Classifiers (document classifications)

Hyunjoong Kim

soy.lovit@gmail.com

https://github.com/lovit

1. From Logistic Regression to Neural Network

2. Support Vector Machine (Linear, RBF kernel)

3. k - Nearest Neighbor Classifier

4. Naïve Bayes Classifier

5. Decision Tree

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 Logistic regression의 multi class classification 버전

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

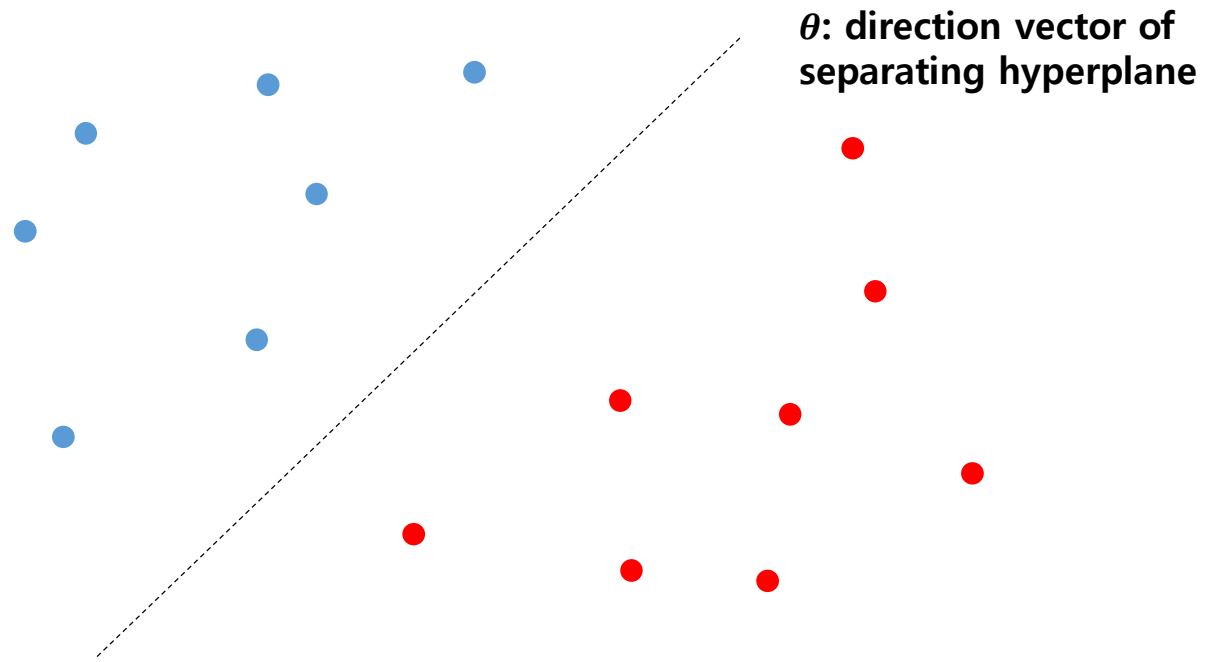
  - 학습 데이터 $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)}\}$ 가 주어졌을 때, loss function은

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

Logistic Regression

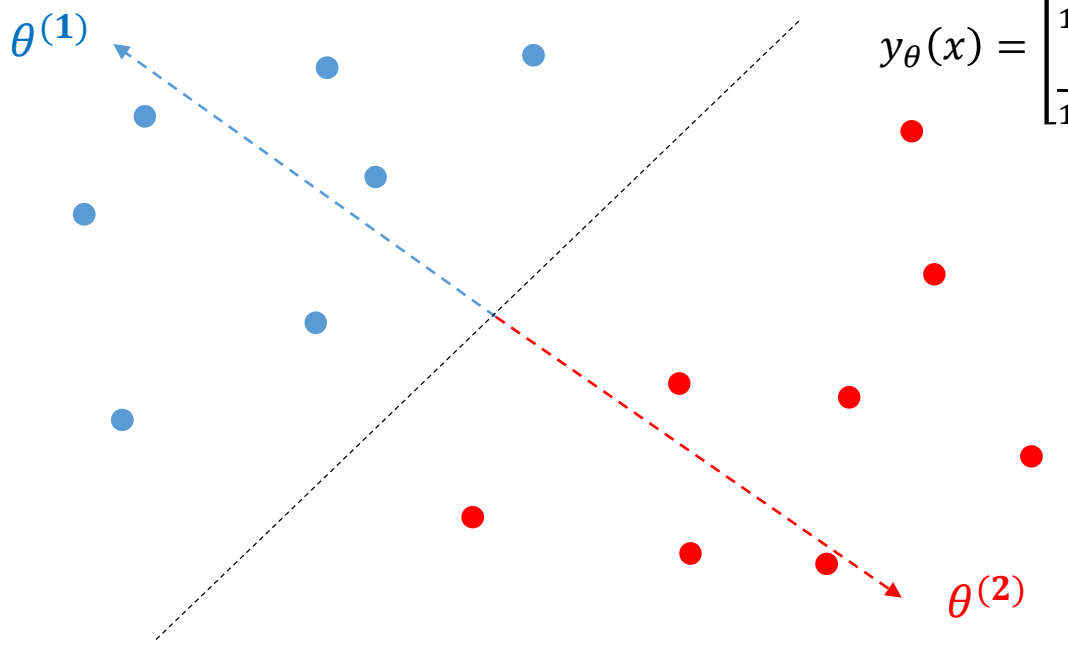
- 일반적으로 LR은 두 클래스의 경계면을 학습한다고 표현합니다

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



Logistic Regression

- Softmax regression 표현하면 $\theta = \theta^{(2)} - \theta^{(1)}$이며, $\theta^{(i)}$는 클래스 i의 대표 벡터로 해석할 수 있습니다
- Cosine measure 처럼 $\theta^{(i)}$의 방향이 중요합니다

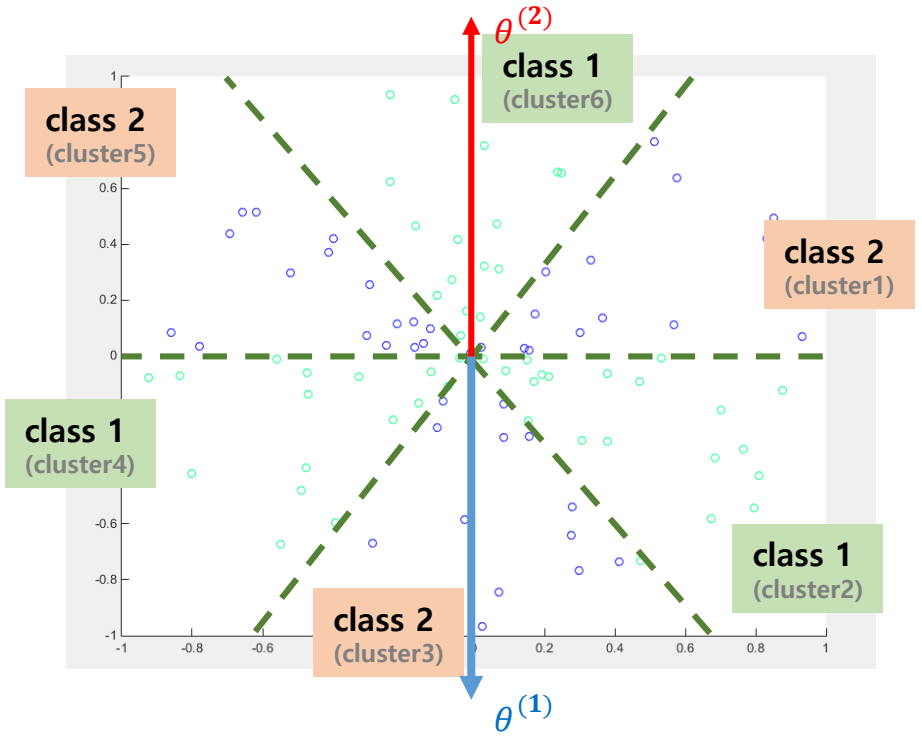


$$y_\theta(x) = \begin{bmatrix} \frac{1}{1+\exp(\theta^T x)} \\ \frac{\exp(\theta^T x)}{1+\exp(\theta^T x)} \end{bmatrix} = \begin{bmatrix} \frac{\exp(\theta^{(1)^T} x)}{\exp(\theta^{(1)^T} x)+\exp(\theta^{(2)^T} x)} \\ \frac{\exp(\theta^{(2)^T} x)}{\exp(\theta^{(1)^T} x)+\exp(\theta^{(2)^T} x)} \end{bmatrix}$$

Logistic Regression

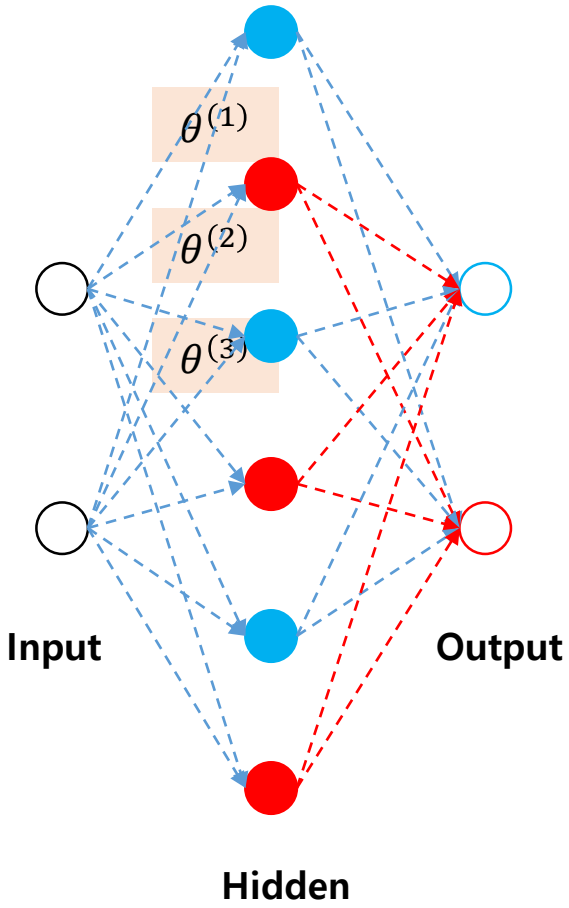
- LR은 각 클래스별로 하나의 대표 벡터 $\theta^{(i)}$를 학습합니다
- 한 클래스의 데이터가 여러 지역에 흩어진다면 잘 작동하지 않습니다.

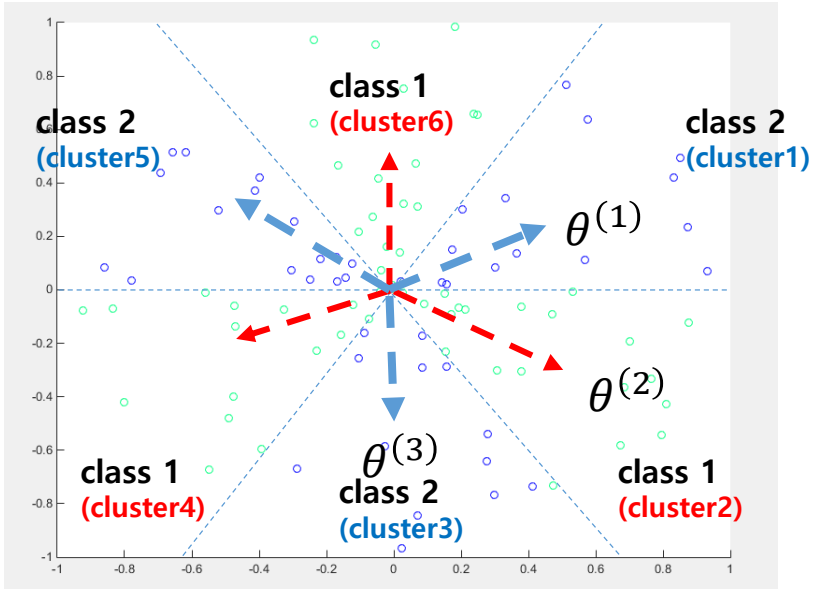
  (linear inseparable)



Feed-forward Neural Network

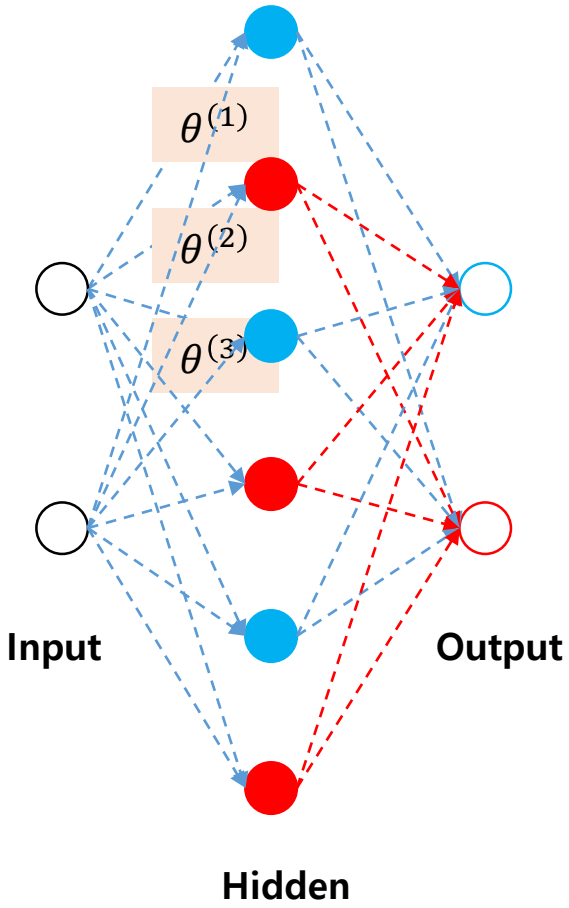
- Neural network의 hidden layers 는 linear inseparable한 데이터를 linear separable한 공간(representation)으로 바꾸는 것입니다.





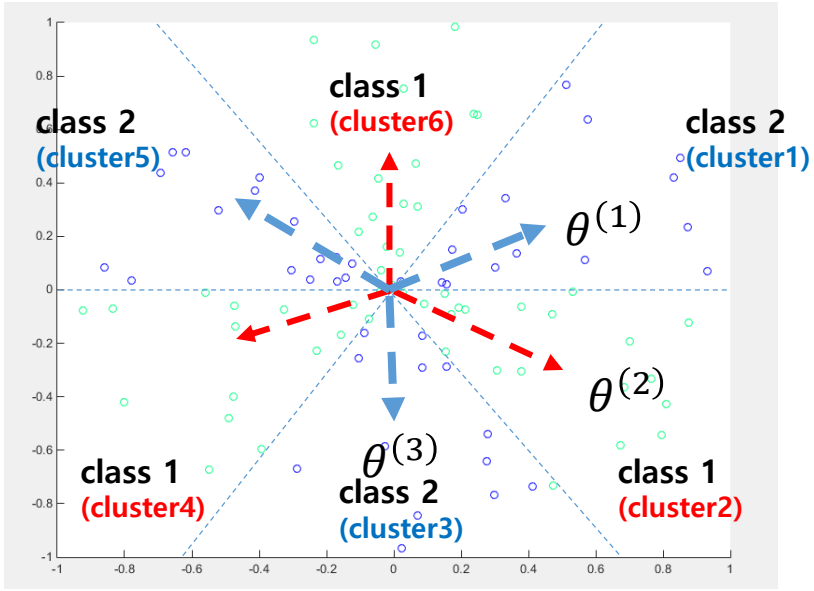
Feed-forward Neural Network

- Sigmoid는 데이터를 Boolean representation에 가깝게 변형시킵니다.



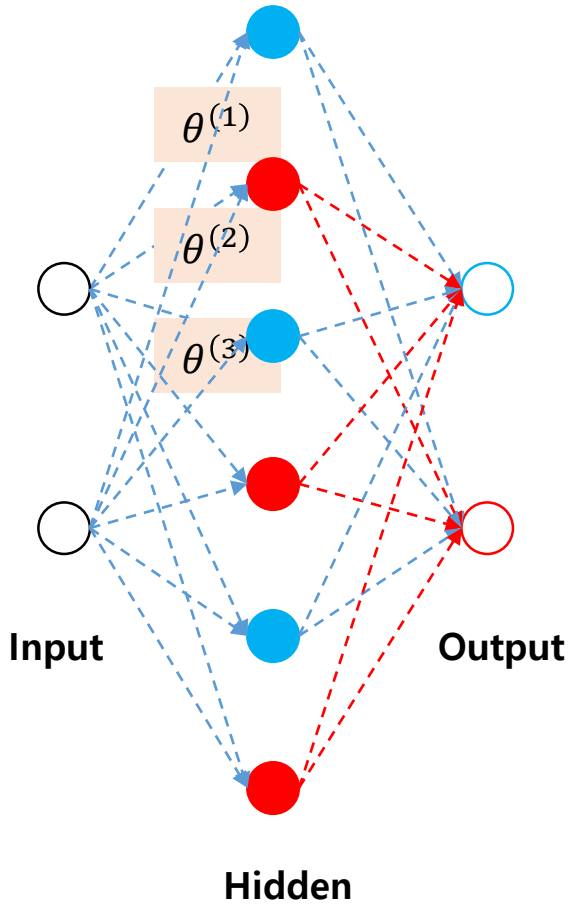
Hidden to output parameters

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



Feed-forward Neural Network

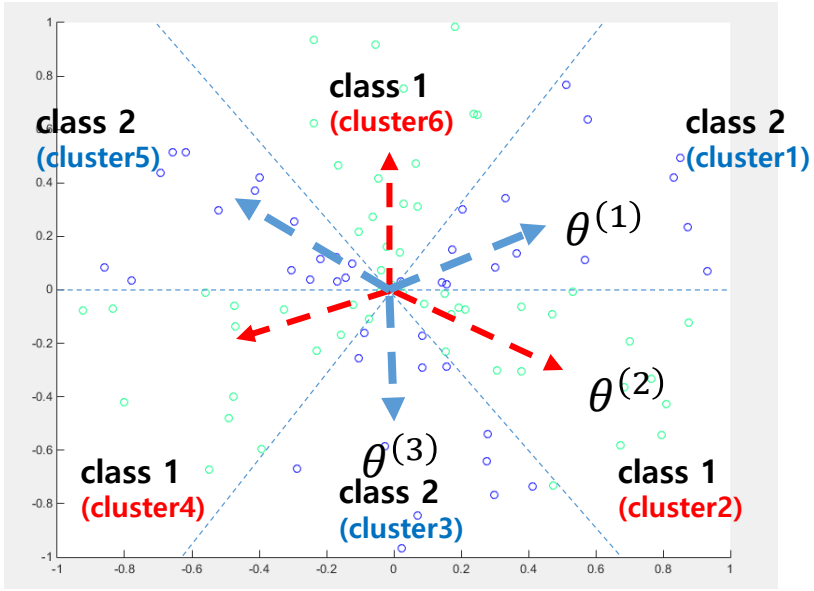
- Hidden layer에 의하여 6차원으로 변한 데이터는 linear separable 입니다.



Hidden to output parameters

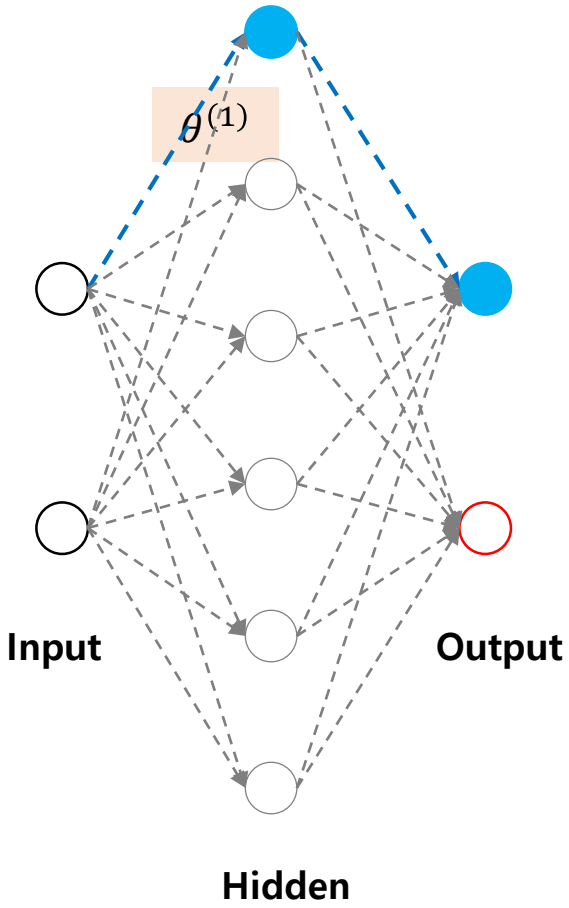
$$h^{(1)} = [1, \quad 0.3, \quad 0.1, \quad 0.03, \quad 0.1, \quad 0.3]$$

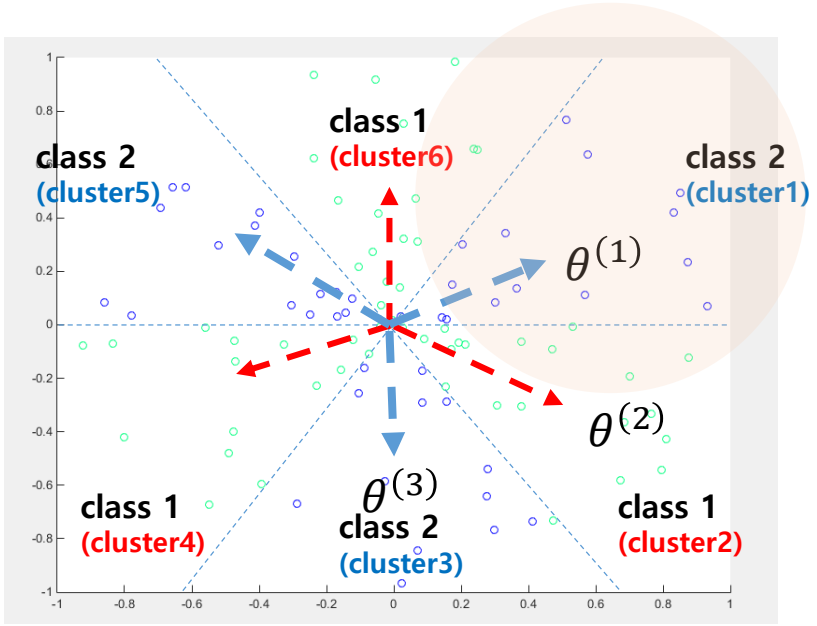
$$\theta^{h \to o_1} = [1, \quad -1, \quad 1, \quad -1, \quad 1, \quad -1]$$



Feed-forward Neural Network

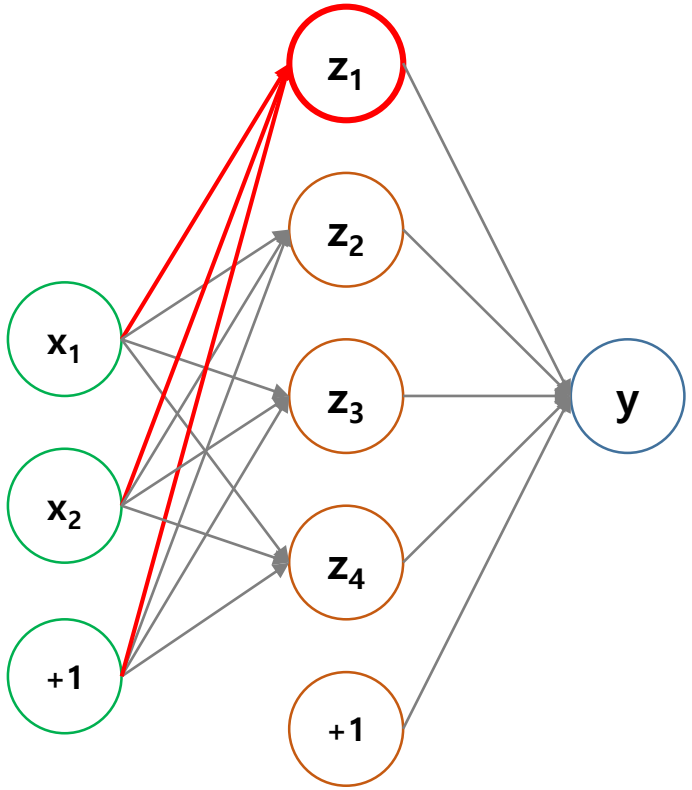
- A unit of hidden layer 는 한 지역을 표현합니다. 군집화와 비슷합니다.
  - Cluster 1에 속하면 파란색 클래스일 가능성이 높습니다.





Feed-forward Neural Network

- Activation function 은 hidden layer에서 Softmax regression 을 수행하는 것과 비슷한 효과를 보입니다.

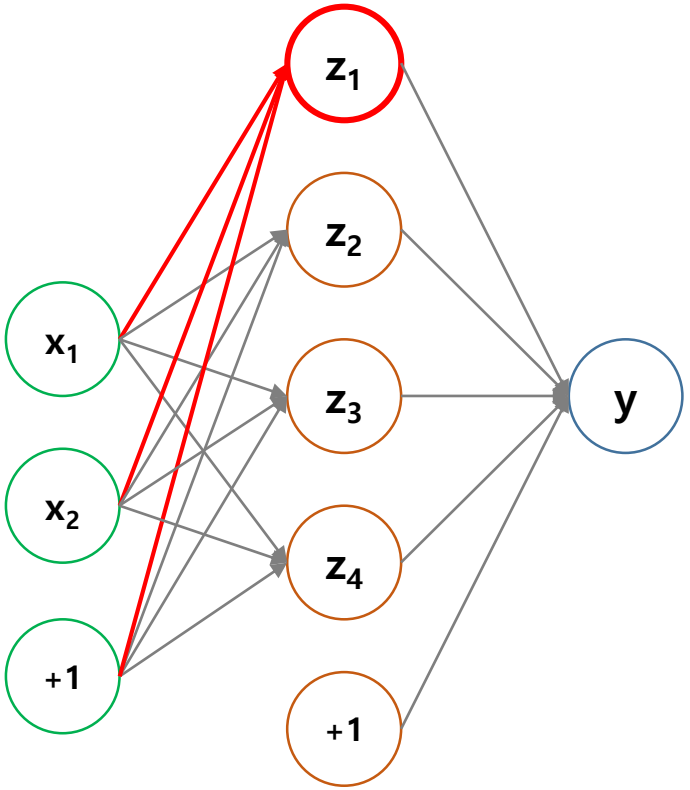


$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \frac{1}{\sum_{j=1}^{4} exp\left(\theta_z^{(j)^T} x\right)} \begin{bmatrix} exp\left(\theta_z^{(1)^T} x\right) \\ \dots \\ exp\left(\theta_z^{(4)^T} x\right) \end{bmatrix}$$

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

Feed-forward Neural Network

- 하지만 $z_i = \frac{exp(\theta_z^{(i)^T} x)}{\sum_{j=1}^{4} exp(\theta_z^{(j)^T} x)}$ 대신, $z_i = \frac{1}{1+exp\left(\theta_z^{(i)^T} x\right)}$ 의 activation function을 이용합니다.

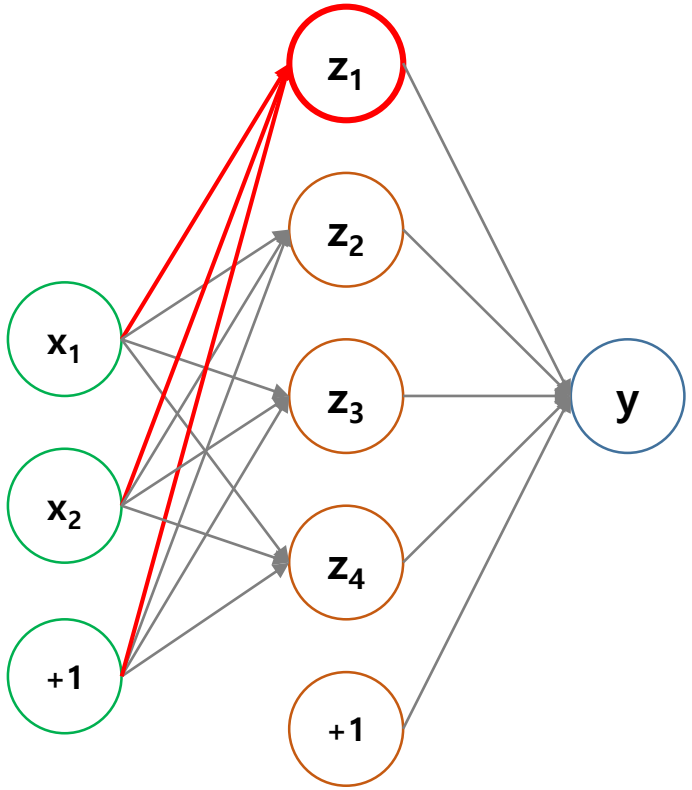


$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} 1 \,/\, \left(1 + exp\left(\theta_z^{(1)^T} x\right)\right) \\ \cdots \\ 1 \,/\, \left(1 + exp\left(\theta_z^{(4)^T} x\right)\right) \end{bmatrix}$$

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

Feed-forward Neural Network

- 하지만 마지막 layer는 Softmax regression을 이용합니다.
  - 각 클래스에 속할 확률을 표현합니다.

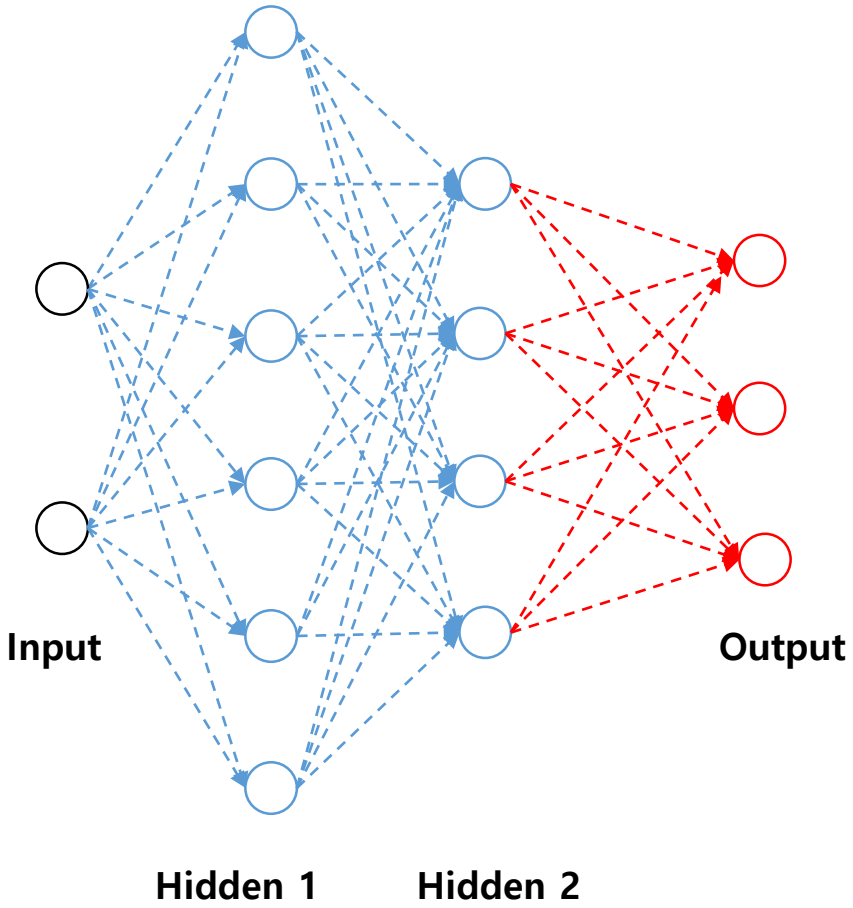


$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} 1 \,/\, \left(1 + exp\left(\theta_z^{(1)^T} x\right)\right) \\ \cdots \\ 1 \,/\, \left(1 + exp\left(\theta_z^{(4)^T} x\right)\right) \end{bmatrix}$$

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

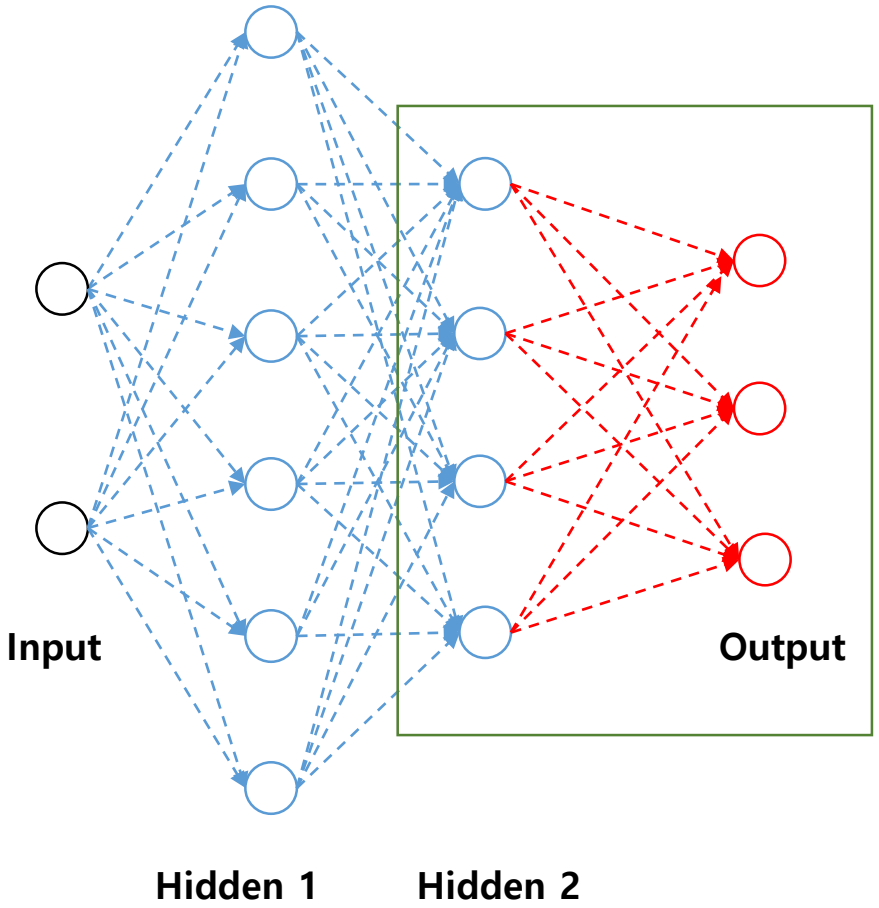
Feed-forward Neural Network

- 여러 개의 hidden layers 는 여러번의 공간 변형을 통하여 문제를 linear separable 하게 변형하는 것입니다. (예시는 2차원의 데이터, 3 class classification)



Feed-forward Neural Network

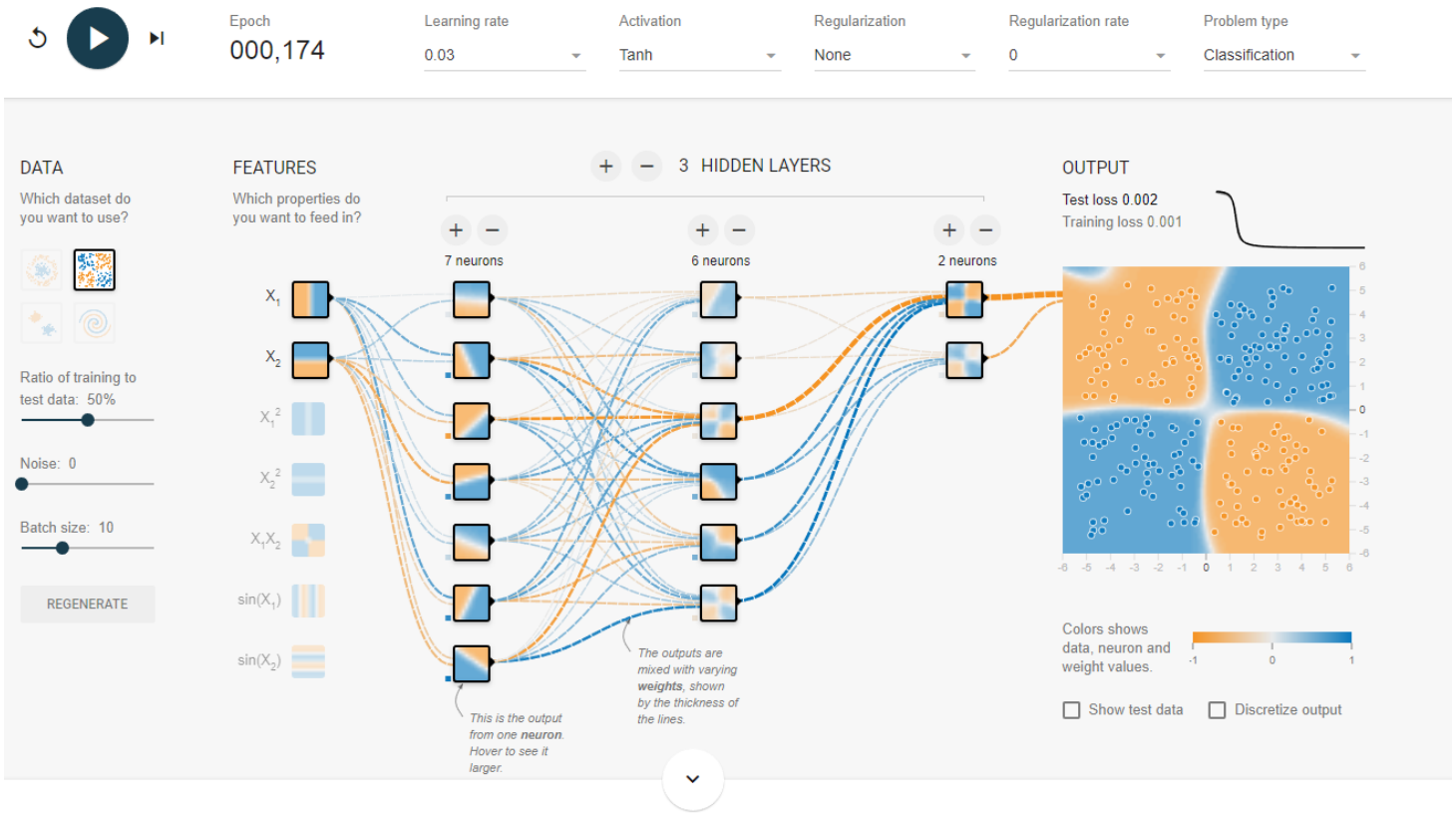
- 여러 개의 hidden layers 는 여러번의 공간 변형을 통하여 문제를 linear separable 하게 변형하는 것입니다. (예시는 2차원의 데이터, 3 class classification)



Hidden 2 의 4차원이 linear separable 하다면,
Softmax (hidden2 → output) 은 잘 작동합니다.

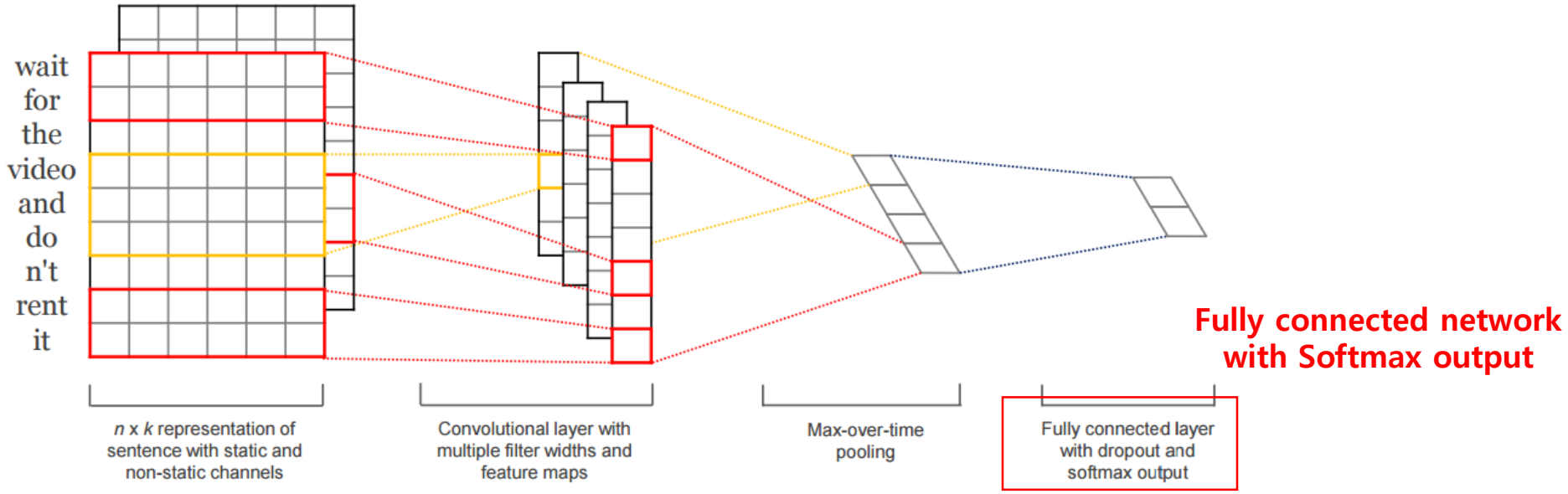
Feed-forward Neural Network

- http://playground.tensorflow.org/ 에서 neural network의 hidden layer 에 의해하여 데이터 공간이 어떻게 변하는지 살펴볼 수 있습니다.



Feed-forward Neural Network

- Deep learning 도 마지막 layer 는 Softmax regression 입니다.
- Deep learning 은 어려운 문제를 쉽게 풀기 위해 다양한 방법으로 데이터를 linear separable 하게 변형합니다.



Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

1. From Logistic Regression to Neural Network

2. Support Vector Machine (Linear, RBF kernel)

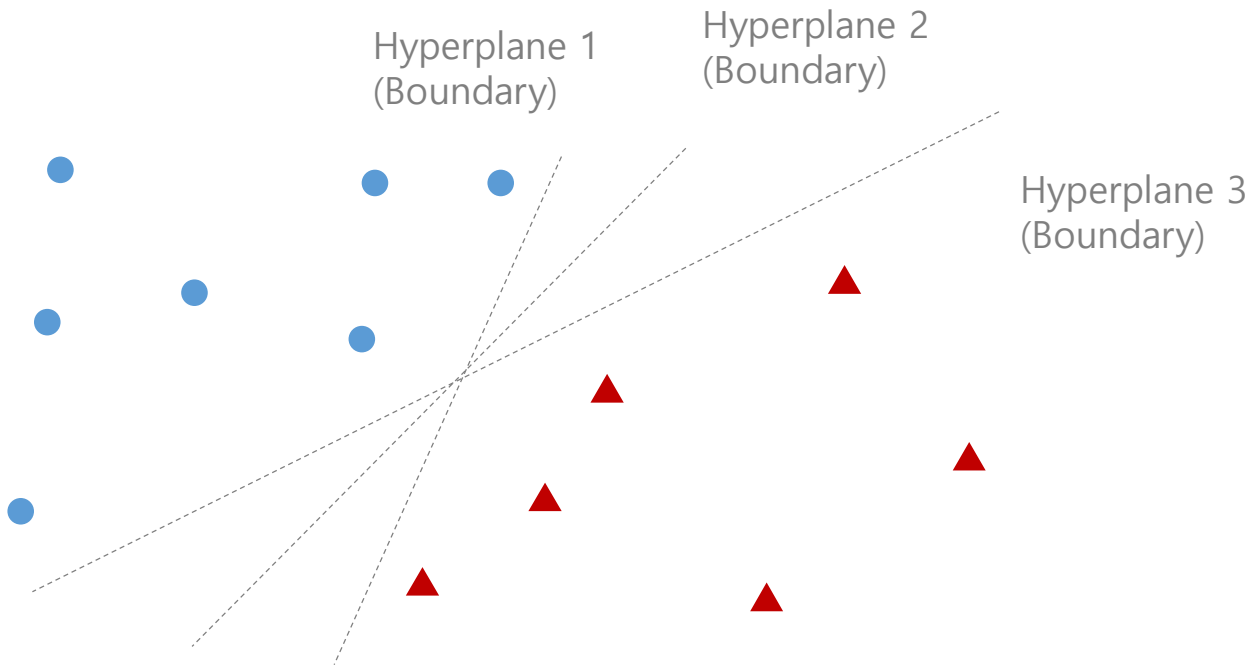
3. k - Nearest Neighbor Classifier

4. Naïve Bayes Classifier

5. Decision Tree

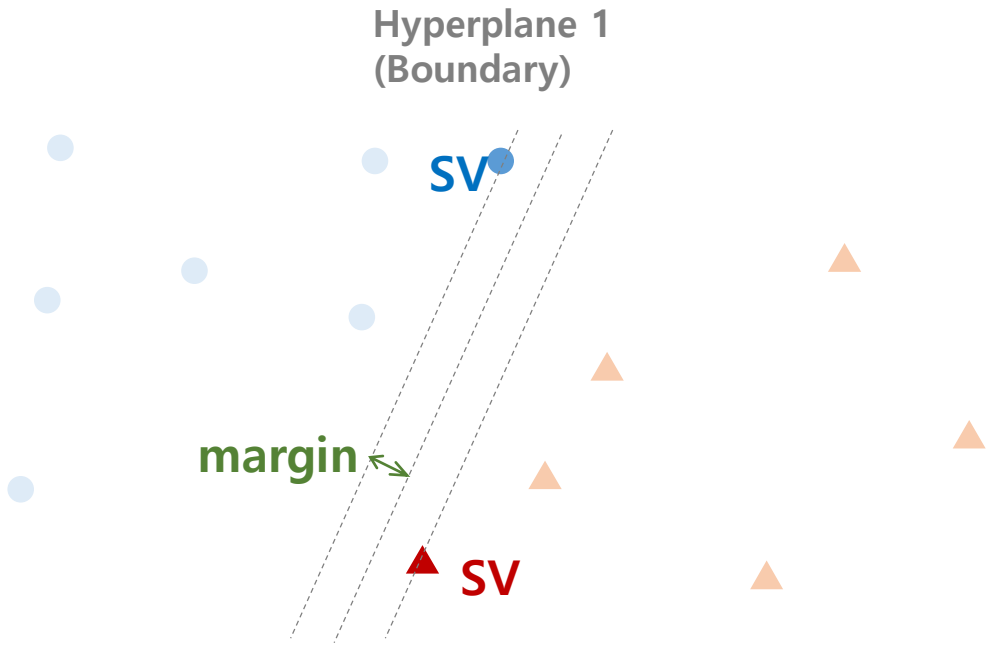
Support Vector Machine

- SVM은 N개의 데이터 중에서 중요한 몇 개의 포인트(support vector)를 선택하여 분류 문제를 풉니다.
- 데이터를 구분하는 경계면은 여러 개가 만들어질 수 있습니다.



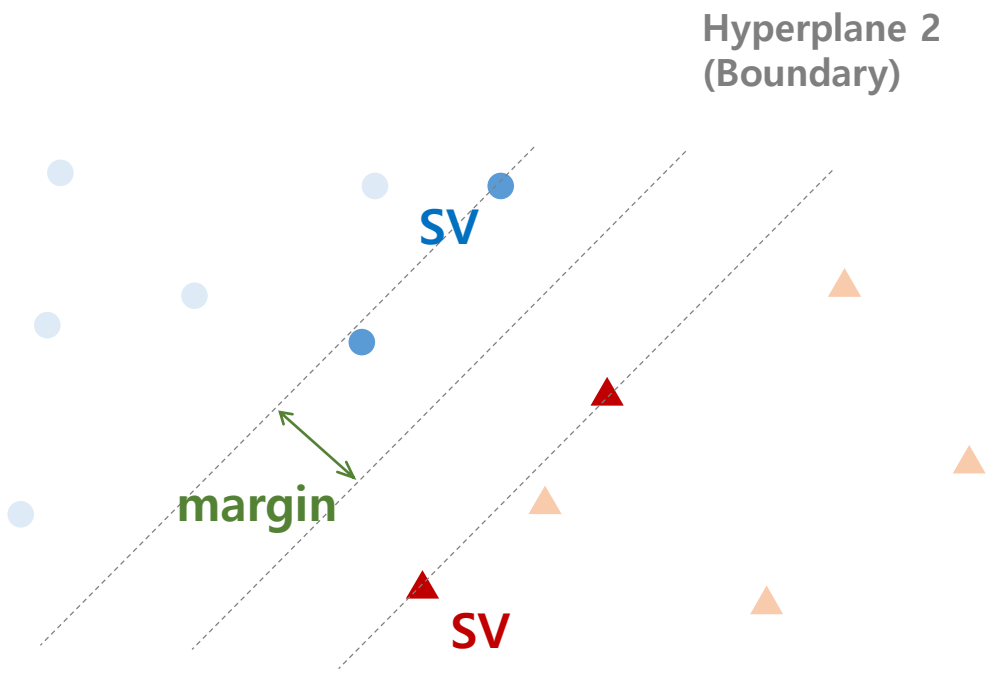
Support Vector Machine

- 여러 개의 경계면 중에서 margin이 가장 큰 경계면을 선택합니다.
- 경계면과 만나는 점들이 Support Vectors 입니다.



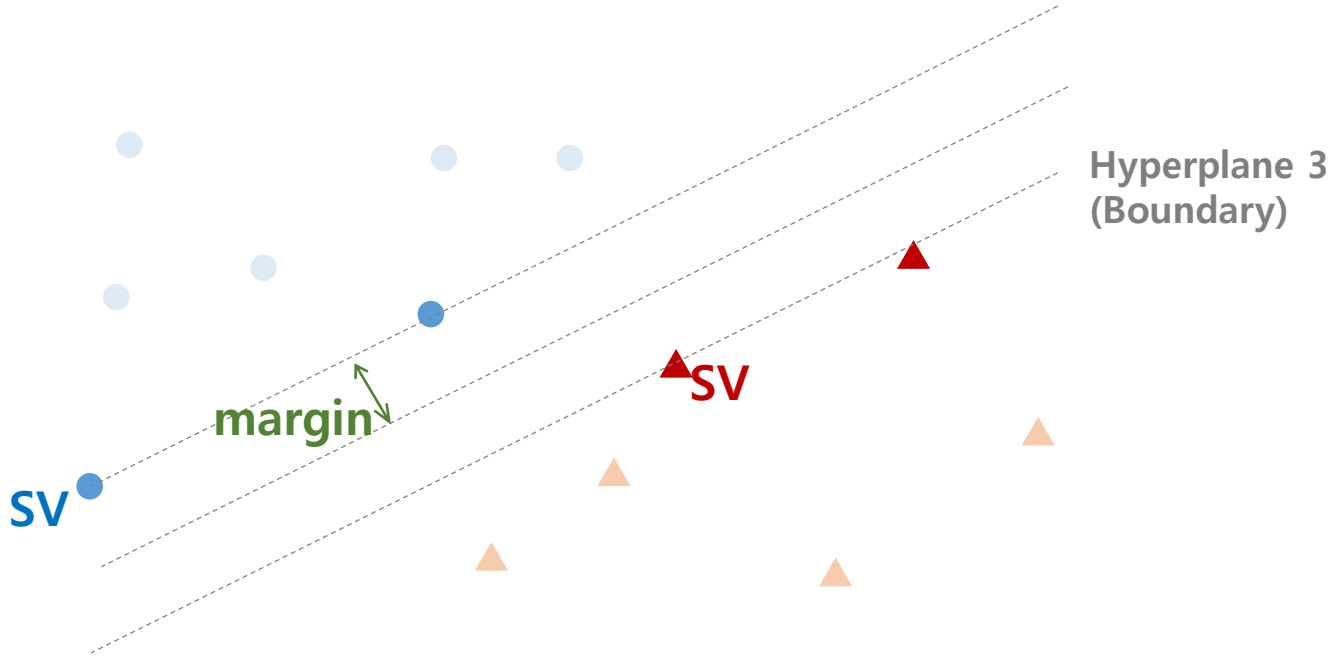
Support Vector Machine

- 경계면 2의 margin 이 가장 큽니다.



Support Vector Machine

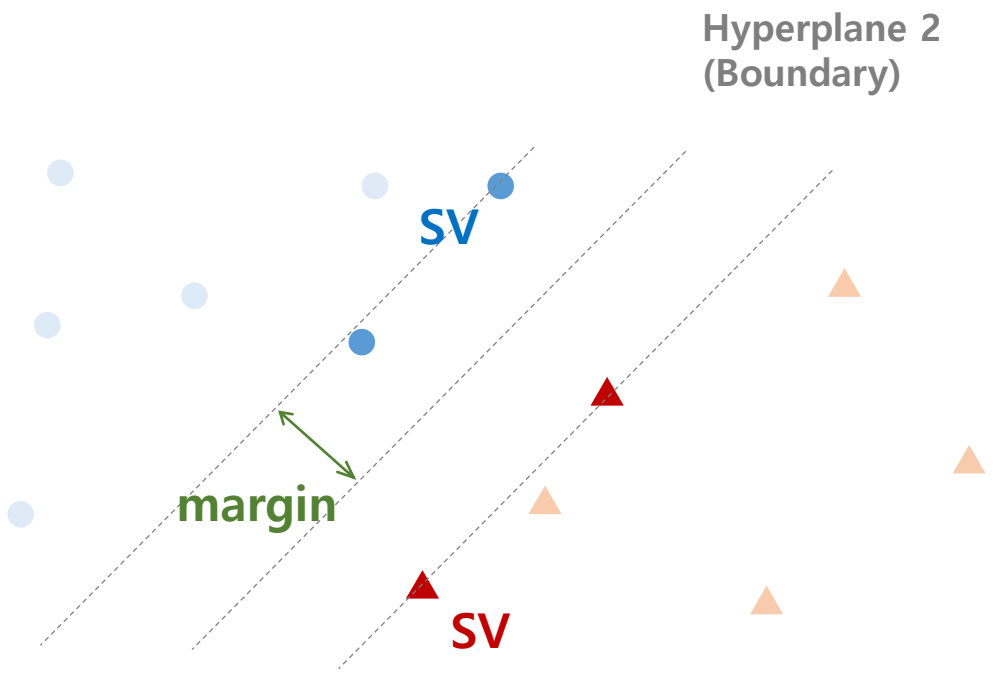
- 여러 개의 경계면 중에서 margin이 가장 큰 경계면을 선택합니다.



Support Vector Machine

- maximum margin 문제를 풀면 classifier 식을 얻습니다.

$$f(q) = \sum_{i \in SV} \alpha_i \, y_i \, (x_i^T \, q) + b$$



Support Vector Machine

- maximum margin 문제를 풀면 classifier 식을 얻습니다.

$$f(q) = \sum_{i \in SV} \alpha_i \, y_i \left(x_i^T \, q \right) + b$$

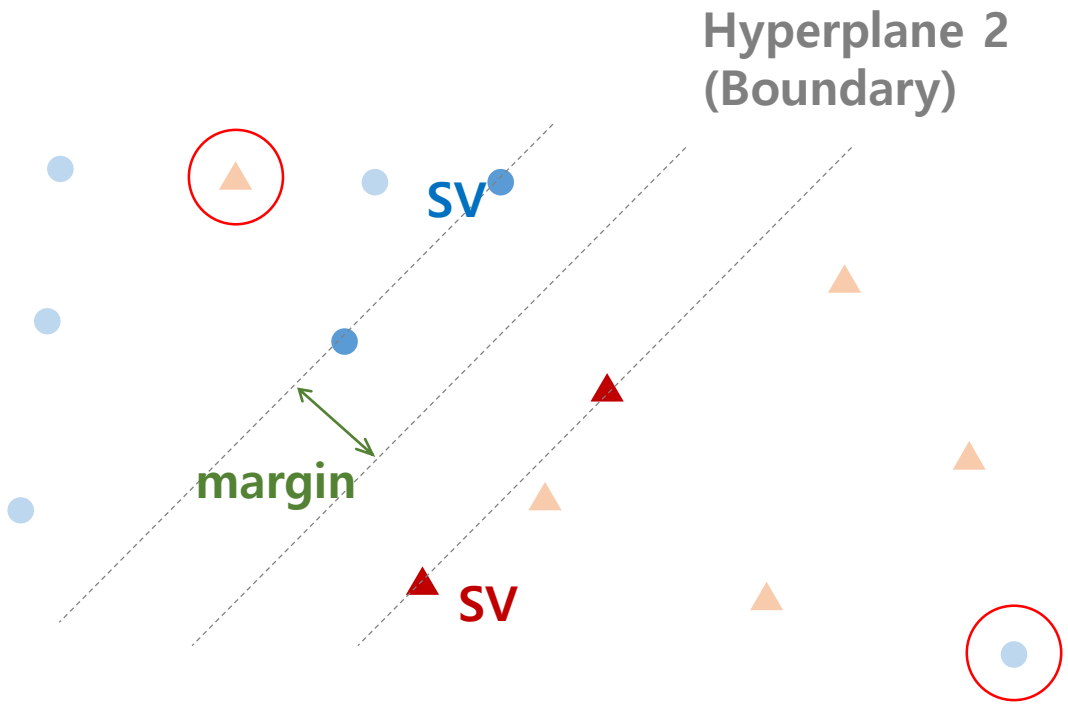
Support vectors의 label y와
이에 대한 가중치 α_i

Support vectors와 query vector와의
Inner product에 의한 유사도

- SVM은 SV 와의 유사도와 가중치를 이용하는 instance learning 입니다.

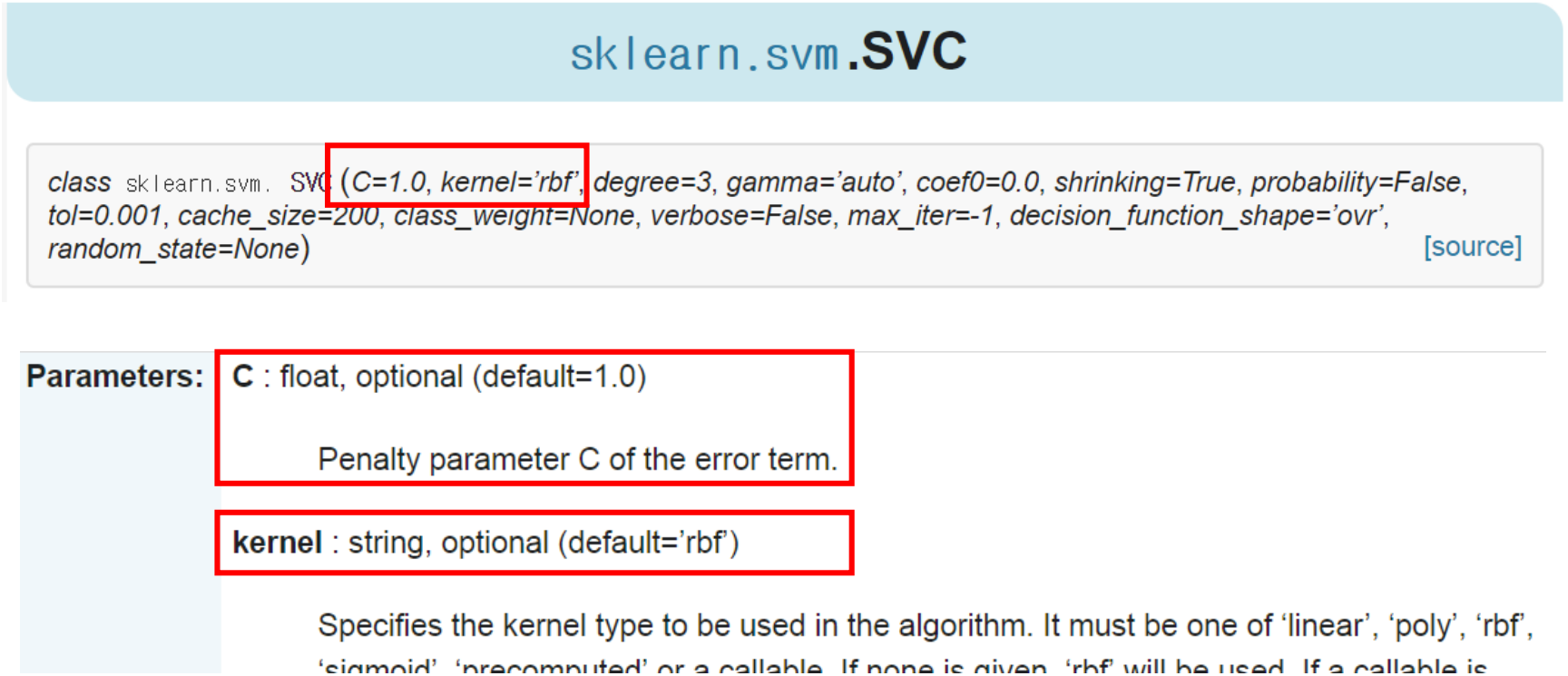
Support Vector Machine

- 오류가 없는 경계면을 hard margin solution 이라 합니다.
- 데이터의 오류를 고려한 경계면을 soft margin solution 이라 합니다.



Support Vector Machine

- Soft margin은 train error를 허용하기 때문에, 이에 대한 penalty를 패러매터로 조절할 수 있습니다.



sklearn.svm.SVC

class sklearn.svm. SVC (*C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None*) [source]

Parameters:

C : float, optional (default=1.0)

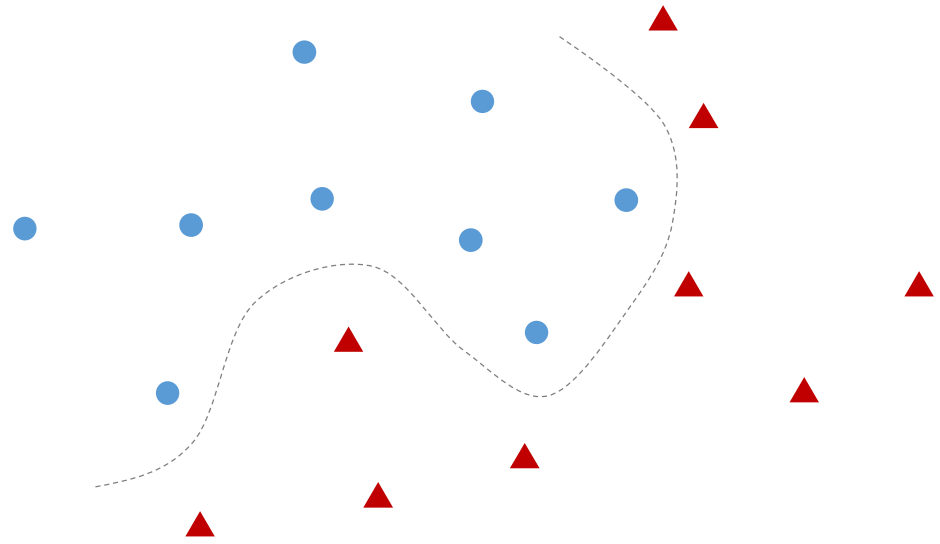
Penalty parameter C of the error term.

kernel : string, optional (default='rbf')

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is

Support Vector Machine

- 경계면이 non-linear 이어야 하는 경우도 있습니다.
- Kernel 을 이용한 SVM은 non-linear 경계면을 학습합니다.
  - (간단히 비유하면) kernel은 두 점 간의 유사도와 비슷합니다.

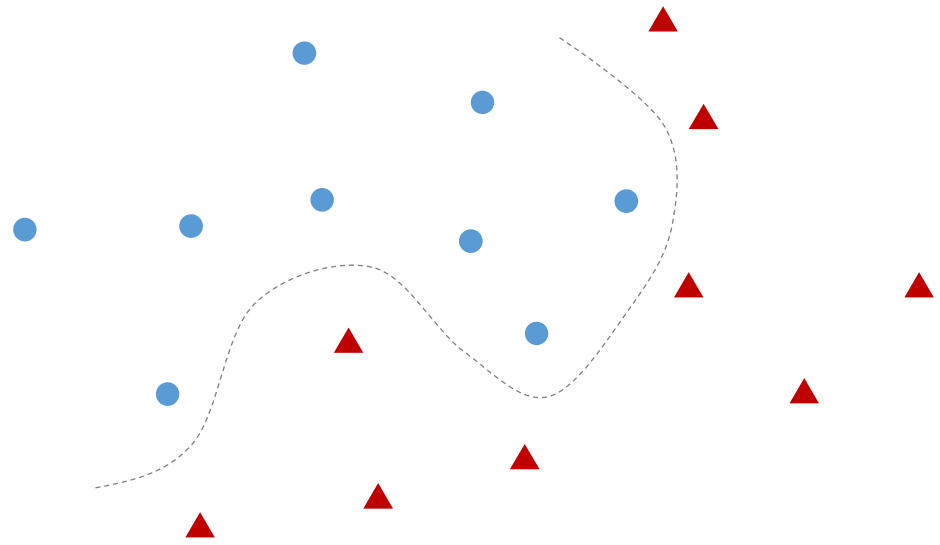


Support Vector Machine

- Kernel SVM 의 판별식은 유사도를 정의하는 부분만 바뀝니다. (RBF kernel)

$$f(q) = \sum_{i \in SV} \alpha_i \, y_i \, K(x_i, q) + b$$

$$K(x_i, q) = c * exp\left(-\frac{(x_i - q)^2}{\sigma^2}\right)$$

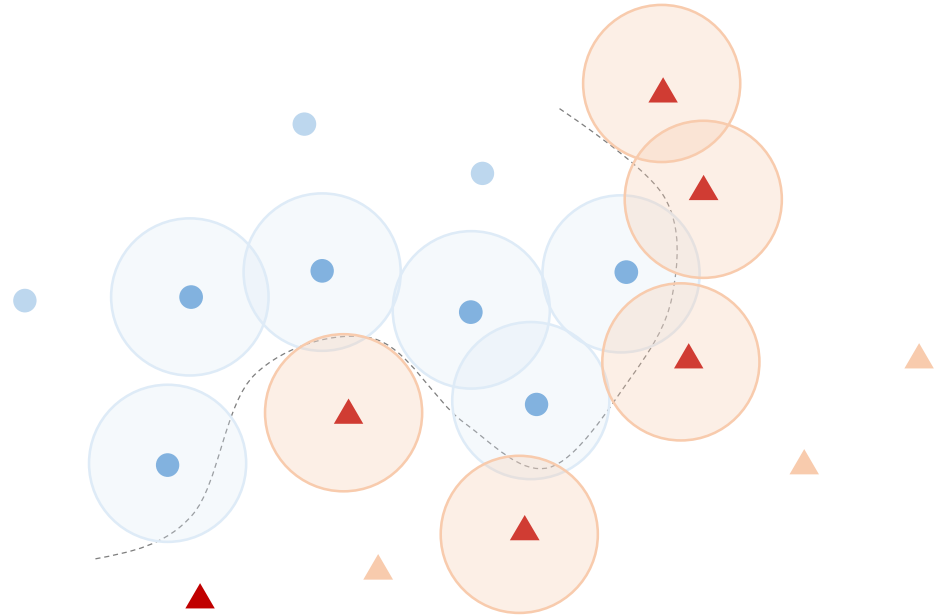


Support Vector Machine

- RBF kernel은 SV 와 Gaussian으로 얼마나 가까운지를 정의합니다.

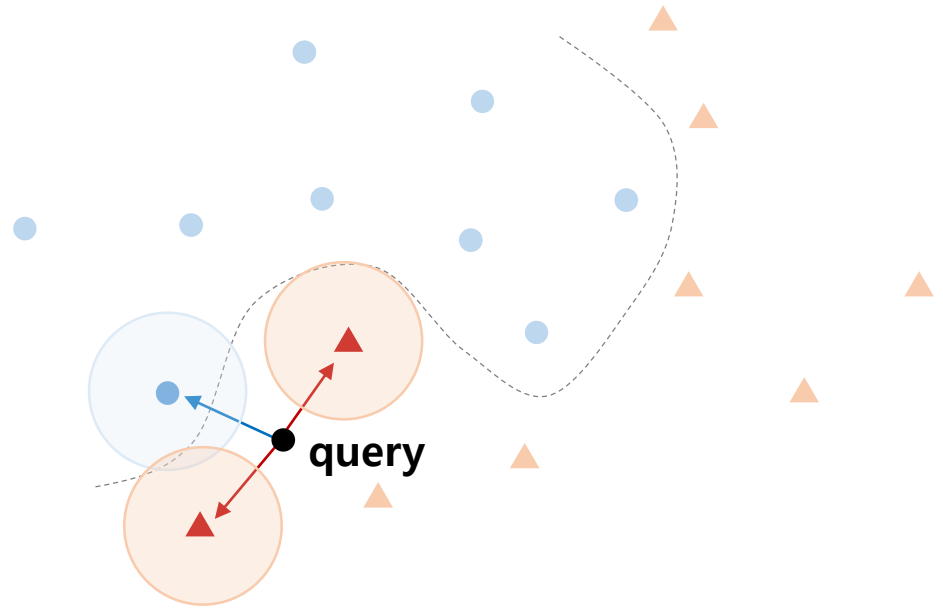
$$f(q) = \sum_{i \in SV} \alpha_i \, y_i \, K(x_i, q) + b$$

$$K(x_i, q) = c * exp\left(-\frac{(x_i - q)^2}{\sigma^2}\right)$$



Support Vector Machine

- 한 query points 에 대하여 SV 와의 유사도를 계산할 수 있습니다.
- 대부분의 SV 는 query 와 멀어 큰 영향을 주지 못합니다.



LASSO Regression

- LASSO Regression 은 유용한 feature (term) 을 선택하였습니다.

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

T11, T13, T14를 이용하면 y=1을 완벽히 인식할 수 있고,
이 문제는 {T1, T2, T3, T11, T13, T14}만 이용해도 잘 풀림

Support Vector Machine

- SVM 은 대표적인 data (document) 를 선택합니다.

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

Support Vector Machine

- 문서 분류는 특정 단어가 포함되었느냐가 중요합니다.
- 내적 (inner product) 는 공통된 단어 유무를 잘 표현하므로, linear kernel 만으로도 문서 분류에서 좋은 성능을 보입니다.

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

Support Vector Machine

- Linear SVM 의 $x_i^T q > 0$인 벡터들은 한 개 이상의 공통된 단어가 있습니다.
  - 하나의 단어라도 공통이면 SV 와 같은 class일 가능성이 증가
  - 적은 수의 SV 로도 많은 공간을 표현할 수 있습니다.

$$f(q) = \sum_{i \in SV} \alpha_i y_i \left(x_i^T q \right) + b$$

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											

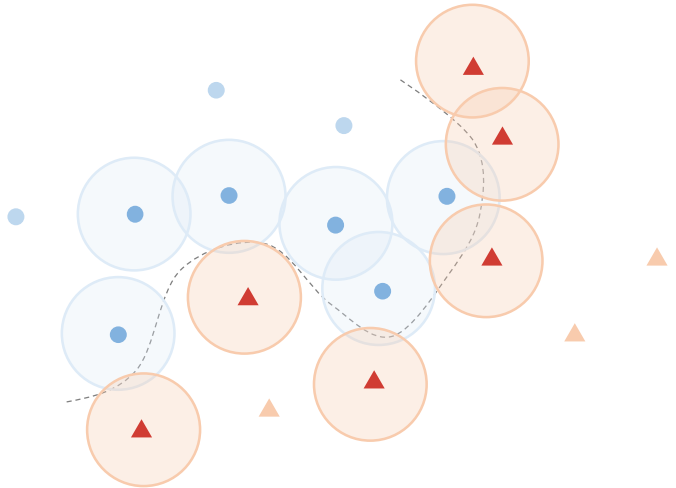
Support Vector Machine

- Euclidean 에 기반한 RBF kernel 은 공통된 단어를 확인하지 않습니다.
- RBF kernel 은 문서 공간보다 더 복잡한 공간을 표현하는데 적합합니다.

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

Support Vector Machine

- RBF kernel 은 한 개의 SV 가 표현하는 (커버하는) 공간이 작습니다.
  - 많은 수의 support vectors 가 필요함
  - 두 벡터 $\boldsymbol{x_i}$, $\boldsymbol{q}$ 가 공통된 단어가 없어도 유사도가 표현됩니다.
  - 문서 판별에는 적합하지 않습니다.



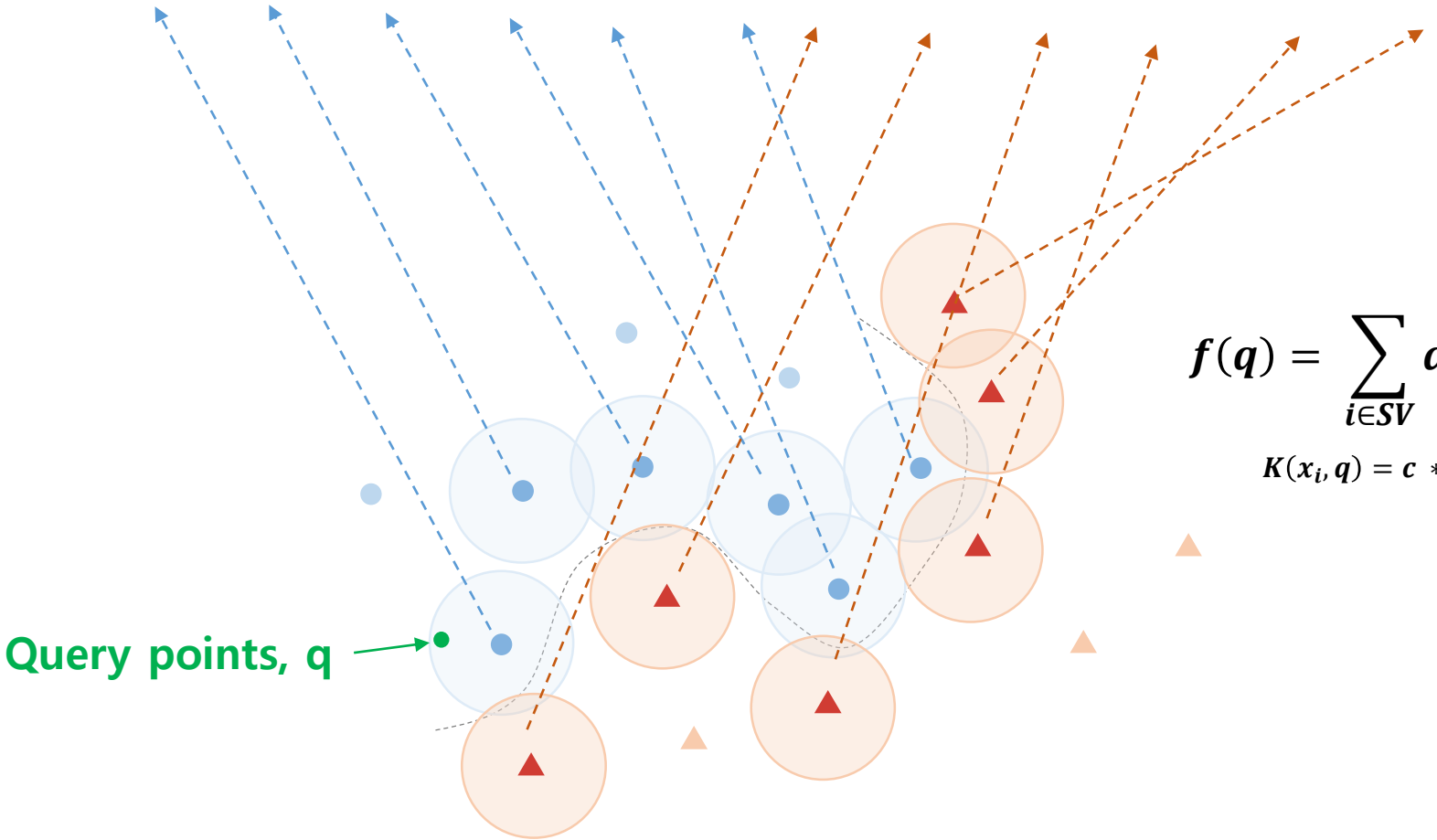
Support Vector Machine

- SVM은 데이터를 SV 개수의 차원 벡터로 변형하는 것과 같습니다.
- Kernel trick 은 non-linear 한 데이터 공간을 SV를 이용하여 linear 공간으로 변환합니다.
  - Query vector q 는 SV 와의 유사도 벡터 $\boldsymbol{K}(\boldsymbol{x_i}, \boldsymbol{q})$ 로 표현됩니다.
  - $\boldsymbol{K}(\boldsymbol{x_i}, \boldsymbol{q})$ 벡터가 가중치 벡터 $\boldsymbol{\alpha}$ 와 내적이 되는 것과 같습니다

Support Vector Machine

Kernel vector(?) ~= Similarity vector(?), $K(x_i, q)$

SV_1	SV_2	SV_3	SV_4	SV_5	SV_6	SV_7	SV_8	SV_9	SV_{10}	SV_{11}	SV_{12}
0.3	0.0001	0	0	0	0	0.0001	0	0	0	0	0

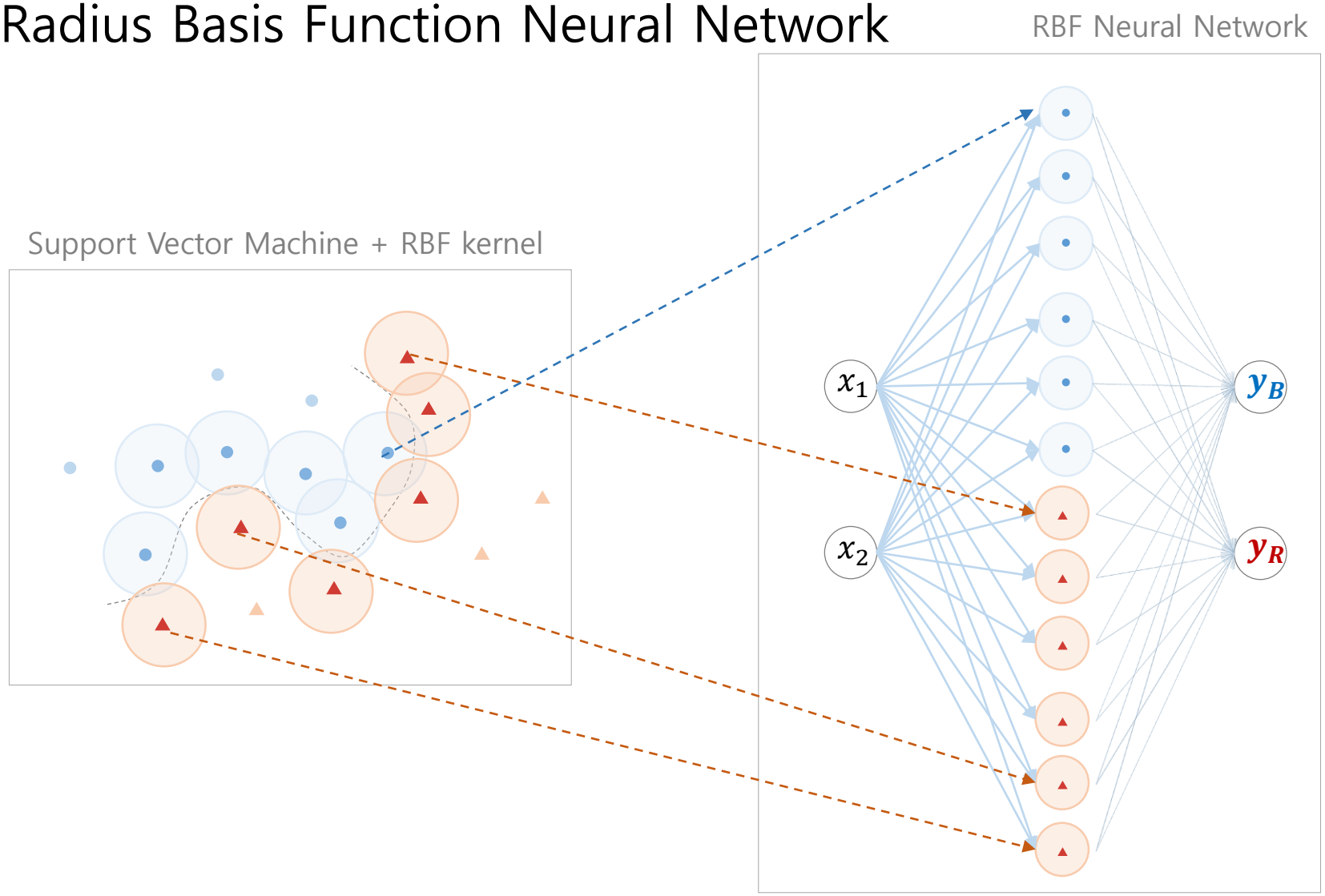


$$f(q) = \sum_{i \in SV} \alpha_i \, y_i \, K(x_i, q) + b$$

$$K(x_i, q) = c * exp\left(-\frac{(x_i - q)^2}{\sigma^2}\right)$$

Support Vector Machine

- Like Radius Basis Function Neural Network



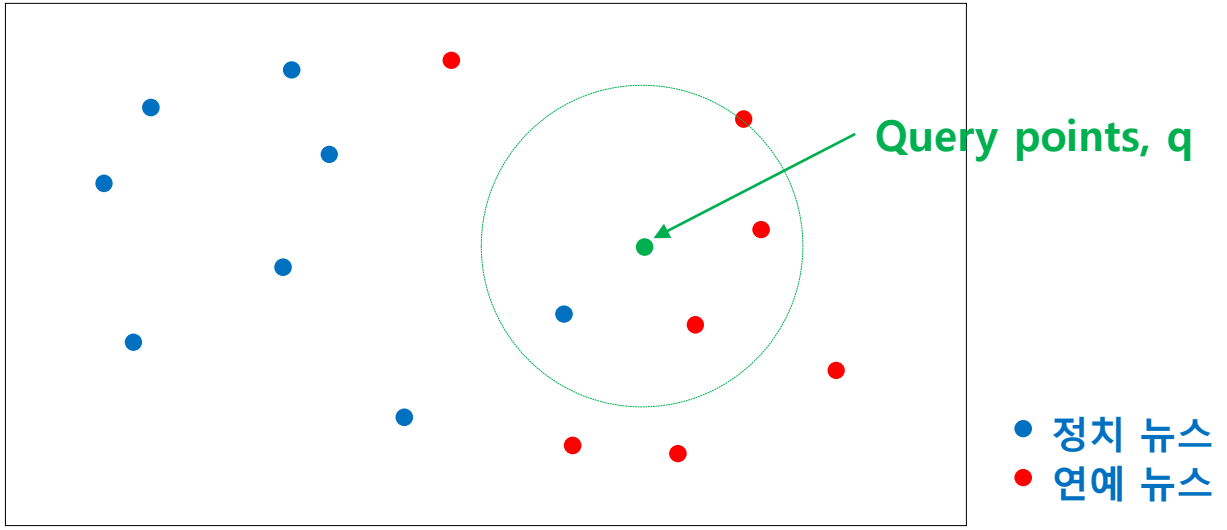
1. From Logistic Regression to Neural Network
2. Support Vector Machine (Linear, RBF kernel)
3. k - Nearest Neighbor Classifier
4. Naïve Bayes Classifier
5. Decision Tree

k-NN classifier

- *k*-NN 분류기는 query points와 가장 가까운 k개의 데이터를 찾은 뒤, 그 점들의 labels 중에서 숫자가 가장 많은 label을 return 합니다.
  - 거리에 반비례한 가중치를 이용할 수 있습니다.

$$\boldsymbol{y(q)} = \sum_{\boldsymbol{x} \in \boldsymbol{K_q}} \boldsymbol{w(q,x)} * \boldsymbol{y(x)}$$

$$\boldsymbol{w(q,x)} := \boldsymbol{exp}\big(-\boldsymbol{d(q,x)}\big)$$



k-NN classifier

- 최인접이웃 모델들은 임의의 유사도/거리 함수를 이용할 수 있습니다.
- 단어 빈도 벡터로 표현된 문서는 Euclidean 보다 Cosine 이 선호됩니다.
  - Euclidean 은 공통된 단어 유무를 표현하기 어렵습니다.

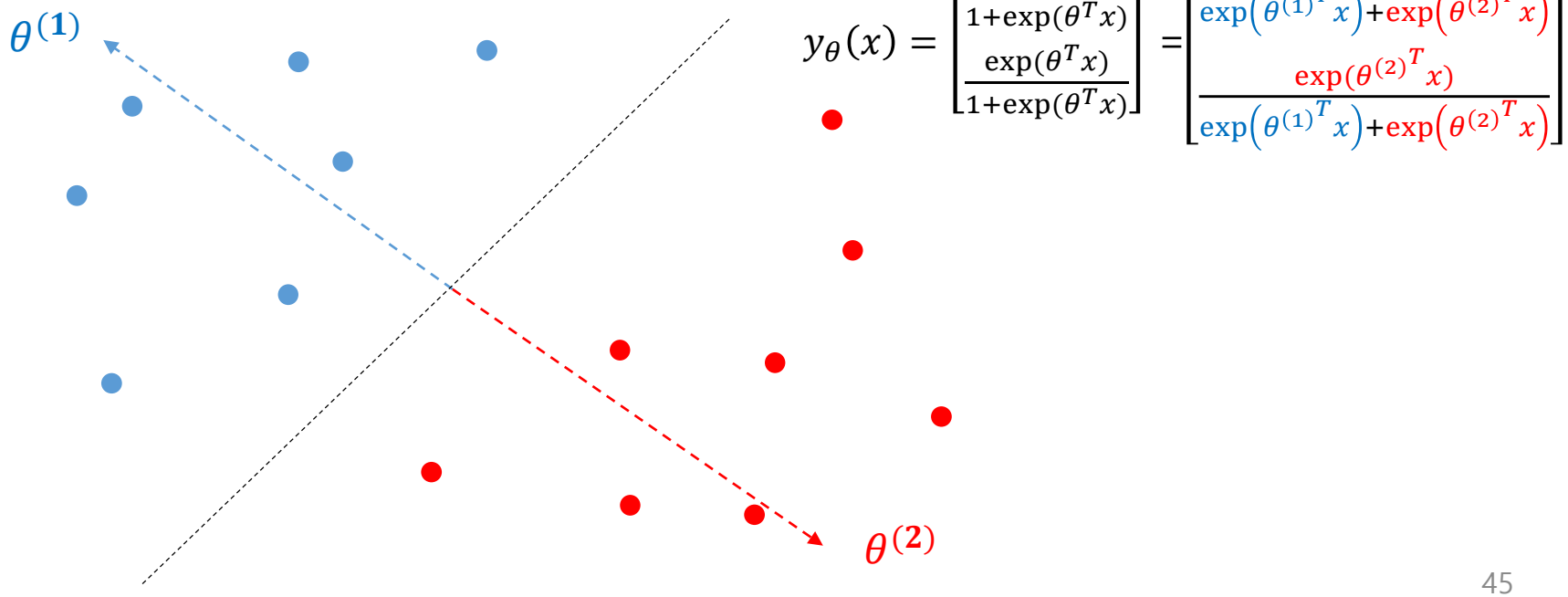
$$\boldsymbol{y}(\boldsymbol{q}) = \sum_{\boldsymbol{x} \in \boldsymbol{K_q}} \boldsymbol{w}(\boldsymbol{q}, \boldsymbol{x}) * \boldsymbol{y}(\boldsymbol{x})$$
$$\boldsymbol{w}(\boldsymbol{q}, \boldsymbol{x}) \coloneqq \boldsymbol{exp}\big(-\boldsymbol{d}(\boldsymbol{q}, \boldsymbol{x})\big)$$

k-NN classifier

- 간단한 모델이며, 직관적입니다.
- 그러나 최인접이웃 모델들은 두 가지 어려움이 있습니다.
  - 검색 비용이 큽니다. 데이터 개수만큼의 거리 계산을 합니다.
    - 그러나 이는 인덱싱 방법으로 해결이 됩니다.
  - 좋은 유사도 함수 / 좋은 벡터 표현이 전제되어야 합니다.
    - 데이터의 벡터 표현만 좋다면 최인접 이웃은 매우 유용한 방법입니다.

k-NN classifier

- Softmax regression 도 대표 벡터 $\theta^{(i)}$ 와의 1-NN 입니다.
  - Query vector 와 가장 가까운 $\theta^{(i)}$ 로 판별을 합니다.



k-NN classifier

- Softmax regression 의 클래스 개수 k 가 매우 크면, 판별 비용도 큽니다.
- 벡터 인덱싱은 이때에도 도움이 됩니다.
  - Softmax regression 은 내적이 가장 큰 대표벡터 $\theta^{(i)}$ 를 찾으면 됩니다.

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

1. From Logistic Regression to Neural Network
2. Support Vector Machine (Linear, RBF kernel)
3. k - Nearest Neighbor Classifier
4. **Naïve Bayes Classifier**
5. Decision Tree

Naïve Bayes Classifier

- Naïve Bayes 역시 문서 분류에 좋은 성능을 보여줍니다.
- Bayes rules을 이용하여 $\boldsymbol{P(y \mid x_1, x_2, \ldots, x_n)}$를 계산합니다

$$\boldsymbol{P(y \mid x_1, x_2, \ldots, x_n) = \frac{P(y) \times P(x|y)}{P(x)}}$$

$$\boldsymbol{argmax_{y^*}\{P(y^* \mid x_1, x_2, \ldots, x_n)\} = argmax_{y^*}\{P(y^*) \times P(x|y^*)\}}$$

Naïve Bayes Classifier

- 주어진 단어 벡터 $[\boldsymbol{x_1 = 3, x_2 = 2, \dots, x_n = 0}]$ 가 나타날법한 클래스를 찾습니다.

$$\boldsymbol{P(y \mid x_1, x_2, \dots, x_n) = \frac{P(y) \times P(x|y)}{P(x)}}$$

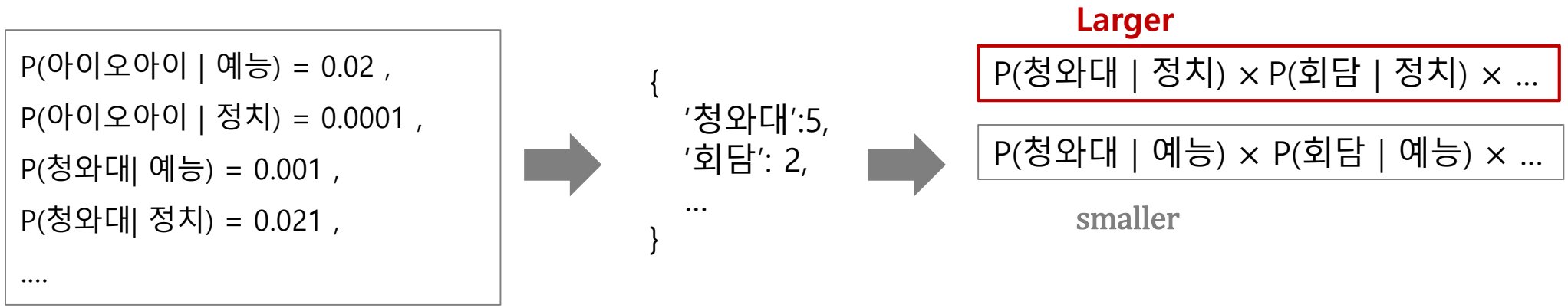
  - $\boldsymbol{P(y)}$: 문서 종류 y가 등장한 비율
  - $\boldsymbol{P(x|y)}$: 문서 종류 y에서 단어 빈도 벡터 $\boldsymbol{x}$ 가 만들어질 확률

Naïve Bayes Classifier

- 각 문서 종류 $\boldsymbol{y}$ 에서의 단어 비율 $\boldsymbol{P(x_i \mid y)}$의 누적곱을 이용합니다.

$$\boldsymbol{P(x|y) = \prod_{i=1\ to\ n} P(x_i \mid y)}$$

$$\boldsymbol{P(y)\ \times P(x|y) = P(y) \times \prod_{i=1\ to\ n} P(x_i \mid y)}$$



Naïve Bayes Classifier

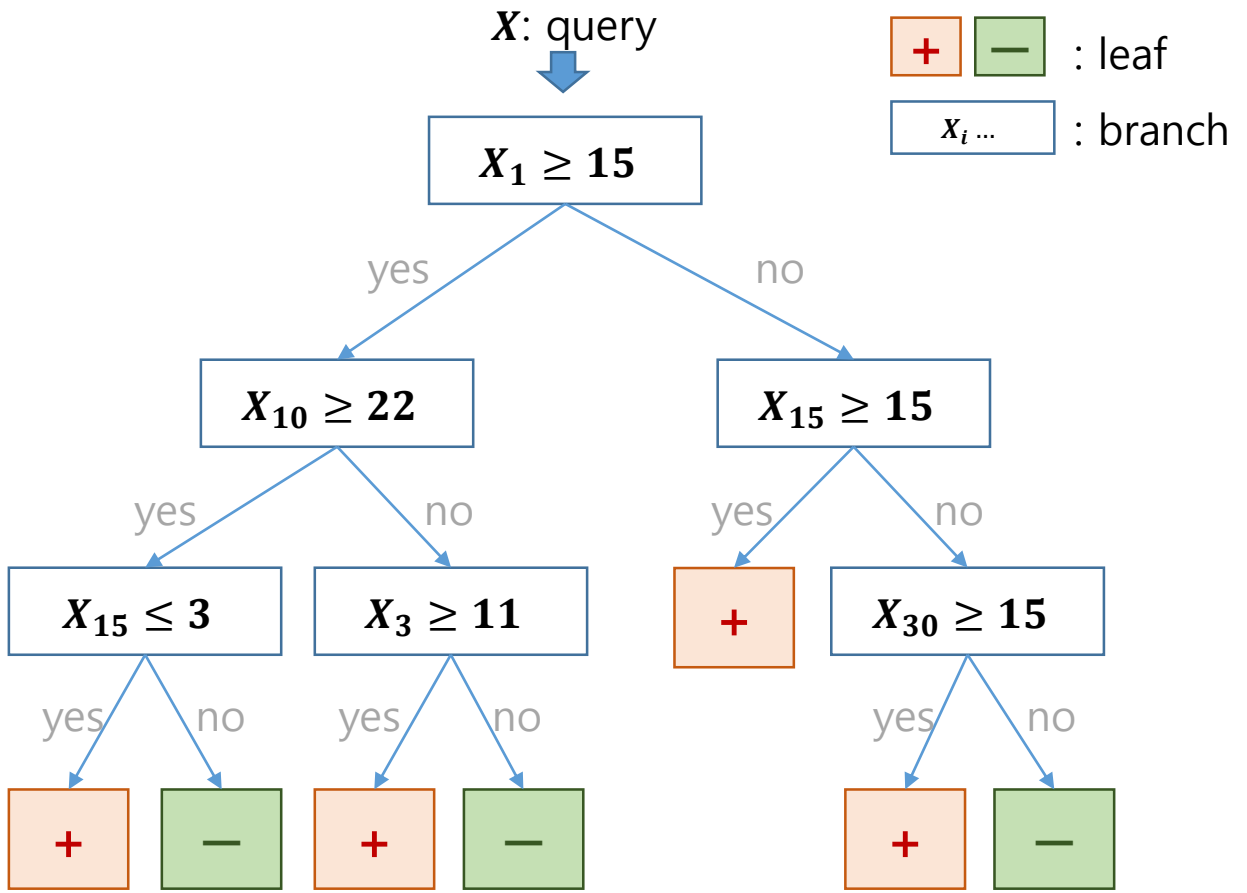
- 문서 분류에서는 ‘특정 단어가 있는가?’ 가 중요합니다.
  - Naïve Bayes classifier 는 이 가정과 공식이 잘 일치합니다.

- 학습 역시 빠릅니다.
  - 각 클래스 별 단어 확률 분포만 학습 (카운팅) 하면 됩니다.

1. From Logistic Regression to Neural Network
2. Support Vector Machine (Linear, RBF kernel)
3. k - Nearest Neighbor Classifier
4. Naïve Bayes Classifier
5. **Decision Tree**

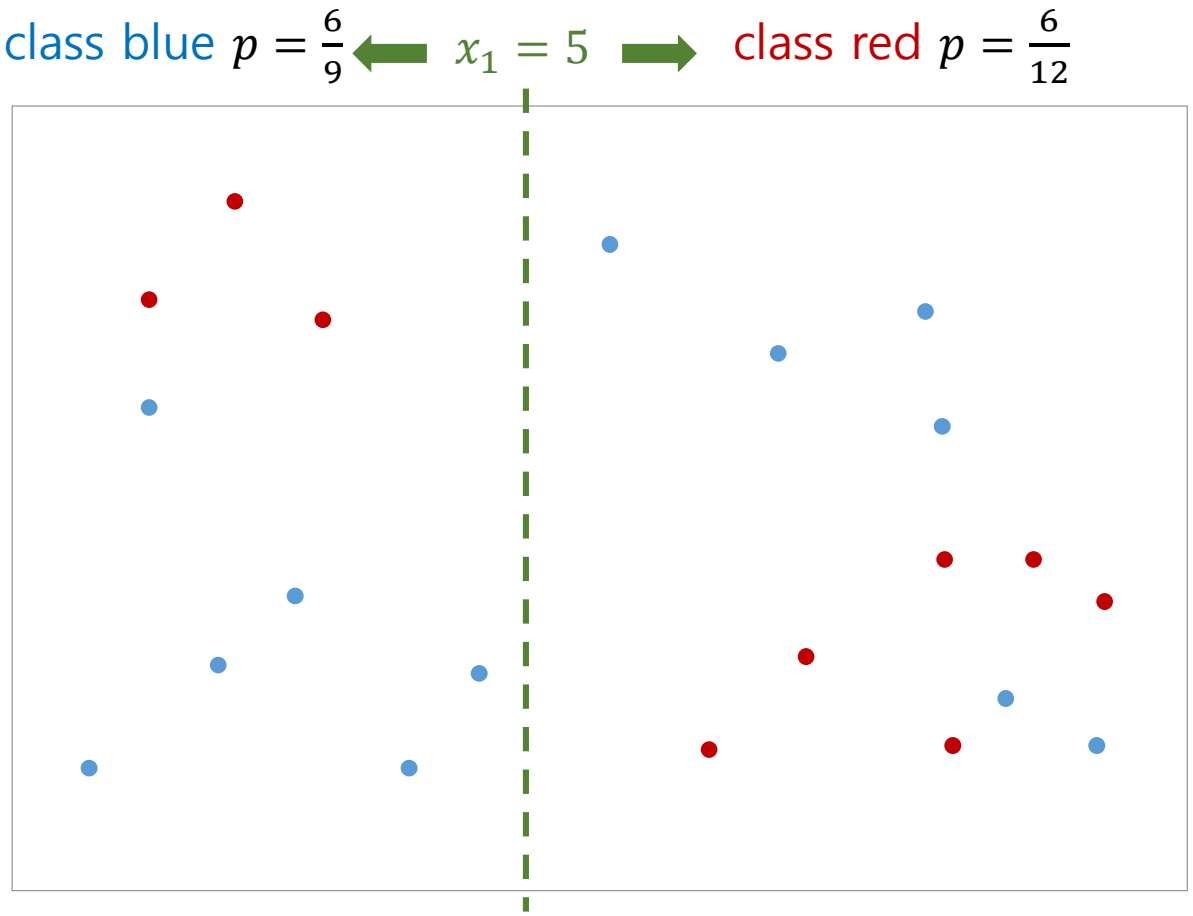
Decision Tree

- 여러 단계의 decision node로 이뤄진 '플로우차트' 같은 분류기입니다.



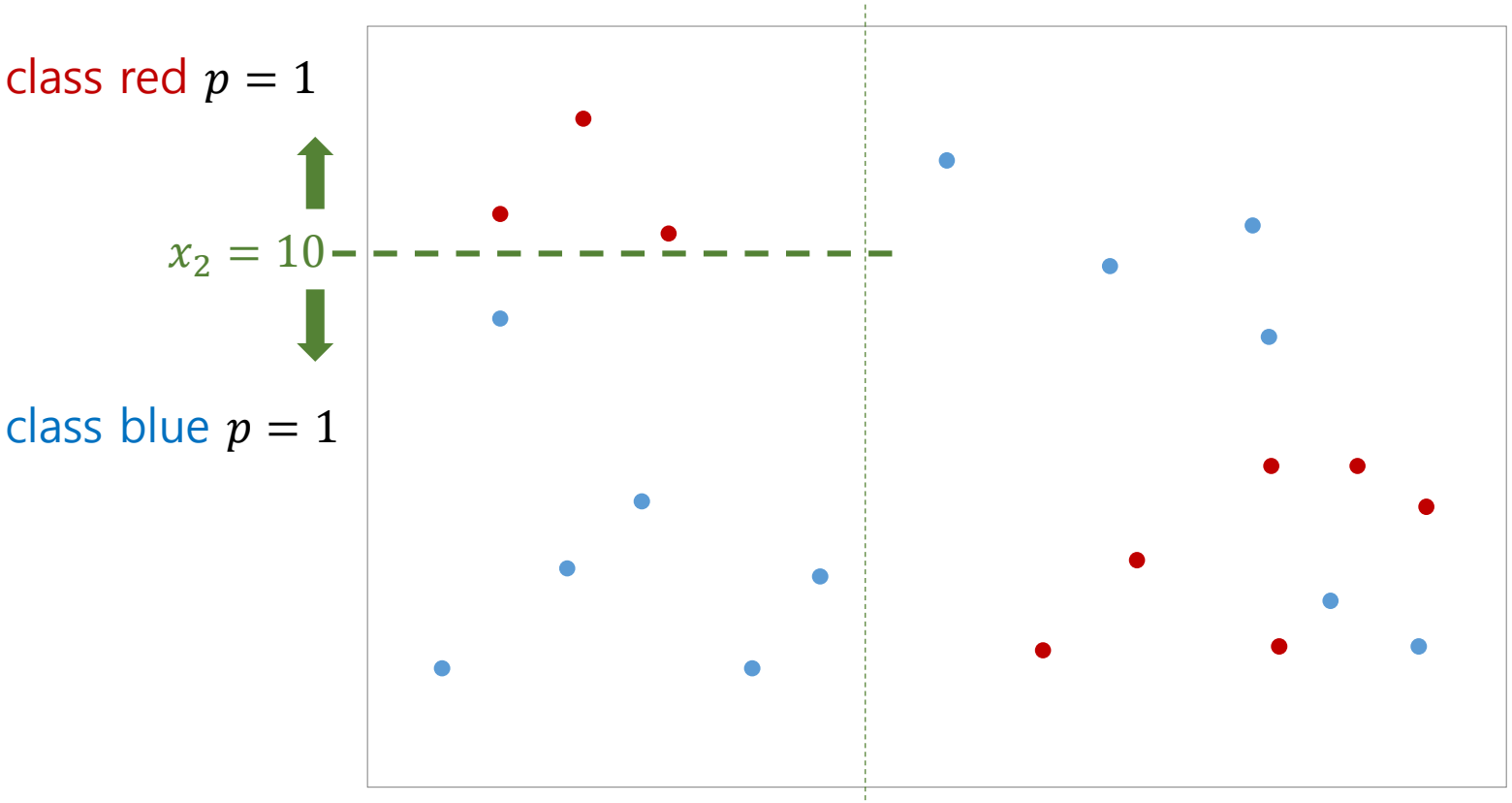
Decision Tree

- 의사결정나무는 재귀적으로 직사각형 (leaf)을 만듭니다.



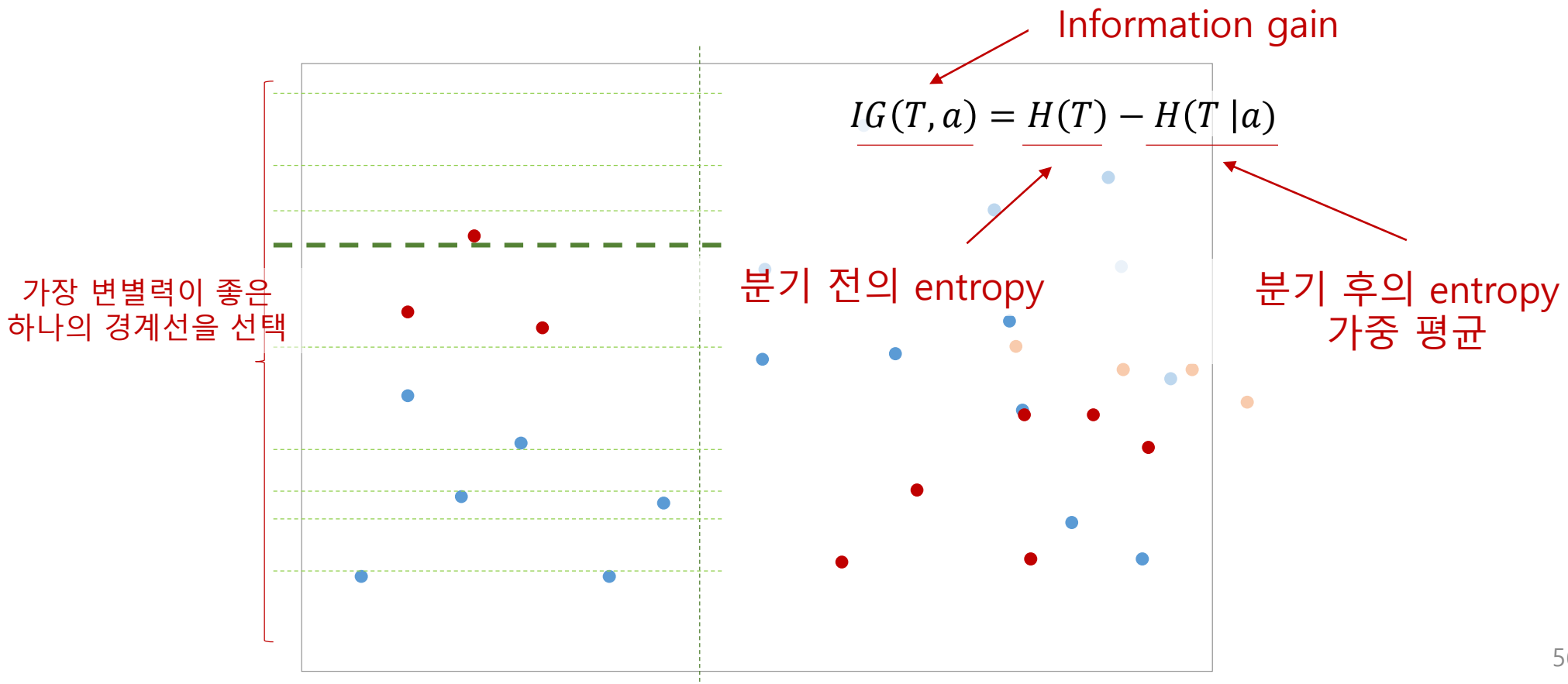
Decision Tree

- 의사결정나무는 재귀적으로 직사각형 (leaf)을 만듭니다.



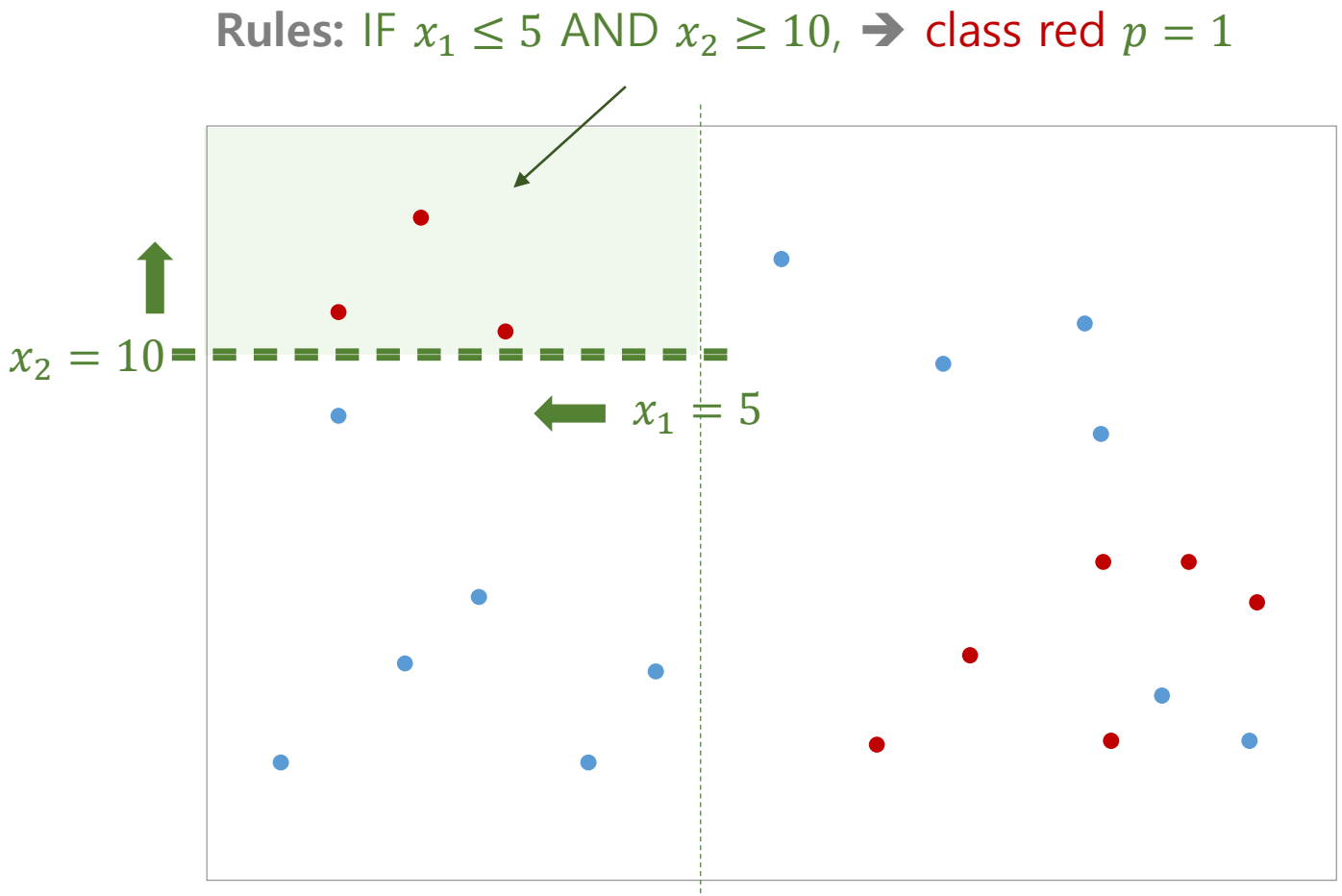
Decision Tree

- Information gain 등의 기준을 이용하여 매 branch 마다 가장 변별력이 좋은 변수와 경계선을 선택합니다.



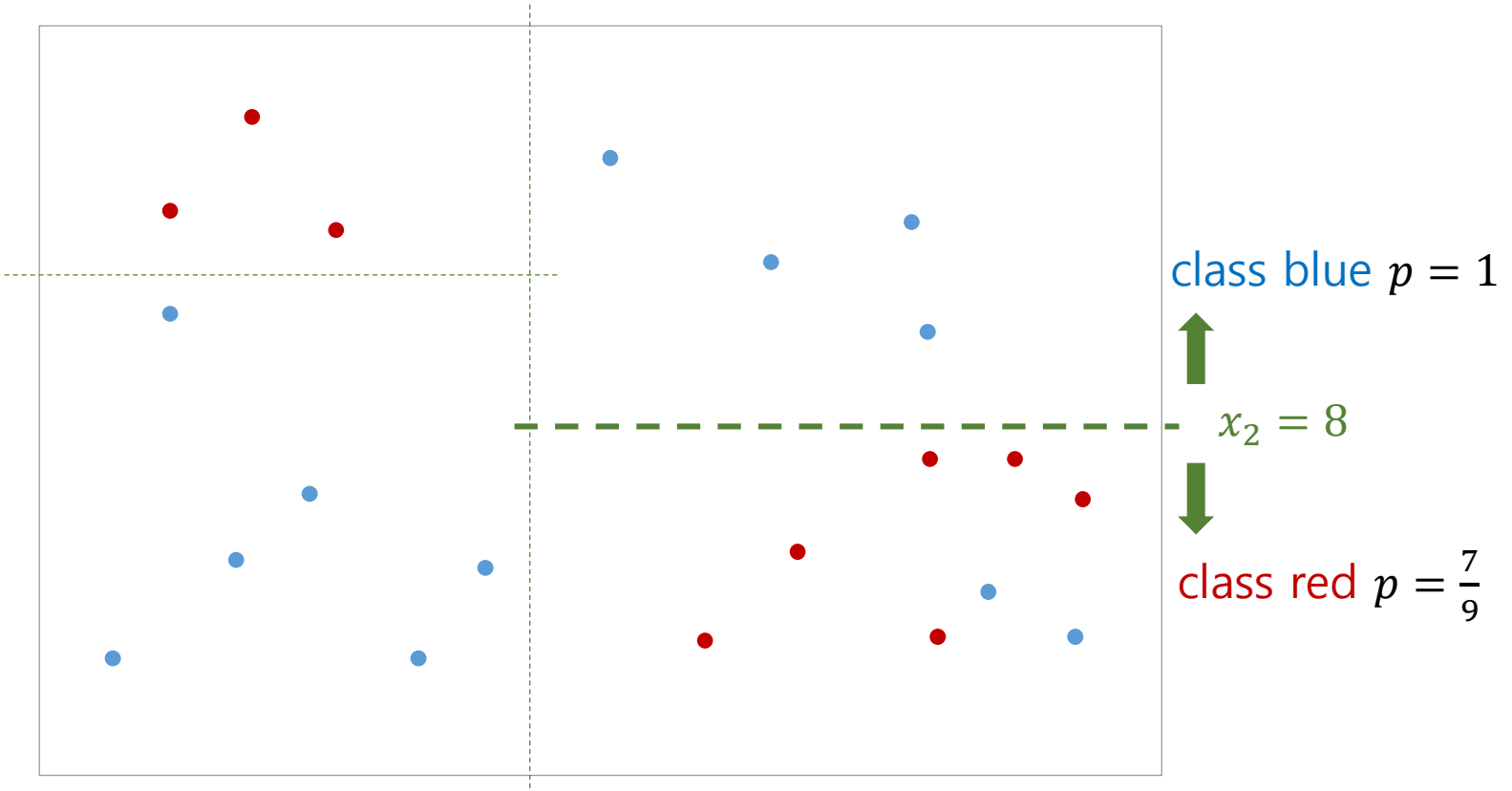
Decision Tree

- 의사결정나무의 각 지역은 규칙(rules)으로 표현이 됩니다.



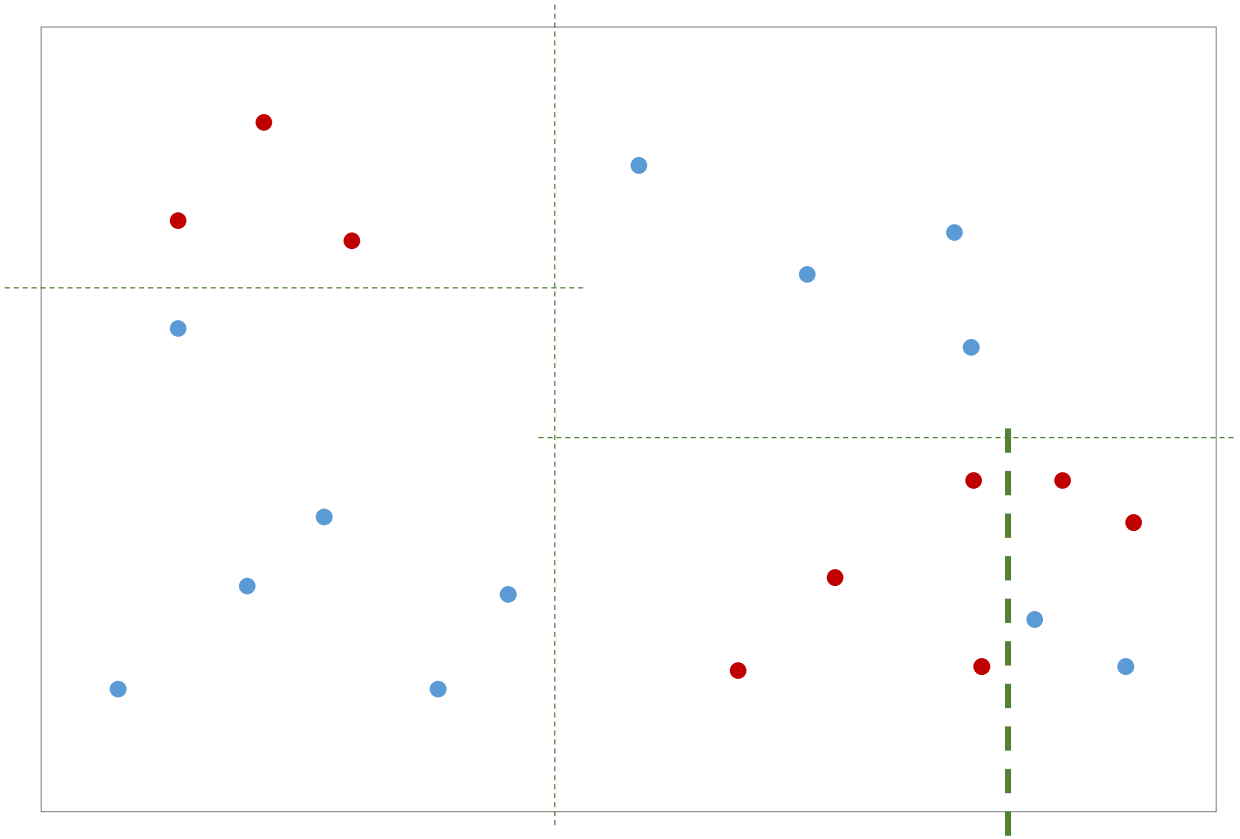
Decision Tree

- 의사결정나무는 재귀적으로 직사각형 (leaf)을 만듭니다.



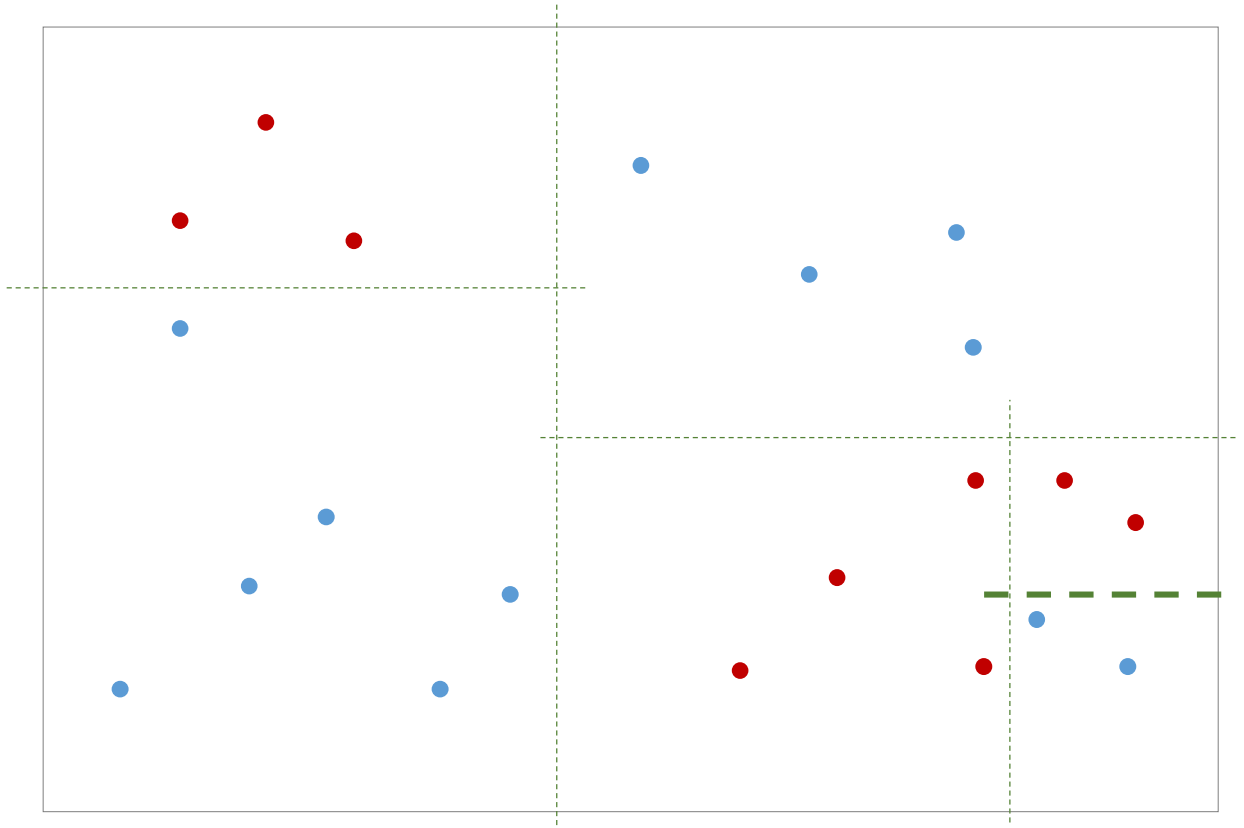
Decision Tree

- 의사결정나무는 재귀적으로 직사각형 (leaf)을 만듭니다.



Decision Tree

- 각각의 직사각형 공간들이 leaves node 입니다.



Decision Tree

- 의사결정나무의 branch 를 늘리는 과정을 나무가 자라난다 표현합니다.
- 나무의 성장에 제한을 걸 수 있습니다 (parameters)
  - Leaf 안의 데이터의 최소 숫자가 min_samples_split 이상
  - Branch의 깊이가 max_depth 이하
  - 사용하는 feature의 개수가 max_features 이하

Decision Tree

- 나무의 성장에 제한을 걸 수 있습니다 (parameters)
  - Leaf 안의 데이터의 최소 숫자가 **min_samples_split** 이상
  - Branch의 깊이가 **max_depth** 이하
  - 사용하는 feature의 개수가 **max_features** 이하

sklearn.tree.DecisionTreeClassifier¶

class sklearn.tree. DecisionTreeClassifier (*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False*) [source]

Decision Tree

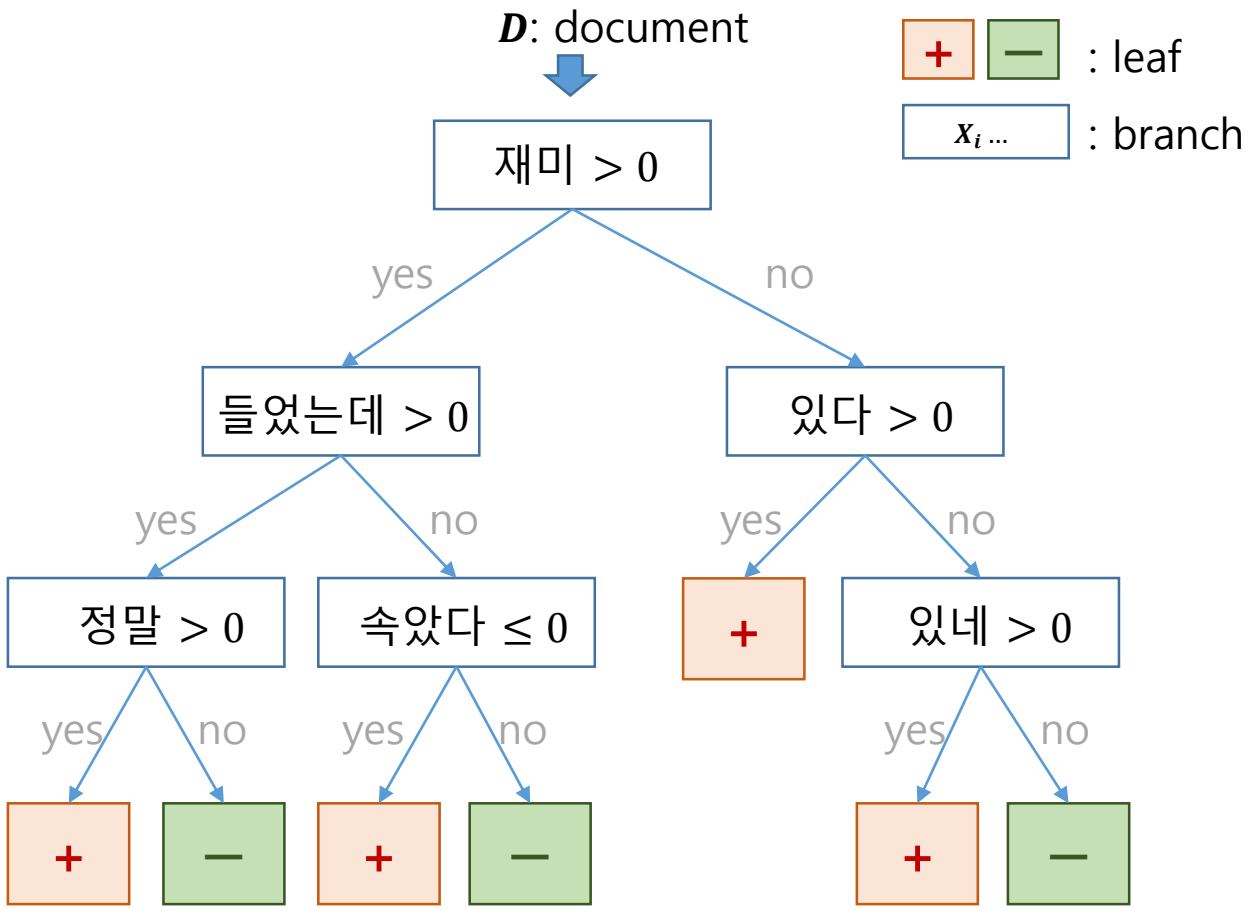
- 의사결정나무는 해석력이 있으며, 모델 피팅이 편리합니다.
  - 스캐일링, missing value 등의 전처리 과정에 영향이 적습니다.
  - (가능한) 적은 수의 분류에 중요한 변수를 선택합니다.
  - 선택된 변수와 경계값을 통하여 해석이 용이한 규칙이 도출됩니다.

Decision Tree

- 노이즈와 데이터의 분포에 민감합니다.
  - 값이 조금만 바뀌어도 branch 에 이용되는 변수가 바뀔 수도 있습니다.
- 비선형 관계를 표현하기 위하여 나무의 깊이가 깊어야 합니다.
  - 단어 빈도 벡터로 표현되는 문서 집합이 비선형인 경우는 드뭅니다.
- 동시에 하나의 feature (단어)만 고려 가능합니다.
  - 소수의 features 만 이용할 거라면 LASSO regression 이 문서 분류에 더 적합/효과적입니다.

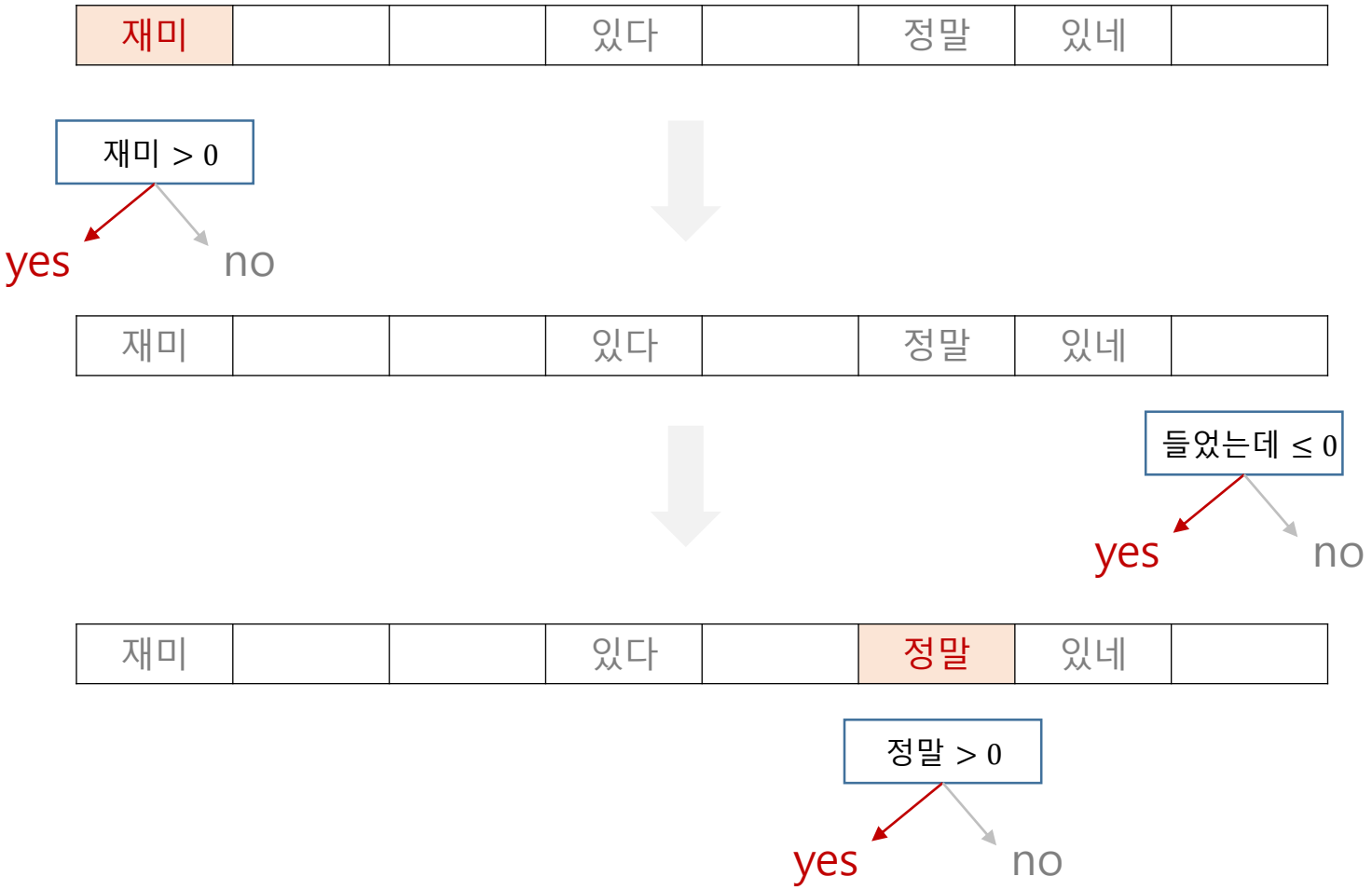
Decision Tree

- Bag of words로 표현되는 텍스트의 의사결정나무의 features는 단어



Decision Tree

- BOW는 고차원 벡터이기 때문에 각 단어를 순서대로 이용하는 것입니다.



Decision Tree

- 의사결정나무는 Sparse vector 에 과적합을 하거나 학습이 제대로 되지 않습니다.
  - 의사결정나무는 적은 수의 features (단어)로 데이터를 분류하려 합니다.
  - 문서의 길이가 짧으면 한 문서에 등장하는 단어의 종류가 적습니다.
  - 제대로된 의사결정을 못하는 문서가 많을 수 있습니다.

재미			있다		정말	있네	

Decision Tree

- 문서 분류의 핵심은 특정 단어 (혹은 n-gram)이 해당 문서에 있느냐 입니다.
  - 질 좋은 판단 기준 단어들을 많이 알면 알수록 좋습니다.
  - 그러나, 이는 의사결정나무가 추구하는 방향과 다릅니다.
- 특히, 문서마다 사용되는 단어가 다르다면 의사결정나무가 이용하는 단어의 수는 어자피 늘어납니다.