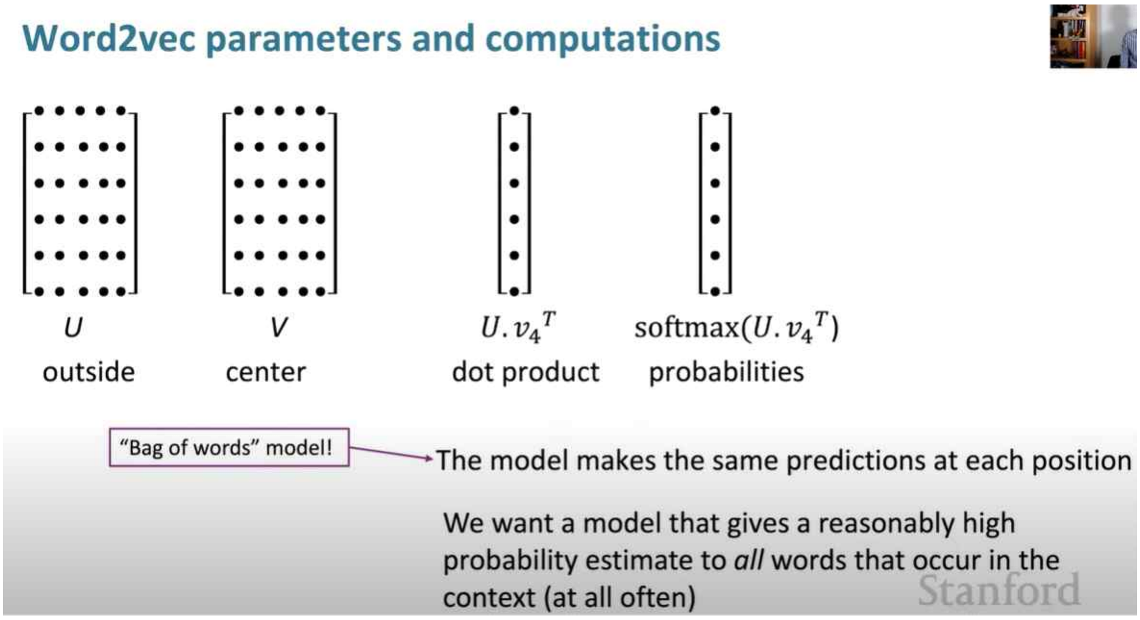
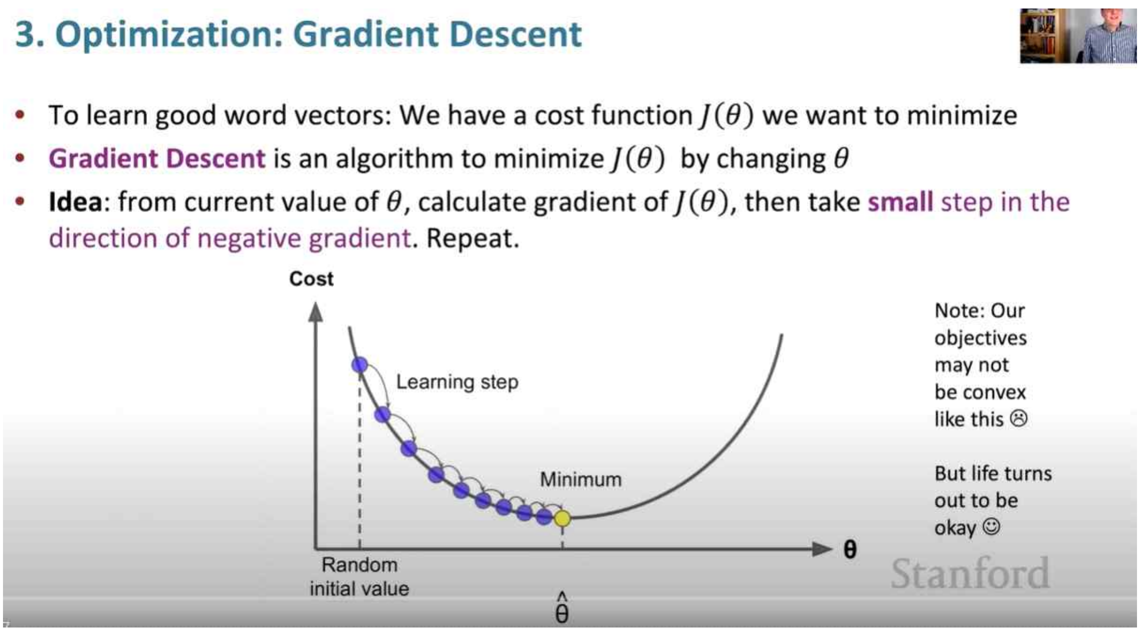
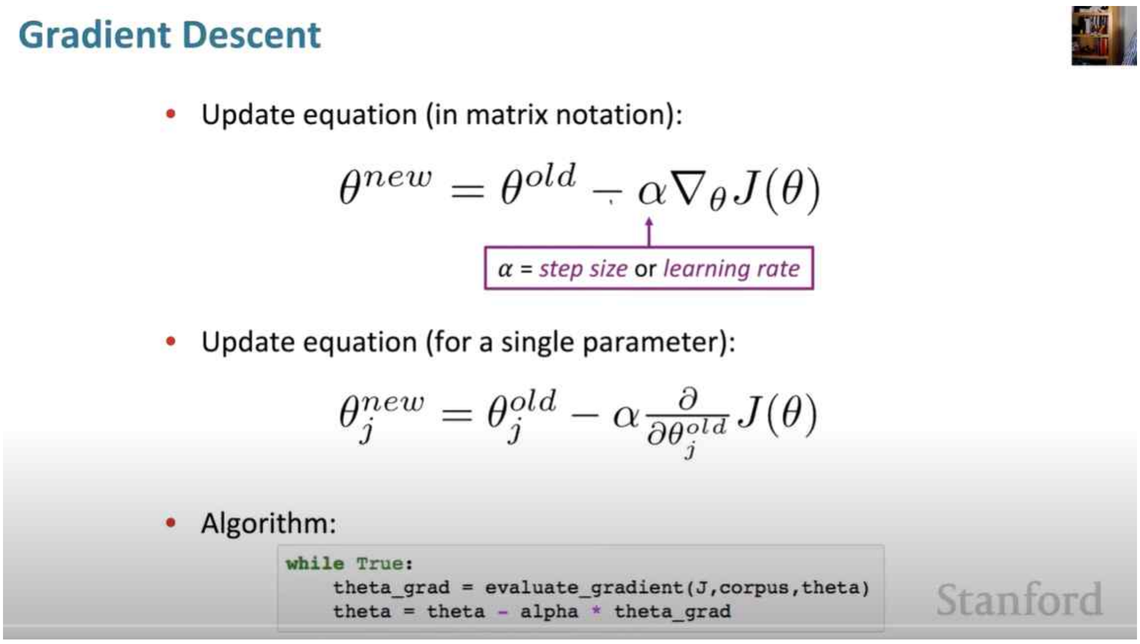
Lec2: Word Vectors, Word Senses, and Neural Network Classifiers



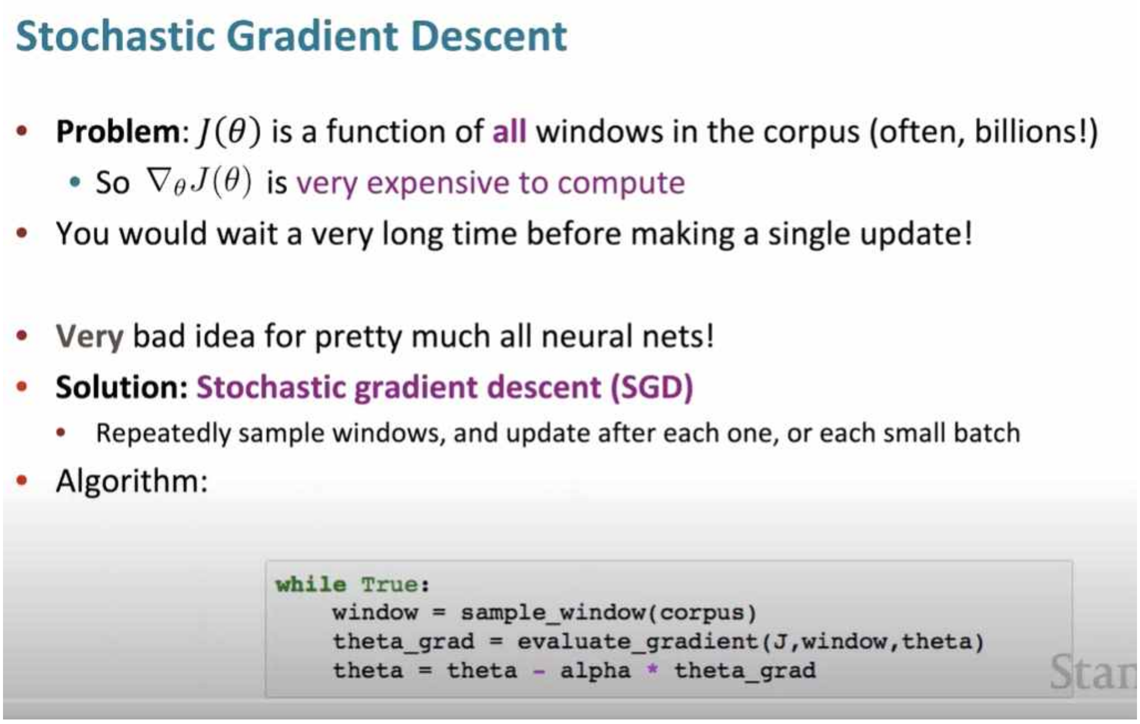
<지난 시간> Word2vec parameters and computations
Bag of words model: 단어 순서나 위치에 상관하지 않음(center word에서 왼or오 상관하지 않음. 즉 추정확률이 모두 같음("대충 만든 모델"), 각각의 위치에서 같은 예측

How do we learn good word vectors?
: Optimization: gradient descent
iterative algorithm: maximize J of theta by changing theta

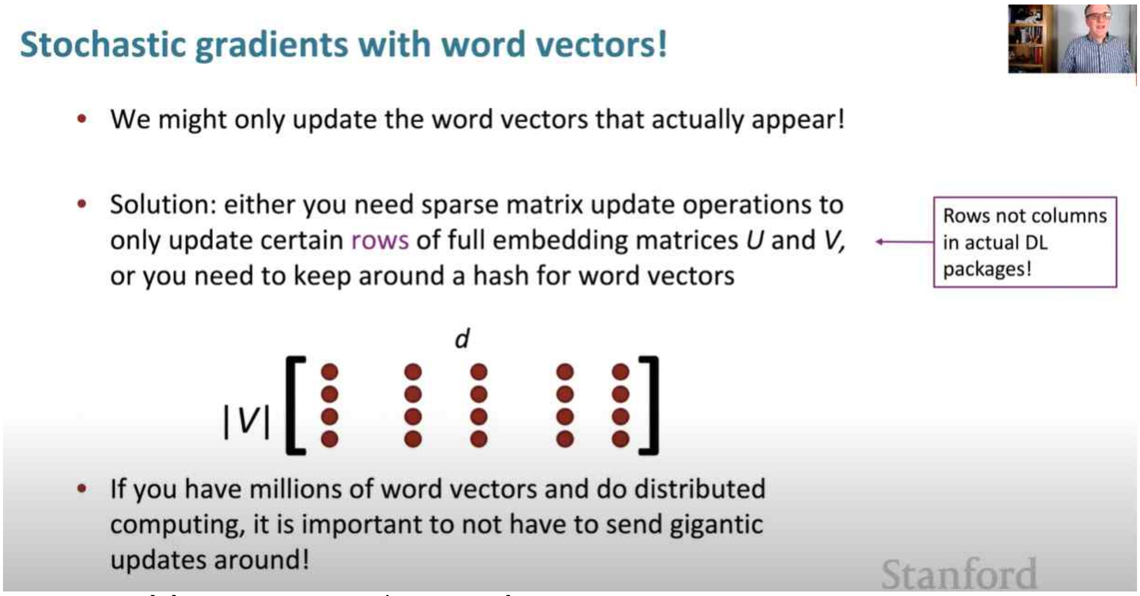




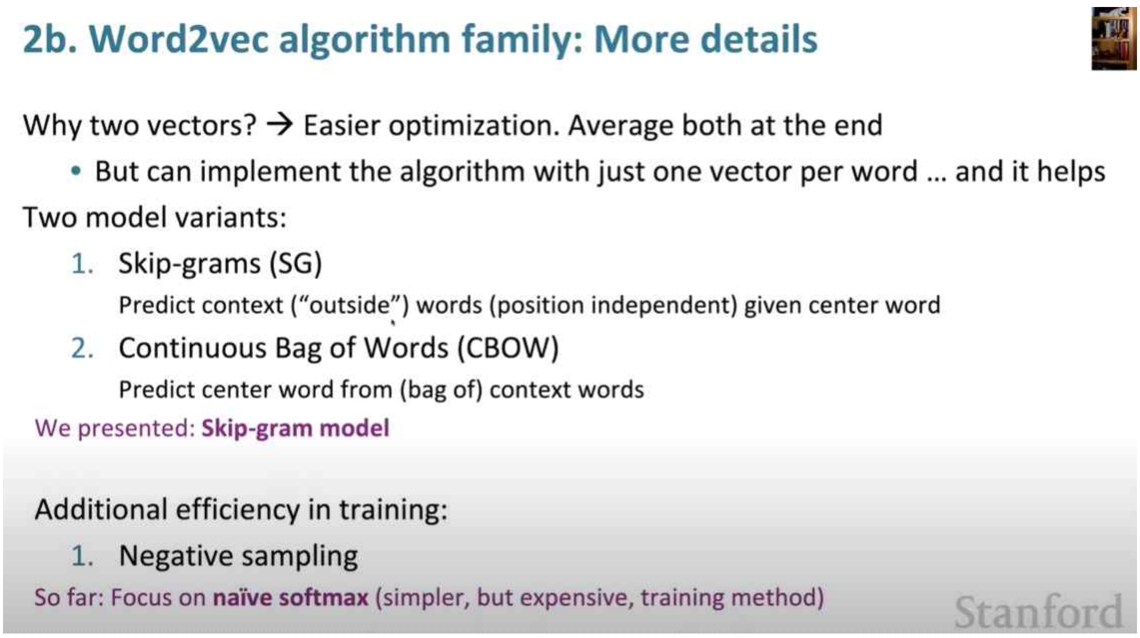
- alpha(step size or learning rate) 사용하여 negative direction으로 gradient 조금씩 이동하며 계산 -> new parameter value -> 아무도 사용하지 않는 방법!



- $J(\theta)$의 문제: extremely expensive(모든 corpus 계산할 수 없음) -> Stochastic Gradient Descent: 모든 말뭉치 사용하는 대신 하나의 center word 또는 small batch 사용



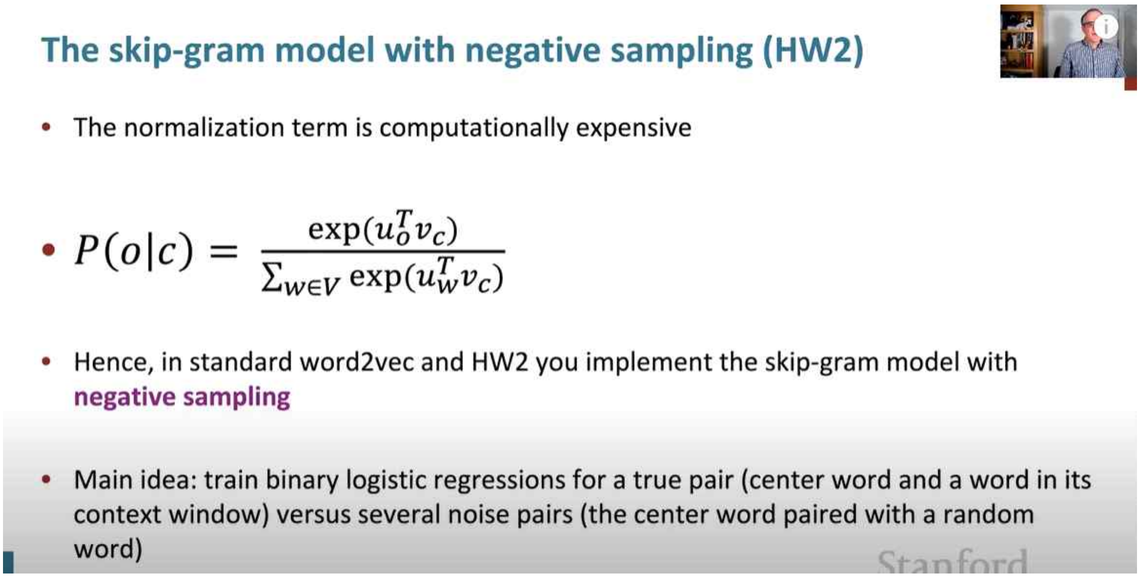
represented by row vectors in pytorch



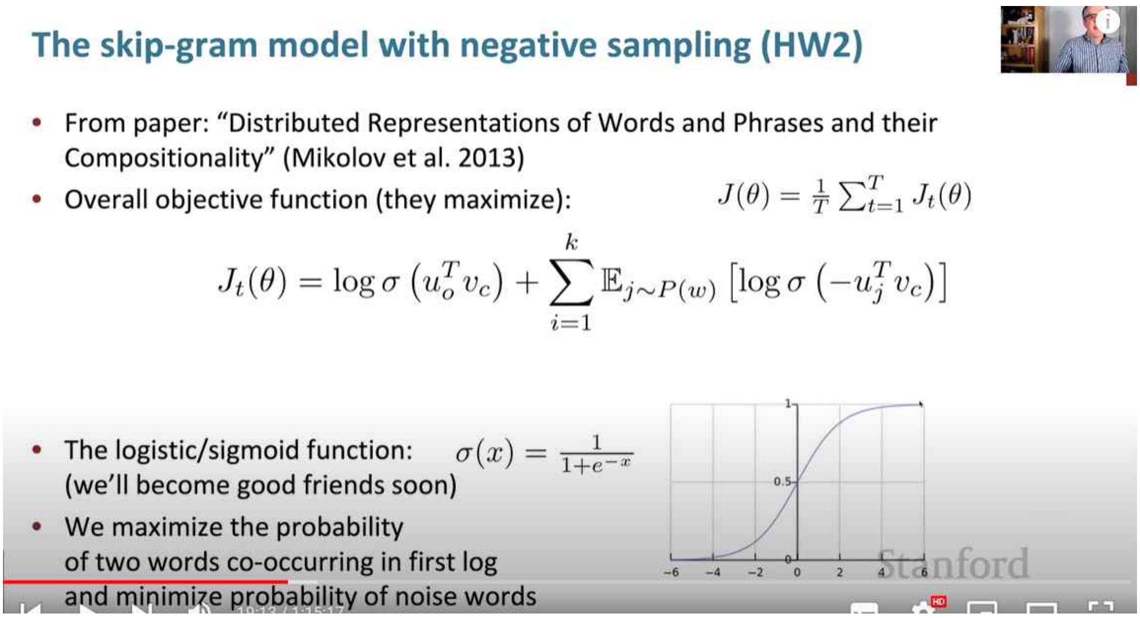
word2vec algorithm -> 각 단어 당 하나의 벡터 사용

Two model variants:
-1. Skip-grams(SG) -> 다양한 상황에서 더 자연스러움
-2. Continuous Bag of Words(CBOW)

naive softmax -> 많이 사용하지만, 분모 부분: expensive(->모든 어휘의 단어들을 iteration&dot product 해야하기 때문)



softmax 대신 nagative sampling: main idea는 이분형 로지스틱 회귀모형을 center word의 true pair와 context word 모두에 훈련시키는 것(versus several noise pairs)



dot product가 크면 lodistic dot product도 virtually 1
sigmoid function은 symmetric -> negative dot product -> 작은 확률

The skip-gram model with negative sampling (HW2)

- Notation more similar to class and HW2:

$$J_{neg-sample}(\boldsymbol{u}_o, \boldsymbol{v}_c, U) = -\log \sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c) - \sum_{k \in \{K \text{ sampled indices}\}} \log \sigma(-\boldsymbol{u}_k^T \boldsymbol{v}_c)$$

- We take *k* negative samples (using word probabilities)
- Maximize probability that real outside word appears, minimize probability that random words appear around center word
- Sample with $P(w)=U(w)^{3/4}/Z$, 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

Stanford

\-

Co-occurrence matrix
-2가지 Building 방법(Window/full document)
-Window: word2vec과 비슷하게 use window around each word

-Simple count co-occurrence vectors: 벡터 사이즈 너무 큼, models are less robust
-Low-dimensional vectors: 더 일반적으로 사용, 가장 많은 정보 포함하면서 최대한 압축하고자 함. -> How?

->Classic Method: Dimensionality reduction on X(SVD)
-->잘 작동x -> scaling

- Scaling the counts in the cells can help ***a lot***
  - Problem: function words (*the, he, has*) are too frequent → syntax has too much impact. Some fixes:
    - log the frequencies
    - min(X,t), with t ≈ 100
    - Ignore the function words

<GloVe algorithm>

5. Towards GloVe: Count based vs. direct prediction

• LSA, HAL (Lund & Burgess), • COALS, Hellinger-PCA (Rohde et al, Lebret & Collobert)	• Skip-gram/CBOW (Mikolov et al) • NNLM, HLBL, RNN (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton)
• 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

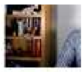
Q: How can we capture ratios of co-occurrence probabilities as linear meaning components in a word vector space?

A: Log-bilinear model: $$w_i \cdot w_j = \log P(i|j)$$

with vector differences $$w_x \cdot (w_a - w_b) = \log \frac{P(x|a)}{P(x|b)}$$

Combining the best of both worlds

GloVe [Pennington, Socher, and Manning, EMNLP 2014]

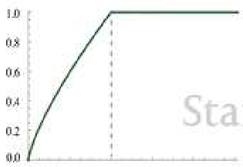


$$w_i \cdot w_j = \log P(i|j)$$

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

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

$f \sim$



Stanford

-General evaluation in NLP: intrinsic vs. extrinsic

Intrinsic word vector evaluation

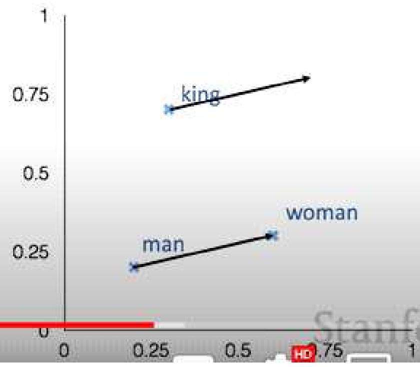
- Word Vector Analogies

a:b :: c:?

man:woman :: king:?

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

- Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions
- Discarding the input words from the search!
- Problem: What if the information is there but not linear?



Stanford

Extrinsic word vector evaluation

- Extrinsic evaluation of word vectors: All subsequent NLP tasks in this class. More examples soon.
- One example where good word vectors should help directly: **named entity recognition**: identifying references to a person, organization or location

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

Stanford