Word Embedding

Ko, Youngjoong

Sungkyunkwan University

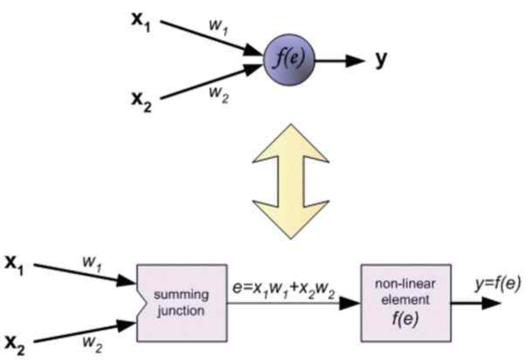
1

Contents

2

Basic Concepts of NN

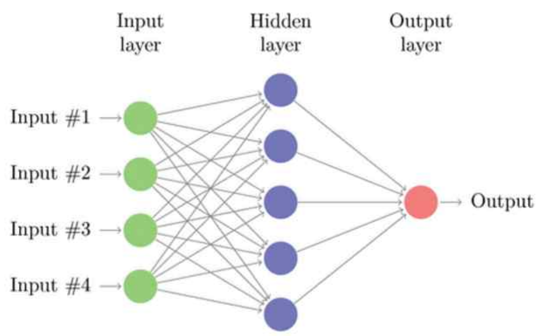
❖ **Perceptron**



3

Basic Concepts of NN

❖ **Multilayer Neural Network**



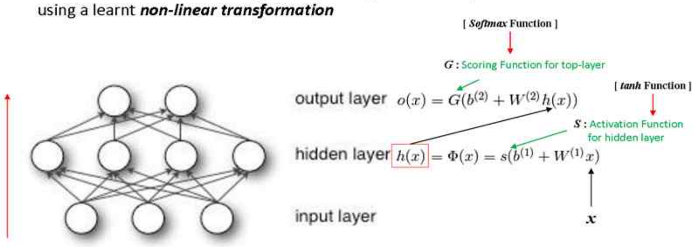
4

Basic Concepts of NN

❖ Multilayer Neural Network (Jeong, 2015)

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***



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

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

D is the size of input vector x

L is the size of output vector $f(x)$

Feed Forward Propagation

5

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**

6

Basic Concepts of NN

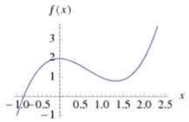
❖ Training (Weight Optimization)

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*|**.



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

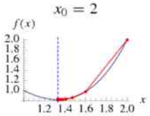
$x_i = x_{i-1} - \epsilon f'(x_{i-1})$

x_i 가 변화가 없을 때까지 위 수식을 반복

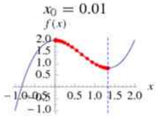
➔결국 gradient 가 가리키는 방향으로 계속해서 parameter 변화 됨

➔Local Minimum 에 빠질 수 있음

$x_0 = 2$



$x_0 = 0.01$



7

Basic Concepts of NN

❖ Training (Activation Functions)

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

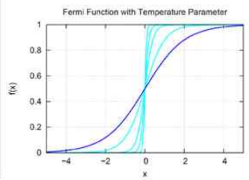
$$f(x) = \frac{1}{1+e^{-x}}$$

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

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

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

Always positive



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

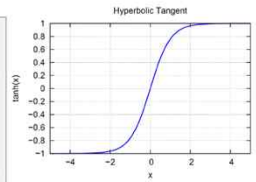
$$f(x) = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

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

and

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

Output = [-1, 1]
Faster Backpropagation



8

Basic Concepts of NN

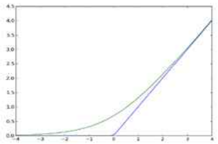
❖ **Training (Activation Functions)**

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)**

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

$$P(Y = i|x, W, b) = softmax_i(Wx + b) \quad \text{관심 class y}$$

$$= \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}} \quad \text{모든 class y}$$

9

9

Why? Deep Learning

❖ **Why was not old NN successful? (Jeong, 2015)**

Initialization | Local Minima | Computation Power | Data

Pre-Training | *Distributed Representation* | *Initialization Techniques*

Activation Function | *Understanding ANN* | *Big Data*

...

Deep Learning

10

10

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"*

Supervised Learning

Small Tagged Data

P(y | x)

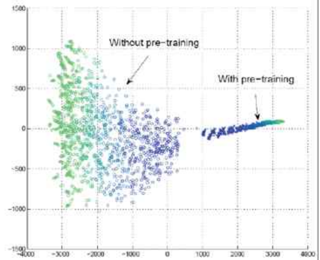
11

11

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)

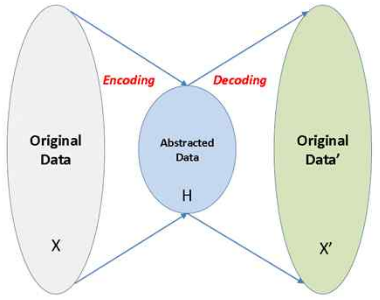
97

12

12

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

13

Learning Word Representation for NLP

No more handcraft feature engineering!

- color = 'red'
- shape = 'round'
- leafs = 'yes'
- dot = 'yes'
- …

Numbers

- 사과를 '사과'로 구별 짓는 표현방식을 스스로 학습

14

Learning Word Representation for NLP

❖ **The vast majority of rule-based and statistical NLP work regards words as atomic symbols**
- ➢ Walk, natural, language, process

❖ **In vector space terms, this is a vector with one (1) and a lot of zeroes (0)**
- ➢ [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)

❖ **"One-hot" representation**
- ➢ It is a localist representation

15

Learning Word Representation for NLP

❖ **For web search,**
- ➢ If user searches for "Seoul motel," we would like to match documents containing "Seoul hotel."

❖ **But**
- ➢ Inner product of motel [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0] and
  hotel [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] = 0
- ➢ Our query and document vectors are orthogonal
- ➢ No natural notion of similarity in a set of one-hot vectors

❖ **Could deal with similarity**
- ➢ Explore a direct approach where vectors encode it

16

Learning Word Representation for NLP

❖ **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**
  - **Get a lot of value by representing a word by means of its neighbors**
  - **It's one of the most successful ideas of modern statistical NLP**

government debt problems turning into	banking	crises as has happened in
saying that Europe needs unified	banking	regulation to replace the hodgepodge

17

17

Learning Word Representation for NLP

❖ **Distributed Representation (Jeong, 2015)**
- DNN이 기존 AI 방법론들에 비해 큰 의미가 있는 것은 실 세계에 있는 실제 Object를 표현할 때 Symbol에 의존하지 않는다는 점이다.

[Representation]

One-Hot Representation

Cat

[0, 0, 0, **1**, 0, ...]

Distributed Representation

Cat

[**34.2, 93.2, 45.3**, ...]

18

18

Learning Word 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)

19

19

Learning Word 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 [-2.3, 1.0, 4.2, 5.3, 2.3]

- *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.

20

20

Approaches for Word Embedding

- **Basic idea of learning neural network word embeddings**
  - Define a model that aims to predict between a center word W_t and context words in terms of word vectors
  - A loss (or cost) function, e.g.,

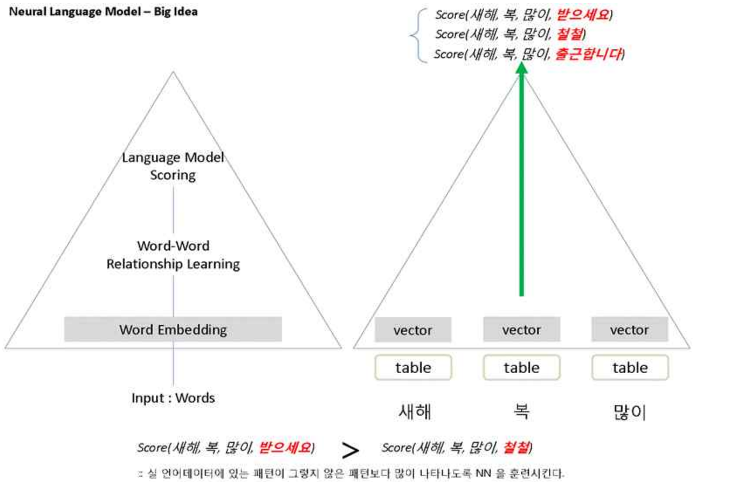
$$J = 1 - p(context \mid w_t)$$

  - Look at many positions *t* in a big language corpus
  - Keep adjusting the vector representations of words to minimize this loss (or cost)

Approaches for Word Embedding

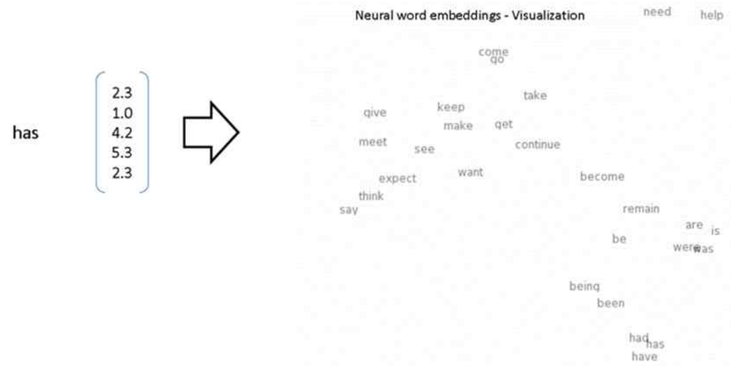
- **Predict between every word and its context words!**
- **Two algorithms**
  - **Skip-grams (SG)**
    - Predict context words given target
  - **Continuous Bag of Words (CBOW)**
    - Predict target word from bag-of-words context
- **Two training methods**
  - Negative sampling

Approaches for Word Embedding



Approaches for Word Embedding

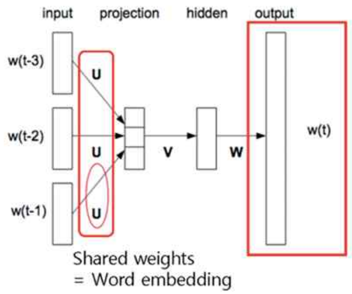
- **Good One – Word Representation**
  - We can compare words without any extra knowledge such as word net!



Approaches for Word Embedding

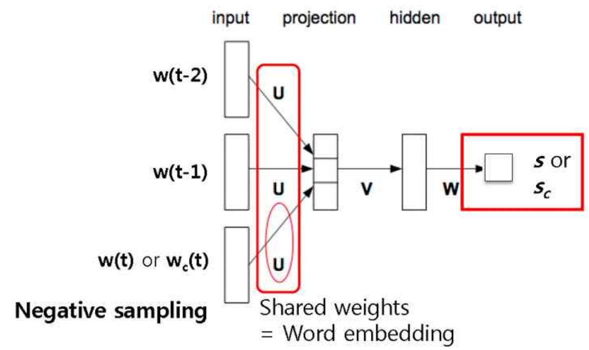
❖ **Neural Network Language Model (Lee, 2015)**

- Idea
  - A word and its context is a positive training sample
  - A random word in that same context —> negative training sample
  - Score(positive) > Score(neg.)
- Training complexity is high
  - Hidden layer —> output
  - Softmax in the output layer
    - ✓ Negative sampling



Approaches for Word Embedding

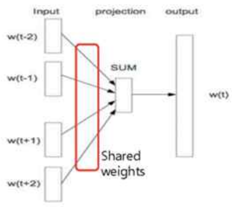
❖ **Ranking-based**

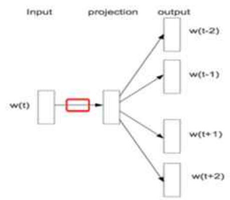


Approaches for Word Embedding

❖ **Word2Vec: CBOW, Skip-Gram**

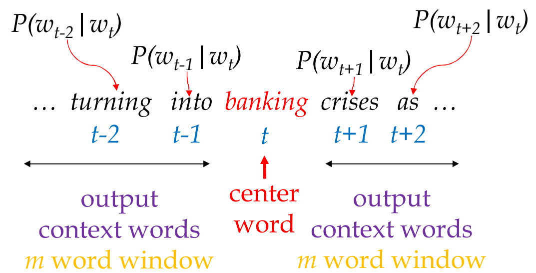
- Remove the hidden layer —> Speedup 1000x
  - Negative sampling
  - Frequent word sampling
  - Multi-thread (no loc)
- Continuous Bag-of-words (CBOW)
  - Predicts the current word given the context
- Skip-gram
  - Predicts the surrounding words given the current word
  - CBOW + DropOut / DropConnect





Approaches for Word Embedding

❖ **Skip-gram prediction**



Approaches for Word Embedding

❖ **Details of Word2vec (Manning, 2017)**

- For each word *t* =1 ... T, predict surrounding words in a window of "radius" m of every word.
- Objective function : Maximize the probability of any context word given the current center word:

$$J'(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P(w_{t+j} | w_t; \theta)$$

Negative Log Likelihood

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P(w_{t+j} \mid w_t)$$

- where θ represents all variables we will optimize

Approaches for Word Embedding

❖ **The objective function – details**

- Terminology : loss function = cost function = objective function
- Usual loss for probability distribution : Cross-entropy loss
- With one-hot w_{t+j} target, the only term left is the negative *log* probability of the true class

Approaches for Word Embedding

❖ **Cross Entropy Loss (Sung, 2017)**

- Linear model

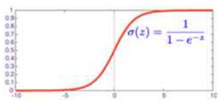
$x \to$ Linear $\to \hat{y}$

Hours (x)	Points
1	2
2	4
3	6
4	?

- Logistic Regression: pass/fail (0/1)

$x \to$ Linear $\to \hat{y}$ → $x \to$ Linear $\to$ Sigmoid $\to \hat{y}$

Hours (x)	Points	Pass/fail
1	2	0
2	4	0
3	6	1
4	?	?



Approaches for Word Embedding

❖ **Cross Entropy Loss (Sung, 2017)**

$x \to$ Linear $\to \hat{y}$

$$\hat{y} = x * w + b$$

$$loss = \frac{1}{N} \sum_{n=1}^{N} (\hat{y_n} - y_n)^2$$

$x \to$ Linear $\to$ Sigmoid $\to \hat{y}$

$$\sigma(z) = \frac{1}{1 - e^{-z}}$$

$$\hat{y} = \sigma(x * w + b)$$

$$loss = -\frac{1}{N} \sum_{n=1}^{N} y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n)$$

y	y_predict	loss
0	0.2	
0	0.8	
1	0.1	
1	0.9	

Approaches for Word Embedding

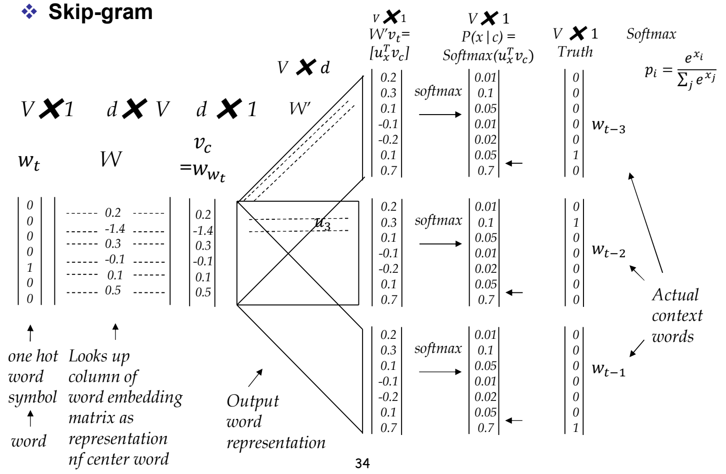
- **Details of Word2Vec**
  - Predict surrounding words in a window of radius *m* of every word
  - For $p(w_{t+j}|w_t)$ the simplest first formulation is

$$p(o\,/\,c) = \frac{\exp\,(u_o^T v_c)}{\sum_{w=1}^{V} \exp\,(u_w^T v_c)}$$

  - Where o is the outside (or output) word index, c is the center word index, v_c and u_o are "center" and "outside" vectors of indices c and o
  - Softmax using word c to obtain probability of word o

Approaches for Word Embedding

- **Skip-gram**



Approaches for Word Embedding

- **To train the model: Compute all vector gradients!**
  - We often define the set of all parameters in a model in terms of one long vector θ
  - In our case with *d*-dimensional vector and *V* many words:

$$\theta = \begin{pmatrix} v_a \\ . \\ . \\ . \\ v_{zebra} \\ u_a \\ . \\ . \\ . \\ u_{zebra} \end{pmatrix} \in R^{2dV}$$

  - We then optimize these parameters
  - Note: Every word has two vector. Makes it simpler.

Approaches for Word Embedding

- **Loss function:**

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m,\\ j\neq 0}} \log P(w_{t+j}|w_t)$$

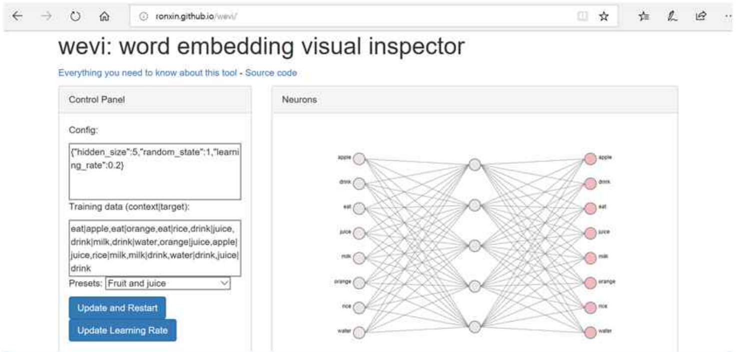
  - Let's derive gradient for center word together
  - For one example window and one example outside word:

$$\log p(o|c) = \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

  - You then also need the gradient for context words. That's all of the parameters θ here.

Approaches for Word Embedding

❖ **Simple Example of Word Embedding**



출처: http: ronxin.github.io/wevi/

37

37

Approaches for Word Embedding

❖ **Simple Example of Word Embedding**

➢ Negative Sampling

38

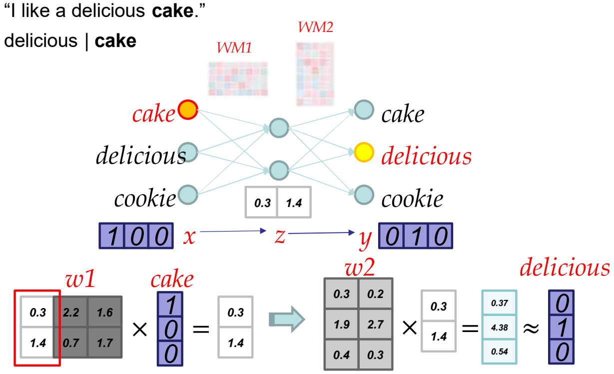
38

Approaches for Word Embedding

❖ **Simple Example of Word Embedding**

➢ "I like a delicious **cake**."

➢ delicious | **cake**



39

39

Approaches for Word Embedding

❖ **Calculating all gradients!**

➢ We went through gradient for each center vector v in a window

➢ We also need gradients for outside vectors u

➢ Generally, in each window, we will compute updates for all parameters that are being used in that window.

➢ For example, window size m = 1, sentence:

"We like learning a lot"

➢ First window computes gradients for:

- Internal vector v_{like} and external vectors u_{We} and $u_{learning}$

40

40

Approximations

- The normalization factor is too computationally expensive.

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

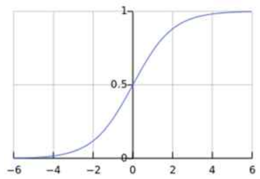
- Hence, you will implement the skip-gram model with **negative sampling**
- Main idea: train binary logistic regressions for a true pair (center word and word in its context window) versus a couple of noise pairs (the center word paired with a random word)

The skip-gram model and negative sampling

- From paper: "Distributed Representations of Words and Phrases and their Compositionality" (Mikolov et al. 2013)
- Overall objective function: $J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J_t(\theta)$

$$J_t(\theta) = \log \sigma(u_o^T v_c) + \sum_{j=1}^{k} \mathbb{E}_{j\sim P(w)}\left[\log \sigma(-u_j^T v_c)\right]$$

- Where *k* is the number of negative samples and we use,
- The sigmoid function! $\sigma(x) = \frac{1}{1+e^{-x}}$ (we'll become good friends soon)
- So we maximize the probability of two words co-occurring in first log



The skip-gram model and negative sampling

- Slightly clearer notation:

$$J_t(\theta) = \log \sigma(u_o^T v_c) + \sum_{j\sim P(w)} \left[\log \sigma(-u_j^T v_c)\right]$$

- Maximize probability that real outside word appears, minimize prob. that random words appear around center word
- $P(w)=U(w)^{3/4}/Z$, the unigram distribution U(w) raised to the 3/4 power (We provide this function in the starter code).
- The power makes less frequent words be sampled more often

Approaches for Word Embedding

- Why not capture cooccurrence counts directly? (Manning, 2017)
  - 2 options: full document *vs.* windows
  - Word-document co-occurrence matrix will give general topics (all sports terms will have similar entries) leading to "Latent Semantic Analysis"
  - Instead: Similar to word2vec, use window around each word —> captures both syntactic (POS) and semantic information

Approaches for Word Embedding

- **Example: Window based co-occurrence matrix**
  - Window length 1 (more common: 5 – 10)
  - Symmetric (irrelevant whether left or right context)
  - Example corpus:
    - I like deep learning.
    - I like NLP.
    - I enjoy flying.

Approaches for Word Embedding

- **Window based co-occurrence matrix**
  - Example corpus:
    - I like deep learning.
    - I like NLP.
    - I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Approaches for Word Embedding

- **Problems with simple co-occurrence vectors**
  - Increase in size with vocabulary
  - Very high dimensional: require a lot of storage
  - Subsequent classification models have sparsity issues

  —> Models are less robust

Approaches for Word Embedding

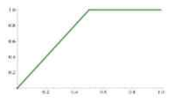
- **Count based vs direct prediction**

LSA, HAL (Lund , Burgeess), COALS (Rohde et al), Hellinger-PCA (Lebret, Collobert)	NNLM, HLBL, RNN, Skip-gram CBOW, (Bengio et al; Collobert, Weston; Huang et al; Mnih, Hinton; Mikolov et al;Mnih , Kavukcuoglu)
• Fast training • Efficient usage of statistics • Primarily used to capture word similarity • Disproportionate importance given to large counts	• Scales with corpus size • Inefficient usage of statistics • Generate improved performance on other tasks • Can capture complex patterns beyond word similarity

Approaches for Word Embedding

- **Combining the best of both worlds: GloVe**

$$J(\theta) = \frac{1}{2}\sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

$f \sim$ 

  - Fast training
  - Scalable to huge corpora
  - Good performance even with small corpus, and small vectors

49

49

Approaches for Word Embedding

- **What to do with the two sets of vectors?**
  - We end up with U and V from all the vectors u and v (in columns)
  - Both capture similar co-occurrence information. It turns out, the best solution is to simply sum them up:

$$X_{final} = U + V$$

  - One of many hyperparameters explored in GloVe: Global Vectors for Word Representation (Pennington et al. (2014))

50

50

Approaches 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

51

51

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.
- Jeong, S., "Introduction to Deep Learning," *tutorial of NLP*, 2015.
- Lee, C., "Word and Phrase Embedding," *tutorial of NLP*, 2015.
- Manning, C. D., Natural Language Processing with Deep Learning, course (http://web.stanford.edu/class/cs224n/), 2017.
- Kim, S., Simple Pytorch tutorial Zero to All, https://github.com/hunkim/PyTorchZeroToAll, 2017

52

52

Thank you for your attention!

http://nlplab.skku.edu

고 영 중