人工智能课程Project 2实验报告

## 任务与思路

给定三张带噪音的图片（图片可以为黑白或彩色），将其尽可能恢复为原图。其中，噪音的生成方式为，在每个通道的每一行，以的概率将其中的像素值改成0。通常取做40%，60%，80%。“尽可能恢复为原图”定义为恢复图和原图之间的距离，该距离为矩阵的二范数（frobenius范数）。

主要思路如下。

噪点已知。本次实验的图像恢复比通常意义上的图像回归要简单，因为我们可以认为噪点都是已知的。根据噪音的生成算法，某通道的噪点一定是像素值为0的点。由于通常图片上不太可能出现一个通道像素为0的点，因此可以认为，某通道的噪点和某通道像素为0的点是等价的。这样，所有的噪点以及非噪点都是已知的。

局部回归。最近上课讲的是回归，并且示例方法也是一个回归方法，因此首先考虑回归的方法。但是，从示例方法上，明显可以看到按行回归是明显不可行的，因为回归要求对象满足一些潜在的规律，而图像上距离较远的像素之间显然几乎没有潜在的规律。并且，如果只考虑图像的一行，那么其中将有许多高频的信号，这要求我们的回归函数应该是高维的；但同时我们的测试点数量非常有限（最少只有20%），这使得我们的回归很可能出现过拟合的情况。因此，按行回归或者按列回归是有明显缺陷的。局部线性回归是一种可行的方法，首先它只考虑点的一个小邻域，这一小邻域内可以认为是相关的（即满足某种潜在的规律），其次这种规律比较简单，可以用线性刻画。局部线性回归的具体方法将在下文专门阐述。

插值方法。除了回归方法，我们认为插值方法可能是一种更好的策略。由于噪点已知，因此恢复的过程相当于根据已有的像素点插值出未知的像素点（噪点）。一种常用的插值策略是距离反比插值，另一种离散点插值的方法为克里金（kriging）插值，两种方法均会在下文具体阐述。

高斯滤波。用上述方法恢复的图像，点和点总会有一些高频的成分，而在一幅正常的图片中，除了边缘，其他位置应该都是局部低频的。所以，做一做高斯滤波总是有好处的。实验结果表明，高斯滤波确实对结果有较好的影响。

## 方法一：局部线性回归

局部线性回归指，当我们要确定的像素值时，首先构造一个3\*3或5\*5的邻域中的回归函数，再令。比如，邻域中有个有效点，则可构造矩阵



并解得。

从线性函数适用性的角度来看，邻域应该取得越小越好（比如3\*3）；同时，我们又希望能有更多的有效点，使得回归后的函数更加精确。由于大量噪点的存在，3\*3的邻域中甚至不一定能找到3个以上的点来构成回归的参数空间，因此有时不得不将邻域适当扩大（比如5\*5）。

在实验过程中，遍历一次所有像素常常不能将所有点都恢复出来。比如，一个点的3\*3（或5\*5）邻域中可能只有2个有效点，那么它就不能立刻被恢复。它需要等待它领域中的一些噪点被恢复后，再通过这些恢复后的噪点进行恢复。因此，这里需要采用迭代的方法，多次遍历以将所有点都恢复过来。但是，多次迭代非常耗时，因此，迭代一定次数后，就需要降低有效点的个数，并增大邻域大小，这在伪代码中得到体现。

伪代码：

|  |
| --- |
| function img2 = My\_regression(img)  window = 3  leastEq = 6  while 存在噪点  for 目标噪点  if 目标噪点的window \* window邻域中至少有leastEq个有效点  将有效点的坐标和像素值存到矩阵A和列向量b  求解系数向量w  使用w和目标噪点的坐标求出目标噪点的像素值  end  end  if 迭代了 5 10 20 30 40次  window = 3 3 5 5 5  leastEq = 5 4 6 5 4  end  end  end |

源代码详见My\_regression.m，输入待恢复图像img，输出使用局部线性回归方法恢复的图像img2。

该方法的恢复图片和范数分析结果见“结果与比较”一节。

## 方法二：高斯滤波

高斯滤波主要是为了消除线性回归（以及下文的插值方法）中产生的高频部分，它使得有效点周围的点更加平滑，因此能产生比较好的效果。对下文的两种插值后的结果，也将运用高斯滤波来得到更好的效果。

在具体编码中，我使用的高斯卷积核为，虽然并不是由高斯函数推导出来的，但原理类似，且在实际中取得了较好的结果。另外，在实际编码中，要注意不要把原本非噪点的位置做高斯滤波，否则效果肯定下降。

源代码详见My\_gaussian.m，输入待恢复图像img和初始有效点掩码mask，输出高斯滤波后的图像img2。

## 方法三：局部距离平方反比插值

由于噪点已知，因此很自然的想法是用插值而非回归，因为简单的回归将较远的点和较近的点一视同仁，且结果仅为线性，而插值则很容易根据远近确定不同权重，且能产生非线性的结果。一种简单的插值方法是距离反比插值，其数学表达如下：



经过试验，发现取2时效果较好，因此为距离平方反比法。另外，出于效率和实际相关性考虑，这个过程同样是在一个3\*3或5\*5的邻域中进行的，因此为局部距离平方反比法。

在实际编码中，为了防止一个点只受一个临近点影响，从而产生较大误差，要求邻域中至少要有3个点。为了实现这一点，同样需要使用迭代法进行求解，并适时地扩大邻域。这一点在伪代码中有所体现。

伪代码：

|  |
| --- |
| function img2 = My\_invdist(img)  window = 3  leastPixel = 3  while 存在噪点  for 目标噪点  if目标噪点的window\*window邻域中至少有leastPixel个有效点  以距离平方反比为，求出每个有效点的权重  目标噪点的像素值为有效点的加权和  end  end  if 迭代了 10 20次  window = 3 5  end  end  end |

源代码详见My\_invdist.m，输入待恢复图像img，输出使用距离平方反比方法恢复的图像img2。

如上文所述，局部距离平方反比插值恢复的图像，再使用高斯滤波，能得到更好的结果。

该方法的恢复图片和范数分析结果见“结果与比较”一节。

## 方法四：带高斯滤波的局部克里金（kriging）插值

在网格平面上求插值，通常还有二维样条插值、双样条插值等，但是，由于在受损图像中，有效点并不是网格状的，因此很难使用上述的这些方法。经过查找资料，我发现克里金插值法是距离反比插值法的拓展，在一定程度上具有更加优良的性质，因此也使用克里金插值来试试图像恢复的效果。

克里金插值定义为：



可以看出，它的本质还是根据附近的点进行加权平均。但是，它的权重并不是简单生成的，而是基于以下的假设：

* 空间中任何一点，都具有相同的期望和方差。
* 插值应遵循无偏性和方差极小化。

无偏性的数学表达为：



方差极小化的数学表达为：



由这两个条件可以解出，因过程非常复杂，这里不再列出，详见参考文献[1]。在局部，这两个假设可以认为是合理的，因此实际编码中，受损图像将被分成一个个有重叠的小块，在小块内部做克里金插值。而重复计算的像素做一个简单平均，得到最终恢复的图像。

在具体编码时，用到了Mathwork里的开源代码kriging.m，以及kriging.m中需要调用的开源代码variogram.m，variogramfit.m，fminsearchbnd.m，fminsearchcon.m，这里已经附在代码中，也可以到Mathwork官网进行搜索下载。

伪代码：

|  |
| --- |
| function img2 = My\_kriging(img)  blockSize = 12  for 每个12\*12的块（每4像素一块，即相邻两块有8\*12的重叠）  [X, Y] = meshgrid(块的坐标范围)  for 块中的有效点  坐标添加到x向量和y向量，像素值添加到z向量  end  使用克里金插值法求出块中所有点的像素  end  for 每个点  该点多次求得的像素值取简单平均  end  end |

源代码详见My\_kriging.m，输入待恢复图像img，输出使用局部克里金插值方法恢复的图像img2。

如上文所述，局部克里金插值恢复的图像，再使用高斯滤波，能得到更好的结果。

该方法的恢复图片和范数分析结果见“结果与比较”一节。

## 结果与比较

原图：



**受损图与恢复图统计表**

（此表需放大查看，或在文件里查看图片）

|  |  |  |
| --- | --- | --- |
| 受损图（40%噪点） | 受损图（60%噪点） | 受损图（80%噪点） |
|  |  |  |
| Regression (dist: 59.7227) | Regression (dist: 81.3165) | Regression (dist: 108.2604) |
|  |  |  |
| Regression\_Gaussian (dist: 52.6775) | Regression\_Gaussian (dist: 68.8056) | Regression\_Gaussian (dist: 87.9679) |
|  |  |  |
| Inv\_Dist (dist: 54.4288) | Inv\_Dist (dist: 71.1281) | Inv\_Dist (dist: 93.0181) |
|  |  |  |
| Inv\_Dist\_Gaussian (dist: 52.6775) | Inv\_Dist\_Gaussian (dist: 68.8047) | Inv\_Dist\_Gaussian (dist: 87.9248) |
|  |  |  |
| Kriging (dist: 54.6942) | Kriging (dist: 70.5423) | Kriging (dist: 90.1661) |
|  |  |  |
| Kriging\_Gaussian (dist: 52.6775) | Kriging\_Gaussian (dist: 68.8043) | Kriging\_Gaussian (dist: 87.9217) |
|  |  |  |

**距离范数统计表**

|  |  |  |  |
| --- | --- | --- | --- |
| Distance | 40% Noise | 60% Noise | 80% Noise |
| Regression | 59.7227 | 81.3165 | 108.2604 |
| Regression\_Gaussian | 52.6775 | 68.8056 | 87.9679 |
| Inv\_Dist | 54.4288 | 71.1281 | 93.0181 |
| Inv\_Dist\_Gaussian | 52.6775 | 68.8047 | 87.9248 |
| Kriging | 54.6942 | 70.5423 | 90.1661 |
| Kriging\_Gaussian | 52.6775 | 68.8043 | 87.9217 |

**各种方法时间统计表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 80%受损图 | Regression | Inv\_Dist | Kriging | Gaussian |
| 大约耗时 | 1 min | 15s | 20min | 30s |
| 评价 | 慢 | 快 | 极极极极极慢 | 一般 |

## 结论

从结果上看，单纯“局部克里金” > 单纯“局部距离平方反比” > 单纯“局部线性回归”。“局部克里金”由于对邻近有效点的关系做了很多考虑，取得最佳的结果是可以理解的（需要注意的是，由于“局部克里金”非常耗时，因此在局部化的时候采用了较疏的块，如果使用较密的块，结果应该可以进一步优化）。而“局部线性回归”则对邻近点考虑不足，稍远点权重太大，因此结果明显较差。

“高斯滤波”对三者的结果都有一定提升，其中对“局部线性回归”提升非常明显，而对另外两者提升不大。值得注意的是，“高斯滤波”倾向于将之前的结果替换为自己的结果，所以高斯滤波之后，三者虽然还保持着原本的优劣关系，但本质上已几乎没有区别。这样，“局部克里金”效果上的优势就完全被抹去，而时间上最优的“局部距离平方反比”法（加上“高斯滤波”）成为最佳的方法。

最终A，B，C三张图的处理结果如下：

|  |  |
| --- | --- |
|  |  |
| A图（受损图） | A图（还原图） |
|  |  |
| B图（受损图） | B图（还原图） |
|  |  |
| C图（受损图） | C图（还原图） |

## 参考文献

* [1] 《克里金(Kriging)插值的原理与公式推导》, <http://blog.csdn.net/tsroad/article/details/50205373>
* [2] Mathwork开源代码：kriging.m，variogram.m，variogramfit.m，fminsearchbnd.m，fminsearchcon.m