2021.11.14周报

纪新龙

本周计划任务

继续复现“基于L0稀疏优化的本征图像分解”方法。若复现好，就转入读一段时间的论文。

具体完成情况

完成

下周计划任务

复现丁守鸿的第二篇论文

具体完成情况

- [x] 复现了反射率L0约束项，但是没跑通
- [x] 初始化r的方法

复现过程

复现了反射率L0约束项原先复现L0项的时候因为写了一个小bug，没看出来那个bug，导致自己根本不清楚离正确复现还有多远。不清楚自己错在理论推导上（矩阵运算、卷积运算里的求导等等）还是代码编写上。发现并改正代码的那个小bug之后，代码能跑起来了，对理论和实践两方面的信心大增，进而复现出整个算法。

在复现代码和阅读别人代码中遇到了一些疑惑。通过网上查资料、和同门同学探讨，在理论学习方面，我重温了矩阵运算方程的变换、求导，傅立叶变换的一些性质，卷积运算，求一阶、二阶梯度的滤波算子。之前很少用matlab，通过这两周密集地边写边学，学习开源的代码，也更学会了很多写法。

为了复现论文我更仔细地查阅了一些文献：

- 5_4_Image Smoothing via L0 Gradient Minimization（有关于L0约束项实现的代码）
- 5_5_Mean_shift_a_robust_approach_toward_feature_space_analysis

前一篇论文是我在不确定L0约束该怎么复现的时候查阅的论文。这篇论文里详细介绍了L0项比L1项的优点，比如尺度无关性。并且证明了L0项为什么可以通过引入辅助变量与原被约束变量交替迭代来优化。同时还用L0来做图像平滑，附赠了代码。他用傅立叶变换来加速对原被约束变量的求解，该思想和刘聪学长的论文以及他引用的*Contrast Enhancement Based on Intrinsic Image Decomposition*TIP2017等论文中一模一样。我在看到之后回想了之前在阅读刘聪论文时在此处遗留下来的疑惑。对照之后发现是刘聪论文里的描述有误，原本几篇英文论文中的$F^*(...)$是“共轭”的意思，刘聪学学长的论文中将它解释为“共轭转置”，可能是因为在右上角加符号太像转置了。

后一篇论文讲的是meanshift这种聚类方法。丁守鸿的论文里用meanshift来分割图像，*然后通过边界两边的点光照相同*来建立约束，并用类似最小二乘的方法建立求解的矩阵方程，求出一个好的初始r。我它通过略读这篇论文了解了meanshift的思想和优点，然后从matlab exchange社区找到一份meanshift的实现代码，然后自己补写了边界检测、构建几个关键矩阵的代码，最终思想*r的初始化这一部分*。

几个最优化方法：

1. 直接使用了多篇论文里用过的 $minimize(...)$ 函数，它有三个方法参数:

```
%     p.method     minimization method, 'BFGS', 'LBFGS' or 'CG'
```

- 梯度下降法：
  沿着梯度（**一阶**）反方向下降到最低点。
- 共轭梯度法：
  CG是介于梯度下降法与牛顿法之间的一个方法，它仅需利用**一阶导数**信息，但克服了梯度下降法收敛慢的缺点，又避免了牛顿法需要存储和计算 Hessian 矩阵并求逆的缺点。
  CG 中，寻求一个和先前先搜索方向共轭的搜索方向，即它不会撤销该方向上的进展。
- 牛顿法：
  牛顿法是基于二阶泰勒级数展开在某点 θ_0 附近来近似$J(\theta)$ 的优化方法，忽略了高阶导数。对于局部的二次函数，牛顿法会直接跳到极小值处。如果目标函数是凸的，但有高阶项，那么该更新是迭代的。
  - 牛顿法的优点：因为利用了**二阶信息**，相比较梯度下降法，**下降速度更快**。
  - 牛顿法的缺点：如果参数数目为 k ，那么需要计算 k × k 矩阵的逆，**算法复杂度**是$O(k^3)$。
- BFGS：
  BFGS 是一种拟牛顿法，使用矩阵 M_t 近似逆，迭代的更新精度以更好的近似 H^{-1}。
  - BFGS 优点是花费较少的时间改进每个线搜索。
  - BFGS 必须存储 Hessian 逆矩阵 M ， 需要 $O(n^2)$ 的存储空间。
- L-BFGS ：
  和 BFGS 相同的方法计算 M 。
  但是假设 M_{t-1} 是单位矩阵，而不需要每一步都**存储**近似值。
  每步存储一些用于更新 M 的向量，每步的存储代价是 O(n) 。

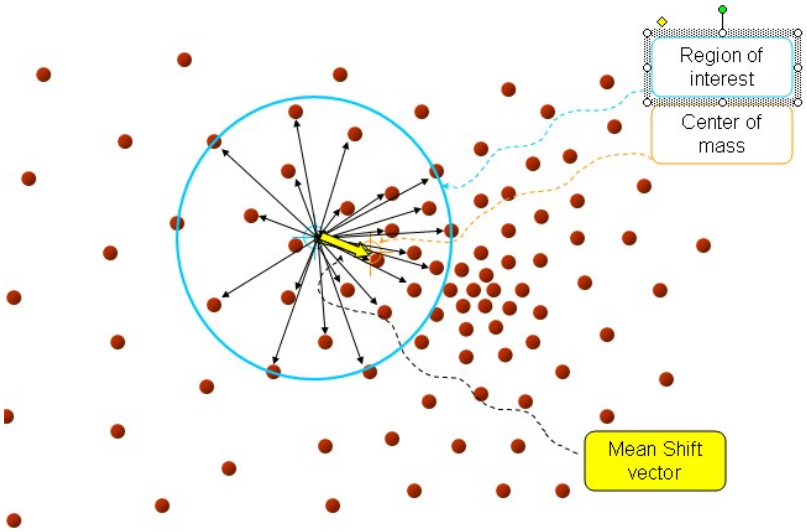
我使用的函数有三种方法可选，分别是BFGS 、L-BFGS 、CG。我们图像太大了，不建议用BFGS，我试了一下另外两种，LBGFS每次迭代次数更少。

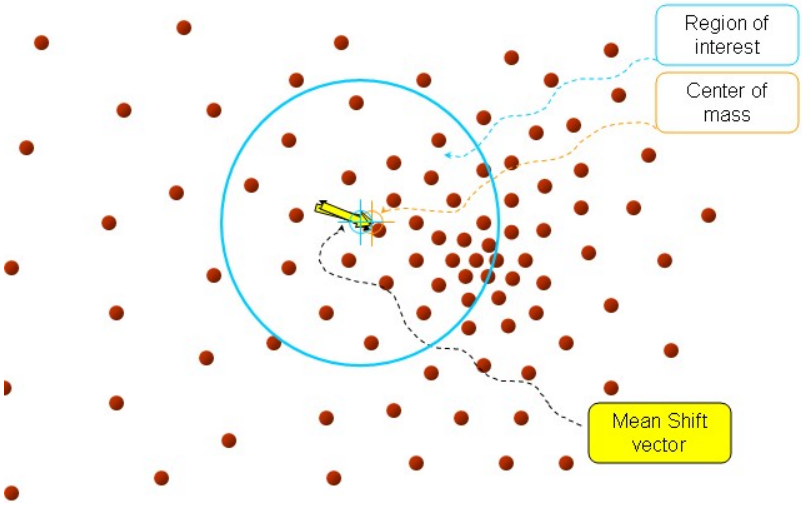
对两种聚类方法的比较分析

- k-means方法，输入是希望最终生成的聚类的数目k。
- meanshift方法，输入是带宽bandwidth，象征每个类的大小。

> k-means步骤
> 1、随机选取k个点，并将作为第一轮迭代的中心点；
> 2、计算每一组数据到k个点中的距离，并将每一个点分配到k个簇中；
> 3、更新中心点：将k个簇中的每一个点的各属性的值进行加和，并计算均值，得到新的中心点；
> 4、多次迭代，直到中心点的值不再发生变化。

> meanshfit步骤
> 在d维空间中，任选一个点，然后以这个点为圆心，h为半径做一个高维球。落在这个球内的所有点和圆心都会产生一个向量，向量是以圆心为起点落在球内的点位终点。然后把这些向量都相加。相加的结果就是Meanshift向量。
> 再以meanshift向量的终点为圆心，再做一个高维的球。如下图所以，重复以上步骤，就可得到一个meanshift向量。如此重复下去，meanshift算法可以收敛到概率密度最大得地方。也就是最稠密的地方。如下图。





- k-means算法优缺点：

1、优点

（1）原理易懂、易于实现；

（2）当簇间的区别较明显时，聚类效果较好；

2、缺点

（1）当样本集规模大时，收敛速度会变慢；

（2）对孤立点数据敏感，少量噪声就会对平均值造成较大影响；

（3）k的取值十分关键，对不同数据集，k选择没有参考性，需要大量实验；

- mean-shift优缺点：

1、优点：

（1）不需要设置簇类的个数；

（2）可以处理任意形状的簇类；

（3）算法只需设置带宽这一个参数，带宽影响数据集的核密度估计

（4）算法结果稳定，不需要进行类似K均值的样本初始化

2、缺点：

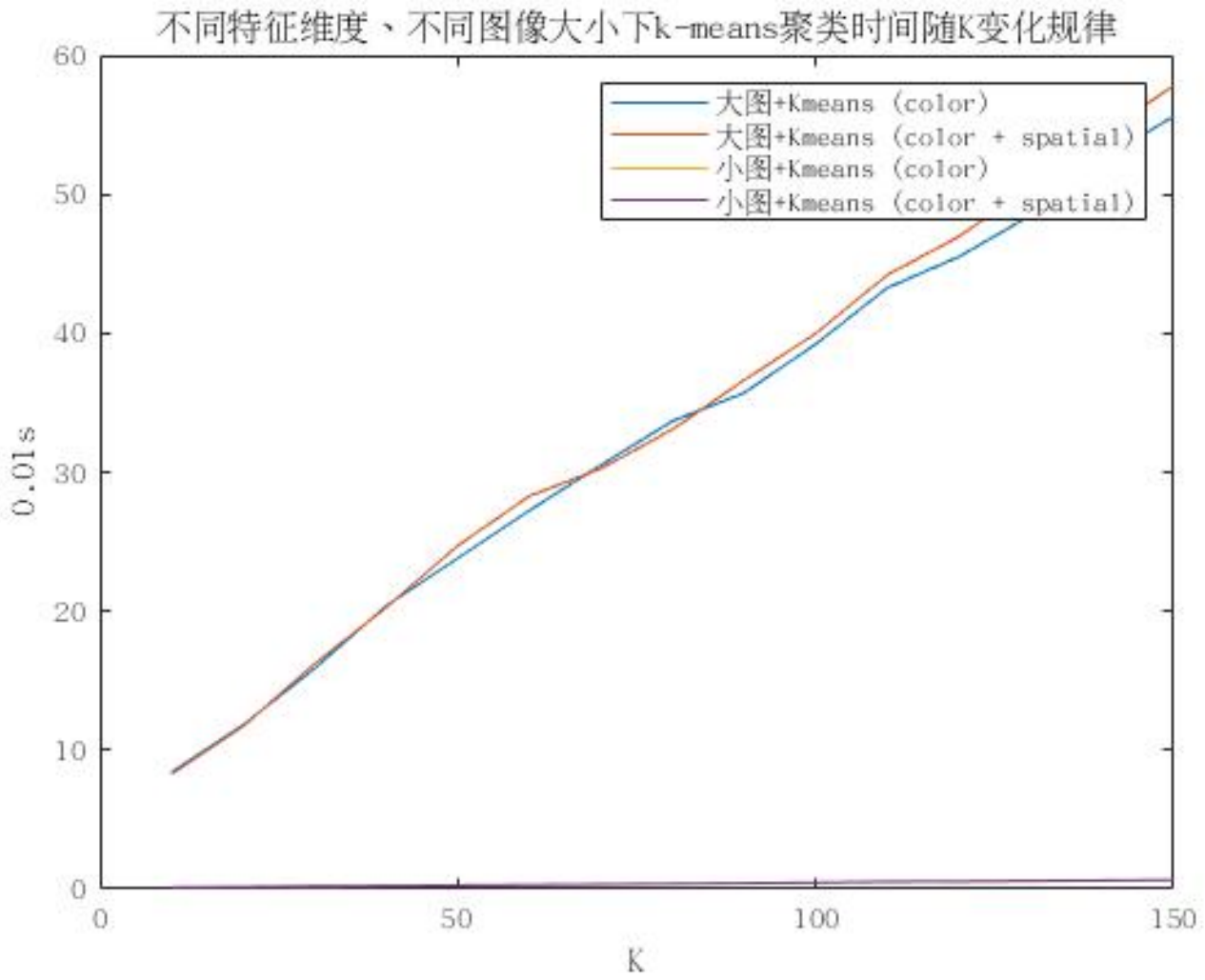
（1）聚类结果取决于带宽的设置，带宽设置的太小，收敛太慢，簇类个数过多；带宽设置的太大，一些簇类可能会丢失。

（2）对于较大的特征空间，计算量非常大。

可能正是因为k-means的*聚类结果不稳定，需要样本初始化*，丁守鸿的在先前方法的基础上强调对r做一个初始化，然后再做k-means聚类的初始化。也就是说，*对r的初始化*不仅仅有利于迭代求解的时候避开局部最优解，还有利于k-means的样本初始化。

由于色块数量比较少的全局性稀疏先验经常被用到，图像经常被进行聚类、色彩分割。我设计了一些实验帮助体会两种聚类方法的特征，改变**输入图片的大小、特征维度、聚类算法的参数设定**，观察**耗费的时间和聚类的效果**。

首先是对K-means的测试，横坐标是K=10,20,30...150，四条曲线分别是不同特征维度（一种是色彩信息，RGB三维；另一种是色彩+位置，5维信息）、不同图片尺寸（小图片58*61；大图片534*600）的输入。对丁守鸿实验里那张经典的太阳头图片来说，它的色块比较少，我自己观察过，聚类的结果总体上看是比较稳定的，我说的稳定是指分割的区域都正确。无非就是k更大、输入特征维度越大，聚类的结果会更精细，大区域会被还分为更小的区域。k的取值对k-means来说很重要，所以我们这里改变k观测运行时间。

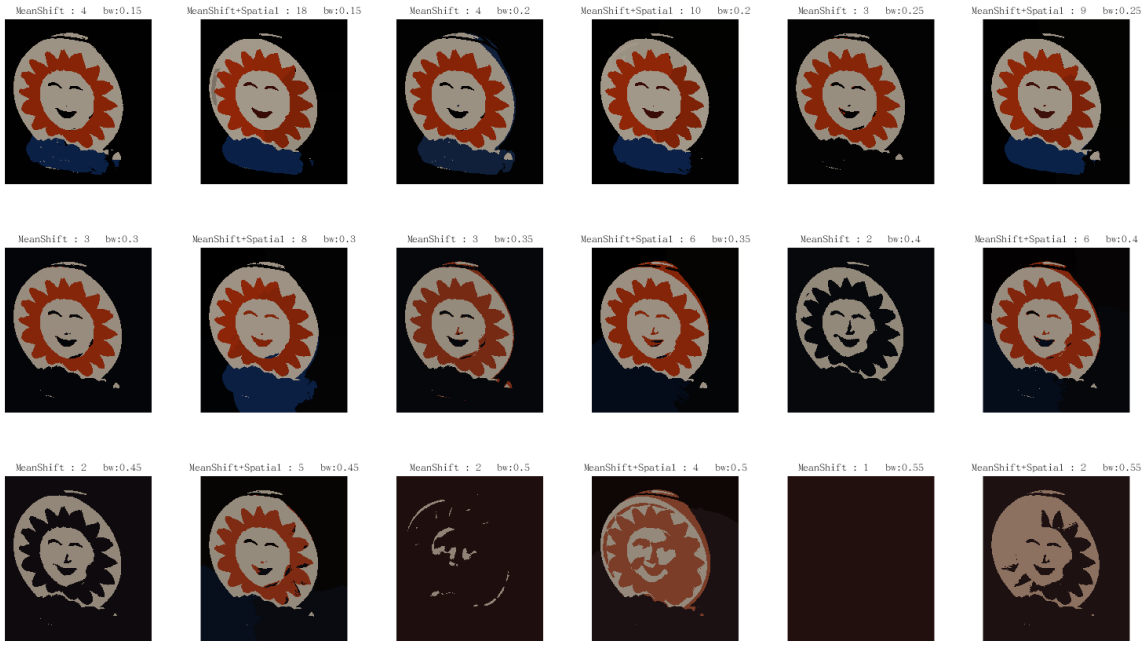


- 可以看出，**特征维度**从3扩大到5没有对k-means算法的耗时产生什么影响。
- k变大，要求最后聚类的个数更多，这才让耗时随着k的增加成正比地增加。k增加4倍时，耗时约增加2倍。其实它的效果是比较稳定的。

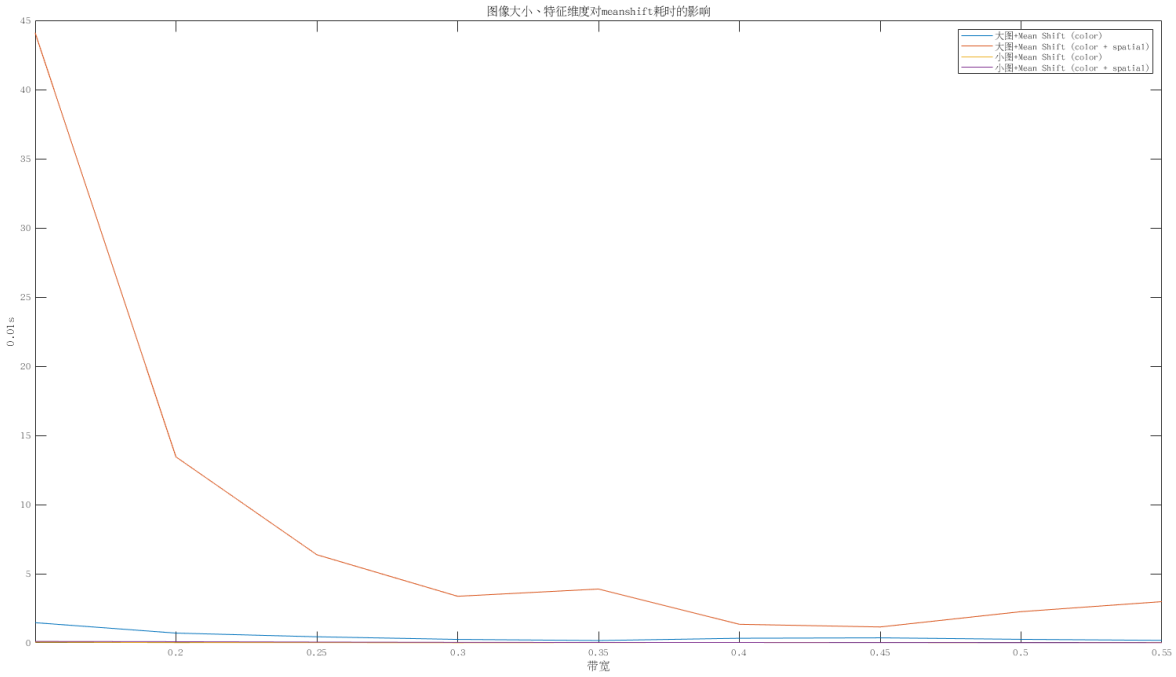
- 影响最大的是图片大小。尺寸扩大10*10倍，导致待聚类的点数增加100倍，经而导致复杂度约增加100*100倍，耗时飙升。
- 实际取k=10比较合理，此时大图的特征是3维还是5维对时间影响不大，可以根据需求来确定特征维数。此时耗时是**0.08s**这样。

meanshift方法：

特征维数、图像尺寸变化如前。带宽设置从0.15-0.55，步长为0.05。这是不同参数产生的聚类效果图，能够直观看出bandwidth在0.15-0.35之间的色彩聚类效果是差不多的，而且特征的维数对效果的影响也不大。bandwidth在0.40~0.45的时候，由于带宽太大，聚合的数目被逼着减少，此时三维特征的色彩效果已经失真了，但是区域分割还是能看出来的，能够勉强用在图像分割场景。此时五维特征在由于多参照了距离，聚类效果和之前一样好。bandwidth继续大到0.55附近时，此时不论用三维还是为五维，聚类结果都已经不可信的。**总的来说，meanshift的效果受bandwith影响最大，bandwith不能太大，这样能满足基本的结果可信，扩充特征的维度有利于在bandwith偏大时勉强维持结果，但是仍然特征维度不是对结果影响最大的因素。**



下面看看耗时：



图中明显高于其他三条曲线的是“大图+color特征+位置特征”组合。比较有意思的是，另一条曲线“大图+color特征”同样是大图，耗时却能因为少用了空间上的两维信息而将耗时迅速降到小图的两个组合附近。我们前面观测到bandwidth在0.15-0.35之间效果是比较好的，那么结合这张曲线图来看，总的来说bandwidth的扩大有利于减少耗时，那么我们不妨定点观测bandwith为0.3的时候，此时“大图+color特征+位置特征”组合耗时约0.05s，而同样的大尺寸图，“大图+color特征”的耗时应该是小于0.001s的。这对我们处理稍微大的图像来说是非常有利的。

我们最终的目标是作出能实际应用的代码，那么现实中的图是很大的。所以，针对大图，拿meanshift最好的结果0.001s（bandwith=0.3，三维特征）和k-means最好的结果0.08s（k=10，三维或五维）来比较，人工研判聚类的效果对应用来说差不多，那么肯定是meanshift速度更快。

总结

- k-means 基于点的多层遍历，耗时随着图像尺寸变大上升非常快。所以在实际应用中不要用它来初始化的实际的大图。但是它的好处是特征维度多一点少一点对耗时影响不大。
- meanshift是均值漂移。耗时对图像尺寸的敏感程度不高，图像尺寸大了100*100倍耗时竟然差不多。与k-means恰恰相反，它对特征维度比较敏感。大图中，当bandwidth取值过小时，对图像分割效果没什么太大帮助，反而让时间指数级增长。**所以我们一定要避免bandwidth过小的取值，在结果差不多的情况下选尽量大的带宽、用尽量少的特在维度。**
- 总的来看，实际图像都很大，我们尽量少用点特征，然后用meanshift是很好的。也不排除有些时候必须使用很高维的特征，比如问题本身强制要求考虑像素的位置和颜色，这种时候meanshift比较被动，可以比较小同效果时k-means的效率再决定使用什么方法。
- 我未作这个对比实验时meanshift是读入5维数据的，后边会考虑改成3维数据，反正只用它来做图像分割，后边初始化r还是主要靠约束方程求解。