Word2vec Assignment 2

Congfeng Yin

2022-05-30

1 Written: Understanding word2vec

1.1 符号说明

- d 词向量的维度
- n 词汇数量
- $\mathbf{U} \in \mathbb{R}^{d\times n}$ 每一列是一个词向量，$\mathbf{u}_w \in \mathbb{R}^{d\times 1}$
- $\mathbf{V} \in \mathbb{R}^{d\times n}$ 每一列是一个词向量，$\mathbf{v}_w \in \mathbb{R}^{d\times 1}$
- $\mathbf{y} \in \mathbb{R}^{n\times 1}$ 真实值，one-hot 向量
- $\hat{\mathbf{y}} \in \mathbb{R}^{n\times 1}$ 预测值，表示属于某个词的概率

1.2 Question (a)

Defination of $\mathbf{y}$:

$$y_w = \begin{cases} 1 & w = o \\ 0 & w \neq o \end{cases}$$

Hence,

$$-\sum_{w=1}^{V} y_w \log(\hat{y}_w) = -y_o \log(\hat{y}_o) = -\log(\hat{y}_o) = -\log P(O=o|C=c)$$

1.3 Question (b)

$$
\begin{aligned}
\frac{\partial \mathcal{J}_{naive-softmax}(\mathbf{v}_c, o, \mathbf{U})}{\partial \mathbf{v}_c} &= \frac{\partial}{\partial \mathbf{v}_c}[-\mathbf{u}_o^T\mathbf{v}_c + \log \sum_{w\in Vocab} \exp(\mathbf{u}_w^T\mathbf{v}_c)] \\
&= \frac{\partial}{\partial \mathbf{v}_c}[\log \sum_{w\in Vocab} \exp(\mathbf{u}_w^T\mathbf{v}_c)] - \mathbf{u}_o \\
&= \frac{\sum_{x\in Vocab} \exp(\mathbf{u}_x^T\mathbf{v}_c)\mathbf{u}_x}{\sum_{w\in Vocab} \exp(\mathbf{u}_w^T\mathbf{v}_c)} - \mathbf{u}_o \quad {}^{1} \\
&= \sum_{x\in Vocab} \frac{\exp(\mathbf{u}_x^T\mathbf{v}_c)}{\sum_{w\in Vocab} \exp(\mathbf{u}_w^T\mathbf{v}_c)}\mathbf{u}_x - \mathbf{u}_o \\
&= \sum_{x\in Vocab} [\frac{\exp(\mathbf{u}_x^T\mathbf{v}_c)}{\sum_{w\in Vocab} \exp(\mathbf{u}_w^T\mathbf{v}_c)}\mathbf{u}_x - y_x\mathbf{u}_x] \\
&= \sum_{x\in Vocab} \mathbf{u}_x(\hat{y}_x - y_x) \\
&= \mathbf{U}(\hat{\mathbf{y}} - \mathbf{y}) \in \mathbb{R}^{d\times 1}
\end{aligned}
$$

推导结果是 outside vector $\mathbf{u}_x$ 的期望减去 $\mathbf{u}_x$ 的实际值,所以可以用 $\mathbf{U}(\hat{\mathbf{y}}-\mathbf{y})$ 表示。

1.4 Question (c)

$$
\frac{\partial \mathcal{J}_{naive-softmax}(\mathbf{v}_c, o, \mathbf{U})}{\partial \mathbf{u}_w} = \frac{\partial}{\partial \mathbf{u}_w}[-\mathbf{u}_o^T\mathbf{v}_c + \log \sum_{w\in Vocab} \exp(\mathbf{u}_w^T\mathbf{v}_c)]
$$

[1] 对数函数的复合函数求导

If $w = o$,

$$\begin{aligned}
\frac{\partial \mathcal{J}_{naive-softmax}(\mathbf{v}_c, o, \mathbf{U})}{\partial \mathbf{u}_w} &= -\mathbf{v}_c + \frac{1}{\sum_{w \in Vocab} \exp(\mathbf{u}_w^T \mathbf{v}_c)} \frac{\partial}{\partial \mathbf{u}_o} \sum_{w \in Vocab} \exp(\mathbf{u}_w^T \mathbf{v}_c) \\
&= -\mathbf{v}_c + \frac{1}{\sum_{w \in Vocab} \exp(\mathbf{u}_w^T \mathbf{v}_c)} \frac{\partial}{\partial \mathbf{u}_o} \exp(\mathbf{u}_o^T \mathbf{v}_c) \\
&= -\mathbf{v}_c + \frac{\exp(\mathbf{u}_o^T \mathbf{v}_c)}{\sum_{w \in Vocab} \exp(\mathbf{u}_w^T \mathbf{v}_c)} \mathbf{v}_c \\
&= [P(O = o|C = c) - 1]\mathbf{v}_c \\
&= (\hat{y}_w - y_w)\mathbf{v}_c \quad (w = o, y_w = 1)
\end{aligned}$$

If $w \neq o$,

$$\begin{aligned}
\frac{\partial \mathcal{J}_{naive-softmax}(\mathbf{v}_c, o, \mathbf{U})}{\partial \mathbf{u}_w} &= \frac{\exp(\mathbf{u}_w^T \mathbf{v}_c)}{\sum_{w \in Vocab} \exp(\mathbf{u}_w^T \mathbf{v}_c)} \mathbf{v}_c \\
&= P(O = w|C = c)\mathbf{v}_c \\
&= (\hat{y}_w - y_w)\mathbf{v}_c \quad (w \neq o, y_w = 0)
\end{aligned}$$

i.e.

$$\frac{\partial \mathcal{J}_{naive-softmax}(\mathbf{v}_c, o, \mathbf{U})}{\partial \mathbf{u}_w} = (\hat{y}_w - y_w)\mathbf{v}_c \in \mathbb{R}^{d \times 1}$$

1.5 Question (d)

$$\begin{aligned}
\frac{\partial \mathcal{J}_{naive-softmax}(\mathbf{v}_c, o, \mathbf{U})}{\partial \mathbf{U}} &= [\frac{\partial}{\partial \mathbf{u}_1}, \cdots, \frac{\partial}{\partial \mathbf{u}_n}]\mathcal{J}_{naive-softmax}(\mathbf{v}_c, o, \mathbf{U}) \\
&= [(\hat{y}_1 - y_1), \cdots, (\hat{y}_n - y_n)]\mathbf{v}_c \\
&= \mathbf{v}_c(\hat{\mathbf{y}} - \mathbf{y})^T \in \mathbb{R}^{d \times n}
\end{aligned}$$

1.6 Question (e)

$$\begin{aligned}\frac{\partial\sigma(x)}{\partial x} &= \frac{\partial}{\partial x}\frac{e^x}{e^x+1} \\ &= \frac{e^x(e^x+1)-e^xe^x}{(e^x+1)^2} \quad {}^1 \\ &= \frac{e^x}{(e^x+1)^2} \\ &= \sigma(x)(1-\sigma(x))\end{aligned}$$

1.7 Question (f)

$$\frac{\partial \mathcal{J}_{neg-sample}(\mathbf{v}_c, o, \mathbf{U})}{\partial \mathbf{v}_c} = -[1-\sigma(\mathbf{u}_o^T\mathbf{v}_c)]\mathbf{u}_o + \sum_{k=1}^{K}[1-\sigma(-\mathbf{u}_k^T\mathbf{v}_c)]\mathbf{u}_k \tag{1}$$

$$\begin{aligned}\frac{\partial \mathcal{J}_{neg-sample}(\mathbf{v}_c, o, \mathbf{U})}{\partial \mathbf{u}_o} &= -[1-\sigma(\mathbf{u}_o^T\mathbf{v}_c)]\mathbf{v}_c \\ \frac{\partial \mathcal{J}_{neg-sample}(\mathbf{v}_c, o, \mathbf{U})}{\partial \mathbf{u}_k} &= [1-\sigma(-\mathbf{u}_k^T\mathbf{v}_c)]\mathbf{v}_c\end{aligned}$$

$\mathcal{J}_{naive-softmax}$ 里边包含 $\hat{\mathbf{y}}$，需要计算 $\mathbf{v}_c$ 和语料库中所有词向量的内积，$\mathcal{J}_{neg-sample}$ 只用计算相关的几个词向量，所以计算效率更高。

1.8 Question (g)

Assume $\mathbf{u}_i = \mathbf{u}_k$ when $i \in \{1, \cdots, m\}$, $\mathbf{u}_i \neq \mathbf{u}_k$ when $i \in \{m+1, \cdots, K\}, m \leq K$,

$$\begin{aligned}\frac{\partial \mathcal{J}_{neg-sample}(\mathbf{v}_c, o, \mathbf{U})}{\partial \mathbf{u}_k} &= \frac{\partial}{\partial \mathbf{u}_k}[-\sum_{k=1}^{m}\log(\sigma(-\mathbf{u}_k^T\mathbf{v}_c)) - \sum_{k=m+1}^{K}\log(\sigma(-\mathbf{u}_k^T\mathbf{v}_c))] \\ &= -\sum_{k=1}^{m}\frac{\sigma(-\mathbf{u}_k^T\mathbf{v}_c)[1-\sigma(-\mathbf{u}_k^T\mathbf{v}_c)]}{\sigma(-\mathbf{u}_k^T\mathbf{v}_c)}(-\mathbf{v}_c) \\ &= \sum_{k=1}^{m}[1-\sigma(-\mathbf{u}_k^T\mathbf{v}_c)]\mathbf{v}_c\end{aligned}$$

[1] 分式求导公式 $(\frac{u}{v})' = \frac{u'v-uv'}{v^2}$

1.9 Question (h)

1.9.1 (i)

$$\frac{\partial \mathcal{J}_{skip-gram}(\mathbf{v}_c, w_{t-m}, \cdots, w_{t+m}, \mathbf{U})}{\partial \mathbf{U}} = \sum_{-m \le j \le m, j \ne 0} \frac{\partial \mathcal{J}(\mathbf{v}_c, w_{t+j}, \mathbf{U})}{\partial \mathbf{U}}$$

含义：这里可以理解为多个窗口词梯度矩阵的叠加，不同梯度矩阵包含的窗口词梯度向量不同

1.9.2 (ii)

$$\frac{\partial \mathcal{J}_{skip-gram}(\mathbf{v}_c, w_{t-m}, \cdots, w_{t+m}, \mathbf{U})}{\partial \mathbf{v}_c} = \sum_{-m \le j \le m, j \ne 0} \frac{\partial \mathcal{J}(\mathbf{v}_c, w_{t+j}, \mathbf{U})}{\partial \mathbf{v}_c}$$

含义：当前窗口内中心词的梯度是所有中心-outside 词对梯度的总和

1.9.3 (iii)

$$\frac{\partial \mathcal{J}_{skip-gram}(\mathbf{v}_c, w_{t-m}, \cdots, w_{t+m}, \mathbf{U})}{\partial \mathbf{v}_w} = 0$$

含义：不在当前窗口内的中心词梯度为零

2 Coding: Implementing word2vec

2.1 (a)

```
def LossAndGradient(
    centerWordVec,
    outsideWordIdx,
    outsideVectors,
    dataset
):
    return loss, gradCenterVec, gradOutsideVecs
```

- centerWordVec: $\mathbf{v}_c \in \mathbb{R}^{1\times d}$（注意 numpy 中 (d,) 是 d 维行向量）
- outsideVectors: $\mathbf{U} \in \mathbb{R}^{n\times d}$
- loss: $\mathcal{J} \in \mathbb{R}$
- gradCenterVec: $\frac{\partial \mathcal{J}}{\partial \mathbf{v}_c} \in \mathbb{R}^{1\times d}$（行向量）
- gradOutsideVecs: $\frac{\partial \mathcal{J}}{\partial \mathbf{U}} \in \mathbb{R}^{n\times d}$

2.1.1 Implement the negative sampling loss and gradient

```python
import numpy as np
def negSamplingLossAndGradient(
    centerWordVec,
    outsideWordIdx,
    outsideVectors,
    dataset,
    K=10
):
    # Negative sampling of words .
    negSampleWordIndices = getNegativeSamples(outsideWordIdx, dataset, K)
    indices = [outsideWordIdx] + negSampleWordIndices

    u_o_v_c = np.dot(outsideVectors[outsideWordIdx].T, centerWordVec)
    u_k_v_c = -np.dot(outsideVectors[negSampleWordIndices],
        centerWordVec)
    loss = -np.log(sigmoid(u_o_v_c)) - np.sum(np.log(sigmoid(u_k_v_c)))
    gradCenterVec = -(1-sigmoid(u_o_v_c)) *
        outsideVectors[outsideWordIdx] \
       + np.sum(np.expand_dims((1-sigmoid(u_k_v_c)), axis=1) *
           outsideVectors[negSampleWordIndices], axis=0)
    gradOutsideVecs = np.zeros(outsideVectors.shape)
    gradOutsideVecs[outsideWordIdx] = -(1-sigmoid(u_o_v_c)) *
        centerWordVec
    for idx, u_k_idx in enumerate(negSampleWordIndices):
       gradOutsideVecs[u_k_idx] += (1 - sigmoid(u_k_v_c[idx])) *
           centerWordVec

    return loss, gradCenterVec, gradOutsideVecs
```

- 注意事项 1
  关于 gradCenterVec 的计算：根据式(1)，第二项 $\sum_{k=1}^{K}[1-\sigma(-\mathbf{u}_k^T\mathbf{v}_c)]\mathbf{u}_k$ 是元素积（使用 $*$，表示对应位置相乘，np.dot 对应的是内积）。
  其中 np.expand_dims((1-sigmoid(u_k_v_c)), axis=1)$\in \mathbb{R}^{n\times 1}$，outsideVectors[negSampleWordIndices]$\in \mathbb{R}^{n\times d}$。

- 注意事项 2
  关于 $\mathbf{u}_k$ 梯度的计算：采用下边两种方式均可（区别在于索引的位置）。

```
gradOutsideVecs[u_k_idx] += np.dot((1 - sigmoid(u_k_v_c[idx])),
    centerWordVec) # method1
gradOutsideVecs[u_k_idx] += np.dot((1 - sigmoid(u_k_v_c)[idx]),
    centerWordVec) # method2
```