

Biomedical Text Mining using Neural Networks

1. Introduction & Word embedding

서울대학교병원 정보화실

고태훈 (taehoonko@snuh.org)

최세원 (swc@snuh.org)

연자 소개



고태훈



최세원

- 서울대학교병원 정보화실 연구교수
- 서울대학교 산업공학과 박사 졸업
- 진료정보교류 및 의료 빅데이터
연구과제 진행

- 서울대학교병원 정보화실 임상교수
- 서울대학교병원 응급의학과 M.D.
- 서울대학교 의과대학 학사졸업 및
의공학과 박사수료
- 정밀의료 플랫폼 구축사업 실무총괄

목차

1. Basics for Neural Networks

2. Bag-of-words

3. Neural Language Model

목차

1. Basics for Neural Networks

2. Bag-of-words

3. Neural Language Model

Limitation of linear models

- 선형 모델의 한계

- 분류:

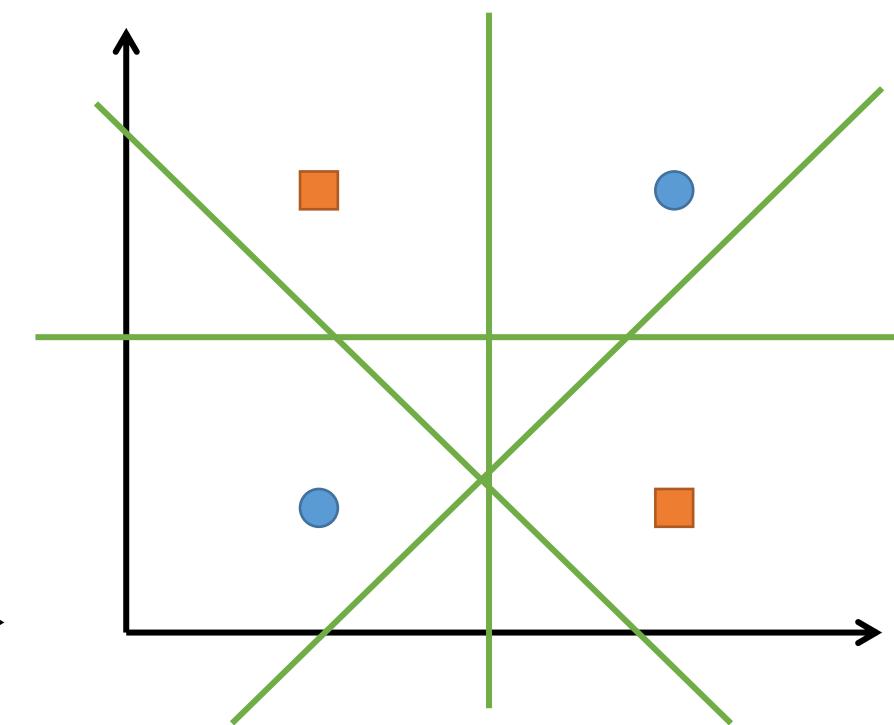
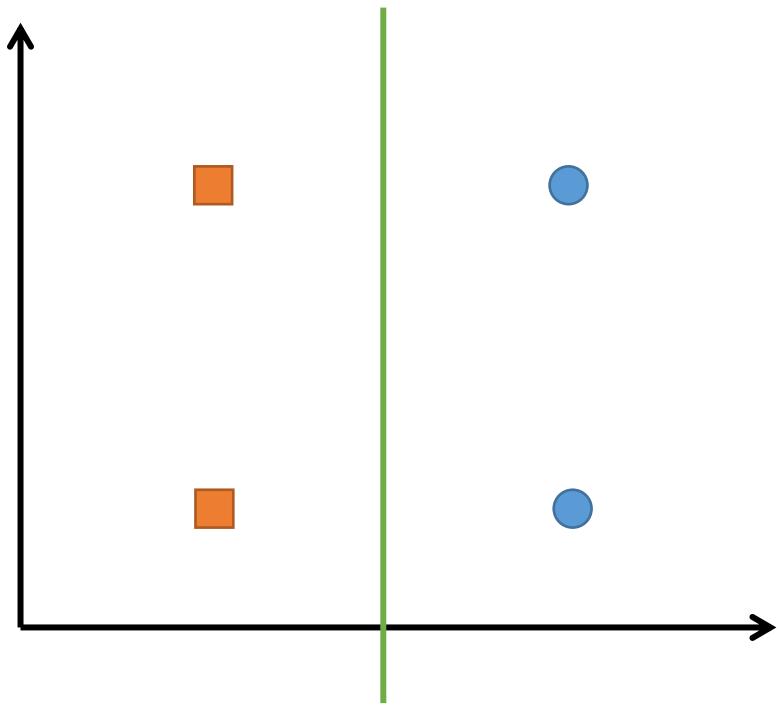
- 선형판별분석 (Linear Discriminant Analysis)
 - 로지스틱 회귀모델 (Logistic Regression Model)
 - 항상 선형 분류 경계면을 생성함 → 비선형 분류 경계면을 생성할 수 없음

- 회귀:

- 다중선형회귀모델 (Multiple Linear Regression)
 - 설명변수와 종속변수 사이의 관계를 선형으로 가정
 - 비선형 관계를 설명할 수 없음
- 분류/회귀 문제의 선형성 가정이 위배될 경우 우수한 예측 성능을 보일 수 없음

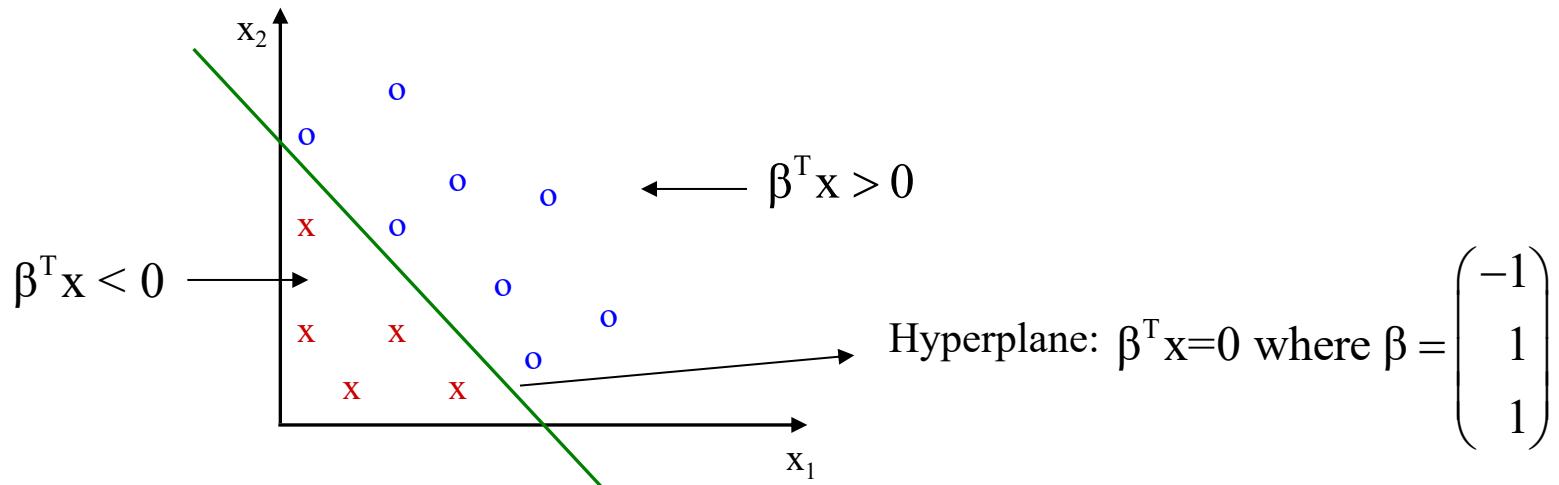
Limitation of linear models

- 단 하나의 직선을 그어서 파란색 원과 붉은색 사각형을 분류할 수 있는가?



Logistic regression

- Logistic regression은 학습 알고리즘을 통해 2-class points를 분류하는 초평면 (hyper-plane)을 찾는 것
- 선형적인 의사결정경계 (linear decision boundary) 만 생성 가능



Classifier

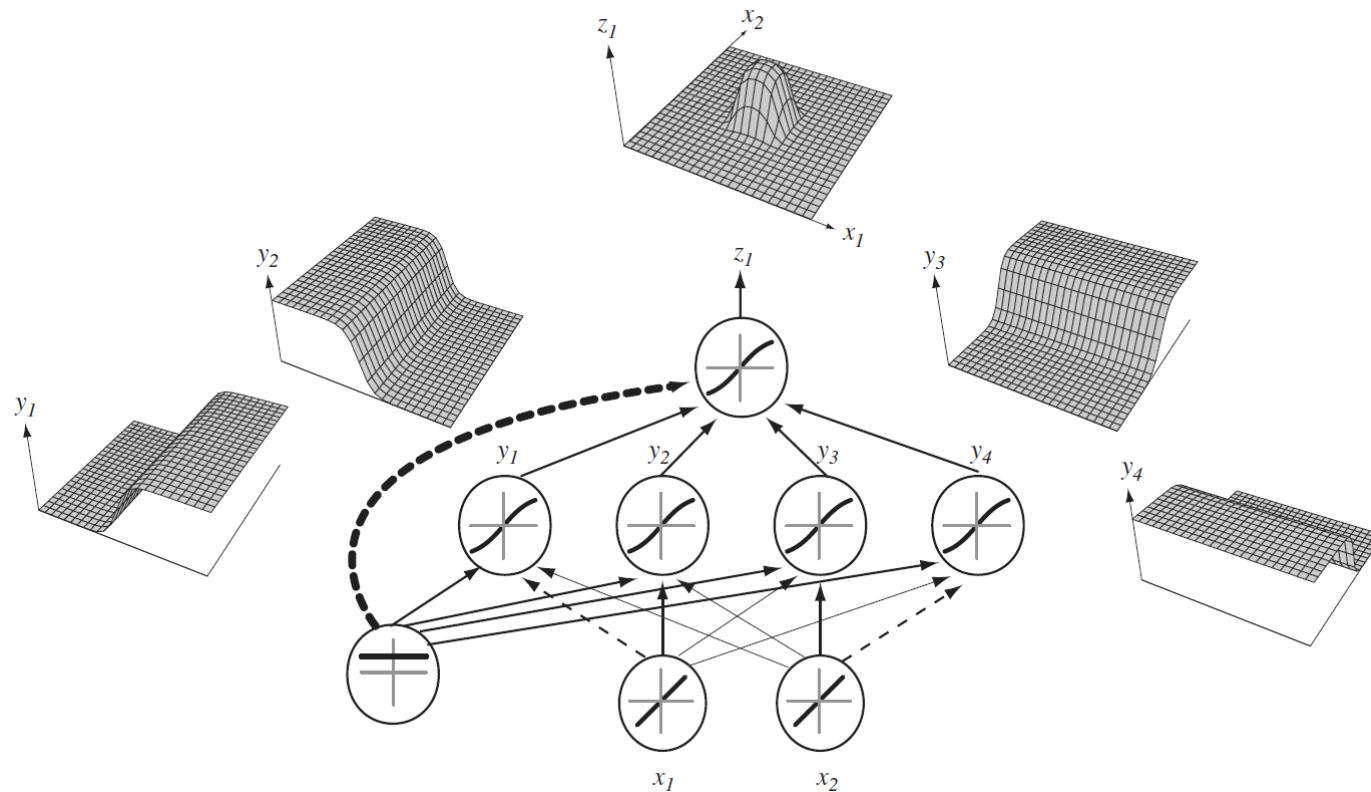
$$y = \frac{1}{(1 + \exp(-\beta^T x))}$$

$$\begin{cases} y \rightarrow 1 & \text{if } \beta^T x \rightarrow \infty \\ y = \frac{1}{2} & \text{if } \beta^T x = 0 \\ y \rightarrow 0 & \text{if } \beta^T x \rightarrow -\infty \end{cases}$$

인공신경망(Artificial Neural Network)

- 비선형 모델의 장점

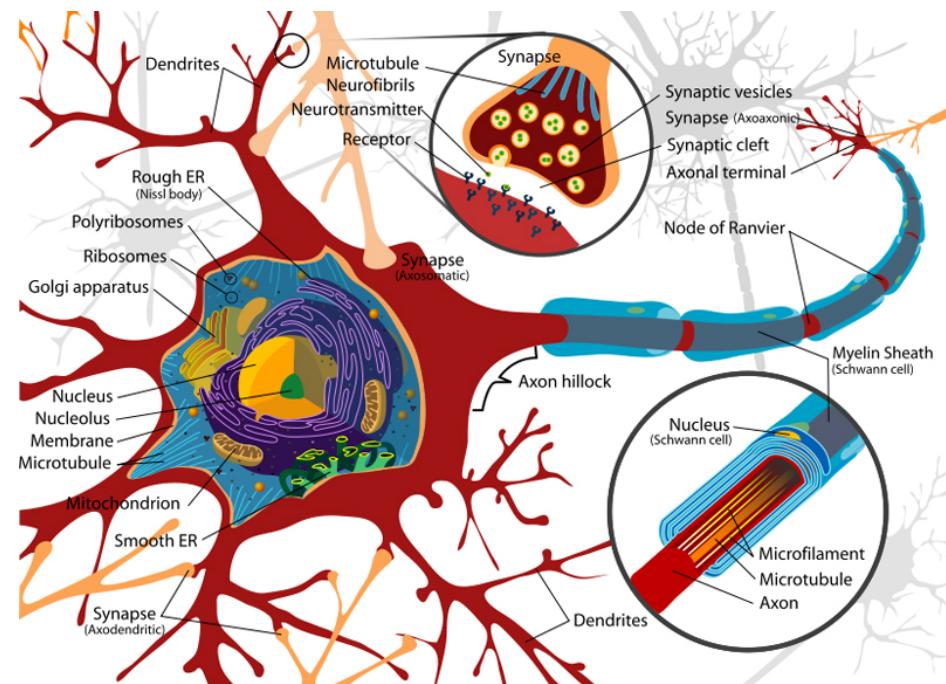
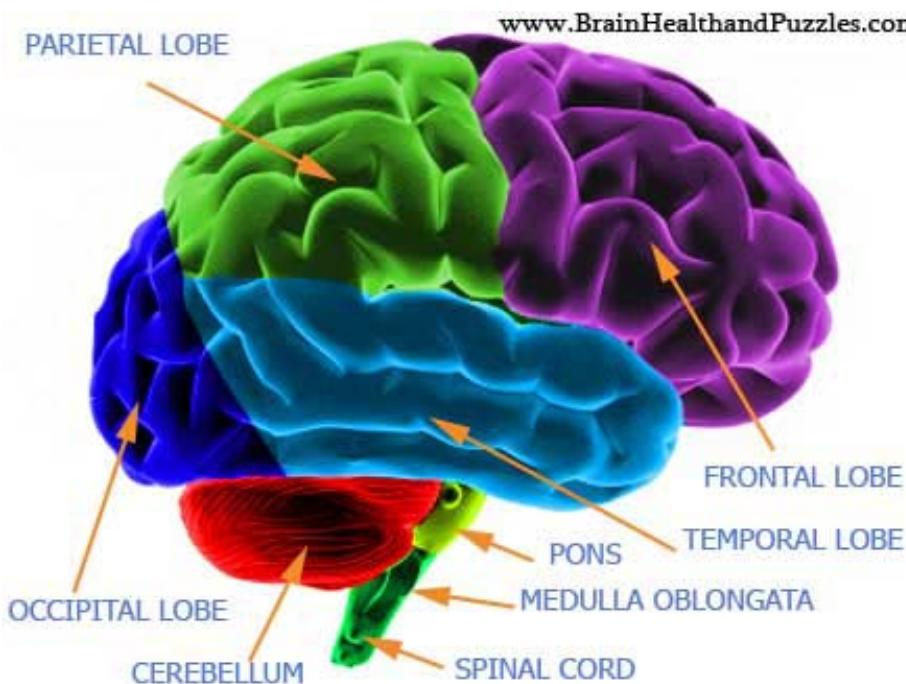
- 임의의 모양을 가진 분류 경계면이나 회귀선을 찾아낼 능력이 있음



인공신경망(Artificial Neural Network)

- 우리 뇌가 정보를 처리하는 방식

- 외부의 자극이 감지되면 자극 감지 기관으로부터 뇌의 특정 부분까지의 신경계가 활성화되면서 해당 정보를 처리함
- 신경세포들은 전기적 신호를 이용하여 메시지를 주고받음



인공신경망(Artificial Neural Network)

- **목적**

- 인간의 뇌가 정보를 처리하는 과정을 모사하는 기계를 만들어 복잡한 문제에 대해서도 우수한 분류/예측 성능을 나타낼 수 있도록 하는 것

- **응용분야**

- 원인 규명보다 결과의 정확성이 중요하게 취급되는 곳

- **장점**

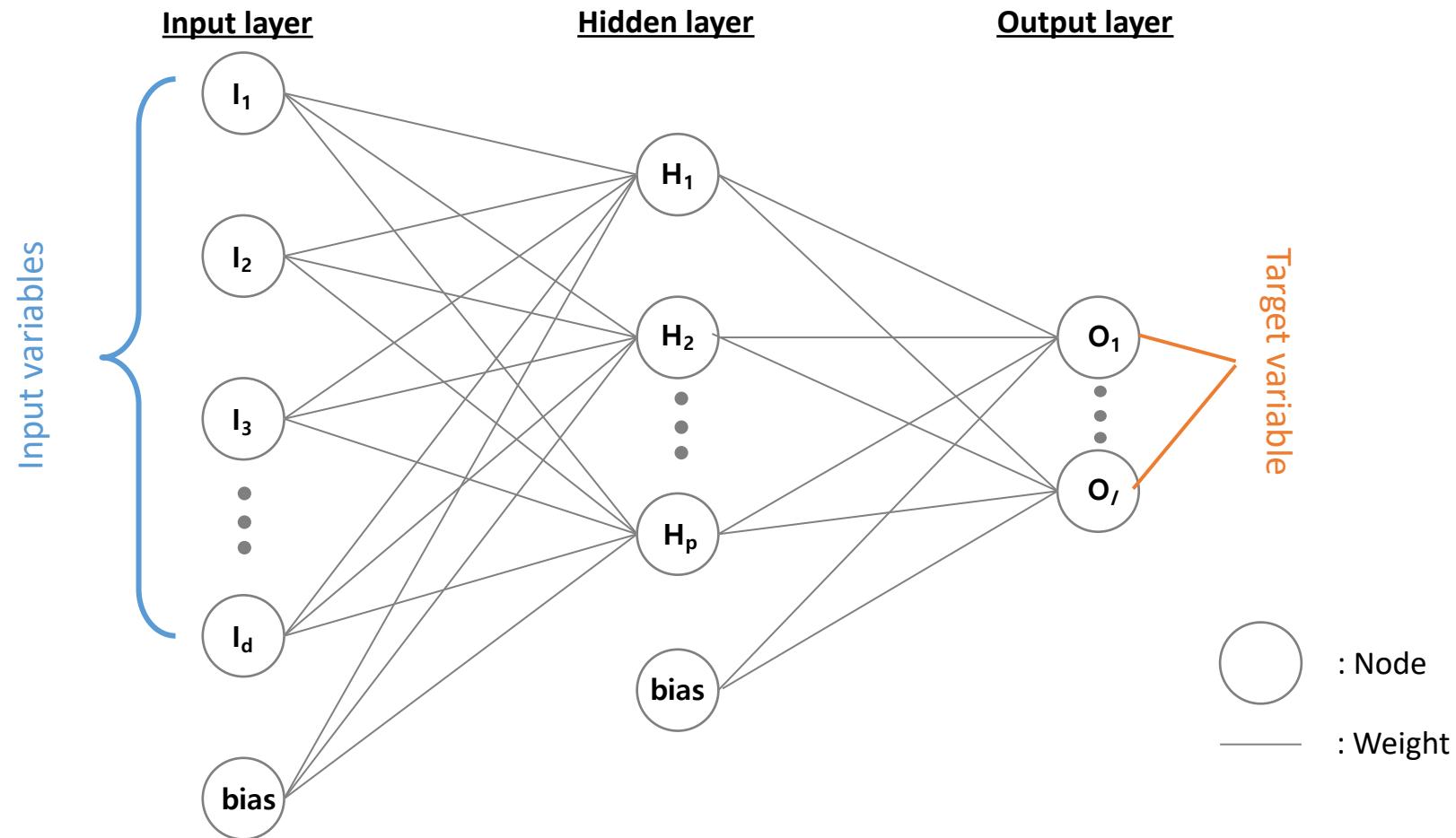
- 비선형 의사결정경계를 찾아내는 능력
 - 우수한 예측 성능
 - (느린 학습 속도에 비해) 매우 빠른 예측 속도
 - (메모리 측면에서) 효율적으로 대용량 데이터 (large-scale data) 를 학습할 수 있음

Feed-forward network

- Feed-forward network (전방향 네트워크)
 - a.k.a. multilayer perceptron (다층 퍼셉트론), fully-connected network
 - 각 층의 노드와 노드들이 모두 연결되어 있으며, 신호가 forward하게 움직이는 인공신경망 모델
 - 구성
 - 입력 층과 입력 노드 ((input layer and input nodes)
 - 은닉 층과 은닉 노드 (hidden layers and hidden nodes)
 - 출력 층과 출력 노드 (output layer and output nodes)
 - 가중치 (weights)

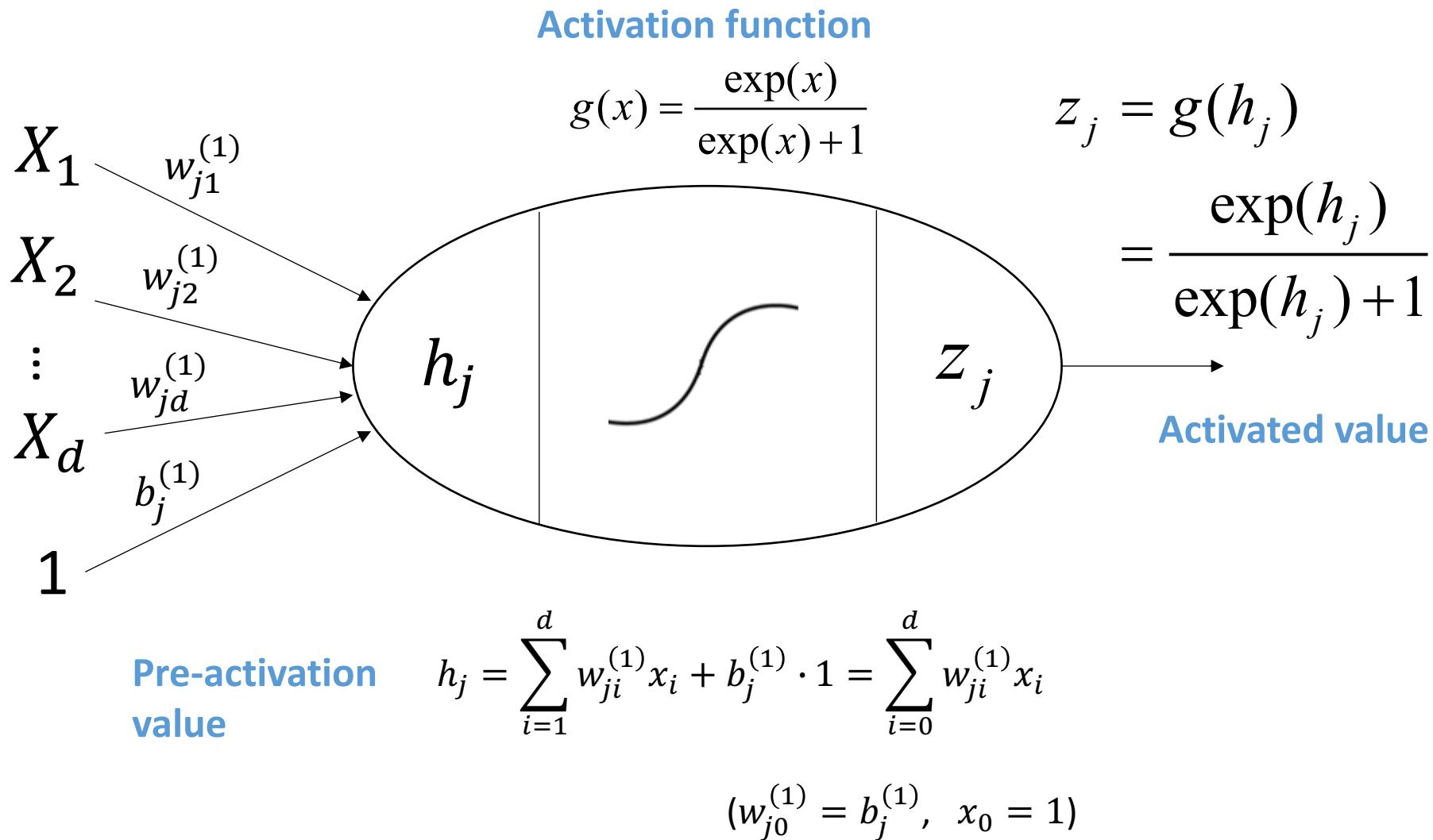
Feed-forward network

- Example of the feed-forward network
 - 3 layers
 - d input nodes, p hidden nodes, l output nodes.



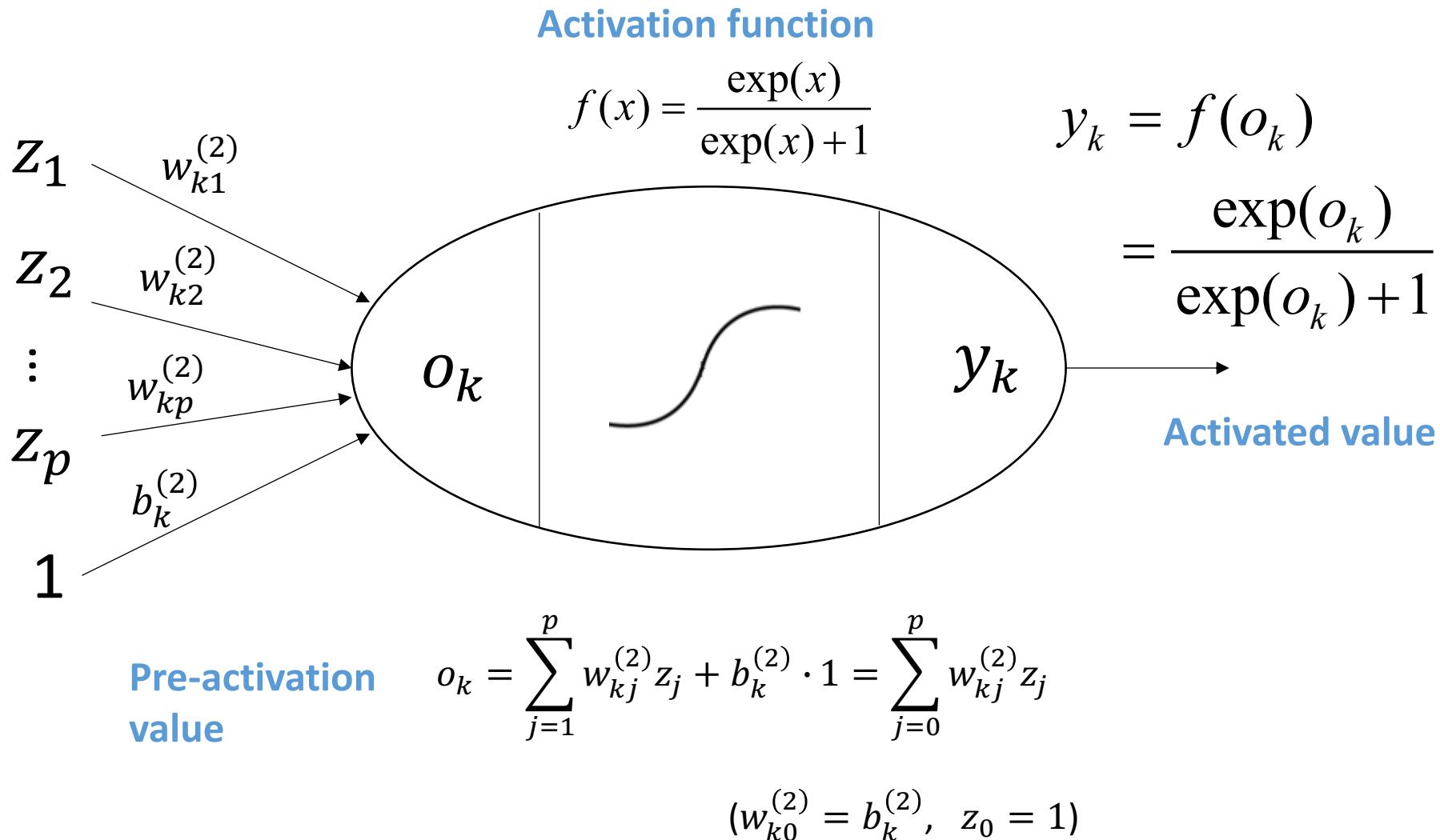
Feed-forward network

- In a hidden node j (for $j = 1, 2, \dots, p$)

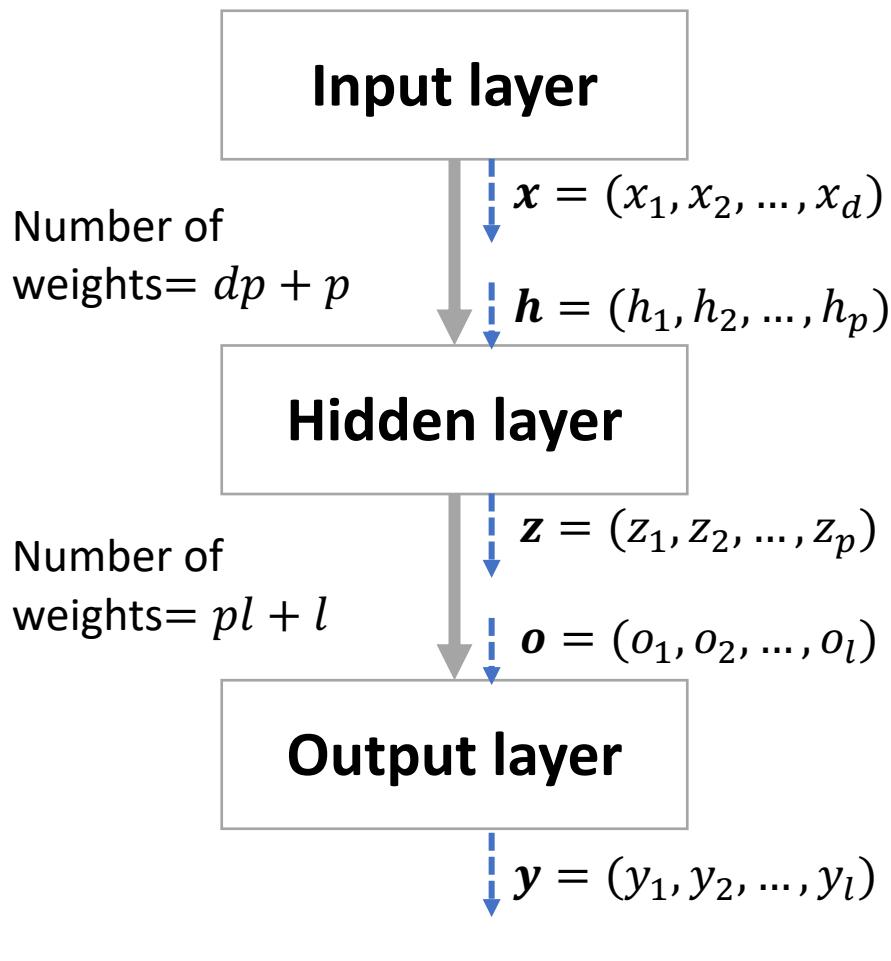


Feed-forward network

- In an output node k (for $k=1, 2, \dots, l$)



Feed-forward network



$$z_j = g(h_j) = g\left(\sum_{i=0}^d w_{ji}^{(1)} x_i\right)$$

$$\begin{aligned} y_k &= f(o_k) = f\left(\sum_{j=0}^p w_{kj}^{(2)} z_j\right) \\ &= f\left(\sum_{j=0}^p w_{kj}^{(2)} g\left(\sum_{i=0}^d w_{ji}^{(1)} x_i\right)\right) \end{aligned}$$

Structure of feed-forward networks

- Feed-forward network에서 다음 사항은 사용자 정의
 - Number of hidden layers
 - Number of hidden nodes in each hidden layer
 - Activation function (활성함수)
- 다음 사항은 데이터에 의존적
 - Number of input nodes
 - Number of output nodes
- 학습해야 하는 것
 - Weights, biases

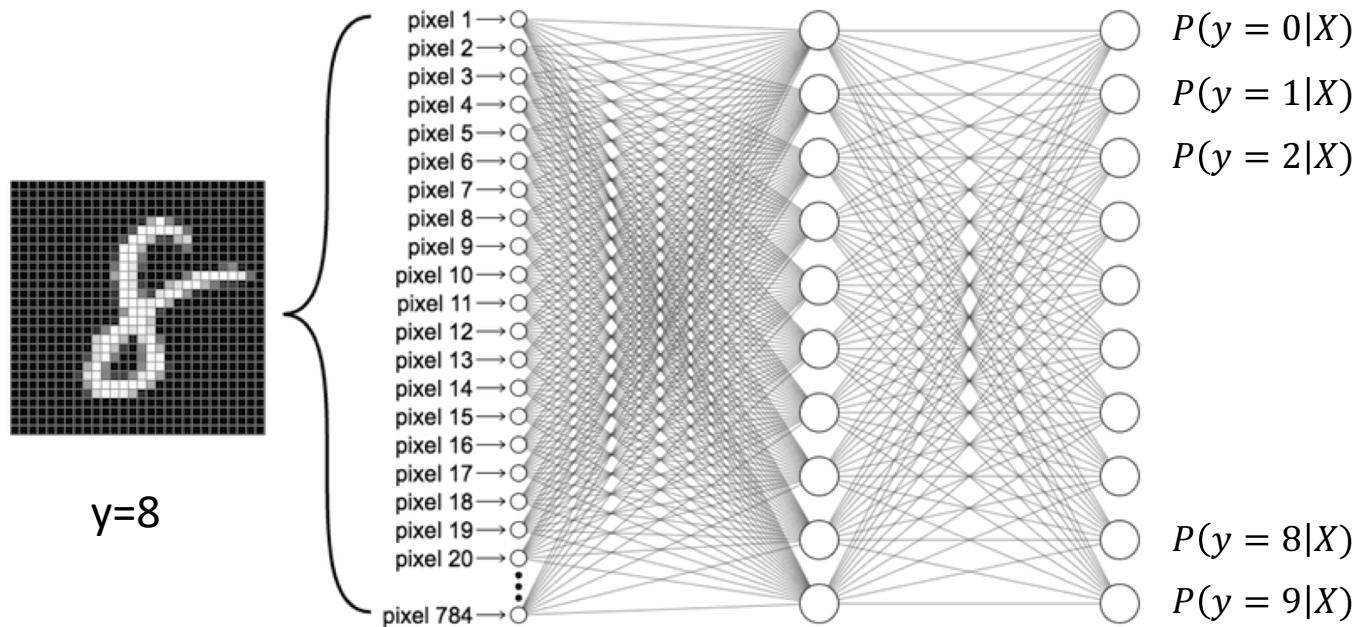
네트워크 층과 노드 수

- **입력층: 1개**
 - 입력 노드 수 = 입력 변수의 수
- **출력층: 1개**
 - (Classification) 출력 노드 수 = 클래스의 수
 - (Regression) 출력 노드 수 = 타겟 변수의 수
- **은닉층: 내 맘대로**
 - 은닉 노드 수: 내 맘대로
 - Sadly, there is no rule of thumb to find out how many hidden layers and nodes are needed.

네트워크 층과 노드 수

- **MNIST data**

- 28 x 28 차원의 손글씨 숫자 (총 784차원)
- 클래스 = 10개 (숫자 0부터 9까지)

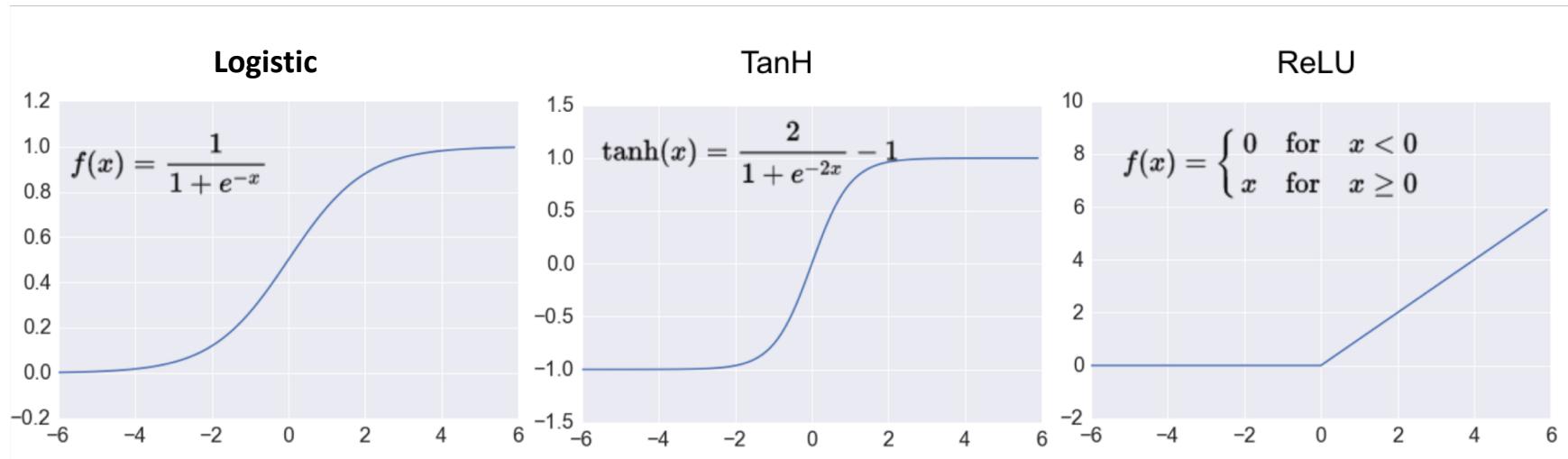


https://ml4a.github.io/ml4a/looking_inside_neural_nets/

Activation function

- 활성함수: Activation function

- 비선형 추정을 가능하게 하는 함수
- 주로 사용하는 함수들
 - Logistic or softmax function
 - Hyperbolic tangent
 - Rectified linear unit (ReLU)



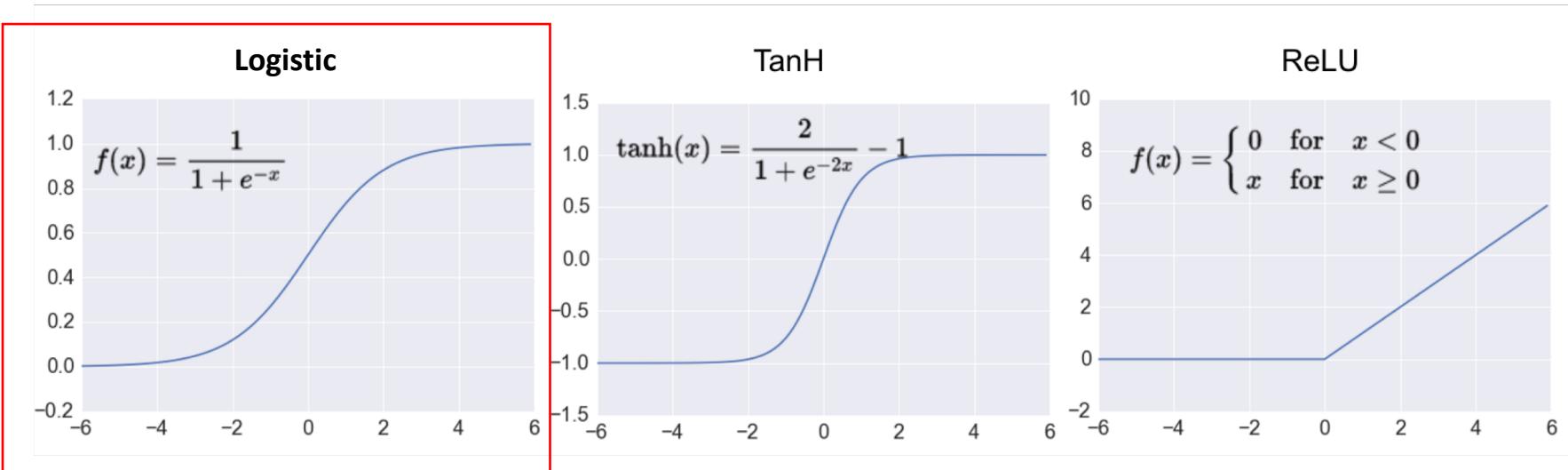
Activation function: Softmax function

- Logistic function, a.k.a. softmax function or sigmoid function

- equation

$$f(x) = \frac{1}{1 + \exp(-x)} = \frac{\exp(x)}{1 + \exp(x)}$$

- range: $0 < f(x) < 1 \rightarrow$ 확률의 값을 모델링하기에 적합
 - Supervised neural network, 다시 말해, **클래스 형태의 타겟 변수가 있고 이 것을 분류하는 인공 신경망에서는 출력층에서 필수적으로 사용**



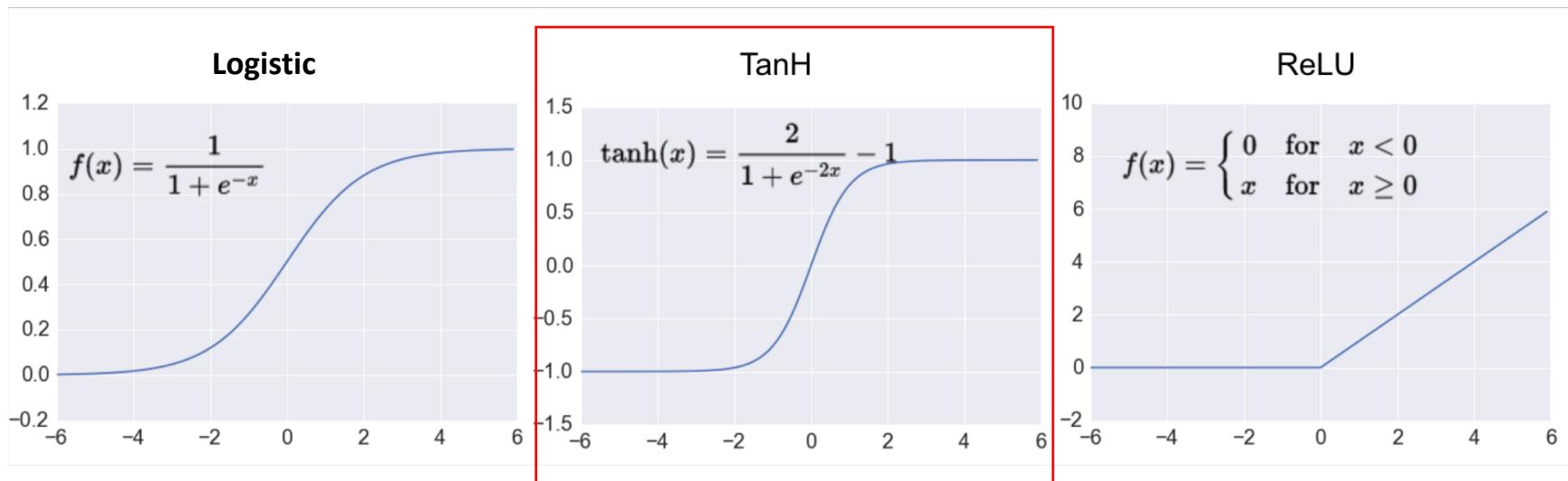
Activation function: Hyperbolic tangent function

- Hyperbolic tangent function (tanh)

- equation

$$f(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = \frac{\exp(2x) - 1}{\exp(2x) + 1} = \frac{2}{1 + \exp(-2x)} - 1$$

- range: $-1 < f(x) < 1$
 - 과거에는 은닉 노드의 활성함수로 많이 이용되었으며, 현재는 **recurrent neural network (RNN)**에서 많이 이용됨



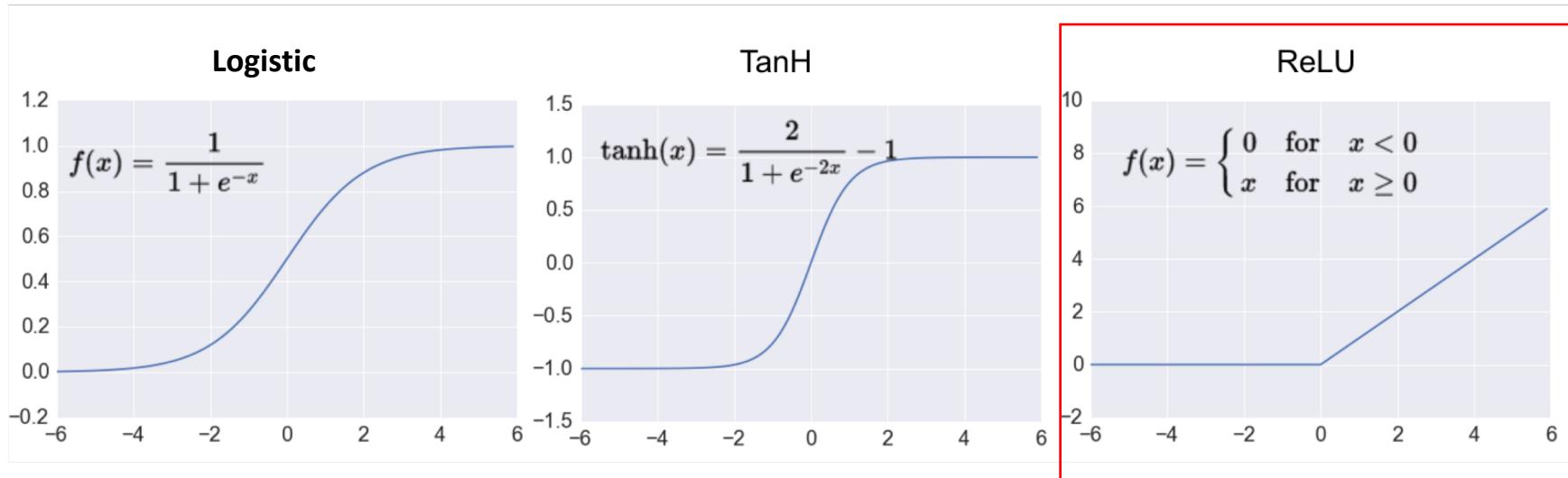
Activation function: Rectified Linear Unit (ReLU)

- Rectified Linear Unit (ReLU)

- equation

$$f(x) = \max(0, x)$$

- range: $0 \leq f(x) < \infty \rightarrow$ 일정 구간($x < 0$)에서 받은 신호를 비활성화
 - 많은 인공신경망에서 **은닉노드의 활성함수**로 많이 이용되고 있음
 - 타겟 변수가 0 이상의 실수인 경우, 이를 모델링하기 위해 출력노드의 활성함수로 이용되기도 함

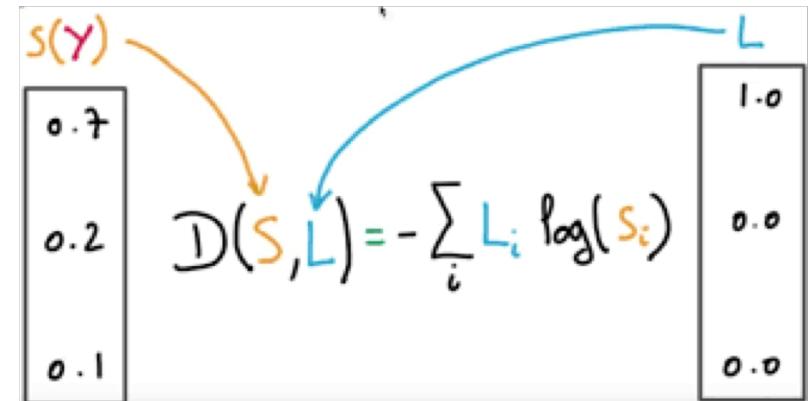
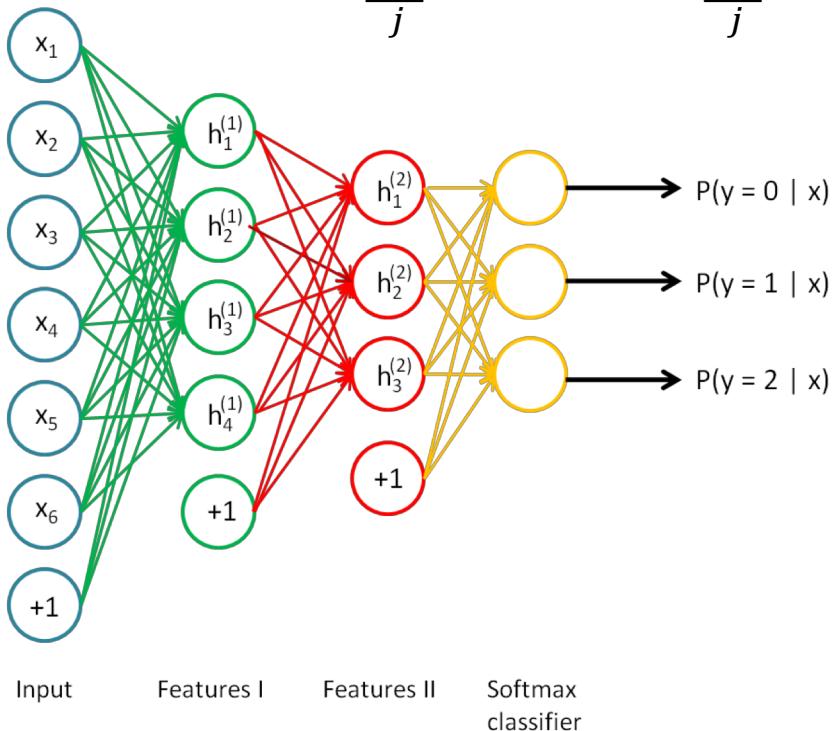


Loss function: Cross-entropy

- **Cross-entropy (in general)**

Cross-entropy for given (X, Y) and parameters β

$$\sum_j -y_j \log \pi_j = \sum_j -y_j \log P(Y = y_j | X, \beta)$$



How to learn NNs: Backpropagation

- 인공신경망, 딥러닝 모델은 모두 역전파 (backpropagation) 방법으로 가중치가 학습된다.
- 원리
 - 초기 가중치는 데이터가 잘 학습이 되어 있지 않기 때문에, 관측된 입력 변수 X 를 신경망에 투입했을 때 도출되는 \hat{y} 가 실제 타겟 변수인 y 와 차이가 클 것이다.
 - 따라서, y 와 \hat{y} 의 차이 혹은 이 둘로 구성된 손실함수 (loss function) 을 줄이는 것을 목표로 경사하강법 (gradient descent)을 사용하면 가중치를 학습(또는 업데이트)할 수 있을 것이다.

How to learn NNs: Backpropagation

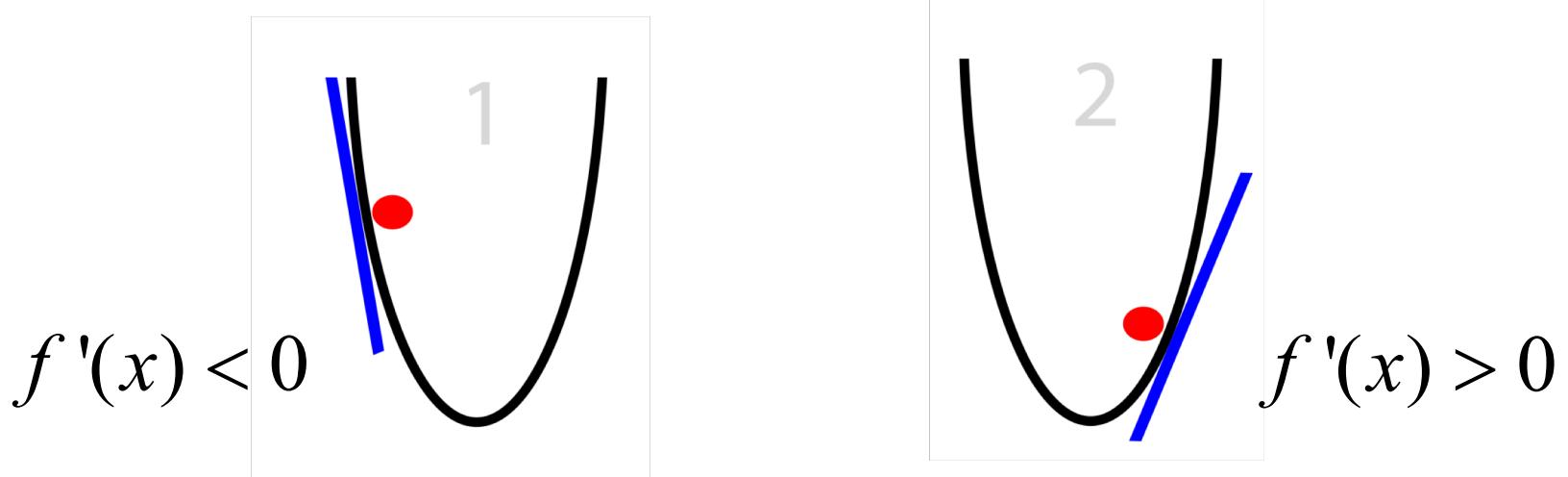
- **Step0: Initialize the network.**
 - You should determine the structure such as number of input/hidden/output nodes and number of hidden layers.
 - Weights and bias terms can be **randomly initialized** by various methods.
- **Step1: Present input to network.**
- **Step2: Propagate forward the activation. (→)**

Compute network output and compare with target.
- **Step3: Propagate backward the error. (←)**

Compute “error signal” δ .
- **Step4: Update weights using gradient descent-based methods.**
- **Step5: Repeat steps 1~4 for each training pattern until the stopping condition is met.**

Gradient descent: Simple example

- The red ball moves in the opposite direction of the gradient.
- If the absolute value of gradient is large, the red ball moves a lot.



$$x_{t+1} = x_t - \eta f'(x_t)$$

η : Learning rate ($\eta \in [0,1]$)

How to learn NNs: Backpropagation

- 학습 방식 1: Case updating (Stochastic backpropagation)

- 각 개체가 투입될 때마다 가중치의 업데이트를 수행함
- 학습 데이터의 모든 개체가 투입되면 1 epoch이 수행된 것으로 간주함
- 1 epoch 이후에는 다시 첫 번째 개체부터 순차적으로 투입하여 가중치 업데이트 수행

- 학습 방식 2: Batch updating (Batch backpropagation)

- 학습 데이터에 속하는 모든 포인트 전부 투입하여 평균 오차를 산출
- 평균 오차를 이용하여 개별 가중치의 업데이트를 수행

- 학습 방식 3: Mini-batch updating (Batch backpropagation)

- 1번과 2번의 중간형태
- 학습 데이터에서 포인트를 샘플링하여 평균 오차를 산출하고 개별 가중치의 업데이트를 수행
- Mini-batch의 사이즈와 epoch 수를 정하여 가중치 업데이트 수행
- 최근 가장 많이 쓰이는 방법

How to learn NNs: Backpropagation

- 학습 방식 1: Case updating (Stochastic backpropagation)

목적함수 = $Loss_i$

개별 데이터포인트 별 loss를 이용

- 학습 방식 2: Batch updating (Batch backpropagation)

목적함수 = $\frac{1}{N} \sum_i Loss_i$

학습 데이터 전체에 대한 평균 loss를 이용
N: 학습 데이터포인트 수

- 학습 방식 3: Mini-batch updating (Batch backpropagation)

목적함수 = $\frac{1}{k} \sum_i Loss_i$

샘플 포인트에 대한 평균 loss를 이용
k: mini-batch의 크기

Squared loss: $\frac{1}{2} (y - \hat{y})^2$

Cross-entropy loss: $\sum_j -y_j \log \hat{y}_j$

Backpropagation

- **학습 방식 1: Case updating**

- 각 개체가 투입될 때마다 가중치의 업데이트를 수행함
- 학습 데이터의 모든 개체가 투입되면 1 epoch이 수행된 것으로 간주함
- 1 epoch 이후에는 다시 첫 번째 개체부터 순차적으로 투입하여 가중치 업데이트 수행

- **학습 방식 2: Batch updating**

- 학습 데이터에 속하는 모든 포인트 전부 투입하여 평균 오차를 산출
- 평균 오차를 이용하여 개별 가중치의 업데이트를 수행

- **학습 방식 3: Mini-batch updating**

- 1번과 2번의 중간형태
- 학습 데이터에서 포인트를 샘플링하여 평균 오차를 산출하고 개별 가중치의 업데이트를 수행
- Mini-batch의 사이즈와 epoch 수를 정하여 가중치 업데이트 수행
- 최근 가장 많이 쓰이는 방법

Backpropagation

- 학습 방식 1: Case updating

목적함수 = $Loss_i$

개별 데이터포인트 별 loss를 이용

- 학습 방식 2: Batch updating

목적함수 = $\frac{1}{N} \sum_i Loss_i$

학습 데이터 전체에 대한 평균 loss를 이용
N: 학습 데이터포인트 수

- 학습 방식 3: Mini-batch updating

목적함수 = $\frac{1}{k} \sum_i Loss_i$

샘플 포인트에 대한 평균 loss를 이용
k: mini-batch의 크기

Squared loss: $\frac{1}{2} (y - \hat{y})^2$

Cross-entropy loss: $\sum_j -y_j \log \hat{y}_j$

목차

1. Basics for Neural Networks

2. Bag-of-words

3. Neural Language Model

Bag-of-words

- 문서들을 등장 단어에 대한 가중치로 표현하는 방법
 - 가중치
 - 등장 유무 (0 or 1)
 - 등장 빈도 수 (term frequency)
 - TF-IDF
- 장점
 - 문서의 벡터 표현이 직관적임

Bag-of-words: Example

- 예제 문장

- Korean: ‘머신러닝과 딥러닝을 이용하여 뉴스를 분류하다.’
- English: ‘Classify news using machine learning and deep learning.’



머신	러닝	과	딥	을	이용	하여	뉴스	분류	하다
1	2	1	1	2	1	1	1	1	1

classify	news	using	machine	learning	and	deep
1	1	1	1	2	1	1

Bag-of-words with n-gram features

- **n-gram**

- n개의 연속된 단어 열
- Unigram ($n=1$) 보다 그 이상의 단어 열이 정보를 더 잘 표현할 수 있음
 - [machine, deep, learning] **vs.** [machine learning, deep learning]

- **BOW + n-gram features: 예제 문장**

- English: ‘Classify news using machine learning and deep learning.’



‘term-frequency’ BOW with unigrams and bigrams

classify	news	using	machine	learning	and	deep
1	1	1	1	2	1	1

classify news	news using	using machine	machine learning	learning and	and deep	deep learning
1	1	1	1	1	1	1

Bag-of-words: TF-IDF

- **Term frequency and inverse document frequency (TF-IDF)**
 - $tf(w)$: term frequency (number of word occurrences in a document)
 - $df(w)$: number of documents containing the word
(number of documents containing the word)

$$TF - IDF(w) = \underline{tf(w)} \times \log \left(\frac{N}{\underline{df(w)}} \right)$$

The word is more important if it appears several times in a target document

The word is more important if it appears in less documents

Bag-of-words: TF-IDF

- Word Weighting: Example

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0.35
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Bag-of-words

- 장점

- 문서의 벡터 표현이 직관적임

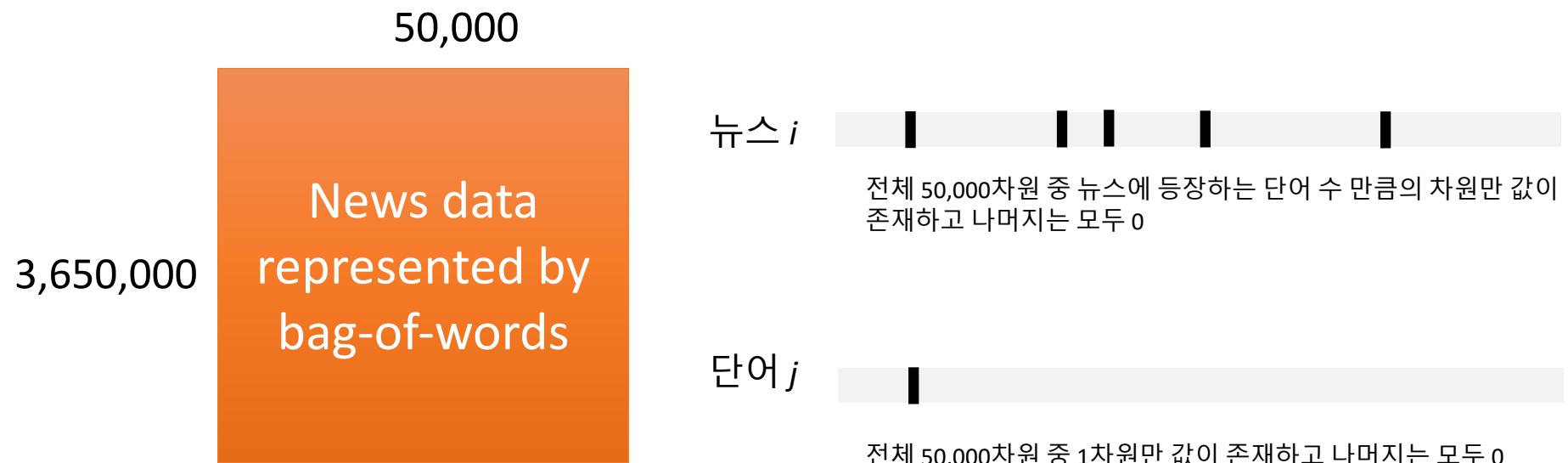
- 단점

- 문서 벡터의 차원이 지나치게 큼
 - 전체 문서에 등장하는 vocabulary size만큼의 차원
 - Curse of dimensionality (차원의 저주)
 - 생성되는 문서-단어 행렬 (document-term matrix) 가 너무 희소함 (too sparse).
 - 희소행렬 (sparse matrix) 은 모델링하기 까다로움

Bag-of-words

네이버 뉴스 1년치 데이터를 bag-of-words로 표현한다고 가정해봅시다.

- 하루에 뉴스가 10,000개 올라온다고 가정하면, 1년에 3,650,000개
- 이 안에 등장하는 단어들로 만든 vocabulary 크기를 50,000개로 가정



목차

1. Basics for Neural Networks

2. Bag-of-words

3. Neural Language Model

Neural probabilistic language model

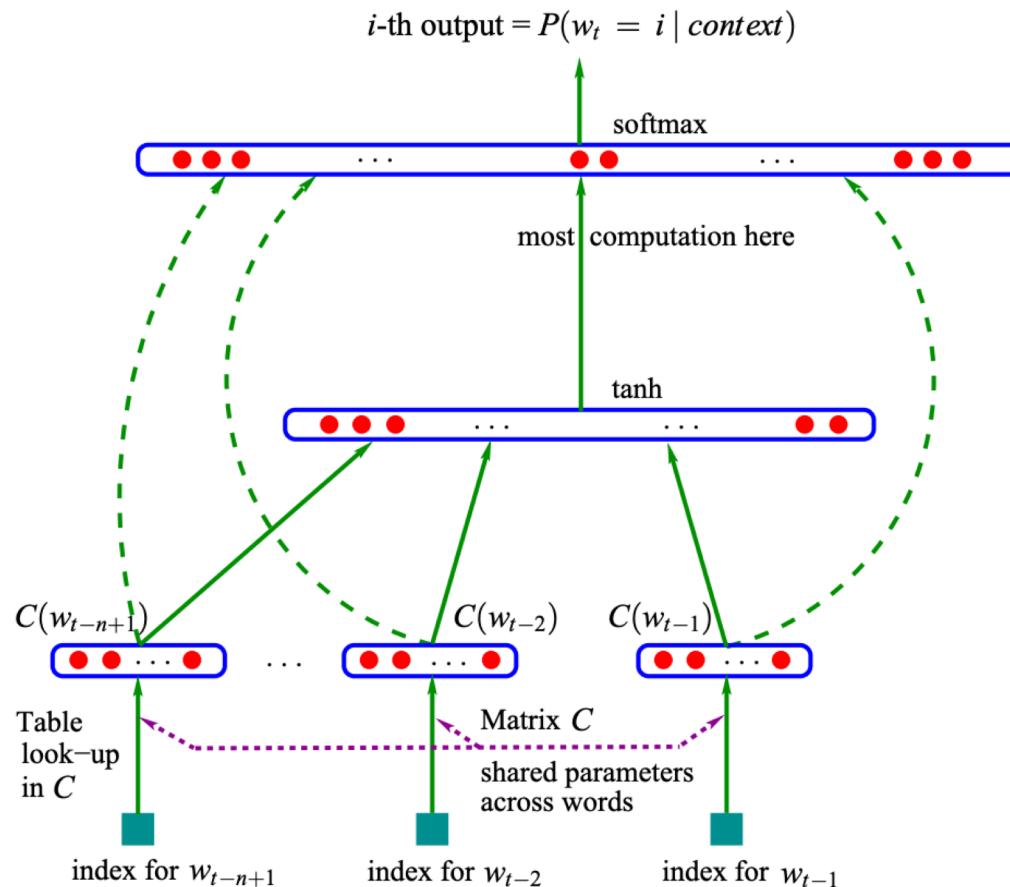


Figure 1: Neural architecture: $f(i, w_{t-1}, \dots, w_{t-n+1}) = g(i, C(w_{t-1}), \dots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i -th word feature vector.

Word2Vec

- Word2Vec은 단어 주변에 등장하는 다른 단어들의 분포의 유사성을 보존하는 벡터 표현 방법
 - 단어나 문서를 정해진 크기(d)의 공간 안에서 표현하는 방법.
 - 각 컬럼이 어떤 의미를 지니지는 않지만, 비슷한 단어/문서는 비슷한 벡터값을 지니도록 하는 방법

'dog'= [0.31, -0.21, 2.01, 0.58, ...]

'cat'= [0.45, -0.17, 1.79, 0.61, ...]

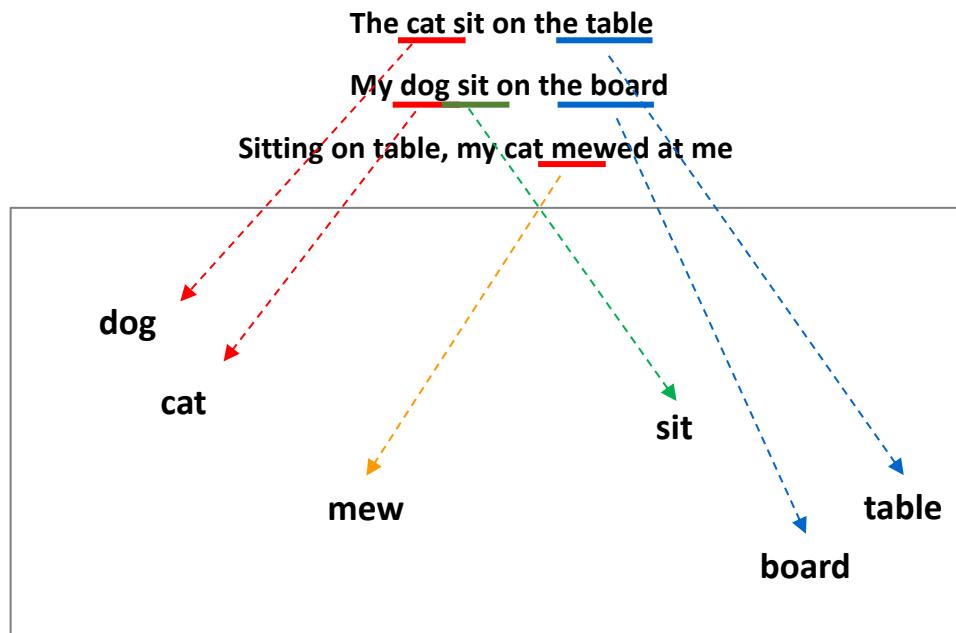
'topic modeling'= [-2.01, 0.03, 0.22, 0.54, ...]

'dim. reduction'= [-1.88, 0.11, 0.19, 0.45, ...]

Word2Vec

- Word2Vec은 단어 주변에 등장하는 다른 단어들의 분포의 유사성을 보존하는 벡터 표현 방법

- 주위에 등장하는 단어 분포가 유사한 두 단어는 비슷한 벡터를 지님
'cat' 대신 'dog'을 넣어도 큰 무리가 없음



< 단어의 의미 공간 >

Embedding

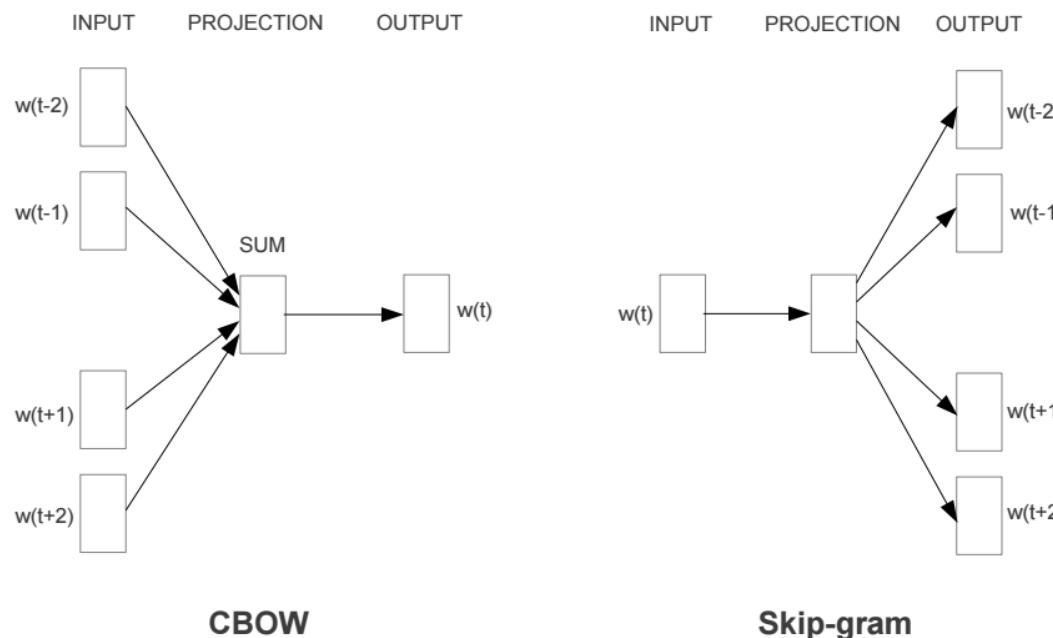
- 임베딩은 X 라는 공간의 데이터에서 원하는 정보를 잘 저장하며,
 Y 라는 새로운 공간으로 보내는 $f: X \rightarrow Y$ 함수
 - (예시) 10,000개 단어로 이루어진 문서 (1만차원)들의 유사도를 잘 보존하여 2차원으로 보내는 것
 - 어떤 정보를 보존할 것이냐에 따라서 다양한 임베딩 방법이 존재

Word2Vec

- Word2Vec은 행렬 형태로 데이터를 가공하지 않은 채 단어나 문서를 벡터로 표현하는 임베딩 방법
 - 행렬 형태로 가공하지 않는다는 것은 알고리즘을 돌릴 때의 입장이며,
 - 수학적으로 해석하면 매우 큰 sparse vector를 저차원의 dense vector로 표현하는 방법 *
 - Word2Vec, Doc2Vec을 이해하는 키는 그들이 “**어떤 정보를 보존**”하며 저차원 dense vector를 학습하는지를 파악하는 것

Word2Vec

- Word2Vec은 CBOW, Skip-gram 두 가지 형태가 있음
 - CBOW는 주위 단어로 현재 단어 $w(t)$ 를 예측하는 모델
 - Skipgram은 현재 단어 $w(t)$ 로 주위 단어 모두를 예측하는 모델



Word2Vec

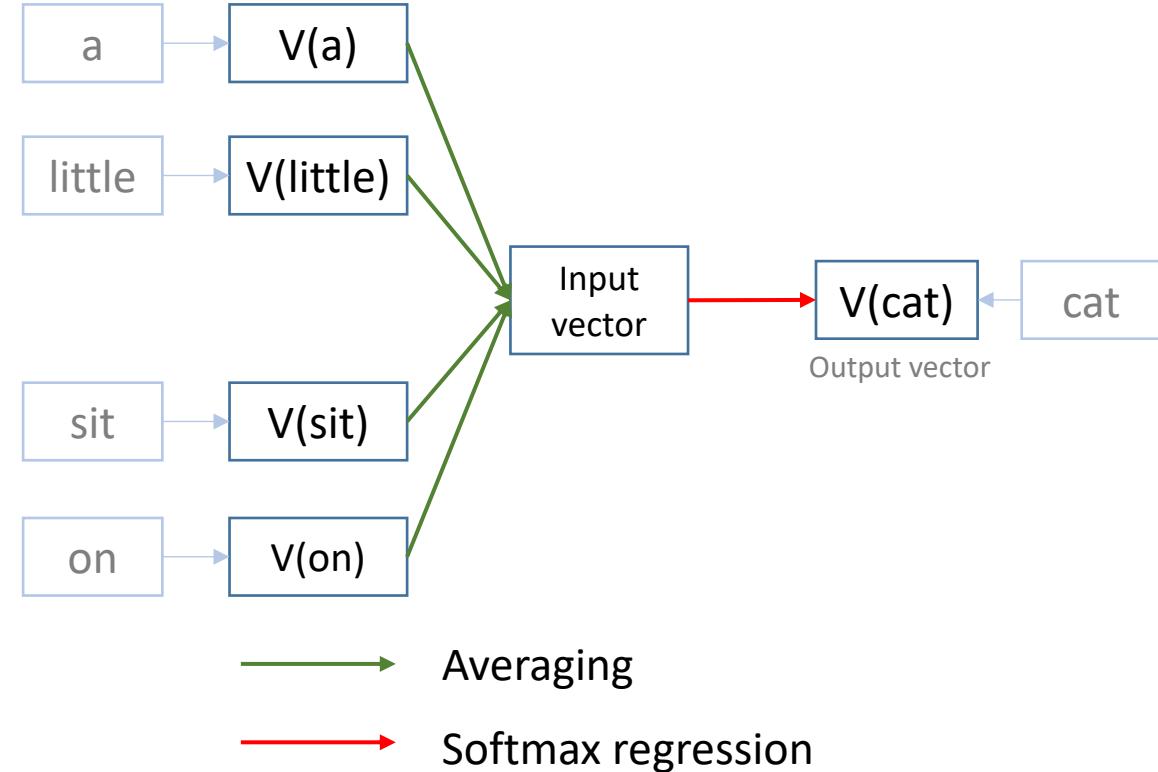
- Word2Vec은 $|V|$ class Softmax regression을 이용하여 각 단어의 벡터를 학습시키는 classifier

Lookup table

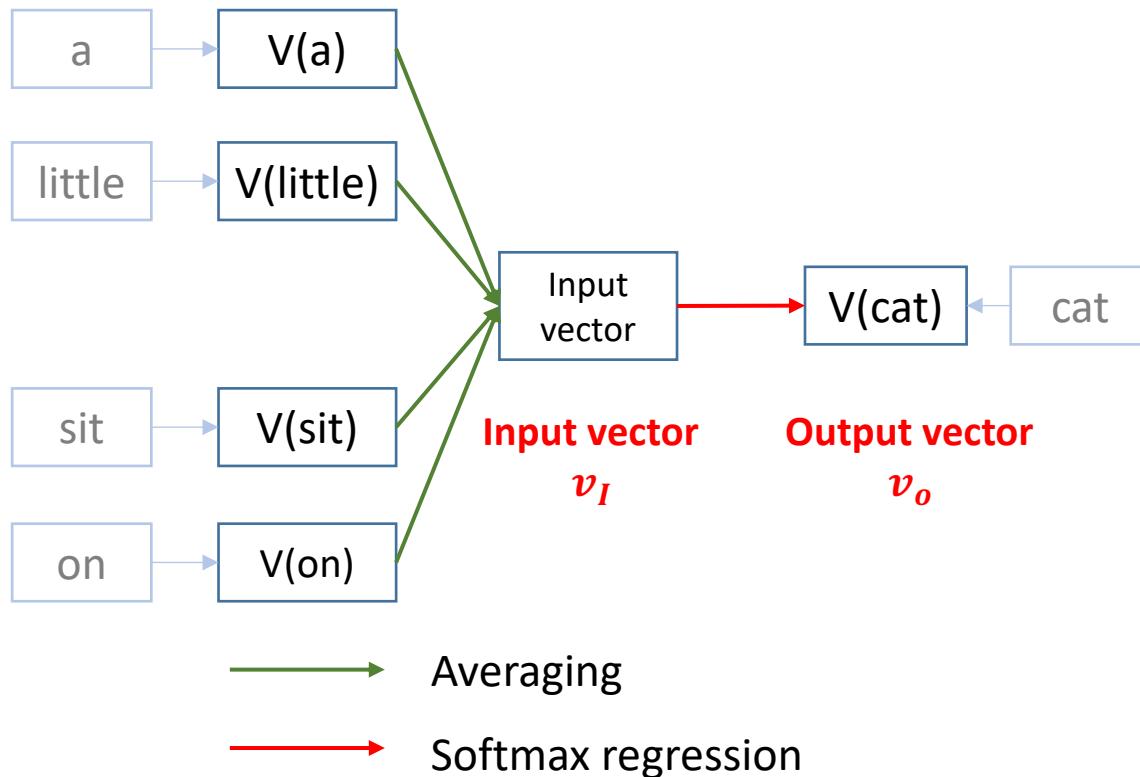
```
{cat: [.31, -.22, .54]  
dog: [.22, -0.3, -0.52]  
on: ...  
the: ...  
little: ...  
.... }
```

Lookup word vector

$$V('cat') = [.31, -.22, .54]$$



Word2Vec



$$y_j = \frac{\exp(v_I^T v_{Oj})}{\sum_j \exp(v_I^T v_{Oj})}$$

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

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

Logistic Regression

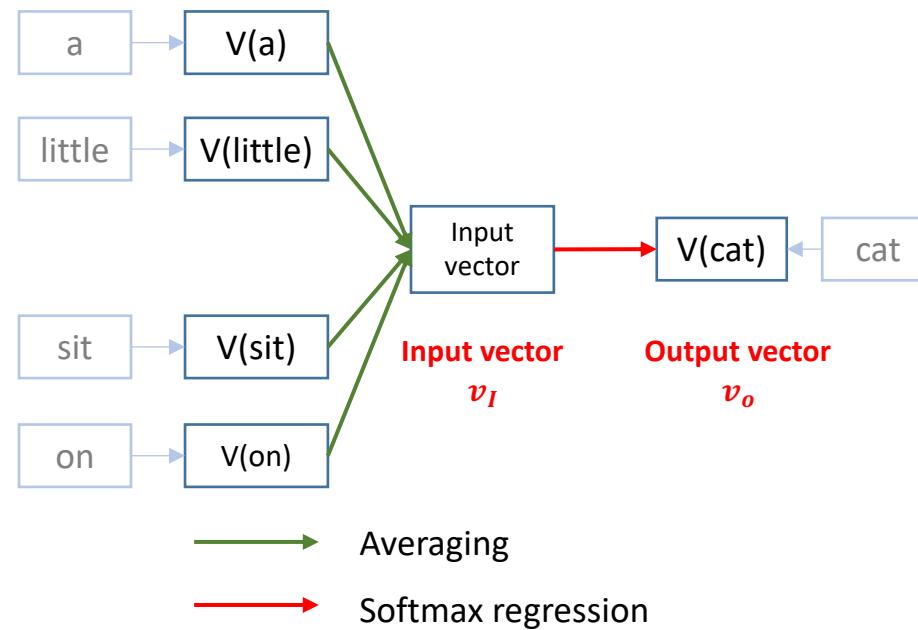
- Softmax regression은 multi class classification 알고리즘으로,
Logistic Regression은 Softmax regression의 특별한 경우

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

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

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

Word2Vec

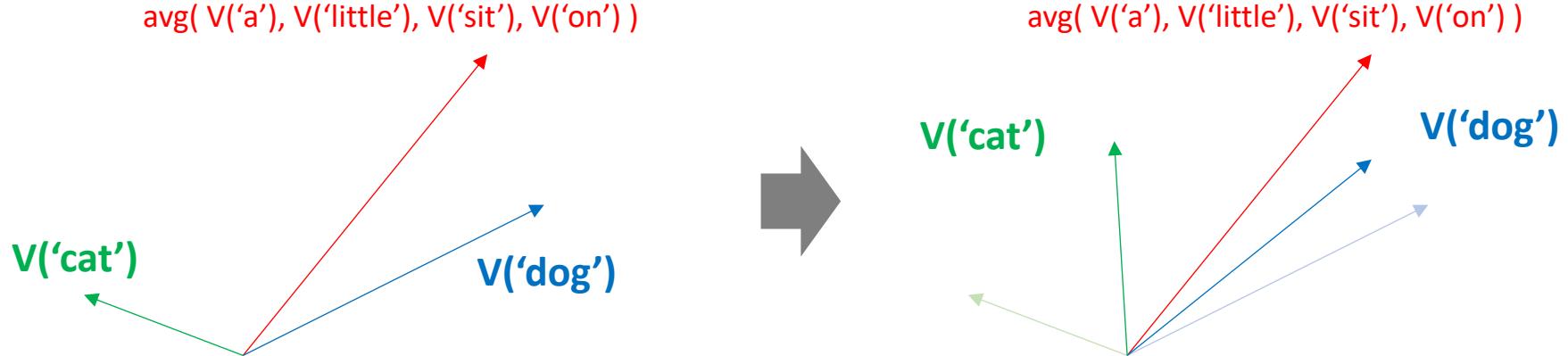


$$h_{\theta}(x) = \begin{bmatrix} P(w_{(t)} = cat) \\ P(w_{(t)} = dog) \\ P(w_{(t)} = table) \\ \dots \\ P(w_{(t)} = Vocab) \end{bmatrix} = \frac{1}{\sum_{j=1}^{|V|} \exp(\theta^{(j)T} x)} \begin{bmatrix} \exp(v(cat)^T v_I) \\ \exp(v(dog)^T v_I) \\ \exp(v(table)^T v_I) \\ \dots \\ \exp(v(Vocab)^T v_I) \end{bmatrix}$$

x : Input vector (average of contextual word vector)

Word2Vec

- ‘cat’과 ‘dog’이 비슷한 word embedding vector를 가지는 것은, 두 단어가 비슷한 문맥(contextual word distribution)을 가지기 때문이다
 - ‘cat’, ‘dog’의 두 단어 벡터 모두 context vector에 가깝게 이동

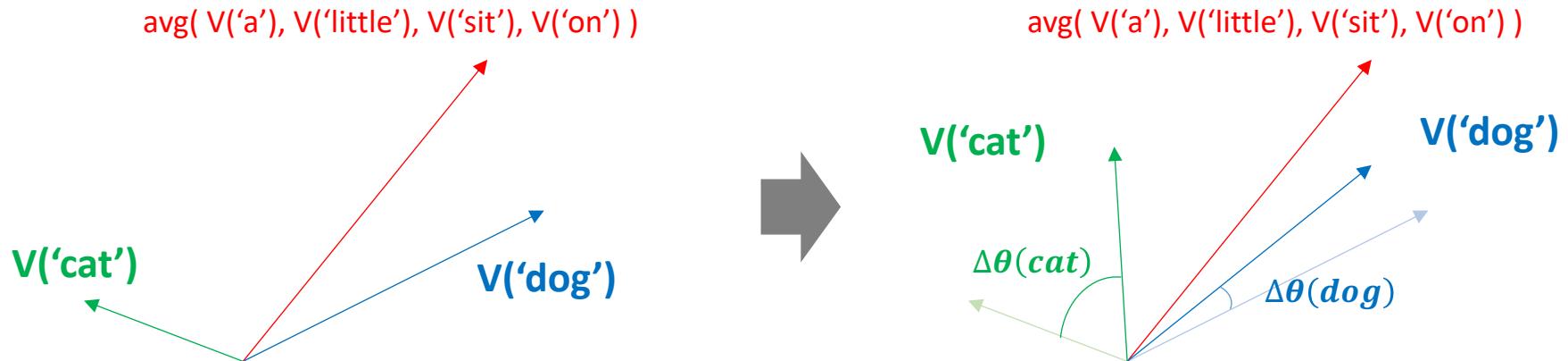


Word2Vec

- ‘cat’과 ‘dog’이 비슷한 word embedding vector를 가지는 것은, 두 단어가 비슷한 문맥(contextual word distribution)을 가지기 때문이다

- ‘cat’, ‘dog’의 두 단어 벡터 모두 context vector에 가깝게 이동
- Context vector와 차이가 많이나는 단어 ‘cat’은 학습량 (벡터의 변화량)도 큼

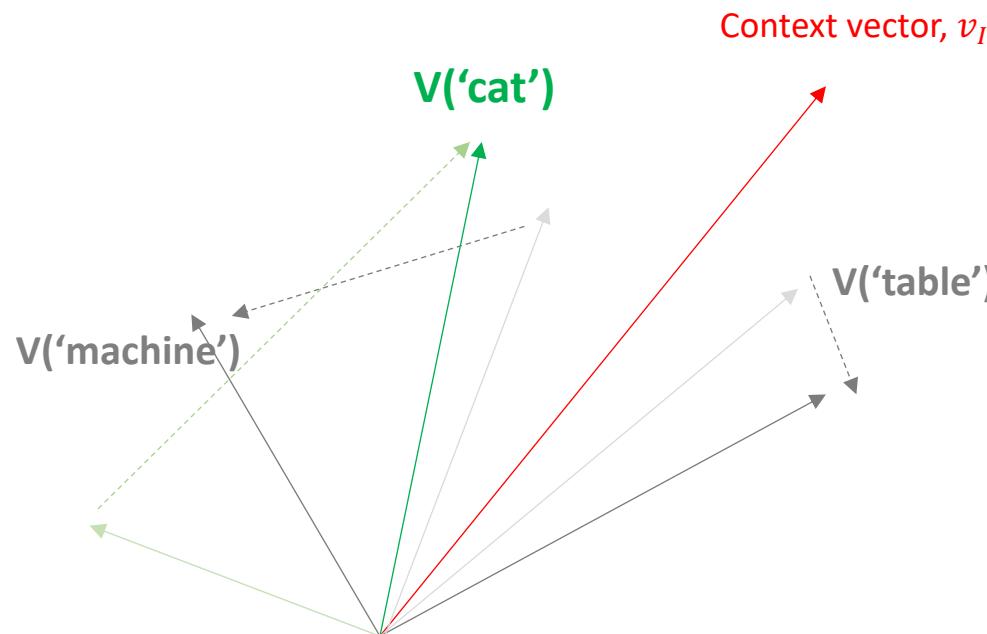
Softmax regression loss



Word2Vec

- Softmax regression은 알맞은 단어는 context vector 방향으로 당기고, 틀린 단어는 반대 방향으로 밀어내는 역할

$$h_{\theta}(x) = \begin{bmatrix} P(w_{(t)} = \text{cat}) \\ P(w_{(t)} = \text{dog}) \\ P(w_{(t)} = \text{table}) \\ \dots \\ P(w_{(t)} = \text{Vocab}) \end{bmatrix} = \frac{1}{\sum_{j=1}^{|V|} \exp(\theta^{(j)^T} x)} \begin{bmatrix} \exp(v(\text{cat})^T v_I) \\ \exp(v(\text{dog})^T v_I) \\ \exp(v(\text{table})^T v_I) \\ \dots \\ \exp(v(\text{Vocab})^T v_I) \end{bmatrix}$$



Word2Vec

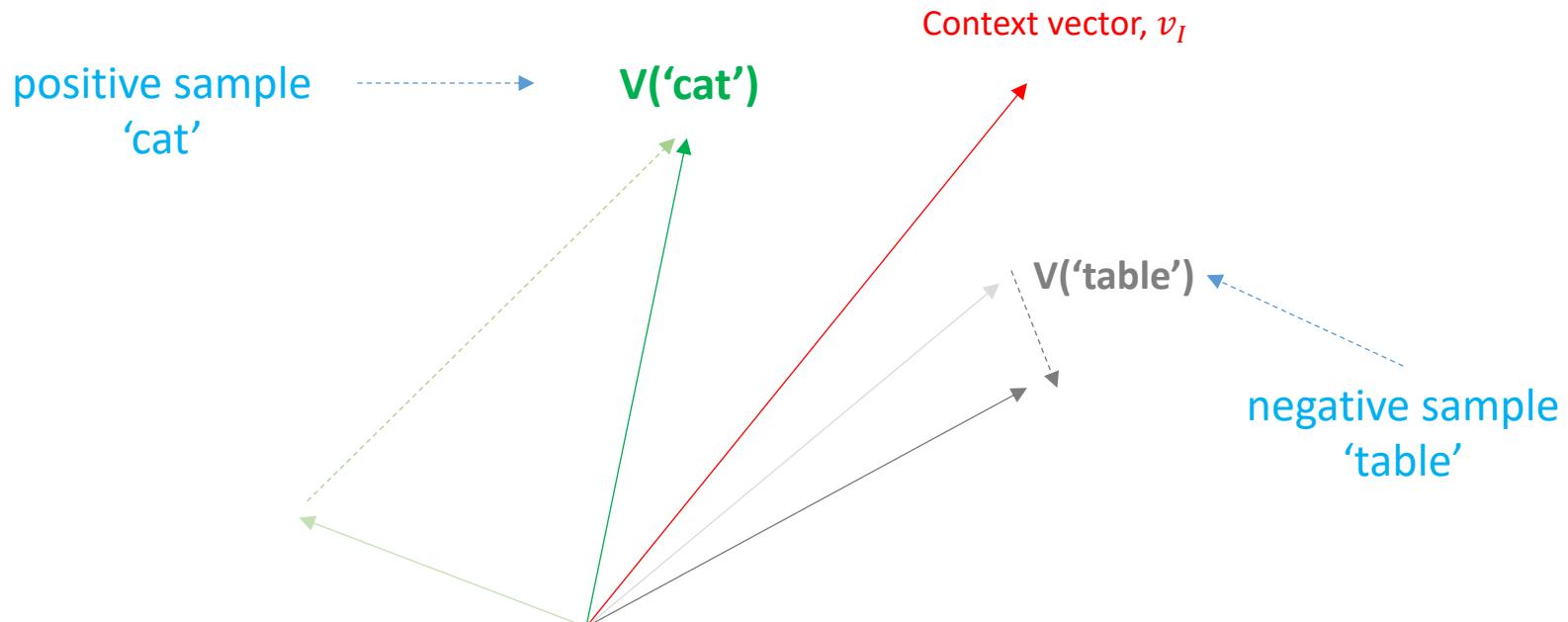
- Softmax regression은 한 단어의 벡터를 학습하기 위해 V 개의 모든 단어의 벡터를 수정하기 때문에 계산량이 매우 큼
 - 학습 데이터 $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ 가 주어졌을 때, loss function은

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

Word2Vec

- Negative sampling은 cat이 아닌 모든 단어를 밀어내는 것이 아니라, 임의로 샘플링한 단어들에 대해서만 context vector에서 밀어내는 것
 - A positive sample 'cat'과, 수십개 수준의 임의로 선정된 단어 negative samples

$$h_{\theta}(x) = \begin{bmatrix} P(w_{(t)} = \text{cat}) \\ P(w_{(t)} = \text{dog}) \\ P(w_{(t)} = \text{table}) \\ \dots \\ P(w_{(t)} = \text{Vocab}) \end{bmatrix} = \frac{1}{\sum_{j=1}^{|V|} \exp(\theta^{(j)T} x)} \begin{bmatrix} \exp(v(\text{cat})^T v_I) \\ \exp(v(\text{dog})^T v_I) \\ \exp(v(\text{table})^T v_I) \\ \dots \\ \exp(v(\text{Vocab})^T v_I) \end{bmatrix}$$



Word2Vec

• (조금 더 디테일하게, 그래서 minor한 문제),

negative sampling을 할 때, 각 단어를 선택하는 확률 모델을 만듦

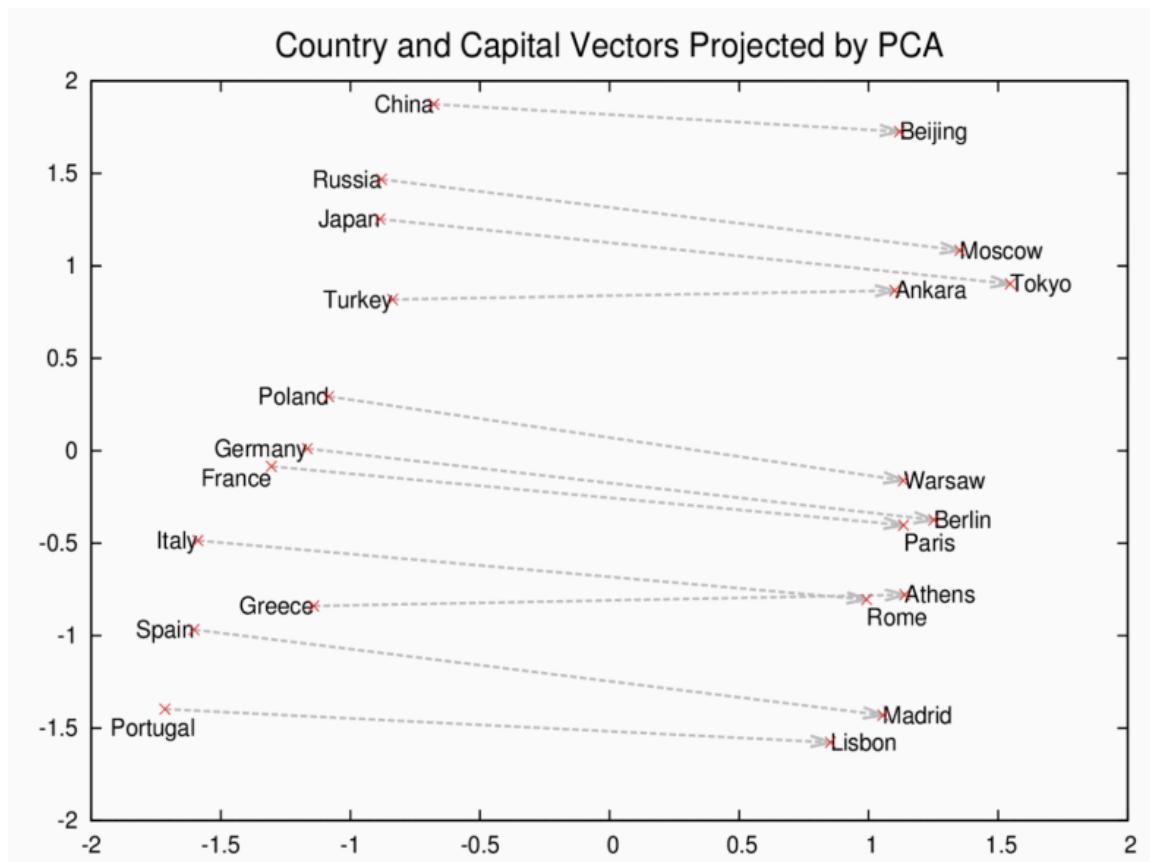
1. $U(w)^{\frac{3}{4}} / Z$: unigram 기준 단어 w 의 빈도수의 $\frac{3}{4}$ 승에 비례하게 단어를 선택
2. $P(w) = 1 - \sqrt{t / f(w)}$: 단어 w 가 negative samples로 뽑힌 뒤에, $P(w)$ 확률로만 학습에 이용함
 - $P(w)$ 는 단어의 빈도수가 높을수록 값이 작아지는데, frequent / infrequent words의 불균형을 맞추기 위한 보완

Word2Vec

- ‘cat’이 등장한 많은 문장이 있기 때문에, 한 번에 조금씩 v(‘cat’)을 학습하면 여러 문맥들을 모두 고려할 수 있음.
 - [a, little, cat, sit, on, table],
[my, pretty, cat, ran, out],
....
 - 조금씩 학습한다 = 작은 learning rate를 이용한다

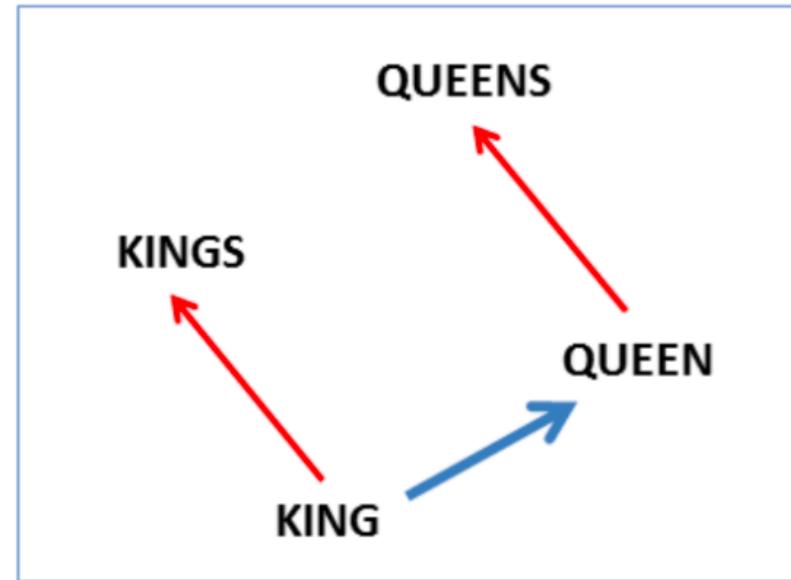
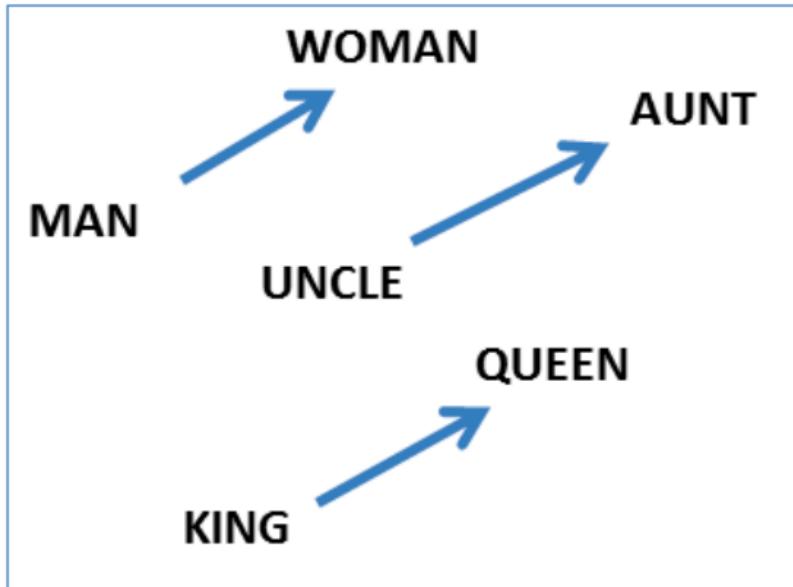
Word2Vec

- Word2Vec은 단어간의 관계성이 추출되는 효과가 있음
 - “ $v(\text{나라}) - v(\text{수도})$ ” 벡터가 나라마다 비슷



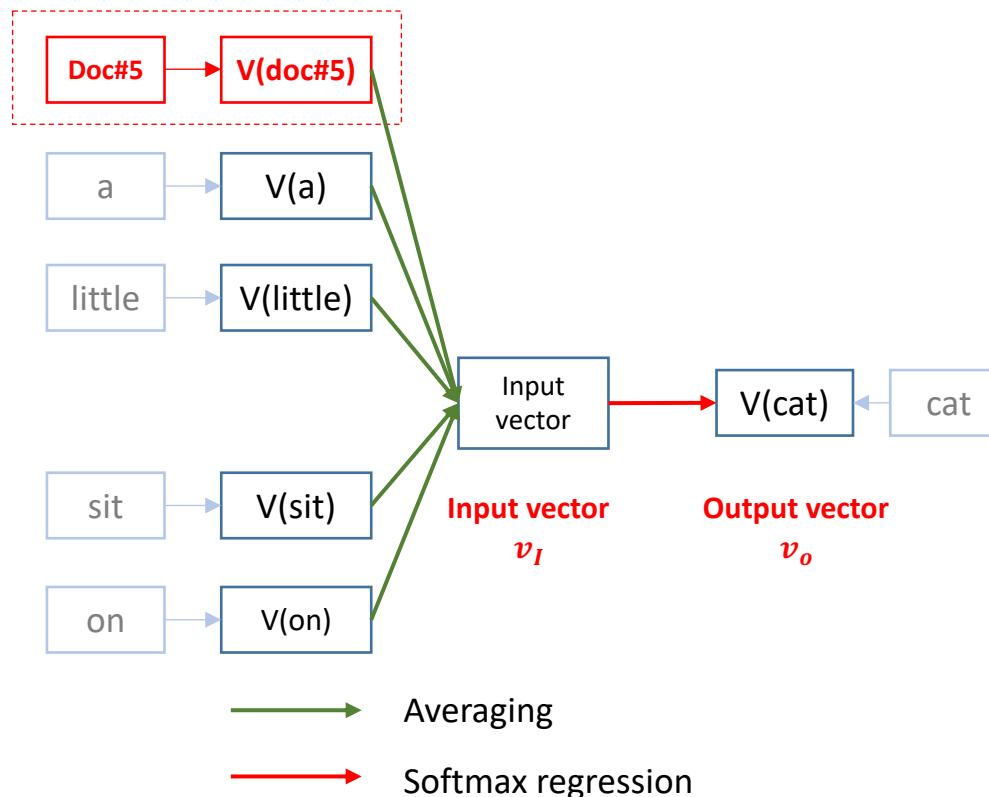
Word2Vec

- Word2Vec은 단어간의 관계성이 추출되는 효과가 있음
 - $V(\text{kings}) - V(\text{king}) = V(\text{queens}) - V(\text{queen})$



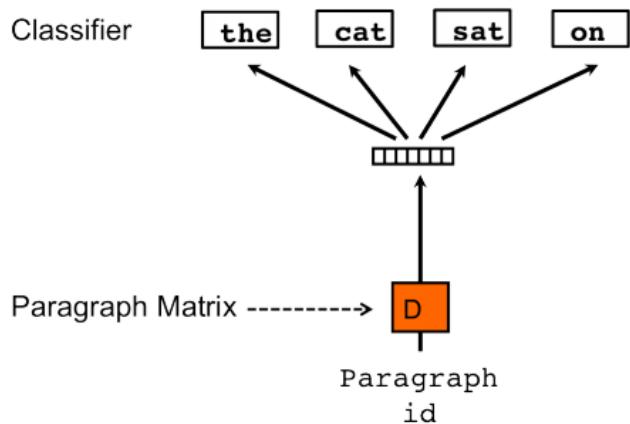
Doc2Vec

- Doc2Vec은 가상의 단어 Document Id를 삽입한 뒤, Word2Vec을 학습하는 것
 - Document id와 word가 같은 embedding 공간에 위치

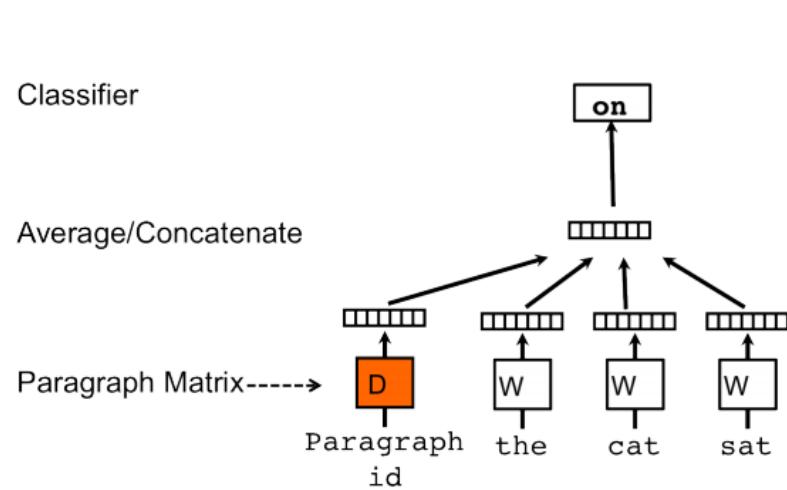


Doc2Vec

- Doc2Vec은 가상의 단어 Document Id를 삽입한 뒤, Word2Vec을 학습하는 것
 - Word2Vec처럼 2가지 모델 버전이 있음



Document id로 모든 단어를 예측하는
Softmax regression을 학습

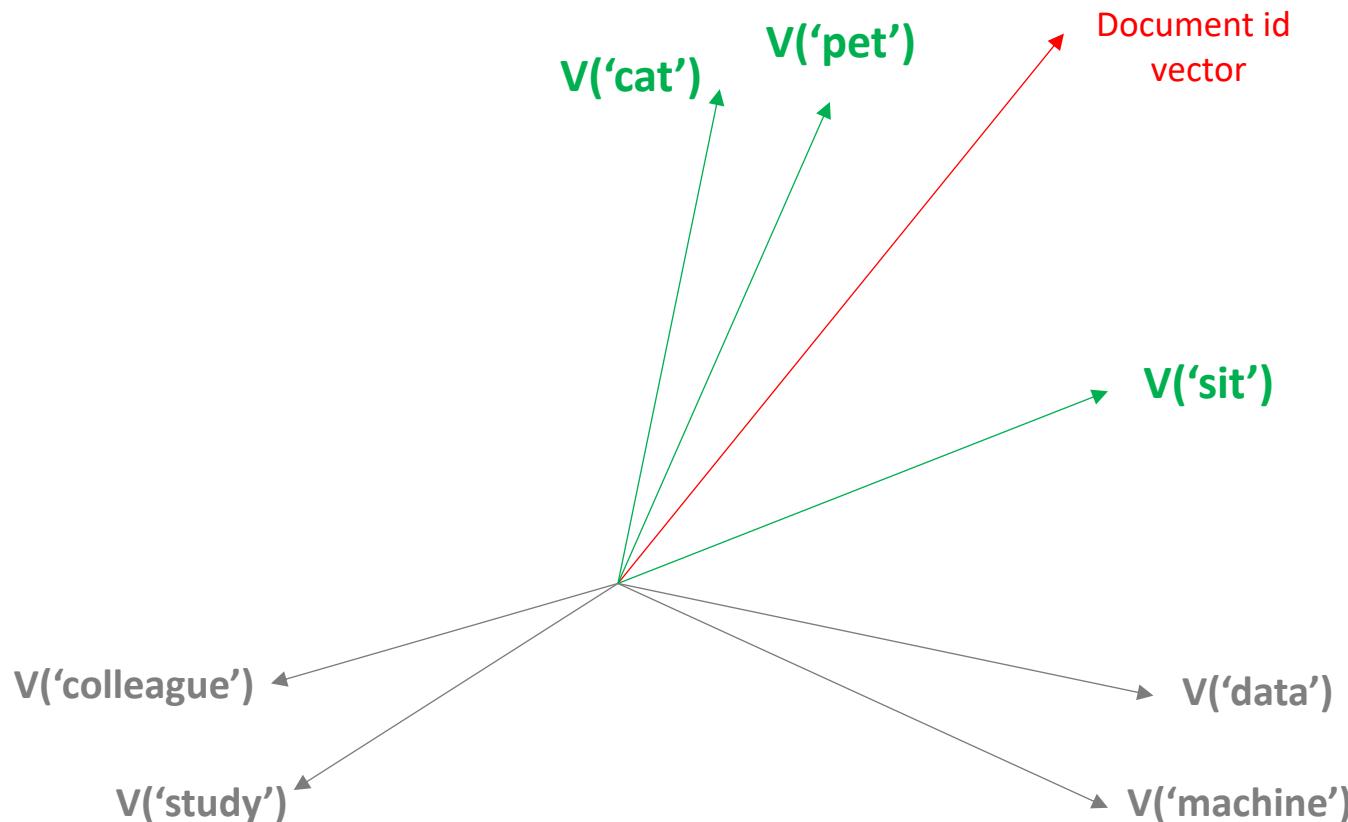


Document id를 단어처럼 취급하여
the, cat, sat 다음의 단어를 예측하는
Softmax regression을 학습

Doc2Vec

- Doc2vec은 한 문서에 등장한 단어 벡터들의 방향으로 document id vector를 위치시키도록 함

e.g) DOC: my pet, a little cat sit on table



Paragraph2Vec

- (Dai et al., 2015) 에서는 Document vector를 이용하여 문서간의 관계성을 추출할 수 있음을 확인
 - 영어 위키피디아의 각 문서 (doc)를 문서 내 단어로 임베딩하여 실험

Article	Cosine Similarity
Christina Aguilera	0.674
Beyonce	0.645
Madonna (entertainer)	0.643
Artpop	0.640
Britney Spears	0.640
Cyndi Lauper	0.632
Rihanna	0.631
Pink (singer)	0.628
Born This Way	0.627
The Monster Ball Tour	0.620

(a) Nearest neighbor of “Lady Gaga”

Article	Cosine Similarity
Ayumi Hamasaki	0.539
Shoko Nakagawa	0.531
Izumi Sakai	0.512
Urbangarde	0.505
Ringo Sheena	0.503
Toshiaki Kasuga	0.492
Chihiro Onitsuka	0.487
Namie Amuro	0.485
Yakuza (video game)	0.485
Nozomi Sasaki (model)	0.485

(b) “Lady Gaga” – “American” + “Japanese”

Paragraph2Vec

- Document vector의 의미는 한 문서 내의 단어 분포 (term frequency)의 압축
 - “Lady Gaga” – “American” + “Japanese” 는 “American”, “Japanese”에서 공통적인 단어들은 그대로 둔 체, “American”에 관계된 단어를 “Japanese”의 단어로 치환한 것과 같은 의미
 - 단어 분포의 압축이기 때문에, 문서의 주제를 표현하기에 적합.
- 소수의 단어에 큰 영향을 받지 않기 때문에, 특정한 단어의 유무가 중요한 작업에는 적합하지 않음.

Google's research

- Google + UCSF + University of Chicago Medicine + Stanford University
- 3개 병원의 모든 기록을 넣어 환자 상태 예측

npj | Digital Medicine www.nature.com/npjdigitalmed

ARTICLE **OPEN**

Scalable and accurate deep learning with electronic health records

Alvin Rajkomar^{1,2}, Eyal Oren¹, Kai Chen¹, Andrew M. Dai¹, Nissan Hajaj¹, Michaela Hardt¹, Peter J. Liu¹, Xiaobing Liu¹, Jake Marcus¹, Mimi Sun¹, Patrik Sundberg¹, Hector Yee¹, Kun Zhang¹, Yi Zhang¹, Gerardo Flores¹, Gavin E. Duggan¹, Jamie Irvine¹, Quoc Le¹, Kurt Litsch¹, Alexander Mossin¹, Justin Tansuwan¹, De Wang¹, James Wexler¹, Jimbo Wilson¹, Dana Ludwig², Samuel L. Volchenboum³, Katherine Chou¹, Michael Pearson¹, Srinivasan Madabushi¹, Nigam H. Shah⁴, Atul J. Butte², Michael D. Howell¹, Claire Cui¹, Greg S. Corrado¹ and Jeffrey Dean¹

Predictive modeling with electronic health record (EHR) data is anticipated to drive personalized medicine and improve healthcare quality. Constructing predictive statistical models typically requires extraction of curated predictor variables from normalized EHR data, a labor-intensive process that discards the vast majority of information in each patient's record. We propose a representation of patients' entire raw EHR records based on the Fast Healthcare Interoperability Resources (FHIR) format. We demonstrate that deep learning methods using this representation are capable of accurately predicting multiple medical events from multiple centers without site-specific data harmonization. We validated our approach using de-identified EHR data from two US academic medical centers with 216,221 adult patients hospitalized for at least 24 h. In the sequential format we propose, this volume of EHR data unrolled into a total of 46,864,534,945 data points, including clinical notes. Deep learning models achieved high accuracy for tasks such as predicting: in-hospital mortality (area under the receiver operator curve [AUROC] across sites 0.93–0.94), 30-day unplanned readmission (AUROC 0.75–0.76), prolonged length of stay (AUROC 0.85–0.86), and all of a patient's final discharge diagnoses (frequency-weighted AUROC 0.90). These models outperformed traditional, clinically-used predictive models in all cases. We believe that this approach can be used to create accurate and scalable predictions for a variety of clinical scenarios. In a case study of a particular prediction, we demonstrate that neural networks can be used to identify relevant information from the patient's chart.

npj Digital Medicine (2018)1:18; doi:10.1038/s41746-018-0029-1



1

Health systems collect and store electronic health records in various formats in databases.



JOHN DOE



2

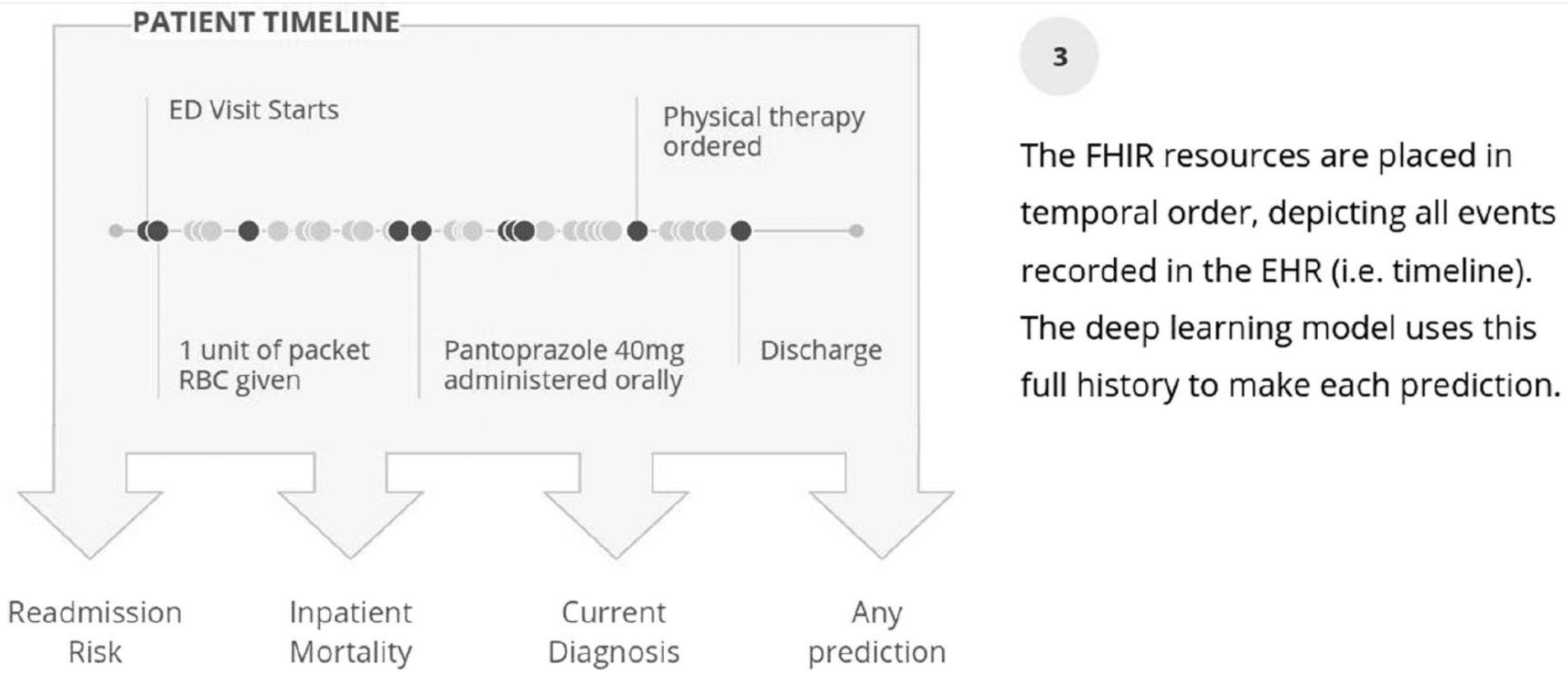
All available data for each patient is converted to events recorded in containers based on the Fast Healthcare Interoperability Resource (FHIR) specification.

12:40 PM - Notes
Hospitalist History
and Physical: This
is a ...

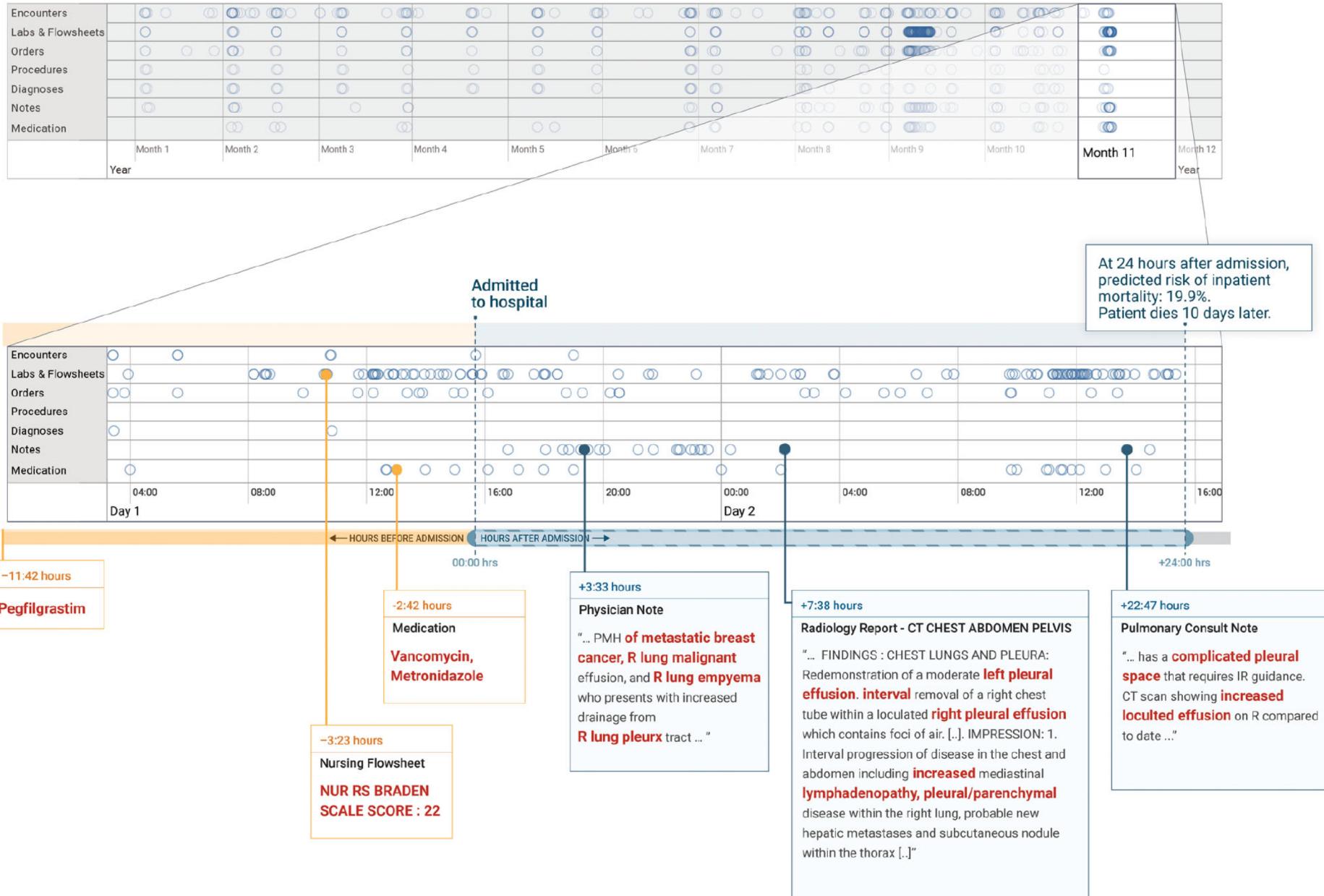
4:21 PM - Order
CBC Ordered

6:50 PM - Test Result
Hemoglobin
result: 6.5 g/dL

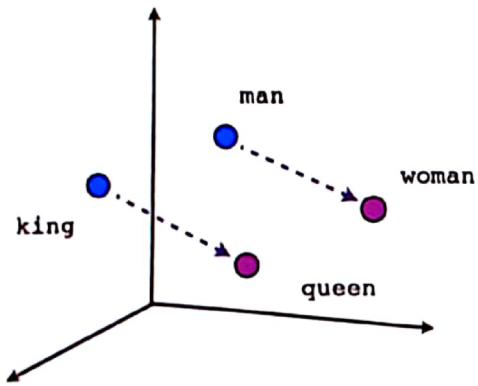
3



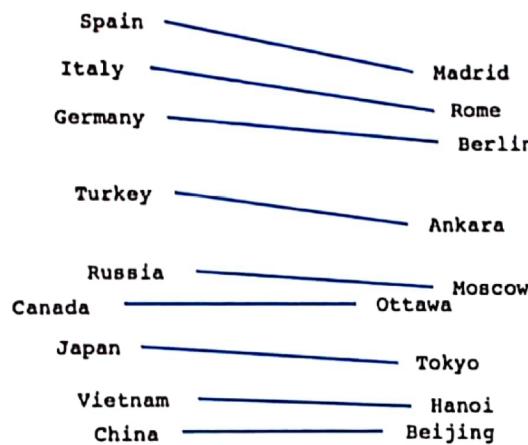
Patient Timeline



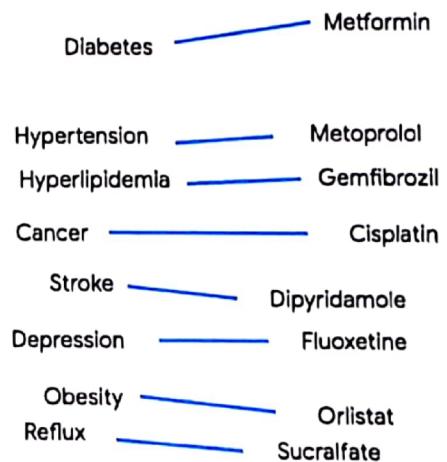
Using Embeddings to handle text



Male - Female

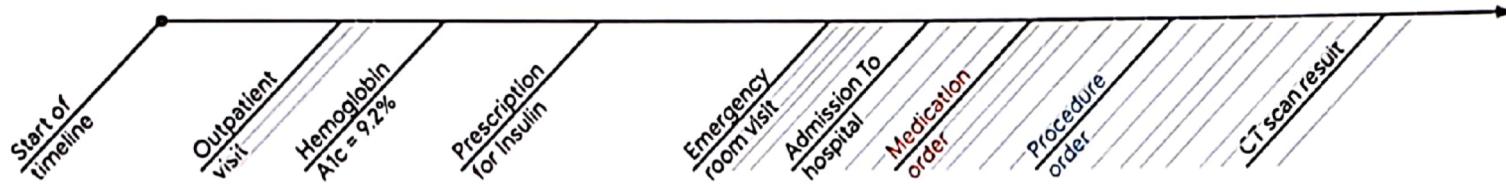


Country - Capital



Disease - Drug

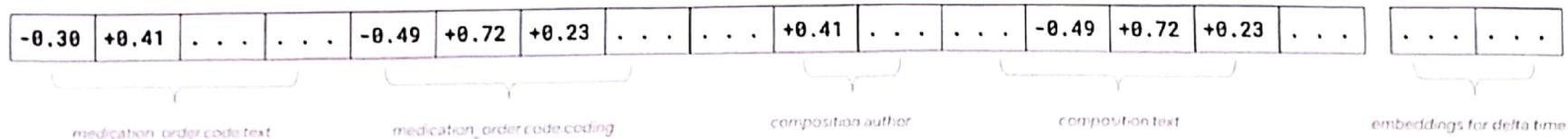
Patient Timeline



FHIR Resource

<u>Token ID</u>	<u>Embedding</u>					<u>Weight</u>
<1-17 >	-0.30	+0.41		+0.60
<2-35 >	-0.49	+0.72	+0.23	. . .		+0.60
<3-85 >	-0.32	+0.39		+0.60
<4-702>	-0.32	+0.39		+0.60
<3-19 >	-0.51	+0.73	+0.21	. . .		+0.60
<4-913>	-0.32	+0.39		+0.60
<5-137>	-0.13	+0.41	+0.23	. . .		+0.60
<5-139>	-0.78	+0.41	-0.98	. . .		+0.60
<6-21 >	-0.30	+0.41	+0.23	. . .		+0.60

Input to Recurrent Neural Network at a single timestep



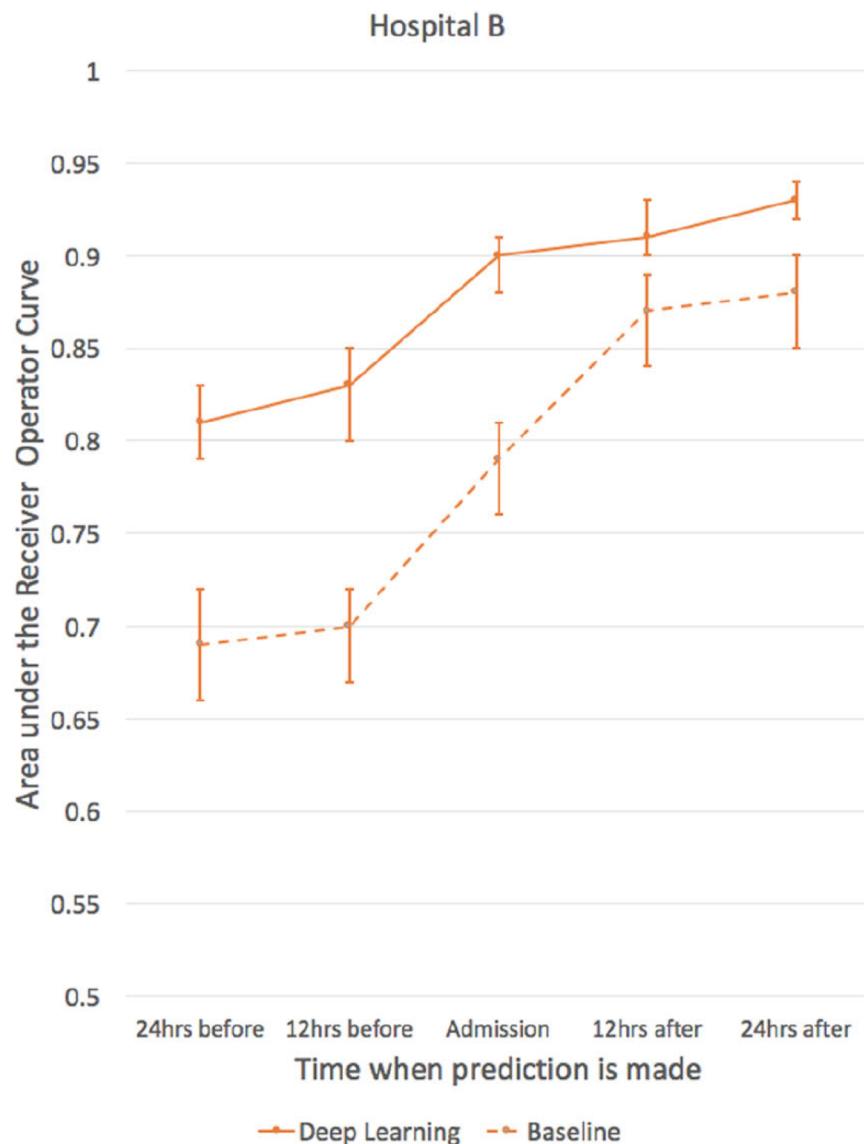
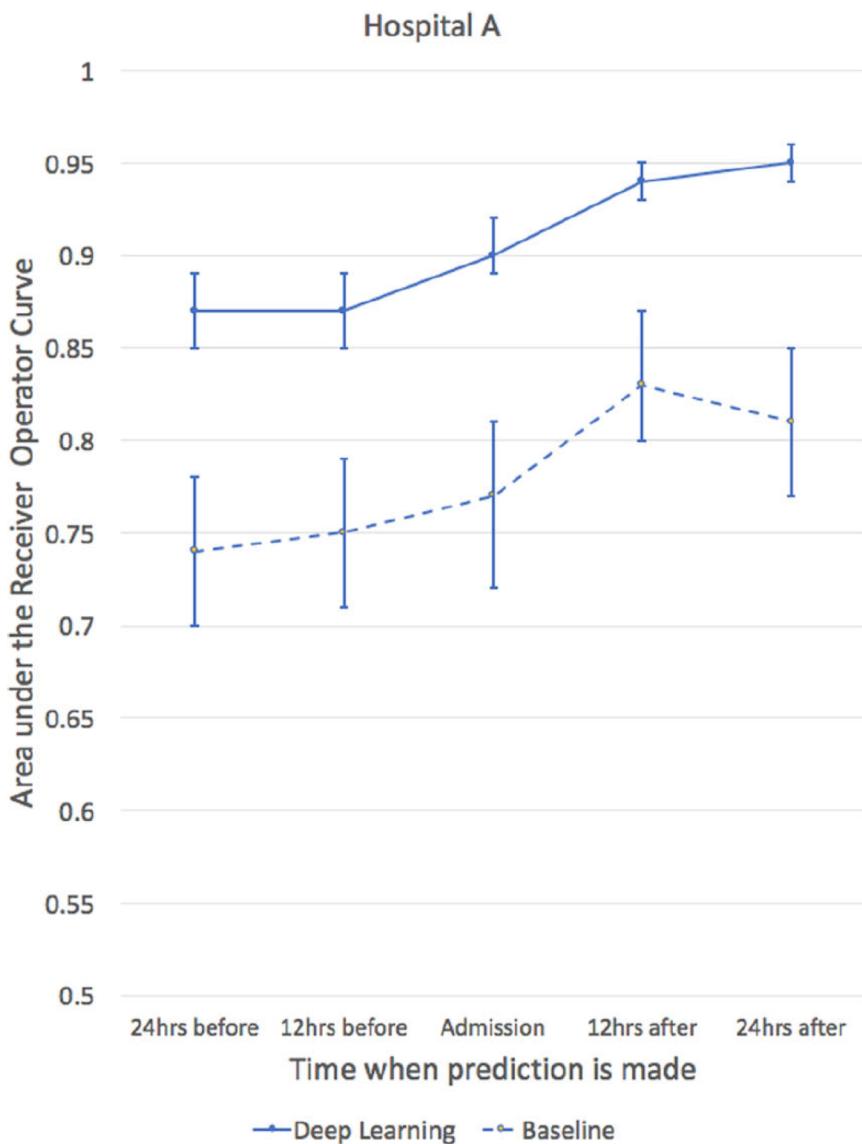


Table 2. Prediction accuracy of each task made at different time points

	Hospital A	Hospital B
<i>Inpatient mortality, AUROC^a (95% CI)</i>		
24 h before admission	0.87 (0.85–0.89)	0.81 (0.79–0.83)
At admission	0.90 (0.88–0.92)	0.90 (0.86–0.91)
24 h after admission	0.95 (0.94–0.96)	0.93 (0.92–0.94)
Baseline (aEWS ^b) at 24 h after admission	0.85 (0.81–0.89)	0.86 (0.83–0.88)
<i>30-day readmission, AUROC (95% CI)</i>		
At admission	0.73 (0.71–0.74)	0.72 (0.71–0.73)
At 24 h after admission	0.74 (0.72–0.75)	0.73 (0.72–0.74)
At discharge	0.77 (0.75–0.78)	0.76 (0.75–0.77)
Baseline (mHOSPITAL ^c) at discharge	0.70 (0.68–0.72)	0.68 (0.67–0.69)
<i>Length of stay at least 7 days, AUROC (95% CI)</i>		
At admission	0.81 (0.80–0.82)	0.80 (0.80–0.81)
At 24 h after admission	0.86 (0.86–0.87)	0.85 (0.85–0.86)
Baseline (Liu ^d) at 24 h after admission	0.76 (0.75–0.77)	0.74 (0.73–0.75)
<i>Discharge diagnoses (weighted AUROC)</i>		
At admission	0.87	0.86
At 24 h after admission	0.89	0.88
At discharge	0.90	0.90

^aArea under the receiver operator curve

^bAugmented Early Warning System score

^cModified HOSPITAL score for readmission

^dModified Liu score for long length of stay

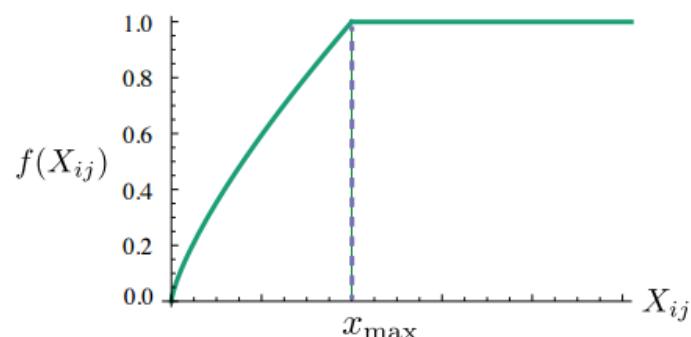
The bold values indicate the highest area-under-the-receiver-operator-curve for each prediction task

[참고] Glove, FastText

GloVe

- Word2Vec은 $P(W_t | W_{[t-2:t+2]})$ 처럼 단어의 등장 확률을 보존하지만, GloVe는 두 단어 w_i, w_j 의 co-occurrence frequency X_{ij} 를 보존하는 임베딩
 - 두 단어의 임베딩 벡터의 곱이 log co-occurrence 가 되도록 임베딩

$$J = \sum_{i,j} f(X_{ij}) * \left(\mathbf{w}_i^T \mathbf{w}_j + \mathbf{b}_i + \mathbf{b}_j - \log(X_{ij}) \right)^2$$



$f(X_{ij})$: co-occurrence에 따른 loss에서의 가중치:
빈도수가 높은 두 단어 w_i, w_j 를 더 잘 학습하도록 유도

FastText

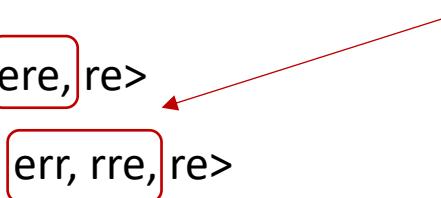
- Mikolov가 out of vocabulary 문제 해결을 위하여 제안한 방법
- 하나의 단어는 여러 개의 subwords로 표현할 수 있음
 - ‘appear’ = ['app', ‘pea’, ‘ar’, ‘pear’]
- 단어 w 의 벡터 $W_{(t)}$ 를 subwords의 벡터의 합으로 표현
 - $W_{(t)} = \sum_{g \in G_w} z_g$
 - ‘appear’의 경우, ['app', ‘pea’, ‘ar’, ‘pear’]이 z_g 에 해당

FastText

- Word Piece Model / Byte Pair Encoding은 단어를 겹치지 않는 subwords 유닛으로 만듦
 - ‘appear’ → ‘app’ + ‘ear’
- Fasttext는 bag of character n-gram으로 단어를 표현
 - 단어의 시작과 끝을 구분하기 위해 <, >를 추가
 - Character 3-gram을 이용할 경우, ‘where’ → ‘<wh, whe, her, ere, re>’
 - where이라는 단어는 <wh 부터 re>까지의 5개의 subwords로 나뉘어짐
 - 논문에서는 3 ~ 6 gram의 모든 subwords를 합쳐서 사용

FastText

- Fasttext는 노이즈가 있거나 미등록단어가 발생할 때 이를 embedding vector로 표현하기 위한 방법

- where → <wh, whe, her, ere, re>
 - wherre → <wh, whe, her, err, rre, re>
- 서로 다른 subwords
- 

- 서로 다른 subwords가 있더라도 대부분이 공통된 subwords라면 두 단어, 'where', 'wherre'는 비슷한 단어 벡터를 가짐

FastText

Word2Vec Loss

$$\log(1 + \exp(-w_{(t)}^T w_{[t-w:t+w]})) + \sum_{n \in N_{t,c}} \log(1 + \exp(w_{(t)}^T w_n))$$



FastText Loss

$$\log\left(1 + \exp\left(-\sum_{g \in G_w} w_g^T w_{[t-w:t+w]}\right)\right) + \sum_{n \in N_{t,c}} \log\left(1 + \exp\left(\sum_{g \in G_w} w_g^T w_{[t-w:t+w]}\right)\right)$$

Positive sample loss Negative sample loss

FastText (text classification)

- Facebook Research github* 에 들어가면 세 개의 논문이 참조문헌으로
잡혀있음

1. Enriching Word Vectors with Subword Information
2. Bag of Tricks for Efficient Text Classification
3. FastText.zip: Compressing text classification models
(효율적인 문서 분류를 위한 모델 압축)

- 앞서 말한 unsupervised word embedding은 1번 논문

FastText (text classification)

- 2번 논문은 문서 분류를 위한 단어 임베딩에 대하여 이야기하는 논문
 - Word2Vec, Doc2Vec, GloVe, FastText (word embedding)은 모두 label 데이터가 있지 않는 상황에서 단어를 벡터로 표현하는 알고리즘들

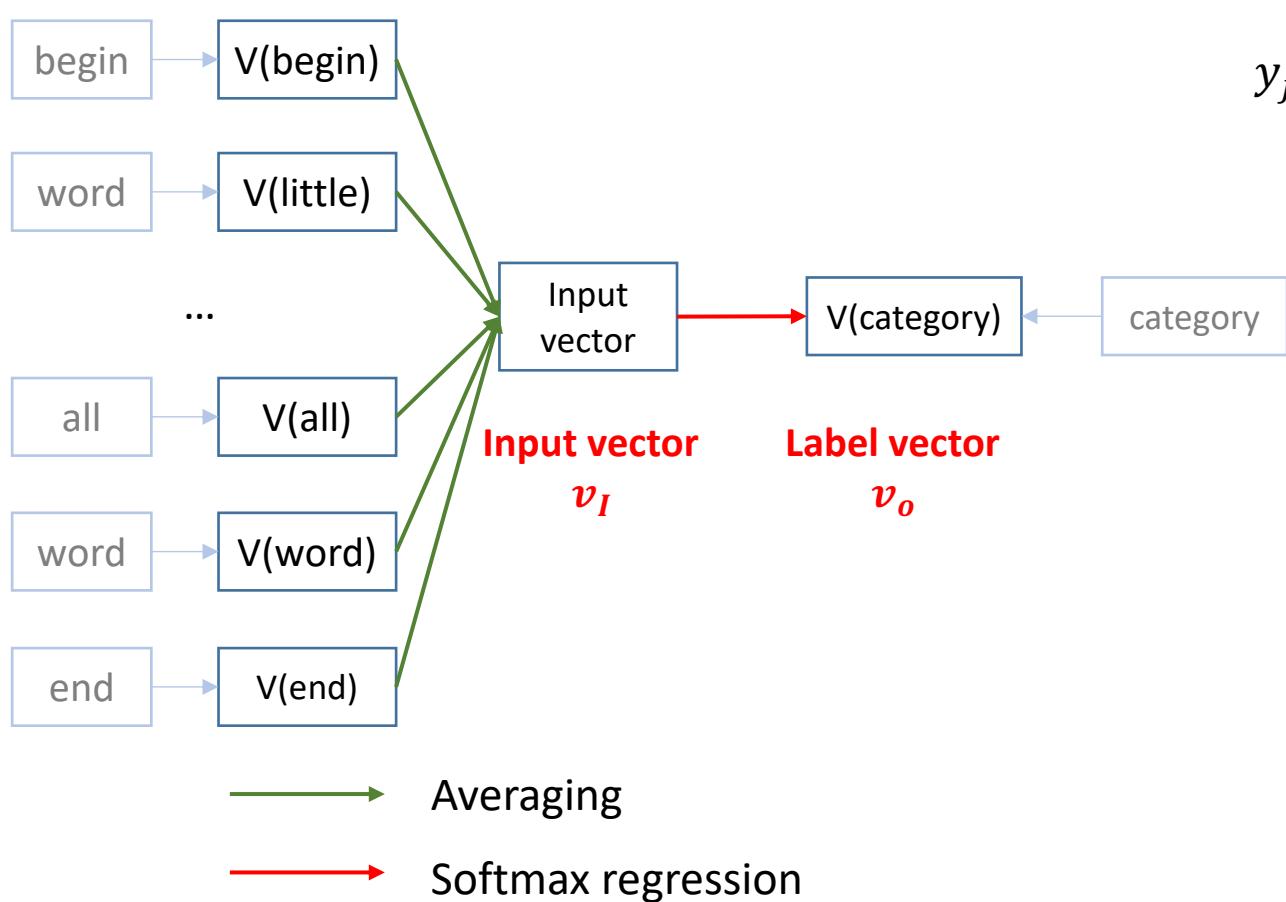
FastText (text classification)

- 문서 분류를 위한 정답데이터가 있을 경우, 이를 적극적으로 이용하자
 - 문맥의 유사성을 보존하는 임베딩 방법을 이용하면 ‘good’, ‘bad’가 비슷한 벡터를 지님
 - 하지만 감성분석 (sentiment classification)의 관점에서는 ‘good/bad’는 구분이 되어야 함
 - “Bag of Tricks ... ”논문에서는 정답데이터를 이용하여 단어를 임베딩하는 방법과 문서 분류 자체에 대하여 이야기 하고 있음

FastText (text classification)

- Doc2Vec과 매우 유사한 구조.

- (1) Window의 크기가 문서 전체이며 / (2) 가운데 단어가 아닌 class y 를 예측



$$y_j = \frac{\exp(v_I^T v_{Lj})}{\sum_j \exp(v_I^T v_{Lj})}$$

FastText (text classification)

- 다양한 데이터의 실험에서 충분히 좋은 성능을 보여줌

- 심지어 Yelp tag prediction 문제 같은 경우에는 label의 개수만 312K 개였지만, 복잡한 딥러닝 모델들과 비교하여도 충분히 좋은 성능을 보임
- 애초에 Bag of Words 모델이 기본적으로 좋은 성능을 보여줌

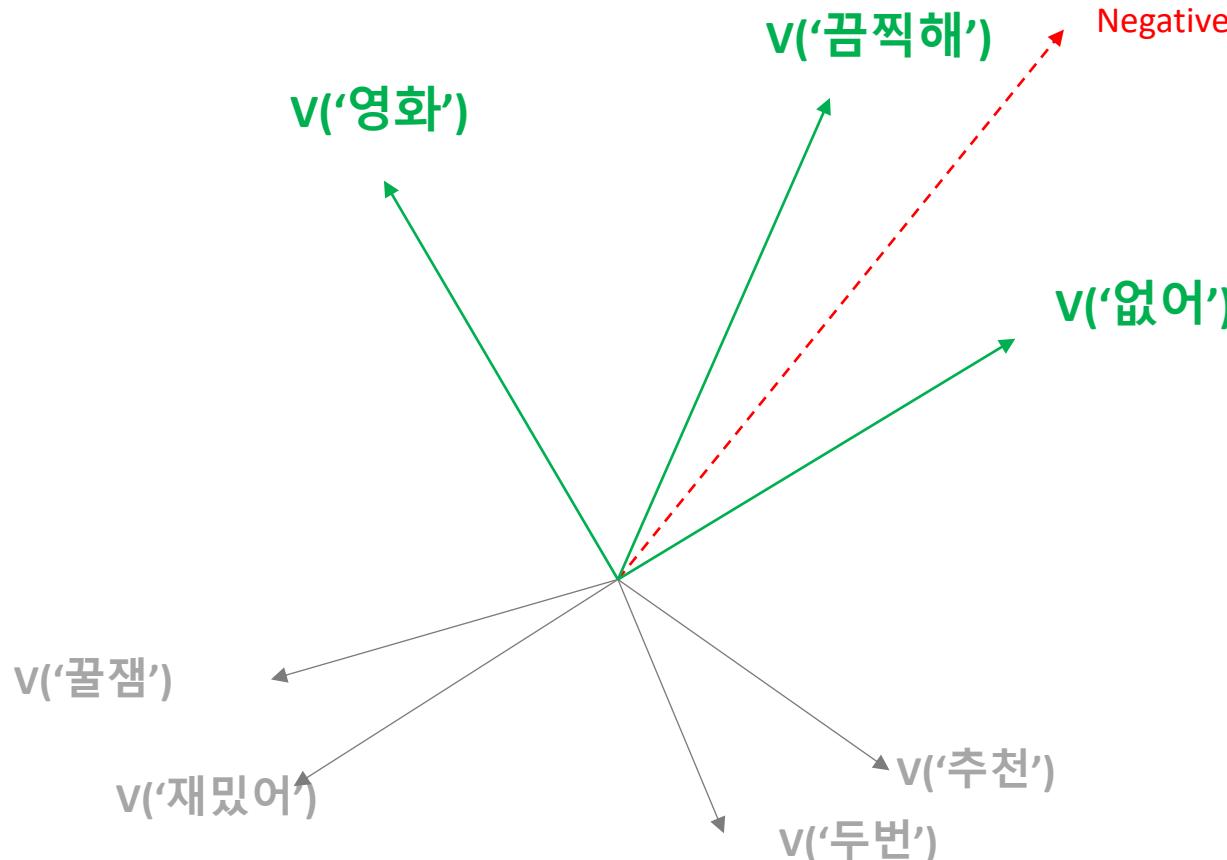
Model	AG	Sogou	DBP	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW (Zhang et al., 2015)	88.8	92.9	96.6	92.2	58.0	68.9	54.6	90.4
ngrams (Zhang et al., 2015)	92.0	97.1	98.6	95.6	56.3	68.5	54.3	92.0
ngrams TFIDF (Zhang et al., 2015)	92.4	97.2	98.7	95.4	54.8	68.5	52.4	91.5
char-CNN (Zhang and LeCun, 2015)	87.2	95.1	98.3	94.7	62.0	71.2	59.5	94.5
char-CRNN (Xiao and Cho, 2016)	91.4	95.2	98.6	94.5	61.8	71.7	59.2	94.1
VDCNN (Conneau et al., 2016)	91.3	96.8	98.7	95.7	64.7	73.4	63.0	95.7
fastText, $h = 10$	91.5	93.9	98.1	93.8	60.4	72.0	55.8	91.2
fastText, $h = 10$, bigram	92.5	96.8	98.6	95.7	63.9	72.3	60.2	94.6

Table 1: Test accuracy [%] on sentiment datasets. FastText has been run with the same parameters for all the datasets. It has 10 hidden units and we evaluate it with and without bigrams. For char-CNN, we show the best reported numbers without data augmentation.

FastText (text classification)

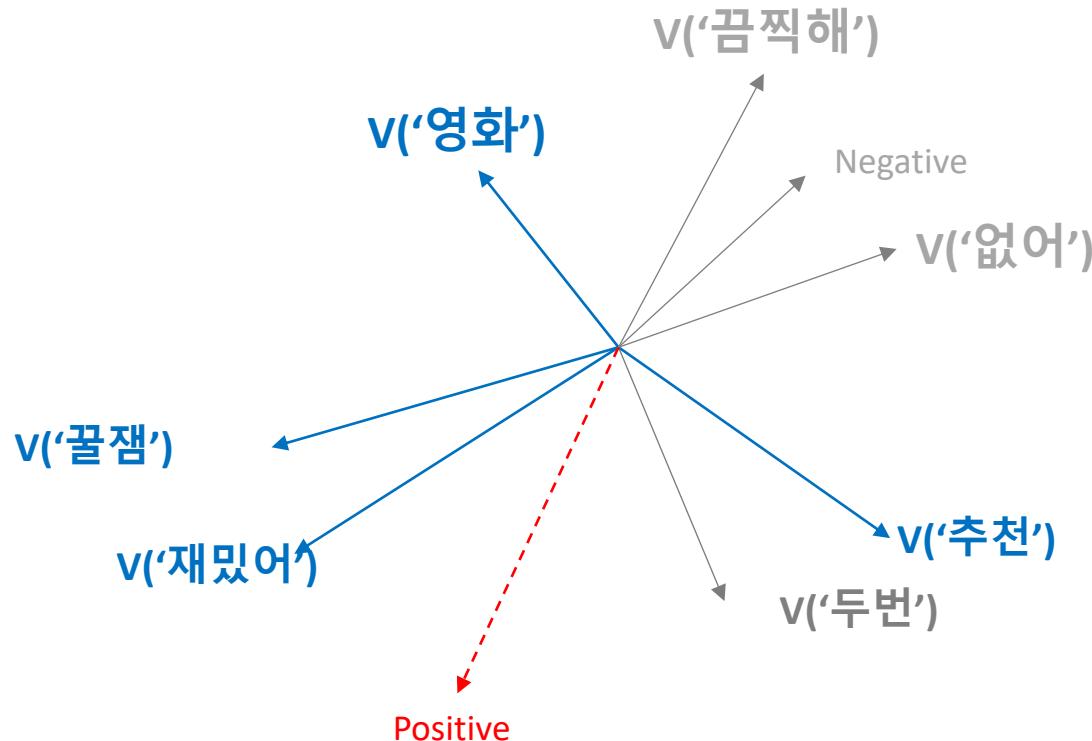
- FastText은 특정 label에 자주 등장한 단어를 자신과 같은 방향으로 끌어당기는 것

Negative: [이, 영화, 진짜, 끔찍해, 재미, 없어]



FastText (text classification)

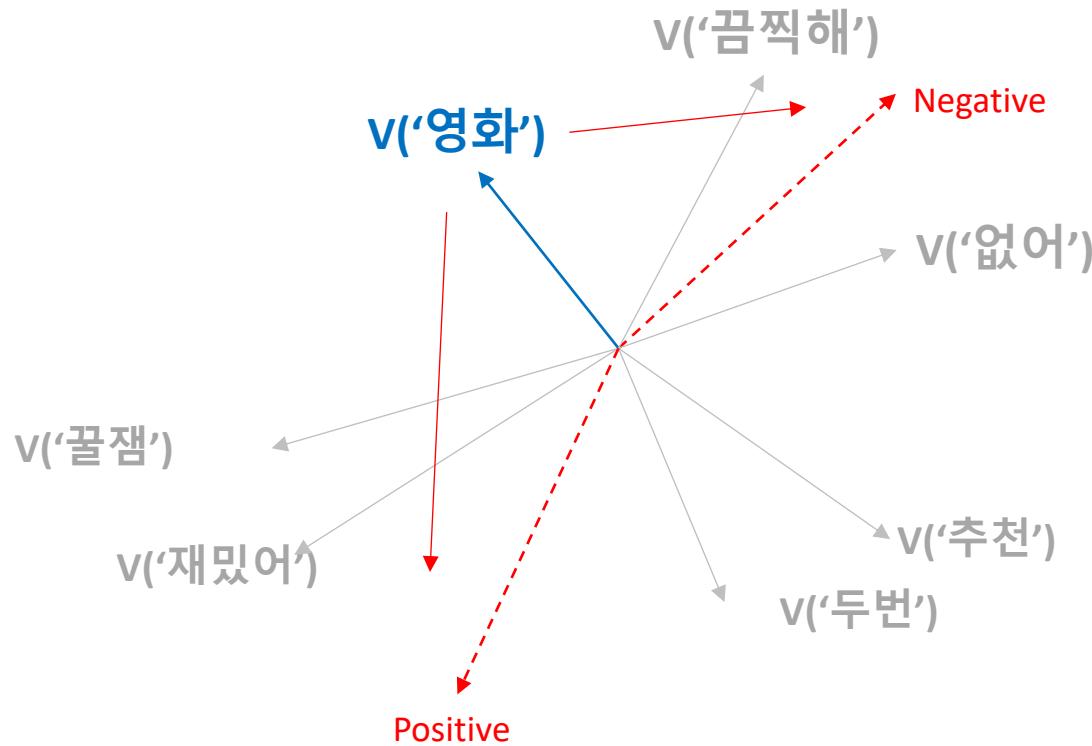
- FastText은 특정 label에 자주 등장한 단어를 자신과 같은 방향으로 끌어당기는 것



Positive: [이, 영화, 진짜, 꿀잼, 완전, 재밌어, 추천, 해]

FastText (text classification)

- 여러 labels에 모두 등장하는 단어는 어떤 label과도 가까워지기 어려움



FastText (text classification)

- ‘not bad’는 부정의 의미가 아니지만, 단어를 하나씩 embedding 하면 이와 같은 경우가 제대로 학습되지 않음
- bigram까지만 이용해도 이런 문제는 충분히 해결됨

