Word Embeddings & Language Modeling

Lili Mou
lmou@ualberta.ca
lili-mou.github.io
Last Lecture

- Logistic regression/Softmax: Linear classification

- Non-linear classification
  - Non-linear feature engineering
  - Non-linear kernel
  - Non-linear function composition

- Neural networks
  - Forward propagation: Compute activation
  - Backward propagation: Compute derivative
    (greedy dynamic programming)
Advantages of DL

- Work with raw data
  - Images processing: pixels
  - Speech processing: frequency

[Graves+, ICASSP'13]
How about Language?

- The raw input of language

\[ I \ like \ the \ course \]

- Problem: *Words are discrete tokens!*
Representing Words

• **Attempt#1:**
  - By index in the vocabulary

• **Problem**
  - Introducing artefacts
    • Order, metric, inner-product
    • Extreme non-linearity

\[
\mathcal{V} = \{0, 1, 2, 3\}
\]
Representing Words

- **Attempt#2**: One-hot representation

  - Separability doesn't generalize
  - Metric is trivial

\[ V = \begin{cases} 0 & \star \\ 1 & \bigstar \\ 2 & \square \\ 3 & \bigcirc \end{cases} \]

\[
\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}
\]
Design a **metric** $d(\cdot, \cdot)$ to evaluate the “distance” of two words in terms of some aspect

- E.g., semantic similarity

*I’d like to have some pop/soda/water/fruit/rest*

- Traditional method: WordNet distance (if it’s a metric).

If not, doesn’t matter.
Metric in the Word Space

- Design a metric $d(\cdot, \cdot)$ to evaluate the “distance” of two words in terms of some aspect
  - E.g., semantic similarity

  I’d like to have some pop/soda/water/fruit/rest

- A straightforward metric on one-hot vector:
  - Discrete metric

  $d(x_i, x_j) = 1$ if $x_i = x_j$, 0 otherwise
# ID and One-Hot

<table>
<thead>
<tr>
<th>ID representation</th>
<th>One-hot representation</th>
</tr>
</thead>
</table>
| 1 3 2 0           | \[
|                  | \begin{bmatrix}
|                  | 0 \\
|                  | 1 \\
|                  | 0 \\
|                  | 0 \\
|                  | 0 \\
|                  | \end{bmatrix} |

<table>
<thead>
<tr>
<th>Dimension</th>
<th>One-dimensional</th>
<th>-dimensional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>Artefact</td>
<td>Non-informative</td>
</tr>
<tr>
<td>Discrete</td>
<td>Non-informative</td>
<td>Non-informative</td>
</tr>
<tr>
<td>Learnable</td>
<td>Difficult</td>
<td>Possible but may not generalize Need to explore more</td>
</tr>
</tbody>
</table>

- Need to explore more
**Something in Between**

- Map a word to a low-dimensional space
  - Not as low as one-dimensional ID representation
  - Not as high as $|\mathcal{V}|$-dimensional one-hot representation

**Attempt#3:** Word vector representation (a.k.a., word embeddings)

- Mapping a word to a vector
- Equivalent to linear transformation of one-hot vector

\[ \begin{pmatrix} 0 \\ 1 \\ \cdot \\ \cdot \\ 0 \end{pmatrix} \text{ embedding of word } i \]

\[ \cdot \begin{pmatrix} 0 \\ 1 \\ \cdot \\ \cdot \\ 0 \end{pmatrix} \text{ one-hot representation of word } i \text{ (sparse)} \]
Obtaining the Embedding Matrix

- **Attempt#1**: Treat as neural weights as usual
  - Random initialization & gradient descent
- Properties of the embedding matrix
  - Huge, \(|V| \times d_{NN}\) parameters (cf. weight for layerwise MLP)
  - Sparsely updated
- Nature of language
  - Power law distribution
- Good if corpus is large
Embedding Learning

- **Attempt #2:**
  - Manually specifying the distance metric/inner-product, etc.
  - Humans are not rational

- **Attempt #3:**
  - Pre-training on a massive corpus with a different (pre-training) objective
  - Then, we can fine-tune those pre-trained embeddings in almost any specific task.
Pretraining Criterion

• Language Modeling
  - Given a corpus $\mathbf{x} = x_1x_2\cdots x_t$
  - Goal: Maximize $p(\mathbf{x})$

• Is it meaningful to view language sentences as a random variable?
  - Frequentist: Sentences are repetitions of i.i.d. experiments
  - Bayesian: Everything unknown is a random variable
Factorization

- \( p(\mathbf{x}) = p(x_1, \ldots, x_t) \) cannot be parametrized

- Factorizing a giant probability

\[
p(\mathbf{x}) = p(x_1, \ldots, x_t) = p(x_1)p(x_2 | x_1) \cdots p(x_t | x_1, \ldots, x_{t-1})
\]

- Still unable to parametrize, especially \( p(x_n | x_1, \ldots, x_{n-1}) \)

- **Questions:**
  - Can we decompose any probabilistic distribution defined on \( \mathbf{x} \) into this form? Yes.
  - Is it necessary to decompose the distribution a probabilistic distribution in this form? No.
Markov Assumptions

$$p(x) = p(x_1, \ldots, x_t)$$

$$= p(x_1)p(x_2 | x_1)\cdots p(x_t | x_1, \ldots, x_{t-1})$$

- Independency
  - Given the current “state,” independent with previous ones
  - State at step $t$: $(x_{t-n+1}, x_{t-n+2}, \ldots, x_{t-1})$
  - $x_t \perp x_{\leq t-n} | x_{t-n+1}, x_{t-n+2}, \ldots, x_{t-1}$

- Stationary property
  - $p(x_t | x_{t-1}, \ldots, x_{t-n+1}) = p(x_s | x_{s-n+1}, \ldots, x_{s-1})$ for all $t, s$
Parametrizing $p(w)$

$p(x) = p(x_1, \ldots, x_t)\\ = p(x_1)p(x_2 \mid x_1)\cdots p(x_t \mid x_1, \ldots, x_{n-1})\\ \approx p(x_1)p(x_2 \mid x_1)\cdots p(x_n \mid x_1, \ldots, x_{t-n+1})$

Direct parametrization:

Each multinomial distribution is directly parametrized

$p(w_n \mid w_1, \ldots, w_{n-1})$  (notation abuse)
N-gram Model

\[ p(x) = p(x_1, \ldots, x_n) \]
\[ = p(x_1)p(x_2 | x_1)\cdots p(x_n | x_1, \ldots, x_{n-1}) \]
\[ \approx p(x_1)p(x_2 | x_1)\cdots p(x_n | x_1, \ldots, x_{t-n+1}) \]

\[ \hat{p}(w_n | w_1, \ldots, w_{n-1}) = \frac{\#w_1\cdots w_n}{\#w_1\cdots w_{n-1}} \]

Questions:

• How many multinomial distributions?

• How many parameters in total?
Problems of n-gram models

- #para $\propto \exp(n)$
- Power-law distribution
  - Severe data sparsity even if $n$ is small

- Normal distribution
  \[ p(x) \propto \exp(-\tau x^2) \]
- Power-law distribution
  \[ p(x) \propto x^{-k} \]
Smoothing Techniques

- Add-one smoothing
- Interpolation smoothing
- Backoff smoothing

Useful link: https://nlp.stanford.edu/~wcmac/papers/20050421-smoothing-tutorial.pdf
Is it possible to parametrize LM by NN?

Yes

- \( p(w_n | w_1, \ldots, w_{n-1}) \) is a classification problem
- NNs are good at (esp. non-linear) classification
Feed-Forward Language Model

\[ i\text{-th output} = P(w_t = i \mid \text{context}) \]

**N.B.** The Markov assumption also holds.


By product: Embeddings are pre-trained in a meaningful way
Recurrent Neural Language Model

- RNN keeps one or a few hidden states
- The hidden states change at each time step according to the input

\[ h_t = \text{RNN}(x_t, h_{t-1}) = f(W_{\text{in}} x_t + W_{\text{hid}} h_{t-1}) \]

\[ p(w_t | w_{0}^{t-1}) \approx \text{softmax} (W_{\text{out}} h_t) \]

- RNN directly parametrizes rather than \( p(w) \approx \prod_{t=1}^{m} p(w_t | w_{1}^{t-1}) \)

How can we use word embeddings?

- Embeddings demonstrate the internal structures of words
  - Relation represented by vector offset
    - “man” – “woman” = “king” – “queen” [Mikolov+NAACL13]
  - Word similarity

- Embeddings can serve as the initialization of almost every supervised task
  - A way of pretraining
  - **N.B.**: may not be useful when the training set is large enough
Word Embeddings in our Brain

“Somatotopic Embeddings” in our Brain

Complexity Concerns

● Time complexity
  - Hierarchical softmax [1]
  - Negative sampling: Hinge loss [2], Noisy contrastive estimation [3]

● Memory complexity
  - Compressing LM [4]

● Model complexity
  - Shallow neural networks are still too “deep.”
  - CBOW, SkipGram [3]

Deep neural networks:
To be, or not to be? That is the question.
CBOW, SkipGram (word2vec)

Hierarchical Softmax and Negative Contrastive Estimation

- HS

\[ p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma \left( [n(w, j+1) = \text{ch}(n(w, j))] \cdot v'_{n(w, j)}^T v_{w_I} \right) \]

- NCE

\[ \log \sigma(v'_{w_O}^T v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[ \log \sigma(-v'_{w_i}^T v_{w_I}) \right] \]

Tricks in Training Word Embeddings

- The # of negative samples?
  - The more, the better.

- The distribution from which negative samples are generated? Should negative samples be close to positive samples?
  - The closer, the better.

- Full softmax vs. NCE vs. HS vs. hinge loss?
Recent Advances in Pretraining

• Pretraining the embedding mapping for words is not enough
  - E: Vocabulary $\rightarrow \mathbb{R}^n$

• Context info?
  - Why not pre-train follow-up layers as well?
  - E.g., ELMo, BERT

  - Represent a word in a context, with LM-like pretraining

  - Factorization of $p(w) = p(w_1)p(w_2 | w_1)\cdots p(w_n | w_1\cdots w_{n-1})$ is unnecessary
Learning Embeddings of Other Stuff

• Node embeddings of a network

• General criteria of embedding learning
  - Atomic token represented by an embedding
  - Training embeddings by predicting “context”
Representing Words

One-hot

Real-valued embedding

Index

One pretraining method

Language modeling
  - Max Pr(corpus)

N-gram
  - Markov assumption
  - MLE = counting %
  - Sparsity
    - Para $\propto \exp(n)$
    - Power law dist.

Embeddings in general
  - Discrete token -> vector
  - Learned by predicting context

[ E ] *

NN-LM
  - Predict the next word
  - Embeddings pretrained
  - Recent advance: Pretrain LM
Suggested Reading


More References