西安交通大學

XI'AN JIAOTONG UNIVERSITY

本科生实验报告

机器学习技术综合训练

实验 4：k-means 和 PCA 上机实践

姓名： 杨豪

班级： 软件 2101

时间： 2023 年 5 月 1 日

学号： 2206213297

目录

实验 4：k-means 和 PCA 上机实践

4.1 实验内容

基于 k-means 算法和 PCA 降维算法在给定的 Holiday 数据集上实现图像检索

4.2 实验原理

4.2.1 K-means

Algorithm 1: K-means

输入: 数据 $x_1, x_2, \cdots, x_n$, 簇的数目 K

输出: 簇中心 $\{\mu_1, \mu_2, \cdots, \mu_K\}$, 聚类结果 $C = C_1, C_2, ..., C_K$

随机选择 K 个数据点作为簇中心 $\{\mu_1, \mu_2, \cdots, \mu_K\}$

repeat

对每一个样本 x_j 进行归簇，距离哪个聚类中心最近，则将其归为哪一簇:

$$x_j \in C_i \Leftrightarrow \min_{t=1,...,K}\{\|x_j - \mu_t\|\} = \|x_j - \mu_i\|$$

重新计算每个簇 C_i 的均值:

$$E\mu_i = \frac{1}{|C_i|}\sum_{x_j \in C_i} x_j$$

将更新后的均值作为新的簇中心

until 簇中心不发生改变;

4.2.2 PCA

PCA 应用于数据预处理，使用 PCA 可以同时去除变量之间的线性关系以及对数据进行归一化：

1. 假设数据 $x_1, x_2, \cdots, x_m$ 的协方差矩阵为 $S = \frac{1}{m}\sum_{i=1}^{m}(x_i - \overline{x})(x_i - \overline{x})^T$
2. 利用 $W^T S W = \Lambda$ 定义一个变换 $y_i = \Lambda^{-\frac{1}{2}} W^T (x_i - \overline{x})$
3. 则 $y_1, y_2, \cdots, y_m$ 的均值为 0，协方差为单位矩阵。该操作称为数据白化 (Whitening) 操作

4.2.3 图像表达

基于内容的图像检索的核心问题就是计算图像表达

三角化嵌入

$$\mathrm{r}_j(x) = \left\{ \frac{x - \mathsf{c}_j}{\|\|x - \mathsf{c}_j\|} \right\} \mathrm{for} j = 1 \ldots |\mathcal{C}|,$$
$$\mathrm{R}(x) = [r_1(x)^T, \ldots, r_{|C|}(x)^T]^T$$

Sum pooling

$$\begin{aligned}\psi(\Phi_\triangle(\mathcal{X})) &= \sum_{x \in \mathcal{X}} \phi_\triangle(x) \\ &= \Sigma^{-1/2} \left(\sum_{x \in \mathcal{X}} \mathrm{R}(X) \right) - n\Sigma^{-1/2}\mathrm{R}_0. \\ \phi_\triangle(x) &= \Sigma^{-1/2}(\mathrm{R}(x) - \mathrm{R}_0),\end{aligned}$$

4.3 框架代码解读、补充与修改

题目已给定的代码有四个部分

- test_image_search.py 项目主文件，调取各个库文件的函数并执行
- run_triemb_learn.py
  - 导入 Flickr60k 训练数据 vtrain.mat 为 vtrain, 格式为 numpy.ndarray(128, 5000404)，已经过 desc_postprocess 函数处理。
  - 利用 k-means 聚类算法，找出 k 个聚类中心 C, k 为超参数。
  - 利用 SIFT 描述子信息 vtrain 和聚类中心 C，求出三角化嵌入后的特征值 eigval，特征向量 eigvec，特征均值 Xmean(128*k, 1)。
  - 将得到的聚类中心 C，特征值 eigval，特征向量 eigvec，特征均值 Xmean 以及投影矩阵 Pemb 保存为本地文件。
- run_embedding.py
  - 导入 Holidays 测试数据 X.mat 为 X，格式为 numpy.ndarray(128, 4455091)，未经过 desc_postprocess 函数处理。
  - 导入描述 Holidays 测试数据中每张图像（共 1491 张）对应 SIFT 描述子个数索引的参数 cndes.mat 为 cndes，格式为 numpy.ndarray，(1492，) 。
  - 导入用于对原始 SIFT 描述子进行 desc_postprocess 处理的参数 desc_mean.mat 为 desc_mean，格式为 numpy.ndarray(128,).

  - 实现对 Holidays 图像的特征表示 psi,(128*kc, 1491)，并保存为本地文件。

- img_eval
  - 导入第二部分保存的 psi 数据，采用 mAP(mean Average Precision) 指标实现对图像检索结果的评价。
  - 其中，qidx.mat 为 Holidays 数据集中检索图像的索引序列，gnd.mat 包含检索图像及对应的正确检索结果。
  - 最终输出不同 Power-law normalization 参数 pw 时对应的 map 。

共有四处需要根据实验原理填充，比较简单

4.3.1 triemb_learn 函数

根据实验原理填充如下

Listing 1: run_triemb_learn.py

```
def triemb_learn(vtrain, C):
    """return [Xmean, eigvec, eigval]"""

    # ......Some Code......

    np.mean(vtrain)
    for j in  range(kc):
        item = vtrain[i : i + slicesize] - C[j]
        item_normed = item / np.linalg.norm(item, axis=1).reshape(slicesize
            , 1)
        Xsum[j * d : j * d + d] += np. sum(item_normed, axis=0)
        Rx[i : i + slicesize, j * d : j * d + d] += item_normed

    Xmean = Xsum / nlearn
    Rx -= Xmean.reshape(1, D)
    covD = (Rx.T @ Rx) / nlearn

    eigval, eigvec = np.linalg.eig(covD)
    idx = eigval.argsort()[::-1]  # descending sort
    eigval = eigval[idx]
    eigvec = eigvec[:, idx]
```

```
21
22      return Xmean, eigvec, eigval
```

这里其实将两部分需要填写的合为了一部分：求取特征均值 Xmean 和求取协方差矩阵 covD 因为其结构相似，将其合并可加快运算速度。

为防止内存占用过大，将 nlearn 切分成若干个 slicesize 大小。

4.3.2 run_triemb_learn 全局函数

根据实验原理填充如下

Listing 2: run_triemb_learn.py

```
1  # ......Some Code......
2
3  size = vtrain.shape[0]
4  centers = np.zeros((kc, 128), dtype=np.float32)
5  vtrain = vtrain.T
6  rand_points = np.random.randint(low=0, high=size, size=kc)
7  for k in  range(kc):
8      centers[k] = vtrain[rand_points[k]]
9  while True:
10     ave = np.zeros((kc, 128), dtype=np.float32)
11     counts = np.zeros(kc, dtype=np.int32)
12     closest = np.argmin(
13         np.linalg.norm(np.array([vtrain - centers[j] for j in
               range(kc)]), axis=2),
14         axis=0,
15     )
16     for i, closest in  enumerate(closest):
17         ave[closest] += vtrain[i]
18         counts[closest] += 1
19     ave = ave / counts.reshape(-1, 1)
20     if np.allclose(ave, centers):
21         break
22     centers = ave
23 centers = centers.T
24 vtrain = vtrain.T
25 # ......Some Code......
```

使用自带的 numpy 自带的 allclose() 函数确定中心点是否变化

4.3.3 triemb_sumagg 函数

根据实验原理填充如下

Listing 3: run_embedding.py

```
def triemb_sumagg(X, C, Xmean):
    """Realize that each column vector in X is concatenated after making a
       difference with k cluster centers,
    and perform L2 normalization on each 128-dimension of the concatenated
       column vector separately, and return
    the processed result. Finally, sum aggregation."""

    # ......Some Code......

    Y = np.zeros(D)
    X = X.T
    C = C.T
    for j in  range(kc):
        Y[j * d : j * d + d] = np. sum(
            (X - C[j]) / np.linalg.norm(X - C[j], axis=1).reshape(n, 1),
               axis=0
        )
    Y = Y.reshape(D, 1)
    Y -= n * Xmean
    return Y
```

4.4 结果展示

K 取不同值时的运行结果如下

```
[--- Compute the scores ---]
[ Results for varying powerlaw ]
Holidays    k = 2    d = 128    pw = 1.00    map = 0.347
HoLidays    k = 2    d = 128    pw = 0.70    map = 0.370
Holidays    k = 2    d = 128    pw = 0.50    map = 0.370
Holidays    k = 2    d = 128    pw = 0.30    map = 0.368
```

```
Holidays    k = 2    d = 128    pw = 0.20    map = 0.359
Holidays    k = 2    d = 128    pw = 0.00    map = 0.279

[--- Compute the scores ---]
[ Results for varying powerlaw ]
Holidays    k = 4    d = 384    pw = 1.00    map = 0.554
Holidays    k = 4    d = 384    pw = 0.70    map = 0.550
Holidays    k = 4    d = 384    pw = 0.50    map = 0.545
Holidays    k = 4    d = 384    pw = 0.30    map = 0.525
Holidays    k = 4    d = 384    pw = 0.20    map = 0.508
Holidays    k = 4    d = 384    pw = 0.00    map = 0.456

[--- Compute the scores ---]
[ Results for varying powerlaw ]
Holidays    k = 8    d = 896    pw = 1.00    map = 0.598
Holidays    k = 8    d = 896    pw = 0.70    map = 0.594
Holidays    k = 8    d = 896    pw = 0.50    map = 0.590
Holidays    k = 8    d = 896    pw = 0.30    map = 0.587
Holidays    k = 8    d = 896    pw = 0.20    map = 0.582
Holidays    k = 8    d = 896    pw = 0.00    map = 0.562

[--- Compute the scores ---]
[Results for varying powerlaw ]
Holidays    k = 16    d = 1920    pw = 1.00    map = 0.637
Holidays    k = 16    d = 1920    pw = 0.70    map = 0.645
Holidays    k = 16    d = 1920    pw = 0.50    map = 0.647
Holidays    k = 16    d = 1920    pw = 0.30    map = 0.640
Holidays    k = 16    d = 1920    pw = 0.20    map = 0.636
Holidays    k = 16    d = 1920    pw = 0.00    map = 0.615
```