机器学习算法系列（21）：SVD

一、简介

SVD实际上是数学专业内容，但它现在已经深入到不同的领域中。SVD的过程不是很好理解，因为它不够直观，但它对矩阵分解的效果却非常好。比如，Netflix（一个提供在线电影租赁的公司）曾经就悬赏100万美金，如果谁能提高他的电影推荐系统评分预测率10%的话。令人惊讶的是，这个目标充满了挑战，来自世界各地的团队运用了各种不同的技术。最终的获胜队伍“BellKor's Pragmatic Chaos”采用的核心算法就是基于SVD。

SVD提供了一种非常便捷的矩阵分解方式，能够发现数据中十分有意思的潜在模式，在这篇文章中，我们将会提供对SVD集合上的理解和一些简单的应用实例。

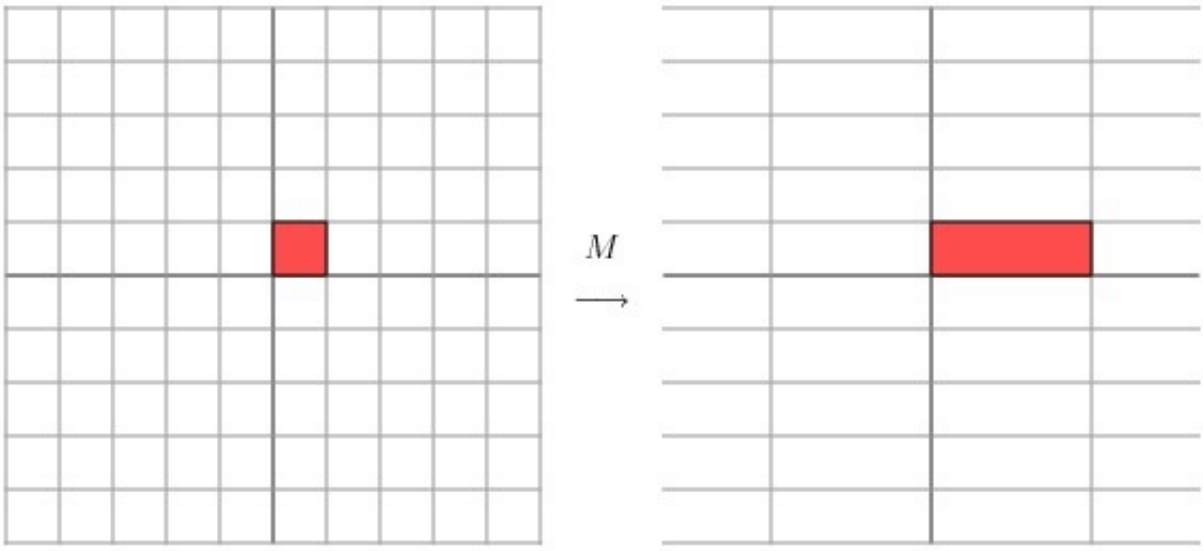
1.1 几何意义

奇异值分解就是把一个线性变换分解成两个线性变换，一个线性变化代表旋转，另一个代表拉伸。

线性代数中最让人印象深刻的一点是，要将矩阵和空间中的线性变化视为同样的事物。比如对角矩阵M作用在任何一个向量上

$$\begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3x \\ y \end{bmatrix}$$

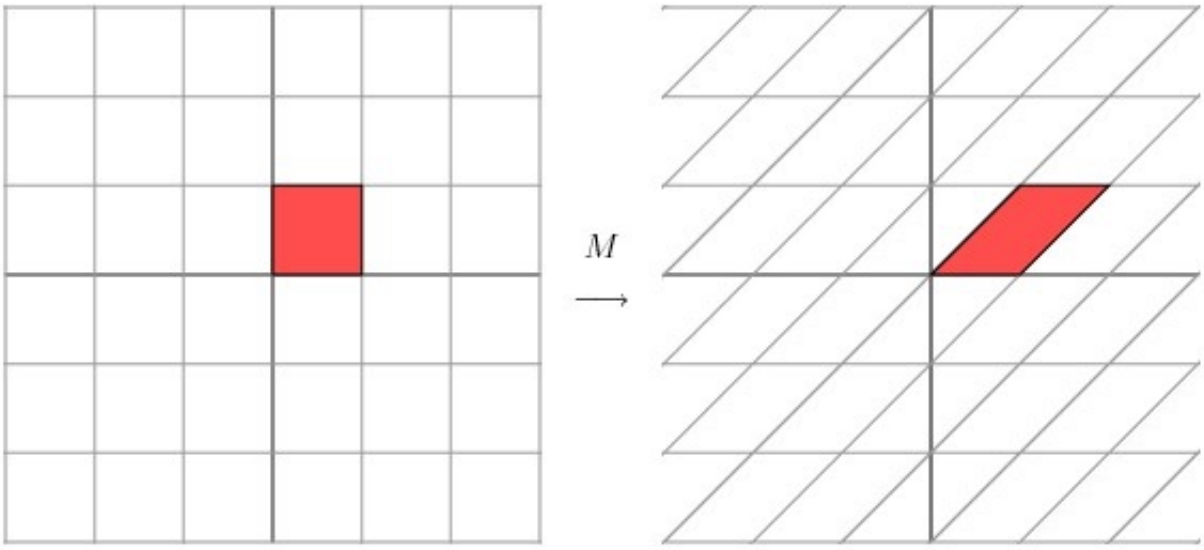
其几何意义为在水平x方向上拉伸3倍，y方向保持不变的线性变换。换言之对角矩阵起到作用是将水平垂直网格作水平拉伸（或者反射后水平拉伸）的线性变换。



如果M不是对角矩阵。而是一个对称矩阵：

$$M = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

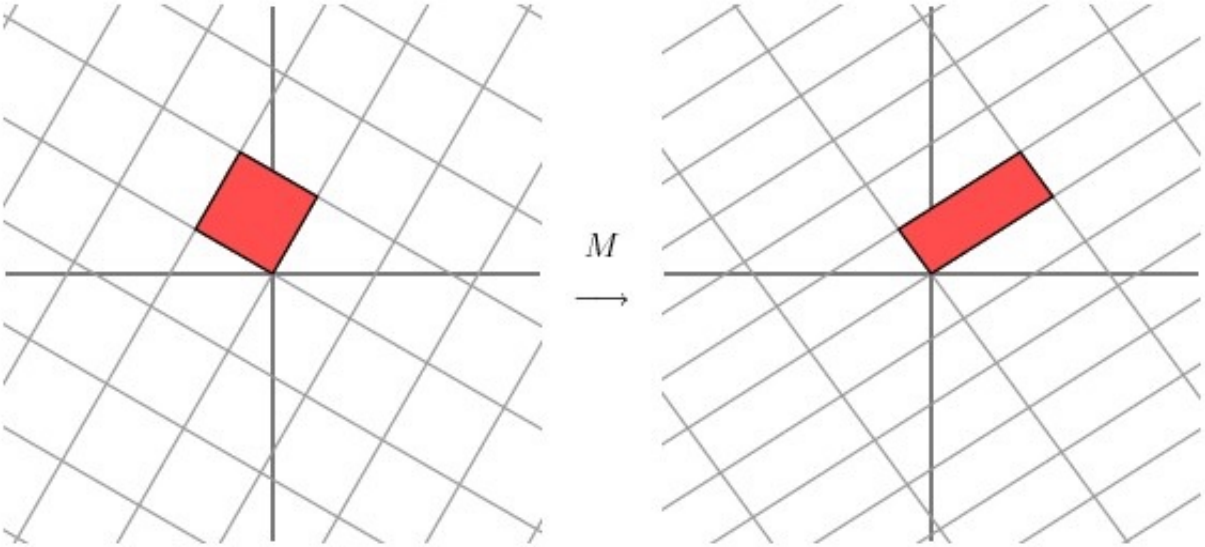
那么我们也总能找到一组网格线，使得矩阵作用在该网格上仅仅表现为（反射）拉伸变换，而没有发生旋转变换。这个矩阵产生的变换效果如下图所示



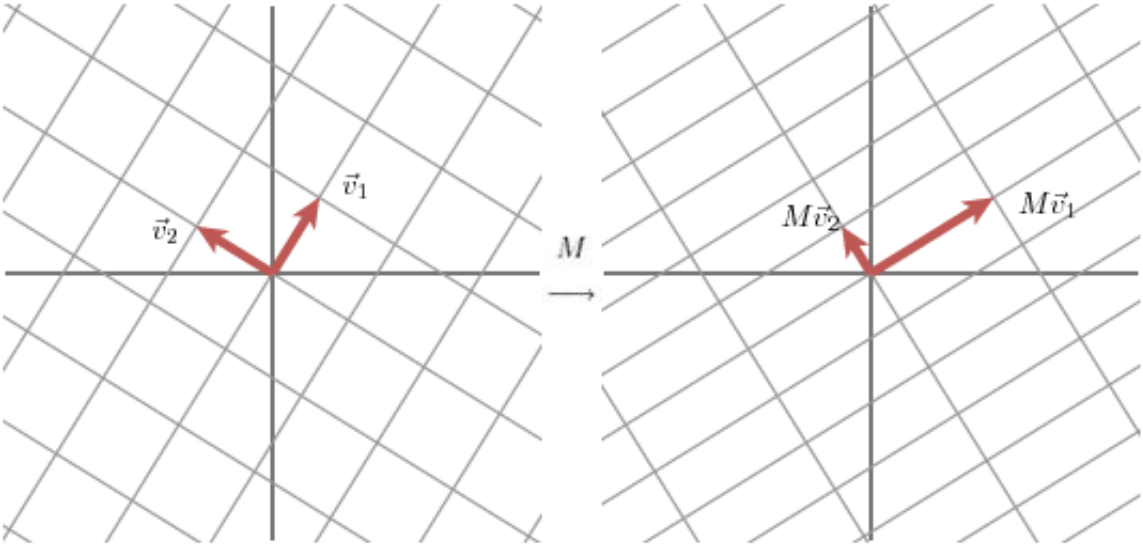
考虑一下更一般的非对称矩阵

$$M = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

很遗憾，此时我们再也找不到一组网格，使得矩阵作用在该网格之后只有拉伸变换（找不到背后的数学原因就是对一般非对称矩阵无法保证在实数域上可对角化）。我们退而求其次，找到一组网格，使得矩阵作用在该网格之后允许有拉伸变换和旋转变换，但要保证变换后的网格依旧互相垂直。这是可以做到的



下面我们就可以自然过渡到奇异值分解的引入。奇异值分解的几何含义为：**对于任何的一个矩阵，我们要找到一组两两正交单位向量序列，使得矩阵作用在此向量序列上后得到新的向量序列保持两两正交。**下面我们要说明的是，奇异值的几何含义为：**这组变换后的新的向量序列的长度。**



当矩阵M作用在正交单位向量v_1和v_2上之后，得到Mv_1和Mv_2也是正交的。令u_1和u_2分别是Mv_1和Mv_2方向上的单位向量，即$Mv_1 = \sigma_1 u_1$，$Mv_2 = \sigma_2 u_2$，写在一起就是$M[v_1, v_2] = [\sigma_1 u_1\ \sigma_2 u_2]$，整理得到

$$M = M\begin{bmatrix} v_1 & v_2 \end{bmatrix}\begin{bmatrix} v_1^T \\ v_2^T \end{bmatrix} = \begin{bmatrix} \sigma_1 u_1 & \sigma_2 u_2 \end{bmatrix}\begin{bmatrix} v_1^T \\ v_2^T \end{bmatrix} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}\begin{bmatrix} v_1^T \\ v_2^T \end{bmatrix}$$

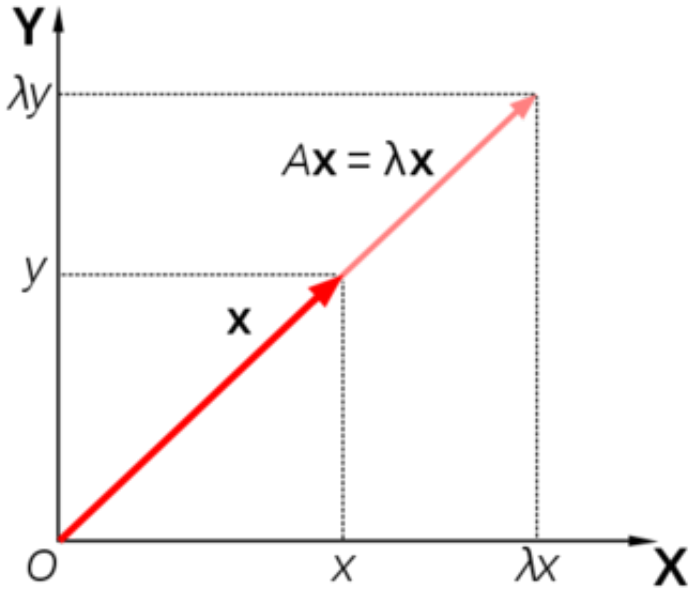
这样就得到矩阵M的奇异值分解。奇异值σ_1和σ_2分别是Mv_1和Mv_2的长度。很容易可以把结论推广到一般n维的情况

二、、奇异值分解

2.1 特征值分解

如果方阵对某个向量只产生伸缩，而不产生旋转效果，那么这个向量就称为矩阵的特征向量，伸缩的比例就是对应的特征值。

$$Ax = \lambda x$$



所以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（变化方向可能不止一个），如果我们想要描述好一个变换，那我们描述好这个变换主要的变化方向就好了。反过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的方向（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。也就是之前受的：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敢很多的事情。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。

学线性代数的时候，我们应该都学过这样一个定理：

> 若A为n阶实对称阵（方阵），则存在由特征值组成的对角阵Λ和特征向量组成的正交阵Q，使得：
>
> $$A = Q\Lambda Q^T$$

这就是我们所说的特征值分解（Eigenvalue decomposition: EVD）（$R^n \rightarrow R^n$），而奇异值分解其实可以看做是特征值分解在任意矩阵$m \times n$上的推广形式($R^n \rightarrow R^m$)。只有对方阵才有特征值的概念，所以对于任意的矩阵，我们引入了奇异值。

2.2 奇异值分解

上面的特征值分解是一个提取矩阵特征很不错的方法，当它只是对方阵而言的，在现实的世界

中，我们看到的大部分都不是方阵，比如说有N个学生，每个学生有M科成绩，这样形成的一个$N \times M$的矩阵就不可能是方阵！那么现在就来分析：对于任意的$m \times n$的矩阵，能否找到一组正交基使得经过它变换后还是正交基？答案是肯定的，它就是SVD分解的精髓所在。

下面我们从特征值分解出发，导出奇异值分解。

首先我们注意到A^TA为n阶对称矩阵，我们可以对它做特征值分解。

$$A^TA = VDV^T$$

这个时候我们可以得到一组正交基，$\{v_1, v_2, \cdots v_n\}$：

$$(A^TA)v_i = \lambda_i v_i$$

$$(Av_i, Av_j) = (Av_i)^T(Av_j) = v_i^TA^TAv_j = v_i^T(\lambda_j v_j) = \lambda_j v_i^T v_j = 0$$

由$r(A^TA) = r(A) = r$，这个时候我们得到了一组正交基，$\{Av_1, Av_2, \cdots, Av_r\}$，先将其标准化，令：

$$u_i = \frac{Av_i}{|Av_i|} = \frac{1}{\sqrt{\lambda_i}}Av_i \Rightarrow Av_i = \sqrt{\lambda_i}u_i = \delta_i u_i$$

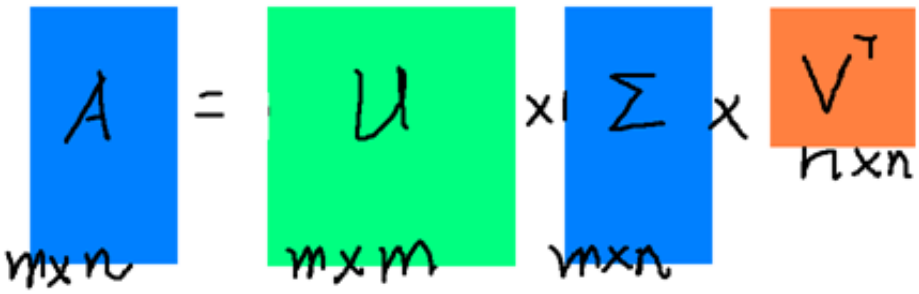
其中

$$|Av_i|^2 = (Av_i, Av_i) = \lambda_i v_i^T v_i = \lambda_i \Rightarrow |Av_i| = \sqrt{\lambda_i} = \delta_i (\text{奇 异 值})$$

将向量组$\{u_1, u_2, \cdots, u_r\}$扩充为$R^m$中的标准正交基$\{u_1, u_2, \cdots, u_r, \cdots, u_m\}$，则：

$$AV = A(v_1 v_2 \cdots v_n) = (Av_1\ Av_2 \cdots Av_r\ 0 \cdots 0) = (\delta_1 u_1\ \delta_2 u_2 \cdots \delta_r u_r\ 0 \cdots 0 = U\Sigma \Rightarrow A = U$$

我们可以从下图中直观的感受奇异值分解的矩阵相乘。

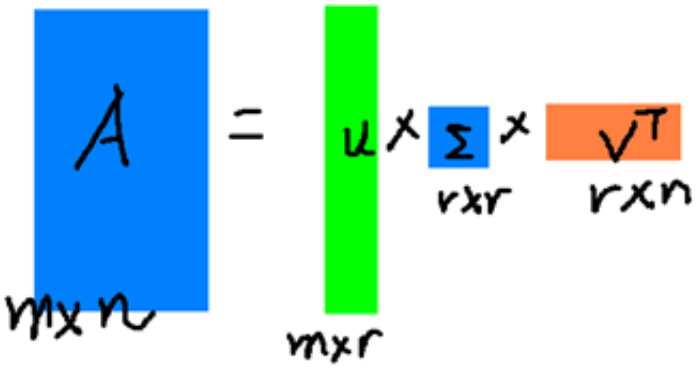


任意的矩阵A是可以分解成三个矩阵。其中V表示了原始域的标准正交基，U表示经过A变化后的$co-domain$的标准正交基，Σ表示了V中的向量与U中相对应向量之间的关系。

在很多情况下，前10%甚至1%的奇异值的和就占了全部分奇异值之和的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：

$$A_{m\times n} \approx U_{m\times r}\Sigma_{r\times r}V^T_{r\times n}$$

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（早存储观点来说，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。

三、应用实例

3.1 推荐系统

我们现在有一批高尔夫球手对九个不同hole的所需挥杆次数数据，我们希望基于这些数据建立模型，来预测选手对于某个给定hole的挥杆次数。（这个例子来自于： Singular Value Decomposition (SVD) Tutorial，强烈建议大家都去看看）

Hole	Par	Phil	Tiger	Vijay
1	4	4	4	4
2	5	5	5	5
3	3	3	3	3
4	4	4	4	4
5	4	4	4	4
6	4	4	4	4
7	4	4	4	4
8	3	3	3	3
9	5	5	5	5

最简单的一个思路，我们队每个hole设立一个难度指标HoleDifficulty，对每位选手的能力也设立一个评价指标PlayAbility，实际的得分取决于这俩者的乘积：

$$PredictedScore = HoleDifficulty \cdot PlayerAbility$$

我们可以简单地把每位选手的Ability都设为1，那么：

Phil	Tiger	Vijay
4	4	4
5	5	5
3	3	3
4	4	4
4	4	4
4	4	4
4	4	4
3	3	3
5	5	5

=

HoleDifficulty
4
5
3
4
4
4
4
3
5

×

PlayerAbility		
Phil	Tiger	Vijay
1	1	1

接着我们将HoleDifficulty 和 PlayAbility这两个向量标准化，可以得到如下的关系：

Phil	Tiger	Vijay
4	4	4
5	5	5
3	3	3
4	4	4
4	4	4
4	4	4
4	4	4
3	3	3
5	5	5

=

HoleDifficulty
0.33
0.41
0.25
0.33
0.33
0.33
0.33
0.25
0.41

×

ScaleFactor
21.07

×

PlayerAbility		
Phil	Tiger	Vijay
0.58	0.58	0.58

好熟悉，这不就是传说中的SVD吗，这样就出来了。

这里面蕴含了一个非常有趣的思想，也是SVD这么有用的核心：

最开始高尔夫球员和Holes之间是没有直接联系的，我们通过feature把它们联系在一起：不同的Hole进洞难度是不一样的，每个球手对进度难度的把控也是不一样的，那么我们就可以通过进洞难度这个feature将它们联系在一起，将它们乘起来就得到了我们想要的挥杆次数。

这个思想很重要，对于我们理解LSI和SVD再推荐系统中的应用相当重要。

SVD分解其实就是利用隐藏的Feature建立起矩阵行与列之间的联系。

大家可能注意到，上面那个矩阵秩为1，所以我们很容易就能将其分解，但是在实际问题中我们就得依靠SVD分解，这个时候的隐藏特征往往也不止一个了。

我们将上面的数据稍作修改：

Phil	Tiger	Vijay
4	4	5
4	5	5
3	3	2
4	5	4
4	4	4
3	5	4
4	4	3
2	4	4
5	5	5

=

HoleDifficulty 1–3		
4.34	−0.18	−0.90
4.69	−0.38	−0.15
2.66	0.80	0.40
4.36	0.15	0.47
4.00	0.35	−0.29
4.05	−0.67	0.68
3.66	0.89	0.33
3.39	−1.29	0.14
5.00	0.44	−0.36

×

PlayerAbility 1–3		
Phil	Tiger	Vijay
0.91	1.07	1.00
0.82	−0.20	−0.53
−0.21	0.76	−0.62

进行奇异值分解，可以得到：

Phil	Tiger	Vijay
4	4	5
4	5	5
3	3	2
4	5	4
4	4	4
3	5	4
4	4	3
2	4	4
5	5	5

=

HoleDifficulty 1–3		
0.35	0.09	−0.64
0.38	0.19	−0.10
0.22	−0.40	0.28
0.36	−0.08	0.33
0.33	−0.18	−0.20
0.33	0.33	0.48
0.30	−0.44	0.23
0.28	0.64	0.10
0.41	−0.22	−0.25

×

ScaleFactor 1–3		
21.07	0	0
0	2.01	0
0	0	1.42

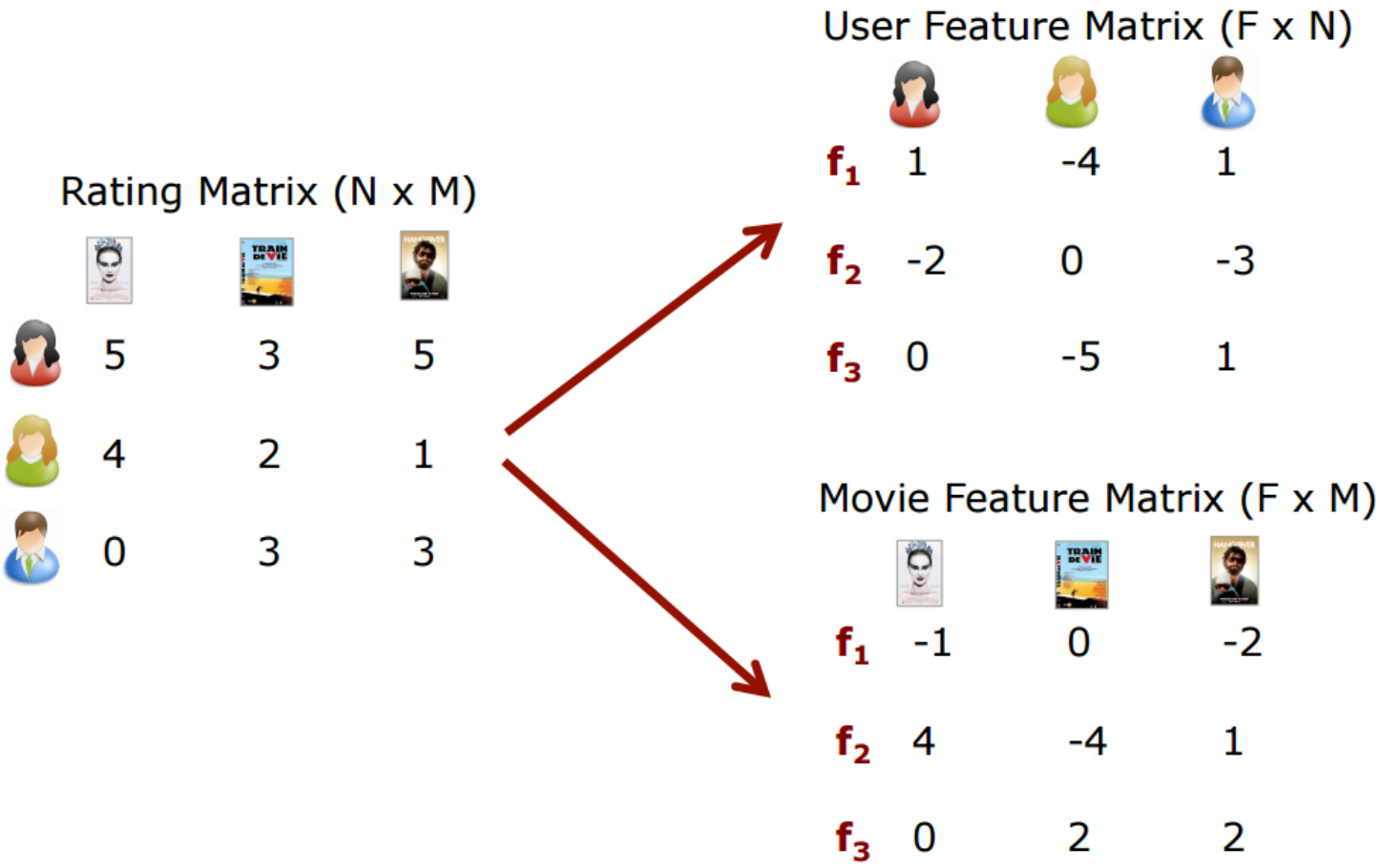
×

PlayerAbility 1–3		
Phil	Tiger	Vijay
0.53	0.62	0.58
−0.82	0.20	0.53
−0.21	0.76	−0.62

隐藏特征的重要性是与其对应的奇异值大小成正比的，也就是奇异值越大，其所对应的隐藏特征也越重要。

我们将这个思想推广一下

在推荐系统中，用户和物品之间没有直接联系。但是我们可以通过feature把它们联系在一起。对于电影来说，这样的特征可以是：喜剧还是悲剧，是动作片还是爱情片。用户和这样的feature之间是有关系的，比如某个用户喜欢看爱情片，另外一个用户喜欢看动作片；物品和feature之间也是有关系的，比如某个电影是喜剧，某个电影是悲剧。那么通过和feature之间的联系，我们就找到了用户和物品之间的关系。



3.2 数据压缩

矩阵的奇异值是一个数学意义上的概念，一般是由奇异值分解（Singular Value Decomposition，简称SVD分解）得到。如果要问奇异值表示什么物理意义，那么就必须考虑在不同的实际工程应用中奇异值所对应的含义。下面先尽量避开严格的数学符号推导，直观的从一张图片出发，让我们来看看奇异值代表什么意义。

这是女神上野树里（Ueno Juri）的一张照片，像素为450 × 333

我们都知道，图片实际上对应着一个矩阵，矩阵的大小就是像素大小，比如这张图对应的矩阵阶数就是450*333，矩阵上每个元素的数值对应着像素值。我们记这个像素矩阵为 AA 。
现在我们对矩阵 AA 进行奇异值分解。直观上，奇异值分解将矩阵分解成若干个秩一矩阵之和，用公式表示就是：

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T$$

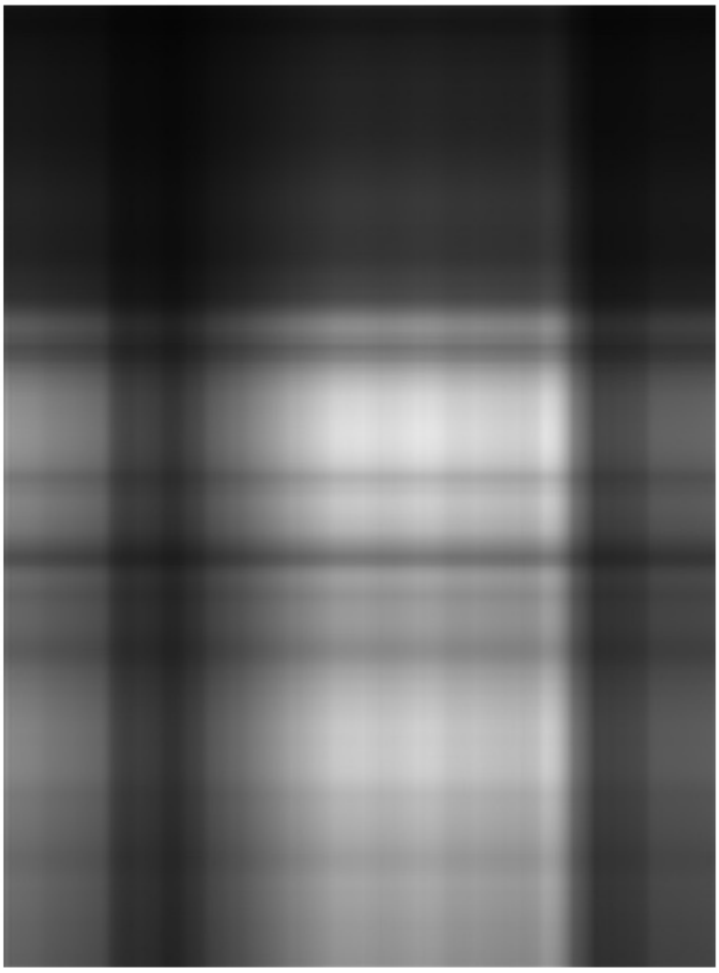
其中等式右边每一项前的系数σ就是奇异值，u和v分别表示列向量，秩一矩阵的意思是秩为1的矩阵。注意到每一项uv^T都是秩为1的矩阵。我们假定奇异值满足

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$$

（奇异值大于0是个重要的性质，但这里先别在意），如果不满足的话重新排列顺序即可，这无非是编号顺序的问题。

既然奇异值有从大到小排列的顺序，我们自然要问，如果只保留大的奇异值，舍去较小的奇异值，这样(1)式里的等式自然不再成立，那会得到怎样的矩阵——也就是图像?

令 $A_1 = \sigma_1 u_1 v_1^T$ ，这只保留(1)中等式右边第一项，然后作图



结果就是完全看不清是啥…我们试着多增加几项进来：

$$A_5 = \sigma_1\mu_1 v_1^T + \sigma_2\mu_2 v_2^T + \ldots + \sigma_5\mu_5 v_5^T$$

再作图

隐约可以辨别这是短发伽椰子的脸……但还是很模糊，毕竟我们只取了5个奇异值而已。下面我们取20个奇异值试试，也就是(1)式等式右边取前20项构成 A20。

虽然还有些马赛克般的模糊，但我们总算能辨别出这是Juri酱的脸。当我们取到(1)式等式右边前50项时：

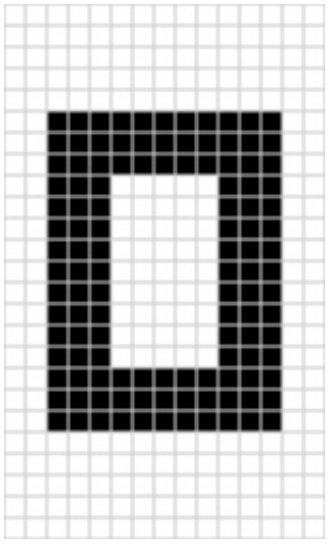
我们得到和原图差别不大的图像。也就是说当k从1不断增大时，A_k不断的逼近A。让我们回到公式

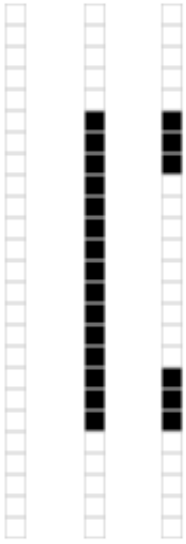
$$A = \sigma_1\mu_1 v_1^T + \sigma_2\mu_2 v_2^T + \ldots + \sigma_r\mu_r v_r^T$$

矩阵表示一个450×333的矩阵，需要保存$450 \times 333 = 149850$个元素的值。等式右边和分别是450×1和333×1的向量，每一项有个元素。如果我们要存储很多高清的图片，而又受限于存储空间的限制，在尽可能保证图像可被识别的精度的前提下，我们可以保留奇异值较大的若干项，舍去奇异值较小的项即可。例如在上面的例子中，如果我们只保留奇异值分解的前50项，则需要存储的元素为，和存储原始矩阵相比，存储量仅为后者的26%。

奇异值往往对应着矩阵中隐含的重要信息，且重要性和奇异值大小正相关。每个矩阵A都可以表示为一系列秩为1的“小矩阵”之和，而奇异值则衡量了这些“小矩阵”对于A的权重。

奇异值分解也可以高效地表示数据。例如，假设我们想传送下列图片，包含15×25个黑色或者白色的像素阵列





因为在图像中只有三种类型的列（如下）,它可以以更紧凑的形式被表示。
就保留主要样本数据来看，该过程跟PCA(principal component analysis)技术有一些联系，PCA也使用了SVD去检测数据间依赖和冗余信息.

$$
M = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
$$

如果对M进行奇异值分解的话，我们只会得到三个非零的奇异值。

$$\sigma_1 = 14.72 \ \sigma_2 = 5.22 \sigma_3 = 3.31$$

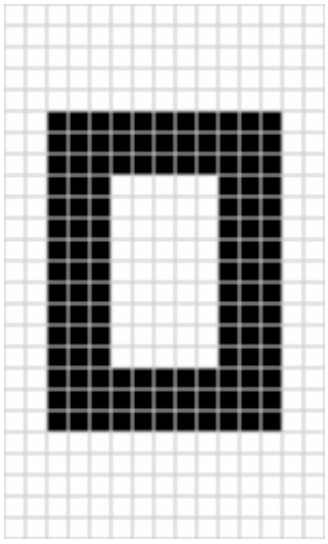
因此，矩阵可以如下表示

$$M = u_1 \sigma_1 v_1^T + u_2 \sigma_2 v_2^T + u_3 \sigma_3 v_3^T$$

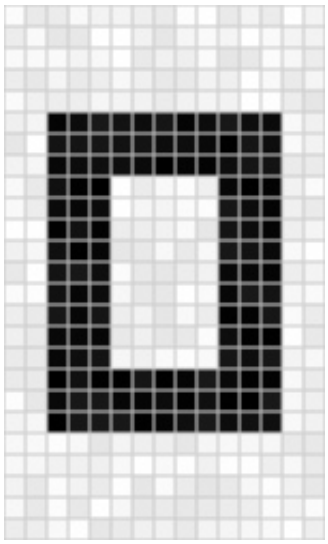
我们有三个包含15个元素的向量v_i，三个包含25个元素的向量u_i，以及三个奇异值σ_i，这意味着我们可以只用123个数字就能表示这个矩阵而不是出现在矩阵中的375个元素。在这种方式下，我们看到在矩阵中有三个线性独立的列，也就是说矩阵的秩是3。

3.3 图像去噪

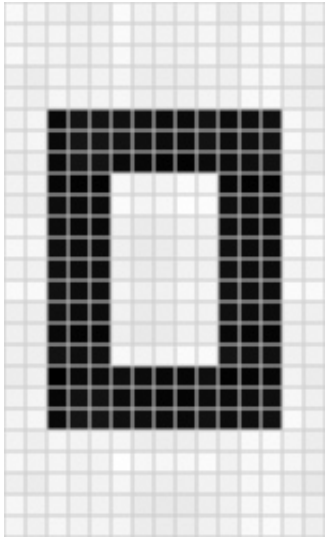
在图像处理领域，奇异值不仅可以应用在数据压缩上，还可以对图像去噪。如果一副图像包含噪声，我们有理由相信那些较小的奇异值就是由于噪声引起的。当我们强行令这些较小的奇异值为0时，就可以去除图片中的噪声。如下是一张$25 * 15$的图像（本例来源于[1]）



但往往我们只能得到如下带有噪声的图像（和无噪声图像相比，下图的部分白格子中带有灰色）：



通过奇异值分解，我们发现矩阵的奇异值从大到小分别为：14.15，4.67，3.00，0.21，……，0.05。除了前3个奇异值较大以外，其余奇异值相比之下都很小。强行令这些小奇异值为0，然后只用前3个奇异值构造新的矩阵，得到

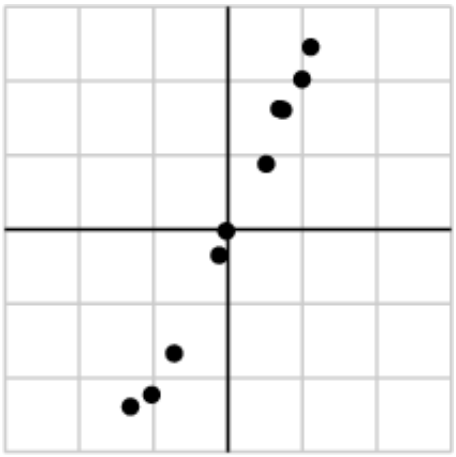


可以明显看出噪声减少了（白格子上灰白相间的图案减少了）。

3.4 数据分析

我们搜集的数据中总是存在噪声：无论采用的设备多精密，方法有多好，总是会存在一些误差的。如果你们还记得上文提到的，大的奇异值对应了矩阵中的主要信息的话，运用SVD进行数据分析，提取其中的主要部分的话，还是相当合理的。

作为例子，假如我们搜集的数据如下所示：



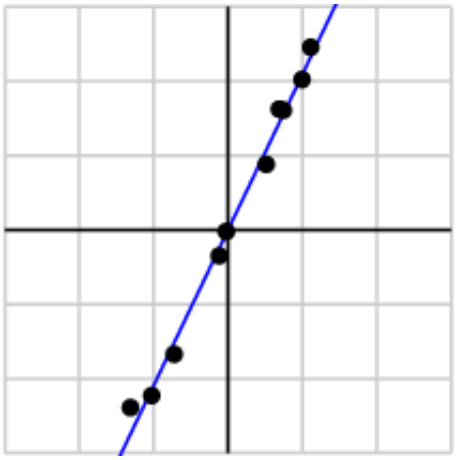
我们将数据用矩阵的形式表示：

$$\begin{matrix} -1.03 & 0.74 & -0.02 & 0.51 & -1.31 & 0.99 & 0.69 & -0.12 & -0.72 & 1.11 \\ -2.23 & 1.61 & -0.02 & 0.88 & -2.39 & 2.02 & 1.62 & -0.35 & -1.67 & 2.46 \end{matrix}$$

经过奇异值分解后，得到

$$\sigma_1 = 6.04 \; \sigma_2 = 0.22$$

由于第一个奇异值远比第二个要大，数据中有包含一些噪声，第二个奇异值在原始矩阵分解相对应的部分可以忽略。经过SVD分解后，保留了主要样本点如图所示



就保留主要样本数据来看，该过程跟PCA(principal component analysis)技术有一些联系，PCA也使用了SVD去检测数据间依赖和冗余信息.

3.5 潜在语义索引LSI

潜在语义索引（Latent Semantic Indexing）与PCA不太一样，至少不是实现了SVD就可以直接用的，不过LSI也是一个严重依赖于SVD的算法，之前吴军老师在矩阵计算与文本处理中的分类问题中谈到：

> “三个矩阵有非常清楚的物理含义。第一个矩阵X中的每一行表示意思相关的一类词，其中的每个非零元素表示这类词中每个词的重要性（或者说相关性），数值越大越相关。最后一个矩阵Y中的每一列表示同一主题一类文章，其中每个元素表示这类文章中每篇文章的相关性。中间的矩阵则表示类词和文章之间的相关性。因此，我们只要对关联矩阵A进行一次奇异值分解，我们就可以同时完成了近义词分类和文章的分类。（同时得到每类文章和每类词的相关性）。”

上面这段话可能不太容易理解，不过这就是LSI的精髓内容，我下面举一个例子来说明一下，下面的例子来自LSA tutorial：

Index Words	Titles								
	T1	T2	T3	T4	T5	T6	T7	T8	T9
book			1	1					
dads						1			1
dummies		1						1	
estate							1		1
guide	1					1			
investing	1	1	1	1	1	1	1	1	1
market	1		1						
real							1		1
rich						2			1
stock	1		1					1	
value				1	1				

这就是一个矩阵，不过不太一样的是，这里的一行表示一个词在哪些title中出现了（一行就是之前说的一维feature），一列表示一个title中有哪些词，（这个矩阵其实是我们之前说的那种一行是一个sample的形式的一种转置，这个会使得我们的左右奇异向量的意义产生变化，但是不会影响我们计算的过程）。比如说T1这个title中就有guide、investing、market、stock四个词，各出现了一次，我们将这个矩阵进行SVD，得到下面的矩阵：

book	0.15	-0.27	0.04
dads	0.24	0.38	-0.09
dummies	0.13	-0.17	0.07
estate	0.18	0.19	0.45
guide	0.22	0.09	-0.46
investing	0.74	-0.21	0.21
market	0.18	-0.30	-0.28
real	0.18	0.19	0.45
rich	0.36	0.59	-0.34
stock	0.25	-0.42	-0.28
value	0.12	-0.14	0.23

*

3.91	0	0
0	2.61	0
0	0	2.00

*

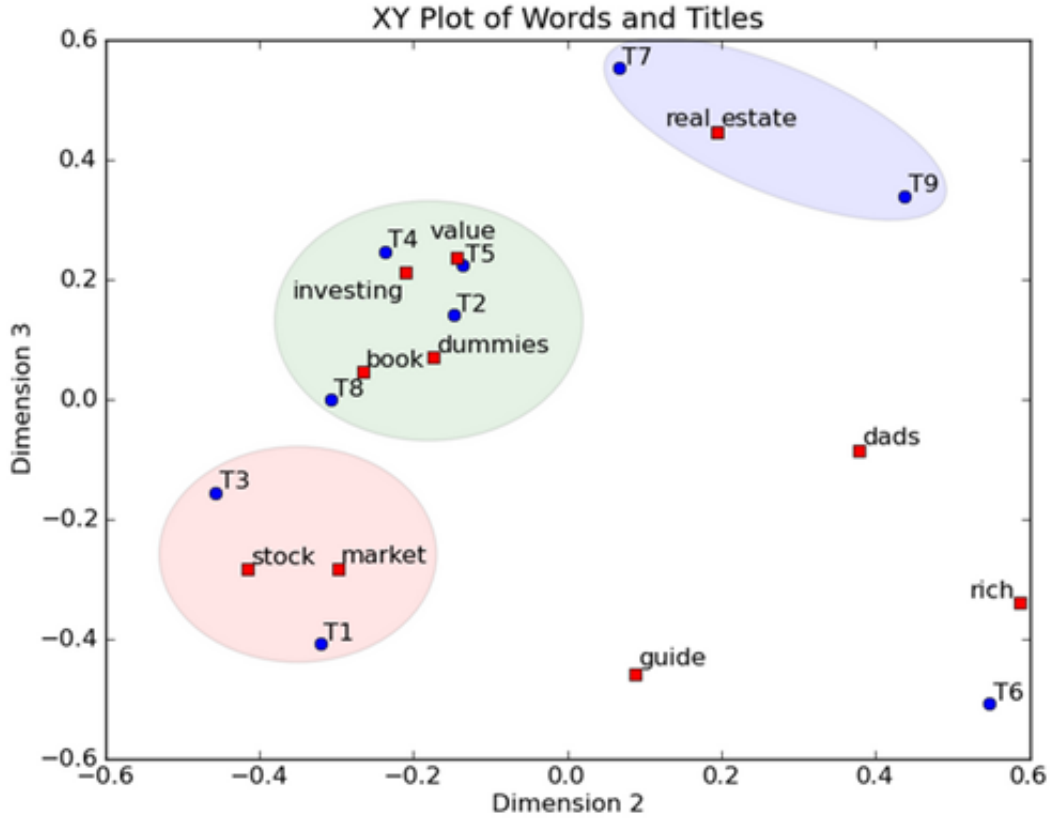
T1	T2	T3	T4	T5	T6	T7	T8	T9
0.35	0.22	0.34	0.26	0.22	0.49	0.28	0.29	0.44
-0.32	-0.15	-0.46	-0.24	-0.14	0.55	0.07	-0.31	0.44
-0.41	0.14	-0.16	0.25	0.22	-0.51	0.55	0.00	0.34

左奇异向量表示词的一些特性，右奇异向量表示文档的一些特性，中间的奇异值矩阵表示左奇异向量的一行与右奇异向量的一列的重要程序，数字越大越重要。

继续看这个矩阵还可以发现一些有意思的东西，首先，左奇异向量的第一列表示每一个词的出现频繁程度，虽然不是线性的，但是可以认为是一个大概的描述，比如book是0.15对应文档中出现的2次，investing是0.74对应了文档中出现了9次，rich是0.36对应文档中出现了3次；

其次，右奇异向量中一的第一行表示每一篇文档中的出现词的个数的近似，比如说，T6是0.49，出现了5个词，T2是0.22，出现了2个词。

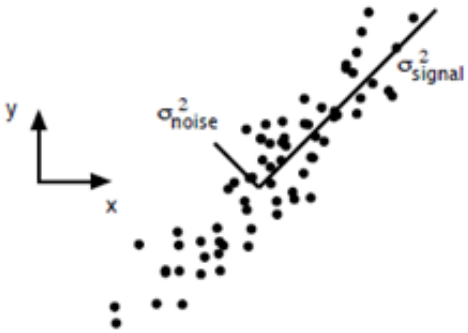
然后我们反过头来看，我们可以将左奇异向量和右奇异向量都取后2维（之前是3维的矩阵），投影到一个平面上，可以得到：



在图上，每一个红色的点，都表示一个词，每一个蓝色的点，都表示一篇文档，这样我们可以对这些词和文档进行聚类，比如说stock 和 market可以放在一类，因为他们老是出现在一起，real和estate可以放在一类，dads，guide这种词就看起来有点孤立了，我们就不对他们进行合并了。按这样聚类出现的效果，可以提取文档集合中的近义词，这样当用户检索文档的时候，是用语义级别（近义词集合）去检索了，而不是之前的词的级别。这样一减少我们的检索、存储量，因为这样压缩的文档集合和PCA是异曲同工的，二可以提高我们的用户体验，用户输入一个词，我们可以在这个词的近义词的集合中去找，这是传统的索引无法做到的。

3.6 主成分分析

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。

$$A_{m\times n}P_{n\times n}=\tilde{A}_{m\times n}$$

而将一个$m\times n$的矩阵A变换成一个$m\times r$的矩阵，这样就会使本来有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：

$$A_{m\times n}P_{n\times r}=\tilde{A}_{m\times r}$$

但是这个和SCD扯上关系的呢？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，按照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。之前得到的SVD式子如下：

$$A_{m\times n}\approx U_{m\times r}\Sigma_{r\times r}V^T_{r\times n}$$

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，所以转置乘以V得到单位阵I，所以可以化成后面的式子：

$$A_{m\times n}V_{r\times n}\approx U_{m\times r}\Sigma_{r\times r}V^T_{r\times n}V_{r\times n}=U_{m\times r}\Sigma_{r\times r}$$

将后面的式子与$A\times P$那个$m\times n$矩阵变换为$m\times r$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。这里将一个$m\times n$的矩阵压缩到一个$m\ timesr$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，对行进行业所可以理解为，讲一些相似的sample合并在一起，或者将一些没有太大价值的sample去掉），同样我们写出一个通用的行压缩例子：

$$P_{r\times m}A_{m\times n}=\tilde{A}_{r\times n}$$

这样从一个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一个转置U得到：

$$U^T_{r\times m}A_{m\times n}\approx U^T_{r\times m}U_{m\times r}\Sigma_{r\times r}V^T_{r\times n}=\Sigma_{r\times r}V^T_{r\times n}$$

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。