_{jp}$

...

$\beta_{k0}, \beta_{k1}, \beta_{k2}, \dots, \beta_{kp}$

β_j 가 아닌 다른 $\beta_{j'}$ 들은 x_i 의 반대 방향으로 이동해야 loss 가 감소합니다.
 β_j 와 $\beta_{j'}$ 는 서로 다른 방향으로 멀어집니다.

Parameter learning using MLE

- Logistic Regression loss 를 다음처럼 기술하기도 합니다.
 - Softmax 로 동일하게 해석할 수 있습니다.
 - x_i 가 y_i 에 속할 확률이 크려면 β_{y_i} 와의 내적이 커야 합니다.

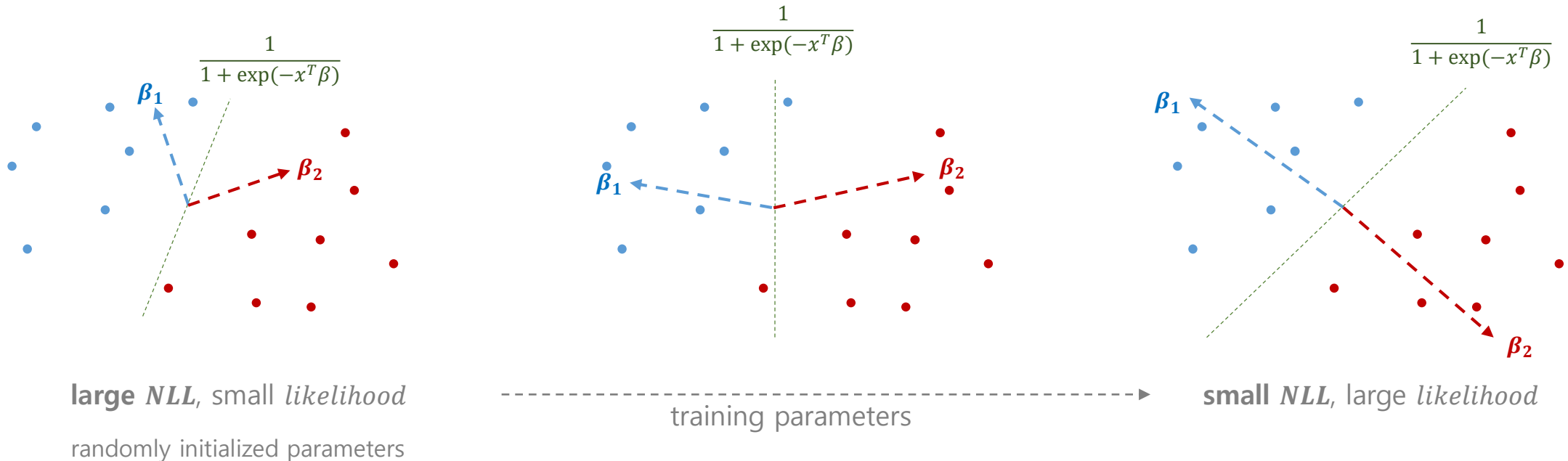
$$\begin{aligned} J(\beta) &= - \left[\sum_{i=1}^n y_i \log f(x_i) + (1 - y_i) \log(1 - f(x_i)) \right] \\ &= - \left[\sum_{i=1}^n y_i \log f(x_i) + (1 - y_i) \log g(x_i) \right] \end{aligned}$$

$$\begin{aligned} f(x) &= \frac{e^{(x^T \beta_1)}}{e^{(x^T \beta_1)} + e^{(x^T \beta_2)}} \\ g(x) &= \frac{e^{(x^T \beta_2)}}{e^{(x^T \beta_1)} + e^{(x^T \beta_2)}} \end{aligned}$$

Parameter learning using MLE

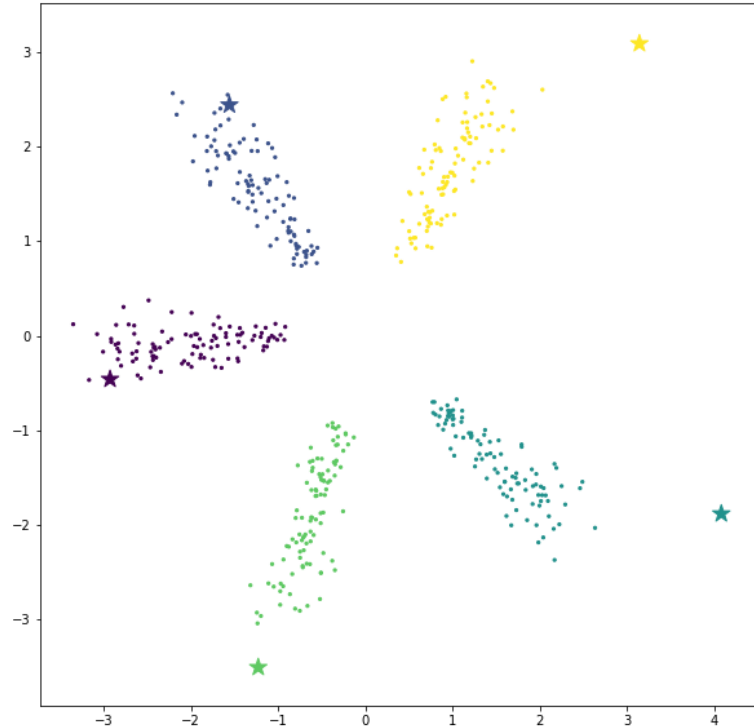
- 각 클래스를 대표하는 벡터로 β 를 학습합니다.

$$y_{\theta}(x) = \begin{bmatrix} \frac{1}{1 + \exp(-x^T \beta)} \\ \frac{\exp(-x^T \beta)}{1 + \exp(-x^T \beta)} \end{bmatrix} = \begin{bmatrix} \frac{\exp(x^T \beta_1)}{\exp(x^T \beta_1) + \exp(x^T \beta_2)} \\ \frac{\exp(x^T \beta_2)}{\exp(x^T \beta_1) + \exp(x^T \beta_2)} \end{bmatrix}$$



Softmax Regression

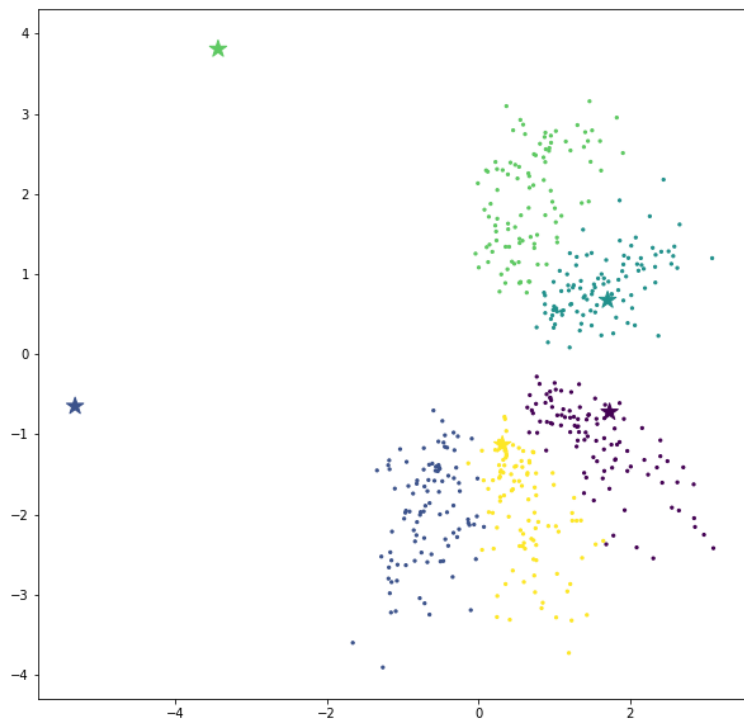
- 클래스별로 데이터분포가 고르면 대표 벡터는 데이터분포와 비슷합니다.



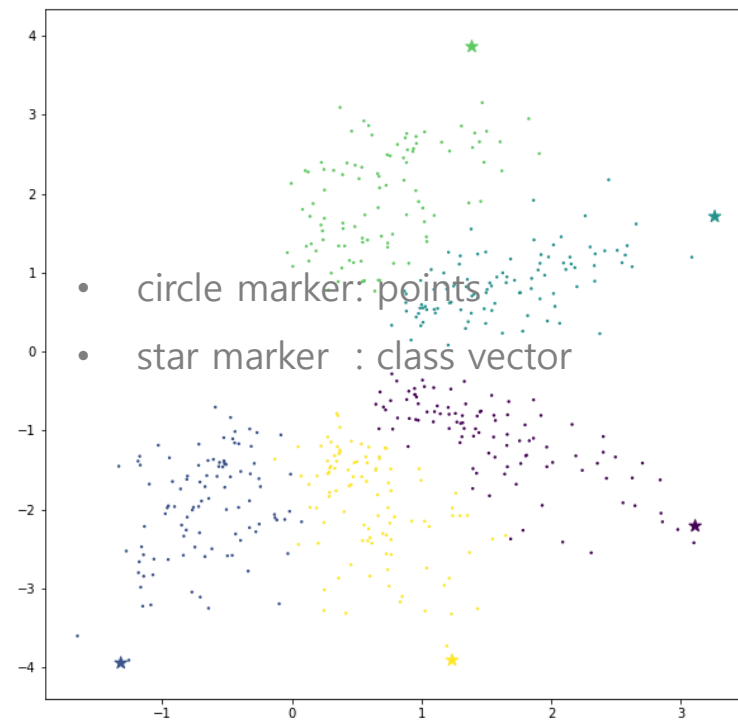
- circle marker: points
- star marker : class vector

Softmax Regression

- 대표 벡터는 각 점이 해당 클래스에 속할 확률이 가장 크도록 학습됩니다.
- 데이터의 분포가 한쪽에 치우쳐지면 대표 벡터는 데이터분포와 다를 수 있습니다



class vector of trained model



2 x class mean vector

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

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

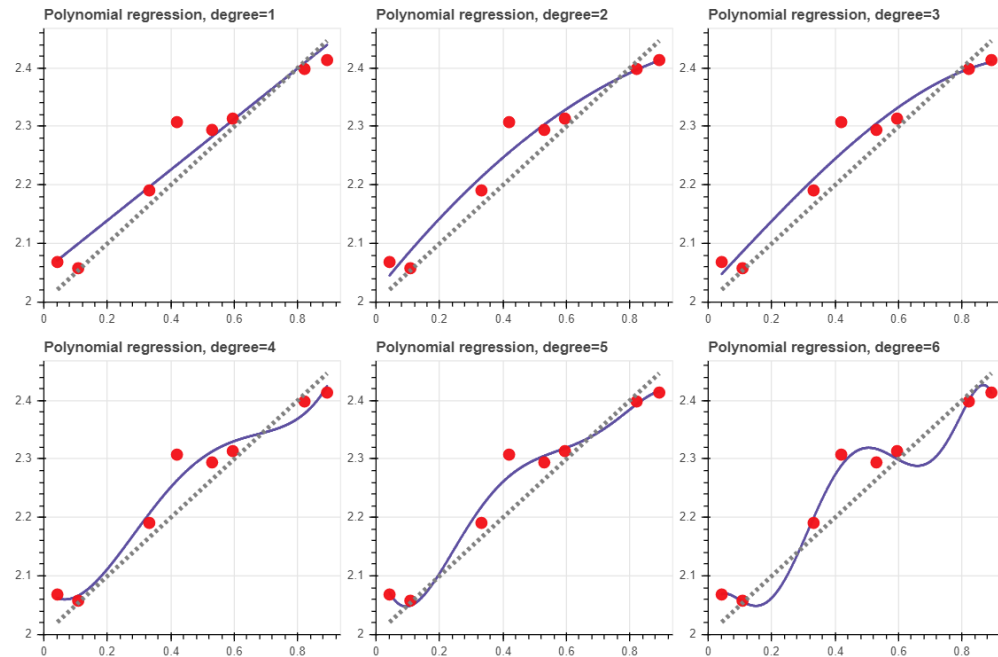
- $|(3, 0, -4)|_{p=1} = \sqrt[1]{|3|^1 + |0|^1 + |-4|^1} = 7$

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

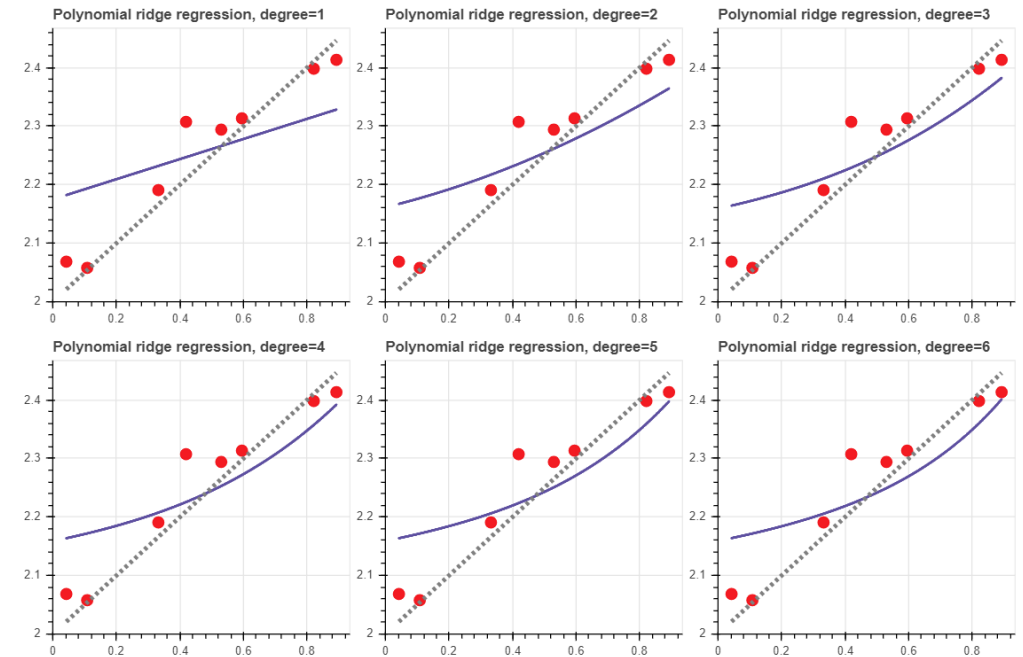
Regularization

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

- L2 regularization 을 이용하는 ridge regression 은 회귀선의 모양을 부드럽게 만듭니다.



polynomial linear regression



polynomial ridge regression

Regularization

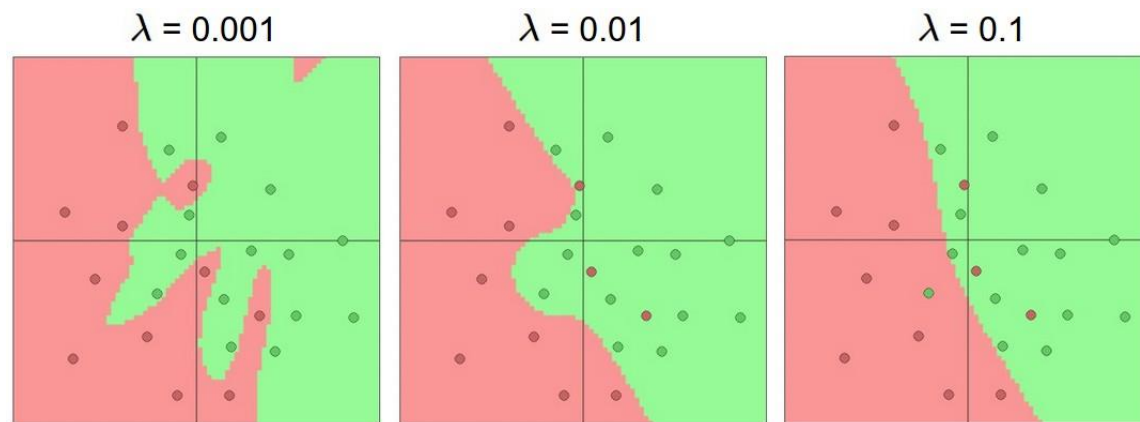
- Logistic regression 의 regularization 도 과적합 방지의 효과가 있습니다.

$$\text{L1 cost} = - \left[\sum_{i=1}^m \sum_{k=1}^K I(y_i = k) \log \frac{\exp(x_i^T \beta_k)}{\sum_{j=1}^K \exp(x_i^T \beta_j)} \right] + \lambda \sum_j^p |\beta_j|_1$$

$$\text{L2 cost} = - \left[\sum_{i=1}^m \sum_{k=1}^K I(y_i = k) \log \frac{\exp(x_i^T \beta_k)}{\sum_{j=1}^K \exp(x_i^T \beta_j)} \right] + \lambda \sum_j^p |\beta_j|_2^2$$

L2 Regularization

- L2 regularization 은 경계면을 복잡하지 않게 만듭니다
 - β_{ij} 중에서 크기가 유독 큰 값이 없도록 만들기 때문입니다.



Feed forward neural network의 예시이지만, Logistic Regression 역시 동일한 모습을 보입니다

(출처) <http://cs231n.github.io/neural-networks-1/>

L2 Regularization

- λ 는 복잡한 경계면과 예측력 간의 영향도를 조절합니다.

$$\text{L2 cost} = - \sum_{i=1}^m \sum_{k=1}^K I(y_i = k) \log \frac{\exp(x_i^T \beta_k)}{\sum_{j=1}^K \exp(x_i^T \beta_j)} + \boxed{\lambda} \sum_j^p |\beta_j|_2^2$$

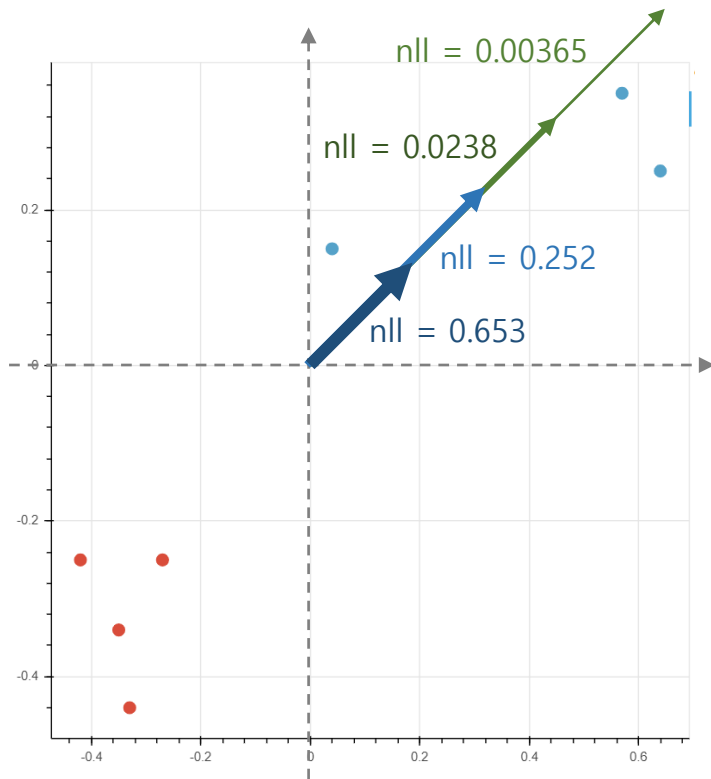
x 를 이용하여 y 를 얼마나 잘 예측하는가?

결정단면이 얼마나 복잡한가?

복잡한 결정단면을 이용하면 예측력이 올라가는가?

L2 Regularization

- Softmax regression 의 β 는 norm 이 커지는 방향으로 과적합합니다.



$(\beta_0, \beta_1, \beta_2)$	NLL
$(0, 0.1, 0.1)$	0.653
$(0, 2.0, 2.0)$	0.252
$(0, 10, 10)$	0.0238
$(0, 20, 20)$	0.00365

L2 Regularization

- Softmax regression 의 β 는 norm 이 커지는 방향으로 과적합합니다.

$$\text{NLL} : \sum_{i=1}^n \sum_{j=1}^K I(y_i = j) - \left(x_i^T \beta_j - \log \sum_{j=1}^K \exp(x_i^T \beta_j) \right)$$

$$\frac{\partial l(x_i, y_i)}{\partial \beta_{jq}} = x_{iq} \frac{\exp(x_i^T \beta_j)}{\sum_{j=1}^K \exp(x_i^T \beta_j)} - x_{iq} = \frac{x_{iq}(\pi_i - 1)}{1}$$

$$\frac{\partial l(x_i, y_i)}{\partial \beta_{j'q}} = x_{iq} \frac{\exp(x_i^T \beta_{j'})}{\sum_{j=1}^K \exp(x_i^T \beta_j)} - 0 = \frac{x_{iq}\pi_i}{1}$$

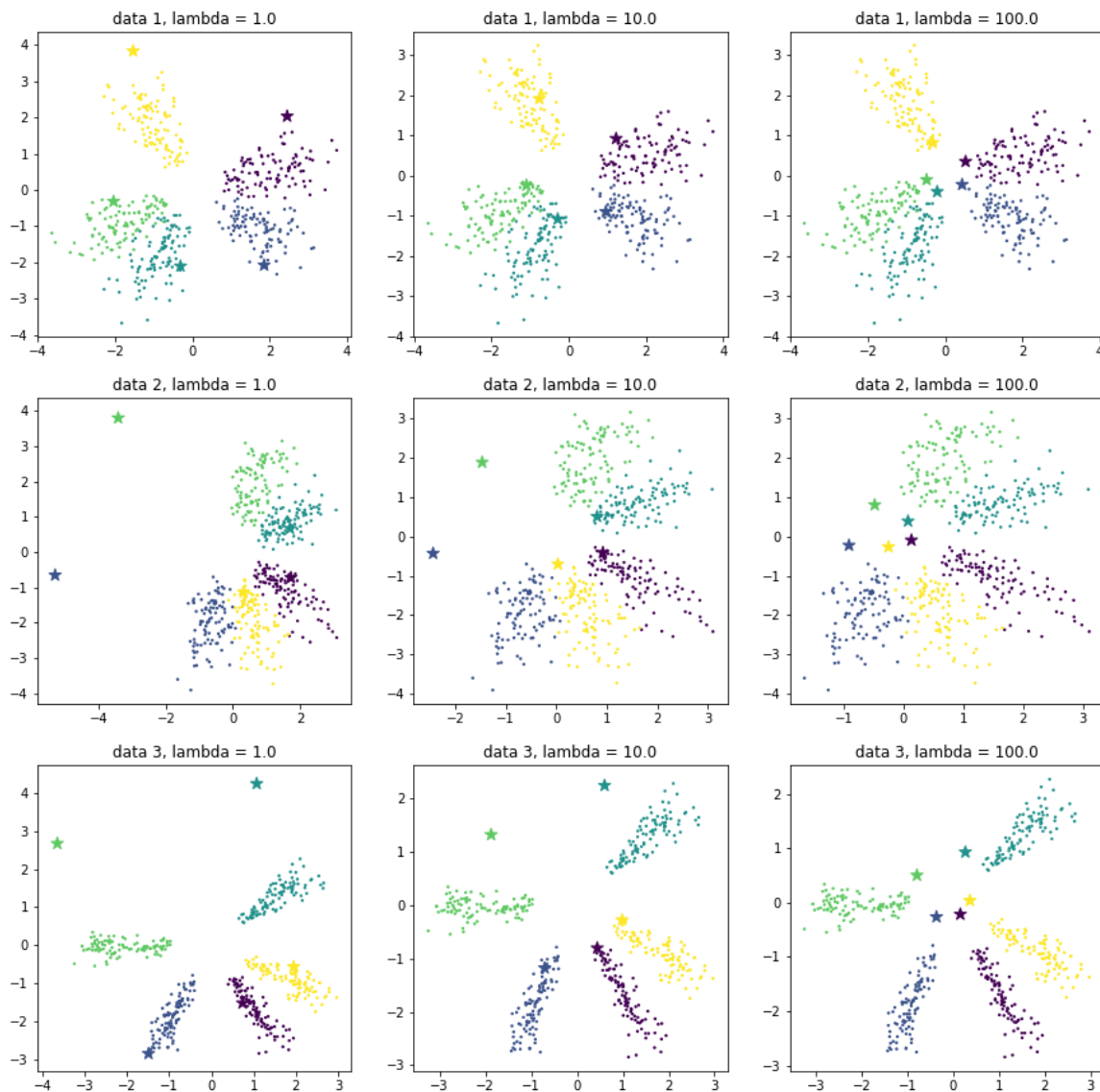
π_i 는 0, 1이 될 수 없기 때문에 iteration 이 진행될때마다 β_j 를 더 크게 업데이트 합니다.

L2 Regularization

- Logistic regression 의 regularization 도 과적합 방지의 효과가 있습니다.
 - β 의 크기가 지나치게 커지지 않도록 β 크기에 제약을 가합니다.
 - 대신 λ 가 지나치게 크면 β 는 0 에 가까워집니다.
(적절한 값을 선택해야 합니다)

$$\text{L2 cost} = - \left[\sum_{i=1}^m \sum_{k=1}^K I(y_i = k) \log \frac{\exp(x_i^T \beta_k)}{\sum_{j=1}^K \exp(x_i^T \beta_j)} \right] + \lambda \sum_j^p |\beta_j|_2^2$$

L2 Regularization



$$\text{L2 cost} = - \sum_{i=1}^m \sum_{k=1}^K I(y_i = k) \log \frac{\exp(x_i^T \beta_k)}{\sum_{j=1}^K \exp(x_i^T \beta_j)} + \lambda \sum_j^p |\beta_j|_2^2$$

λ 가 클수록 β 가 원점 주변으로 모입니다
(L2 norm 이 작아집니다)

L1 Regularization

- L1 regularization 중요한 변수를 선택하는 효과가 있습니다.
- β_j 의 많은 값이 0 이 되기 때문에 sparse modeling 이라 부릅니다.

$$\text{L1 cost} = - \sum_{i=1}^m \sum_{k=1}^K I(y_i = k) \log \frac{\exp(x_i^T \beta_k)}{\sum_{j=1}^K \exp(x_i^T \beta_j)} + \lambda \sum_j^p |\beta_j|_1$$

x 를 이용하여 y 를 얼마나 잘 예측하는가?

모델이 몇 개의 변수를 이용하는가?

변수를 더 많이 이용하면 예측력이 올라가는가?

L1 Regularization

- L1 은 L0 과 L2 의 특징을 모두 지니고 있습니다
 - L0 처럼 소수의 변수를 선택합니다
 - L2 처럼 미분을 통하여 패러미터의 학습이 가능합니다.
 - L0 는 미분이 되지 않으며, 조합최적화 해법을 이용하여 해를 찾아야 합니다
(NP-hard problem)

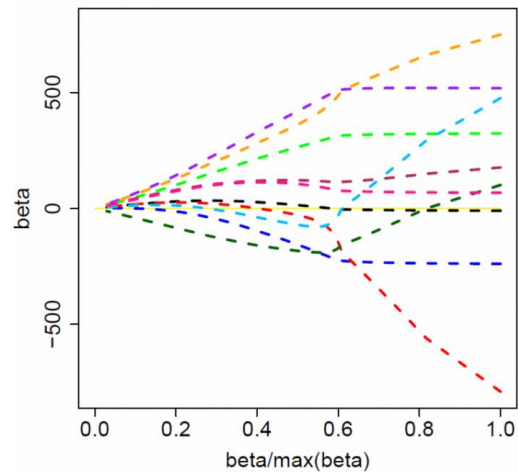
L1 Regularization

- L1 regularization 을 이용한 방법을 LASSO라 부르며, Lasso regression 은 중요한 변수를 데이터 기반으로 추출합니다
 - LASSO 는 성능을 저하하지 않으면서 적은 변수를 이용하려 합니다.
- λ 의 크기에 따라서 모델이 선택하는 변수의 개수와 종류가 달라집니다
 - λ 값이 작을수록 (regularization을 덜 할수록) 더 많은 변수를 사용

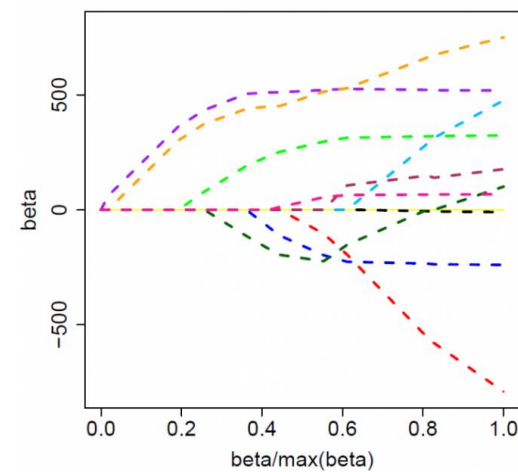
$$\text{cost} = - \sum_{i=1}^m \sum_{k=1}^K I(y_i = k) \log \frac{\exp(x_i^T \beta_k)}{\sum_{j=1}^K \exp(x_i^T \beta_j)} + \lambda \sum_j^p |\beta_j|_1$$

L1 Regularization

- LASSO path는 λ 에 따른 β 의 변화를 시각적으로 표현합니다
 - L2 은 λ 가 작을수록 ($\beta/\max(\beta)$ 가 커짐에 따라) 여러 β 의 값이 증가합니다
 - L1 은 step function 처럼 λ 가 어느 정도 작아져야 새로운 변수가 이용됩니다
(해당 변수의 β 의 크기가 0이 아니게 됩니다)



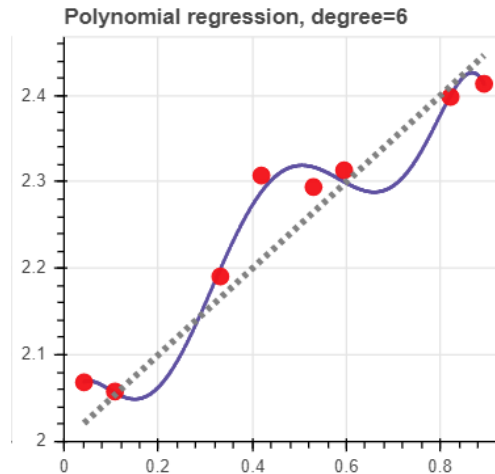
coef. path of L2 model



coef. path of LASSO model

L1 Regularization

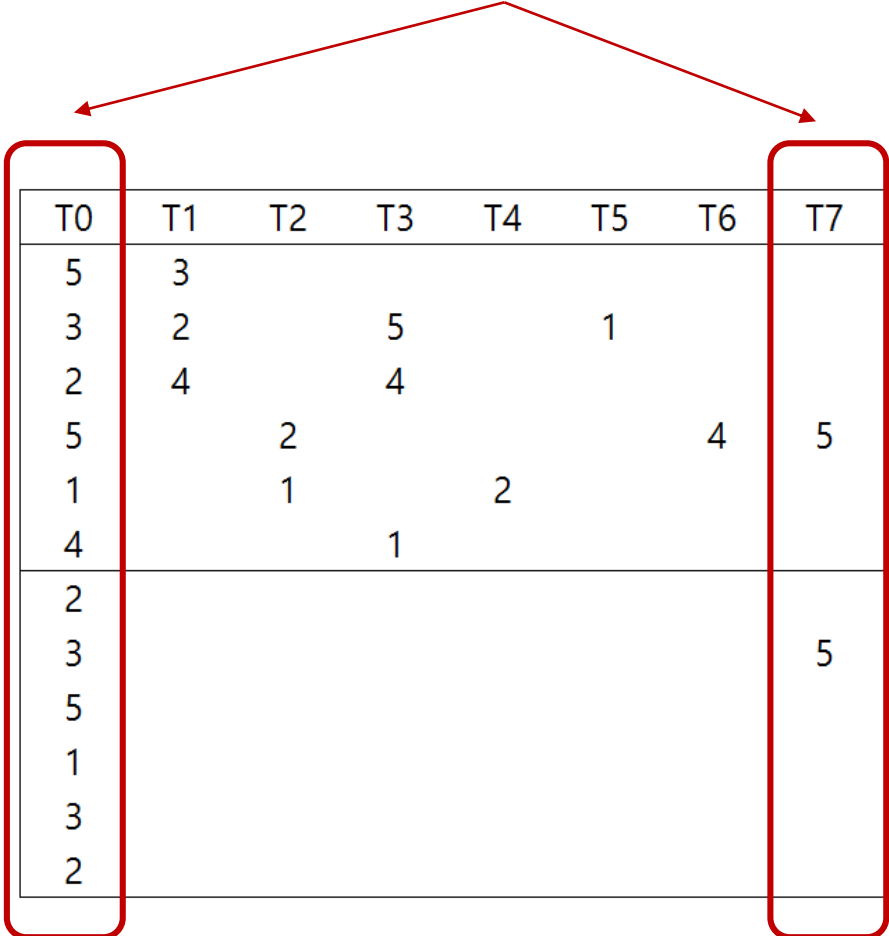
- 데이터의 개수 n 에 비하여 변수의 개수 p 가 큰 데이터를 fat data 라 하며, 모든 변수를 이용하면 과적합이 발생할 가능성이 높습니다.
 - 머신러닝의 판별 모델은 "반복된 데이터"로부터 패턴을 학습합니다.
 - 모든 변수를 이용하면 데이터가 반복되지 않을 가능성이 높습니다.



$n = 8, p = 7$ 인 linear regression

L1 Regularization

classification 에 전혀 도움이 되지 않는 변수들



y	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
0	5	3													
0	3	2		5		1			2						
0	2	4		4											
0	5		2				4	5	3						
0	1		1		2										
0	4			1											
1	2								2					2	
1	3							5					4	4	
1	5								1	1		3			
1	1								2			2			3
1	3								4		1		2	1	1
1	2								4				1		2

L1 Regularization

T1, T2만 이용하여도 $y=0$ 의 5/6을 인식할 수 있음

y	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
0	5	3													
0	3	2		5		1			2						
0	2	4		4											
0	5		2				4	5	3						
0	1		1		2										
0	4			1											
1	2								2					2	
1	3							5					4	4	
1	5								1	1		3			
1	1								2			2			3
1	3								4		1		2	1	1
1	2								4				1		2

L1 Regularization

여유가 된다면 ($=\lambda$ 가 작다면, =classification 성능에 더 집중해도 된다면)
T3까지 이용하면 $y=0$ 을 완벽히 구분할 수 있음

y	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
0	5	3													
0	3	2		5		1			2						
0	2	4		4											
0	5		2				4	5	3						
0	1		1		2										
0	4			1											
1	2								2					2	
1	3							5					4	4	
1	5								1	1		3			
1	1								2			2			3
1	3								4		1		2	1	1
1	2								4				1		2

L1 Regularization

T8 은 $y=1$ 일 때 더 많이 등장하지만, 이를 이용하면 오분류의 가능성이 있음

y	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
0	5	3													
0	3	2		5		1			2						
0	2	4		4											
0	5		2				4	5	3						
0	1		1		2										
0	4			1											
1	2								2					2	
1	3							5					4	4	
1	5								1	1		3			
1	1								2			2			3
1	3								4		1		2	1	1
1	2								4				1		2

L1 Regularization

T11, T13, T14 를 이용하면 $y=1$ 을 완벽히 구분할 수 있고,
이 문제는 {T1, T2, T3, T11, T13, T14} 만 이용해도 잘 풀림

y	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
0	5	3													
0	3	2		5		1			2						
0	2	4		4											
0	5		2				4	5	3						
0	1		1		2										
0	4			1											
1	2								2					2	
1	3							5					4	4	
1	5								1	1		3			
1	1								2			2			3
1	3								4		1		2	1	1
1	2								4				1		2

Elastic Net

- L1, L2 regularization 을 모두 이용하는 모델을 Elastic Net 이라 합니다.

$$\text{cost} = - \sum_{i=1}^m \sum_{k=1}^K I(y_i = k) \log \frac{\exp(x_i^T \beta_k)}{\sum_{j=1}^K \exp(x_i^T \beta_j)} + \lambda_1 \sum_j^p |\beta_j|_1 + \lambda_2 \sum_j^p |\beta_j|_2^2$$

Evaluation

Evaluation measurement

- Precision : $\frac{|true_{pos} \cap pred_{pos}|}{|pred_{pos}|}$
- Recall : $\frac{|true_{pos} \cap pred_{pos}|}{|true_{pos}|}$
- F1 – measure : $2 \times \frac{precision \times recall}{precision + recall}$
- Accuracy : $\frac{|true_{pos} \cap true_{neg}|}{|pos+neg|}$

Confusion matrix

	Predict positive	Predict negative	sum
True positive	800	200	1000
True negative	400	600	1000
sum	1200	800	2000

Precision	$\frac{ true_{pos} \cap pred_{pos} }{ pred_{pos} } = \frac{800}{1200}$
Recall	$\frac{ true_{pos} \cap pred_{pos} }{ true_{pos} } = \frac{800}{1000}$
F1 measure	$2 \times \frac{precision \times recall}{precision + recall} = 2 \times \frac{\frac{8}{12} \times \frac{8}{10}}{\frac{8}{12} + \frac{8}{10}}$
Accuracy	$\frac{ true_{pos} \cap true_{neg} }{ pos + neg } = \frac{800 + 600}{2000}$

Confusion matrix

- Multiclass classification 으로 확장할 수 있습니다.

	predict A	predict B	predict C	sum
true A	500	0	0	500
true B	20	300	30	350
true C	80	20	200	300
sum	600	320	230	1150

Precision (A)	$\frac{ true_A \cap pred_A }{ pred_A } = \frac{500}{600}$
Recall (A)	$\frac{ true_A \cap pred_A }{ true_A } = \frac{500}{500}$
F1 measure (A)	$2 \times \frac{precision \times recall}{precision + recall} = 2 \times \frac{\frac{5}{6} \times 1}{\frac{5}{6} + 1}$
Accuracy	$\frac{500 + 300 + 200}{1150}$

Model fitness

- Logistic regression 은 잔차가 정의되지 않아 R^2 를 계산할 수 없습니다.
 - 선형회귀 모델의 $R^2 = 1 - \frac{\sum_i (e_i)^2}{\sum_i (y_i - \bar{y})^2}$ 는 모델이 설명하지 못하는 잔차의 비중입니다.
- 대안으로 모델의 likelihood 로부터 모델의 적합도를 측정할 수 있습니다.
 - 하지만, likelihood 는 학습데이터의 개수에 따라 scale 이 다릅니다.
 - R^2 와 비슷하게 모델의 품질을 평가한다는 의미로 pseudo R^2 라 부르는 방법들이 제안되었습니다 (여러 종류의 pseudo R^2 가 있습니다).

Model fitness (McFadden pseudo R^2)

- 잔차의 비중 대신, loss 의 비중을 이용하여 품질을 정의합니다.
 - Input variables 을 이용하지 않으면 클래스의 개수로 결과값을 예측할 수 있습니다.

(예시): {a:100, b:200} $\rightarrow P(y = a | x) = \frac{1}{3}$

- 학습이 잘 이뤄지면 각각의 확률값은 증가합니다. $\frac{\log(L_M)}{\log(L_0)} \rightarrow 0$ 에 가까워집니다.

$$P(y = a | x_a; M) \rightarrow 1, P(y = b | x_b; M) \rightarrow 1$$

$$R_{McF}^2 = 1 - \frac{\log(L_M)}{\log(L_0)}$$

L_0 : x 를 이용하지 않을 때의 likelihood

L_M : 학습한 모델의 likelihood

Model fitness (McFadden pseudo R^2)

- 학습이 잘 이뤄지면 각각의 확률값은 증가합니다. $\frac{\log(L_M)}{\log(L_0)} \rightarrow 0$ 에 가까워집니다.
 - L_M 의 최대값이 1 이기 때문에 $\log(L_M)$ 의 최대값은 0
 - $\log(L_0)$ 은 절대값이 큰 음수

$$R_{McF}^2 = 1 - \frac{\log(L_M)}{\log(L_0)}$$

L_0 : x 를 이용하지 않을 때의 likelihood
 L_M : 학습한 모델의 likelihood

