분류기 (Classifiers)

- Logistic Regression (Softmax Regression)
- Neural Networks
- LASSO Regression (regulization)

Logistic Regression

- **Logistic Regression (LR)은 대표적인 binary classification 알고리즘**
  - positive class에 속할 점수를 [0, 1] 사이로 표현하기 때문에 확률 모형처럼 이용

$$y_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)}$$

  - 학습 데이터 $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)}\}$ 가 주어졌을 때, loss function은

$$J(\theta) = -\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right)\right]$$

Logistic Regression

- **Softmax regression은 multi class classification 알고리즘으로, Logistic Regression은 Softmax regression의 특별한 경우**
  - Loss function에 대해서는 Word2Vec 때 다룰 예정

$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ \dots \\ P(y=K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp\left(\theta^{(j)T} x\right)} \exp\left(\begin{bmatrix} \theta^{(1)T} x \\ \dots \\ \theta^{(K)T} x \end{bmatrix} \right)$$

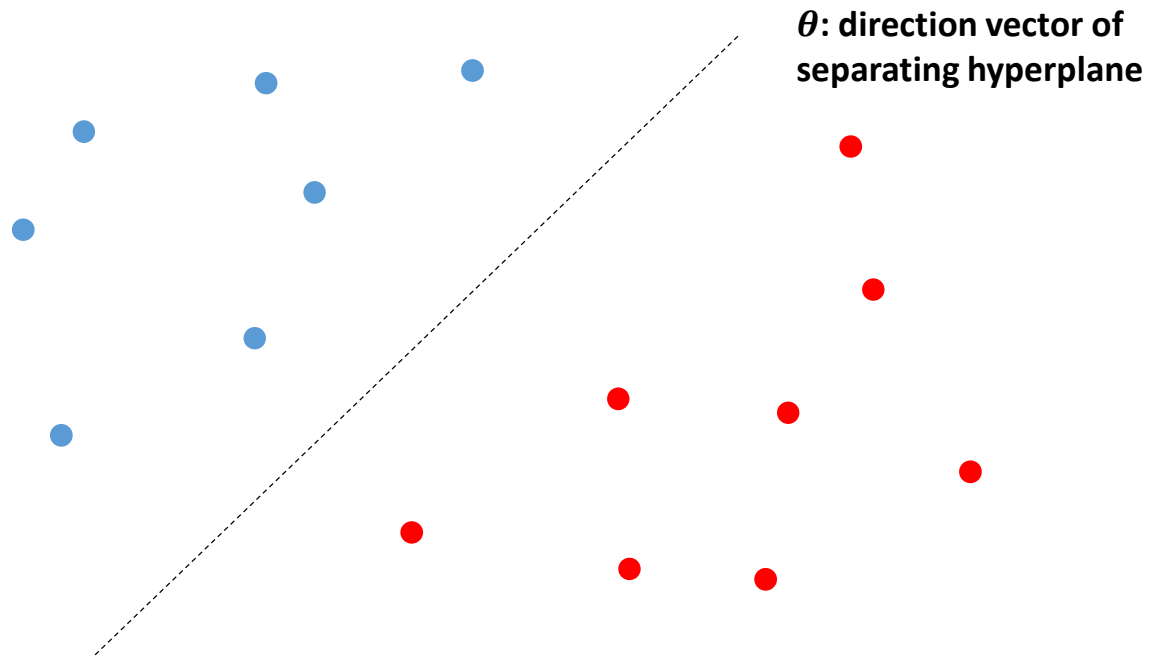
  - 학습 데이터 $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)}\}$ 가 주어졌을 때, loss function은

$$J(\theta) = -\left[\sum_{i=1}^{m} \sum_{k=1}^{K} 1\{y^{(i)} = k\} \log \frac{\exp\left(\theta^{(j)T} x^{(i)}\right)}{\sum_{j=1}^{K} \exp\left(\theta^{(j)T} x^{(i)}\right)} \right]$$

Logistic Regression

- 일반적으로 LR은 두 클래스의 경계면을 학습한다고 말함

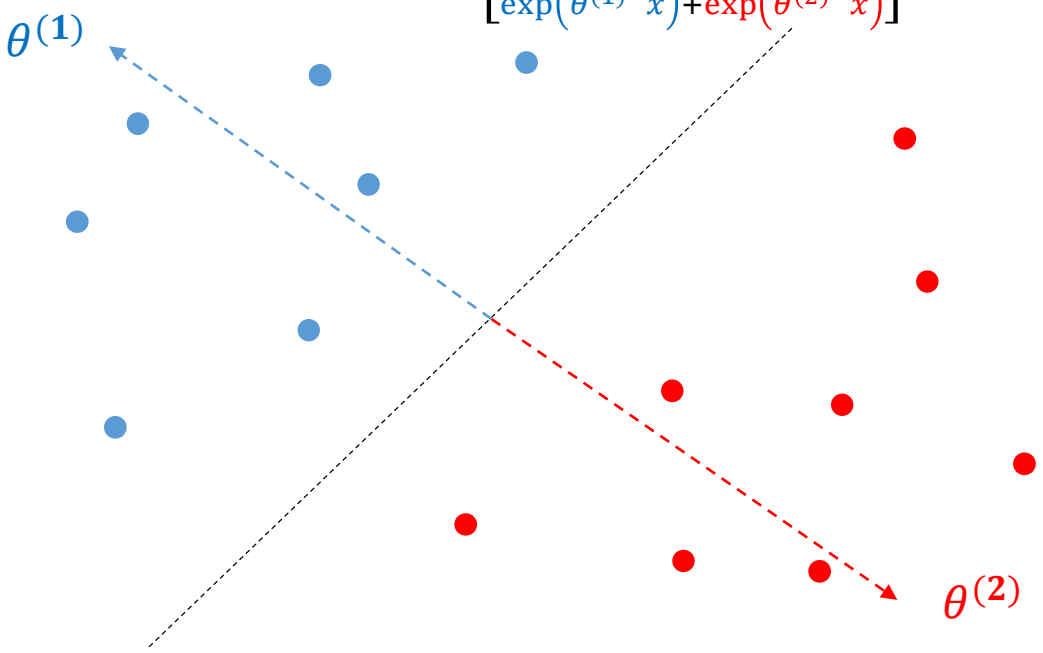
$$y_\theta(x) = \frac{1}{1 + \exp(\theta^T x)}$$



Logistic Regression

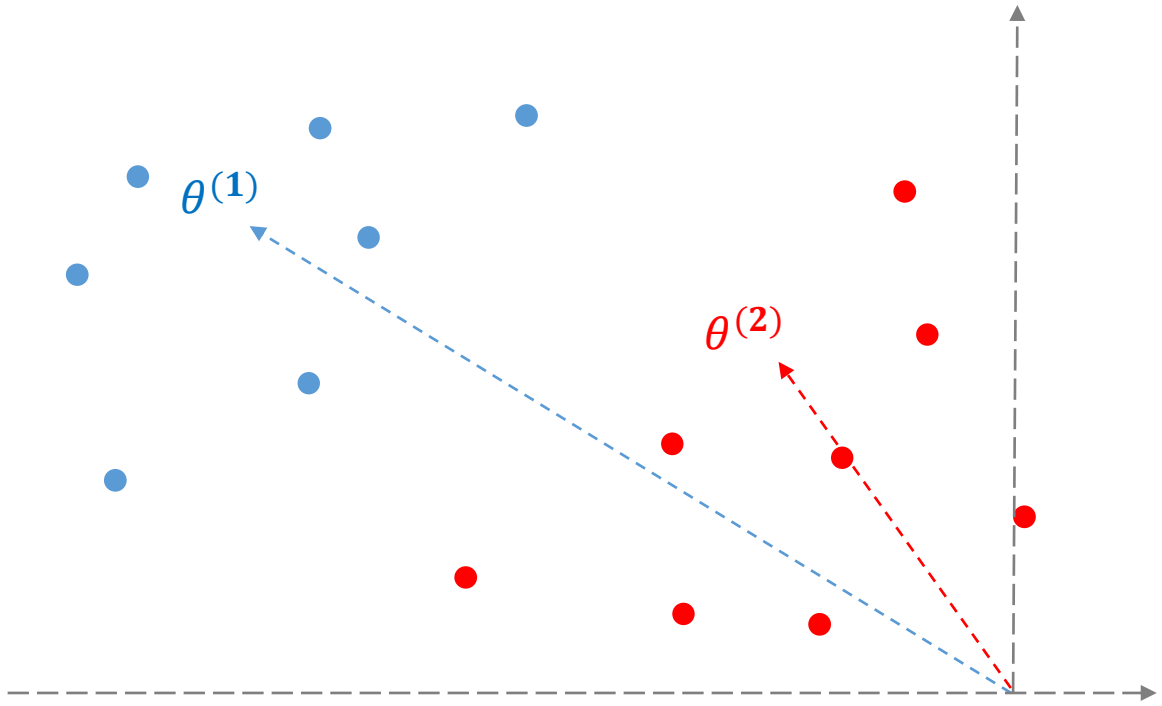
- 본래는 Softmax regression 형식으로 학습되며, $\theta = \theta^{(2)} - \theta^{(1)}$로 표현된 것
- $\theta^{(i)}$는 클래스 i의 대표 벡터로 해석할 수 있음 (Cosine similarity와 유사)

$$y_\theta(x) = \begin{bmatrix} \frac{1}{1+\exp(\theta^T x)} \\ \frac{\exp(\theta^T x)}{1+\exp(\theta^T x)} \end{bmatrix} = \begin{bmatrix} \frac{\exp(\theta^{(1)^T} x)}{\exp(\theta^{(1)^T} x)+\exp(\theta^{(2)^T} x)} \\ \frac{\exp(\theta^{(2)^T} x)}{\exp(\theta^{(1)^T} x)+\exp(\theta^{(2)^T} x)} \end{bmatrix}$$



Logistic Regression

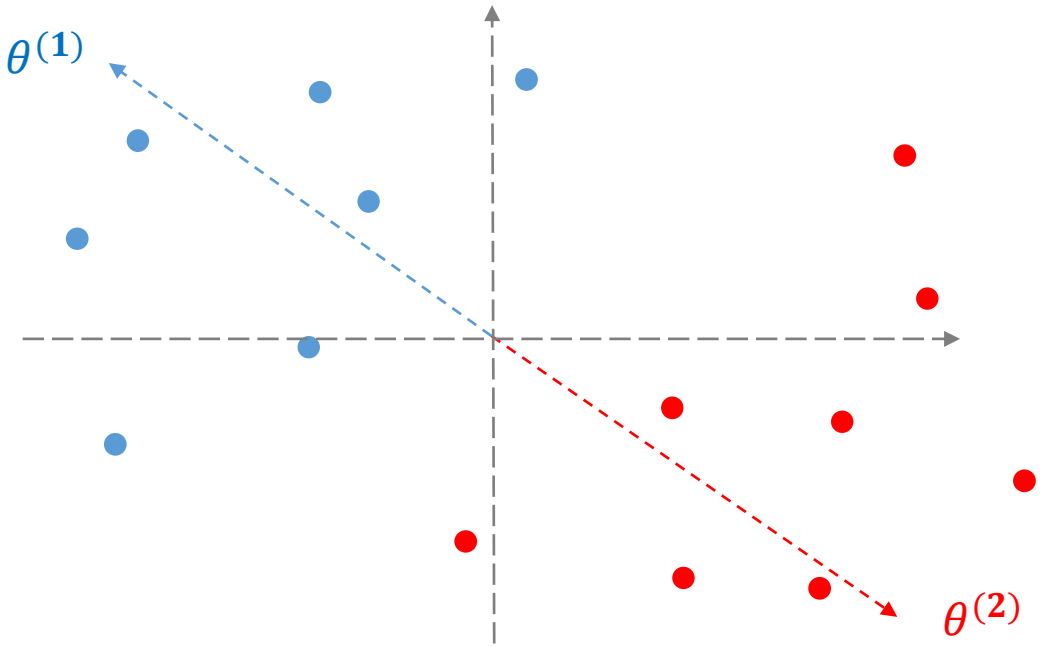
- **두 클래스의 데이터가 같은 방향에 존재할 경우 $\theta^{(i)}$ 만으로는 두 클래스를 구분하기 어려울 수도 있음**
  - 단어 빈도 벡터의 값은 모두 양수



Logistic Regression

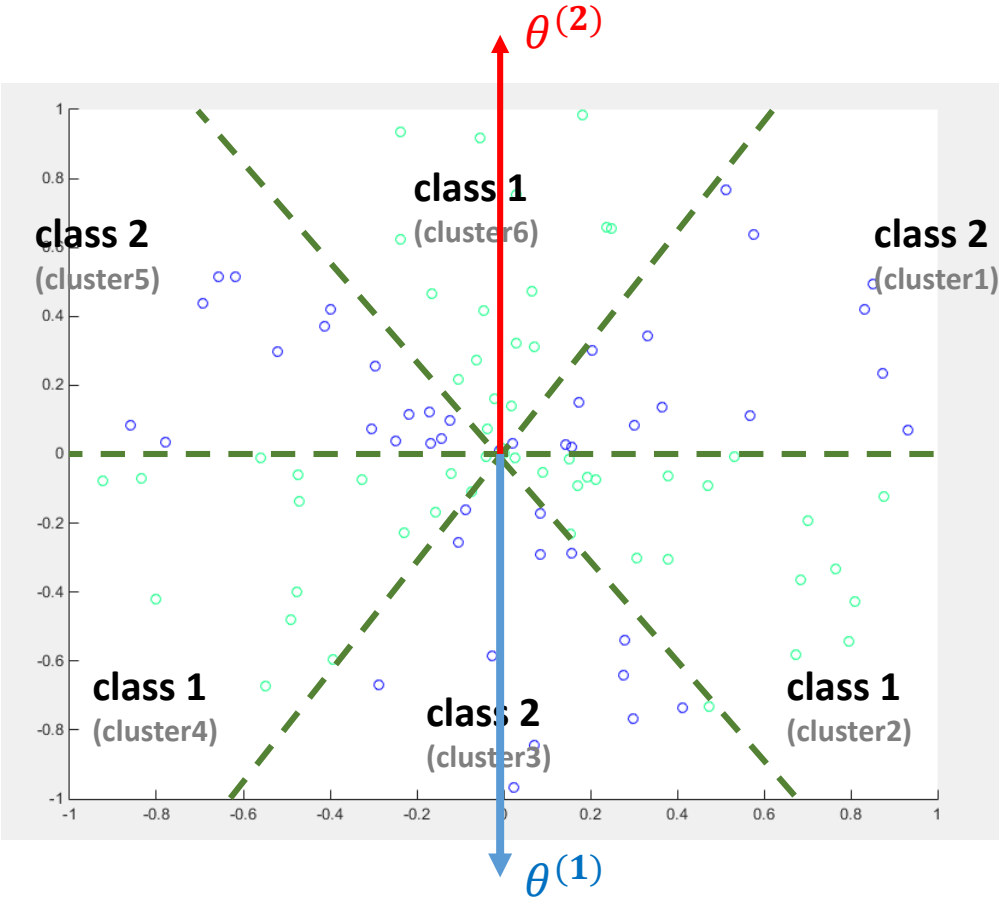
- **Bias term은 클래스를 잘 구분하도록 입력변수 x를 평행이동 시키는 역할**

$$\exp(\theta^T x) = \exp\left(\theta_0 + \theta_1 x_1 + \ \ldots + \theta_p x_p\right) = \exp\left(\theta_1 (x_1 - k_1) + \ \ldots + \theta_p (x_p - k_p)\right)$$



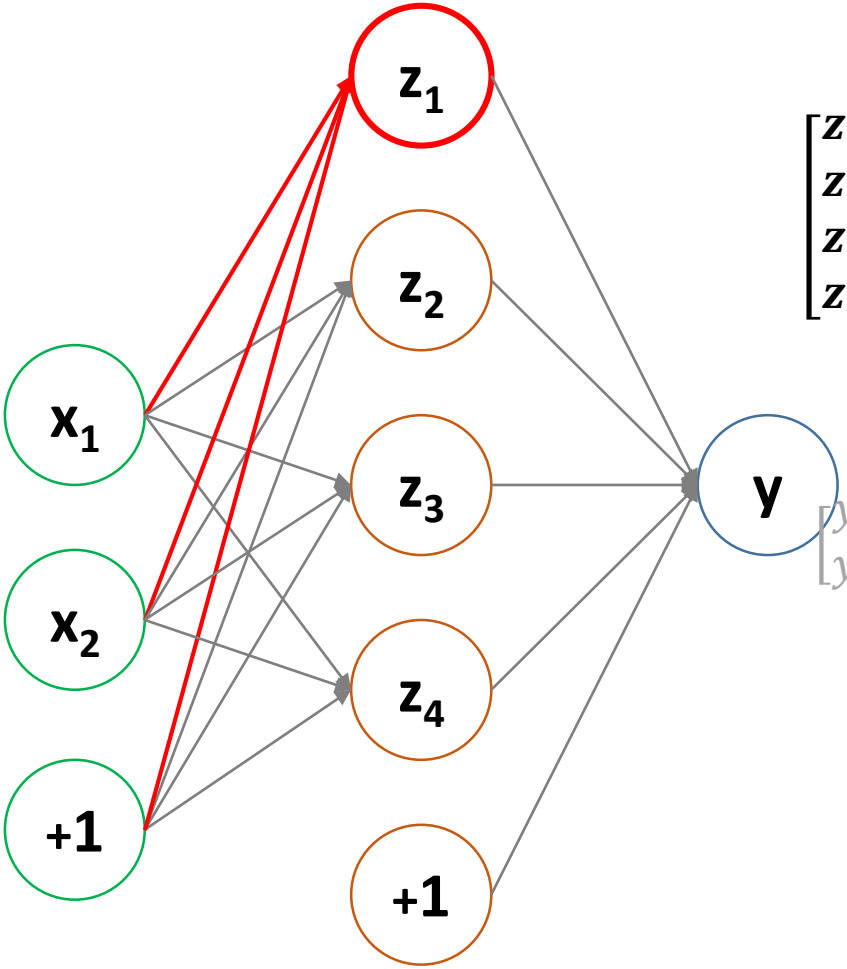
Logistic Regression

- LR은 각 클래스별로 하나의 대표 벡터 $\theta^{(i)}$를 학습하기 때문에, 클래스 별로 데이터가 흩어져 있을 경우에는 (linear inseparable) 잘 작동하지 않음



Neural Network

- Sigmoid를 이용하는 feed-forward neural network는 logistic regression을 여러번 하는 것과 비슷

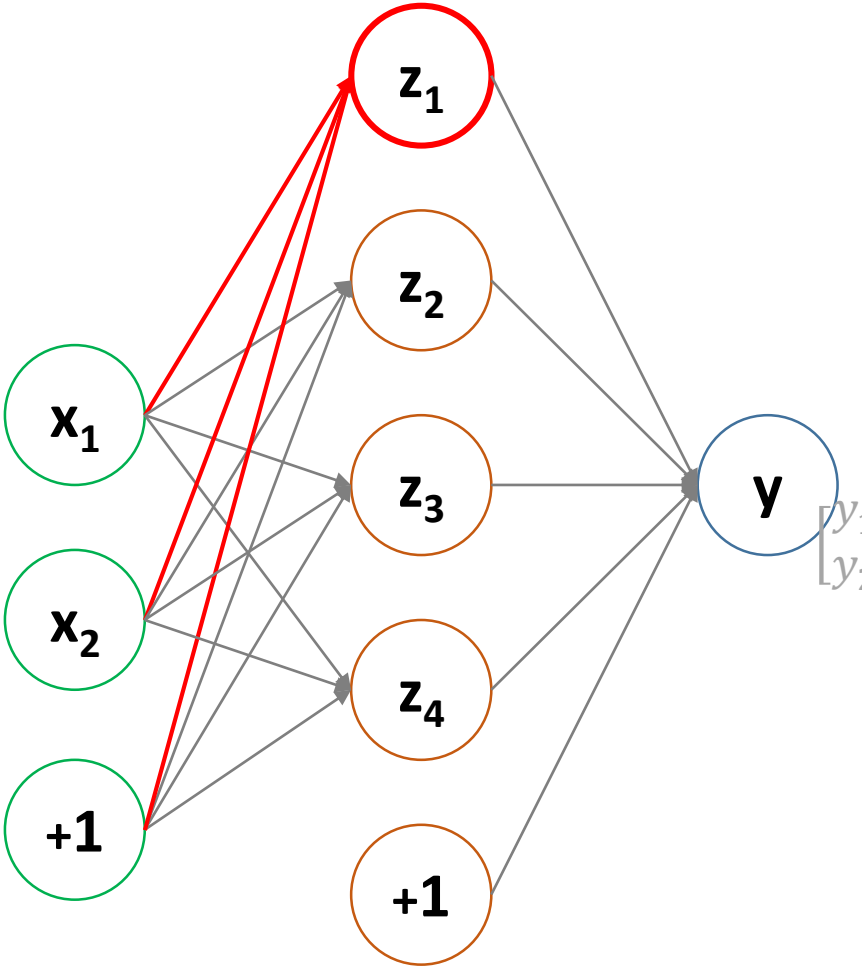


$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \frac{1}{\sum_{j=1}^{4} exp\left(\theta_z^{(j)^T} x\right)} \begin{bmatrix} exp\left(\theta_z^{(1)^T} x\right) \\ \dots \\ exp\left(\theta_z^{(4)^T} x\right) \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \frac{1}{\sum_{j=1}^{2} \exp\left(\theta_y^{(j)^T} z\right)} \begin{bmatrix} \exp\left(\theta_y^{(1)^T} z\right) \\ \exp\left(\theta_y^{(2)^T} z\right) \end{bmatrix}$$

Neural Network

- Softmax regression에서 $z_i = \frac{exp(\theta_z^{(i)T} x)}{\sum_{j=1}^{4} exp(\theta_z^{(j)T} x)}$ 대신, $z_i = \frac{1}{1+exp\left(\theta_z^{(i)T} x\right)}$ 으로 activation 이 다름

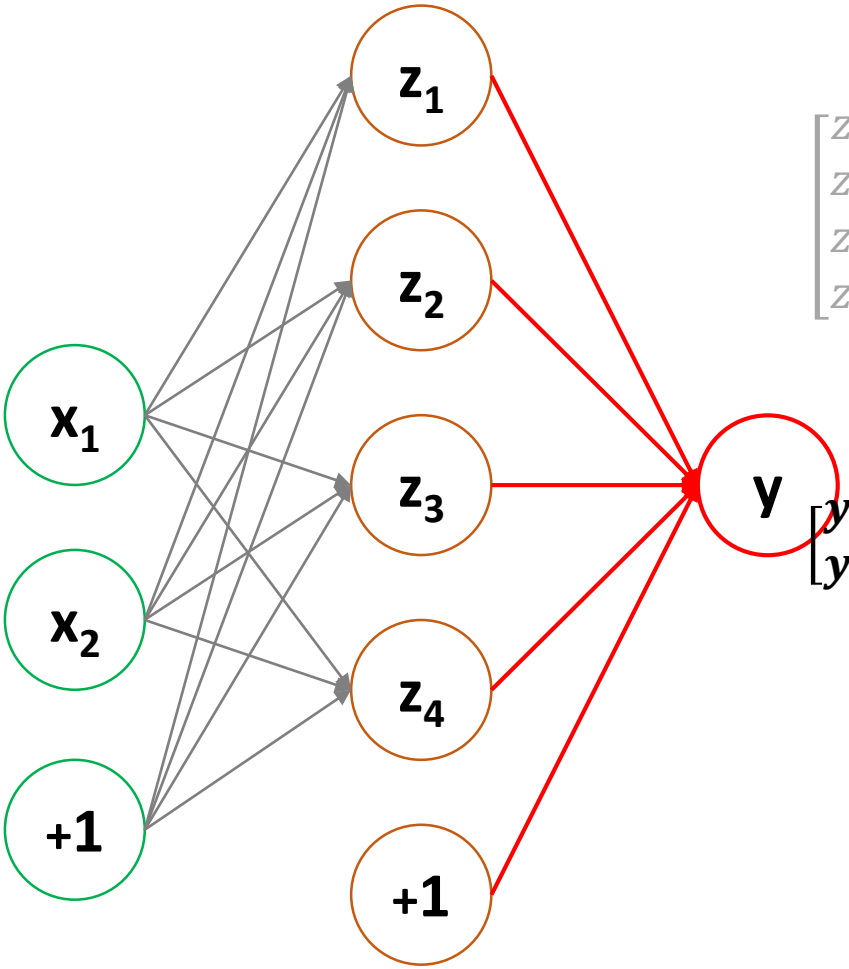


$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} 1 \,/\, \left(1 + exp\left(\theta_z^{(1)T} x\right)\right) \\ \cdots \\ 1 \,/\, \left(1 + exp\left(\theta_z^{(4)T} x\right)\right) \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \frac{1}{\sum_{j=1}^{2} \exp\left(\theta_y^{(j)T} z\right)} \begin{bmatrix} \exp\left(\theta_y^{(1)T} z\right) \\ \exp\left(\theta_y^{(2)T} z\right) \end{bmatrix}$$

Neural Network

- Sigmoid를 이용하는 feed-forward neural network는 logistic regression을 여러번 하는 것과 비슷. 마지막 layer는 Softmax regression

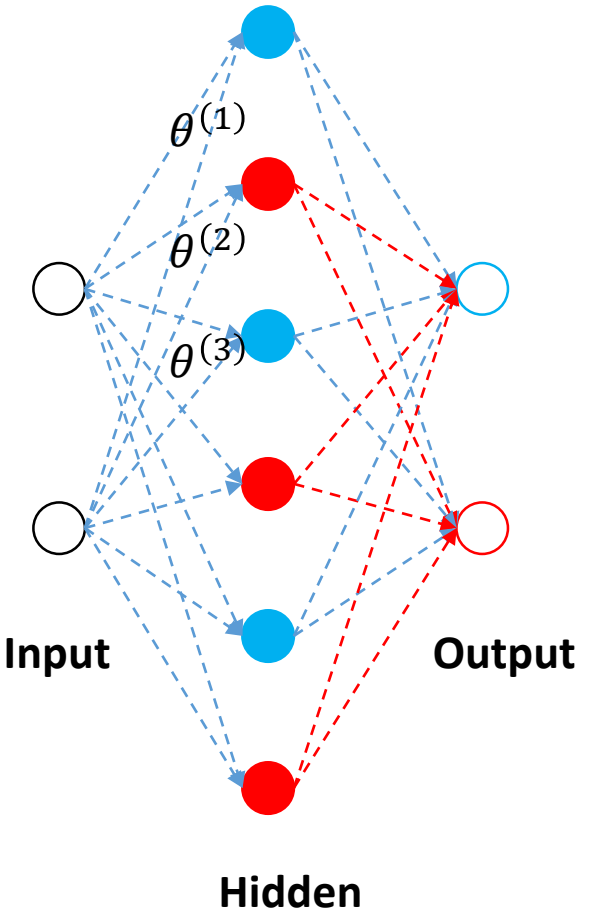


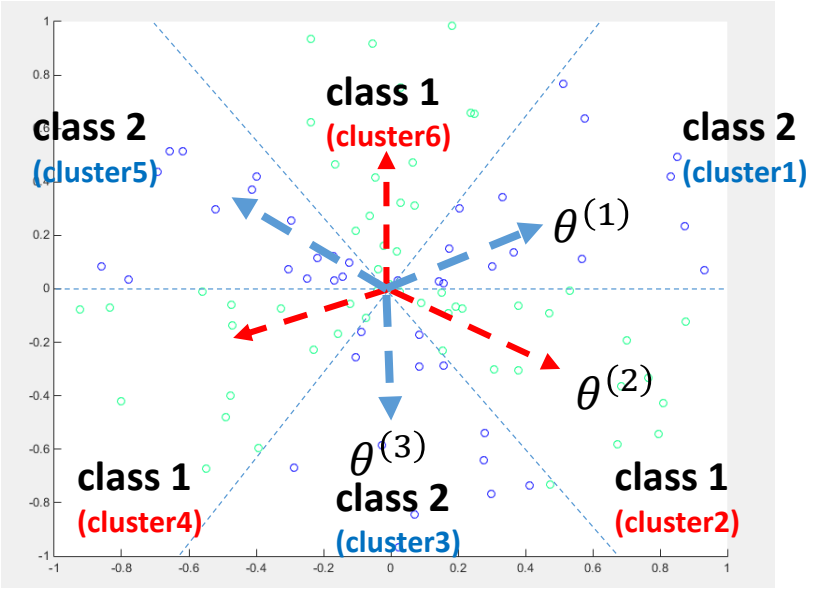
$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \frac{1}{\sum_{j=1}^{4} \exp\left(\theta_z^{(j)T} x\right)} \begin{bmatrix} \exp\left(\theta_z^{(1)T} x\right) \\ \dots \\ \exp\left(\theta_z^{(4)T} x\right) \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \frac{1}{\sum_{j=1}^{2} exp\left(\theta_y^{(j)T} z\right)} \begin{bmatrix} exp\left(\theta_y^{(1)T} z\right) \\ exp\left(\theta_y^{(2)T} z\right) \end{bmatrix}$$

Neural Network

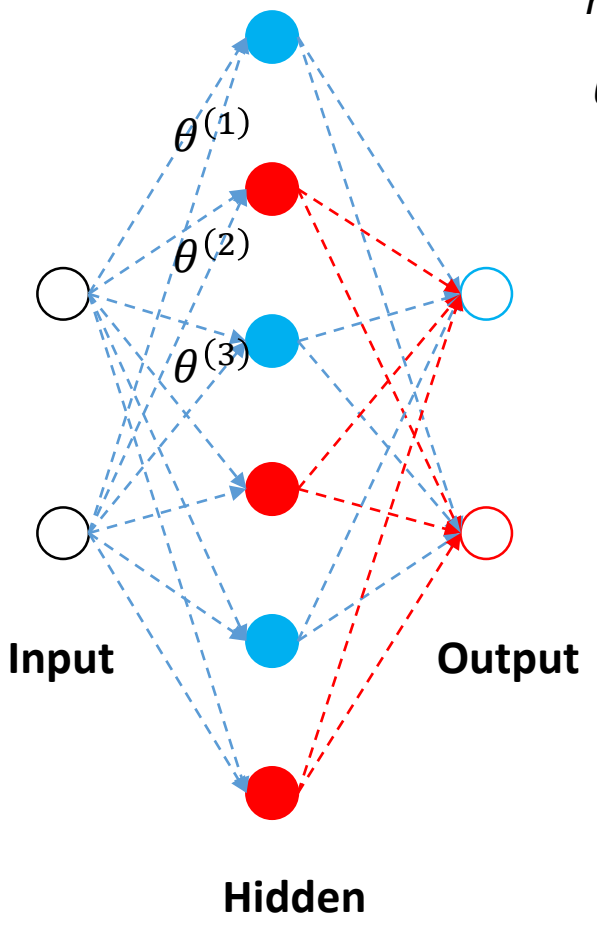
- Neural network의 hidden layers 역할은 linear inseparable한 데이터를 조금씩 linear separable한 공간(representation)으로 바꾸는 것





Neural Network

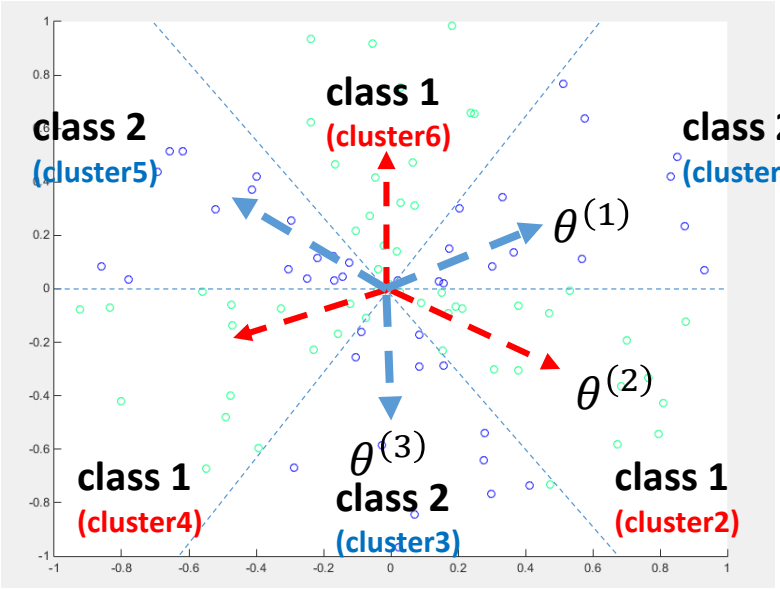
- Sigmoid는 데이터를 Boolean representation에 가깝게 변형시킴



Hidden to output parameters

$$h^{(1)} = [1, \quad 0.3, \quad 0.1, \quad 0.03, \quad 0.1, \quad 0.3]$$

$$\theta^{h \to o_1} = [1, \quad -1, \quad 1, \quad -1, \quad 1, \quad -1]$$



Regularization

- **p-norm은 벡터의 크기를 정의하는 방법**

  - $|X|_p = \sqrt[p]{|X_1|^P + \ldots + |X_q|^P}$ 로 정의되며, X_j는 j번째 차원의 값

  - p =2이면, L2 norm이라 부르며, Euclidean distance에 해당
  - p=1 이면, X의 각 차원의 값의 절대값의 합. Manhattan distance에 해당
  - p=0 이면, X의 각 차원의 값이 0이 아닌 개수

Regularization

- **모델의 복잡도에 제약/페널티를 부여**
  - 학습데이터에 과적합 (overfitting)하는 것을 방지
  - L2, L1 regularization이 대표적
  - Logistic Regression 모델의 경우,

$$\text{Loss} = \sum_i^n \left(y_i - \frac{1}{1+\exp\left(-\left(\beta_0+\beta_1 x_{i1}+ \ldots \beta_p x_{ip}\right)\right)} \right)^2$$

L1 regularization

$$\text{Cost} = \sum_i^n \left(y_i - \frac{1}{1+\exp\left(-\left(\beta_0+\beta_1 x_{i1}+ \ldots \beta_p x_{ip}\right)\right)} \right)^2 + \lambda \sum_j^p |\beta_j|$$

L2 regularization

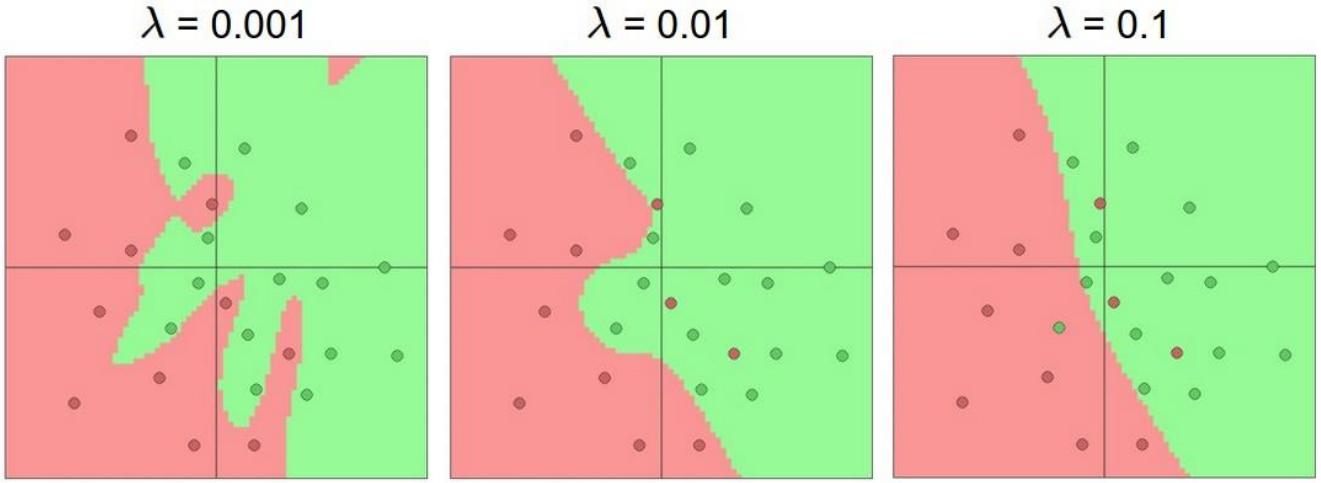
$$\text{Cost} = \sum_i^n \left(y_i - \frac{1}{1+\exp\left(-\left(\beta_0+\beta_1 x_{i1}+ \ldots \beta_p x_{ip}\right)\right)} \right)^2 + \lambda \sum_j^p |\beta_j|^2$$

Regularization

- **L2 regularization은 경계면을 날카롭지 않게 하는 역할**
  - β_i 중에서 크기가 유독 큰 값이 없도록 만들기 때문
  - Regression에 L2 penalty를 줄 경우, ridge regression이라 부름

Cost = Loss + Regularization term

$$= \sum_i^n \left(y_i - \frac{1}{1+\exp\left(-(\beta_0+\beta_1 x_{i1}+ \ldots \beta_p x_{ip})\right)} \right)^2 + \lambda \sum_j^p |\beta_j|^2$$



http://cs231n.github.io/neural-networks-1/

Regularization

- **L1 regularization은 입력변수 계수 β_j 중 일부를 0에 매우 가깝게 만들어, 중요한 입력변수를 선택하는 효과가 있음 (sparse modeling)**

  - L1은 L0 regularization의 근사로, L0이 미분이 되지 않기 때문에 쓰이게 되었으나 중요한 변수를 고르는 효과 측면에서는 비슷.
  - 오히려 L0에 비하여 β_j크기까지 제한하는 효과가 있음

$$\text{Cost} = \sum_i^n \left(y_i - \frac{1}{1+\exp\left(-(\beta_0+\beta_1 x_{i1}+ \ldots \beta_p x_{ip})\right)} \right)^2 + \lambda \sum_j^p |\beta_j|$$

http://cs231n.github.io/neural-networks-1/

Regularization

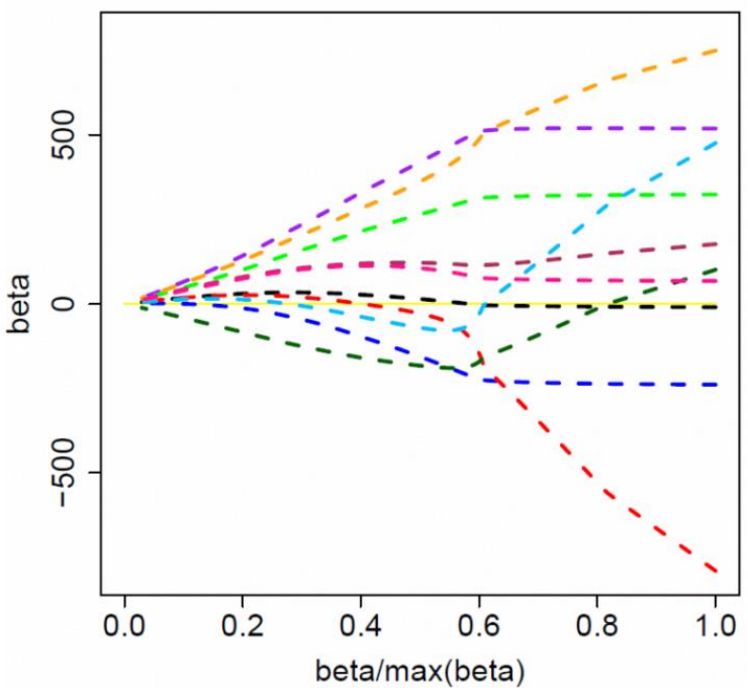
- **L1 regularization을 이용한 방법을 LASSO라 부르며, Lasso regression의 경우 중요 변수를 추출해주는 용도로도 쓰임**

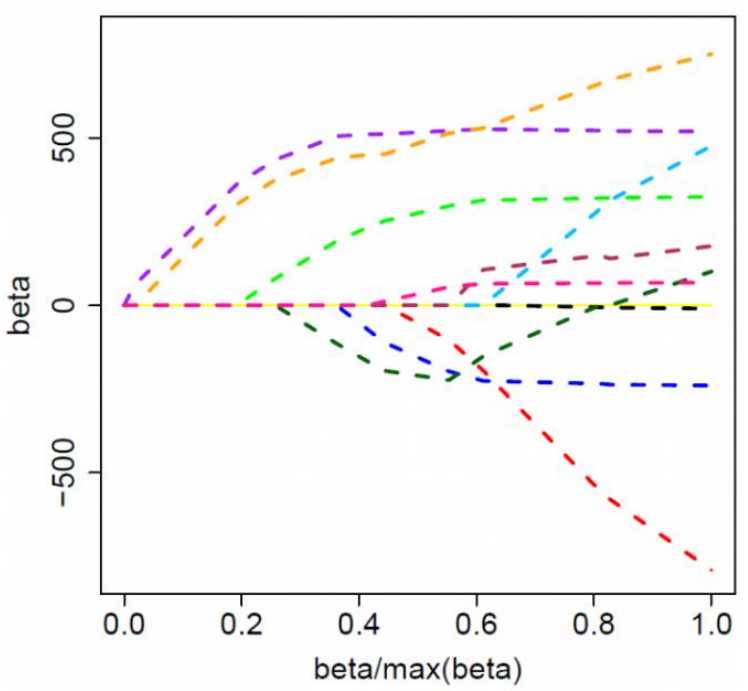
  - Cost = $\sum_i^n \left(y_i - \frac{1}{1+\exp\left(-\left(\beta_0+\beta_1 x_{i1}+ \ldots \beta_p x_{ip}\right)\right)} \right)^2 + \lambda \sum_j^p |\beta_j|$

- **λ의 크기에 따라서 모델이 선택하는 변수가 달라짐**
  - λ 값이 작을수록 더 많은 변수를 사용
  - λ에 따라 어떤 변수들이 이용되는지 살펴볼 수 있음 (Lasso path)

Regularization

- **Lambda에 따른 beta의 값의 변화를 그리면 Lasso path를 그릴 수 있음**
  - L2 regularization은 lambda가 작아짐에 따라 (=beta/max(beta)가 커짐에 따라) 많은 beta들이 동시에 커지나
  - L1 regularization은 lambda가 작아짐에 따라 변수를 추가적으로 이용





https://onlinecourses.science.psu.edu/stat857/node/158