

Deep Learning for Natural Language Processing (cs224d) Lecture Notes

Yihong Gu^{*}

Contents

1	NLP and DL: introduction	2
1.1	Natural Language Processing	2
1.2	Deep Learning	2
2	Word Vector	3
2.1	Statistical Based Method: SVD	3
2.2	Iteration Based Methods: word2vec	4
2.3	Evaluation: Intrinsic and Extrinsic	7
3	Neural Networks	7
3.1	Neuron, Terminology, Forward Propagation	7
3.2	Back Propagation, Computational Graph	7
3.3	Tips and Tricks	7
4	Recurrent Neural Networks	7
5	Recursive Neural Networks	7
6		7

^{*}gyh15@mails.tsinghua.edu.cn

1 NLP and DL: introduction

1.1 Natural Language Processing

Hierarchical Stages

首先，需要了解关于 Natural Language Processing 的几个层次：

- **Phonetic Analysis/OCR** 阶段：主要解决的是自然语言的输入端的问题，其中 **Phonetic** 为语音、**OCR** 为文本。
- **Morphological Analysis**: 主要解决词的合成的问题，比如词的前缀和后缀的意义 (e.g. un-)。
- **Syntactic Analysis**: 主要解决句法合成的问题，研究词如何通过语法结构进行组合。
- **Semantic Interpretation**: 主要解决句子含义 (meaning) 的问题。
- **Discourse Processing**: 主要解决整篇文章的理解的问题。

Difficulty

传统的 NLP 的困难之处主要在以下两个方面，第一个方面是知识 (knowledge) 的表示，以下句子为例

Jane hit June and then **she** [fell/ran]

显然，fell 和 ran 分别让 she 有了不同的指代 (Jane/June)，如果要让机器能够准确作出正确的指代关系的话，仅仅考虑单词的意思和语法结构是不够的，机器需要理解 hit 和 fell/ran 之间的逻辑关系，这需要进一步的先验知识。

第二个方面是模糊性 (ambiguity)，即多重含义。

1.2 Deep Learning

Representation Learning

我们考虑一个传统的 (预测类) 机器学习过程，我们把它分成两部分

$$\text{Machine Learning} = \text{Feature Engineering} + \text{Learning Algorithm}$$

在这样的过程中，我们拿到数据 (data)，先要人工设计数据的特征表示 (representation)，然后再使用特定的机器学习算法。通常而言，特征表示在整个过程的地位更高并且需要花费更多的精力。

而 Deep Learning 是什么？我们可以把它看成一种特征学习 (representation learning)，它能够从大量的数据中自动地学习特征。在具体实践中，Deep Learning 更加灵活，适用性也更强，一个适用于某个问题的 DL 模型可以直接运用于另外一个问题上。

Visualization

值得注意的是，可视化 (**visualization**) 在 DL 中非常重要，我们通常会 (使用 PCA 等方法) 把我们从 DL 中学到的 feature 投影到二维空间作可视化，我们可以从中发现一些非常有趣的性质：对于 NLP 问题来说，一个非常广泛的性质在于意思相近的单词/短语/句子在这样的欧几里得空间中也靠得更近。

DL+NLP

我们考虑 DL 在 NLP 上的应用，其核心在于把单词/短语/句子映射成 d 维空间中的向量，进而进行进一步的分析。

2 Word Vector

Word Vector 的意义在于把单词映射成 d 维空间中的向量。关于 word vector(以及之后的 NLP), 我们有一些相关的术语:

- **corpus**: 指用来作训练的文本全集。
- **token**: 文本中的某个单一的元素, 可以是一个单词, 也可以是一个标点, 也可以是开始符 / 结束符。

传统的 NLP 中, 关于单词有三种表示方法:

- **one-hot**: 单词的词向量 $\in \mathbb{R}^{|V|}$, 其中 $|V|$ 为单词总数。
- **taxonomy**: 建立词与词之间的从属关系 (is-a), 比如 flower 从属于 plant (flower **is-a** plant), 相关的工作有 WordNet。
- **synonym set**: 建立同义词/近义词集合。

2.1 Statistical Based Method: SVD

Model

首先, 我们通过文本构造 **Window based Co-occurrence Matrix**, 考虑构造这样一个矩阵 $X \in \mathbb{M}_{|V| \times |V|}$, 其 **window size** 为 W , 那么, 如果单词 w_j 在单词 w_i 的大小为 W 的 window 中出现, $X_{i,j}$ 就累加 1, 下面是一个例子

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

其对应的 co-occurrence matrix X 为:

$$X = \begin{matrix} & \begin{matrix} I & like & enjoy & deep & learning & NLP & flying & . \end{matrix} \\ \begin{matrix} I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{matrix} & \begin{pmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \end{matrix} \quad (1)$$

注意到这里的矩阵 X 为对称矩阵, 我们对其作 SVD, 令 $X = \mathcal{U}S\mathcal{V}$, 其中 $\mathcal{U}, \mathcal{V} \in \mathbb{M}_{|V| \times |V|}$ 并且是正定矩阵, $S \in \mathbb{M}_{|V| \times |V|}$ 并且是对角矩阵, 记 $\sigma_i = S_{i,i}$, 取 \mathcal{U} 的前 K 列 $\mathcal{U}^{[1:K]}$ 作为最后的 K 维 word vector, 这时候第 i 行的向量 $\mathbf{u}_i = \mathcal{U}_i^{[1:K]}$ 即为单词 w_i 的 word vector。

Drawbacks and Extensions

这样的模型主要有以下几个问题:

- 由矩阵本身引起的问题: 矩阵太大并且非常稀疏、加入新单词后拓展非常耗时间。
- 关于 co-occurrence 的问题: 需要解决部分词出现太频繁所导致的问题 (比如 the etc.)。

我们考虑这两类的问题的解决方案, 首先是 co-occurrence 的问题, 我们可以用以下的方案来解决:

1. 令 $X_{i,j}^* = \min(X_{i,j}, t)$, 其中 $t \sim 100$

2. 使用 pearson correlation, 同时让负值取 0 从而代替原来的简单的计数。
3. 使用 ramp window : 给 window 的不同位置加权, 一个比较自然的想法是距离更近的位置的权值更高。

我们无法很好的解决第一个问题, 这就引入了下面的2.2的内容。

2.2 Iteration Based Methods: word2vec

我们在本节的模型中都会运用到以下记号：

- w_i 为单词集 (vocabulary) 中的第 i 个单词
- n 为单词空间的维数
- $\mathcal{V} \in \mathbb{R}^{n \times |V|}$ 为 input word matrix, 其中 v_i 是 \mathcal{V} 的第 i 列, 表示单词 w_i 的 input vector
- $\mathcal{U} \in \mathbb{R}^{n \times |V|}$ output word matrix, 其中 u_i 是 \mathcal{U} 的第 i 列, 表示单词 w_i output vector
- window size 为 m 。

CBOW Model

在 CBOW(Continuous Bag of Word) 模型中, 我们考虑一个问题：使用周围的单词 (context) 预测中间的单词 (center word), 我们考虑使用以下的预测模型结构：

假设我们需要预测的单词是 $w^{(c)}$, 其周围的单词为 $w^{(c-m)}, \dots, w^{(c-1)}, w^{(c+1)}, \dots, w^{(c+m)}$, 为了方便表示, 我们使用这些单词的 one-hot 表示形式 $x^{(c-m)}, \dots, x^{(c-1)}, x^{(c+1)}, \dots, x^{(c+m)}$, 则相应的单词 $w^{(c+i)}$ 所对应的 input vector 为

$$v^{(c+i)} = \mathcal{V}x^{(c+i)} \quad (2)$$

我们对这 $2m$ 个向量取平均, 得到一个 n 维向量 \hat{v}

$$\hat{v} = \frac{1}{2m}(v^{(c-m)} + \dots + v^{(c-1)} + v^{(c+1)} + \dots + v^{(c+m)}) \quad (3)$$

接下来我们得到了一个得分向量 (score vector) z , 这里 $z \in \mathbb{R}^{|V|}$ 其第 i 个元素表示 \hat{v} 和第 i 个单词的输出向量的相似程度, 这里的相似程度用点乘的大小来度量)：

$$z = \mathcal{U}\hat{v} \quad (4)$$

最后我们把得分向量作一个 softmax 变换得到中心的单词的概率分布

$$\hat{y} = \text{softmax}(z) \quad (5)$$

我们定义损失函数 (统计/决策论意义上的) H 为：

$$H(y, \hat{y}) = - \sum_{y=1}^{|V|} y_i \log(\hat{y}_i) \quad (6)$$

实际上, 如果让 y 为 center word 的 one-hot 向量, 其中 $y_k = 1$, 那么, $H(y, \hat{y})$ 可以简化为

$$H(y, \hat{y}) = -y_k \log(\hat{y}_k) \quad (7)$$

我们写出最后的 object function 的化简形式

$$J = -\log \mathbb{P}(\mathbf{u}_c | w^{(c-m)}, \dots, w^{(c-1)}, w^{(c+1)}, \dots, w^{(c+m)}) \quad (8)$$

$$= -\log \frac{\exp(\mathbf{u}_c^T \hat{\mathbf{v}})}{\sum_{j=1}^{|V|} \exp(\mathbf{u}_j^T \hat{\mathbf{v}})} \quad (9)$$

$$= -\mathbf{u}_c^T \hat{\mathbf{v}} + \log \sum_{j=1}^{|V|} \exp(\mathbf{u}_j^T \hat{\mathbf{v}}) \quad (10)$$

其中 \mathbf{u}_c 为 center word 的 output vector。

结合上面的表达式，我们需要对这两个 vector 有一个直观的认识，实际上，我们的优化目标是让每个 center word 周围的单词的 input vector 的平均值尽量接近其 output vector。

Skip-Gram Model

Skip-Gram Model 的方向正好和 CBOW 相反，在 Skip-Gram 中，我们给定中间的 center word，需要预测周围的 context，为了简化描述，我们直接给出概率形式的表达：式 [11] 给出了给定中间单词 w_c ，单词 w_o 在其 window 中出现的概率；式 [12] 给出了损失函数的形式，这里我们使用了进一步的独立性假设 $\mathbb{P}(w_{c-m}, \dots, w_{c-1}, w_{c+1}, \dots, w_{c+m} | w_c) = \prod_{j=-m, j \neq 0}^m \mathbb{P}(w_{c+j} | w_c)$ (虽然这非常不符合实际)；我们把 [11] 代入 [12] 并且进一步化简得到式 [13]

$$\mathbb{P}(w_o | w_c) = \frac{\exp(\mathbf{u}_o^T \mathbf{v}_c)}{\sum_{k=1}^{|V|} \exp(\mathbf{u}_k^T \mathbf{v}_c)} \quad (11)$$

$$J = -\log \prod_{j=-m, j \neq 0}^m \mathbb{P}(w_{c+j} | w_c) \quad (12)$$

$$J = -\sum_{j=-m, j \neq 0}^m \mathbf{u}_{c+j}^T \mathbf{v}_c + 2m \log \sum_{k=1}^{|V|} \exp(\mathbf{u}_k^T \mathbf{v}_c) \quad (13)$$

我们给出梯度的表达式：

$$\frac{\partial J}{\partial \mathbf{v}_c} = -\sum_{j=-m, j \neq 0}^m \mathbf{u}_{c+j}^T + 2m \sum_{x=1}^{|V|} \frac{\exp(\mathbf{u}_x^T \mathbf{v}_c)}{\sum_{k=1}^{|V|} \exp(\mathbf{u}_k^T \mathbf{v}_c)} \mathbf{u}_x^T \quad (14)$$

$$\frac{\partial J}{\partial \mathbf{u}_x} = m \frac{\exp(\mathbf{u}_x^T \mathbf{v}_c)}{\sum_{k=1}^{|V|} \exp(\mathbf{u}_k^T \mathbf{v}_c)} \mathbf{v}_c^T - y_x \mathbf{v}_c^T \quad (15)$$

但是，最后我们的形式仅考虑一对 $\mathbb{P}(o|c)$ ，最后的 object function 的结果如式 [16]，梯度的表达式如式 [17] 和 [18]

$$J = -\mathbf{u}_{c+j}^T \mathbf{v}_c + \log \sum_{k=1}^{|V|} \exp(\mathbf{u}_k^T \mathbf{v}_c) \quad (16)$$

$$\frac{\partial J}{\partial \mathbf{v}_c} = -\mathbf{u}_{c+j}^T + \sum_{x=1}^{|V|} \frac{\exp(\mathbf{u}_x^T \mathbf{v}_c)}{\sum_{k=1}^{|V|} \exp(\mathbf{u}_k^T \mathbf{v}_c)} \mathbf{u}_x^T \quad (17)$$

$$\frac{\partial J}{\partial \mathbf{u}_x} = \frac{\exp(\mathbf{u}_x^T \mathbf{v}_c)}{\sum_{k=1}^{|V|} \exp(\mathbf{u}_k^T \mathbf{v}_c)} \mathbf{v}_c^T - y_x \mathbf{v}_c^T \quad (18)$$

Negative Sampling

考虑到以上的模型在实际计算中运算量非常大，所以我们考虑另外一种近似方法，这种近似方法的阐述也分成两个部分：第一个部分是 object function，第二个部分是 gradient。

首先，我们考虑 Skip-Gram Model 的 Negative Sampling 的近似优化方法：

我们使用 (x, c) 来分别表示 (context, center)，同时记 $D = \{(x, c) | w_x = w^{(c+j)}, -m \leq j \leq m, j \neq 0\}$ ，那么，我们可以得到另外一种 $\mathcal{P}(w_o | w_c)$ 的表达式 [19]

$$\mathbb{P}((o, c) \in D) = \sigma(\mathbf{u}_o^T \mathbf{v}_c) = \frac{1}{1 + \exp(-\mathbf{u}_o^T \mathbf{v}_c)} \quad (19)$$

同时使用独立性条件，我们写出 object function 的表达式 [20]，将 [19] 的结果代入，得到最终的表达式 [21]

$$J = -\log \left\{ \prod_{(o,c) \in D} \mathbb{P}((o, c) \in D) \prod_{(x,c) \in \bar{D}} \mathbb{P}((w, c) \in \bar{D}) \right\} \quad (20)$$

$$J = - \sum_{(o,c) \in D} \log(\sigma(\mathbf{u}_o^T \mathbf{v}_c)) - \sum_{(w,c) \in \bar{D}} \log(\sigma(-\mathbf{u}_w^T \mathbf{v}_c)) \quad (21)$$

我们把 \bar{D} 称为 negative corpus，在具体求解的时候，我们在 \bar{D} 中随机抽 K 个组成 (一对的) object function [22]，其梯度为 [23], [24] 和 [25]，其中每个单词被抽到的概率服从分布 [26]：

$$J = -\log(\sigma(\mathbf{u}_{c+j}^T \mathbf{v}_c)) - \sum_{k=1}^K \log(\sigma(-\bar{\mathbf{u}}_k^T \mathbf{v}_c)) \quad (22)$$

$$\frac{\partial J}{\partial \mathbf{v}_c} = -(1 - \sigma(\mathbf{u}_{c+j}^T \mathbf{v}_c)) \mathbf{u}_{c+j}^T + \sum_{k=1}^K (1 - \sigma(-\bar{\mathbf{u}}_k^T \mathbf{v}_c)) \bar{\mathbf{u}}_k^T \quad (23)$$

$$\frac{\partial J}{\partial \mathbf{u}_{c+j}} = -(1 - \sigma(\mathbf{u}_{c+j}^T \mathbf{v}_c)) \mathbf{v}_c^T \quad (24)$$

$$\frac{\partial J}{\partial \bar{\mathbf{u}}_k} = (1 - \sigma(-\bar{\mathbf{u}}_k^T \mathbf{v}_c)) \mathbf{v}_c^T \quad (25)$$

$$\mathbb{P}(w) \propto (\text{frequency of word } w)^{3/4} \quad (26)$$

同样，我们考虑 CBOW Model 的 Negative Sampling 的近似优化方法：沿用之前的记号，类似的，我们可以得出 object function：

$$J = -\log(\sigma(\mathbf{u}_c^T \hat{\mathbf{v}})) - \sum_{w \neq c} \log(\sigma(-\mathbf{u}_w^T \hat{\mathbf{v}})) \quad (27)$$

同样，我们可以得出 (一对的) object function [28]，梯度 [29], [30] 和 [31]。

$$J = -\log(\sigma(\mathbf{u}_c^T \hat{\mathbf{v}})) - \sum_{k=1}^K \log(\sigma(-\bar{\mathbf{u}}_k^T \hat{\mathbf{v}})) \quad (28)$$

$$\frac{\partial J}{\partial \mathbf{v}_{c+j}} = \frac{1}{2m} \left\{ - (1 - \sigma(\mathbf{u}_c^T \hat{\mathbf{v}})) \mathbf{u}_c^T + \sum_{k=1}^K (1 - \sigma(-\bar{\mathbf{u}}_k^T \hat{\mathbf{v}})) \bar{\mathbf{u}}_k^T \right\} \quad (29)$$

$$\frac{\partial J}{\partial \mathbf{u}_c} = - (1 - \sigma(\mathbf{u}_c^T \hat{\mathbf{v}})) \hat{\mathbf{v}}^T \quad (30)$$

$$\frac{\partial J}{\partial \bar{\mathbf{u}}_k} = (1 - \sigma(-\bar{\mathbf{u}}_k^T \hat{\mathbf{v}})) \hat{\mathbf{v}}^T \quad (31)$$

$$(32)$$

2.3 Evaluation: Intrinsic and Extrinsic

3 Neural Networks

3.1 Neuron, Terminology, Forward Propagation

3.2 Back Propagation, Computational Graph

3.3 Tips and Tricks

4 Recurrent Neural Networks

5 Recursive Neural Networks

6