Deep Learning for NLP
- Word Embedding -

Ko, Youngjoong

September 9, 2015

Dept. of Computer Engineering,
Dong-A University

Contents

1. **Basic Concepts of Neural Network (NN)**

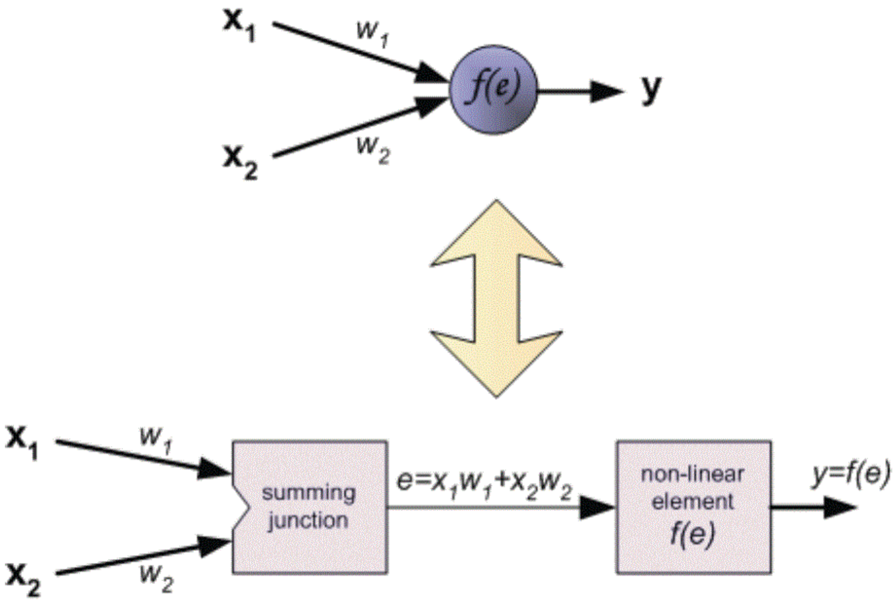
2. **Why do we need Deep Learning?**

3. **Learning Representation for NLP**

4. **Tools for Word Embedding**
   - **Word2Vector**
   - **Ranking-based**

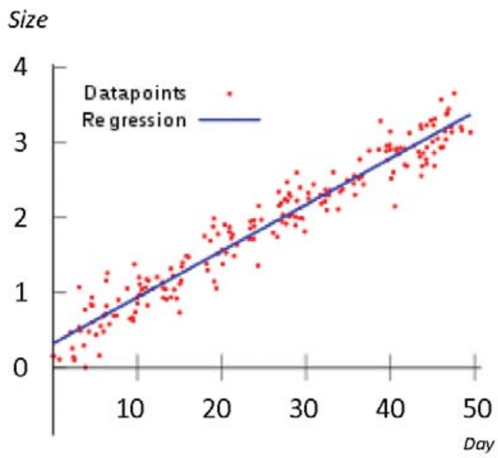
Basic Concepts of NN

❖ **Perceptron**



Basic Concepts of NN

❖ **Illustration Example (Apple Tree)**

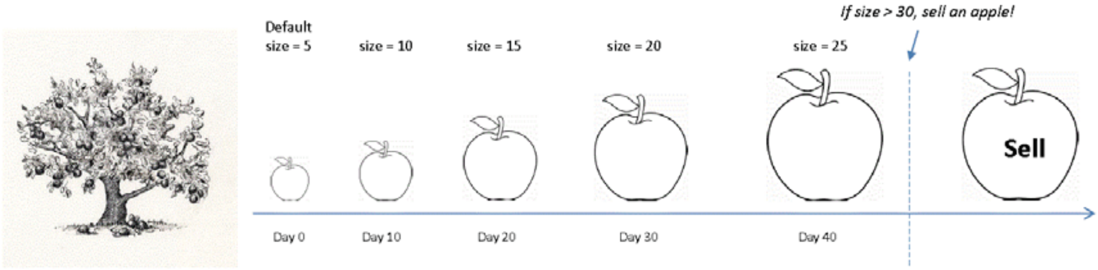


- "어떤 사과나무에 대해서 몇 년에 걸쳐 날짜 별로 사과들의 크기를 측정, 기록"
- 농부는 **특정 크기**가 넘을 때만 시장에 사과를 내다 팔 수 있다고 할 때,

- ***Q** : 올해 Day -50 에 사과를 내다 팔 수 있을까? 없을까?*

Basic Concepts of NN

❖ Illustration Example (Apple Tree)

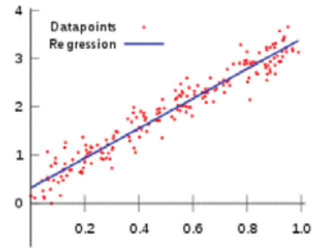


상황 1 : 작년까지 이 사과나무는 위의 경향대로 사과 열매를 맺었다.
조건 : 사과의 크기가 30이 넘으면 팔 수 있다.

Question : 올해 Day-50 에 사과를 팔 수 있을까?

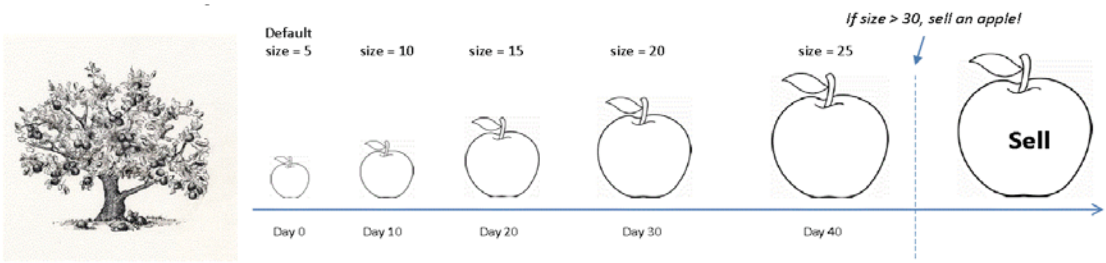
↓

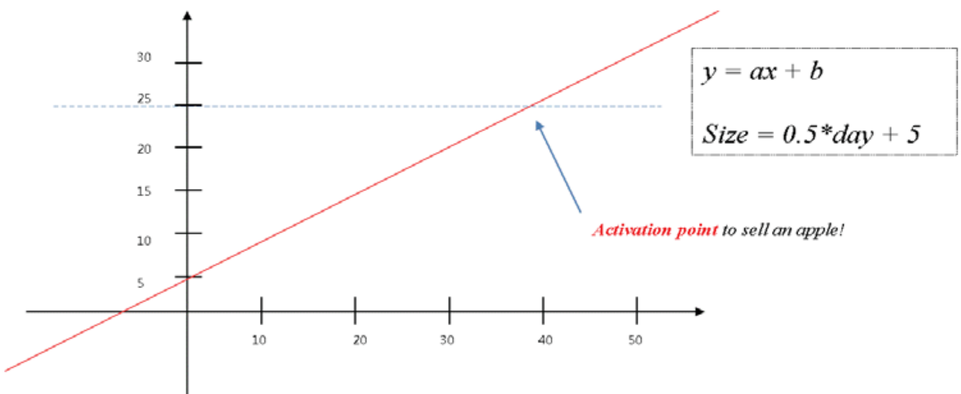
Very Typical ***Regression Problem***



Basic Concepts of NN

❖ Illustration Example (Apple Tree)

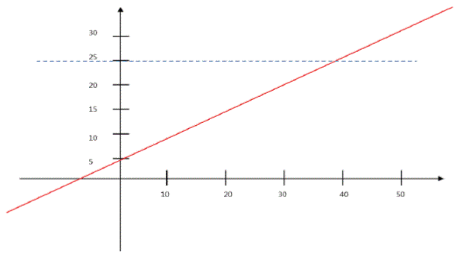


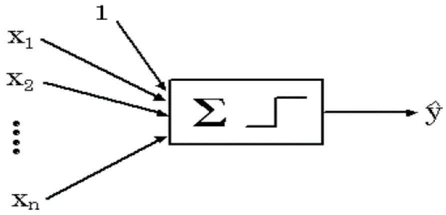


Regression
→ learn the parameter ***'a'*** and ***'b'*** from the data

Basic Concepts of NN

❖ Illustration Example (Apple Tree)

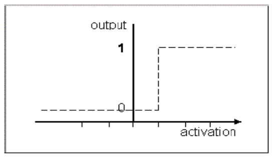




$y = ax + b$ → 입력값을 변형해 새로운 값 계산 (day → size) → $Y = WX + b$

정규화

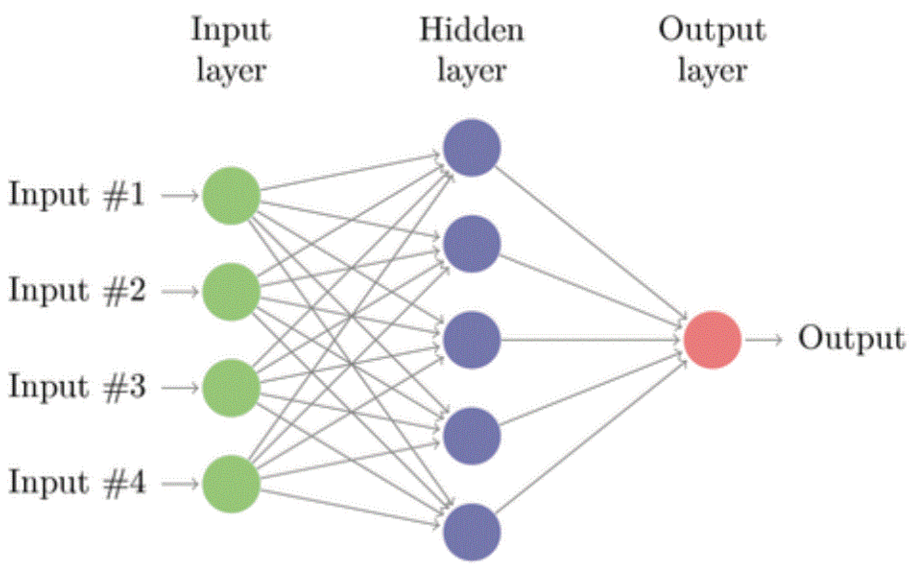
If y > 30 → *sell an apple* → 새로운 값을 다시 해석해 최종 결과 산출 (size → 팔까(1)/말까(0)) → *Activation function*



Step Function

Basic Concepts of NN

❖ Multilayer Neural Network

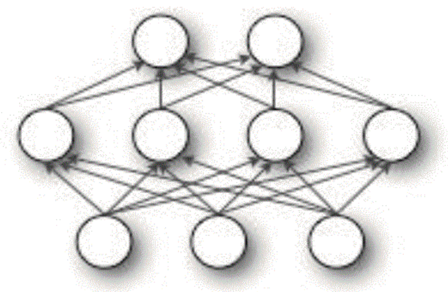


Basic Concepts of NN

❖ Multilayer Neural Network

The single-hidden layer Multi-Layer Perceptron (MLP).

An ***MLP*** can be viewed as a ***logistic regressor***, where the input is first transformed using a learnt ***non-linear transformation***



[*Softmax* Function]

G : Scoring Function for top-layer

[*tanh* Function]

S : Activation Function for hidden layer

output layer $o(x) = G(b^{(2)} + W^{(2)} h(x))$

hidden layer $\boxed{h(x)} = \Phi(x) = s(b^{(1)} + W^{(1)} x)$

input layer $\boldsymbol{x}$

$$f : R^D \rightarrow R^L$$

$$f(x) = G(b^{(2)} + W^{(2)}(s(b^{(1)} + W^{(1)}x))),$$

$\boldsymbol{D}$ is the size of input vector $\boldsymbol{x}$

$\boldsymbol{L}$ is the size of output vector $\boldsymbol{f(x)}$

Feed Forward Propagation

Basic Concepts of NN

❖ Training (Weight Optimization)

$$\theta = \{W^{(2)}, b^{(2)}, W^{(1)}, b^{(1)}\}$$

- How to learn the weights??

“Backpropagation Algorithm”

최종 결과물을 얻고	**Feed Forward and Prediction**
그 결과물과 우리가 원하는 결과물과의 차이점을 찾은 후	**Cost Function**
그 차이가 무엇으로 인해 생기는 지	**Differentiation (미분)**
역으로 내려가면서 추정하여	**Back Propagation**
새로운 Parameter 값을 배움	**Weight Update**

Basic Concepts of NN

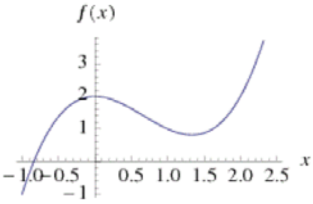
❖ Training (Weight Optimization)

Backpropagation = Backpropagation of errors

↓

Gradient descent procedures are generally used where we want to maximize or minimize n-dimensional functions.

The ***gradient*** is a vector ***g*** that is defined for any ***differentiable*** point of a function, that points from this point exactly towards the ***steepest ascent*** and indicates the gradient in this direction by means of its norm **|*g*|**.



$$f(x) = x^3 - 2x^2 + 2$$

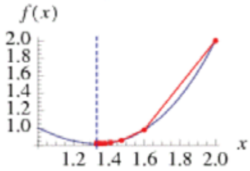
$$x_i = x_{i-1} - \epsilon f'(x_{i-1})$$

x_i 가 변화가 없을 때까지 위 수식을 반복

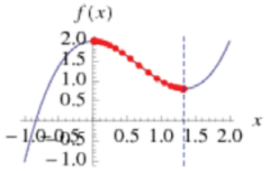
➔결국 gradient 가 가리키는 방향으로 계속해서 parameter 변화 됨

➔Local Minimum 에 빠질 수 있음

$x_0 = 2$



$x_0 = 0.01$



Basic Concepts of NN

❖ Training (Activation Functions)

$sigmoid(a) = 1/(1 + e^{-a})$ *... also called '**logistic function**', 'Fermi function'*

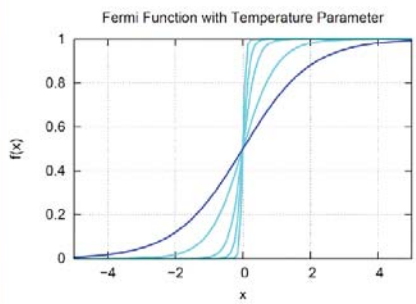
$$f(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{d}{dx} f(x) = f(x)(1 - f(x))$$

$$1 - f(x) = f(-x).$$

$$2\,f(x) = 1 + \tanh\left(\frac{x}{2}\right).$$

Always positive



$tanh(a) = (e^a - e^{-a})/(e^a + e^{-a})$

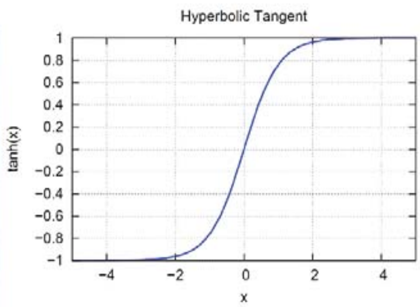
$$f(x) = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

$$e^x = \cosh x + \sinh x$$

and

$$e^{-x} = \cosh x - \sinh x$$

Output = [-1, 1]
Faster Backpropagation



Basic Concepts of NN

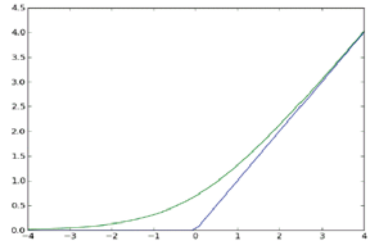
❖ Training (Activation Functions)

Rectified Linear Unit $f(x) = \max(0, x)$

Smooth approximation – "*softplus*" function

$$f(x) = \log(1 + e^x)$$

$$f'(x) = e^x/(e^x + 1) = 1/(1 + e^{-x})$$



❖ Scoring Functions (Softmax)

$$softmax_{ij}(x) = \frac{\exp x_{ij}}{\sum_k \exp(x_{ik})}$$

$$P(Y = i|x, W, b) = softmax_i(Wx + b) = \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}}$$

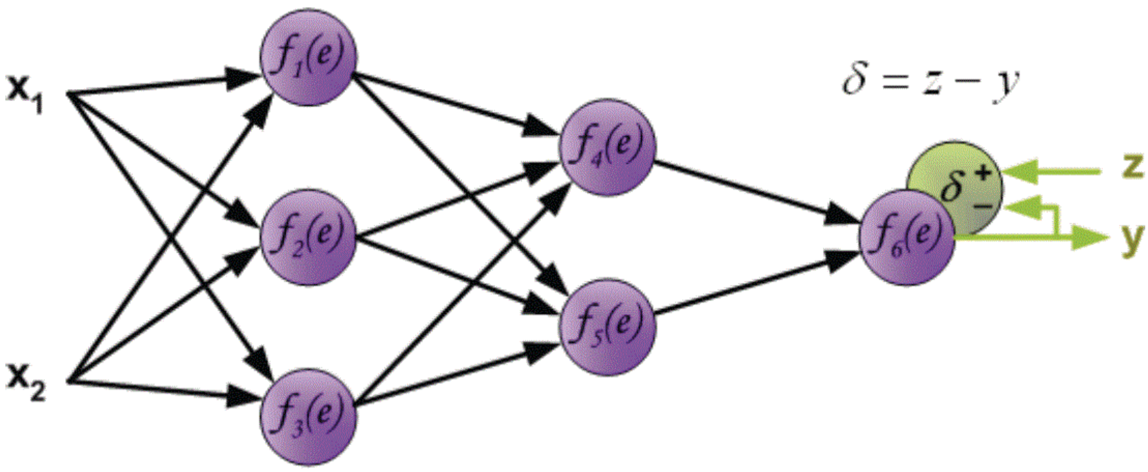
관심 class y

모든 class y

Basic Concepts of NN

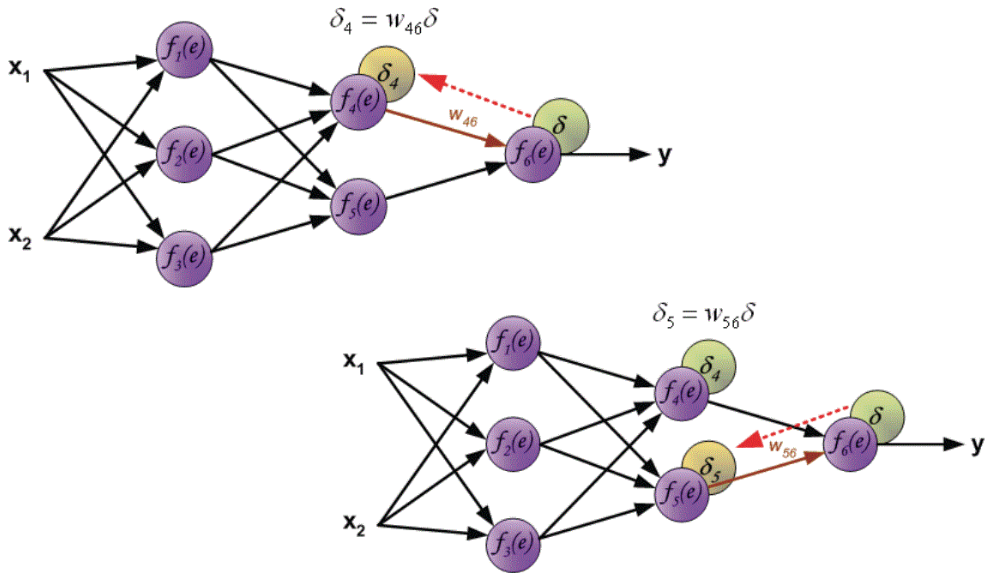
❖ Learning: Backpropagation

- Calculate error at the output
- Back-propagation = gradient descent + chain rule



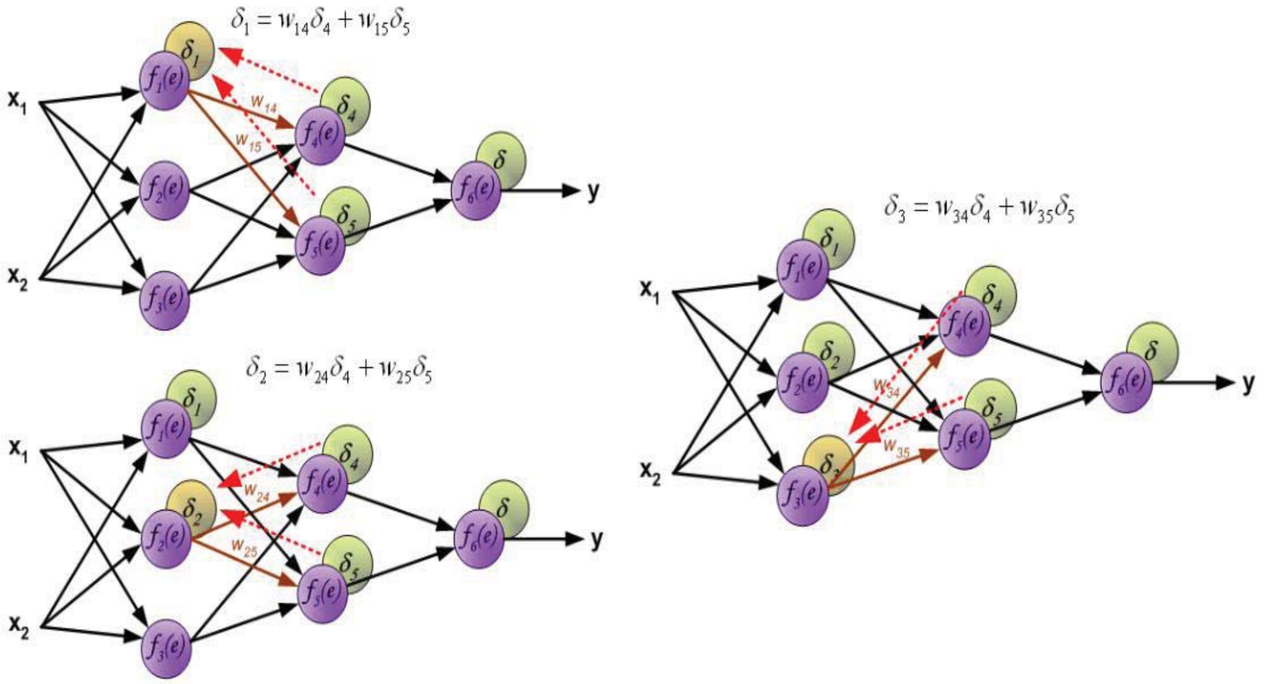
Basic Concepts of NN

❖ **Learning: Backpropagation**



Basic Concepts of NN

❖ **Learning: Backpropagation**

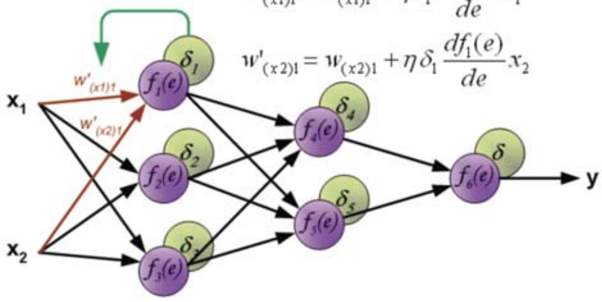


Basic Concepts of NN

- **Learning: Backpropagation**
  - Calculate error at the output

$$w'_{(x1)1} = w_{(x1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1$$

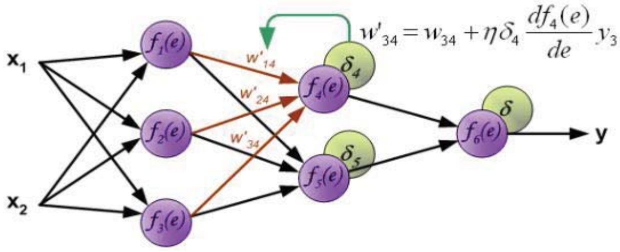
$$w'_{(x2)1} = w_{(x2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2$$



$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2$$

$$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$$

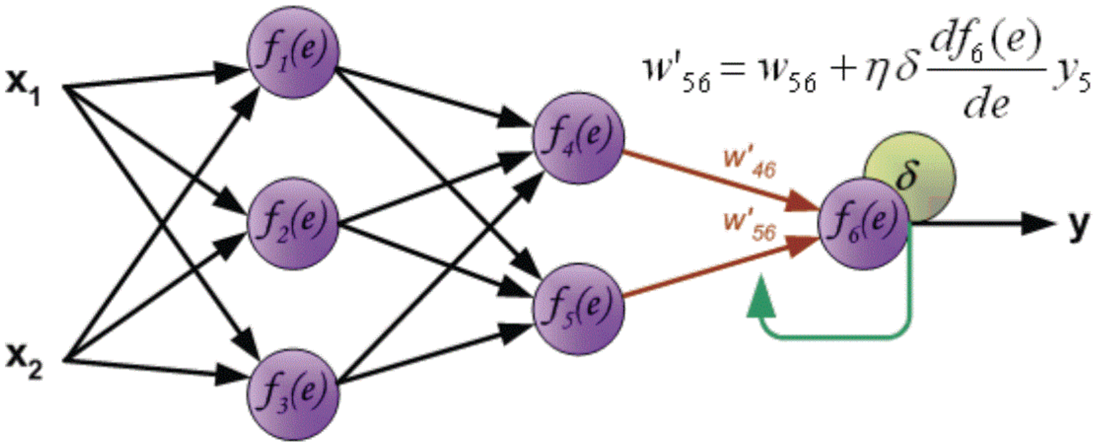


Basic Concepts of NN

- **Learning: Backpropagation**
  - Calculate error at the output

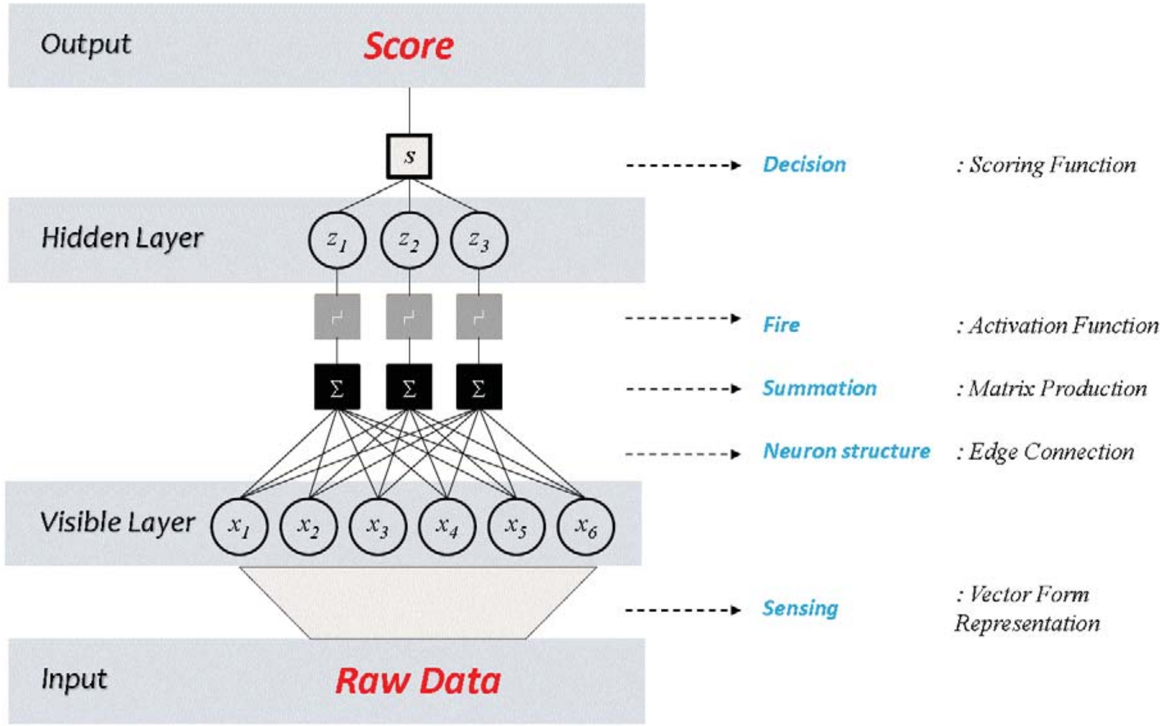
$$w'_{46} = w_{46} + \eta\delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta\delta \frac{df_6(e)}{de} y_5$$



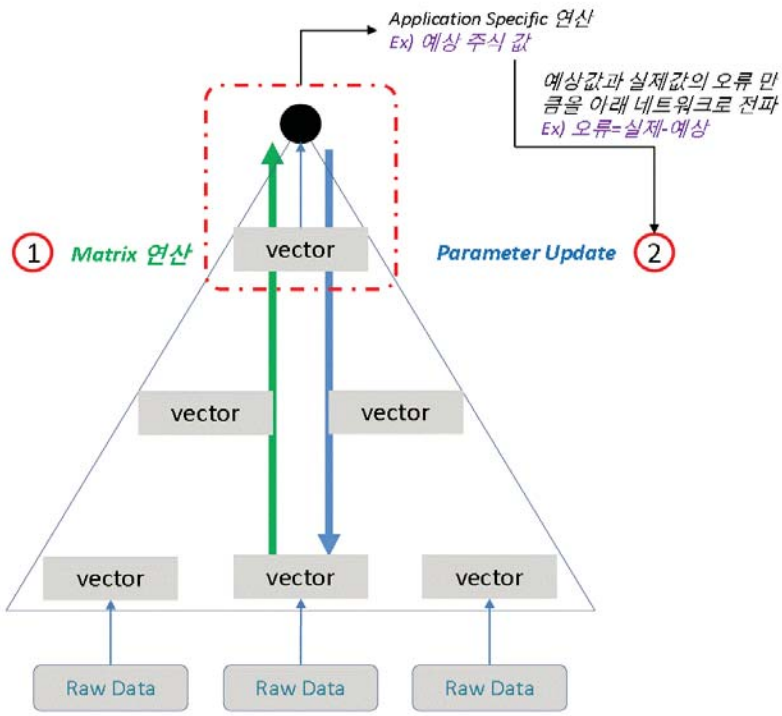
Basic Concepts of NN

❖ **Neural Network-Core Components**



Basic Concepts of NN

❖ **Neural Network-Process**



Why? Deep Learning

❖ **Why was not old NN successful?**

Initialization	Local Minima	Computation Power	Data

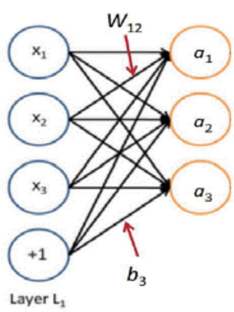
Pre-Training	*Distributed Representation*	*Initialization Techniques*
Activation Function	*Understanding ANN*	*Big Data*
	...	

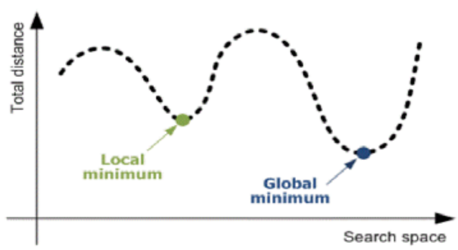
Deep Learning

Why? Deep Learning

❖ **Neural Network-Process**

- 네트워크가 깊어지고, 복잡해질수록 parameter 수가 많아짐
- Parameter가 많아질수록 Local Minima에 빠질 가능성이 높아짐





$$\boldsymbol{W} = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} \qquad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

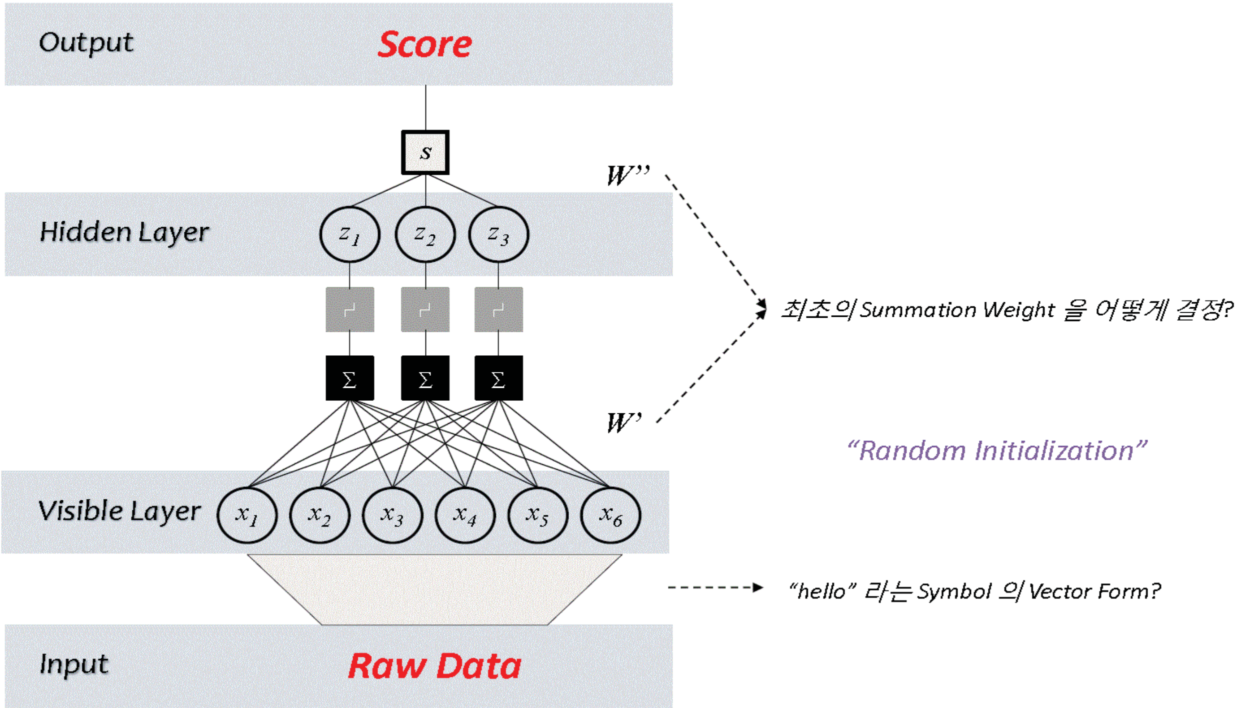
$$a_1 = f(W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + b_1)$$
$$a_2 = f(W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + b_2)$$
$$a_3 = f(W_{31}x_1 + W_{32}x_2 + W_{33}x_3 + b_3)$$

In Matrix notation

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

Why? Deep Learning

❖ Initialization Problem



Why? Deep Learning

❖ Initialization Tip

Initial Value

초기값 Settings

- Random 하게 주되 특정 구역안에서 Random 하게 주는것이 좋다.

tanh 를 Activation 으로 사용하는 경우

$$\text{Interval} = \left[-\sqrt{\frac{6}{fan_{in}+fan_{out}}}, \sqrt{\frac{6}{fan_{in}+fan_{out}}}\right]$$

fan_{in} = the number of units in the *(i-1)*th layer.
fan_{out} = the number of units in the *i*th layer

sigmoid 를 Activation 으로 사용하는 경우

$$\text{Interval} = \left[-4\sqrt{\frac{6}{fan_{in}+fan_{out}}}, 4\sqrt{\frac{6}{fan_{in}+fan_{out}}}\right]$$

Why? Deep Learning

- **Deeper Network, Harder Learning**
  - Network가 깊으면 깊을 수록 최종성능이 좋다. 단, 깊어질수록 Error Propagation이 어려워짐. ReLU가 사용되는 이유

Why? Deep Learning

- **Pre-Training**
  - Pre-training으로 NN의 성능이 비약적으로 향상됨
  - AutoEncoder 계열과 Restricted Boltzmann Machine 계열이 있음

Unsupervised Learning

Large Raw Data

↓

$P(x)$ - - - - → *"Pretraining"*

↓

Small Tagged Data

↓

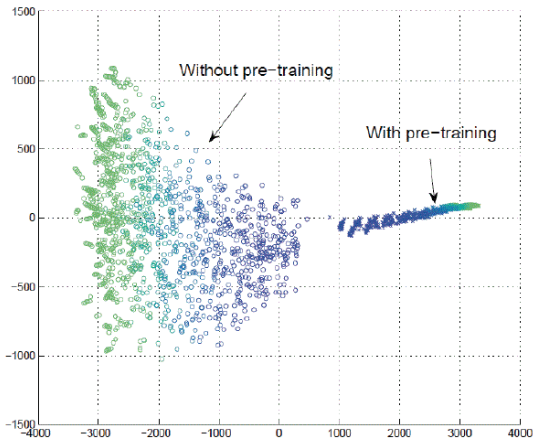
$P(y \mid x)$

Supervised Learning

Why? Deep Learning

❖ Pre-Training-Performance

- Regularization hypothesis:
  - Representations good for P(x) are good for P(y|x)
- Optimization hypothesis:
  - Unsupervised initializations start near better local minimum of supervised training error
  - Minima otherwise not achievable by random initialization

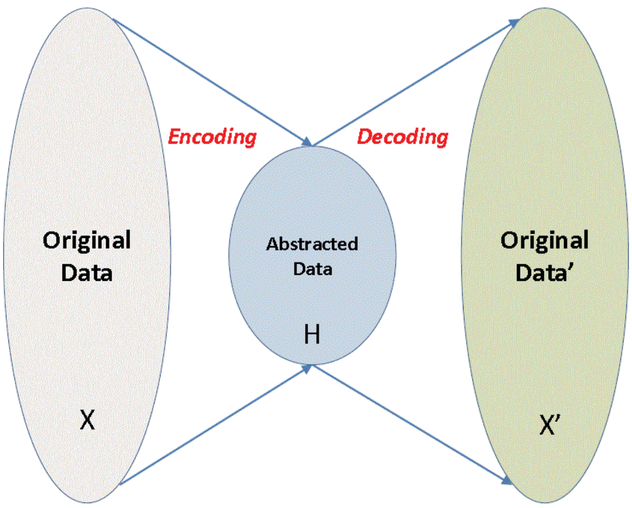


Erhan, Courville, Manzagol, Vincent, Bengio (JMLR, 2010)

97

Why? Deep Learning

❖ Auto Encoder



원래의 데이터 X 를 H 로 프로젝션 시킨 후, H 로 부터 X' 를 다시 생성시킴

- 비교
  - 압축 알고리즘 (Zip, MPEG, PNG …)
  - Principle Component Analysis (PCA)
  - Kernel Function in SVM (original space → hyper space)

X → H → X' 에서 Difference(X, X') 가 적으면 적을수록 추상화는 완벽하게 이루어진 것이라 생각할 수 있다.
그러한 Projection 이 완벽하게 훈련된다면,

- Abstracted Data 는 그 자체로 원래 데이터를 설명하는 Feature라고 볼 수 있을 것이다.
- Feature Learning 이 자동으로 이루어지는 것이라 할 수 있음

Why? Deep Learning

❖ **Auto Encoder**

Input

Hidden

Prediction

$$\mathbf{x} \in [0,1]^d \longrightarrow \boxed{\mathbf{y} = s(\mathbf{W}\mathbf{x} + \mathbf{b})} \longrightarrow \mathbf{y} \in [0,1]^{d'} \longrightarrow \underset{\textit{reconstruction}}{\boxed{\mathbf{z} = s(\mathbf{W}'\mathbf{y} + \mathbf{b}')}} \longrightarrow \mathbf{z}$$

(predication of x) – x'

Encoding/Decoding Error

$$L(\mathbf{x}, \mathbf{z}) = ||\mathbf{x} - \mathbf{z}||^2$$ … real value case

$$L_H(\mathbf{x}, \mathbf{z}) = -\sum_{k=1}^{d} [\mathbf{x}_k \log \mathbf{z}_k + (1 - \mathbf{x}_k) \log(1 - \mathbf{z}_k)]$$ … bit vector or … vectors of bit probability

Cross-entropy

Learning Representation for NLP

No more handcraft feature engineering!

$$\left[\begin{array}{l} \text{▪ color = 'red'} \\ \text{▪ shape = 'round'} \\ \text{▪ leafs = 'yes'} \\ \text{▪ dot = 'yes'} \\ \text{▪ …} \end{array}\right] \Longrightarrow \left[\begin{array}{c} \text{Numbers} \end{array}\right]$$

- 사과를 '사과'로 구별 짓는 표현방식을 스스로 학습

Learning Representation for NLP

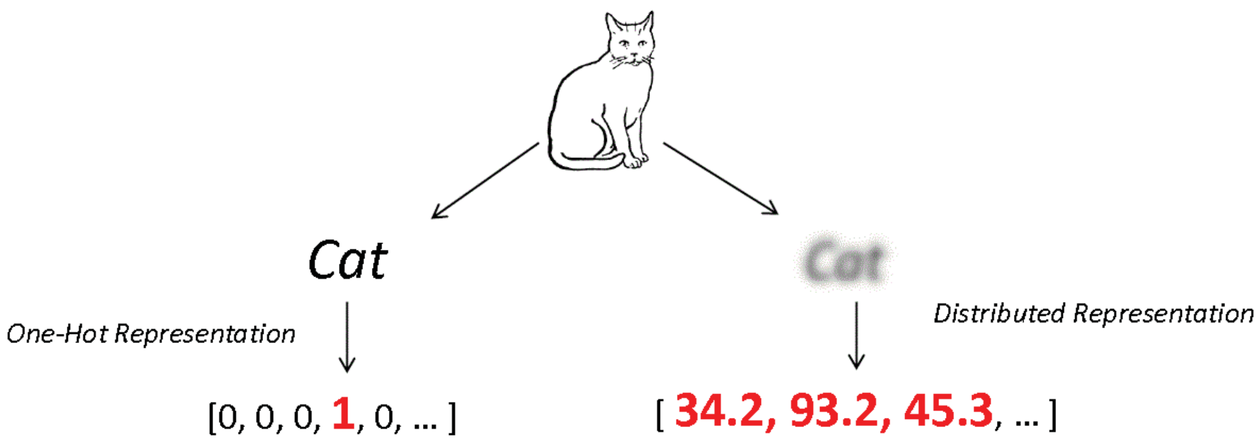
- **One-hot representation (or symbolic)**
  - Ex) [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
  - Dimensionality
    - 20K (speech) – 50K (PTB) – 500K (big vocab) – 3M (Google 1T)

- **Continuous representation**
  - Latent Semantic Analysis, Random projection
  - Latent Dirichlet Allocation, HMM clustering
  - Distributed Representation (Neural word embedding)
    - **Dense vector**
    - By **adding supervision** from other tasks **-> improve the representation**

Learning Representation for NLP

- **Distributed Representation**
  - DNN이 기존 AI 방법론들에 비해 큰 의미가 있는 것은 실 세계에 있는 실제 Object를 표현할 때 Symbol에 의존하지 않는다는 점이다.

[Representation]



Learning Representation for NLP

❖ **Distributed Representation**

- 유사한 것은 ‘유사하게’ 표현되어야 함
- Curse of Dimensionality 극복 가능

Apple = 001

Pear = 010

Ball = 100

Distance(Apple ~ Pear) = Distance(Apple ~ Ball)

Learning Representation for NLP

Local Representation	**Distributed Representation**
Only one neuron (or very few) is active	many features, each of which can separately each be active or inactive

Cat

[0,0,0,0,0,0,1,0,0,0,0,0]

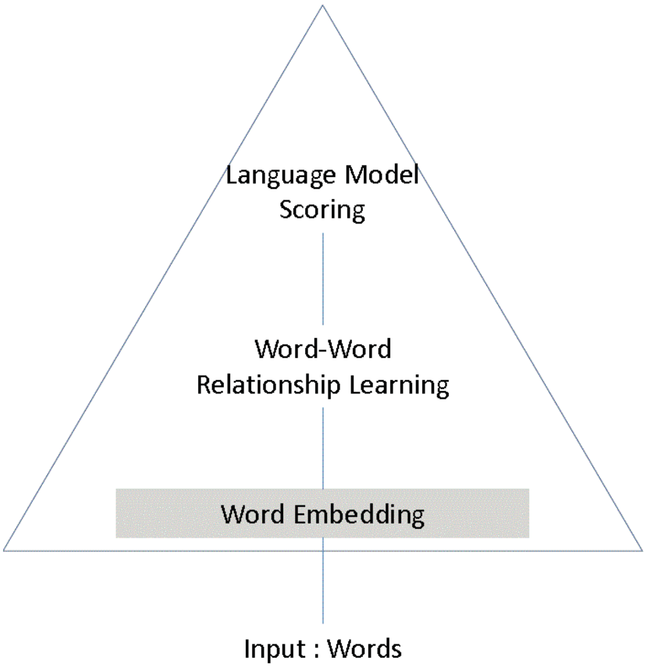
Cat $\begin{pmatrix} -2.3 \\ 1.0 \\ 4.2 \\ 5.3 \\ 2.3 \end{pmatrix}$

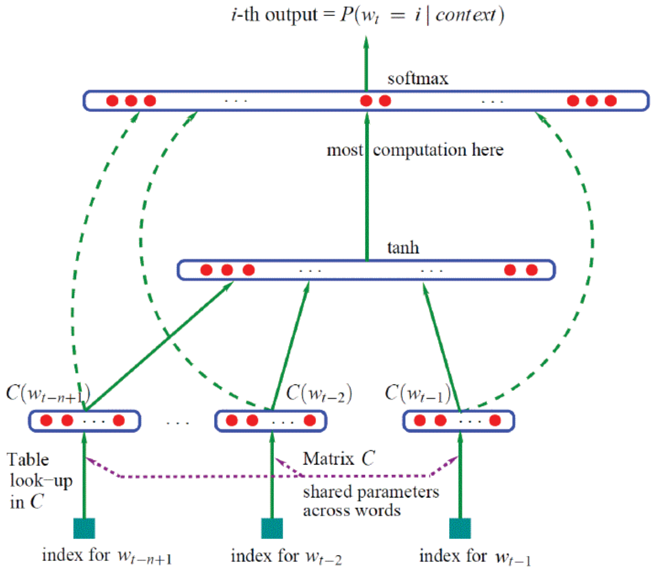
- *One-Hot Representation*
  - ***Integer Space***
  - *Very Sparse*
  - *Very high dimensionality*

- *Word embedding*
  - ***Real value space***
  - *Dense*
  - *Low Dimensionality*

Ex) word → hash to DB Access?
It means ‘integer’ space.

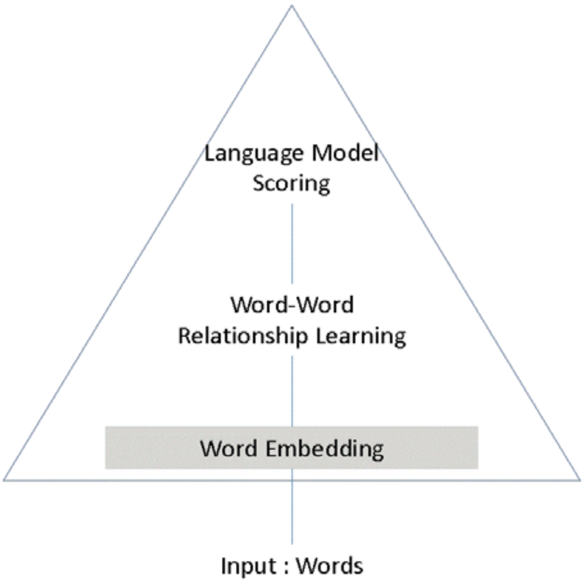
Learning Representation for NLP

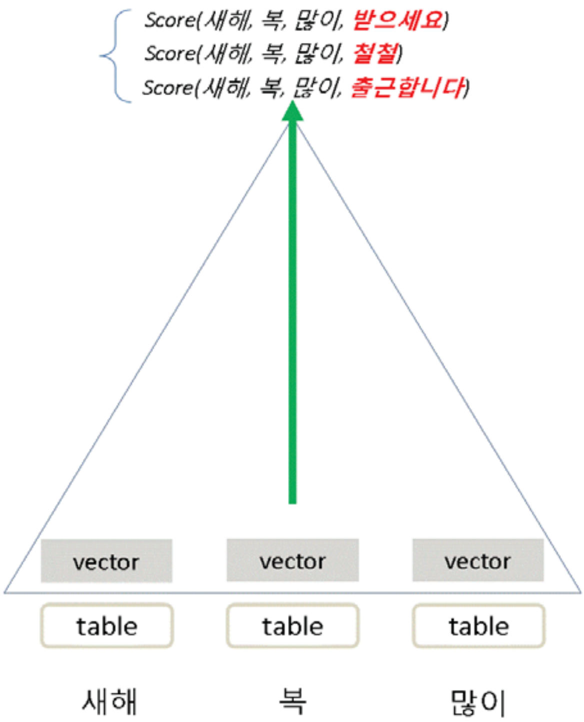




Learning Representation for NLP

Neural Language Model – Big Idea





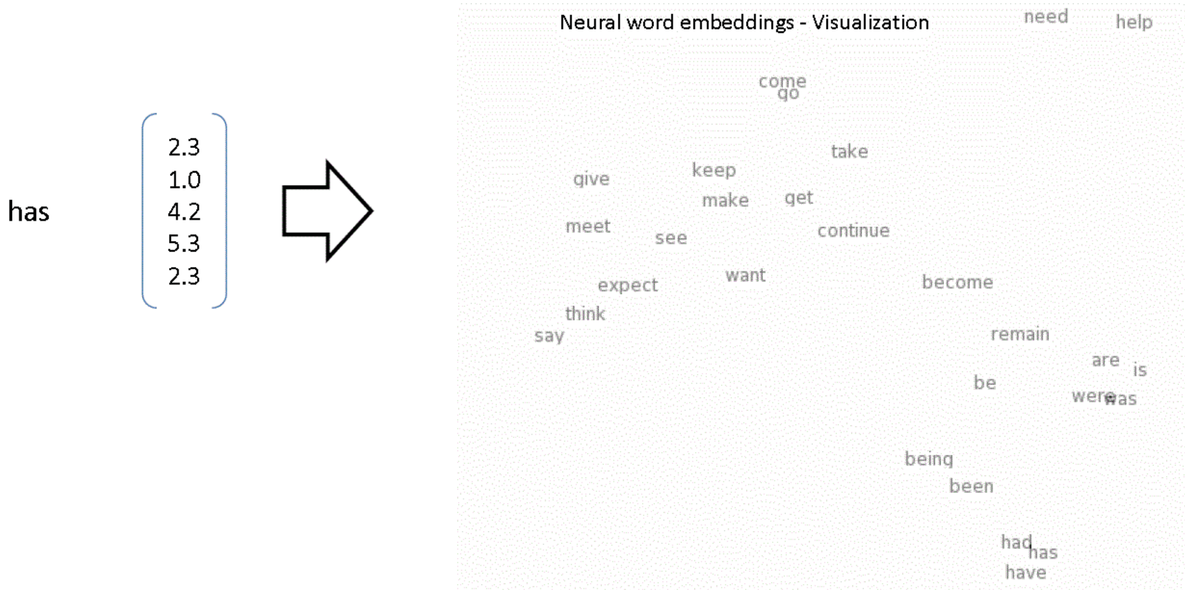
*Score(새해, 복, 많이, **받으세요**)* > *Score(새해, 복, 많이, **철철**)*

:: 실 언어데이터에 있는 패턴이 그렇지 않은 패턴보다 많이 나타나도록 NN 을 훈련시킨다.

Learning Representation for NLP

❖ Good One – Word Representation

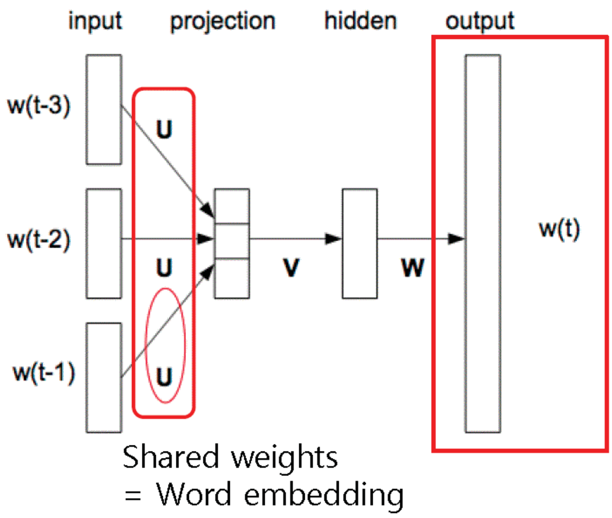
We can compare words without any extra knowledge such as word net!!!



Learning Representation for NLP

❖ Neural Network Language Model

- Idea
  - A **word and its context** is a **positive** training sample
  - A **random word in that sam e context → negative** trainin g sample
  - Score(positive) > Score(neg.)
- Training complexity is high
  - **Hidden layer → output**
  - **Softmax in the output layer**
    - **Hierarchical softmax**
    - **Negative sampling**
    - **Ranking(hinge loss)**



Input	Dim: 1	Dim: 2	Dim: 3	Dim: 4	Dim: 5
1 (boy)	0.01	0.2	-0.04	0.05	-0.3
2 (girl)	0.02	0.22	-0.05	0.04	-0.4

Learning Representation for NLP

❖ Back-Propagation Algorithm

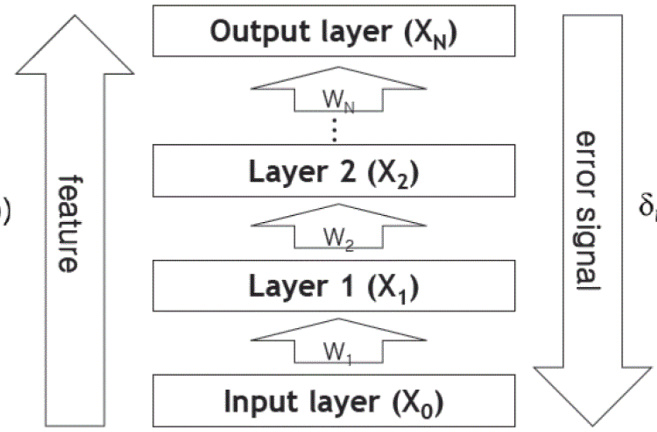
- Gradient descent algorithm w.r.t. error E.

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}}$$

$$\delta_j = \frac{\partial E}{\partial net_j}$$

$$x_j = f(\textstyle\sum_i w_{ij} x_i + \theta)$$



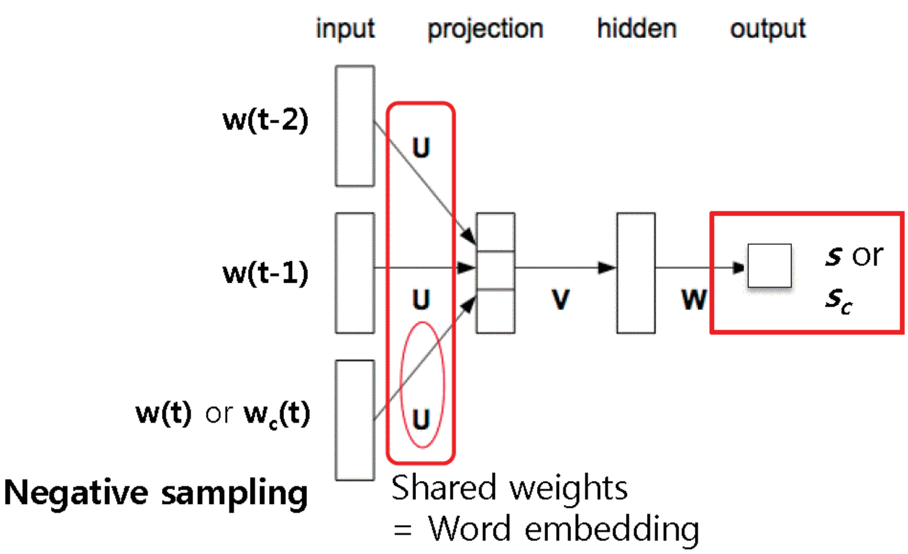
$$\delta_i = f'(net_j) \sum_j w_{ij} \delta_j$$

Learning Representation for NLP

❖ Ranking-based (Collobert)

Ranking(hinge loss)[2,11]: $\max(0,\ 1 - s + s_c)$

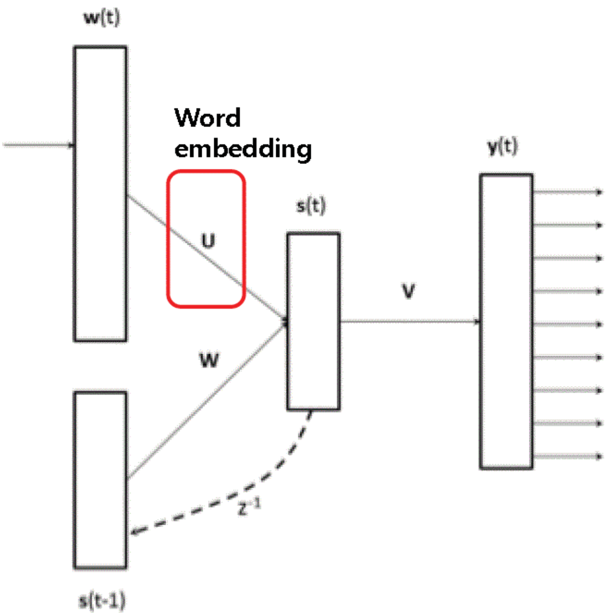
$\boxed{\text{Ranking(logit loss): } \log(1 + \exp(s_c - s))}$ → 본 연구 추가



Learning Representation for NLP

❖ **Recurrent Neural Network**

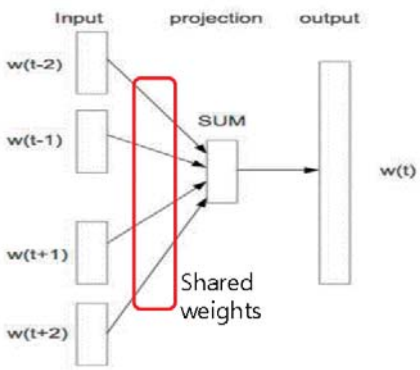
- RNN
  - The **hidden layer** s(t) maintains a represen tation of the **senten ce history**

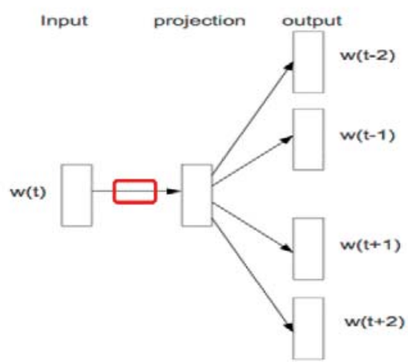


Learning Representation for NLP

❖ **Word2Vec: CBOW, Skip-Gram**

- **Remove the hidden layer → Speedup 1000x**
  - Negative sampling
  - Frequent word sampling
  - Multi-thread (no lock)
- **Continuous Bag-of-words (CBOW)**
  - Predicts the current word given the con text
- **Skip-gram**
  - Predicts the surrounding words given the current word
  - CBOW + DropOut/DropConnect

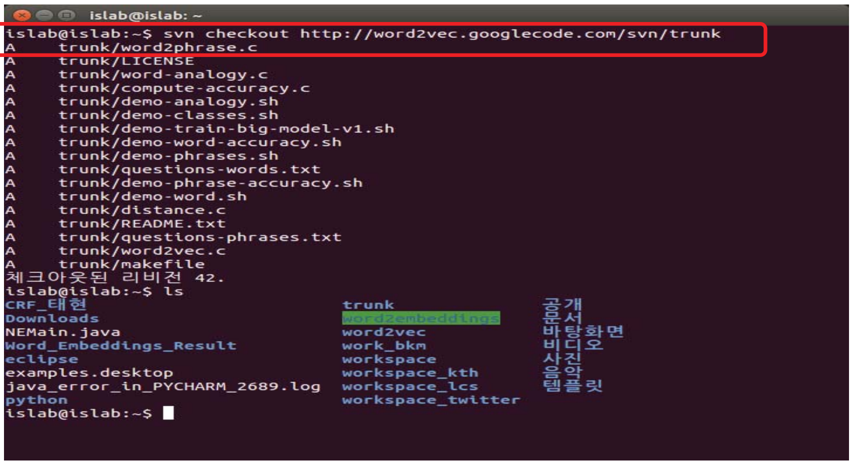




Tools for Word Embedding

❖ **Word2Vec**

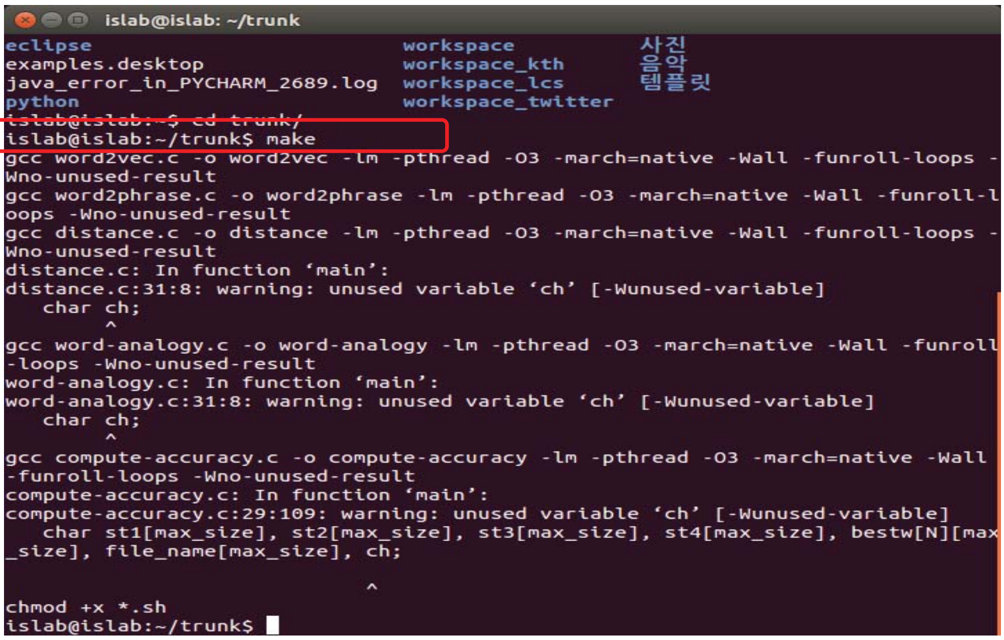
- https://code.google.com/p/word2vec/
- http://deeplearning4j.org/word2vec.html#just
- Ubutu 버전 (JAVA)
  - Googlecode에서 파일 다운로드
  - svn checkout http://*word2vec.googlecode.com/svn/trunk*
  - trunk 폴더에 파일 다운로드 됨



Tools for Word Embedding

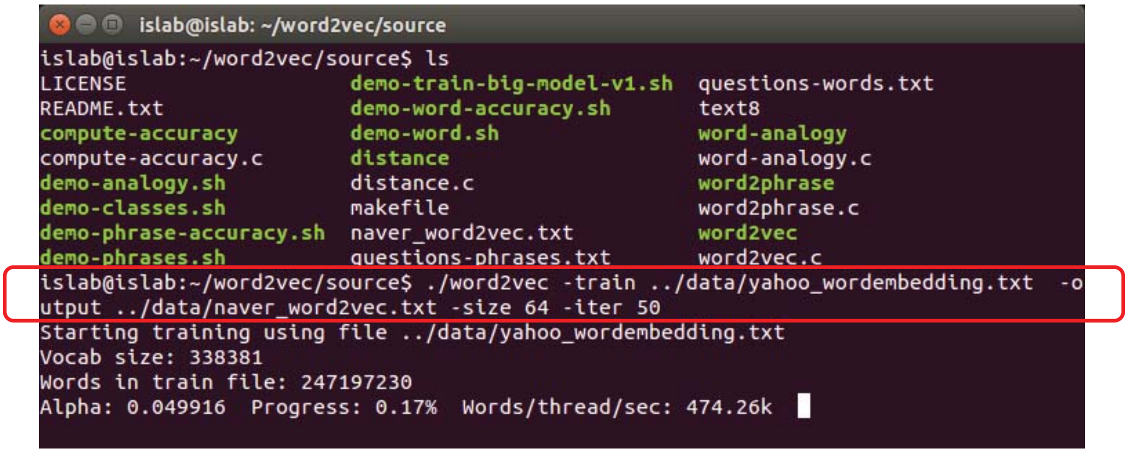
❖ **Word2Vec 설치**

- Make 명령어 실행
  - Warning은 무시



Tools for Word Embedding

❖ Word2Vec 실행



Tools for Word Embedding

❖ Word2Vec parameters

- -output
  - 출력파일
- -size
  - 생성할 word vector의 차원 (default value: 100)
- -windows
  - Max skip length (default value: 5)
- -cbow
  - 1: continuous bag of word model, 0: skip-gram model
- -iter
  - 반복 횟수 (default value: 5)
- -min-count
  - 지정 횟수보다 작게 나온 단어 제거 (default value: 5)
- -save-vocab
  - 단어 목록 출력 파일
- -read-vocab
  - 단어목록을 미리 지정

Tools for Word Embedding

- **Word2Vec 학습파일 포맷**
  - -train
  - 한 문장 별로 한 라인에 문장 자질로 구성

```
지금/MAG 도/JX 경기도/NNP 나/NP 강원도/NNP 산세/NNG 좋/VA 은/ETM 곳/NNG 에/JKB 가면/NNG 산/NNG 과/JC 숲/NNG 을/JKO 온통/MAG 파헤치/VV 는/ETM 골프장/NN
멀쩡/XR 하/XSA ㄴ/ETM 산/NNG 을/JKO 헐/VV 어/EC 내/VX 고/EC 언덕/NNG 을/JKO 만드/NA 려니/EC 얼마나/MAG 힘/NNG 이/JKS 들/VV 겠/EP 는/ETM 가/NNG ./SF
게다가/MAG 그/MM 위/NNB 에/JKB 잔디/NNG 까지/JX 입히/VV 려니/EC 문제/NNG 가/JKS 한/NNP 둘이/NNG 아니/VCN 다/EF ./SF
원래/NNG 우리나라/NNG 기후/NNG 에/JKB 는/JX 잔디/NNG 가/JKS 잘/MAG 맞/VV 지/EC 않/VX 는다/EF ./SF
야산/NNG 에/JKB 잡목/NNG 과/JC 덤불/NNG 숲/NNG 만/JX 우거졌/NA 지/VX 초원/NNG 이/JKS 생겨나/VV 지/EC 않/VX 은/ETM 것/NNB 도/JX 기/NNG 후/NNG 때문/NNB
```

  - Tutorial
    - http://alexminnaar.com/word2vec-tutorial-part-ii-the-continuous-bag-of-words-model.html

Tools for Word Embedding

- **Ranking-based Model**
  - https://bitbucket.org/aboSamoor/word2embeddings
  - Python, Theano
  - Result file (pickle)
    - *a = np.load(sys.argv[2])*

References

- Ronan Collbert, et al. “Natural Language Processing (Almost) from Scratch,” *Journal of Machine Learning Research*, 2011.
- Mikolov, T., et al. “Recurrent Neural Network based Language Model,” 2010.
- Mikolov, T., et al., “Distributed Representations of Words and Phrases and their Compositionality,” *NIPS*, 2013.
- Le, Q., Mikolov, T., “Distributed Representations of Sentences and Documents,” *ICML*, 2014.
- Kalchbrenner, N., Grefenstette, E. and Blunsom, P. “A Convolutional Neural Network for Modelling Sentences,” *ACL*, 2014.
- Kim, Y. “Convolutional Neural Networks for Sentence Classification,” *EMNLP*, 2014.
- Al-Rfou, R., Perozzi, B., Skiena, S., “Polyglot: Distributed Word Representations for Multiligual NLP,” ACL, 2013.
- 정상근, “Introduction to Deep Learning,” *자연언어처리 튜토리얼*, 2015.
- 이창기, “Word and Phrase Embedding,” *자연언어처리 튜토리얼*, 2015.

Thank you for your attention!

http://web.donga.ac.kr/yjko/

고 영 중