**奇异值分解(SVD)详解及其应用**

**1.前言**

第一次接触奇异值分解还是在本科期间，那个时候要用到点对点的刚体配准，这是查文献刚好找到了四元数理论用于配准方法（点对点配准可以利用四元数方法，如果点数不一致更建议应用ICP算法）。一直想找个时间把奇异值分解理清楚、弄明白，直到今天才系统地来进行总结。

上一次学习过关于PCA的文章，PCA的实现一般有两种，一种是用特征值分解去实现的，一种是用奇异值分解去实现的。特征值和奇异值在大部分人的印象中，往往是停留在纯粹的数学计算中。而且线性代数或者矩阵论里面，也很少讲任何跟特征值与奇异值有关的应用背景。奇异值分解是一个有着很明显的物理意义的一种方法，它可以将一个比较复杂的矩阵用更小更简单的几个子矩阵的相乘来表示，这些小矩阵描述的是矩阵的重要的特性。就像是描述一个人一样，给别人描述说这个人长得浓眉大眼，方脸，络腮胡，而且带个黑框的眼镜，这样寥寥的几个特征，就让别人脑海里面就有一个较为清楚的认识，实际上，人脸上的特征是有着无数种的，之所以能这么描述，是因为人天生就有着非常好的抽取重要特征的能力，让机器学会抽取重要的特征，SVD是也一个重要的方法。在机器学习领域，有相当多的应用与奇异值都可以扯上关系，比如做feature reduction的PCA，做数据压缩（以图像压缩为代表）的算法，还有做搜索引擎语义层次检索的LSI（Latent Semantic Indexing）。  
本文主要关注奇异值的一些特性，还会稍稍提及奇异值的计算。另外，本文里面有部分不算太深的线性代数的知识，如果完全忘记了线性代数，看文可能会有些困难。

**2.奇异值分解详解**

特征值分解和奇异值分解两者有着很紧密的关系，特征值分解和奇异值分解的目的都是一样，就是提取出一个矩阵最重要的特征。先谈谈特征值分解吧：  
**1 特征值**：  
如果说一个向量v是方阵A的特征向量，将一定可以表示成下面的形式：

https://img-blog.csdn.net/20161024222333660

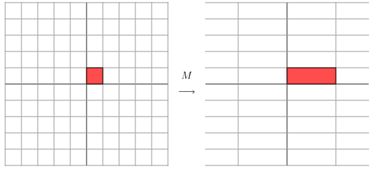
这时候λ就被称为特征向量v对应的特征值，一个矩阵的一组特征向量是一组正交向量。特征值分解是将一个矩阵分解成下面的形式：

https://img-blog.csdn.net/20161024222436129

其中Q是这个矩阵A的特征向量组成的矩阵，Σ是一个对角阵，每一个对角线上的元素就是一个特征值。首先，要明确的是，一个矩阵其实就是一个线性变换，因为一个矩阵乘以一个向量后得到的向量，其实就相当于将这个向量进行了线性变换。比如说下面的一个矩阵：

https://img-blog.csdn.net/20161024222545207

 它其实对应的线性变换是下面的形式：



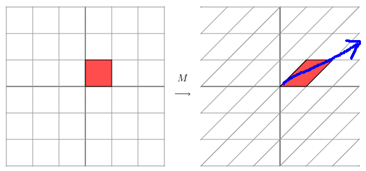
 因为这个矩阵M乘以一个向量(x,y)的结果是：

https://img-blog.csdn.net/20161024223335086

上面的矩阵是对称的，所以这个变换是一个对x，y轴的方向一个拉伸变换（每一个对角线上的元素将会对一个维度进行拉伸变换，当值>1时拉长，当值<1时缩短），当矩阵不是对称的时候，假如说矩阵是下面的样子：

https://img-blog.csdn.net/20161024223535337

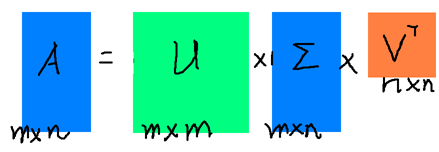
它所描述的变换是下面的样子：



这其实是在平面上对一个轴进行的拉伸变换（如蓝色的箭头所示），在图中，蓝色的箭头是一个最主要的变化方向（变化方向可能有不止一个），如果我们想要描述好一个变换，那我们就描述好这个变换主要的变化方向就好了。反过头来看看之前特征值分解的式子，分解得到的Σ矩阵是一个对角阵，里面的特征值是由大到小排列的，这些特征值所对应的特征向量就是描述这个矩阵变化方向（从主要的变化到次要的变化排列）  
当矩阵是高维的情况下，那么这个矩阵就是高维空间下的一个线性变换，这个线性变化可能没法通过图片来表示，但是可以想象，这个变换也同样有很多的变换方向，我们通过特征值分解得到的前N个特征向量，那么就对应了这个矩阵最主要的N个变化方向。我们利用这前N个变化方向，就可以近似这个矩阵（变换）。也就是之前说的：提取这个矩阵最重要的特征。总结一下，特征值分解可以得到特征值与特征向量，特征值表示的是这个特征到底有多重要，而特征向量表示这个特征是什么，可以将每一个特征向量理解为一个线性的子空间，我们可以利用这些线性的子空间干很多的事情。不过，特征值分解也有很多的局限，比如说变换的矩阵必须是方阵。  
**2 奇异值：**  
下面重点谈谈奇异值分解。特征值分解是一个提取矩阵特征很不错的方法，但是它只是对方阵而言的，在现实的世界中，我们看到的大部分矩阵都不是方阵，比如说有N个学生，每个学生有M科成绩，这样形成的一个N \* M的矩阵就不可能是方阵，我们怎样才能描述这样普通的矩阵呢的重要特征呢？奇异值分解可以用来干这个事情，奇异值分解是一个能适用于任意的矩阵的一种分解的方法：

https://img-blog.csdn.net/20161024223944449

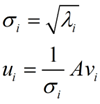
 假设A是一个N \* M的矩阵，那么得到的U是一个N \* N的方阵（里面的向量是正交的，U里面的向量称为左奇异向量），Σ是一个N \* M的矩阵（除了对角线的元素都是0，对角线上的元素称为奇异值），V’(V的转置)是一个N \* N的矩阵，里面的向量也是正交的，V里面的向量称为右奇异向量），如下图所示：



那么奇异值和特征值是怎么对应起来的呢？首先，我们将一个矩阵A的转置 \* A，将会得到一个方阵，我们用这个方阵求特征值可以得到：

https://img-blog.csdn.net/20161024224141050

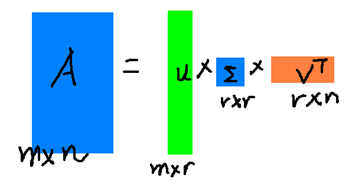
这里得到的v，就是我们上面的右奇异向量。此外我们还可以得到：



这里的σ就是上面说的奇异值，u就是上面说的左奇异向量。奇异值σ跟特征值类似，在矩阵Σ中也是从大到小排列，而且σ的减少特别的快，在很多情况下，前10%甚至1%的奇异值的和就占了全部的奇异值之和的99%以上了。也就是说，我们也可以用前r大的奇异值来近似描述矩阵，这里定义一下部分奇异值分解：

https://img-blog.csdn.net/20161024224331381

 r是一个远小于m、n的数，这样矩阵的乘法看起来像是下面的样子：



 右边的三个矩阵相乘的结果将会是一个接近于A的矩阵，在这儿，r越接近于n，则相乘的结果越接近于A。而这三个矩阵的面积之和（在存储观点来说，矩阵面积越小，存储量就越小）要远远小于原始的矩阵A，我们如果想要压缩空间来表示原矩阵A，我们存下这里的三个矩阵：U、Σ、V就好了。

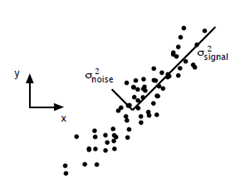
**3.如何计算奇异值**

 奇异值的计算是一个难题，是一个O(N^3)的算法。在单机的情况下当然是没问题的，matlab在一秒钟内就可以算出1000 \* 1000的矩阵的所有奇异值，但是当矩阵的规模增长的时候，计算的复杂度呈3次方增长，就需要并行计算参与了。  
其实SVD还是可以用并行的方式去实现的，在解大规模的矩阵的时候，一般使用迭代的方法，当矩阵的规模很大（比如说上亿）的时候，迭代的次数也可能会上亿次，如果使用Map-Reduce框架去解，则每次Map-Reduce完成的时候，都会涉及到写文件、读文件的操作。个人猜测Google云计算体系中除了Map-Reduce以外应该还有类似于MPI的计算模型，也就是节点之间是保持通信，数据是常驻在内存中的，这种计算模型比Map-Reduce在解决迭代次数非常多的时候，要快了很多倍。  
Lanczos迭代就是一种解对称方阵部分特征值的方法（之前谈到了，解A’\* A得到的对称方阵的特征值就是解A的右奇异向量），是将一个对称的方程化为一个三对角矩阵再进行求解。  
由于奇异值的计算是一个很枯燥，纯数学的过程，而且前人的研究成果（论文中）几乎已经把整个程序的流程图给出来了。更多的关于奇异值计算的部分，将在后面的参考文献中给出，这里不再深入，我还是focus在奇异值的应用中去。

**4.奇异值分解应用**

**奇异值与主成分分析（PCA）**

这里主要谈谈如何用SVD去解PCA的问题。PCA的问题其实是一个基的变换，使得变换后的数据有着最大的方差。方差的大小描述的是一个变量的信息量，我们在讲一个东西的稳定性的时候，往往说要减小方差，如果一个模型的方差很大，那就说明模型不稳定了。但是对于我们用于机器学习的数据（主要是训练数据），方差大才有意义，不然输入的数据都是同一个点，那方差就为0了，这样输入的多个数据就等同于一个数据了。以下面这张图为例子：



这个假设是一个摄像机采集一个物体运动得到的图片，上面的点表示物体运动的位置，假如我们想要用一条直线去拟合这些点，那我们会选择什么方向的线呢？当然是图上标有signal的那条线。如果我们把这些点单纯的投影到x轴或者y轴上，最后在x轴与y轴上得到的方差是相似的（因为这些点的趋势是在45度左右的方向，所以投影到x轴或者y轴上都是类似的），如果我们使用原来的xy坐标系去看这些点，容易看不出来这些点真正的方向是什么。但是如果我们进行坐标系的变化，横轴变成了signal的方向，纵轴变成了noise的方向，则就很容易发现什么方向的方差大，什么方向的方差小了。  
 一般来说，方差大的方向是信号的方向，方差小的方向是噪声的方向，我们在数据挖掘中或者数字信号处理中，往往要提高信号与噪声的比例，也就是信噪比。对上图来说，如果我们只保留signal方向的数据，也可以对原数据进行不错的近似了。  
PCA的全部工作简单点说，就是对原始的空间中顺序地找一组相互正交的坐标轴，第一个轴是使得方差最大的，第二个轴是在与第一个轴正交的平面中使得方差最大的，第三个轴是在与第1、2个轴正交的平面中方差最大的，这样假设在N维空间中，我们可以找到N个这样的坐标轴，我们取前r个去近似这个空间，这样就从一个N维的空间压缩到r维的空间了，但是我们选择的r个坐标轴能够使得空间的压缩使得数据的损失最小。  
 还是假设我们矩阵每一行表示一个样本，每一列表示一个feature，用矩阵的语言来表示，将一个m \* n的矩阵A的进行坐标轴的变化，P就是一个变换的矩阵从一个N维的空间变换到另一个N维的空间，在空间中就会进行一些类似于旋转、拉伸的变化。

https://img-blog.csdn.net/20161024225222282

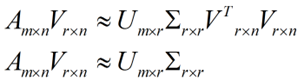
而将一个m \* n的矩阵A变换成一个m \* r的矩阵，这样就会使得本来有n个feature的，变成了有r个feature了（r < n)，这r个其实就是对n个feature的一种提炼，我们就把这个称为feature的压缩。用数学语言表示就是：

https://img-blog.csdn.net/20161024225301735

但是这个怎么和SVD扯上关系呢？之前谈到，SVD得出的奇异向量也是从奇异值由大到小排列的，按PCA的观点来看，就是方差最大的坐标轴就是第一个奇异向量，方差次大的坐标轴就是第二个奇异向量…我们回忆一下之前得到的SVD式子：

https://img-blog.csdn.net/20161024225338470

在矩阵的两边同时乘上一个矩阵V，由于V是一个正交的矩阵，所以V转置乘以V得到单位阵I，所以可以化成后面的式子：



将后面的式子与A \* P那个m \* n的矩阵变换为m \* r的矩阵的式子对照看看，在这里，其实V就是P，也就是一个变化的向量。这里是将一个m \* n 的矩阵压缩到一个m \* r的矩阵，也就是对列进行压缩，如果我们想对行进行压缩（在PCA的观点下，对行进行压缩可以理解为，将一些相似的sample合并在一起，或者将一些没有太大价值的sample去掉）怎么办呢？同样我们写出一个通用的行压缩例子：

https://img-blog.csdn.net/20161024225536299

这样就从一个m行的矩阵压缩到一个r行的矩阵了，对SVD来说也是一样的，我们对SVD分解的式子两边乘以U的转置U'：

https://img-blog.csdn.net/20161024225608924

 这样我们就得到了对行进行压缩的式子。可以看出，其实PCA几乎可以说是对SVD的一个包装，如果我们实现了SVD，那也就实现了PCA了，而且更好的地方是，有了SVD，我们就可以得到两个方向的PCA，如果我们对A’A进行特征值的分解，只能得到一个方向的PCA。

**5.感谢**

1.感谢杜克大学 方贺好友

2.感谢博客园 王大田博主

3.感谢清华大学数学系 刘永中教授