第一题：

在本题中，你将构建一个简单的卷积模型，分为以下几步：

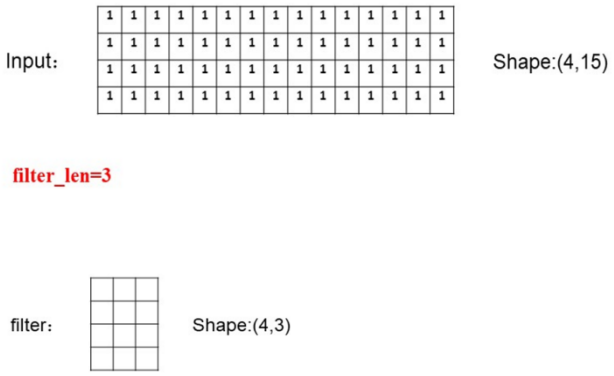
1. 定义函数 Conv1D，有三个参数(input_array, filter_len, stride)。

1）其中 input_array 是一个随机二维数组。

filter_len 为一个整数，用于产生第二个数组 filter_array。第二个数组 filter_array 由 np.random.randn()函数产生，列数为 filter_len，行数与 input_array 行数相同。

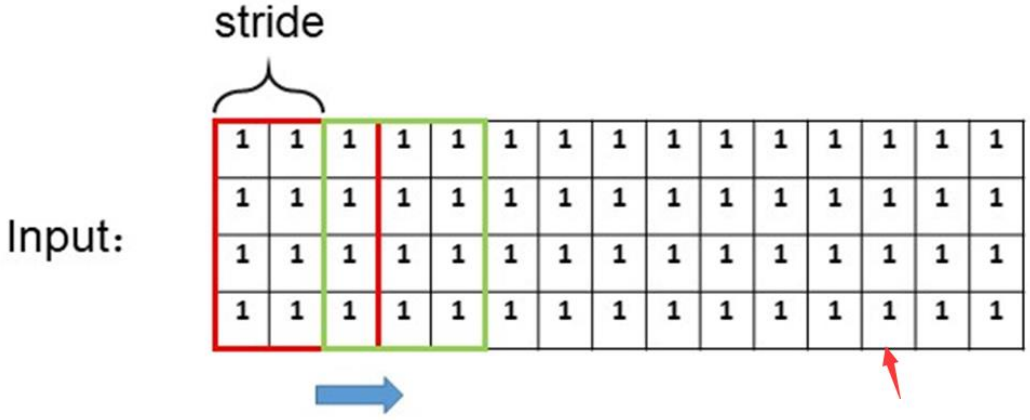
stride 为一个自定义的整数，范围为 1-5。

示例：如下图所示，input_array 为一个(4,15)的二维数组，在 filter_len 等于 3 的时候，生成的 filter_array 的形状为（4，3）。



2）卷积运算

a) 在 filter_len 为 3 的情况下，filter_array 与 input_array 前 3 列对应位置元素相乘，所有位置的乘积相加，得到结果列表的第 1 个元素。
b) 然后往右移动 stride 列，filter_array 的元素再与 input_array 对应位置元素相乘，乘积相加得到结果的第 2 个元素。依次类推，得到第 3,4,5…个输出元素。直到右端剩余列数小于 stride。
c) 将输出元素按照顺序组合成输出列表。



示例：在 filter_len 为 3，stride 等于 2 的情况下，结果的第一个元素为 filter_array 与 input_array 的[0-2]列(前 3 列)相乘后所有乘积相加的结果。第二个元素为

filter_array 与 input_array 的[2-4]列相乘，乘积相加的结果，以此类推，红色箭头处为循环终止位值。

2. One-hot 编码，函数 one_hot()

该函数实现将 DNA 序列编码成数组矩阵，具体要求如下：

输入为一条 DNA 序列(可以是列表、一维数组或者字符串)，建立一个空列表作为输出列表。然后遍历 DNA 序列，如果是 A 则编码成[1,0,0,0]添加到输出列表中，T 则编码成[0,1,0,0]，C 则编码成[0,0,1,0]，G 则编码成[0,0,0,1]。最后将输出列表转换成数组，然后转置，变成行数为 4，列数等于 DNA 序列长度的二维数组。

```
>>> one_hot('AATTCGCG')
array([[1., 1., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 1., 0.],
       [0., 0., 0., 0., 0., 1., 0., 1.]], dtype=float32)
```

```
>>> one_hot(['A','A','T','T','C','G','C','G'])
array([[1., 1., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 1., 0.],
       [0., 0., 0., 0., 0., 1., 0., 1.]], dtype=float32)
```

3. 序列文件读取、处理及保存

1） 用 numpy 从 seq.txt 读取序列，跳过前面只包含#的行。

读取序列名和序列信息存储到字典 **d_seq** 中。

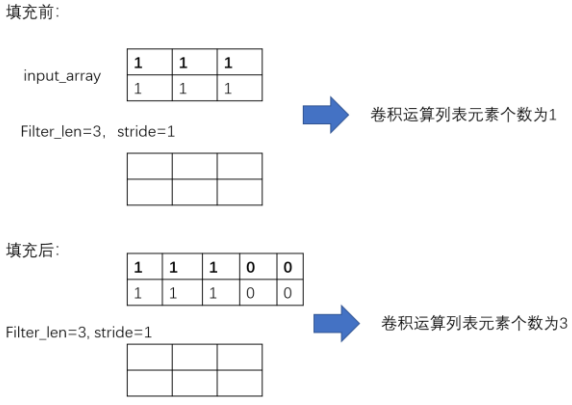
2） 建立新字典 d_code，键仍为序列名，值为对应序列的 ont-hot 编码

3） 自定义参数 filter_len 和 stride 的值，利用函数 Conv1D()对 d_code 中的值进行卷积计算，将序列名和结果保存到字典 **d_conv** 中并输出。

4. 边界填充

修改 Conv1D 函数得到一个新函数 Conv1D_new，该函数增加了一个参数 padding_type，可以支持边界填充，调用时如果实参为 T 则为填充，实参为 F 则为不填充。

在填充设置为 T 的情况下，对 input_array 进行填充，填充的行数与 input_array 相同，填充的列数根据 filter_len 而定，填充矩阵的值为 0。填充后使在 stride 为 1 时卷积运算后的输出的列表长度与 input_array 的列数相同。



第二组：

文件：

1.DEP.txt 为抑郁症字段数据

2.ukbref.txt 为 UKB 常见基本信息数据

3.IDP.txt 为部分脑图谱表型字段数据

问题：

1. ukbref.txt 提供了 UKB 人群的性别，出生年份，收录信息日期，以及人种信息。合并 ukbref.txt 和 DEP.txt，根据 eid,保留在两个文件中同时存在的个体信息，输出 DEP.txt.1。分别利用 pandas 和非 pandas 方法完成该问题，利用 time.time()函数分别输出所用时间。

2. 利用 numpy 或 pandas 方法完成该题。对于 DEP.txt.1 中存在的空值，用-1 进行填充。选择 20441、20446 同时为 1，并且 20546-0.1、20546-0.2、 20546-0.3 中至少有一个为 3 的人群作为严格判定的抑郁症患者人群，根据抑郁症患者人数，从剩余 20441、20446 同时为 0 的非抑郁症患者人群中利用随机函数随机挑选相同人数作为对照组，生成新的列 DEP,按照抑郁症值为 1，非抑郁症值为 2，填入该列，输出 DEP.txt.2。随机生成 100 次对照组，计算每次对照组的男女比例，对 100 次随机产生的对照组的男女比例求平均值和方差。输出 DEP.txt.3。

3. 利用 numpy 完成该题。对 IDP.txt 进行 PCA 分析。

1) .读取 IDP.txt 文件，生成矩阵 arr1。

2) .将数据矩阵标准化:标准 0-1 变换,按列获取每列最大 x_{max} 最小值 x_{min}，$x=(x-x_{min})/(x_{max}-x_{min})$。即每个位置的值等于该位置原始值减去该列最小值后除以该列最大最小值的差值，得到矩阵 arr2。

3) 安装模块 scipy,利用 zscore 对矩阵 arr2 按列进行标准化（zscore(arr2)），得到矩阵 arr3。

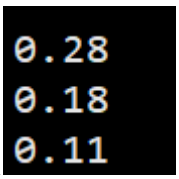
```
from scipy.stats import zscore
```

4) .求相关系数矩阵 $R=B^T.B$ ， 可通过 np.dot($arr3^T$, arr3)获得,得到矩阵 arr4。

5) .求特征值和特征向量，利用 np.linalg.eig(arr4)得到特征值 eig_vals 和特征向量 eigen_vecs。

6) .主成分贡献率即为该特征值除以全部特征值之和。对应的主成分系数则为对

应的特征向量。对得到的特征值排序，当前 n 个特征值对应的主成分贡献率>90%时，认为这 n 个新变量可以较好地代替原来的变量。输出这 n 个对应的特征值到IDP.txt.1



7) 利用 np.dot(arr3，eigen_vecs)可以将数据投影到 PC 上，即每个个体对应的各 PC 的系数，输出 IDP.txt.2。

```
1000186 -0.67 0.21 -0.03 0.18 1.14 0.59 -0.70 -1.17 0.86 -0.22
1000236 4.64 -0.21 0.50 0.23 -0.09 -0.11 -0.99 0.91 0.44 0.22
1000443 -1.73 -1.05 -0.33 0.40 -0.31 -0.40 0.89 -1.12 1.00 0.68
```

第三题：

文件：

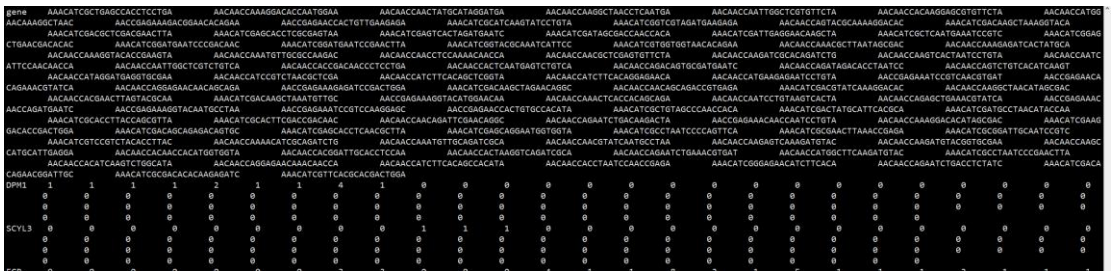
1. barcodes.tsv 为单细胞细胞标签数据。

2. features.tsv 为单细胞基因标签数据。

3. matrix.mtx 为单细胞矩阵坐标数据。

问题：

1. 将 matrix.mtx 文件转为矩阵格式，以基因作为行标签，细胞作为列标签，表达量为矩阵内数值，其中行标签根据 barcodes.tsv 做替换，列标签根据 features.tsv 做替换。如果一个基因在一个细胞内没有表达值，代表这个细胞未检测到该基因表达，用 0 填补矩阵内该位置。输出 matrix.txt.1。



2.利用 numpy 或 pandas 完成该题。读取 matrix.txt.1，对矩阵进行转置，统计每个细胞内有表达的基因数以及总的基因表达量。在这些基因中，存在线粒体基因，以“MT-”开头命名，计算线粒体基因的表达总量占全部基因的总表达量的百分百，保留两位小数。在矩阵后添加三列，分别为"nFeature_RNA", "nCount_RNA", "percent.mt"，将基因数、总的基因表达量、线粒体基因百分比分别填入这三列。统计每个基因存在表达的细胞数并计算在所有细胞中的平均表达量。在矩阵后添加两行，分别为"nCount_cell","average",将存在表达的细胞数和平均表达量分别填入这两行。将最终矩阵输出 matrix.txt.2。

3.利用 numpy 或 pandas 完成该题。对单细胞矩阵进行标准化处理。

1).根据 nCount_RNA 列作为每个细胞文库的大小。以计算所有细胞文库均值 meanCount。

2).计算每个细胞的细胞文库 size.factor，size.factor=nCount_RNA/meanCount。

3).然后根据每个细胞的细胞文库 size.factor 对该细胞的所有基因表达进行归一化处理，即基因表达量除以细胞文库因子 size.factor，得到 normalized。

4).归一化后进行 log 转换，由于单细胞测序结果的稀疏性，很多基因的表达值为 0，无法取 log，所以最终值取 log2(normalized+1)。得到新的最终标准化后的表达矩阵，输出 matrix.txt.3。

	AC009093.8	AC116667.2	AC079325.2	AC073111.5	AL033527.5	AL359762.1	AC005280.2	AC107959.4	BX255925.3	TBCE	AC016831.7	GABPB1-IT1	AL162377.3
AAACATCGCTGAGCCACCTCCTGA	1.03513074963	0.0	3.38456022429	0.0	0.0	0.0	0.0	0.0	0.0	1.03513074963	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.377733871

4. 安装 scipy,利用 kmeans 进行聚类。K-means 算法：对于给定的样本集，按照样本之间的距离大小，将样本集划分为 K 个簇。让簇内的点尽量紧密的连在一起，而让簇间的距离尽量的大。

```
from scipy.cluster.vq import vq,kmeans,whiten
from scipy import stats
```

1) .读取 matrix.txt.3，生成矩阵 arr1。

2) kmeans(arr1，n)进行 k-means 聚类，其中 n 为目标聚类簇数量，如 n=10,则分为 10 个簇。kmeans 输出聚类中心和损失 distortion 两维，取聚类中心，即 kmeans()[0],输出 arr2。（此处可以直接设定 n=10）

3) 使用 vq(arr2,arr1),根据聚类中心对所有数据进行分类,vq 的输出也是两维的,vq()[0]表示的是所有数据的 label。输出 arr3。每个数字代表了每个细胞所在的簇。下图为 100 个细胞的分类结果示意图，该图显示细胞 1-6 被分到第 9 类，第七个细胞被分到第 8 类，以此类推。

```
[9 9 9 9 9 8 4 2 8 9 9 4 9 9 4 0 4 6 5 9 4 8 9 9 9 4 9 9 6 9 9 6 4 9 8 9
 8 4 6 9 9 4 8 9 9 9 8 8 8 8 8 6 9 9 9 8 9 8 9 8 8 8 9 9 9 9 9 9 3 9 9 6 6
 1 9 9 9 9 6 9 6 6 9 7 6 9 9 9 9 6 6 9 3 9 9 9 6 9 9]
```

4）根据聚类结果，利用 scipy.stats.ttest_1samp 对每个簇的每个基因做单样本 t 检验。p_value = stats.ttest_1samp(data, n)[1]，其中 data 为该簇中某一基因对应的表达值组成的一维数组，n 为剩余簇中该基因的平均表达量。假设一共有 100 个细胞，其中一个簇内含有 10 个细胞。那对于该簇中的 DPM1 基因，取所有 10 个细胞中该基因的表达值组成一个长度为 10 的一维数组即为 data。同时在剩余的 90 个细胞中该基因表达的平均值即为 n。对每个簇分别统计每个基因的 P 值，并按照 P 值显著性进行排序。保存为 matrix.txt.3

cluster	gene	p_val
0	LYZ	0
0	VCAN	0
0	CST3	0