**字典学习和稀疏表示**

**——数值计算方法大作业设计**

组员1：余鹏 学号：15352394

组员2：袁子豪 学号：15352396

组员3：詹巽霖 学号：15352399

组员4：张海涛 学号：15352405

组员5：张镓伟 学号：15352408

院系：数据科学与计算机

专业：软件工程(移动信息工程)

指导老师：纪庆革

组员合照：

****

从左到右依次为：张镓伟、余鹏、詹巽霖

张海涛 袁子豪

2017年5月28日

1. **引言**

字典学习和稀疏表示一般被称为稀疏字典学习。该算法包括两个阶段：字

典构建阶段和利用字典洗漱的表示样本阶段。如下图：

成千上万的样本

第一阶段：字典的产生

第二阶段：样本的稀疏表示

以字典表示样本

大量字典

**为什么要进行字典学习？**

首先说一下字典的来源，我们知道人类社会的一切知识都必然要通过句子来表示出来，但无论有多少句子要被书写，对于每一个句子来说，它都有一个最本质上的特征，那就是构成这个句子字。可以这样说，无论人类的知识多么渊博，一本小小的字典就足以表达所有的知识，那些知识只不过是字典中的字的组合而已。所以我们看到字典学习的好处，它实质上是对于庞大数据集的一种降维表示。另外，正如同字是句子的最质朴的特征一样，字典学习总是尝试学习蕴含在样本背后最质朴的特征。所以我们要探讨字典学习。

**为什么需要稀疏表示？**

我想大家都有过这样的体会，在接触一个新的知识点时会有一种非常累的感觉，但是当我们把这些知识点完全掌握后，再次遇到相同的问题就会感觉到轻松，这是因为和初次接触知识点时相比，在完全掌握知识点后大脑可以调动尽可能少的脑区消耗尽可能少的能量进行同样有效的计算，此时大脑的计算速度也会变快，可以这样说，这是因为大脑学会了知识的稀疏表示。因为稀疏表示的本质就是用尽可能少的资源表示尽可能多的知识，这种表示还能带来计算上的的加快。

**二、字典学习的方法**

字典学习称之为稀疏编码。从矩阵分解角度看字典学习过程：给定样本数据集Y，Y的每一列表示一个样本；字典学习的目标是把Y矩阵分解成D、X矩阵：

IMG_256

同时满足约束条件：X尽可能稀疏，同时D的每一列是一个归一化向量。

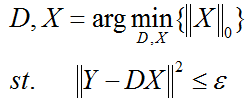
D称之为字典，D的每一列称之为原子；X称之为编码矢量、特征、系数矩阵；字典学习可以有三种目标函数形式

(1)第一种形式：

IMG_257

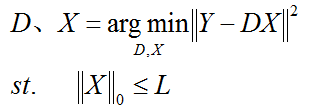
这种形式因为L0难以求解，所以很多时候用L1正则项替代近似。

(2)第二种形式：



ε是重构误差所允许的最大值。

(3)第三种形式：



L是一个常数，稀疏度约束参数，上面三种形式相互等价。

**SRC基本模型：**

1、对x进行编码，对系数加l1范数约束，取最小值。

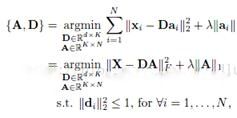


2、对x进行分类判决。



其中δ(a) 是提取与第i个类别相关的元素的向量指示函数。

**传统字典学习框架：**



其中，共N个信号。A=[a1,a2,...,aN]是编码系数矩阵。矩阵A的1范数等价于A的各个列向量的1范数之和。

但SRC方法是有缺点的：

1、预定字典包含冗余和琐碎的不利于人脸识别的信息。

2、训练数据增加，稀疏编码的计算量增加。

为了解决以上的问题，可以使用以下两类已经存在的基于DL的方法：一类是直接学习具有识别力的字典，另一类是稀疏化稀疏，是字典具有识别力。

**1、直接学习具有识别力的字典**

**Meta-Face Learning方法：**

该方法是针对每一个类别学习得到一个自适应的字典。

C:\Users\Admin\Documents\Tencent Files\1107676552\Image\C2C\91C46AD84CDB1D7F83FCA4B0550966F9.jpg

其中矩阵Xi包含第i个类别的所有样本，Di是第i个类别对应的字典。

**Dictionary Learning with Structured Incoherence（DLSI)方法：**

不同类型的子字典具有连贯性，则重构查询图像时的原子是可以互相代替的。这导致无法用重构误差进行判决，为了解决这一问题，Ramirez等人增加了一个不连贯项的约束，使不同类型的子字典之间尽可能互相独立。

不连贯项：



最终的字典学习算法为：

C:\Users\Admin\Documents\Tencent Files\1107676552\Image\C2C\74D54368B19B91358EA25566133AAA96.jpg

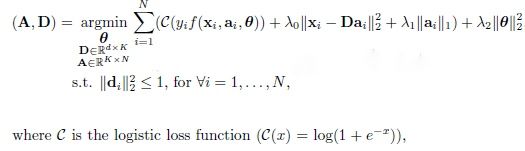
不连贯项的含义是：在重构误差是，忽略与公共原子相关的系数的绝对值，一次提高系统的判决能力。

**2、使系数具有识别力的方法**

该类方法使稀疏系数具有识别力，间接使字典具有识别力，该类方法只需要学习一个整体的字典，不需要每个类别都学习一个相应的字典。

**监督字典学习（SDL）方法：**

Mairal等人提出讲逻辑回归与传统字典学习框架结合。优化公式为：



λ2是防止过拟合的正则项参数，f是与系数a呈线性关系的函数：

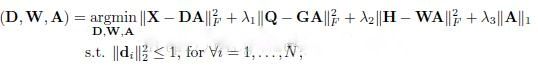


或者与a成双线性关系的函数：



**用于字典学习的具有识别力的K-SVD方法：**

D-KSVD在传统DL框架上加了一个简单的线性回归作为惩罚项。



H=[h1,h2,……hN]是训练图像的标签。hn=[0,……0,1,0……0],非零元素的位置即为所属的类别。W是分类器的参数。由上式可是，前两项可以混合为一项，最后一项根据KSVD可以去掉。最终优化得到D和W后，即可迅速对查询图像进行分类。

**接下来详细介绍K-SVD方法**

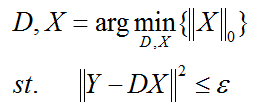
K-SVD是字典学习的一种经典[算法](http://lib.csdn.net/base/31)，其求解方法是固定其中一个，然后更新另外一个变量，交替迭代更新。字典D的每一列就相当于一个聚类中心，稀疏编码X的每一列允许有几个非零元素，给定训练数据Y,Y的每一列表示一个样本，我们的目标是求解字典D的每一列。

1. **随机初始化字典D：**

从样本集Y中随机挑选k个样本作为D的原子，并且初始化编码 矩阵X为0矩阵。

**2.固定字典，求取每个样本的稀疏编码：**

编码过程可以采用如下公式：



其中ε是重构误差所允许的最大值。

假设我们的单个样本是向量y，为了简单起见我们就假设原子只有这4个，也就是字典D=[α1、α2、α3、α4]，且D是已经知道的；我们的目标是计算y的编码x，使得x尽量的稀疏。

(1)首先从α1、α2、α3、α4中找出与向量y最近的那个向量，也就是分别计算点乘：

α1\*y、α2\*y、α3\*y、α4\*y

然后求取最大值对应的原子α。

(2)假设α2\*y最大，那么我们就用α2，作为我们的第一个原子，然后我们的初次编码向量就为：

x1=（0,b,0,0）

b是一个未知参数。

(3)求解系数b：y-b\*α2=0

方程只有一个未知参数b，是一个高度超静定方程，求解最小二乘问题。

(4)然后我们用x1与α2相乘重构出数据，然后计算残差向量：y’=y-b\*a2

如果残差向量y’满足重构误差阈值范围ε，那么就结束，否则进入下一步。

(5)计算剩余的字典α1、α3、α4与残差向量y’的最近的向量，也就是计算

α1\*y’、α3\*y’、α4\*y’

然后求取最大值对应的向量α，假设α3\*y’为最大值，那么就令新的编码向量为：

x2=（0,b,c,0）

b、c是未知参数。

(6)求解系数b、c,于是我们可以列出方程：

y-b\*α2-c\*α3=0

方程中有两个未知参数b、c，我们可以进行求解最小二乘方程，求得b、c。

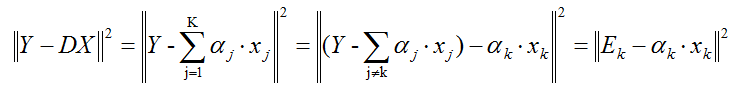
(7)更新残差向量y’：

y’=y-b\*α2-c\*α3

如果y’的模长满足阈值范围，那么就结束，否则就继续循环，就这样一直循环下去。

**3、逐列更新字典，并更新对应的非零编码**

通过上一步我们已经知道样本的编码，接着我们的目标是更新字典，同时还要更新编码，K-SVD采用的方法是更新字典，就是更新第k列原子的时候其他的院子固定不变，假设我们当前更新第k个原子ak，矩阵X对应的第k行为xk，则目标函数是：



我们需要注意的是xk不是把X一整行都拿出来更新（因为Xk中既有另元素又有非零元素，如果全部取出来的话那么计算的时候xk就不再保持以前的稀疏性），所以我们只能抽取出非零的向量。然后Ek只保留xk对应的非零元素项。

对于上面的方程，我们可以通过求解最小二乘的方法来求解αk，不过这样有存在一个问题，我们求解的αk不是一个单位向量，因此我们需要采用svd分解，才能得到单位向量αk。进过svd分解后，我们以最大奇异值所对应的正交单位向量，作为新的αk，同时我们还需要更新系数编码xk中的非零元素(根据svd分解)。然后算法就在1和2之间一直迭代更新，直到收敛。

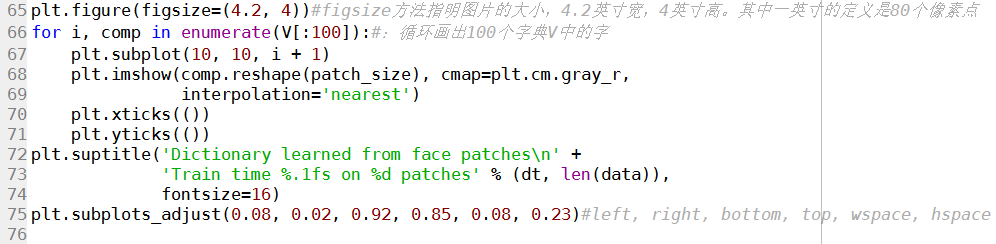
**三、字典学习和稀疏表示的演示和实现**

下面使用基于python3.5实现的代码演示，测试平台为Anaconda 的spyder IDE。代码及意义如下：

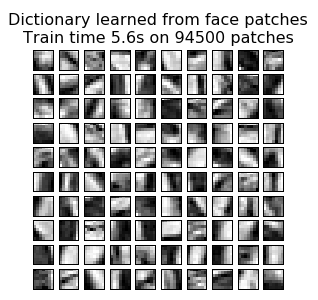
第一步先导入各种工具包和测试样例，此测试样例为scipy库自带的图片



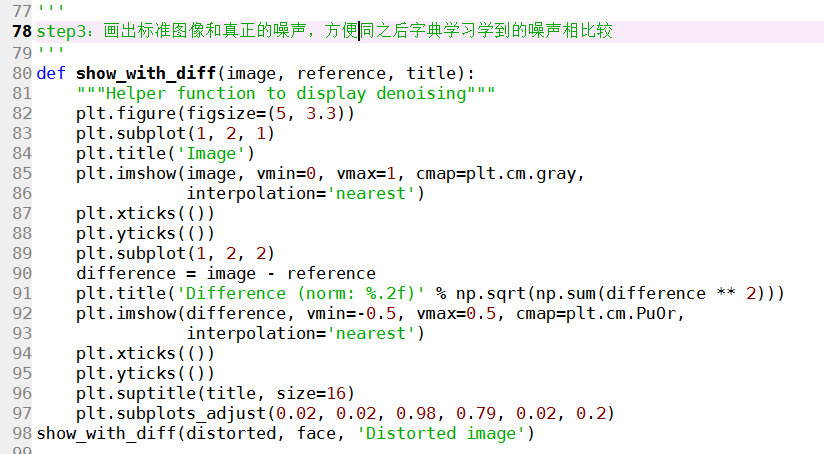
第二步计算样例图片的字典V

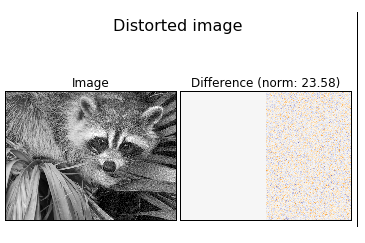


本步骤之后可以看到字典V如下：

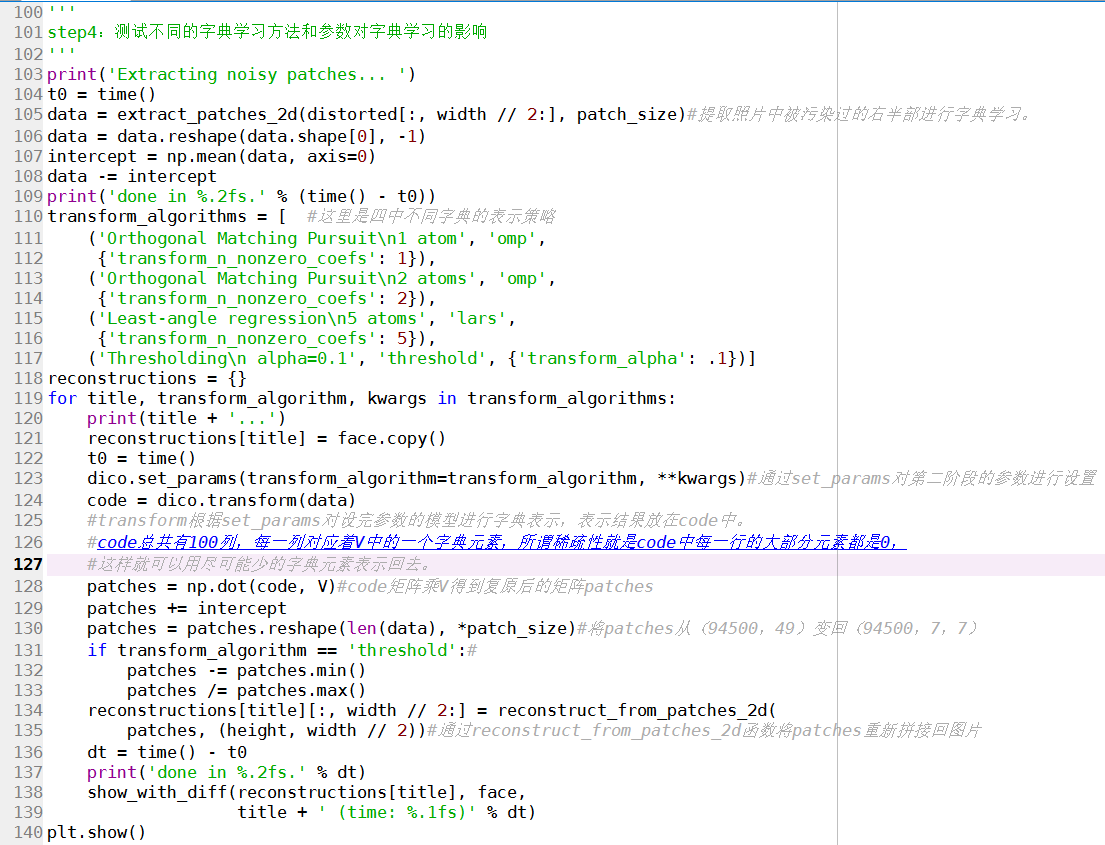


第三步我们画出标准图像和真正的噪声，方便同之后字典学习学到的噪声相比较

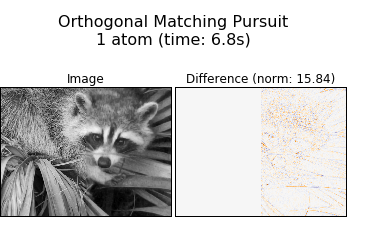


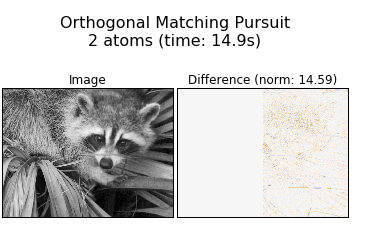


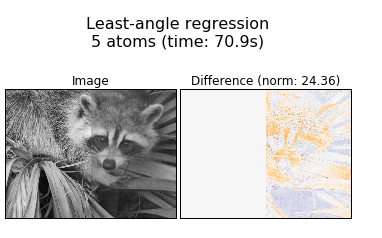
第四步我们测试不同字典学习方法对字典学习的影响

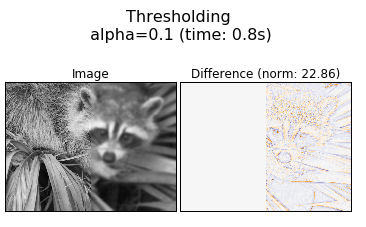


下面是4种不同的字典学习方法的结果：









**提交附件**

test.py 该算法的实现代码