无监督学习-降维

ML08

礼欣
www.python123.org

NMF方法及实例

非负矩阵分解（NMF）

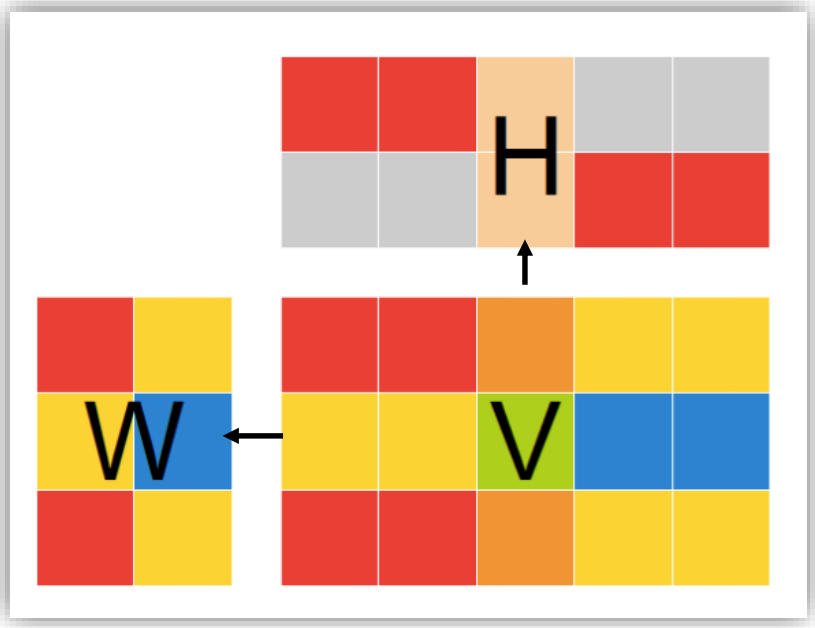
非负矩阵分解（Non-negative Matrix Factorization ，NMF）是在矩阵中所有元素均为非负数约束条件之下的矩阵分解方法。

基本思想：给定一个非负矩阵V，NMF能够找到一个非负矩阵W和一个非负矩阵H，使得矩阵W和H的乘积近似等于矩阵V中的值。

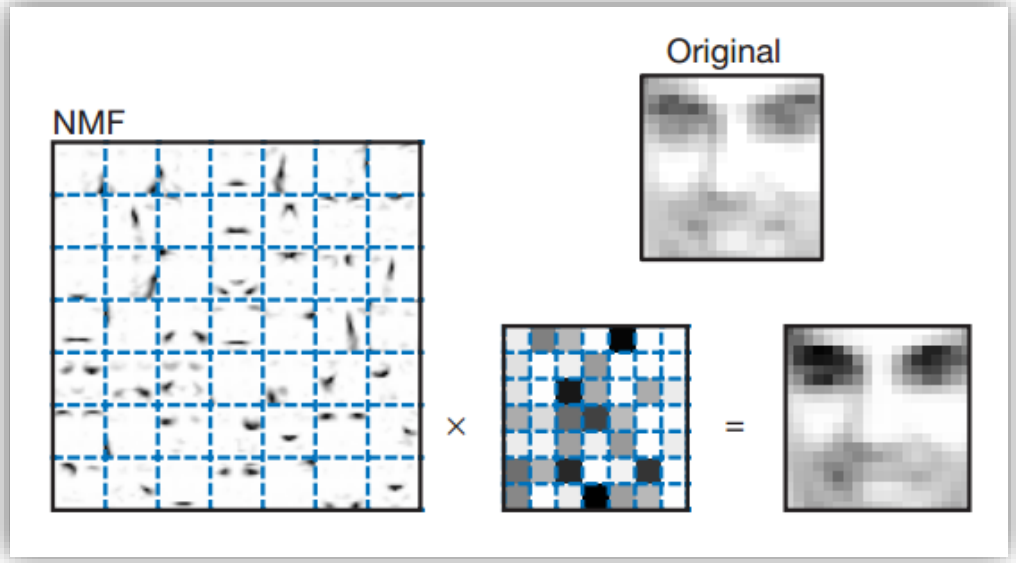
$$\boldsymbol{V_{n*m} = W_{n*k} * H_{k*m}}$$

非负矩阵分解

- W矩阵：基础图像矩阵，相当于从原矩阵V中抽取出来的特征
- H矩阵：系数矩阵。
- NMF能够广泛应用于图像分析、文本挖掘和语音处理等领域。



非负矩阵分解



上图摘自NMF作者的论文，左侧为W矩阵，可以看出从原始图像中抽取出来的特征，中间的是H矩阵。可以发现乘积结果与原结果是很像的。

非负矩阵分解

矩阵分解优化目标：最小化W矩阵H矩阵的乘积和原始矩阵之间的差别，目标函数如下：

$$\boldsymbol{argmin}\frac{\boldsymbol{1}}{\boldsymbol{2}}\|\boldsymbol{X}-\boldsymbol{WH}\|^{\boldsymbol{2}}=\frac{\boldsymbol{1}}{\boldsymbol{2}}\sum_{\boldsymbol{i,j}}\left(\boldsymbol{X_{ij}}-\boldsymbol{WH_{ij}}\right)^{\boldsymbol{2}}$$

非负矩阵分解

基于KL散度的优化目标，损失函数如下：

$$\boldsymbol{argmin}\, \boldsymbol{J}(\boldsymbol{W},\boldsymbol{H}) = \sum_{\boldsymbol{ij}} \left(\boldsymbol{X_{ij}} \boldsymbol{ln} \frac{\boldsymbol{X_{ij}}}{\boldsymbol{WH_{ij}}} - \boldsymbol{X_{ij}} + \boldsymbol{WH_{ij}} \right)$$

非负矩阵分解

至于公式的推导，就不在课程中讲述了，有兴趣的同学可以参考下面的链接。

参考链接：http://blog.csdn.net/acdreamers/article/details/44663421/

sklearn中非负矩阵分解

在sklearn库中，可以使用sklearn.decomposition.NMF加载NMF算法，主要参数有：

- n_components：用于指定分解后矩阵的单个维度k；
- init：W矩阵和H矩阵的初始化方式，默认为‘nndsvdar’。

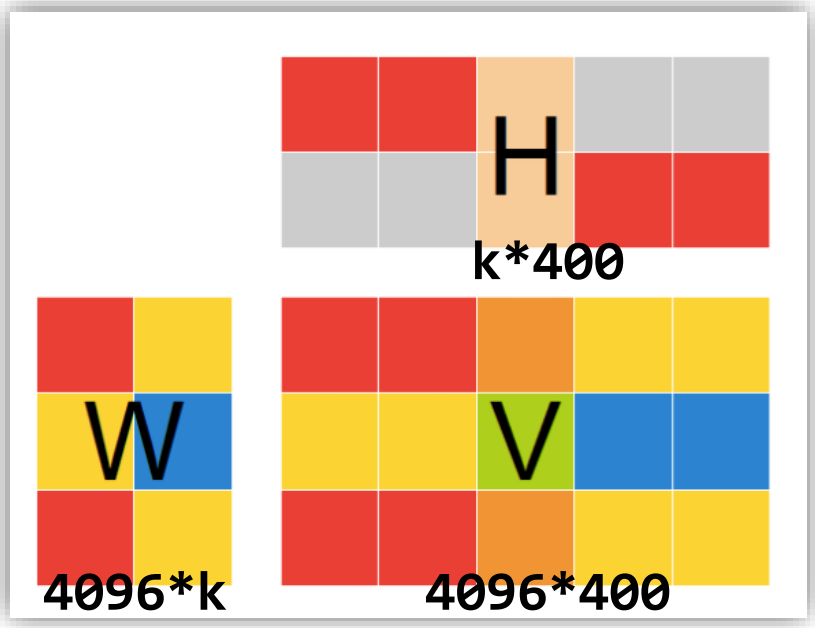
NMF人脸数据特征提取

目标：已知Olivetti人脸数据共400个，每个数据是64*64大小。由于NMF分解得到的W矩阵相当于从原始矩阵中提取的特征，那么就可以使用NMF对400个人脸数据进行特征提取。

图. 人脸图像数据

NMF人脸数据特征提取

通过设置k的大小，设置提取的特征的数目。在本实验中设置k=6，随后将提取的特征以图像的形式展示出来。



实例程序编写

1. 建立工程，导入sklearn相关工具包：

```
>>> import matplotlib.pyplot as plt
#加载matplotlib用于数据的可视化
>>> from sklearn import decomposition
#加载PCA算法包
>>> from sklearn.datasets import fetch_olivetti_faces
#加载Olivetti人脸数据集导入函数
>>> from numpy.random import RandomState
#加载RandomState用于创建随机种子
```

实例程序编写

2. 设置基本参数并加载数据：

```
>>> n_row, n_col = 2, 3
#设置图像展示时的排列情况，如右图
>>> n_components = n_row * n_col
#设置提取的特征的数目
>>> image_shape = (64, 64)
#设置人脸数据图片的大小
>>> dataset = fetch_olivetti_faces(shuffle=True,
                random_state=RandomState(0))
>>> faces = datasets.data   #加载数据，并打乱顺序
```

实例程序编写

3. 设置图像的展示方式：

```
def plot_gallery(title, images, n_col=n_col, n_row=n_row):
    plt.figure(figsize=(2. * n_col, 2.26 * n_row))
    plt.suptitle(title, size=16)

    for i, comp in enumerate(images):
        plt.subplot(n_row, n_col, i + 1)
        vmax = max(comp.max(), -comp.min())

        plt.imshow(comp.reshape(image_shape), cmap=plt.cm.gray,
                   interpolation='nearest',
                   vmin=-vmax, vmax=vmax)
        plt.xticks(())
        plt.yticks(())
    plt.subplots_adjust(0.01, 0.05, 0.99, 0.93, 0.04, 0.)
```

设置标题及字号大小

创建图片，并指定图片大小（英寸）

选择画制的子图

对数值归一化，并以灰度图形式显示

去除子图的坐标轴标签

对子图位置及间隔调整

实例程序编写

3. 创建特征提取的对象NMF，使用PCA作为对比：

```
>>> estimators = [
    ('Eigenfaces - PCA using randomized SVD',
      decomposition.PCA(n_components=6,whiten=True)),
    ('Non-negative components - NMF',
      decomposition.NMF(n_components=6, init='nndsvda',
       tol=5e-3))]
#将它们存放在一个列表中
```

提取方法名称

NMF和PCA实例

实例程序编写

4. 降维后数据点的可视化：

```
>>> for name, estimator in estimators: #分别调用PCA和NMF
        estimator.fit(faces) #调用PCA或NMF提取特征
        components_ = estimator.components_
        #获取提取的特征
        plot_gallery(name, components_[:n_components])
        #按照固定格式进行排列
>>> plt.show()
#可视化
```

结果展示

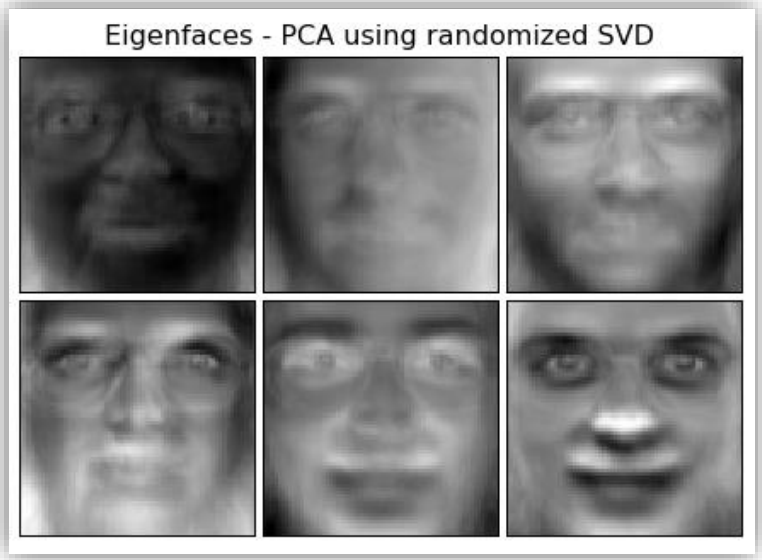


图. PCA提取的特征

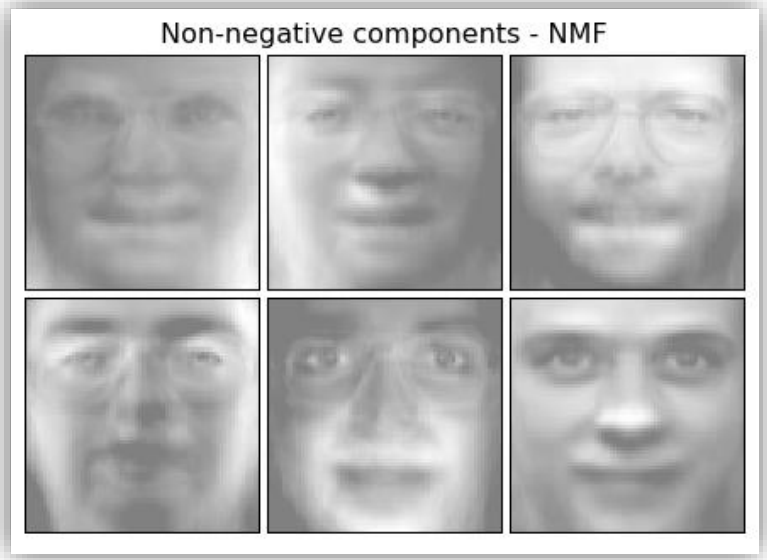


图. NMF提取的特征