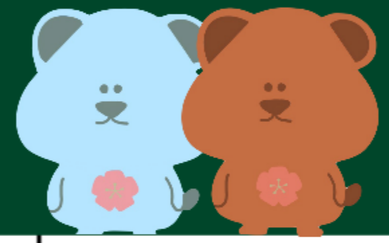


Neural Networks

Week3_발표자 : 임세영, 조서영

EWHA EURON

목차

01 Classification review & introduction

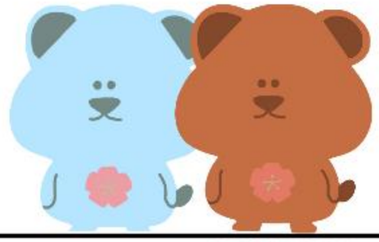
- Classification intuition
- Classification in NLP
- Supervised learning
- Loss function

02 Neural Network

- Overview
- Activation function
- Decision boundary

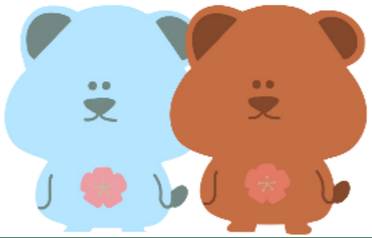
03 Window Classification

- Named Entitiy Recognition (NER)
- Softmax
- Gradient descent algorithm



EWHA EURON

01 Classification review & introduction

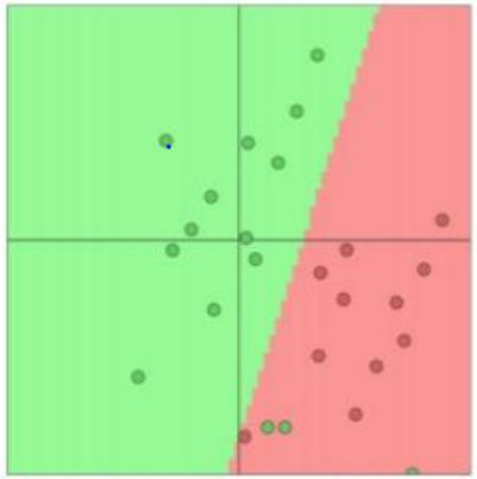


EWHA EURON

01 Classification review & introduction

#1 Classification intuition

Simple illustration case:
- Fixed 2D word vectors to classify
- Using softmax / logistic regression
- Linear decision boundary



Visualizations with ConvNetJS by Karpathy!

ML / Deep Learning 방법으로 분류

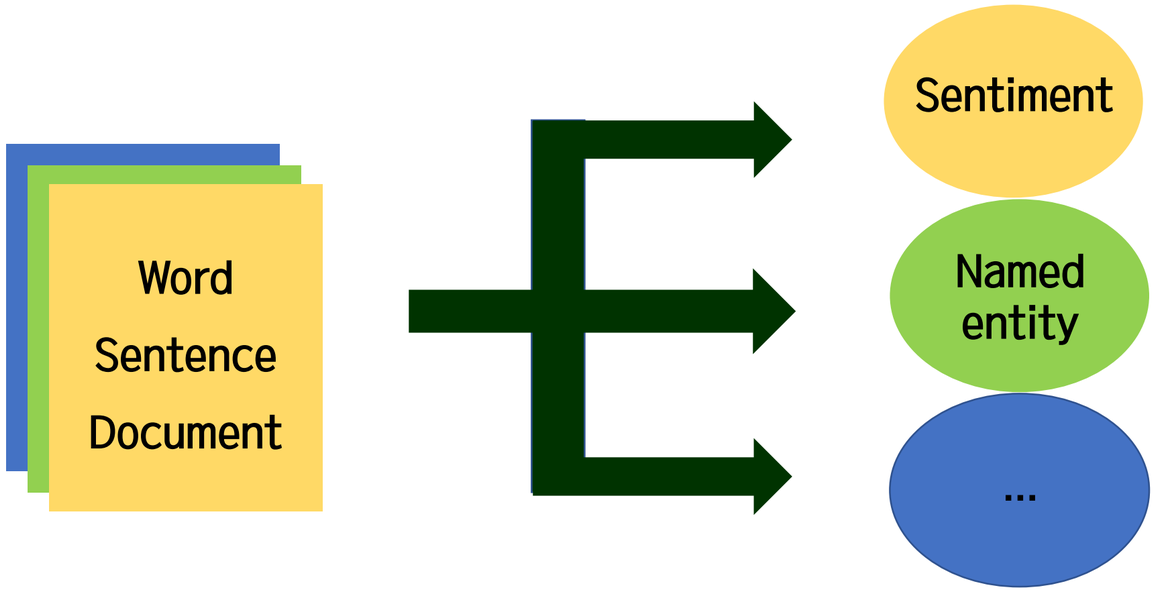
비슷한 output끼리 모이도록 경계를 긋는 것

> 전통적인 ML 접근에서는 softmax / logistic regression 을 이용해서
output의 class를 구분할 경계선을 결정하는 것을 의미

01 Classification review & introduction

#2 Classification in NLP

INPUT DATA (X)



CLASS (Y)

< 모델의 학습 >

Decision boundary 를 결정할 weight 학습

01 Classification review & introduction

#3 Supervised learning

Supervised learning

입력 데이터(X), 결과 데이터(Y) 모두 이용

선형 회귀, 로지스틱 회귀,
Ridge regression, LASSO,
K-NN(K-최근접이웃)알고리즘,
SVM(서포트벡터머신)

- **Classification (이산)**
- **Regression (연속)**

Unsupervised learning

입력 데이터셋(X) 만 이용

모델로는 K-Means
Clustering(K-평균 알고리즘),
DBSCAN(밀도 기반 클러스터링),
Kernal Density Estimation

EWHA EURON

01 Classification review & introduction

#3 Supervised learning

1) N개의 input-output으로 구성된 훈련세트 (Train set : D_{train}) 준비

$$D = \{(x_1, y_1), \cdots, (x_N, y_N)\}$$

2) 검정 세트 (Validation set : D_{val}) , 테스트 세트 (Test set : D_{test}) 준비

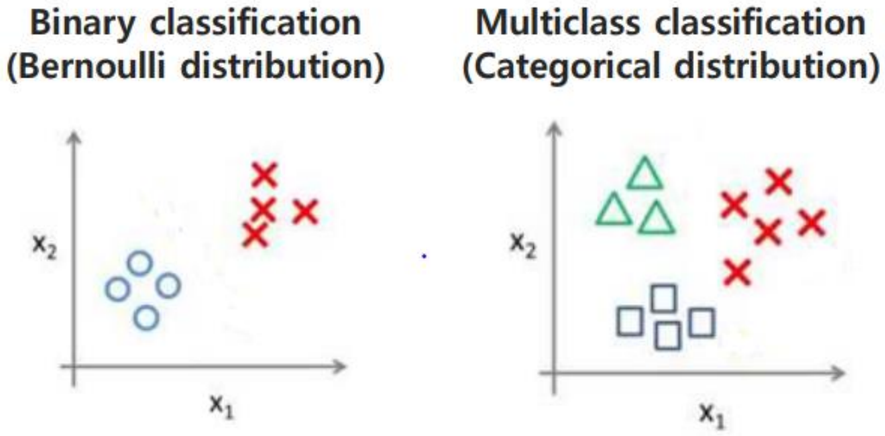
- 기존에 보지 않았던 데이터로 모델 및 성능 평가

>>> ML 모델 (M)의 output 값과 실제 값 (y)를 평가할 손실 함수 (Loss function) 준비

$$L(M(x), y)) \geq 0$$

01 Classification review & introduction

#4 Loss function



Output (y) > Probability (y')

- 분류를 확률로 바꾸어 확률분포로 표현

$$f_\theta(x) = ? \quad \rightarrow \quad p(y = y'|x) = ?$$

모델이 출력한 조건부 확률분포(Conditional distribution)과 훈련 샘플의 확률분포가 같아지도록 학습

- 최대 우도 추정(Maximum Likelihood Estimator) : 모든 훈련 샘플의 확률 최대화

$$arg\max_\theta log\, p_\theta(D) = arg\max_\theta \sum_{n=1}^{N} log\, p_\theta(y_n|x_n)$$

Loss function = 음의 로그확률(Negative Log-probabilities)의 합

$$L(\theta) = \sum_{n=1}^{N} l(M_\theta(x_n), y_n) = -\sum_{n=1}^{N} log\, p_\theta(y_n|x_n)$$

01 Classification review & introduction

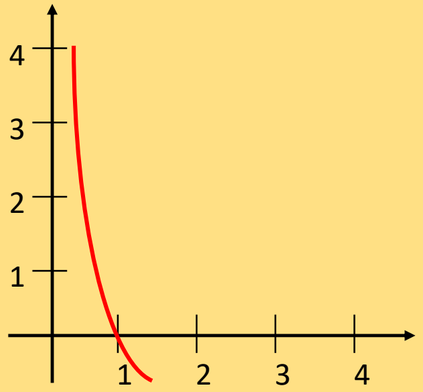
p: 실제 확률 분포
q: 예측한 확률 분포

#4 Loss function

1. Entropy

정보이론) 동등한 가능성이 여러 개 생각되는 경우 가능성의 하나를 지정하는 것

$$H(P) = H(x) = -\sum_{x} P(x) \log P(x)$$



p와 q **차이 최소화**

2. Cross Entropy

$$H(P,Q) = H(P) + D_{KL}(P||Q)$$

$$H(P,Q) = -E_{X\sim P}[\log Q(x)] = -\sum_{x} P(x) \log Q(x)$$

KL Divergence
: 확률분포 차이를 계산

$$D_{KL}(P||Q) = E_{X\sim P}\left[\log \frac{P(x)}{Q(x)}\right]$$

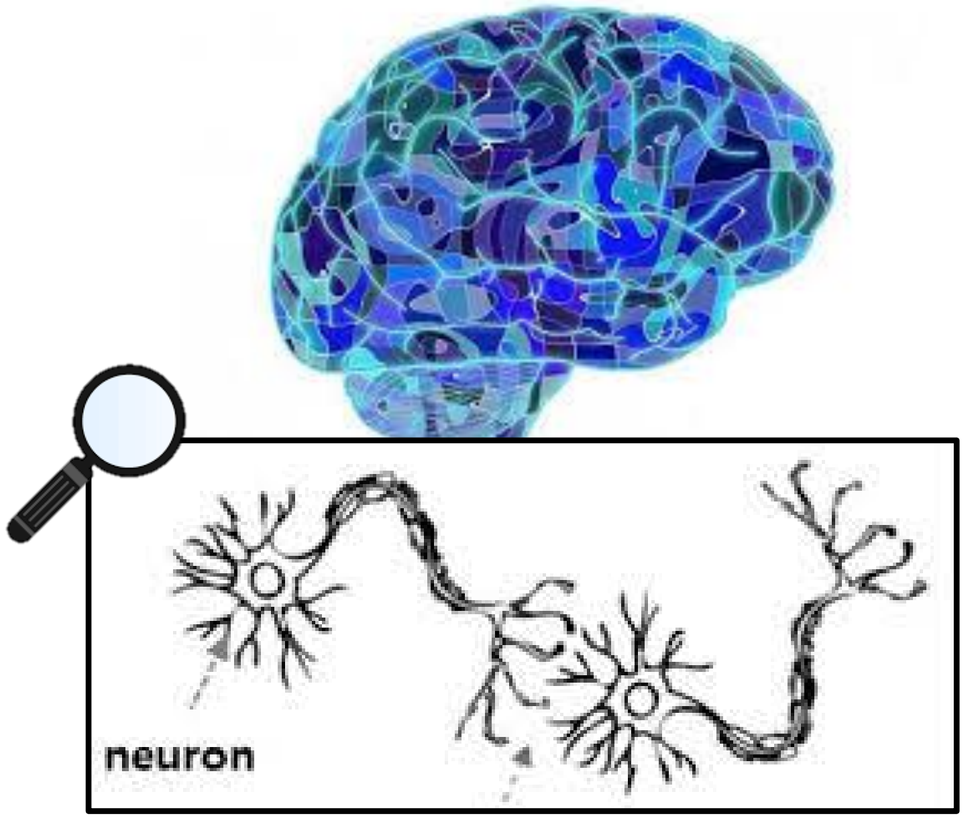
$$= E_{X\sim P}[\log P(x) - \log Q(x)]$$

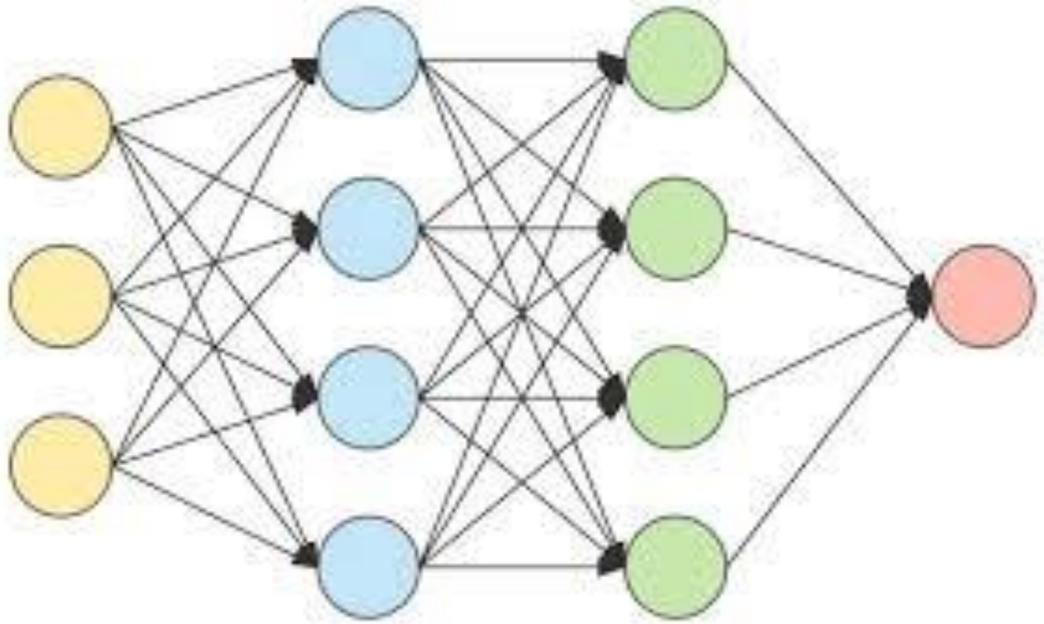
02 Neural Network

EWHA EURON

02 Neural Network

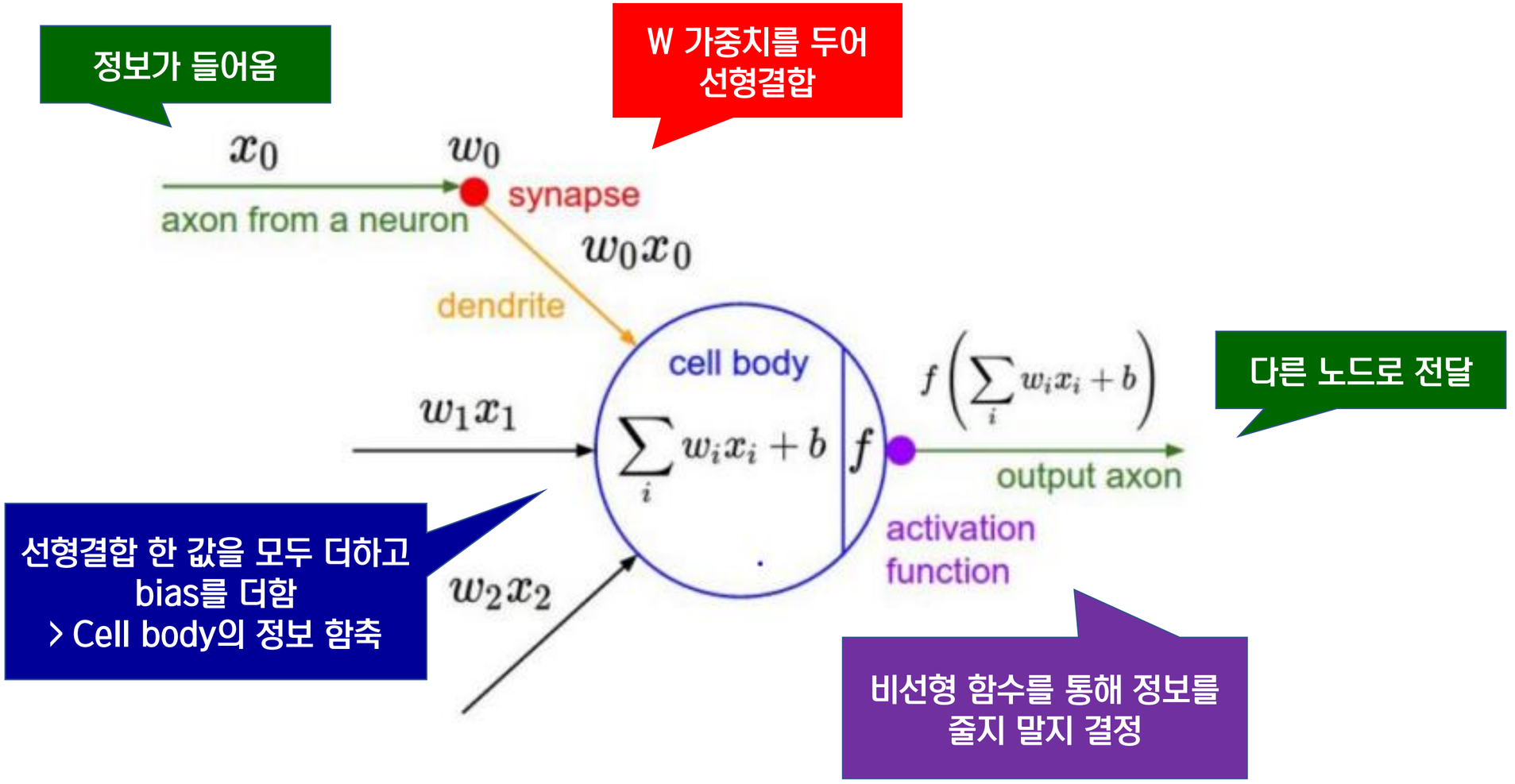
#1 Overview





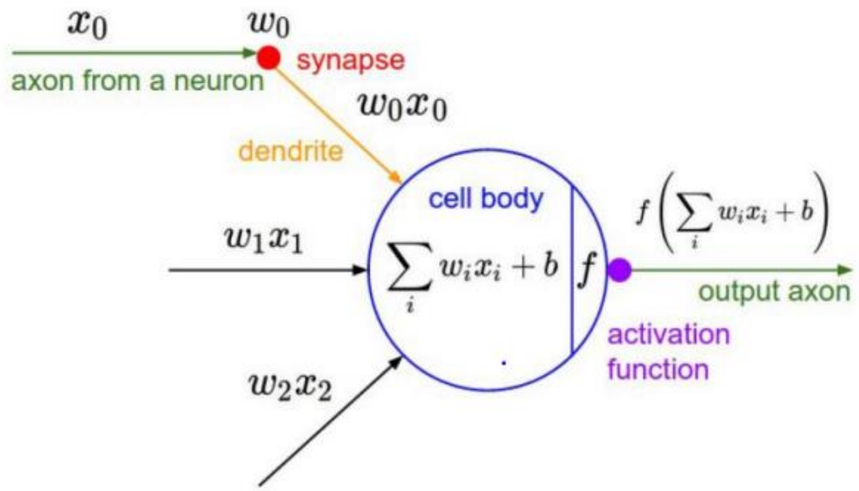
02 Neural Network

#1 Overview

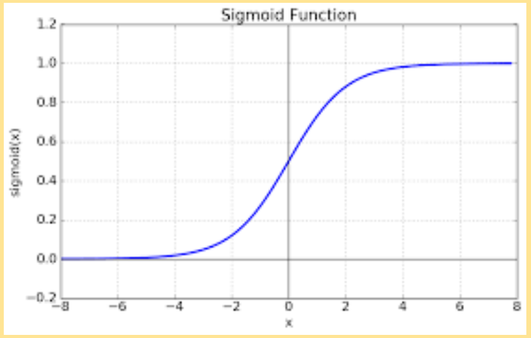


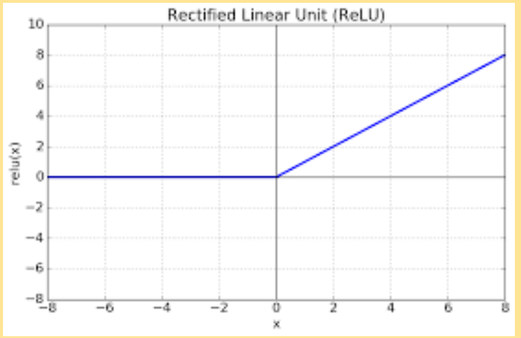
02 Neural Network

#2 Activation function



- "비" 선형함수 (Sigmoid, Relu 등)





- Non-linear decision boundary 학습 가능

- Layers 쌓을 수 있음

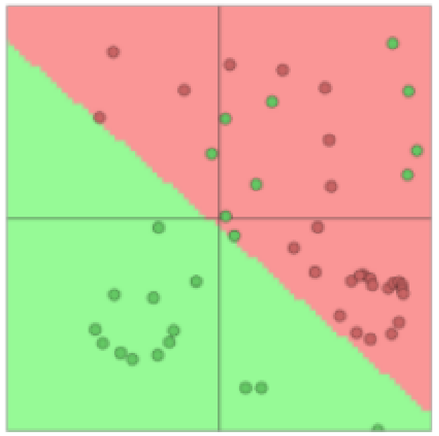
$$W_1 W_2 x = Wx$$

Activation이 선형일 경우
layer를 쌓아도 linear transform

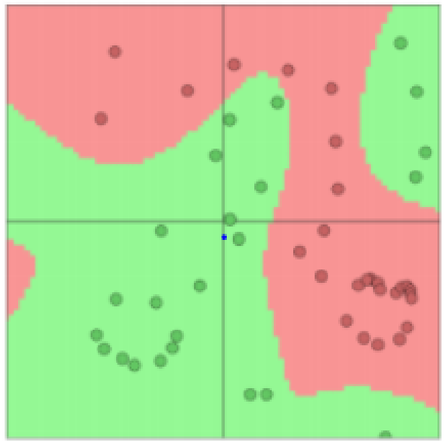
02 Neural Network

#3 Decision boundary

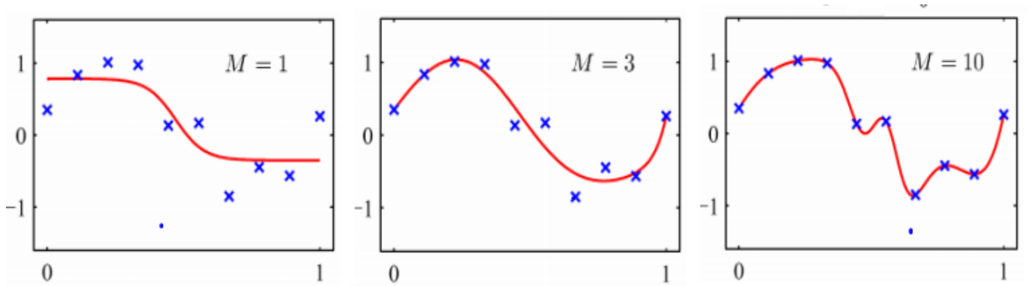
선형 경계면
ex) logistic regression



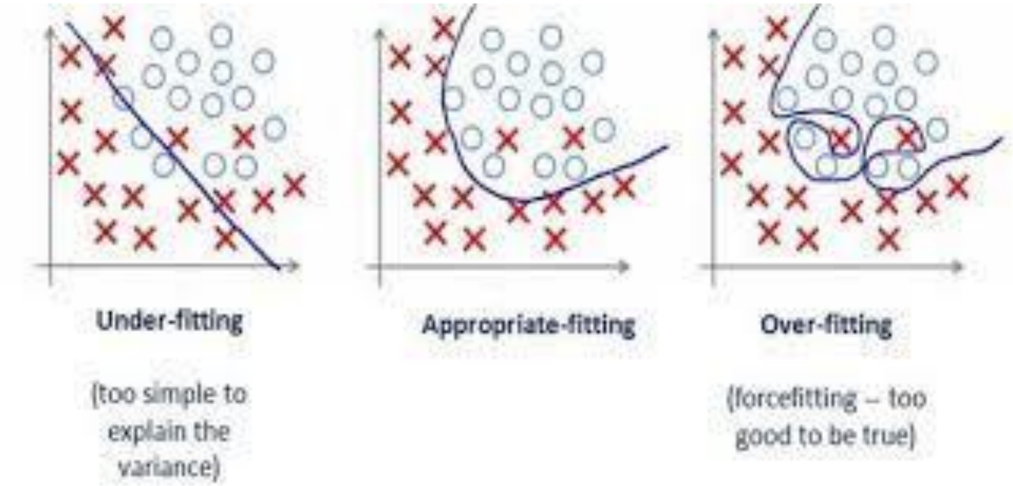
비선형 경계면
ex) neural network



층이 늘어날수록 실제 데이터의 클래스를 더 잘 맞춤



하지만 over-fitting을 주의해야 함

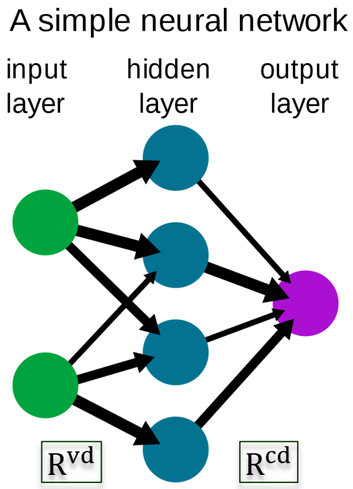


02 Neural Network

#3 Decision boundary

NLP에서 parameter (weight)와 word vectors (X) 함께 학습
ex) Word2Vec, Glove, ELMO, BERT

v: 단어 개수
d: hidden layer의 nod 개수
c: 최종 class 개수
vd + cd = parameter 개수



$$\nabla_\theta J(\theta) = \begin{bmatrix} \nabla_{W_{\cdot 1}} \\ \vdots \\ \nabla_{W_{\cdot d}} \\ \nabla_{x_{aardvark}} \\ \vdots \\ \nabla_{x_{zebra}} \end{bmatrix} \in \mathbb{R}^{Cd+Vd}$$

Very large number of parameters!

학습하는데 많은 시간이 걸리고
연산 양도 많아짐

>>> **pre-trained word vector**
학습이 빠르고 정확함
random으로 initializing해서 자체적으로 학습을 시킬 수 있음 (데이터의 단어 개수가 아주 많을 때)
but, pre-trained된 word vector를 사용해서 task에 맞게 fine-tuning하는 것이 더 효율적

기존 모델의 구조와 pre-trained된 가중치를 기반으로
우리의 목적에 맞게 학습된 모델의 가중치를 미세하게 조정하는 것

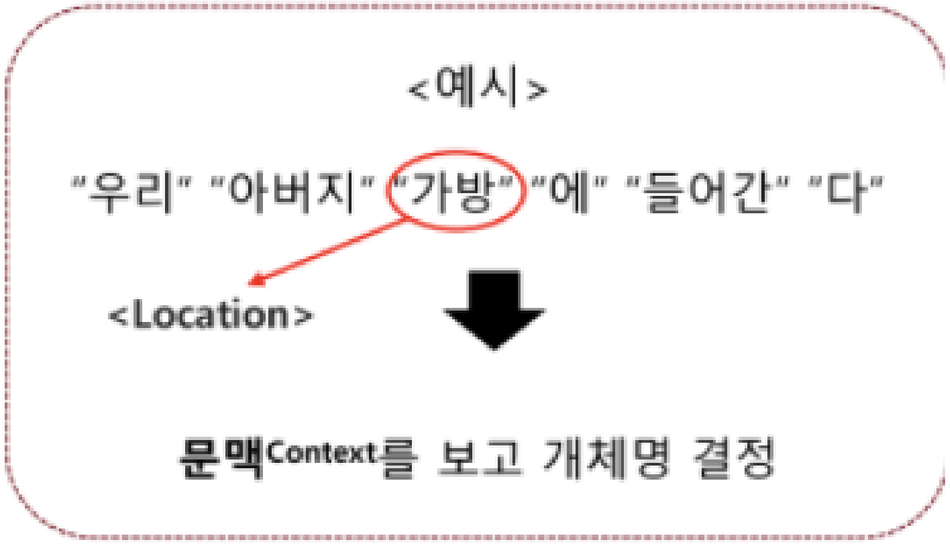
#03 Window classification

EWHA EURON

03 Window classification

What is NER?

- 특정 글에서 이름을 찾고 분류하는 것!
- 좀 더 쉽게 설명하면, 어떤 이름을 의미하는 단어를 보고는 그 단어가 어떤 유형인지를 인식하는 것을 말함
- 문장에서 Location, Person, Organization 등 개체명을 분류하는 방법론



03 Window classification

What is NER?

- Ex) Text: 유정이는 2018년에 골드만삭스에 입사했다.
  - 유정 – 사람
  - 2018년 – 시간
  - 골드만삭스 - 조직

03 Window classification

보기엔 쉬워 보이지만…

- ‘개체’의 범위를 구하는 것이 쉽지 않다.
  - 텍스트가 ‘첫 번째 국가 은행이 기부하다 ' 라고 했을 때, 개체를 ‘첫 번째 국가 은행 ' 으로 할 건지? 아니면 ‘국가 은행 ' 이라고 할 건지?
- 무엇을 ‘개체 ' 라고 인식할 건지?
  - ‘우리 은행 ' 이라고 했을 때 ‘나와 너가 일하는 은행 ' 인지 아니면 상호 ‘우리 은행 ' 인지?
- 알려지지 않았거나 새로운 개체를 분류하기 어렵다
- 개체 분류는 모호하고 맥락에 좌우된다
- 이렇게 모호한 단어들은?

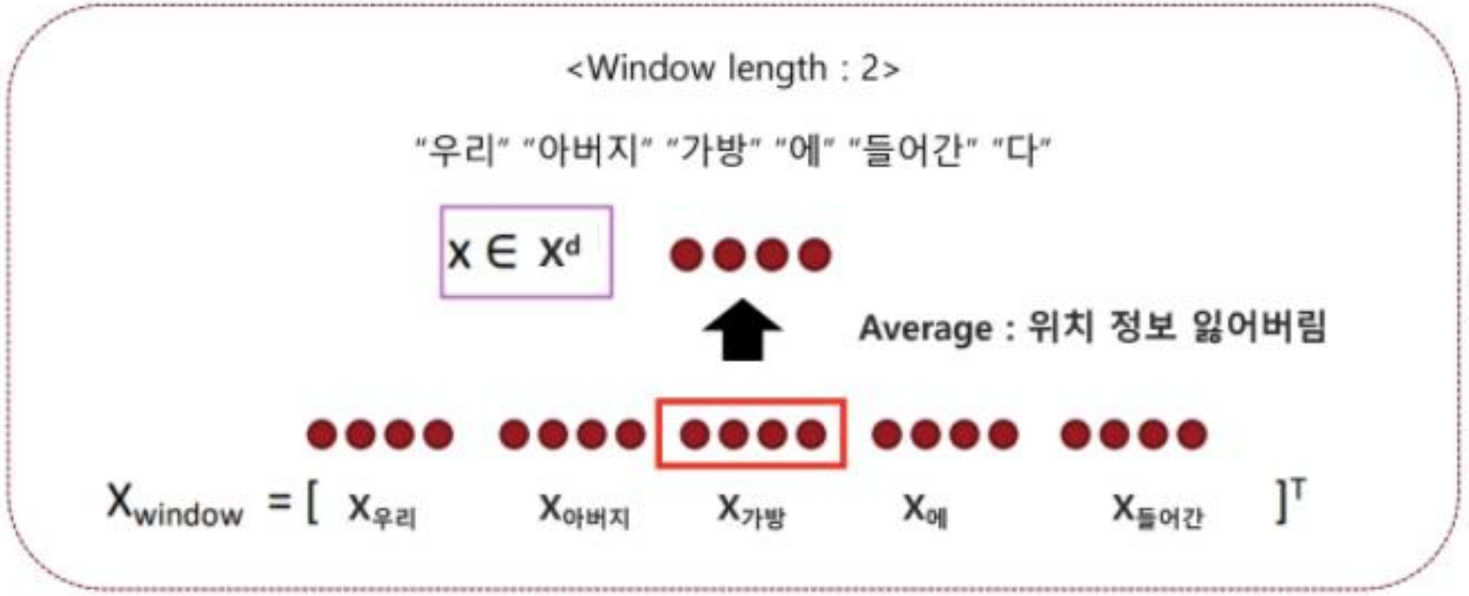
03 Window classification

Window classification

- 단어는 문맥 안에서 분류 되어야 하므로, 중심 단어와 주변 단어를 window 로 묶어 이를 활용해서 분류하자!

- 가장 간단한 방법은. Window 안의 word vector 들을 평균 낸 average vector를 분류하는 것이다.
  - 그러나 이 방법은 position 정보를 잃어버리는 단점이 있다.

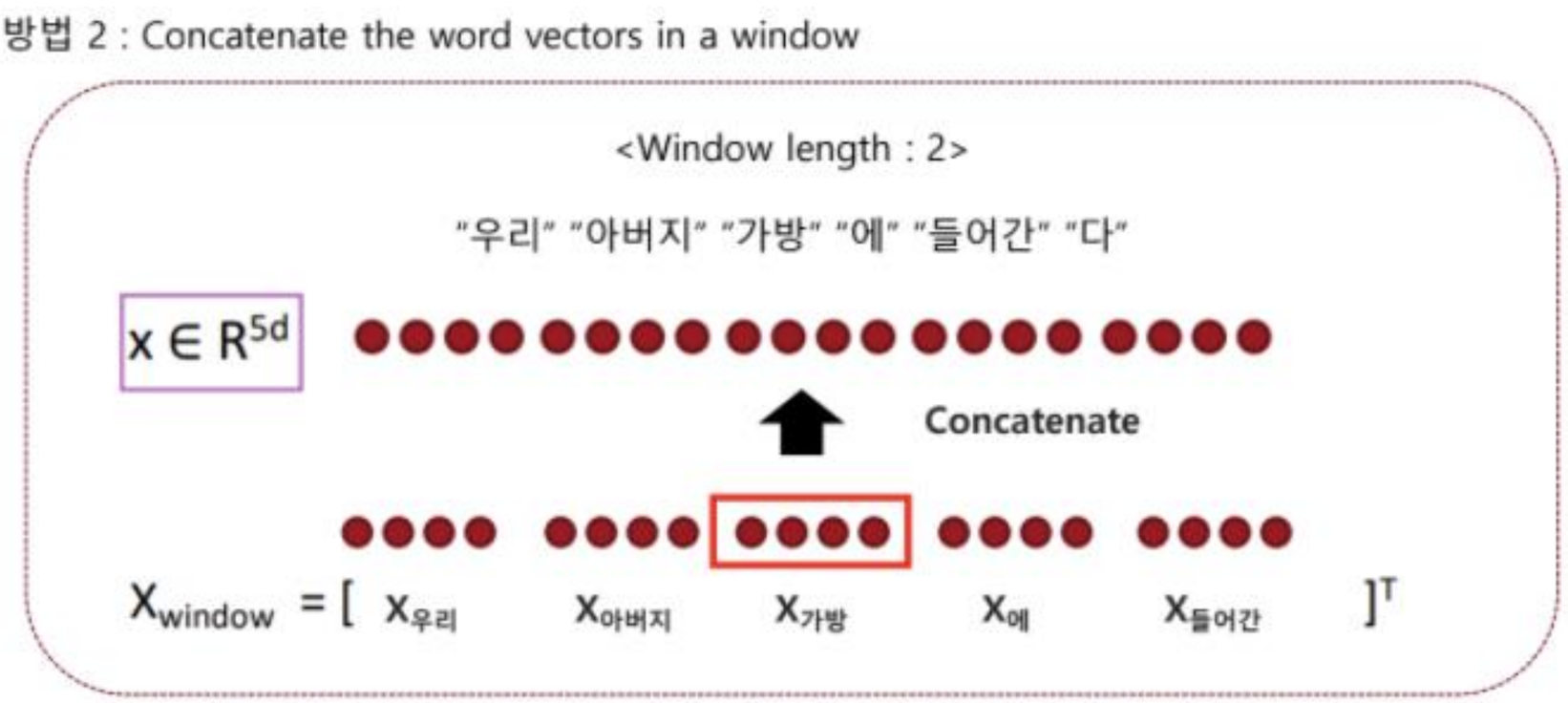
▪ 방법 1 : Average the word vectors in a window



03 Window classification

Window classification

- Centre word를 분류하기 위해서 3-layer neural net을 이용해 score를 구하는데, 이때 window 안의 모든 word vector를 합친 것을 이용한다.



03 Window classification

Simplest window classifier: Softmax

- R^{5d} ?

$$w_i^k = \left(w_{i1}^k, w_{i2}^k, \cdots, w_{id}^k\right)^T$$

$$W^k = \left(w_1^k \quad w_2^k \quad \cdots \quad w_h^k\right)^T$$

$$W^k = \begin{pmatrix} w_{11}^k & w_{21}^k & \cdots & w_{i1}^k & \cdots & w_{h1}^k \\ w_{12}^k & w_{22}^k & & & & \vdots \\ \vdots & & \ddots & & & \vdots \\ w_{1j}^k & & & w_{ij}^k & & w_{hj}^k \\ \vdots & & & & \ddots & \vdots \\ w_{1d}^k & \cdots & \cdots & w_{id}^k & \cdots & w_{hd}^k \end{pmatrix}^T$$

$$W^k x + b^k = \begin{pmatrix} w_{11}^k & \cdots & w_{1d}^k \\ \vdots & \ddots & \vdots \\ w_{h1}^k & \cdots & w_{hd}^k \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} + \begin{pmatrix} b_1^k \\ b_2^k \\ \vdots \\ b_h^k \end{pmatrix}$$

h*d d*1 h*1

03 Window classification

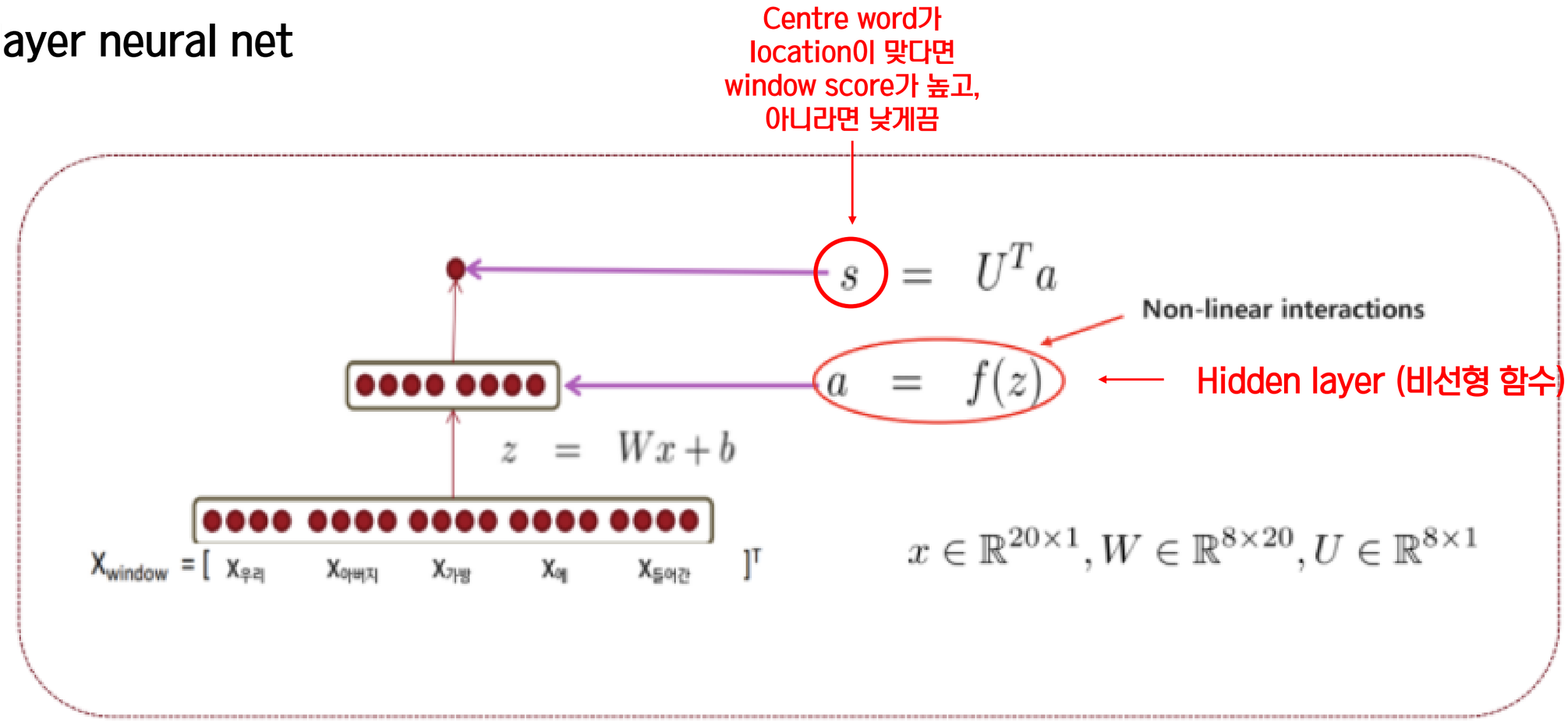
Window classification

- **3-layer neural net**
  - 만약 center word가 장소인지 아닌지를 분류하고 싶다고 해보자.
  - Center word가 장소로 분류되면 높은 score를, 그게 아니라면 낮은 score를 받아야 한다.

03 Window classification

Window classification

- 3-layer neural net



03 Window classification

Window classification

- 3-layer neural net
  - 학습 - **Softmax**

Score에 대한 확률

$$x = x_{window}$$

predicted model output probability

$$\hat{y}_y = p(y|x) = \frac{\exp(W_y.x)}{\sum_{c=1}^{C} \exp(W_c.x)}$$

- With cross entropy error as before:

same

Weight update ←

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} -\log\left(\frac{e^{f_{y_i}}}{\sum_{c=1}^{C} e^{f_c}}\right)$$

03 Window classification

Window classification

- 3-layer neural net
  - 학습 - **Softmax**

predicted model output probability

$$\hat{y}_y = p(y|x) = \frac{\exp(W_y.x)}{\sum_{c=1}^{C} \exp(W_c.x)}$$

- With cross entropy error as before:

same

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} -\log \left(\frac{e^{f_{y_i}}}{\sum_{c=1}^{C} e^{f_c}} \right)$$

$$p(y|\mathbf{s}) = \frac{\exp(\mathbf{s})}{\sum_{c=1}^{C} \exp(\mathbf{s})} = \text{softmax}(\mathbf{s})$$

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} -\log p(y|\mathbf{s})$$

03 Window classification

Window classification

- 3-layer neural net
  - 학습 – **Max-margin loss**

$$\text{minimize } J = \max(1 + s_c - s, 0)$$

- 정답과 오답 사이의 거리를 최대로 만들어주는 margin을 찾는 것
- 정답과 오답 사이의 차이가 k 이상일 경우 loss를 0으로 만든다. (= 정답으로 본다.)
- 사진의 공식은 k = 1 일 때

"우리" "아버지" "가방" "에" "들어간" "다"

- S : True window's score

$$X_{window} = [X_{우리}\ X_{아버지}\ X_{가방}\ X_{에}\ X_{들어간}]^T$$

- S_C : Corrupt window's score

$$X_c = [X_{아버지}\ X_{가방}\ X_{에}\ X_{들어간}\ X_{다}]^T$$

03 Window classification

Window classification

- 3-layer neural net
  - 학습 – **Max-margin loss**

$$\text{minimize } J = \max(1 + s_c - s, 0)$$

- $s_c = U^T f(Wx_c + b)$, Corrupt window's score

  $s = U^T f(Wx + b)$, True window's score

- $(S_C - S) > 1$: 학습시킨다

  - Positive margin $\Delta = 1$

    -> Margin이 없으면 Optimization objective가 risky 함

'Location' 이 있는 window와 아닌 window의 구분이 모호해질 수 있기 때문

03 Window classification

Window classification

- 3-layer neural net
  - 학습 – **Max-margin loss**

    - Word2Vec 학습 방식과 유사점

      - Window가 corpus를 따라 움직이며 모든 위치에 대해 학습시킴

      - Negative sampling: “무관한 단어들에 대해서는 weight를 업데이트하지 않아도 된다 "

03 Window classification

Gradient descent algorithm

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$

Learning late

- 함수 (J)에 대한 각 Parameter(θ)의 미분값 요구

$\nabla_\theta J(\theta)$ → Backpropagation algorithm 으로 구함

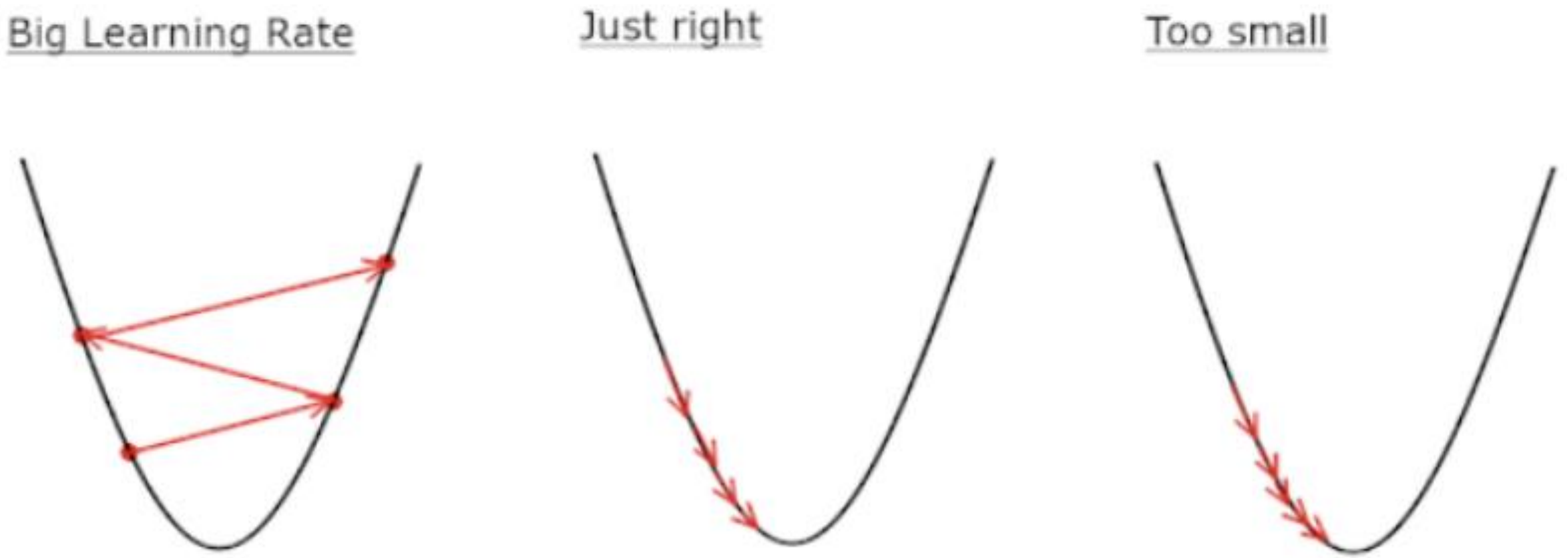
- Neural network에서 Parameter(θ) 개수 多

- Weight update를 할 때, 이전 weight,에서 해당 파라미터의 loss 미분값과 learning rate를 곱한 값을 빼준다.
- 파라미터: W, U, b, X
- 미분을 하는 이유: 각 parameter가 loss에 미치는 영향(기여도)을 구하기 위함
- Learning rate = step size
  - 다음 지점을 결정

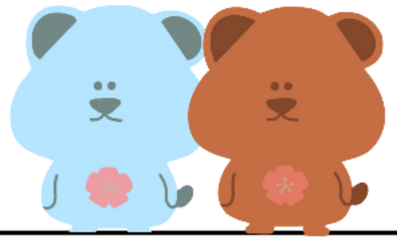
03 Window classification

Gradient descent algorithm

- Learning rate = step size
  - 다음 지점을 결정
  - 너무 크면 데이터가 무질서하게 이탈
  - 너무 작으면 학습시간이 매우 오래 걸림



THANK YOU



EWHA EURON