Deep Learning Basics
Lecture 10: Neural Language Models

Princeton University COS 495

Instructor: Yingyu Liang

Natural language Processing (NLP)

- The processing of the human languages by computers

- One of the oldest AI tasks
- One of the most important AI tasks
- One of the hottest AI tasks nowadays

Difficulty

- Difficulty 1: ambiguous, typically no formal description

- Example: “*We saw her duck.*”
- 1. We looked at a duck that belonged to her.
- 2. We looked at her quickly squat down to avoid something.
- 3. We use a saw to cut her duck.

Difficulty

- Difficulty 2: computers do not have human concepts
- Example: “*She like little animals. For example, yesterday we saw her duck.*”
- 1. We looked at a duck that belonged to her.
- 2. We looked at her quickly squat down to avoid something.
- 3. We use a saw to cut her duck.

Statistical language model

Probabilistic view

- Use probabilistic distribution to model the language
- Dates back to Shannon (information theory; bits in the message)

Statistical language model

- Language model: probability distribution over sequences of tokens
- Typically, tokens are words, and distribution is discrete

- Tokens can also be characters or even bytes

- Sentence: "*the quick brown fox jumps over the lazy dog*"

Tokens: x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9

Statistical language model

- For simplification, consider fixed length sequence of tokens (sentence)

$$(x_1, x_2, x_3, \dots, x_{\tau-1}, x_\tau)$$

- Probabilistic model:

$$\mathrm{P}\left[x_1, x_2, x_3, \dots, x_{\tau-1}, x_\tau\right]$$

N-gram model

n-gram model

- n-gram: sequence of n tokens

- n-gram model: define the conditional probability of the n-th token given the preceding $n-1$ tokens

$$\mathrm{P}[x_1, x_2, \ldots, x_\tau] = \mathrm{P}[x_1, \ldots, x_{n-1}] \prod_{t=n}^{\tau} \mathrm{P}[x_t | x_{t-n+1}, \ldots, x_{t-1}]$$

n-gram model

- n-gram: sequence of n tokens

- n-gram model: define the conditional probability of the n-th token given the preceding $n-1$ tokens

$$\mathrm{P}[x_1, x_2, \ldots, x_\tau] = \mathrm{P}[x_1, \ldots, x_{n-1}] \prod_{t=n}^{\tau} \mathrm{P}[x_t | x_{t-n+1}, \ldots, x_{t-1}]$$

Markovian assumptions

Typical n-gram model

- $n = 1$: unigram
- $n = 2$: bigram
- $n = 3$: trigram

Training n-gram model

- Straightforward counting: counting the co-occurrence of the grams

For all grams $(x_{t-n+1}, \ldots, x_{t-1}, x_t)$

- 1. count and estimate $\hat{\mathrm{P}}[x_{t-n+1}, \ldots, x_{t-1}, x_t]$
- 2. count and estimate $\hat{\mathrm{P}}[x_{t-n+1}, \ldots, x_{t-1}]$
- 3. compute

$$\hat{\mathrm{P}}[x_t | x_{t-n+1}, \ldots, x_{t-1}] = \frac{\hat{\mathrm{P}}[x_{t-n+1}, \ldots, x_{t-1}, x_t]}{\hat{\mathrm{P}}[x_{t-n+1}, \ldots, x_{t-1}]}$$

A simple trigram example

- Sentence: "*the dog ran away*"

$$\hat{\mathrm{P}}[the\ dog\ ran\ away] = \hat{\mathrm{P}}[the\ dog\ ran]\ \hat{\mathrm{P}}[away|dog\ ran]$$

$$\hat{\mathrm{P}}[the\ dog\ ran\ away] = \hat{\mathrm{P}}[the\ dog\ ran]\frac{\hat{\mathrm{P}}[dog\ ran\ away]}{\hat{\mathrm{P}}[dog\ ran]}$$

Drawback

- Sparsity issue: $\hat{\mathrm{P}}[\ldots]$ most likely to be 0

- Bad case: *"dog ran away"* never appear in the training corpus, so $\hat{\mathrm{P}}[dog\ ran\ away] = 0$
- Even worse: *"dog ran"* never appear in the training corpus, so $\hat{\mathrm{P}}[dog\ ran] = 0$

Rectify: smoothing

- Basic method: adding non-zero probability mass to zero entries

- Back-off methods: restore to lower order statistics
- Example: if $\hat{\mathrm{P}}[away|dog\ ran]$ does not work, use $\hat{\mathrm{P}}[away|ran]$ as replacement
- Mixture methods: use a linear combination of $\hat{\mathrm{P}}[away|ran]$ and $\hat{\mathrm{P}}[away|dog\ ran]$

Drawback

- High dimesion: # of grams too large

- Vocabulary size: about 10k=2^14
- #trigram: about 2^42

Rectify: clustering

- Class-based language models: cluster tokens into classes; replace each token with its class
- Significantly reduces the vocabulary size; also address sparsity issue

- Combinations of smoothing and clustering are also possible

Neural language model

Neural Language Models

- Language model designed for modeling natural language sequences by using a distributed representation of words

- Distributed representation: embed each word as a real vector (also called word embedding)
- Language model: functions that act on the vectors

Distributed vs Symbolic representation

- Symbolic representation: can be viewed as one-hot vector
- Token i in the vocabulary is represented as e_i

i-th entry

0	0	0	0	1	0	0	0	0	0

- Can be viewed as a special case of distributed representation

Distributed vs Symbolic representation

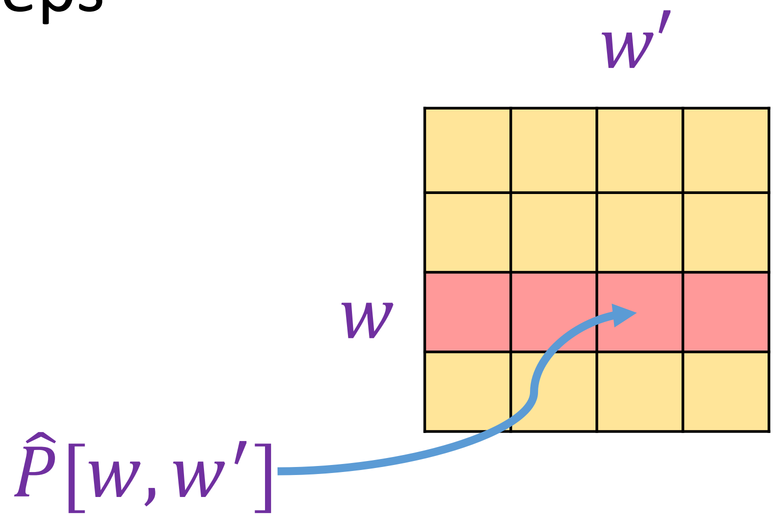
- Word embeddings: used for real value computation (instead of logic/grammar derivation, or discrete probabilistic model)

- Hope that real value computation corresponds to semantics
- Example: inner products correspond to token similarities

- One-hot vectors: every pair of words has inner product 0

Co-occurrence

- Firth's Hypothesis (1957): the meaning of a word is defined by "the company it keeps"

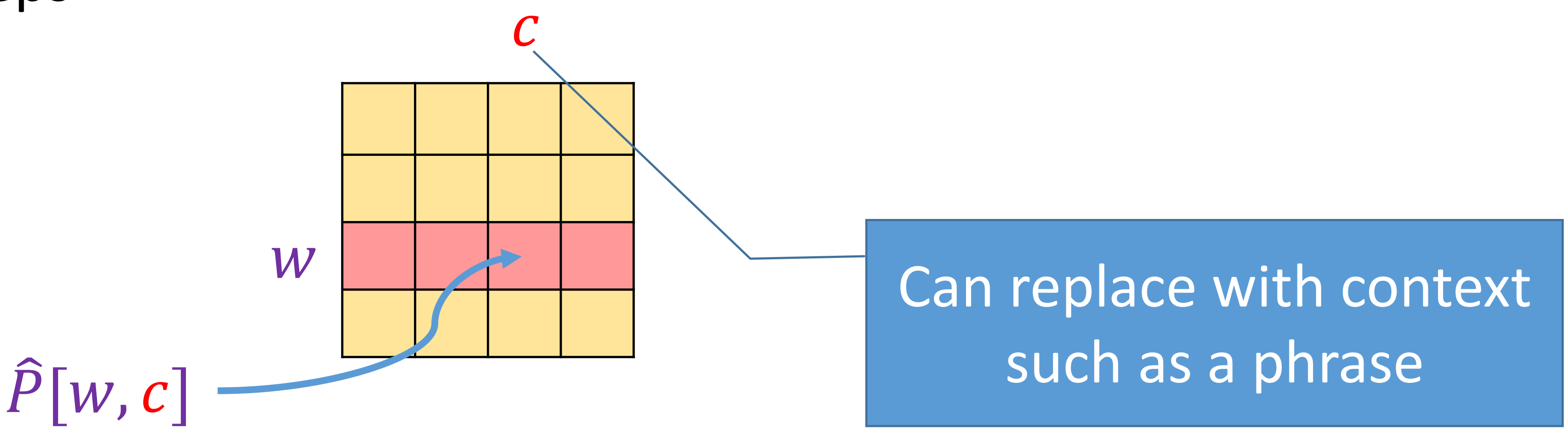


- Use the co-occurrence of the word as its vector:

$$v_w \; := \; \hat{P}[w, :]$$

Co-occurrence

- Firth's Hypothesis (1957): the meaning of a word is defined by "the company it keeps"



- Use the co-occurrence of the word as its vector:

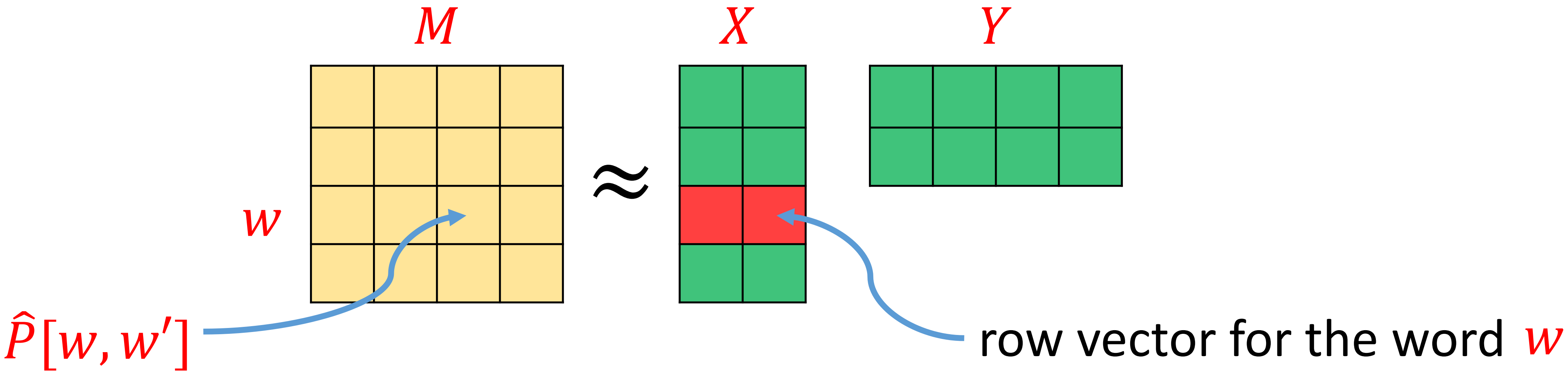
$$v_w \ := \ \hat{P}[w, :]$$

Drawback

- High dimensionality: equal vocabulary size (~10k)
- can be even higher if context is used

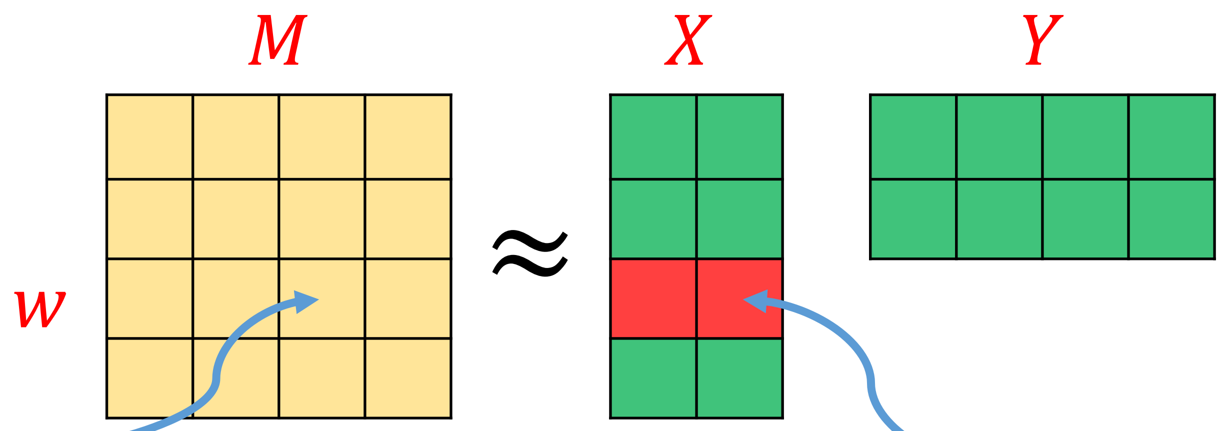
Latent semantic analysis (LSA)

- LSA by Deerwester et al., 1990: low rank approx. of co-occurrence



Variants

- low rank approx. of the transformed co-occurrence



$\sqrt{\hat{P}[w, w']}$

row vector for the word w

Or $\mathrm{PMI}(\mathrm{w}, \mathrm{w}') = \ln \frac{\hat{P}[w,w']}{\hat{P}[w]\hat{P}[w']}$

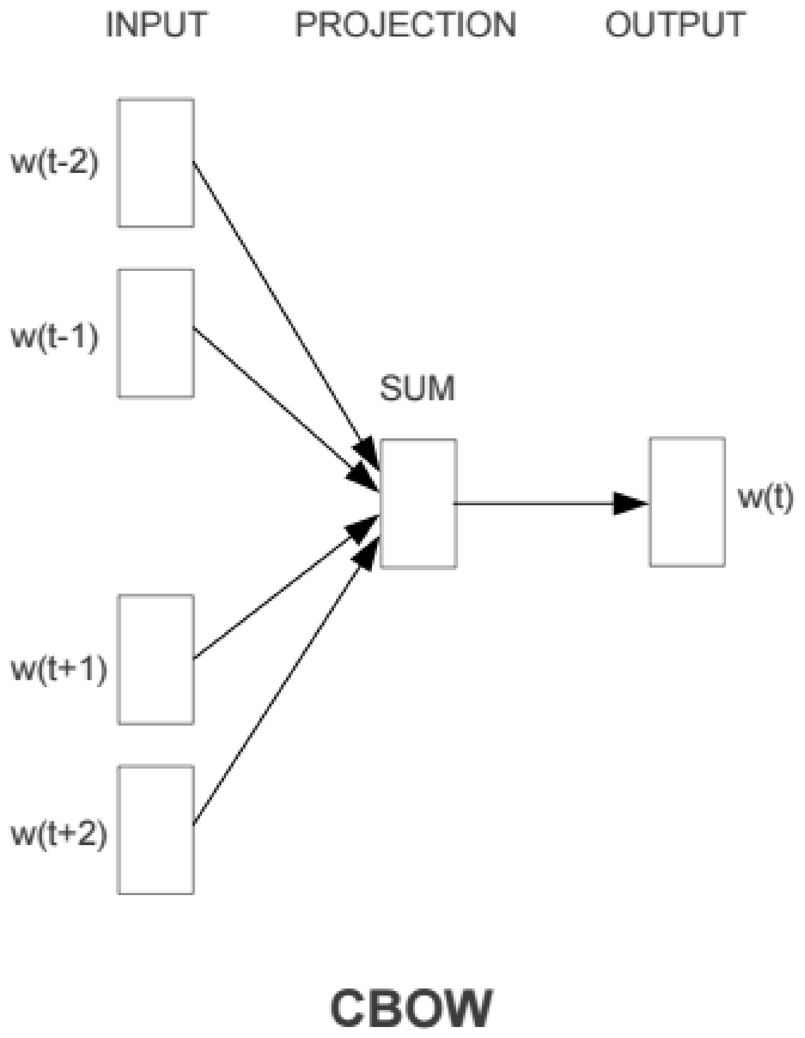
State-of-the-art word embeddings

Updated on April 2016

Word2vec

- Continous-Bag-Of-Words

Figure from
Efficient Estimation of Word Representations in Vector Space,
By Mikolov, Chen, Corrado, Dean



$$\mathrm{P}[w_t | w_{t-2}, \dots, w_{t+2}] \propto \exp[v_{w_t} \cdot mean(v_{w_{t-2}}, \dots, v_{w_{t+2}})]$$

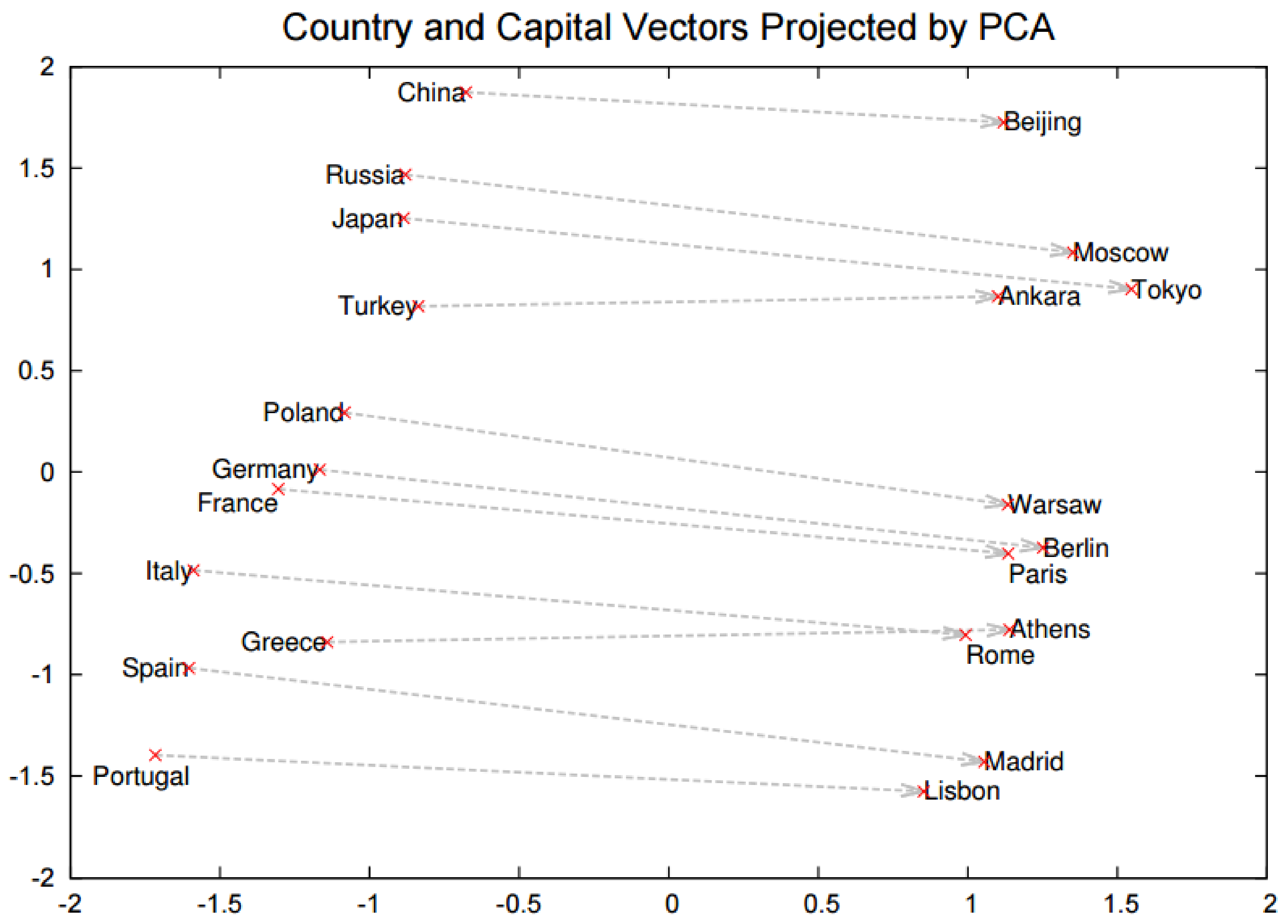
Linear structure for analogies

- Semantic: “man:woman::king:queen”

$$v_{man} - v_{woman} \approx v_{king} - v_{queen}$$

- Syntatic: “run:running::walk:walking”

$$v_{run} - v_{running} \approx v_{walk} - v_{walking}$$


Country and Capital Vectors Projected by PCA
China
Beijing
Russia
Japan
Moscow
Tokyo
Turkey
Ankara
Poland
Germany
France
Warsaw
Berlin
Paris
Italy
Greece
Spain
Athens
Rome
Madrid
Lisbon
Portugal
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

GloVe: Global Vector

- Suppose the co-occurrence between word i and word j is X_{ij}
- The word vector for word i is w_i and $\widetilde{w_i}$
- The GloVe objective function is

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

- Where $b_i's$ are bias terms, $f(x) = min\{100, x^{3/4}\}$

Advertisement

Lots of mysterious things

What are the reasons behind

- The weird transformation on the co-occurrence?
- The model of word2vec?
- The objective of GloVe? The hyperparameters (weights, bias, etc)?

What are the connections between them? A unified framework?

Why do the word vector have linear structure for analogies?

Advertisement

- We proposed a generative model with theoretical analysis:

[RAND-WALK: A Latent Variable Model Approach to Word Embeddings]

- Next lecture by Tengyu Ma, presenting this work

Can't miss!