Deep Learning

Ali Ghodsi

University of Waterloo

Some slides courtesy of Richard Socher

Language Models

A language model computes a probability for a sequence of words: $P(w_1, \ldots, w_T)$

- Useful for machine translation
  - Word ordering:
    p (the cat is small) $>$ p (small the is cat)
  - Useful for machine translation Word choice:
    p (walking home after school) $>$ p (walking house after school)

'One-hot' representation

- Most Statistical NLP work regards words as atomic symbols: book, conference, university
- In vector space, this is a sparse vector with one 1 and a lot of zeros.

$$[0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

Problem:

1. Dimensionality of the vector will be the size of vocabulary. e.g. 13 M for Google 1T and 500K for big vocab.
2. $$book \quad [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

   $$library \quad [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]$$

Problem

Dimensionality of the vectors increase with vocabulary size.

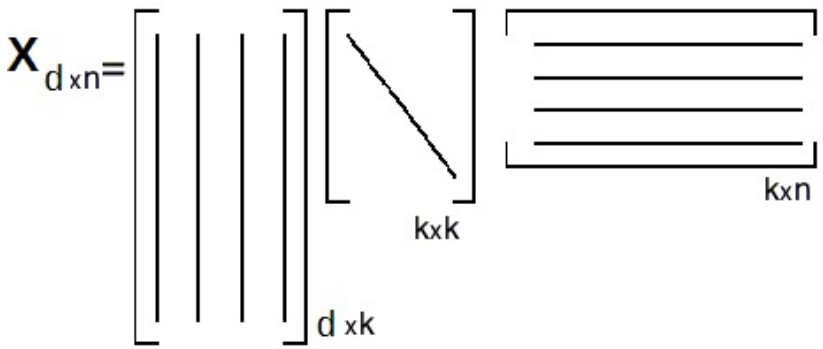
Vectors will be very high dimensional.

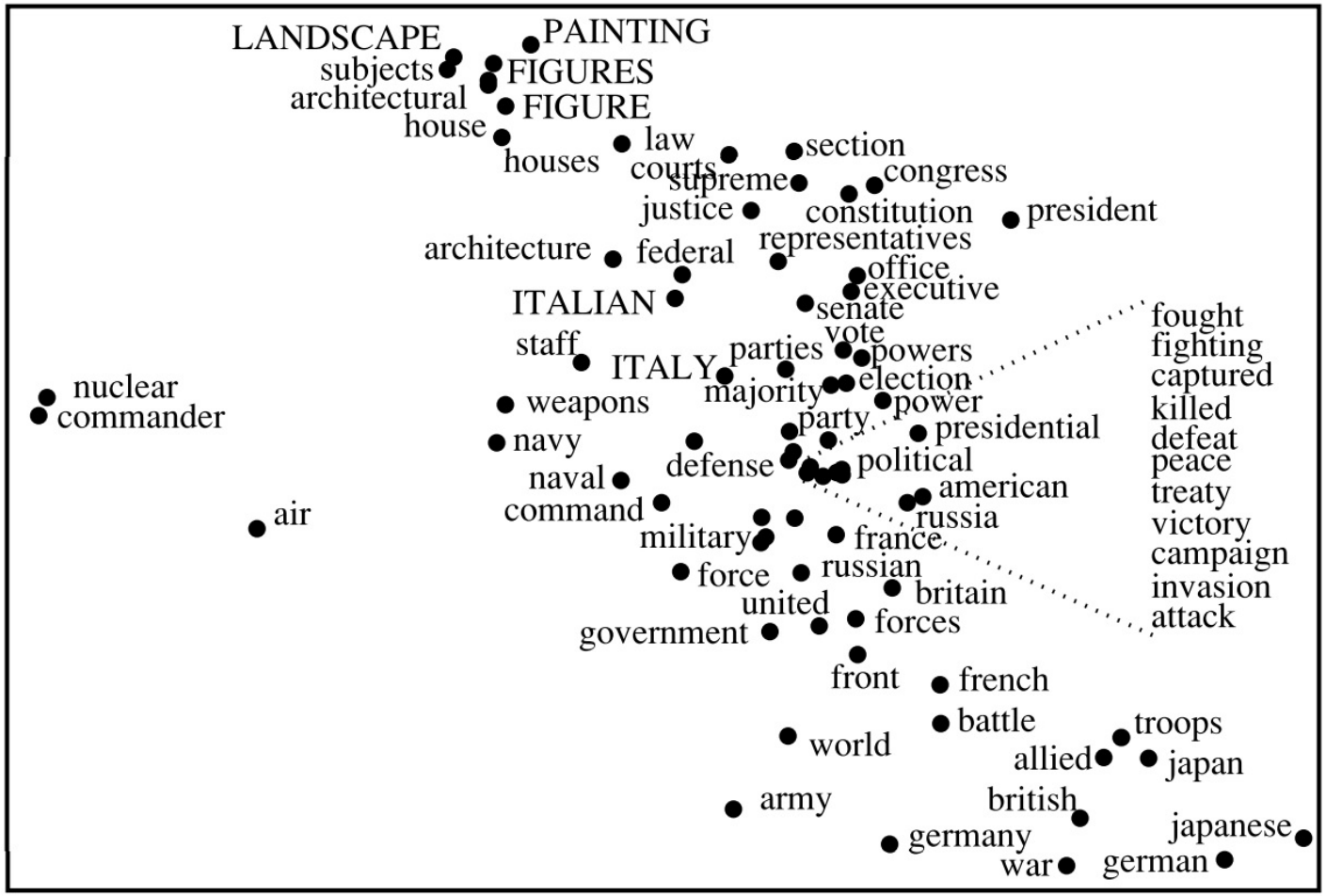
Idea 1: Reduce the dimensionality (Dimensionality reduction)

Singular Value Decomposition (SVD)

$$X = U\Sigma V^T$$

- The columns of U contain the eigenvectors of XX^T.
- The columns of V contain the eigenvectors of X^TX.
- Σ is diagonal matrix. Diagonal values are eigenvalues of XX^T or X^TX.





Nonlinear dimensionality reduction by locally linear embedding. Sam Roweis & Lawrence Saul. Science, v.290,2000

Figure: Arranging words in a continuous semantic space. Each word was initially represented by a high-dimensional vector that counted the number of times it appeared in different encyclopedia articles. LLE was applied to these word-document count vectors.

Computational Cost

Computational cost in SVD scales quadratically for $d \times n$ matrix: $O(nd^2)$ (when $d < n$)

This is not possible for large number of words or document.

It is hard for out of sample words or documents.

Idea 2: Learn low-dimensional vectors directly

- Learning representations by back-propagating errors.
  (Rumelhnrt, Hinton, Williams 1986)
- A Neural Probabilistic Language Model
  (Bengio et al., 2003)
- Natural Language Processing (Almost) from Scratch
  (Collobert et al., 2011)
- Efficient Estimation of Word Representations in Vector Space
  (Mikolov et al., 2013)
- GloVe: Global Vectors for Word Representation
  (Pennington et al., 2014)

Idea 2

Predict surrounding words of every word(word2rec)

Capture co occurrence counts directly (Glove)

They are fast and can incorporate a new sentence/document or add a word to the vocabulary.

Distributional Hypothesis

The *Distributional Hypothesis* in linguistics is derived from the semantic theory of language usage, i.e.

words that are used and occur in the same contexts tend to purport similar meanings.

Harris, Z. (1954). Distributional structure

'a word is characterized by the company it keeps'.

Firth, J.R. (1957). A synopsis of linguistic theory 1930-1955

Co-occurrence matrix

Co-occurrence can be interpreted as an indicator of semantic proximity of words.

Silence is the language of God, all else is poor translation.

Rumi (1207,1273)

counts	silence	is	the	language	of	God
silence	0	1	0	0	0	0
is	1	0	1	0	0	0
the	0	1	0	1	0	0
language	0	0	1	0	1	0
of	0	0	0	1	0	1
God	0	0	0	0	1	0

Vectors for Word Representation

word2vec and *GloVe* learn relationships between words.

The output are vectors with interesting relationships.

For example:

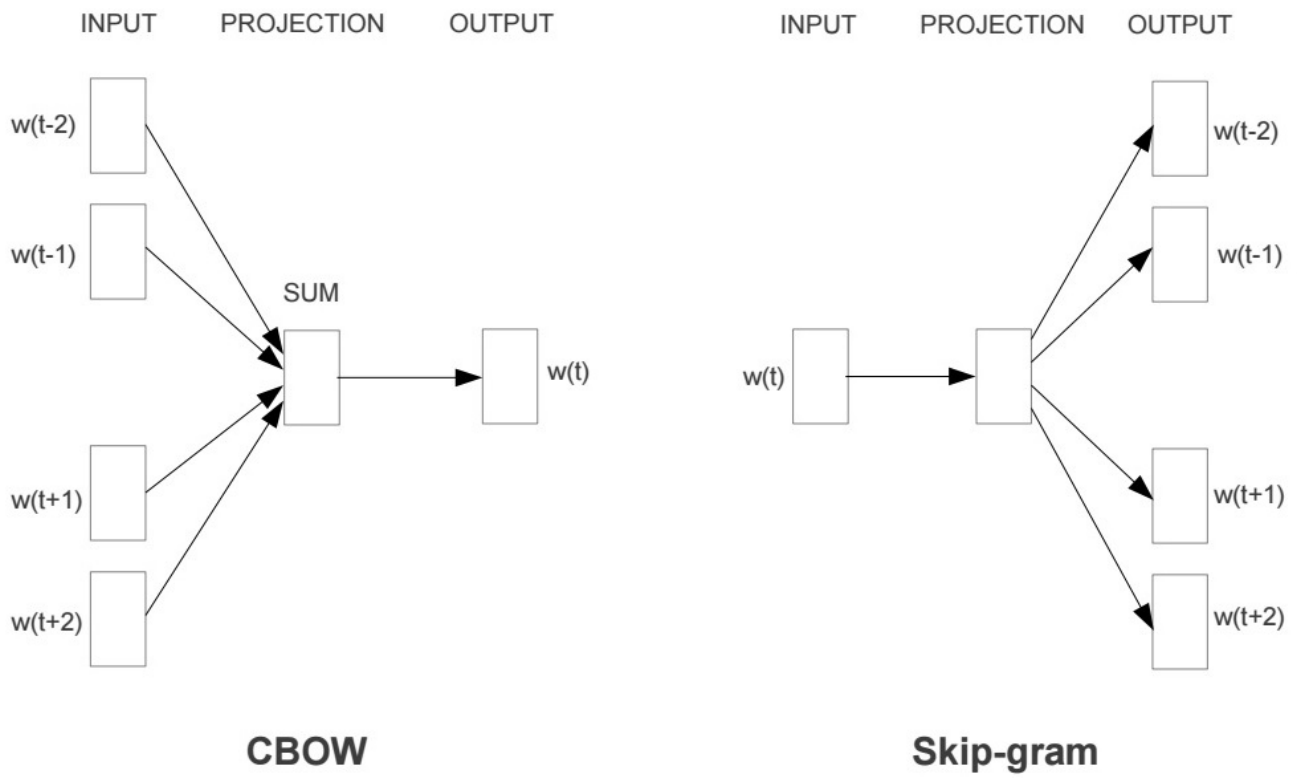
$$vec(king) – vec(man) + vec(woman) \approx vec(queen)$$

or

$$vec(Montreal\ Canadiens) – vec(Montreal) + vec(Toronto)$$

$$\approx vec(Toronto\ Maple\ Leafs)$$

Continuous bag-of-word and skip-gram

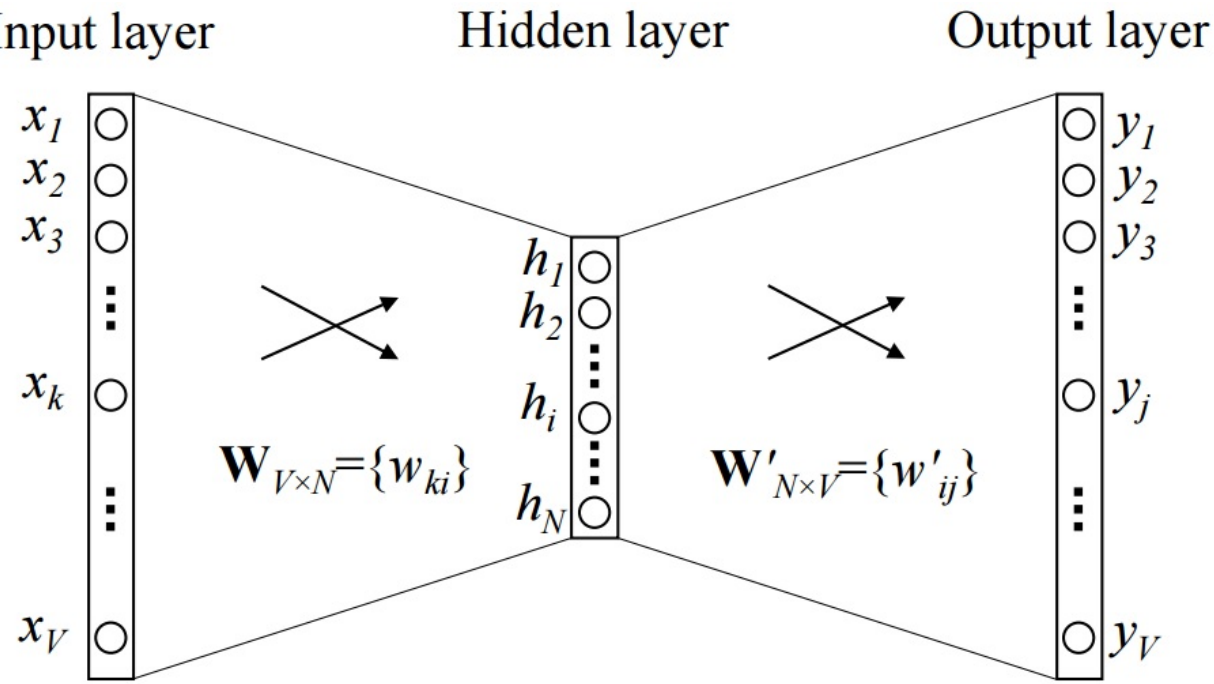


Efficient Estimation of Word Representations in Vector Space. Tomas Mikolov, et. al., arXiv 2013

Figure: The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

A very simple CBOW model

- Assume that $c = 1$. i.e. The length of the window is 1.
- Predict one target word, given one context word.
- This is like a bigram model.

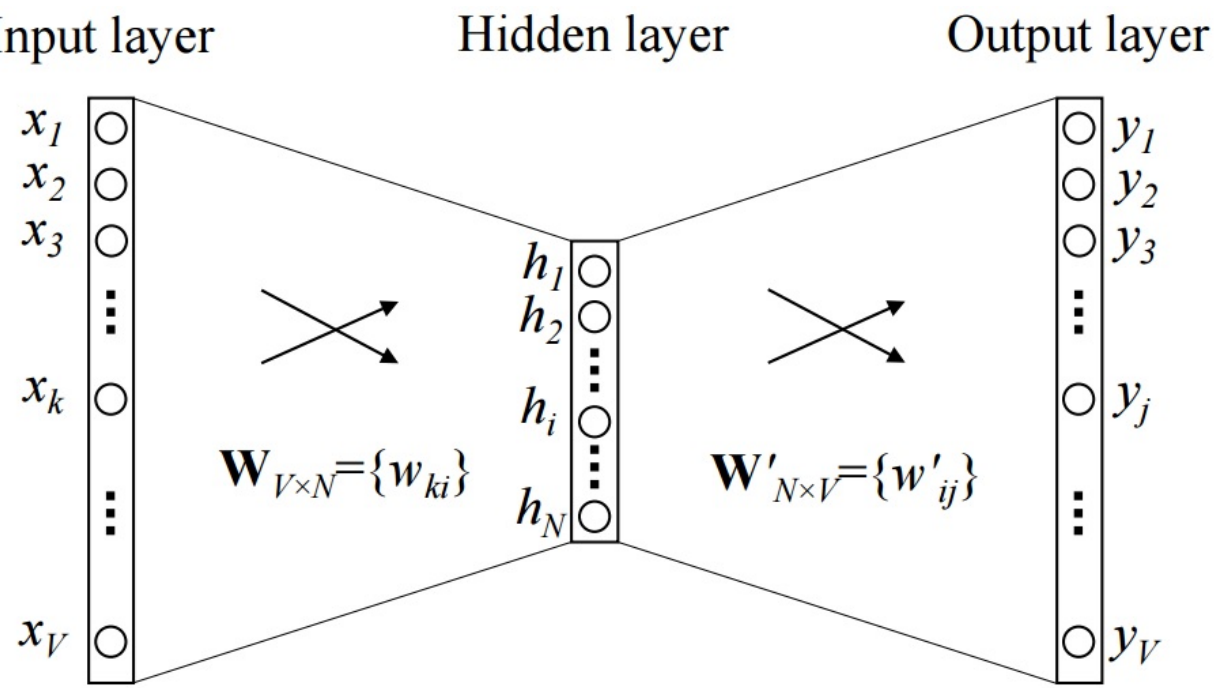


Credit: Xin Rong

Figure: A very simple CBOW model.

A very simple CBOW model

- Assume only $\mathbf{x}_k = 1$, i.e. $\mathbf{x}^T = [0\ 0\ 0\ \ldots\ 1\ \ldots\ 0\ 0]$
- $\mathbf{h} = \mathbf{x}^T W$ is the k-th row of W, $W_{k:}$.
- Let's define $\mathbf{u}_c = \mathbf{h}$ as the vector representation of the central (input) word.



Credit: Xin Rong

Details (Blackboard)

The skip-gram model and negative sampling

From paper: "Distributed Representations of Words and Phrases and their Compositionality" (Mikolov et al. 2013)

$$log\sigma({\nu'_{w0}}^T \nu_{wI}) + \sum_{i=1}^{k} \mathbb{E}_{wi} \sim P_n(w)[log\sigma({\nu'_{w0}}^T \nu_{wI})]$$

Where k is the number of negative samples and we use,

The sigmoid function! $\sigma(x) = \frac{1}{1+e^{-x}}$

So we maximize the probability of two words co-occurring in first log

The skip-gram model and negative sampling

$$log\sigma(\nu'_{w0}{}^{T}\nu_{wI}) + \sum_{i=1}^{k} \mathbb{E}_{wi} \sim P_n(w)[log\sigma(\nu'_{w0}{}^{T}\nu_{wI})]$$

Max. probability that real outside word appears, minimize prob. that random words appear around center word

$P_n = U(s)^{3/4}/Z$, the unigram distribution $U(w)$ raised to the 3/4rd power (We provide this function in the starter code).

The power makes less recent words be sampled more often

What to do with the two sets of vectors?

We end up with L and L' from all the vectors v and v'

Both capture similar co-occurrence information. It turns out, the best solution is to simply sum them up:

$$L_{\text{final}} = L + L'$$

One of many hyper parameters explored in GloVe: Global Vectors for Word Representation (Pennington et al. (2014)

PMI and PPMI

PMI of two words may produce negative value which is not interpretable. For instance, if two words w_i, w_j are never involved in a single document, we would have

$$\#(w_i, w_j) = 0, \quad \text{PMI}(w_i, w_j) = \log(0) = -\infty.$$

A convenient way to cope with the difficulty described above is to replace negative values by zeros, namely Positive PMI.

$$\text{PPMI}(w_i, w_j) = max(\text{PMI}(w_i, w_j), 0).$$

PMI and Word2vec

Levy and Goldberg (2013) proved the equivalence between *word2vec* and matrix factorization model.

Since

$$E_{w_n \sim P_n}[\log \sigma(-v_i \cdot u_n)] = \sum_n \frac{\#(w_n)}{T} \log \sigma(-v_i \cdot u_n),$$

This can be re-written as

$$J(w_i, w_j) = \#(w_i, w_j) \cdot \log \sigma(v_i \cdot u_j) + k \cdot \#(w_i) \cdot \frac{\#(w_j)}{T} \log \sigma(-v_i \cdot u_j).$$

PMI and Word2vec

Let $x = v_i \cdot u_j$, and optimize with respect to x

$$\frac{\partial J(w_i, w_j)}{\partial x} = 0,$$

$$e^{2x} - \left(\frac{\#(w_i, w_j)}{k \cdot \#(w_i) \cdot \frac{\#(w_j)}{T}} - 1 \right) e^x - \frac{\#(w_i, w_j)}{k \cdot \#(w_i) \cdot \frac{\#(w_j)}{T}} = 0. \qquad (1)$$

Quadratic with respect to e^x

PMI and Word2vec

Two solutions

One is not valid (i.e. $e^x = -1$)

$$e^x = \frac{\#(w_i, w_j) \cdot T}{\#(w_i)\#(w_j)} \cdot \frac{1}{k}$$

since $x = v_i \cdot u_j$,

$$v_i \cdot u_j = \log(\frac{\#(w_i, w_j) \cdot T}{\#(w_i)\#(w_j)}) - \log k$$

Skip-gram is equivalent to a matrix factorization model with the matrix entry being $\log(\frac{\#(w_i,w_j)\cdot T}{\#(w_i)\#(w_j)}) - \log k$.