Lab 6: Word2Vec

1 Intro

skip-gram Skip-gram算法是指在给出目标单词（中心单词）的情况下，预测它的上下文单词（除中心单词外窗口内的其他单词，本实验中窗口大小是2，也就是左右各两个单词）。

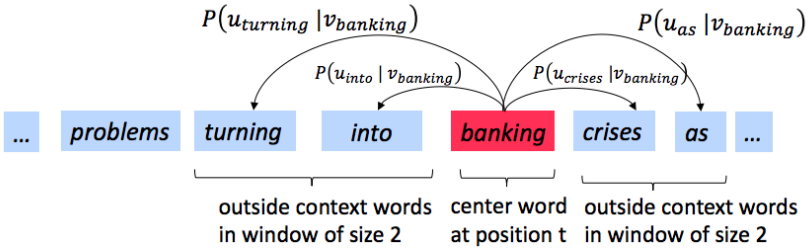

Figure 1: skip-gram

求两个词向量的相似度：设d维列向量$\boldsymbol{u}_o(shape:d*1)$和$\boldsymbol{v}_c(shape:d*1)$分别是外部的词向量(outside vector)和中心单词的词向量(center vector)，矩阵$\boldsymbol{U}(shape:n*d)$的每一行代表每一个外部词向量$\boldsymbol{u}_o^T$，矩阵$\boldsymbol{V}(shape:n*d)$的每一行代表所有的中心词向量$\boldsymbol{v}_c^T$（其中$n$代表词表大小，$o,c$下标代表$U,V$中的某一行），$\boldsymbol{U}$和$\boldsymbol{V}$矩阵都包含了所有的单词。

使用$softmax$函数计算$\boldsymbol{u}_o,\boldsymbol{v}_c$的相似度为：

$$P(O=o|C=c)=\frac{exp(\boldsymbol{u}_o^T\boldsymbol{v}_c)}{\sum_{w\in Vocab}exp(\boldsymbol{u}_w^T\boldsymbol{v}_c)} \tag{1}$$

损失函数为：

$$J(\boldsymbol{v}_c,o,\boldsymbol{U})=-logP(O=o|C=c) \tag{2}$$

对损失函数求导：

$$\frac{\partial J(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{v}_c}=\boldsymbol{U}^T(\hat{\boldsymbol{y}}_c-\boldsymbol{y}_c) \tag{3}$$

$$\frac{\partial J(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{U}}=(\hat{\boldsymbol{y}}_c-\boldsymbol{y}_c)\boldsymbol{v}_c^T \tag{4}$$

其中：

- $\hat{\boldsymbol{y}}_c:=P(O|C=c)(shape:n*1)$是给定$\boldsymbol{v}_c$计算outside words的概率分布
- $\boldsymbol{y}_c(shape:n*1)$是一个one-hot向量，只有在单词$o$（true outside word）对应的index位置是1、其余位置是0，可看作$\hat{\boldsymbol{y}}_c$要学习的目标

2 Environment

创建环境：

```
conda env create -f env.yml
conda activate a2
```

退出环境：

```
conda deactivate
```

3 TODO

在word2vec.py文件中完成下列任务

1. 实现sigmoid 函数
2. 实现损失函数
3. 实现损失函数求导得到gradCenterVec, gradOutsideVecs

Note: How to run

1. (optional)终端输入'sh get_datasets.sh'，完成数据准备（可跳过，数据已附在压缩包）
2. 通过运行'python word2vec.py'来检验word2vec.py文件里的三个TODO是否完成、是否有bug
3. 运行'python run.py'即可训练word2vec（训练时间可能较长，默认的40000iter会有一两个小时左右），训练完成即可查看生成在当前文件夹下的word_vectors.png图片（图片是对部分默认单词的可视化，可在run.py的visualizeWords中修改）

4 Submit

最终提交实验代码以及实验报告，报告中包括但不限于对词向量的认识，实验生成图片的理解等。

- 2022xxxxxx_xiaoming_lab6.zip (./code ./report.pdf)
- 提交至`https://k.ruc.edu.cn`, DDL 2025.11.21 23:59

Q&A

1. conda指令与sh指令
   - conda env create -f是根据给定yaml文件创建一个conda虚拟环境；conda指令在Windows cmd/Mac 终端/Linux 终端/Pycharm 终端/VScode 终端均可运行
   - sh指令可以在Mac/Linux的自带终端或者IDE附带终端中运行，Windows需要额外的配置才能运行.sh文件
   - 本实验中.sh文件是在实验文件中创建指定路径、并下载数据到这个路径；给出的实验文件压缩包中已经提前为大家下载好，所以准备数据这一步可跳过
2. 实验需要做什么
   - 文档中写的3个TODO：
     运行'python word2vec.py'即可检验三个TODO是否正确，代码输出'Your Result'和'Expected Result'一致，即为正确通过
   - 运行'python run.py'：
     需要在完成3个TODO之后才能正常运行
     该文件是根据你完成的代码训练一套word2vec，并在训练结束后自动在文件夹生成一个word_vectors.png图片，图片中展示的单词是在run.py中写定的(visualizeWords)，给的run.py代码中已经有一些默认单词，这些单词可以修改
     在实验文件压缩包中已经给定了一版这些单词的word_vectors.png图片，有条件的话希望大家都能重新自己训练并生成这个图片
3. 运行run.py相关
   - 训练时间：run.py配置中默认训练iter是40000(可修改)，训练时长大概在1至2h
   - 生成图片：默认的代码会在训练结束后自动在当前实验文件夹中生成word_vectors.png图片文件，不会在终端显示
   - 训练loss：在训练初级阶段不降低、有些抖动是正常的，按照给定的默认配置会在5000iter左右会开始有明显的下降趋势；loss数值一开始会在20左右、最后数值在10左右是差不多收敛位置
4. 向量相关
   - 行向量与列向量：
     论文里书写的单个向量符号默认是列向量，因此我们在实验文档中定义和公式位置书写的向量也都是按照列向量形式，$\boldsymbol{u}_o(shape: d*1), \boldsymbol{v}_c(shape: d*1), \hat{\boldsymbol{y}}_c(shape: n*1), \boldsymbol{y}_c(shape: n*1)$，$\boldsymbol{U}$和$\boldsymbol{V}$也都是按列排列，每一列是一个词向量
     但单个向量在代码实现中一般是行的形式（比如单个一维的numpy.array），多个向量一般也都按行排列；因此实验代码中给的outsideVectors也是按行排列的，每一行是一个词向量
5. 报错相关
   - conda env create：HTTP error是网络问题，可更换网络或者更换conda源来解决
   - 数值/运算操作相关的type error或者value error大概率是因为在TODO部分有了不正确的数值处理，希望大家能仔细检查该部分，也可对该部分书写的计算单独拉出来进行测试
6. TODO相关代码书写格式
   - 一般给大家留的TODO里，会留一些变量名，比如'# $y\_hat =$'这种格式，留的意图是起到一些引导作用，大家并非必须要采用这种格式，输入输出一致即可，中间变量按照自己思路书写即可
   - TODO之外的代码，大家如果觉得书写得和自己思路差异大、且想修改的话，可以在不影响算法目的和效果、输入输出的情况下自行修改，在报告里说明一下即可