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)

\[(x_1, x_2, x_3, ..., x_{\tau-1}, x_{\tau})\]

• Probabilistic model:

\[P [x_1, x_2, x_3, ..., x_{\tau-1}, x_{\tau}]\]
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

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

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

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

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

• Sentence: “the dog ran away”

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

\[
\hat{P}[\text{the dog ran away}] = \hat{P}[\text{the dog ran}] \frac{\hat{P}[\text{dog ran away}]}{\hat{P}[\text{dog ran}]}
\]
Drawback

• Sparsity issue: $\hat{P}[...]$ most likely to be 0

• Bad case: “dog ran away” never appear in the training corpus, so $\hat{P}[\text{dog ran away}] = 0$

• Even worse: “dog ran” never appear in the training corpus, so $\hat{P}[\text{dog ran}] = 0$
Rectify: smoothing

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

• Back-off methods: restore to lower order statistics

• Example: if $\hat{P}[\text{away}|\text{dog ran}]$ does not work, use $\hat{P}[\text{away}|\text{ran}]$ as replacement

• Mixture methods: use a linear combination of $\hat{P}[\text{away}|\text{ran}]$ and $\hat{P}[\text{away}|\text{dog ran}]$
Drawback

• High dimension: # 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 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”

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

• 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:

\[ \nu_w := \hat{P}[w, :] \]
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

\[ \hat{P}[w, w'] \approx \text{row vector for the word } w \]
Variants

• low rank approx. of the transformed co-occurrence

\[ \text{Or } \text{PMI}(w, w') = \ln \frac{\hat{P}[w,w']}{\hat{P}[w]\hat{P}[w']} \]
State-of-the-art word embeddings

Updated on April 2016
Word2vec

- Continuous-Bag-Of-Words

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

\[ P[w_t|w_{t-2}, \ldots, w_{t+2}] \propto \exp[v_{w_t} \cdot \text{mean}(v_{w_{t-2}}, \ldots, v_{w_{t+2}})] \]
Linear structure for analogies

- Semantic: “man:woman::king:queen”

  \[ v_{\text{man}} - v_{\text{woman}} \approx v_{\text{king}} - v_{\text{queen}} \]

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

  \[ v_{\text{run}} - v_{\text{running}} \approx v_{\text{walk}} - v_{\text{walking}} \]
Country and Capital Vectors Projected by PCA
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 \(\bar{w}_i\)
• The GloVe objective function is

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

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