# k-means 和 PCA 上机实践

**授课教师：庞善民**

**助教：张浩、刘卓**

**2023 年 4 月 23 日**

# 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

# K-means



**算法步骤：**

**输入：数据$\{x_1, x_2, \dots, x_n\}$，簇的数目$K$**

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

2、开始如下迭代

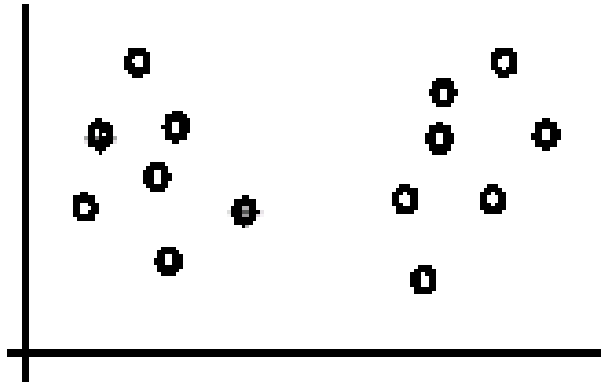
a 对每一个样本$x_j$进行归簇，距离哪个聚类中心最近，则将其归为哪一簇：$x_j \in C_i \Leftrightarrow \min\limits_{t=1,\dots,K}\{\|x_j - \mu_t\|\} = \|x_j - \mu_i\|$

b 重新计算每个簇$C_i$的均值：$\mu_i = \frac{1}{|C_i|}\sum_{x_j \in C_i} x_j$，将更新后的均值作为新的簇中心

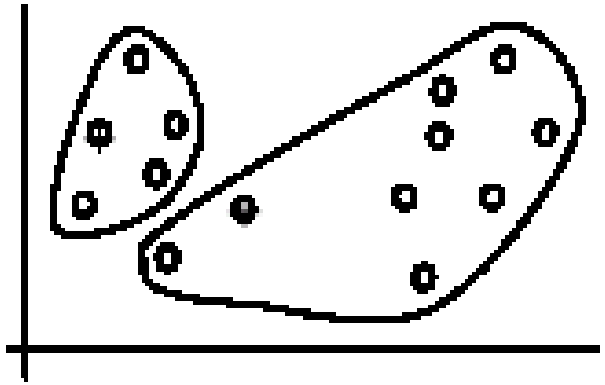
3、簇中心不发生改变时终止迭代。

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

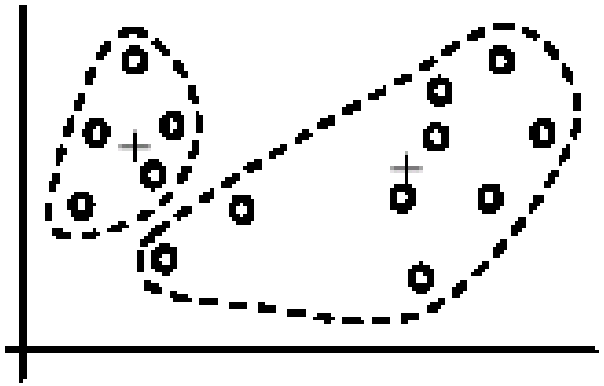
例子



(A). Random selection of $k$ centers

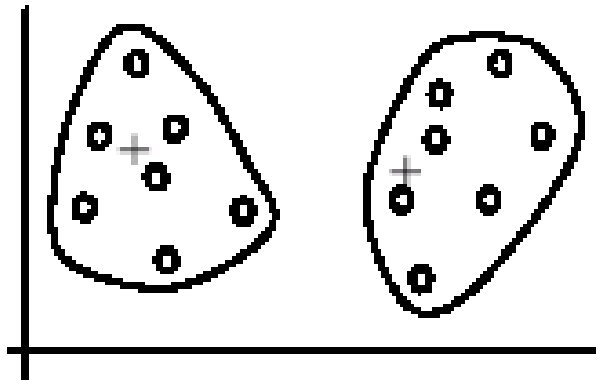


*Iteration* 1: (B). Cluster assignment

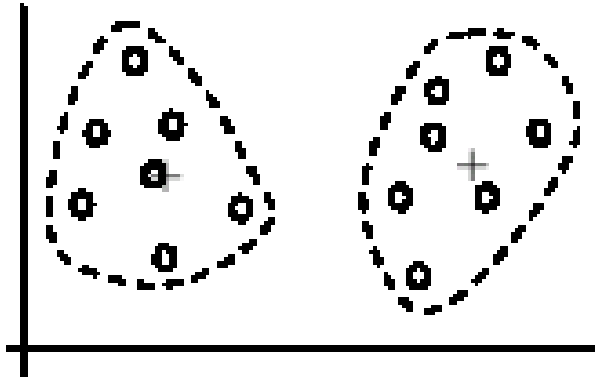


(C). Re-compute centroids

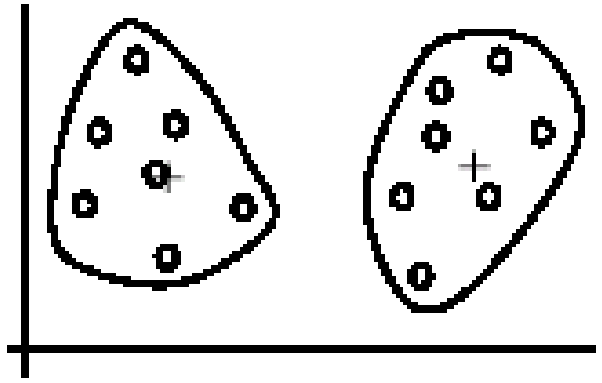
# K-means



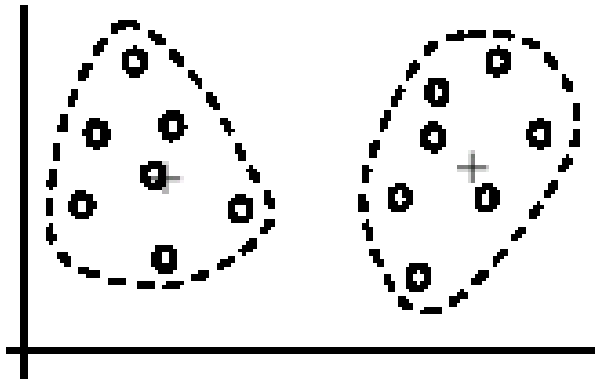
*Iteration* 2: (D). Cluster assignment



(E). Re-compute centroids



*Iteration* 3: (F). Cluster assignment



(G). Re-compute centroids

# PCA应用：数据预处理

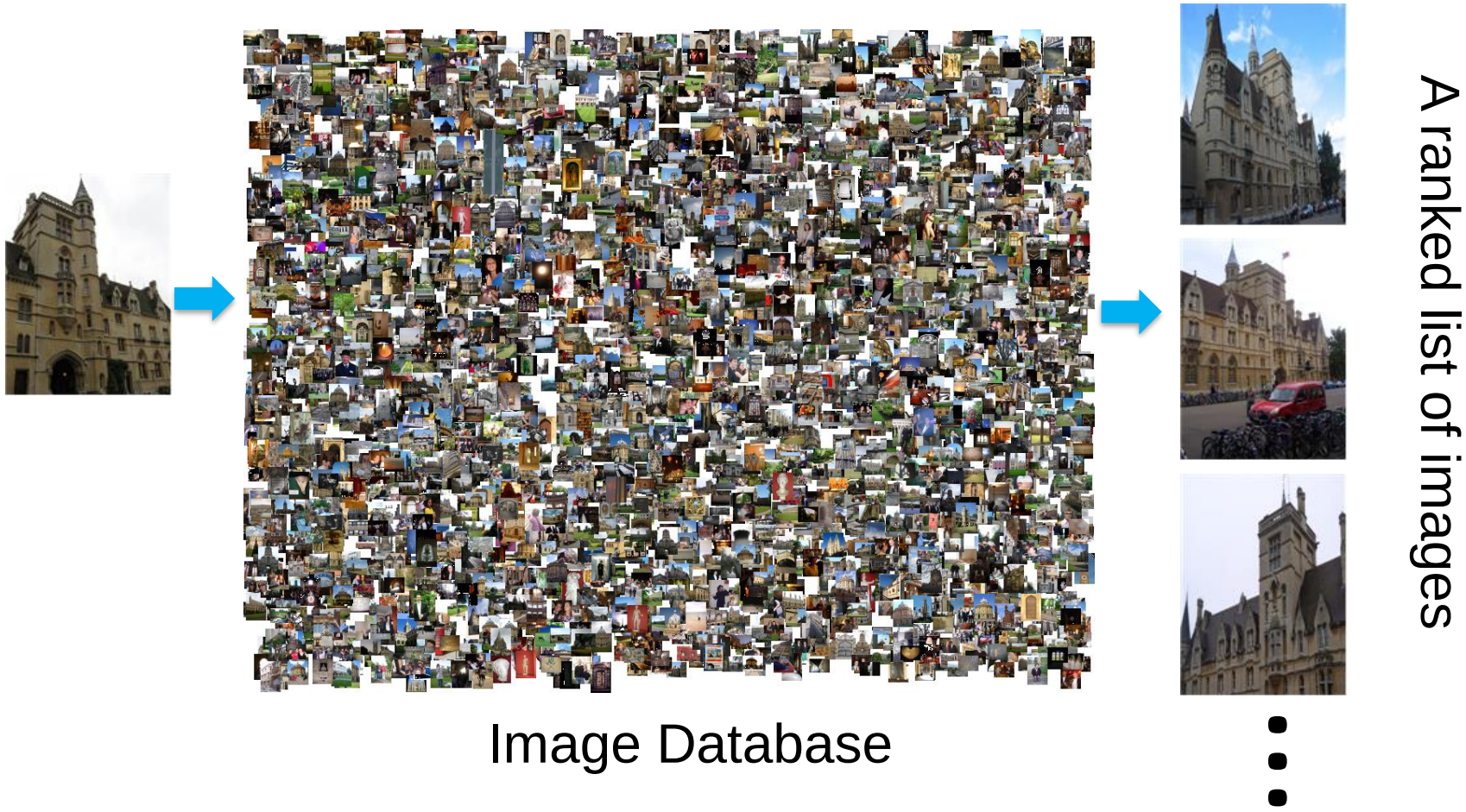


- 使用PCA，可以同时去除变量之间的线性关系以及对数据进行归一化：
  - 假设数据$\{x_1, x_2, \dots, x_m\}$的协方差矩阵为$S = \frac{1}{m}\sum_{i=1}^{m}(x_i - \overline{x})(x_i - \overline{x})^T$
  - 利用$W^T S W = \Lambda$，定义一个变换

$$y_i = \Lambda^{-\frac{1}{2}} W^T (x_i - \overline{x})$$

  - 则$\{y_1, y_2, \dots, y_m\}$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

# 基于内容的图像检索



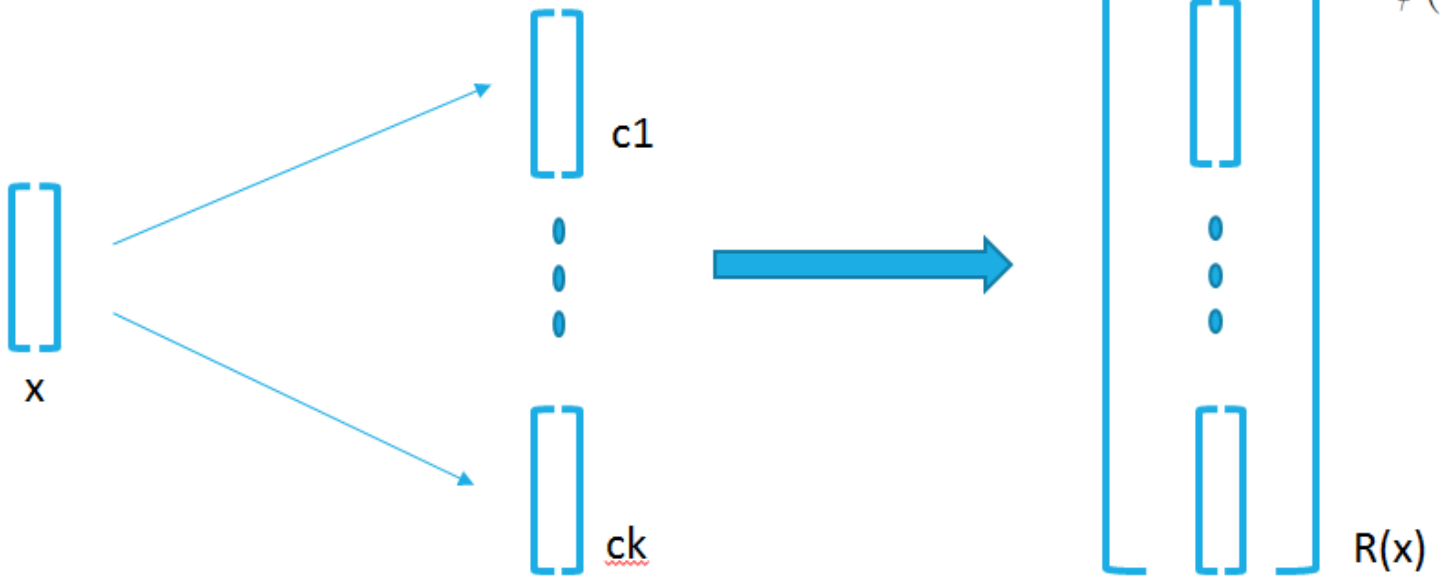

核心问题：计算图像表达

# 图像表达


## 三角化嵌入

$$\mathrm{r}_j(x) = \left\{ \frac{x - \mathrm{c}_j}{\|x - \mathrm{c}_j\|} \right\} \quad \text{for} \quad j = 1 \ldots |\mathcal{C}|,$$

$$\mathrm{R}(x) = [r_1(x)^\top, \ldots, r_{|\mathcal{C}|}(x)^\top]^\top$$




## Sum pooling

$$\psi(\Phi_\triangle(\mathcal{X})) = \sum_{x \in \mathcal{X}} \phi_\triangle(x)$$

$$= \Sigma^{-1/2} \left( \sum_{x \in \mathcal{X}} \mathrm{R}(X) \right) - n\Sigma^{-1/2}\mathrm{R}_0.$$

$$\phi_\triangle(x) = \Sigma^{-1/2}(\mathrm{R}(x) - \mathrm{R}_0),$$

# INRIA Holidays 数据集简介



Holidays 公开数据集侧重风景图片，根据场景的变化分割成 500 组，涵盖自然风景、人工建筑等方面的高分辨率图像，其中也包含具有干扰的图像，包括旋转、角度、光照变化和不同程度的模糊等干扰，以测试算法的稳定性和鲁棒性。

该公开数据集一共 1491 张图，500 张 query（即用来检索的图片，一张图对应一个组）和对应的 991 张相关图像，已提取了 128 维的 SIFT 点 4455091 个。

*Sample images below from INRIA Holidays dataset





# Flickr60k 数据集简介

在 Holidays 公开数据集中，待检索图像和与之相关的图像涵盖了较多的场景，但对于实际需要检索的图像而言，**其场景覆盖度仍然是有限的**，不能保证实际检索图像涵盖于数据集内，因此在训练聚类中心时需要采用 Flickr60k 数据集中提取到的 SIFT 描述子进行训练。

该公开数据集已提取了 128 维的 SIFT 点 5000404 个，以此作为训练数据求取 k 个聚类中心和投影矩阵，最终**将这些结果用于表征 Holidays 公开数据集中的图像**。

*Sample images below from Flickr Image dataset

## 第一部分 run_triemb_learn

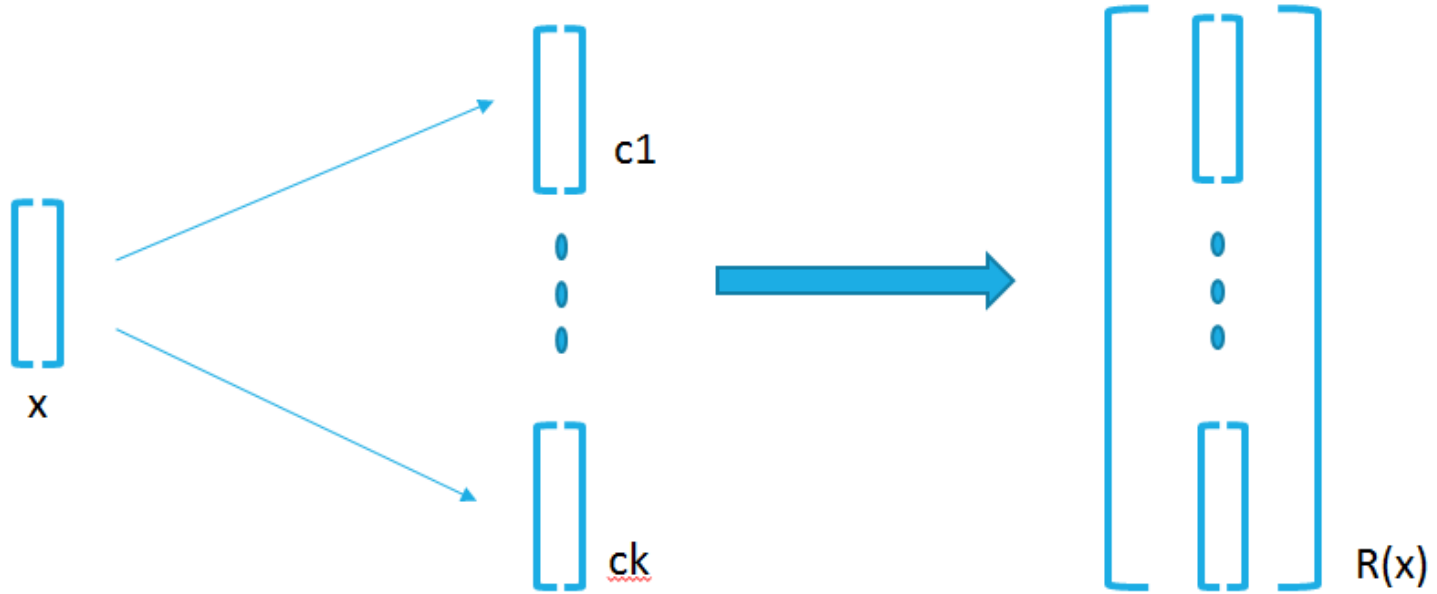
- 导入 Flickr60k 训练数据 vtrain.mat 为 vtrain，格式为 numpy. ndarray，(128，5000404)，已经过 desc_postprocess 函数处理。
- 利用 k-means 聚类算法，找出 k 个聚类中心 C，k 为超参数。
- 利用 SIFT 描述子信息 vtrain 和聚类中心 C，求出三角化嵌入后的特征值 eigval，特征向量 eigvec，特征均值 Xmean(128*k，1)。
- 将得到的聚类中心 C，特征值 eigval，特征向量 eigvec，特征均值 Xmean 以及投影矩阵 Pemb 保存为本地文件。

# 图像表达


$$r_j(x) = \left\{ \frac{x - c_j}{\|x - c_j\|} \right\} \quad \text{for} \quad j = 1 \ldots |\mathcal{C}|,$$

$$R(x) = [r_1(x)^\top, \ldots, r_{|\mathcal{C}|}(x)^\top]^\top$$

## 三角化嵌入



# PCA应用：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：
  - 假设数据$\{x_1, x_2, \dots, x_m\}$的协方差矩阵为$S = \frac{1}{m}\sum_{i=1}^{m}(x_i - \overline{x})(x_i - \overline{x})^T$
  - 利用$W^T S W = \Lambda$，定义一个变换

$$y_i = \Lambda^{-\frac{1}{2}} W^T (x_i - \overline{x})$$

  - 则$\{y_1, y_2, \dots, y_m\}$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


$$\psi(\Phi_\triangle(\mathcal{X})) = \sum_{x\in\mathcal{X}} \phi_\triangle(x)$$

$$= \Sigma^{-1/2}\left(\sum_{x\in\mathcal{X}} \mathrm{R}(X)\right) - n\Sigma^{-1/2}\mathrm{R}_0.$$

$$\phi_\triangle(x) = \Sigma^{-1/2}(\mathrm{R}(x) - \mathrm{R}_0),$$

## 第二部分 run_embedding

- 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。

## 第三部分 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 。

**注：该部分已有完整代码，不需要对其进行改动。**

# 第三方库用途



## import mat73

- 实现对 `.mat` 文件的读取。

## import time

- 实现计时功能。

## import os

- 实现对当前工作路径的切换。

## import warnings

- 实现在终端输出中忽略 `RuntimeWarning` 警告。

## import numpy as np  # 00

- np.reshape(*a, newshape, order='C'*)

  **保持 a 数值不变的情况下赋予新的形状**

```
>>> a = np.arange(6).reshape((3, 2))
>>> a
array([[0, 1],
       [2, 3],
       [4, 5]])
>>> np.reshape(a, (2, 3))
array([[0, 1, 2],
       [3, 4, 5]])
```

- np.linalg.norm(*x, ord=None, axis=None, keepdims=False*)

  **计算矩阵或向量 x 的范数**

```
>>> a = np.arange(9) - 4
>>> a
array([-4, -3, -2, ...,  2,  3,  4])
>>> np.linalg.norm(a)
7.745966692414834
```


**NumPy是Python语言的一个扩展程序库。支持高阶大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。**

## import numpy as np  # 01

- np.zeros(shape, dtype=float, order='C', *, like=None)

  返回 shape 形状的 array，其中的值为 0

```
>>> np.zeros(5)
array([ 0.,  0.,  0.,  0.,  0.])
>>> np.zeros((2, 1))
array([[ 0.],
       [ 0.]])
```

- np.random.randn(*d0, d1, ..., dn*)

  返回随机生成的指定形状的标准正态分布，其均值为0，方差为1

```
>>> np.random.randn()
2.192387535537315
>>> 3 + 2.5 * np.random.randn(2, 4)
array([[-4.49401501,  4.00950034,
        -1.81814867,  7.29718677],

       [ 0.39924804,  4.68456316,
        4.99394529,  4.84057254]])
```

NumPy是Python语言的一个扩展程序库。支持高阶大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。

## import numpy as np  # 02

- np.dot(a, b, out=None)

  **两个数组的点积**

```
>>> np.dot(3, 4)
12
>>> a = [[1, 0], [0, 1]]
>>> b = [[4, 1], [2, 2]]
>>> np.dot(a, b)
array([[4, 1],
       [2, 2]])
```

- np.argmax(a, axis=None, out=None, *, keepdims=<no value>)

  **返回沿某一轴的最大值的索引**

```
>>> b = np.arange(6)
>>> b
array([0, 5, 2, 3, 4, 5])
>>> np.argmax(b)
1
```

NumPy

**NumPy是Python语言的一个扩展程序库。支持高阶大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。**

# 代码部分



## 完形填空

- 本次实验设置了四处缺失的代码，主要涉及 k-means， PCA 及图像表示。

```
os.chdir(os.path.dirname(__file__))  # change the work directory

timeStart = time.time()
data_dict = mat73.loadmat('./data/vtrain.mat')
vtrain = data_dict['vtrain']  # (128, 5000404) attention whether need transpose befor clustering
print("* Descriptors loaded and processed in %.3fs" % (time.time() - timeStart))

timeStart = time.time()
# ********************************************************************** #
# CODING HERE 利用 k-means 聚类算法，找出 kc 个聚类中心，C.shape = (128, kc)   #
# ********************************************************************** #



print("* kmeans cluster centers processed in %.3fs" % (time.time() - timeStart))
```

# 代码部分



## test_image_search.py

- 完成全部代码补写之后，运行该文件实现对图像检索结果的评价。

```
kc = 2

if __name__ == "__main__":

    print("\n[--- Learning the parameters from Flickr ---]")
    import run_triemb_learn

    print("\n[--- Compute image representation - Holidays ---]")
    import run_embedding

    print("\n[--- Compute the scores ---]")
    import img_eval
```

# 部分实验结果



## 聚类中心数量 k = 2

```
[--- Compute the scores ---]
[ Results for varying powerlaw ]
Holidays    k = 2    d = 128    pw = 1.00    map = 0.355
Holidays    k = 2    d = 128    pw = 0.70    map = 0.380
Holidays    k = 2    d = 128    pw = 0.50    map = 0.387
Holidays    k = 2    d = 128    pw = 0.30    map = 0.376
Holidays    k = 2    d = 128    pw = 0.20    map = 0.368
Holidays    k = 2    d = 128    pw = 0.00    map = 0.290
```

## 聚类中心数量 k = 64

```
[--- Compute the scores ---]
[ Results for varying powerlaw ]
Holidays    k = 64    d = 8064    pw = 1.00    map = 0.681
Holidays    k = 64    d = 8064    pw = 0.70    map = 0.699
Holidays    k = 64    d = 8064    pw = 0.50    map = 0.702
Holidays    k = 64    d = 8064    pw = 0.30    map = 0.704
Holidays    k = 64    d = 8064    pw = 0.20    map = 0.703
Holidays    k = 64    d = 8064    pw = 0.00    map = 0.686
```

# 部分实验结果

## 不同聚类中心数量时的实验结果

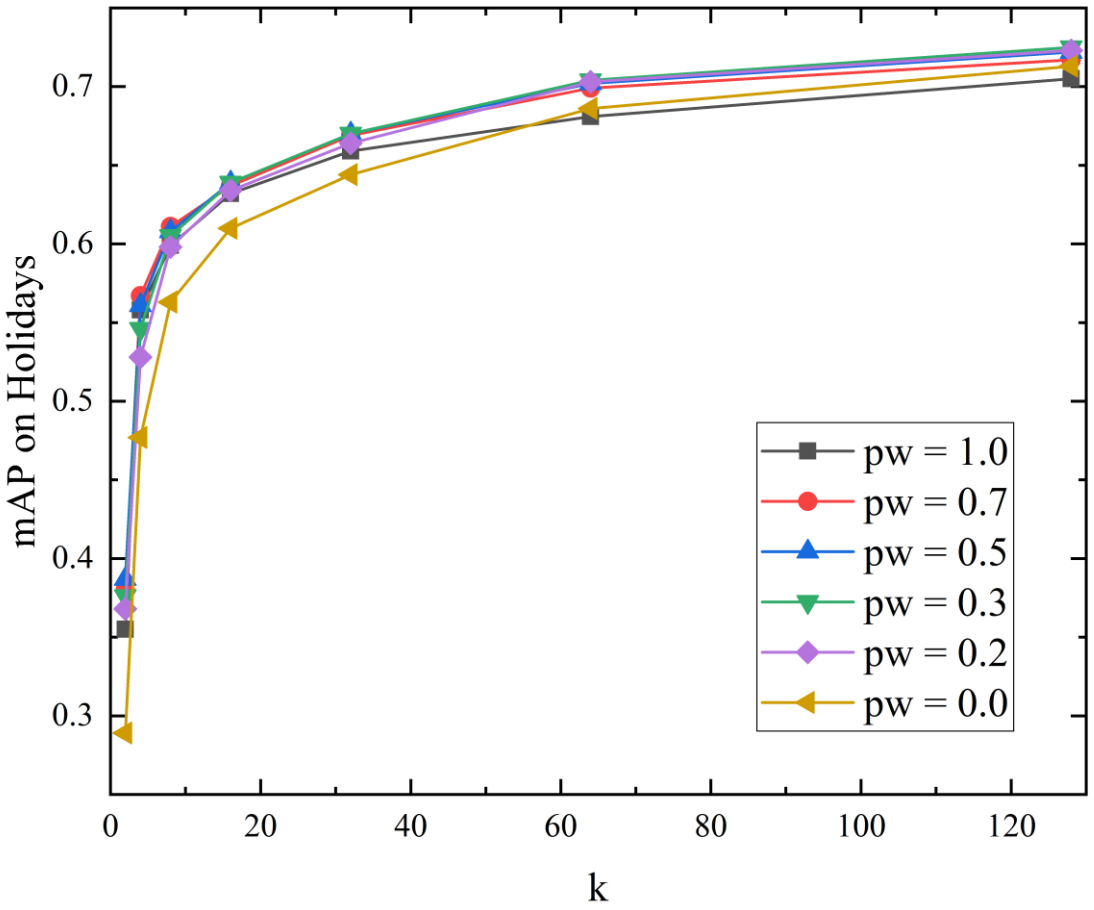



图1 mAP随聚类中心数量k的变化曲线

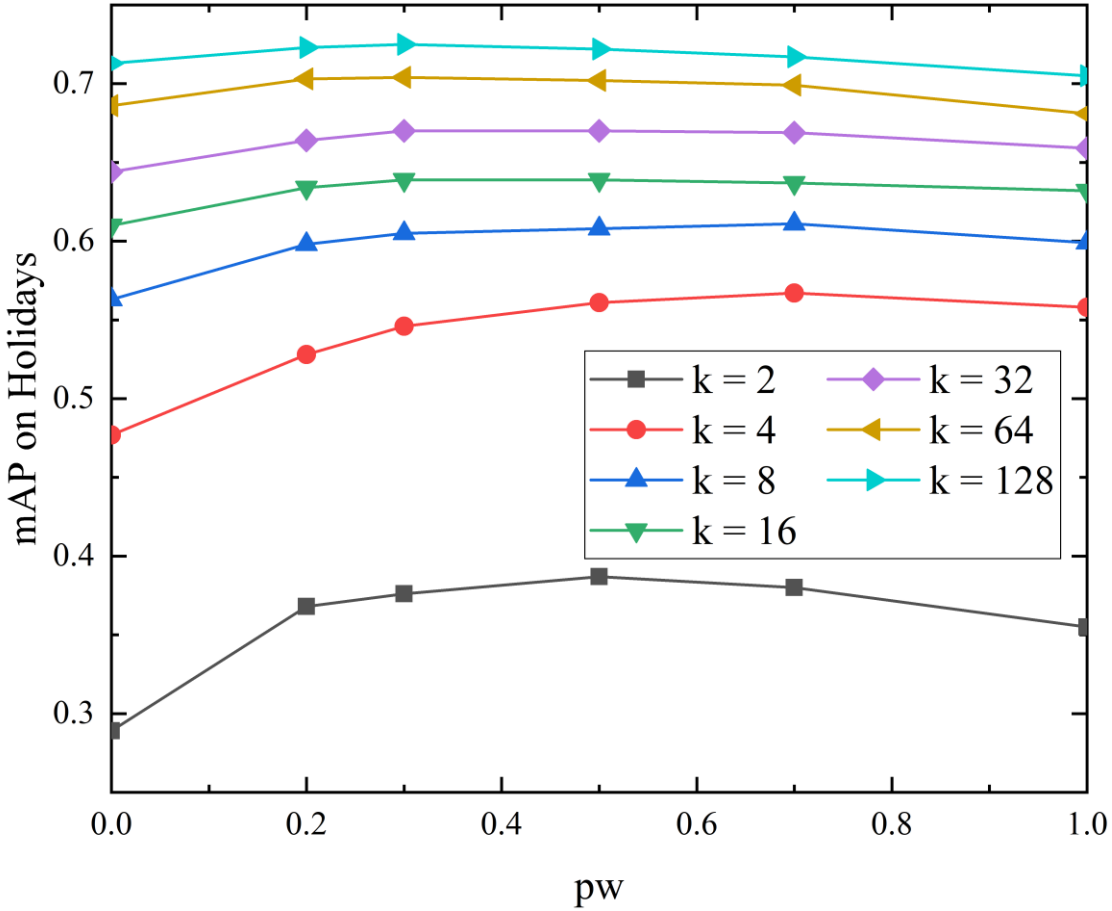



图2 mAP随幂定律归一化参数pw的变化曲线

# Thank You　Q & A

张浩：1050852440@qq.com

刘卓：lzpmbw@163.com

2023 年　4 月　23 日