ENGR-E 533 “Deep Learning Systems” Lecture 15: Word Embedding

Minje Kim
Department of Intelligent Systems Engineering
Email: minje@indiana.edu
Website: http://minjekim.com
Research Group: http://saige.sice.indiana.edu
Meeting Request: http://doodle.com/minje

INDIANA UNIVERSITY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

How to Represent Words?

- Once again we seek latent representations

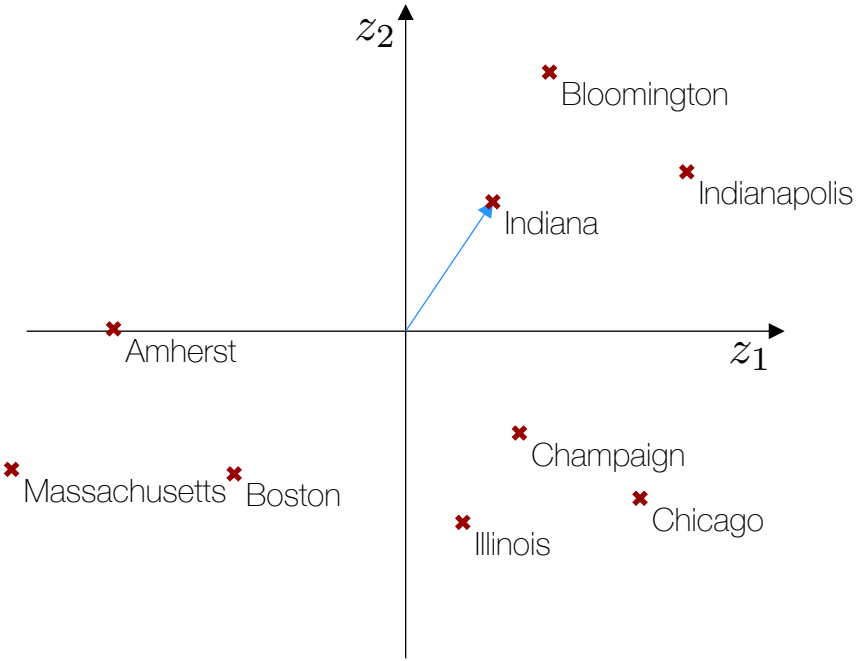
- Say that we're interested in these words:
  - Illinois, Chicago, Champaign, Indiana, Indianapolis, Bloomington
- How do we represent them for computer algorithms to process?
  - One-hot vectors have been common
- Any problems with this?
  - Too high dimensional
    - There are about 700k words in English
  - Not really a feature vector
    - Very discriminant but that's it
    - Difficult to compare them

Illinois	1	0	0	0	0	0
	…	…	…	…	…	…
	…	…	…	…	…	…
Chicago	0	1	0	0	0	0
	…	…	…	…	…	…
	…	…	…	…	…	…
	…	…	…	…	…	…
Champaign	0	0	1	0	0	0
Indiana	0	0	0	1	0	0
	…	…	…	…	…	…
	…	…	…	…	…	…
Indianapolis	0	0	0	0	1	0
	…	…	…	…	…	…
Bloomington	0	0	0	0	0	1

How to Represent Words?

- Once again we seek latent representations

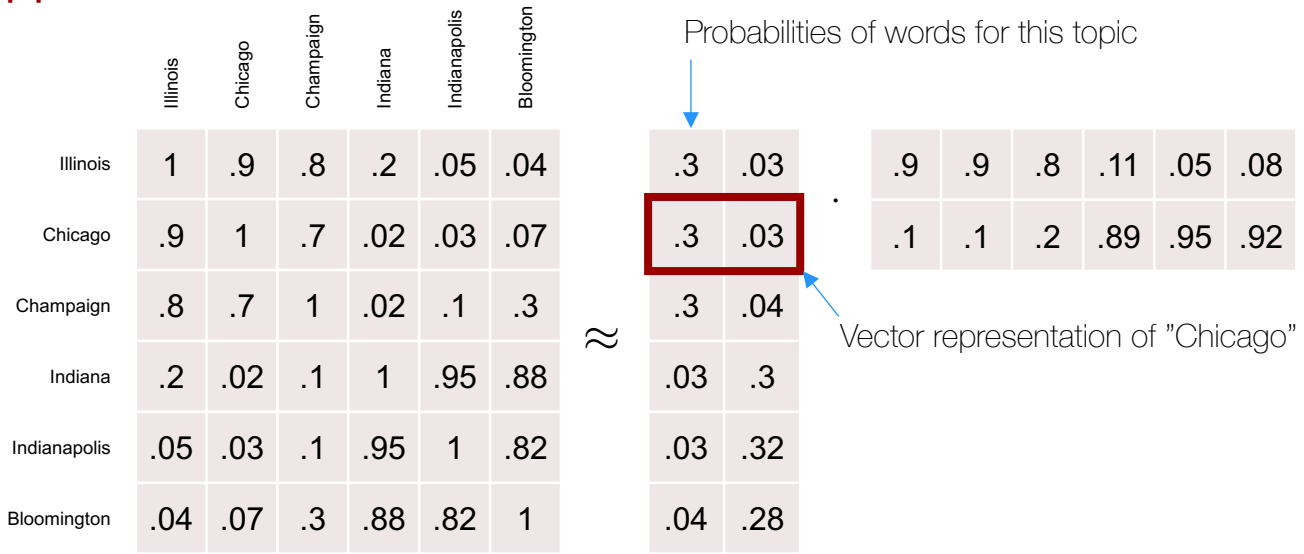
- A better latent representation



Vector Representation of Words

- A matrix factorization approach

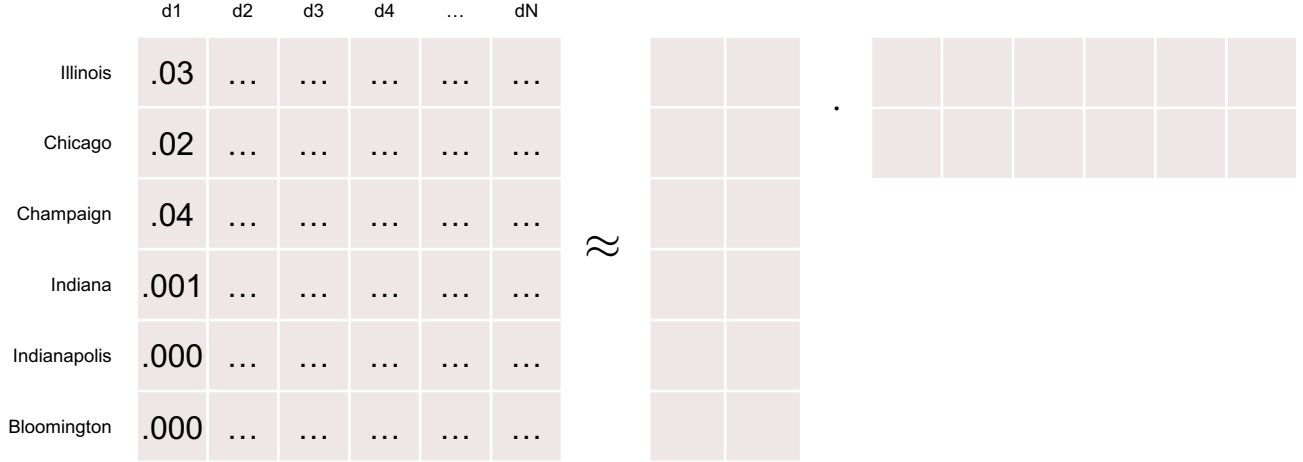
- Similarity matrix
  - e.g. co-occurrence
- Matrix factorization
  - Eigendecomposition
  - SVD
  - NMF
  - Topic modeling
  - You name it
- Implication
  - Each word has a K-d vector representation
    - K is the low rank of your choice
  - Similar words often appear in the same topic together
  - Each word has a unique activation pattern over the latent variables



Vector Representation of Words

- A matrix factorization approach

- Same story for the Term-Frequency (TF) matrix

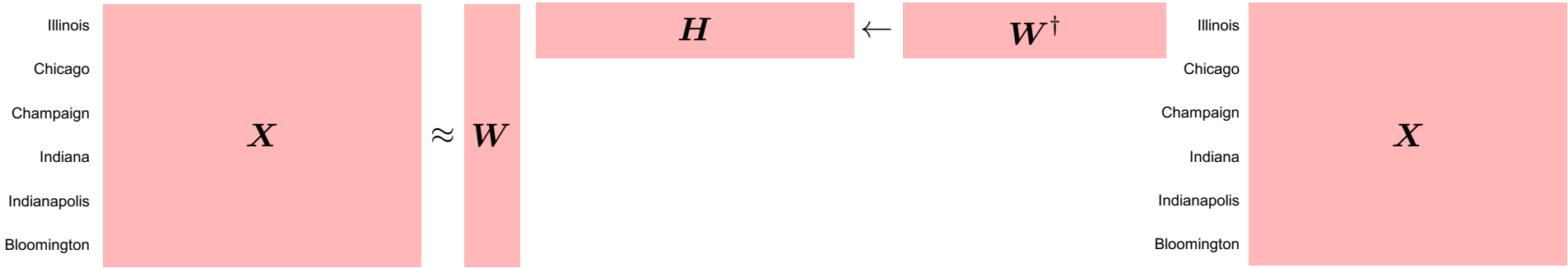


- An equation for this procedure $\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}$
- Where am I going?
  - Autoencoder

Autoencoder for Word Embedding

- The basic structure

- The autoencoding process



$$X \approx WW^{\dagger}X$$

- You'll see this is just a shallow neural network
- In order to make it practical, we need to define
  - The error function
  - Input data
  - Targets
  - Nonlinearity

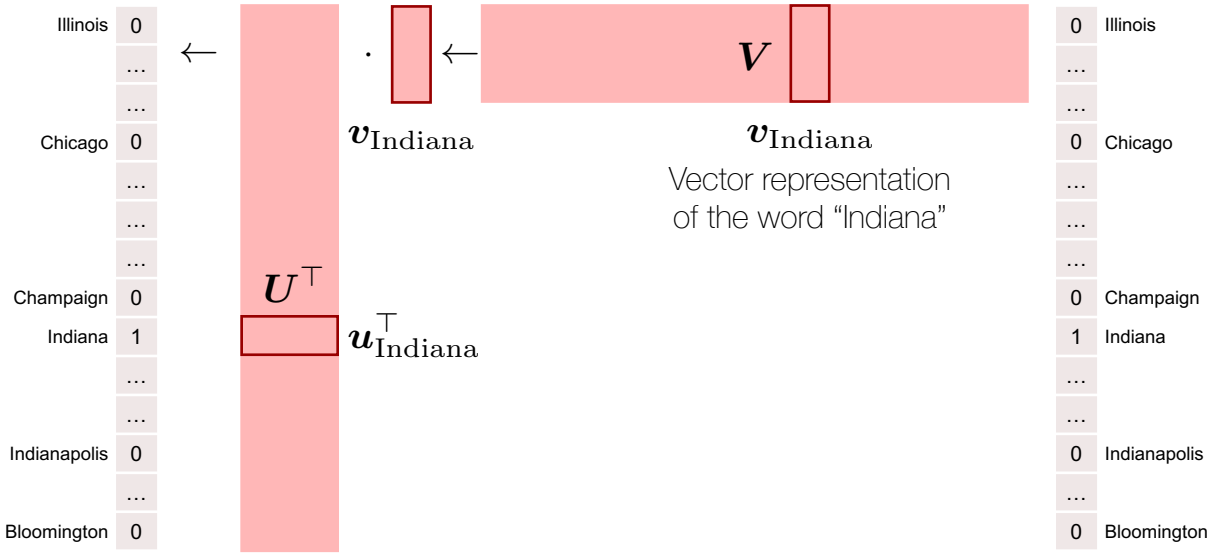
Word2vec

- Skip-gram

- Basic autoencoding network
- Activation function for the output
  - Softmax

$$p(w_t|w_t;\boldsymbol{V},\boldsymbol{U}) = \frac{\exp(\boldsymbol{u}_t^\top \boldsymbol{v}_t)}{\sum_{t'}\exp(\boldsymbol{u}_{t'}^\top \boldsymbol{v}_t)}$$

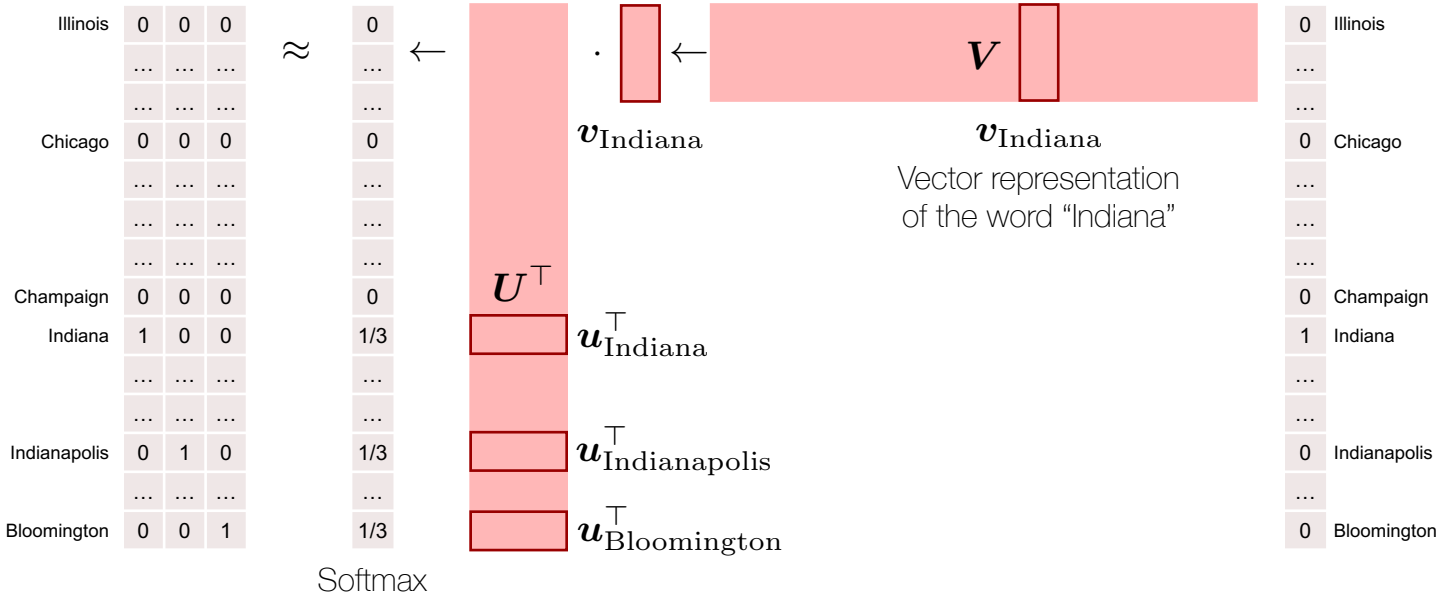
- If this autoencoding job is successful
  - $\boldsymbol{v}_{\text{Indiana}}$ is totally different from any other vectors in $\boldsymbol{U}$, but $\boldsymbol{u}_{\text{Indiana}}$
- Is this good enough? What's the problem with this?
  - Is "Indiana" closer to "Indianapolis" than to "Illinois"?



Word2vec

- Skip-gram

- Another prediction mechanism that encodes co-occurunce



Word2vec

- Negative sampling

- Average negative log-likelihood

$$\mathcal{J}(\boldsymbol{U}, \boldsymbol{V}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{|\tau|\leq m, \tau\neq 0} \log p(w_{t+\tau}|w_t) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{|\tau|\leq m, \tau\neq 0} \log \frac{\exp(\boldsymbol{u}_{t+\tau}^{\top}\boldsymbol{v}_t)}{\sum_{t'}\exp(\boldsymbol{u}_{t'}^{\top}\boldsymbol{v}_t)}$$

context window

except for the input word itself

  - For a given input word word2vec predicts its surrounding words
  - Note that word2vec doesn't predict the input word

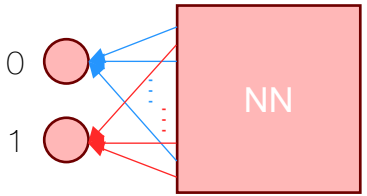
- Any problem?
  - Denominator involves all words

- Negative sampling
  - Word-wise binary classification [Prob. of predicting the word of interest] versus [A randomly sampled word outside of the context]
  - Suppose softmax with two classes
    - You can maximize this instead $\log \sigma(\boldsymbol{u}_{t+\tau}^{\top}\boldsymbol{v}_t) + \left(1 - \log \sigma(\boldsymbol{u}_j^{\top}\boldsymbol{v}_t)\right)$

      Prob. of predicting one of the context word

      Prob. of predicting a negative word
    - Or, we could check on a few more negative samples

$$\log p(w_{t+\tau}|w_t) \approx \log \sigma(\boldsymbol{u}_{t+\tau}^{\top}\boldsymbol{v}_t) + \sum_{j\in P(w)} \left(\log \sigma(-\boldsymbol{u}_j^{\top}\boldsymbol{v}_t)\right)$$



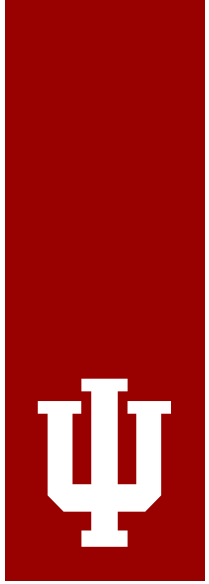
The Other Things & Reading

- GloVe
  - Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.

$$\mathcal{J}(\boldsymbol{U}, \boldsymbol{V}) = \frac{1}{2} \sum_{i,j}^{\mathcal{W}} f(P_{i,j})(\boldsymbol{u}_i^\top \boldsymbol{v}_j - \log P_{i,j})^2$$

Co-occurrence

- FastText https://fasttext.cc
  - Takes the sum of the context words as the input
  - Supported by Facebook AI Research
- Word2vec
  - Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).
  - Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013.
- https://www.tensorflow.org/tutorials/word2vec



Thank You!