机器学习降维算法四：Laplacian Eigenmaps 拉普拉斯特征映射

作者：[xbinworld](http://my.csdn.net/xbinworld)

原创书写，转载请注明此文出自：<http://www.cnblogs.com/xbinworld>，<http://blog.csdn.net/xbinworld>

Laplacian Eigenmaps

继续写一点经典的降维算法，前面介绍了PCA,LDA，LLE，这里讲一讲Laplacian Eigenmaps。其实不是说每一个算法都比前面的好，而是每一个算法都是从不同角度去看问题，因此解决问题的思路是不一样的。这些降维算法的思想都很简单，却在有些方面很有效。这些方法事实上是后面一些新的算法的思路来源。

Laplacian Eigenmaps[1] 看问题的角度和LLE有些相似，也是用局部的角度去构建数据之间的关系。

它的直观思想是希望相互间有关系的点（在图中相连的点）在降维后的空间中尽可能的靠近。Laplacian Eigenmaps可以反映出数据内在的流形结构。

Laplacian Eigenmaps也通过构建相似关系图（对应的矩阵为[clip_image002](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119125155.gif)）来重构数据流形的局部结构特征。Laplacian Eigenmaps算法的主要思想是，如果两个数据实例i和j很相似，那么i和j在降维后目标子空间中应该尽量接近。设数据实例的数目为n，目标子空间的维度为m。定义[clip_image006](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119133270.gif)大小的矩阵[clip_image008](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119136367.gif)，其中每一个行向量[clip_image010](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119141799.gif)是数据实例i在目标m维子空间中的向量表示，Laplacian Eigenmaps要优化的目标函数如下

[clip_image012](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119151898.gif)

定义对角矩阵[clip_image014](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119162553.gif)，对角线上[clip_image016](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119294463.gif)位置元素等于矩阵[clip_image002[1]](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119307943.gif)的第i行之和，经过线性代数变换，上述优化问题可以用矩阵向量形式表示如下：

[clip_image018](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119319121.gif)

其中矩阵[clip_image020](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119462691.gif)是图拉普拉斯矩阵。限制条件[clip_image022](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119522696.gif)保证优化问题有解，并且保证映射后的数据点不会被“压缩”到一个小于m维的子空间中。使得公式最小化的Y的列向量是以下广义特征值问题的m个最小非0特征值（包括重根）对应的特征向量：

[clip_image024](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119538401.gif)

使用时算法具体步骤为：

步骤1：构建图

使用某一种方法来将所有的点构建成一个图，例如使用KNN算法，将每个点最近的K个点连上边。K是一个预先设定的值。

步骤2：确定权重

确定点与点之间的权重大小，例如选用热核函数来确定，如果点i和点j相连，那么它们关系的权重设定为：

[clip_image026](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119559745.gif)

另外一种可选的简化设定是[clip_image028](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119576322.gif)如果点i，j相连，否则[clip_image030](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119588373.gif)。

步骤3：特征映射

计算拉普拉斯矩阵L的特征向量与特征值：[clip_image032](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119588406.gif)

其中D是对角矩阵，满足[clip_image034](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292120115267.gif)，[clip_image036](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292120115300.gif)。

使用最小的m个非零特征值对应的特征向量作为降维后的结果输出。

前面提到过，Laplacian Eigenmap具有区分数据点的特性，可以从下面的例子看出：

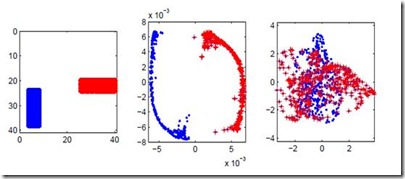
[](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292120128987.jpg)

图1 Laplacian Eigenmap实验结果

见图1所示，左边的图表示有两类数据点（数据是图片），中间图表示采用Laplacian Eigenmap降维后每个数据点在二维空间中的位置，右边的图表示采用PCA并取前两个主要方向投影后的结果，可以清楚地看到，在此分类问题上，Laplacian Eigenmap的结果明显优于PCA。

代码中 X=np.dot(D.I,L) np.linalg.eig(X)

[clip_image032](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292119588406.gif) ==》 D-1Ly = λy 所以用np.linalg.eig 求D-1L的特征值和特征向量即可

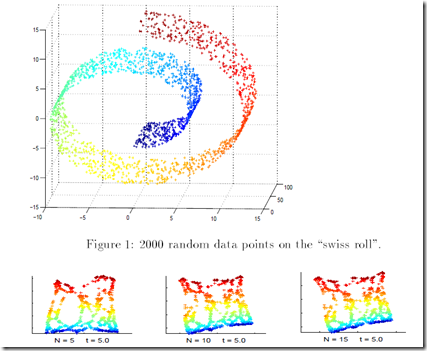
[](http://images.cnblogs.com/cnblogs_com/xbinworld/201211/201211292146267266.png)

图2 roll数据的降维

图2说明的是，高维数据（图中3D）也有可能是具有低维的内在属性的（图中roll实际上是2D的），但是这个低维不是原来坐标表示，例如如果要保持局部关系，蓝色和下面黄色是完全不相关的，但是如果只用任何2D或者3D的距离来描述都是不准确的。

下面三个图是Laplacian Eigenmap在不同参数下的展开结果（降维到2D），可以看到，似乎是要把整个带子拉平了。于是蓝色和黄色差的比较远。