Deep Learning - Embedding
深度学习 - 嵌入

Jian Li
IIIS, Tsinghua
Word2Vec
“You shall know a word by the company it keeps”

(J. R. Firth 1957: 11)

Build a cooccurrence matrix (using a moving window)

- I like deep learning.
- I like NLP.
- I enjoy flying.

<table>
<thead>
<tr>
<th>counts</th>
<th>I</th>
<th>like</th>
<th>enjoy</th>
<th>deep</th>
<th>learning</th>
<th>NLP</th>
<th>flying</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>like</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>enjoy</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>deep</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>learning</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NLP</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>flying</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- high dimensional, and sparse
- we want relatively low-dim representation
Word Embedding

- Map each word to a vector (in relatively low-dim space)

\[
\begin{align*}
    x_{apple} - x_{apples} & \approx x_{car} - x_{cars} \approx x_{family} - x_{families} \\
    x_{shirt} - x_{clothing} & \approx x_{chair} - x_{furniture} \\
    x_{king} - x_{man} & \approx x_{queen} - x_{woman}
\end{align*}
\]
**Approach 1: SVD/PCA**

- Some notes: remove "of" "the" (syntactic words)

- Use Pearson correlations (rather than count) $\rho_{xy} = \frac{\text{Cov}(x,y)}{\delta_x \delta_y} = \frac{\text{EI}(x-x)\text{EI}(y-y)}{\text{Var}(x)\text{Var}(y)}$

- Set negative values to zero

**Problem:** SVD expensive for large vocabulary

Special algorithm (doesn't fit into DL pipeline)
Word2Vec

Word 2 Vec (Mikolov et al. 2013)

Predict surrounding words in a window (of len m) of every word

First define a probabilistic model

Each word w has two vectors $\{u_w, v_w\}$ (the embedding of w when w is outside, center)

We want to learn $u_w, v_w$ for $w \in W$

$$P(o|c) = \frac{\exp \langle u_o, v_c \rangle}{\sum_{w=1}^W \exp \langle u_w, v_c \rangle} \in \text{softmax/logistic regression}$$

If $u_o, v_c$ in the same direction, $P(o|c)$ is large.
Objective of Word2Vec

\[ J = \frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log P(w_{t+j} | w_t) \]

- Sum over all word
- \( \log \) Prob of surrounding word of \( w_t \)

Diagram:
- \( w_{t-m} \) to \( w_t \) to \( w_{t+m} \)
Training: $\log P(w_w | w_c)$ is expensive ($O(W)$ time)

1 Hierarchical Softmax (it is an approximation)

$h(w_j)$: $j$th node on the root $\rightarrow w$ path

$l(w)$: length of root $\rightarrow w$ path

for every inner node, we also have a vector $U^{'}_{h_i(w_j)}$

for every leaf word $w$, only one vector $v_w$

$$p(w | w_c) = \prod_{j=0}^{l-1} \delta \left( \frac{l(w)-1}{j} \right) \delta \left( h(w_{j+1}) = \text{ch}(h(w_j)) \right) \langle U^{'}_{h_i(w_j)}, v_{w_c} \rangle$$

$$\delta(x) = \frac{1}{1 + e^{-x}}$$

$$\sum_{w=1}^{L} p(w | w_c) = 1 \quad (\text{easy to check})$$

- not necessarily a full binary tree. e.g., the paper uses Huffman tree
- time: len of the path. (frequent word $\Rightarrow$ shorter path)
Training method 2

Negative Sampling.

**Noise Contrastive Estimation (NCE) (Gutmann & Hyvarinen)**

A good model should be able to differentiate data from noise (by logistic regression).

**Negative sampling objective**

\[
\log \delta(\langle w_0, v_w \rangle) + \sum_{i=1}^{K} \log \delta(-\langle w_i, v_w \rangle)
\]

Replace by \( P(w_0|v_w) \) by

\[
\log \delta(\langle w_0, v_w \rangle) + \sum_{i=1}^{K} \log \delta(-\langle w_i, v_w \rangle)
\]

\( w_i \sim P(w) \sim \text{noise distributions} \)

- Uniform distr
- Unigram distr \( U(w) \) (i.e., word freqNCY)
- \( U(w)^{2/3} \)

(\( K \sim 5-20 \) typically. If data is large, \( K \sim 2-5 \))
\[
\begin{align*}
\text{apple} - \text{apples} & \approx \text{car} - \text{cars} \approx \text{family} - \text{families} \\
\text{shirt} - \text{clothing} & \approx \text{chair} - \text{furniture} \\
\text{king} - \text{man} & \approx \text{queen} - \text{woman}
\end{align*}
\]

\[
d = \arg \max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}
\]

\(W_x\) produces the largest \(\langle x, W_x \rangle\)

direction of \(w_b - w_a + w_c\)
• **Word2Vec**: unsupervised learning
  • Huge amount of training data (no label is needed)

• **Can be incorporated to deep learning pipeline**
  • The corresponding layers is usually called the **embedding layer**
  • The resulting vectors obtained from word2vec can be used to initialize the parameters of NN
  • We can also get the embedding from training a specified DNN (for a specific task)
    • Computational complexity too high (much higher than word2vec)
    • The embedding may not be useful in other tasks
    • On the other hand, word2vec captures a lot of semantic information, which is useful in a variety of tasks

According to Mikolov:
**CBOW** (Continuous Bag of Words): Use context to predict the current word.
--several times faster to train than the skip-gram, slightly better accuracy for the frequent words

**Skip-gram**: Use the current word to predict the context.
--works well with small amount of the training data, represents well even rare words or phrases.
GloVe

GloVe: Global Vectors for Word Representation
Two methodologies

1. Global matrix factorization
2. Local context window (skip-gram)

predicting the context given a word
Ratio Matters

Co-occurrence matrix: 

\[ X = \begin{bmatrix} X_{ij} \end{bmatrix} \] 

\# times word \( j \) occurs in the context of word \( i \)

\[ P_{ij} = P(j|i) = \frac{X_{ij}}{X_i} = \frac{X_{ij}}{\sum_j X_{ij}} \]

<table>
<thead>
<tr>
<th>Probability and Ratio</th>
<th>( k = solid )</th>
<th>( k = gas )</th>
<th>( k = water )</th>
<th>( k = fashion )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(k</td>
<td>ice) )</td>
<td>( 1.9 \times 10^{-4} )</td>
<td>( 6.6 \times 10^{-5} )</td>
<td>( 3.0 \times 10^{-3} )</td>
</tr>
<tr>
<td>( P(k</td>
<td>steam) )</td>
<td>( 2.2 \times 10^{-5} )</td>
<td>( 7.8 \times 10^{-4} )</td>
<td>( 2.2 \times 10^{-3} )</td>
</tr>
<tr>
<td>( P(k</td>
<td>ice)/P(k</td>
<td>steam) )</td>
<td>( 8.9 )</td>
<td>( 8.5 \times 10^{-2} )</td>
</tr>
</tbody>
</table>

- Solid is more relevant to ice than to steam.
- Gas is less relevant to ice than to steam.
Derivation

Define $F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$

Using the property of $F$, we try to derive the form of $<w_i, \tilde{w}_k>$

Want $F(w_i, w_j, \tilde{w}_k) = F(<w_i-w_j, \tilde{w}_k>)$

enforce a linear structure

2. Imagine we switch the role of a word & a context
   i.e., $X \rightarrow X^T$
   then we’d better have $w_i \rightarrow \tilde{w}_i$ (symmetric)

So we choose $F = \exp$

$(\exp (\langle w_i-w_j, \tilde{w}_k \rangle) = \frac{\exp \langle w_i, \tilde{w}_k \rangle}{\exp \langle w_j, w_k \rangle} = \frac{P_{ik}}{P_{jk}} \frac{X_iy_i}{X_jy_j}$

$\Rightarrow$ $\langle w_i, \tilde{w}_k \rangle + b_i + \tilde{b}_k = \log (X_{ik})$

$\log \frac{X_iy_i}{X_jy_j}$

(verify $\langle w_i-w_j, \tilde{w}_k \rangle = \log \left( \frac{X_{ik}}{X_{jk}} \right) - (b_j - b_i)$)

We try to factorize $\log X$ (i.e., $\log \frac{X_iy_i}{X_jy_j} = \log (\frac{X_i}{X_j})$)
Objective (weighted least square)

\[ J(\Theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(X_{ij}) \left( \langle w_i, \tilde{w}_j \rangle + b_i + \tilde{b}_j - \log X_{ij} \right)^2 \]

\[ f(x) = \begin{cases} \left( \frac{x}{X_{\text{max}}} \right)^\alpha & x < X_{\text{max}} \\ 1 & 0 \leq x \leq X_{\text{max}} \end{cases} \]

\[ X_{\text{max}} = 100 \]

\[ (\alpha = 3/4) \]

Training: SGD, Adagrad
Glove results

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus
Reference

- Baroni, Marco, Georgiana Dinu, and Germán Kruszewski. Don’t count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. ACL 14
  - An early paper claims that the prediction formulation (like word2vec) is better than factorizing a co-occurrence matrix

  (comparing several SNSG, GloVe, and SVD)

A very reasonable blog discussing the relations between different models
http://sebastianruder.com/secret-word2vec/index.html


Deep Walk
-embedding node in a social network
Embedding node (using pairwise relations)
Key Idea

treat vertex as words, random walks as sentences
Algorithm 1 DeepWalk($G, w, d, \gamma, t$)

Input: graph $G(V, E)$
- window size $w$
- embedding size $d$
- walks per vertex $\gamma$
- walk length $t$

Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

1: Initialization: Sample $\Phi$ from $U^{|V| \times d}$
2: Build a binary Tree $T$ from $V$
3: for $i = 0$ to $\gamma$
   4: $\mathcal{O} = \text{Shuffle}(V)$
   5: for each $v_i \in \mathcal{O}$
      6: $\mathcal{W}_{v_i} = \text{RandomWalk}(G, v_i, t)$
      7: SkipGram($\Phi$, $\mathcal{W}_{v_i}$, $w$)
   8: end for
9: end for

Algorithm 2 SkipGram($\Phi$, $\mathcal{W}_{v_i}$, $w$)

1: for each $v_j \in \mathcal{W}_{v_i}$
   2: for each $u_k \in \mathcal{W}_{v_j}[j - w : j + w]$
      3: $J(\Phi) = -\log \Pr(u_k | \Phi(v_j))$
      4: $\Phi = \Phi - \alpha \cdot \frac{\partial J}{\partial \Phi}$
   5: end for
6: end for

Use Hierarchical Softmax to approximate.
Multimodal representation learning
---Image Caption 2

Kires et al. Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models
there is a cat sitting on a shelf.
an article with a fork and a piece of cake.
a black and white photo of a window.
a young boy standing on a parking lot next to cars.
a wooden table and chairs arranged in a room.
a kitchen with stainless steel appliances.
this is a herd of cattle out in the field.
a car is parked in the middle of nowhere.
a ferry boat on a marina with a group of people.
a little boy with a bunch of friends on the street.
a giraffe is standing next to a fence in a field. (hallucination)
the two birds are trying to be seen in the water. (counting)
a parked car while driving down the road. (contradiction)
the handlebars are trying to ride a bike rack. (nonsensical)
a woman and a bottle of wine in a garden. (gender)

Figure 1: Sample generated captions. The bottom row shows different error cases. Additional results can be found at [http://www.cs.toronto.edu/~rkiros/lstm_schlm.html](http://www.cs.toronto.edu/~rkiros/lstm_schlm.html)
Overview

Map CNN codes and RNN code to a common space

Details of SC-NLM. Please see the paper

Figure 2: **Encoder**: A deep convolutional network (CNN) and long short-term memory recurrent network (LSTM) for learning a joint image-sentence embedding. **Decoder**: A new neural language model that combines structure and content vectors for generating words one at a time in sequence.
Figure 4: Multimodal vector space arithmetic. Query images were downloaded online and retrieved images are from the SBU dataset.
\[ \mathbf{v}_{\text{car}} \approx \mathbf{I}_{\text{bcar}} - \mathbf{v}_{\text{blue}} \]
\[ \mathbf{v}_{\text{red}} + \mathbf{v}_{\text{car}} \approx \mathbf{I}_{\text{bcar}} - \mathbf{v}_{\text{blue}} + \mathbf{v}_{\text{red}} \]
\[ \mathbf{I}_{\text{rcar}} \approx \mathbf{I}_{\text{bcar}} - \mathbf{v}_{\text{blue}} + \mathbf{v}_{\text{red}} \]

Figure 5: PCA projection of the 300-dimensional word and image representations for (a) cars and colors and (b) weather and temperature.
Details

• LSTM notations used in this work

Let $X_t$ denote a matrix of training instances at time $t$. In our case, $X_t$ is used to denote a matrix of word representations for the $t$-th word of each sentence in the training batch. Let $(I_t, F_t, C_t, O_t, M_t)$ denote the input, forget, cell, output and hidden states of the LSTM at time step $t$. The LSTM architecture in this work is implemented using the following equations:

\[
\begin{align*}
I_t & = \sigma(X_t \cdot W_{xi} + M_{t-1} \cdot W_{hi} + C_{t-1} \cdot W_{ci} + b_i) \\
F_t & = \sigma(X_t \cdot W_{xf} + M_{t-1} \cdot W_{hf} + C_{t-1} \cdot W_{cf} + b_f) \\
C_t & = F_t \cdot C_{t-1} + I_t \cdot tanh(X_t \cdot W_{xc} + M_{t-1} \cdot W_{hc} + b_c) \\
O_t & = \sigma(X_t \cdot W_{xo} + M_{t-1} \cdot W_{ho} + C_t \cdot W_{co} + b_o) \\
M_t & = O_t \cdot tanh(C_t)
\end{align*}
\]

where $(\sigma)$ denotes the sigmoid activation function, $(\cdot)$ indicates matrix multiplication and $(\cdot \cdot)$ indicates component-wise multiplication.
Details

Let \( q \in \mathbb{R}^D \) denote an image feature vector

- \( D \): length of the CNN code (CNN can be AlexNet, VggNet, or ResNet)

\[ x = W_I \cdot q \in \mathbb{R}^K \] be the image embedding

image description \( S = \{w_1, \ldots, w_N\} \) with words \( w_1, \ldots, w_N \)

\( \{w_1, \ldots, w_N\}, w_i \in \mathbb{R}^K, i = 1, \ldots, n \) denote the corresponding word representations to words \( w_1, \ldots, w_N \) (entries in the matrix \( W_T \)). The representation of a sentence \( v \) is the hidden state of the LSTM at time step \( N \) (i.e. the vector \( m_t \)).

\( W_T \): precomputed using e.g. word2vec
Details

- Optimize pairwise rank loss ($\theta$: parameters needed to be learnt: $\mathcal{W}_I$ and LSTM parameters) \(\text{ (similar to negative sampling in spirit)}\)

$$\min_\theta \sum_x \sum_k \max\{0, \alpha - s(x, v) + s(x, v_k)\} + \sum_v \sum_k \max\{0, \alpha - s(v, x) + s(v, x_k)\}$$

Max-margin formulation. $\alpha$ margin
We have nonzero loss if $s(x,v)$ is less than $s(x,v_k)+\alpha$

scoring function $s(x, v) = x \cdot v$.

$v_k$ is a contrastive (non-descriptive) sentence for image embedding $x$, and vice-versa with $x_k$. 