# NLP Week 2

## Lecture Plan

Lecture 2: Word Vectors, Word Senses, and Neural Network Classifiers

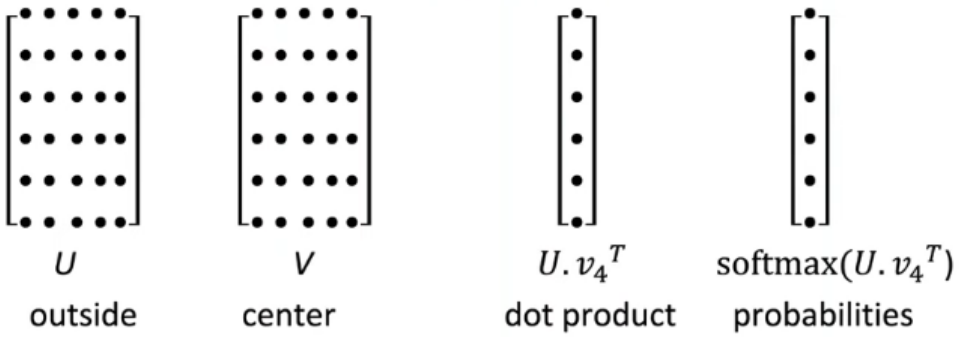
1. Course organization (2 mins)
2. Finish looking at word vectors and word2vec (10 mins)
3. Optimization basics (8 mins)
4. Can we capture the essence of word meaning more effectively by counting? (8m)
5. The GloVe model of word vectors (8 min)
6. Evaluating word vectors (12 mins)
7. Word senses (6 mins)
8. Review of classification and how neural nets differ (8 mins)
9. Introducing neural networks (14 mins)

Key Goal: To be able to read word embeddings papers by the end of class

Stanford

2

## Word2vec parameters and computations




**Word2vec maximizes objective function(J) by putting similar words nearby in space**

## Gradient Descent

- Update equation (in matrix notation):

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

$\alpha$ = *step size* or *learning rate*

- Update equation (for a single parameter):

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

## Stochastic Gradient Descent

Problem : J is a function of all windows in the corpus

Solution : SGD

repeatedly sample windows and update after each one or each small batch

- But in each window, we only have at most $2m + 1$ words, so $\nabla_\theta J_t(\theta)$ is very sparse!

## 2b. Word2vec algorithm family : More details

Two model variants

1. Skip-grams
2. CBOW

additional efficiency

1. Negative sampling

## The skip-gram model with negative Sampling

- The normalization term is computationally expensive

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

Main : train binary logistic regression for a true pair (center-context) VS several noise pairs(the center word-random words)

- objective function

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

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

- logistic/sigmoid function y: 0~1
- Maximize the prob of two words co-occurring in first log

and minimize the prob of noise words

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

- take k negative samples
- Max-prob that real outside word appears, Min-prob that random words appear around center words
- Sample with P(w) = U(w)^(3/4) / Z (less frequent→ sampled more)

## 4. Why not capture co-occurrence counts directly?

Building a co-occurrence matrix X

- windows vs full document

- window : captures syntactic(구문) and semantic info
- Word-docu co-occurrence matrix will give general topics leading to ‘LSA’(잠재의미분석)

## 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

| 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 |

Stanford

17

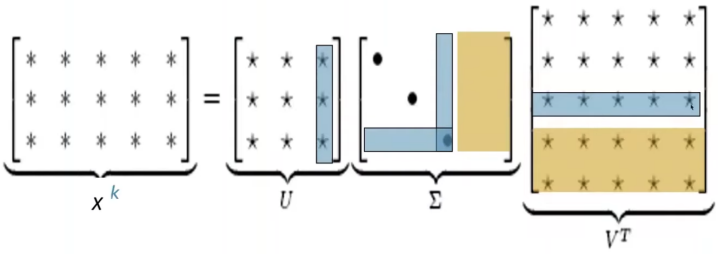
# Co-occurrence vectors

- simple count co-occur vec
  - very high dimensional
  - sparsity issues
- Low-dimensional vec
  - idea: store ‘most’ of the important info in a fixed small # of dimension : a **dense vector**

## Classic Method: Dimensionality Reduction on X (HW1)

Singular Value Decomposition of co-occurrence matrix $X$

Factorizes $X$ into $U\Sigma V^T$, where $U$ and $V$ are orthonormal



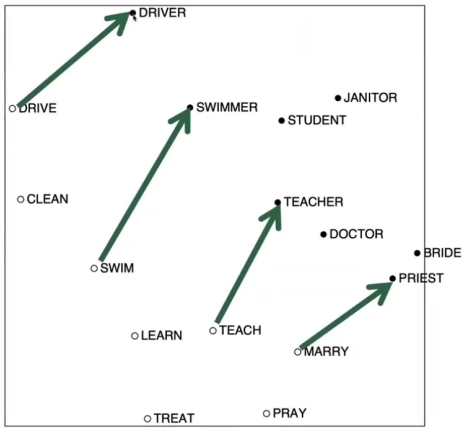

Retain only $k$ singular values, in order to generalize.
$\hat{X}$ is the best rank $k$ approximation to $X$, in terms of least squares.
Classic linear algebra result. Expensive to compute for large matrices.





delete more of the matrix

## Interesting semantic patterns emerge in the scaled vectors




COALS model from

## 5. Towards GloVe: Count based vs. direct prediction

- LSA, HAL, COALS, PCA
- Fast training
- efficient usage of statistics
- used to capture word similarity
- Disproportionate importance given to large corpus

- Skip-Gram/CBOW, NNLM,HLBL, RNN
- Scales with corpus size
- Inefficient usage of statistics
- generate improved performance
- Can capture complex patterns (ex.analogy)

## Encoding meaning  in vector differences

Crucial insight : Ratio of co-occurrence probs can encode meaning components

| | $x$ = solid | $x$ = gas | $x$ = water | $x$ = random |
|---|---|---|---|---|
| $P(x\|\text{ice})$ | large | small | large | small |
| $P(x\|\text{steam})$ | small | large | large | small |
| $\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$ | large | small | ~1 | ~1 |

Stanf

Q : How can we capture ratios of co-occurrence probs 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

objective funtion J

wi,wj - log Xij : as small as possible

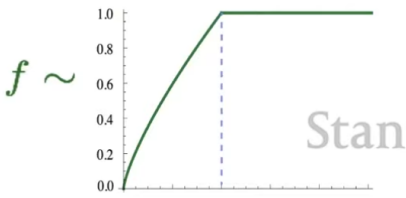
f(Xij) : depending on the frequency of a word

: pay more attention to more common words.

But  having extremely common words like function words (he, the..) leads you astray. so paid more attention to words that co-occur until a certain point. Then the curve went flat.

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

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



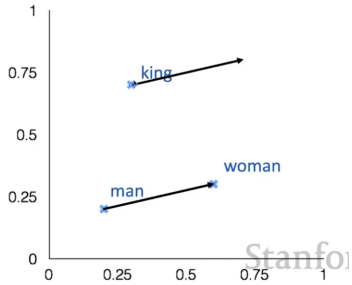

## Intrinsic word vector evaluation

- Word Vector Analogies

a:b :: c:?

man:woman :: king:?

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




cosine distance

problem : what is the information is there but not linear?

## 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

## Linear Algebraic Structure of Word Senses, with Applications to Polysemy (Arora, …, Ma, …, TACL 2018)

- Different senses of a word reside in a linear superposition (weighted sum) in standard word embeddings like word2vec
- $v_{\text{pikė}} = \alpha_1 v_{\text{pike}_1} + \alpha_2 v_{\text{pike}_2} + \alpha_3 v_{\text{pike}_3}$
- Where $\alpha_1 = \frac{f_1}{f_1+f_2+f_3}$, etc., for frequency $f$
- Surprising result:
  - Because of ideas from *sparse coding* you can actually separate out the senses (providing they are relatively common)!

| tie | | | | |
|---|---|---|---|---|
| trousers | season | scoreline | wires | operatic |
| blouse | teams | goalless | cables | soprano |
| waistcoat | winning | equaliser | wiring | mezzo |
| skirt | league | clinching | electrical | contralto |
| sleeved | finished | scoreless | wire | baritone |
| pants | championship | replay | cable | coloratura |

Stanford

41