cs224n Lecture 2~3

이번주 하려고 했었던 것들

- 1. Lecture2 최종정리
- 2. Lecture3 정리, 발표
- 3. Markdown 학습

Gradient

기울기의 수학적 정의 [편집]

스칼라 함수 $f(x)$의 기울기는 ∇f로 표현한다. ∇ 기호는 벡터 미분 연산자로 나블라(nabla) 혹은 델(del)연산자라고 부른다.

기울기는 f의 각 성분의 편미분으로 구성된 열벡터로 정의하며 다음과 같이 표시한다.

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$$

Gradinet 의 방향은 함수값이 커지는 방향

-gradient는 함수값이 작아지는 방향이 된다.

Vector_derivative

Scalar-by-vector [edit]

The derivative of a scalar y by a vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$,

$$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} & \cdots & \frac{\partial y}{\partial x_n} \end{bmatrix}.$$

Gradient와 같은 맥락이다.

or

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

Negative Log Likelihood $J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\ne 0}} \log p(w_{t+j}|w_t)$

For $p(w_{t+j}|w_t)$ the simplest first formulation is

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

$$\frac{\partial}{\partial v_c} \log(p(o|c)) = u_o - \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \cdot \left(\sum_{x=1}^{V} \exp(u_x^T v_c)\, u_x \right)$$

$$= u_o - \sum_{x=1}^{V} \frac{\exp(u_x^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}\, u_x$$

Distribute term across sum

$$= u_o - \underbrace{\sum_{x=1}^{V} p(x|c)\, u_x}$$

This is an expectation:

Uo는 context vector들의 평균.

so $\nabla_\theta J_t(\theta)$ is very sparse!

$$\nabla_\theta J_t(\theta) = \begin{bmatrix} 0 \\ \vdots \\ \nabla_{v_{like}} \\ \vdots \\ 0 \\ \nabla_{u_I} \\ \vdots \\ \nabla_{u_{learning}} \\ \vdots \end{bmatrix} \in \mathbb{R}^{2dV}$$

Gradient Descent

- To minimize $J(\theta)$ over the full batch (the entire training data) would require us to compute gradients for all windows

- Updates would be for each element of θ :

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$

- With step size α
- In matrix notation for all parameters:

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

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

Stochastic Gradient Descent

- But Corpus may have 40B tokens and windows
- You would wait a very long time before making a single update!

- **Very** bad idea for pretty much all neural nets!
- Instead: We will update parameters after each window t
  → Stochastic gradient descent (SGD)

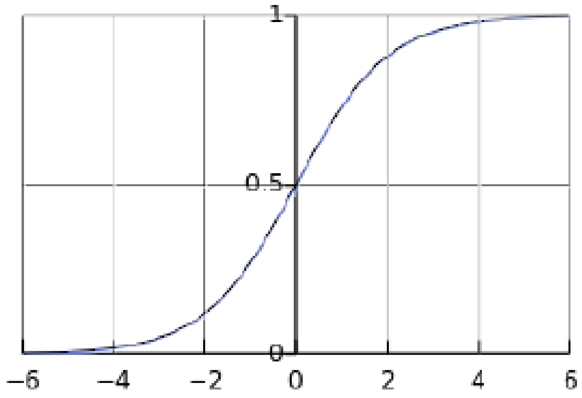
$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J_t(\theta)$$

Ass 1: 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: $J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J_t(\theta)$

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

- 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 →



Ass 1: The skip-gram model and negative sampling

- Slightly clearer notation:

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

- Negative Log Likelihood $\quad J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log p(w_{t+j} \mid w_t)$

- $P(w) = U(w)^{3/4}/Z$,
  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

Co-occurrence matrix X

With a co-occurrence matrix X

- 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

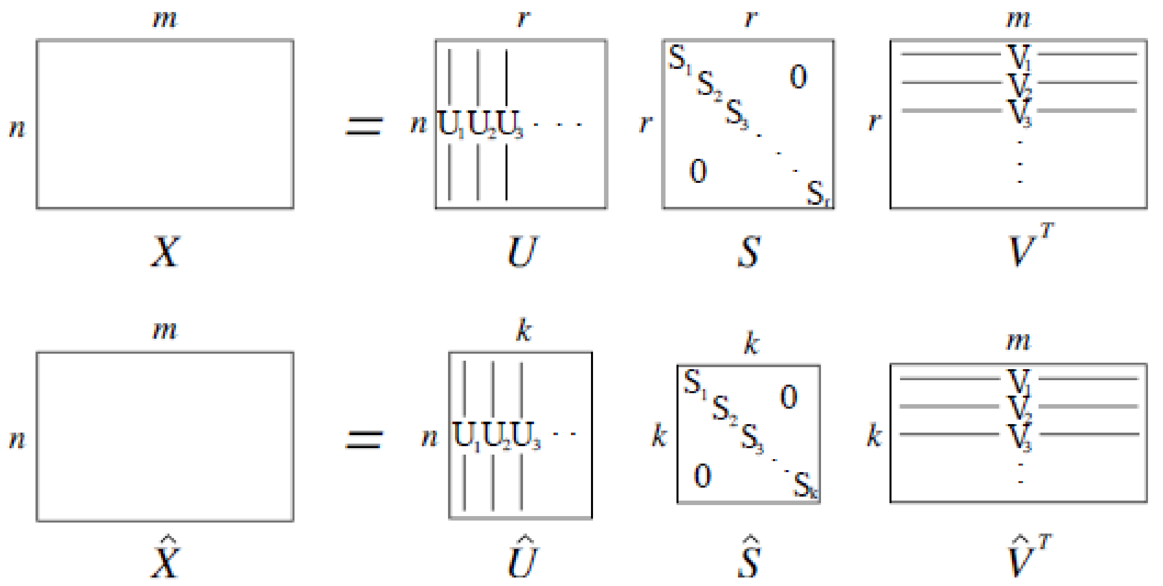
Window based co-occurrence matrix

- 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

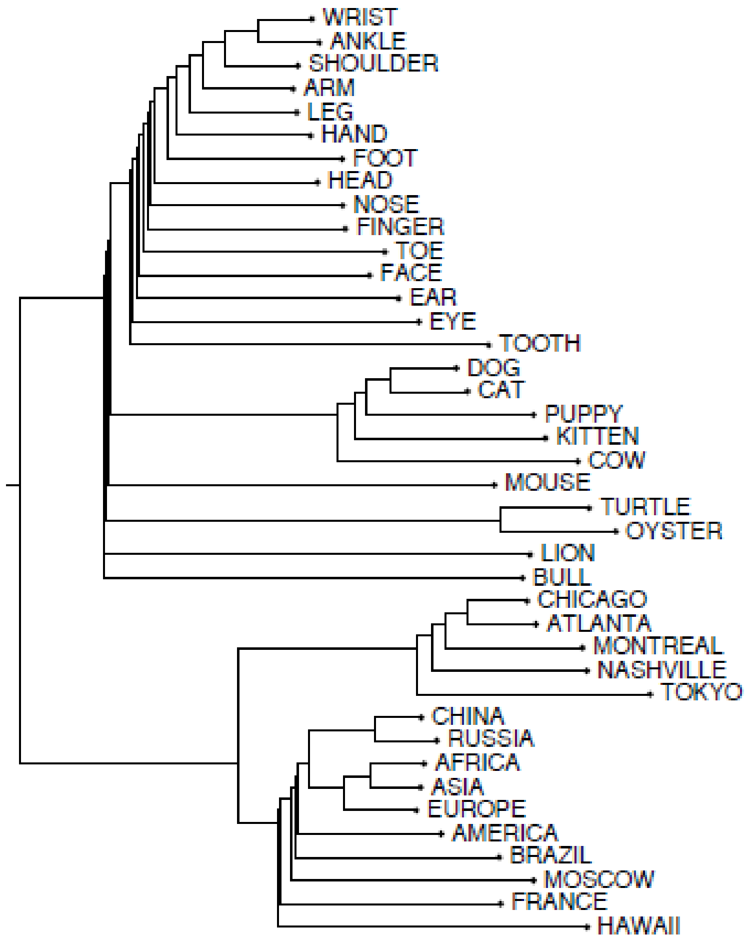
Method 1: Dimensionality Reduction on X

Singular Value Decomposition of co-occurrence matrix *X*.



$\hat{X}$ is the best rank k approximation to X, in terms of least squares.

Interesting semantic patters emerge in the vectors



An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence
Rohde et al. 2005

Count based vs direct prediction

LSA, HAL (Lund & Burgess),
COALS (Rohde et al),
Hellinger-PCA (Lebret & Collobert)

- Fast training
- Efficient usage of statistics
- Primarily used to capture word similarity
- Disproportionate importance given to large counts

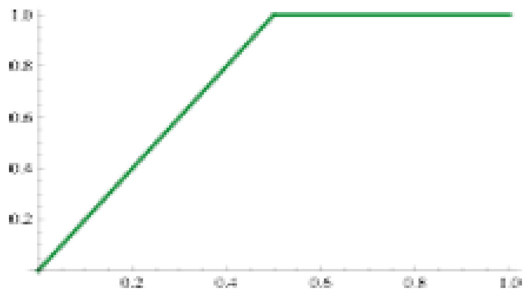
NNLM, HLBL, RNN, Skip-gram/CBOW, (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton; Mikolov et al; Mnih & Kavukcuoglu)

- Scales with corpus size
- Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity

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

다음 세션까지 할 것들

- 1. Lecture3 최종정리

-Unigram분포 이해

-SVD 이해

- 2. 다음 Lecture

- 3. Markdown 숙지