主成分分析（PCA）简明教程（翻译）

作者：Lindsay I Smith

时间：2002.2.26

译者：程明波

英文文章地址

原译文地址

第一章

前言

这是一篇帮助读者理解主成分分析（PCA）的教程。PCA是一种统计技术，在人脸识别和图像压缩等领域都有应用。同时，PCA也是一种高维数据模式发现的一种常用方法。

在讲PCA之前，本文先介绍了PCA用到的一些数学概念。其中包括标准差、协方差、特征向量和特征值等。这些背景知识意在帮助我们理解PCA部分，如果你对这些概念已经非常清晰可以跳过此部分。

示例贯穿于整个教程，以便于通过直观的例子讨论概念。如果你还想了解更多内容，霍华德.安东著有约翰威立国际出版公司出版的数学课本《Elementary Linear Algebra 5e》提供了非常好的这方面数学背景知识。

第二章

数学背景知识

本章试图介绍便于理解主成分分析计算过程所需的基本数学知识。各个主题互相独立，同时各主题会举例说明。理解为什么使用这些技术以及一些关于数据计算结果所告诉我们的比记住枯燥的数学原理更重要。尽管不是所有这些知识都应用于PCA，一些看起来不是直接相关知识是这些最重要技术的基石。

我安排其中的一节介绍统计学知识，主要着眼于分布的度量，或者说数据是怎样离散的，其他的部分主要讲矩阵代数的一些知识，包括特征值、特征向量以及一些PCA所需的重要矩阵性质。

2.1 统计知识

整个统计学都基于你有一个很大数据集的前提，以及你想分析关于数据集中各个数据点的关系。我会介绍一些量度这些数据的一些方法，帮助你理解这些数据本身。

2.1.1 标准差

要了解标准差，我们需要一个数据集。统计学经常会使用总体中的一些采样。以选举为例，总体就是一个国家的所有人，因此一个采样就是统计学家用来度量的总体的一个子集。统计学伟大之处是我们只需度量（例如电话调查等）总体中的采样，你就可以计算出最接近所有整体的度量。

本节我假设我们的数据集是某个很大总体的采样。本节后面会提供总体以及采样的更多信息。这是一个示例数据集：

$$X = [1\ 2\ 4\ 6\ 12\ 15\ 25\ 45\ 68\ 67\ 65\ 98]$$

我们简单地假设字符X表示包含这些所有数字的集合。如果我想表示这个集合中某个单独的数字，我会用X加上下标表示某个具体的数。例：X_3 表示X集合中的第三个数，也就是数字4。注意，有些书用X_0 表示第一个数字，我们这里用X_1 。另外，我们用字符n表示集合中元素的数量。

我们可以计算一个集合的很多维度，比如，我们可以计算样本的均值。这里我假设读者明白什么是一个样本的均值。这里仅给出公式：

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

注意，我们用字符$\overline{X}$标识集合的均值。这个公式表示：把所有的数字加起来再除以他们的数量。

很不幸，均值除了告诉我们某种中心以外，并没有告诉关于数据的更多信息。比如以下两个数据集合的均值(10)完全一样，但是显然它们区别很大。

$$[0\ 8\ 12\ 20] \quad \text{和} \quad [8\ 9\ 11\ 12]$$

那么，这两个集合有何不同呢？这两个集合的离散程度是不同的。一个数据集的标准差（Standard Deviation, 缩写SD）是衡量这个集合数据离散程度的一个指标。怎么计算呢？SD的定义是这样的：每个数据点到这份数据均值点的平均距离。计算每一个数据点到均值点的距离平方，然后相加，再除以$n-1$，再开方，公式如下：

$$s = \sqrt{\frac{\sum_{i=1}^{n} (X - \overline{X})^2}{(n-1)}}$$

这里s常用来标识样本方差。可能有人会问：“为啥分母除以$n-1$而不是n呢？”答案有点儿复杂，大体来说，如果你的数据集合是一个采样，比如，你取样于真实世界（比如调查500人的选举情况）得到一个子集，那么你就必须用$n-1$，因为这个结果比你用n更接近于你用全部的整体算出的标准差。但是，如果你不是计算一个样本的而是整体的标准差，这种情况下，你就应该除以n而不是$n-1$。如果想了解更多的关于标准差的内容，可以访问这里，链接文章用类似的方法讲了标准差，提供了不同分母计算的区别实验，同时还探讨了采样和总体的异同。

数据集1

X	$(X - \overline{X})$	$(X - \overline{X})^2$
0	−10	100
8	−2	4
12	2	4
20	10	100
总计		208
除以(n − 1)		69.333
平方根		8.3266

数据集2

X	$(X-\overline{X})$	$(X-\overline{X})^2$
8	−2	4
9	−1	1
11	1	1
12	2	4
总计		10
除以(n − 1)		3.333
平方根		1.8.257

表2.1标准差计算

从上表2.1，我们可以看到标准差的计算过程。

因此，和我们预想的一样，第一个数据的的标准差要比第二个大得多。原因是数据离散于均值点的程度更高。再举一个例子，数据集:

[10 10 10 10]

的均值也是10，但是它的标准差是0， 因为所有的数字是相同的。没有任何数据点偏离均值。

2.1.2 方差

方差是数据离散程度的另一个度量。实际上，它和标准差几乎相同，公式如下:

$$s^2 = \frac{\sum_{i=1}^{n}(X-\overline{X})^2}{(n-1)}$$

你会注意到方差就是标准差的平方，标识上也有$s(s^2)$。s^2 经常用来标识一个数据集的方差。方差和标准差都是用来衡量数据的离散程度。标准差使用的更普遍，方差也常使用。之所以介绍方差是因为下一节我们介绍的协方差是基于方差的。

练习

计算下列数据集的均值、标准差和方差。

[12 23 34 44 59 70 98]

[12 15 25 27 32 88 99]

[15 35 78 82 90 95 97]

2.1.3 协方差

我们之前介绍的前两种度量方式只针对纯1维情况。1维数据集合可能是这种形式：屋里所有人的身高，或者上学期计算机科目101的成绩等等。但许多数据集是大于1维的情况， 对于这种数据集，统计分析的目标经常是分析不同的维度之间的关系。例如，我们可能有个数据集同时包含了课堂上学生的身高，以及他们论文的分数。接着我们就可以利用统计分析工具来观察学生的身高对他们的成绩是否有影响。

标准差和方差只是对单一维度的计算，因此，你只能对数据的每一个维度单独计算标准差。然而，如果有一种类似的度量能找到各维度相互在偏离均值的变化关系会非常有用。

协方差就是这样一种度量。协方差总是用来度量两个维度，如果你计算一个维度和他自己维度的协方差，这时协方差就退化为这一个维度的方差。因此，如果你有一个3维数据集(x, y, z),协方差的计算公式与方差非常相似。方差的计算公式也可以这样表示：

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \overline{X_i})(X_i - \overline{X_i})}{(n-1)}$$

这里我简单的对平方项进行了展开。有了以上知识，我们现在可以写出协方差的公式了：

$$cov(X, Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X_i})(Y_i - \overline{Y_i})}{(n-1)}$$

除了第二个括号中的X全部被替换成了Y以外，协方差和方差的公式完全一样。我们可以这么表述：“对于每个数据项，把每个x和x均值的差与每个y和y均值的差相乘，再加和除以$(n-1)$”。协方差是怎样的一种工作机理呢？我们这里用一些数据来举例。想象你通过调查得到一个2维数据。假设我们问了一堆学生他们花在科目COSC241的总小时数，以及他们的学期末成绩。现在我们有了两个维度，第一个维度是H，标识学习的小时数，第二个维度是M，标识学生的成绩。图**2.2**展示了我们假设的数据以及两个维度学习小时数和成绩之间的协方差$cov(H, M)$。

这张图告诉我们什么呢？协方差的值没有它的符号重要（正或负）。如果值是正的，比如我们这里，那么意味着两个维度一起增减。即，一般来说，如果学习的小时数增加，那么这个学生最后取得的成绩就会高。

但是如果协方差的值是负的，那么如果其中一个维度增加，另一个维度就会减少。如果我们刚刚计算的协方差的结果是负值。那我们的说法就变成了随着学习小时数的增加，期末成绩会降低。

最后一种情况，如果协方差是0，那么说明两个维度是相互独立的。

我们很容易画一张图如图2.1.3，得出结论：学习成绩随着学习的小时数增加而增加。但是，只有两维或三维这种低维的奢侈情况，我们才能通过可视化观察趋势。由于在一个数据集中可以计算任意两个维度的协方差，这种技术经常是高维数据可视化非常困难的情况下寻找维度之间关系的一种方法。

你可能会问，$cov(X, Y)$与$cov(Y, X)$是否相等？简单一看我们就会发现，它们是完全相等的，因为两个式子计算的唯一不同是在$cov(Y, X)$中$(X_i - \overline{X_i})(Y_i - \overline{Y_i})$被替换成了$(Y_i - \overline{Y_i})(X_i - \overline{X_i})$。我们知道乘法满足交换率，也就是说，无论乘数和被乘数的位置怎么变化，结果都是一样，也就是说这两个协方差结果是相同的。

2.1.4

我们知道，协方差总是用来计算两个维度之间的关系。如果我们有一个超过2维的数据集合，那么我们要计算的协方差的值的数量就不止一个了。比如，一个三维的数据集（x, y, z**三个维度**)。你可以计算的协方差就有$cov(x, y)$、$cov(x, z)$ 和 $cov(x, z)$。事实上，对于一个n维的数据集，你可以计算$\frac{n!}{(n-2)!*2}$ 不同的值。

数据：

	小时数(H)	成绩(M)
数据	9	39
	15	56
	25	93
	14	61
	10	50
	18	75
	0	32
	16	85
	5	42
	19	70
	16	66
	20	80
总数	167	749
平均	13.92	62.42

协方差：

H	M	$(H_i - \overline{H})$	$(M_i - \overline{M})$	$(H_i - \overline{H})(M_i - \overline{M})$
9	39	−4.92	−23.42	115.23
15	56	1.08	−6.42	−6.93
25	93	11.08	−30.58	338.83
14	61	0.08	−1.42	−0.11
10	50	−3.92	−12.42	48.69
18	75	4.08	12.58	51.33
0	32	−13.92	−30.42	423.45
16	85	2.08	−22.58	46.97
5	42	−8.92	−20.42	182.15
19	70	5.08	−7.58	38.51
16	66	2.08	−3.58	7.45
20	80	6.08	17.58	106.89
总数				1149.89
平均				104.54

想求出所有不同维度的协方差，非常有用的方法是把他们全计算出来然后放入矩阵。我假设你对矩阵比较熟悉，以及矩阵怎样定义。因此，对于一个n维的数据集的协方差矩阵：

$$C^{n\times n} = (c_{i,j}, c_{i,j} = cov(Dim_i, Dim_j))$$

，
这里$C^{n\times n}$是一个n行n列的矩阵，Dim_x是第x维。上面非常不美观的公式说的是，如果你有一个n维数据集，那么协方差矩阵就是一个n行n列的矩阵，矩阵的每一个元素是两个维度之间的协方差计算结果。例如，矩阵的第2行第三列就是维度2和维度3之间的协方差计算结果。

一个例子。我们假设有一个3维的数据集，分别使用x,y,z表示3个维度。那么协方差矩阵是一个3行3列的矩阵，矩阵中的元素就是：

$$\begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$$

几个需要注意：主对角线计算的某一维和它自己的协方差，也就是这些维度的方差。剩下的元素，因为$cov(a,b) = cov(b,a)$，所以矩阵关于主对角线对称。

练习

1. 计算以下关于x和y的2维数据集的协方差，然后描述一下协方差结果可能推导出数据什么方面的结论。

项目id	1	2	3	4	5
x	10	39	19	23	28
y	43	13	32	21	20

2. 计算下列3维数据的协方差矩阵：

项目id	1	2	3
x	1	-1	4
y	2	1	3
z	1	3	-1

2.2 矩阵代数

本节会介绍PCA所用到的一些矩阵代数的背景知识，我将重点介绍对给定矩阵计算特征向量和特征值的相关知识。这里我假设你了解矩阵的基本知识。

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

图2.2：非特征向量和1个特征向量

$$2 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} 24 \\ 16 \end{pmatrix} = 4 \times \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

图2.3：缩放特征向量后仍为特征向量

2.2.1 特征向量

如你所知，只要两个矩阵的大小相容，你就可以将两个矩阵相乘。特征向量是矩阵相乘的的特殊形式。我们现在考虑如图2.2所示的矩阵和向量相乘的情况。

第一个例子中，计算结果不是整数与原始矩阵相乘的形式，但到了第二的例子，计算结果的就是一个整数乘以与左边完全相同的一个向量。为何能产生这样的结果呢？实际上，向量就是2维空间的一个矢量。向量$\begin{pmatrix} 3 \\ 2 \end{pmatrix}$(第二个相乘的例子)代表从原点$(0,0)$一个指向$(3,2)$的一个箭头，另一个矩阵可以被认为是变换矩阵。如果你在向量的左边

乘以一个矩阵，结果就是把这个向量从其原始位置进行了变换。

上面说得就是变换就是特这向量的本质。想象一个变换矩阵，以及一个在直线$y = x$上的向量，矩阵左乘这个向量。如果你发现结果仍然位于$y = x$这条直线上，那么这就向是量的自反射。这个向量（所有的乘子，因为我们不关心向量的大小）就是这个变换矩阵的一个特征向量。

这些特征向量有什么性质呢？第一你要知道的就是只有方矩阵才有特征向量。其次是不是所有的方矩阵都有特征向量。最后，如果一个$n \times n$矩阵只要有，那么就一定有n个特征向量。如果一个3×3的矩阵有特征向量，那就有3个。

特征向量的另一个性质是：如果我在相乘之前对其缩放一定量，那么我可以仍然得到相同的乘积形式（如图2.3）。这是因为如果你缩放一个的向量，你做的仅仅是把这个向量变长，而没有改变其方向。最后，一个矩阵的所有的特征向量都是*垂直*的。也就是说，无论你有多少维的向量，他们都是互相形成直角。另一个更数学化的说法叫*正交*。这么描述非常重要，原因是我们可以更方便表述这些垂直的正交向量，而不用在x轴和y轴的坐标系中描述。在PCA介绍部分我们会用到这些。

另一点重要的是，数学家们在寻找特征向量时，他们总喜欢找长度为1的特征向量。原因我们已经知道，向量的长度并不是影响因素，方向才是。所以为了使特征向量有标准形式，我们的一般做法是将其缩放成长度为1的向量。这样，所有的特征向量就都有相同的长度了。下面我们把例子中的向量标准化。

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

是一个特征向量，这个向量的长度为：

$$\sqrt{(3^2 + 2^2)} = \sqrt{13}$$

所以我们把原始的向量除以这个长度，就得到了长度唯一的特征向量。

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} \div \sqrt{13} = \begin{pmatrix} 3/\sqrt{13} \\ 2/\sqrt{13} \end{pmatrix}$$

怎么找到这些神秘的特征向量呢？很不幸，只有当矩阵足够小时，特征向量才好找，比如不超过3×3的矩阵。如果矩阵大小再变大，通常的做法是用复杂的迭代方式求解，这些方法此教程不会讲解。如果你想在程序中使用计算特征向量的方法，很多数学库都有实现，一个有用的数学库包。

如果想进一步了解特征向量和特征值以及正交等内容，请参考霍华德.安东著有约翰威立国际出版公司出版的数学课本《Elementary Linear Algebra 5e》，ISBN 0-471-85223-6。

2.2.2 特征值

特征值和特征向量高度相关，其实我们已经在图2.2看到过特征值。还记得被矩阵缩放以后的特征向量有相同的大小么？在那个例子中，这个值是4。这里4就是特征向量相关的特征值。无论我们对特征向量怎么缩放，我们始终得到的特征值都一直是一样的，如图2.3的例子特征值一直是4。

现在我们发现特征向量和特征值总是成对出现。如果你现在需要某个编程库计算特征向量，通常特征值也被同时计算出来了。

练习

对于下面的矩阵

$$\begin{pmatrix} 3 & 0 & -1 \\ -4 & 1 & 2 \\ -6 & 0 & -2 \end{pmatrix}$$

判断下面是否有此矩阵的特征向量，如果有，请求出对应的特征值。

$$\begin{pmatrix} 2 \\ 2 \\ -1 \end{pmatrix} \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ 3 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}$$

第三章 主成分分析（Principal Components Analysis）

终于到了主成分分析（PCA）部分了，PCA可以在数据中识别模式，并通过此种方式突出数据中相似和不同的部分。由于很难用图像表示高维数据，也就意味着在高维数据中寻找模式变得非常困难，这时，PCA就成了极为强大的数据分析工具。

另一个PCA的优点是，如果你通过其找到了数据中的模式，你还可以用来压缩数据，即，在不损失太多信息的前提下降低数据的维度。这个技术被用于图像压缩，我们在稍后的章节中会有涉及。

本章我们将针对一个数据集，一步一步实现PCA计算。这里我不准备描述*为什么*
PCA表现出色。我做的是为你提供每一步都发生了什么，这样，将来如果你想使用此技术时，就会有足够多的知识帮助你做决策。

3.1 方法

第一步：数据集

在我们这个简单的例子中，我会使用我编造的一个数据集。这个数据集只有两维，之所以选择这份数据是因为我可以通过画出图形来分析PCA的每一步都发生了什么。

第二步：减掉均值

如果想实现PCA，我们首先要把每一维的数据减掉均值。就是说要对每一维求平均值，接着把每一维的每个数据都减掉均值。我们这里所有的x值都要减掉$\bar{x}$(x维度所有数据的均值)，所有的y值都减掉$\bar{y}$。这样我们就构造了一个均值为0的数据集。

数据 =

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

调整后的数据 =

x	y
0.69	0.49
−1.31	−1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	−0.31
−0.81	−0.81
−0.31	−0.31
−0.71	−1.01

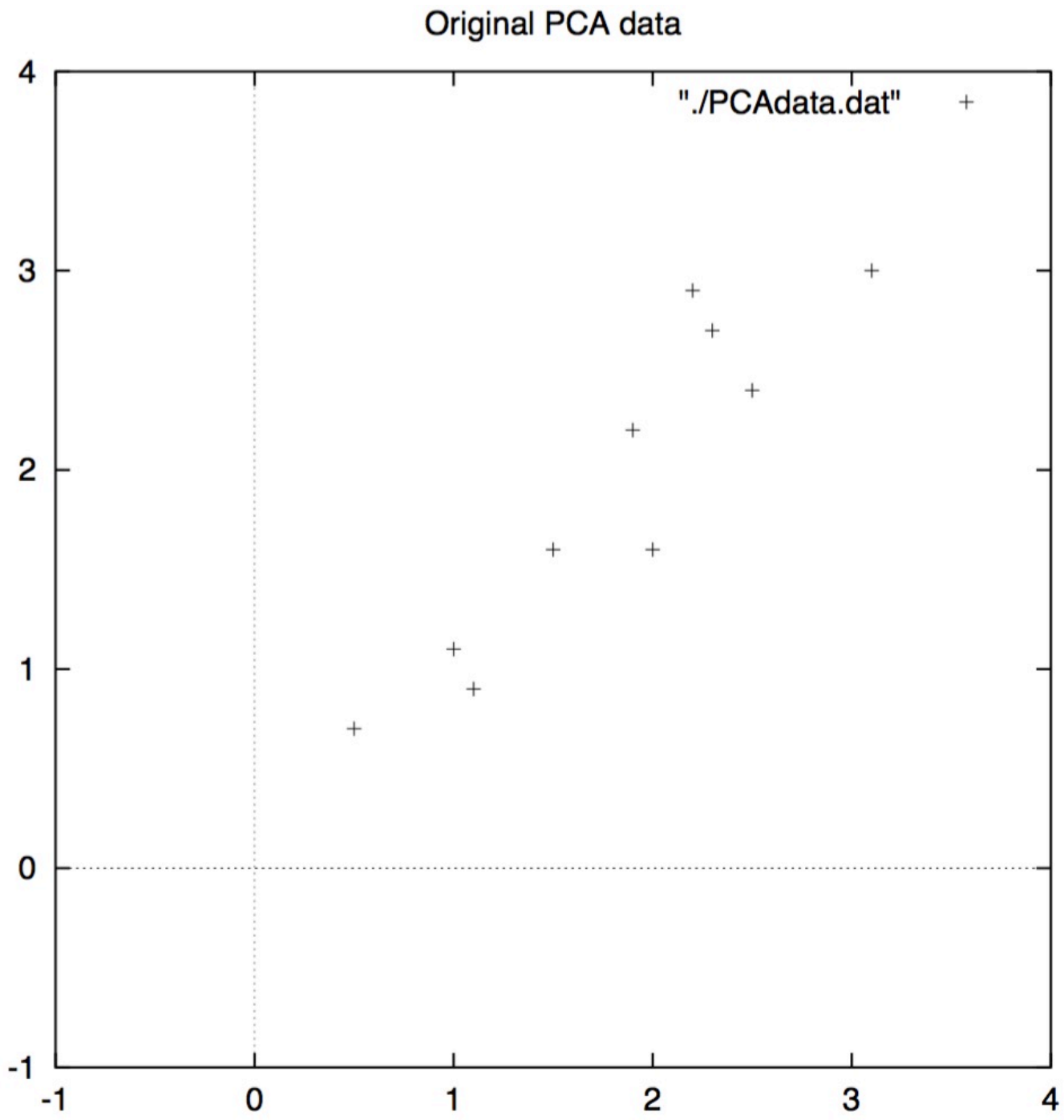

图3.1：*PCA*示例数据，左边为原始数据，右边为减掉均值的数据

第三步：计算协方差矩阵

协方差矩阵我们在2.1.4小节已经讨论过。由于我们的数据是2维的，所有协方差矩阵就是2×2。协方差矩阵计算没特别说明的，我直接给出结果：

$$cov = \begin{pmatrix} 0.616555556 & 0.615444444 \\ 0.615444444 & 0.716555556 \end{pmatrix}$$

由于协方差矩阵非对角线元素都是正值，所以我们可以预期x和y一起增减。

第四步：计算协方差矩阵的特征向量和特征值

协方差矩阵是方阵，所以我们可以计算其特征向量和特征值。这极为重要，因为他们可以告诉我们关于数据的有用信息。我一会儿会说明原因，现在我们来看一下特征值和特征向量：

$$\textbf{特征值} = \begin{pmatrix} \mathbf{0.0490833989} \\ 1.28402771 \end{pmatrix} \textbf{特征向量} = \begin{pmatrix} \mathbf{-0.7351178656} & -0.6778873399 \\ 0.677873399 & -0.735178656 \end{pmatrix}$$

一定要注意到两个特征向量都是单位向量。也就是说他们的长度都是1。这个结果对PCA非常重要，幸运的是，大部分数学工具包计算特征向量提供的都是单位向量。

那么，这些计算结果都是什么意思呢？如果你观察图3.2的数据点，你会发现这些数据有非常强的模式。和我们用协方差预期的一致，这些数据确实一起增减。我利用数据同时也画了两个特征向量，这两个特征向量看起来像图3.2的对角线。我们在特征向量小节介绍过，两个特征向量是相互垂直的。但是，更重要的是特征向量为我们提供了数据中的模式信息。可以看出，其中一条线（译者注：大概45度倾角的这条线）看起来像画了拟合这些数据点的一条线。这个特征向量告诉我们两个数据维度沿着线的相关性（译者注：原文是两个数据集，我认为是两个数据维度）。第2个特征向量（译者注：大概135度倾角的这条线）给我们提供了另外一些重要性稍低的数据中的模式，数据点分布在线的两边。

因此，通过从特征矩阵中取出特征向量进行分析，我们已经提取出了刻画数据特点的线。接下来的步骤会包括数据变换以便于用我们这些线来表达数据。

第五步：选择成分及构建特征的向量

现在我们来讨论数据压缩和降维的概念。如果你学习了前面小节的特征向量和特征值的相关信息，你会注意到特征向量是之间有很大不同。事实上，具有更大特征值对应的特征向量是数据集的主成分(Principal Component)。在我们这个例子中，对应更大特征值特的特征向量是基本拟合数据点的这条线。这维特征向量描述了数据维度之间最重要的关系。

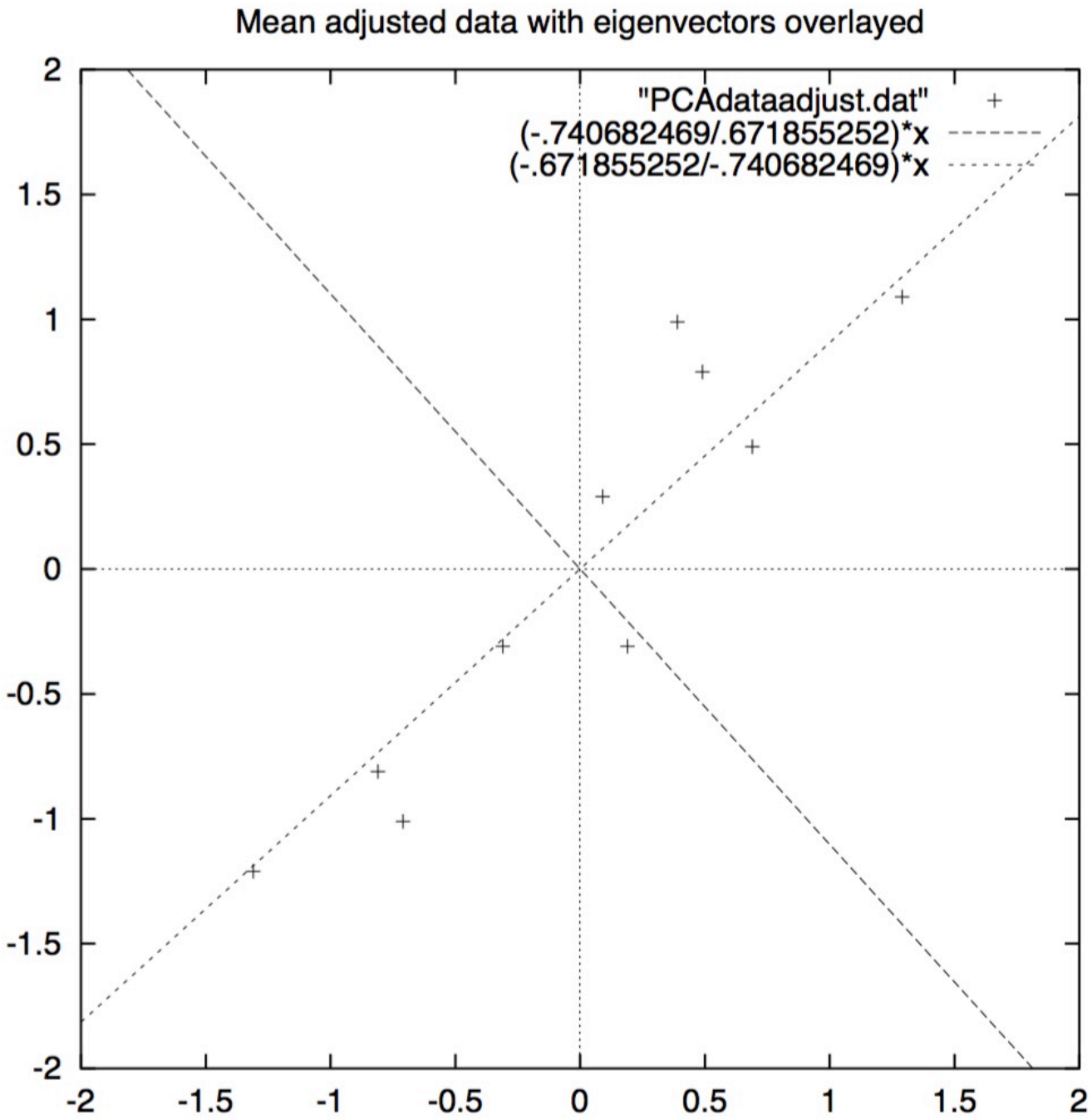

图3.2：**标准化（减掉均值）后的数据图以及协方差矩阵中特征向量图**

一般来说，一旦从协方差矩阵中找出特征向量，下一步我们要做的就是把他们对应的特征值从高到低排列。排序表明了成分（component）的重要性高低。现在，如果你如果愿意，可以忽略那些重要性没那么高的成分，你就会丢失一些信息，但是如果特征值非常小，你丢失的信息并不会太多。如果你扔掉一些成分，最终的数据集的维度会低于原始数据的维度。具体来说，如果你的原始数据有n维，因此你可以计算出n个特征向量和n个特征值，接下来如果你只选取前p维特征向量，那么最终的数据就变成了p维的数据集。

现在你需要做的是构造一个特征(feature)的向量，其实就是一个向量的矩阵。矩阵是通过挑选你希望留下的特征向量，组成一个每列1个特征向量的矩阵(译者注：最后一维我认为是第p维更好，可以与上面一段对应)。

$$\textbf{特征的向量} = (\text{eig}_1, \text{eig}_2, \text{eig}_3, \ldots, \text{eig}_p)$$

来看我们的例子，现在我们有2个特征向量，我们现在有两个选择，第一选择是两个特征向量都被用于构造特征的向量：

$$cov = \begin{pmatrix} -0.77873399 & -0.735178656 \\ -0.735178656 & 0.677873399 \end{pmatrix}$$

或者，我们可以扔掉不重要的成分，那么特征的向量只有1列：

$$cov = \begin{pmatrix} -0.677873399 \\ -0.735178656 \end{pmatrix}$$

下一节我们将针对上面两种新的数据集进行讨论。

第六步：生成新数据集

这是PCA的最后同时是最简单的一步。一旦我们选择了我们希望保留到成分（特征向量集），我们只需把特征的矩阵转置，左乘调整后的数据（原始数据减掉均值），然后再转置。

最终数据 = 行特征的向量 × 行调整后的数据

这里**行特征的向量**是特征的向量组成的矩阵进行转置，也就是说现在特征向量现在是以行的形式排列，最重要的特征向量在第一行。**行调整后的数据**是经过均值调整后的数据，也进行了转置，也就是说数据项在每一列，而每行是一个独立的维度。抱歉数据转置可能来得有点儿突然，但是如果我们现在对特征向量的矩阵和数据进行转置，后面的公式就会简单很多，而不是一直带着个转置的上标符号T。**最终数据**是最终的数据集合，其中每一列是一个数据项，每一行是一个维度。

做完这些我们可以得到什么呢？我们可以得到和我们选择向量完全相关的原始数据。我们的原始数据有x轴和y轴两个坐标的坐标系，所以我们的数据与这两个坐标的坐标系相关。其实你可以用任何你喜欢的两个坐标轴的坐标系来表示你的数据。如果坐标轴互相垂直，这种表示方法是最高效的，这就是为何特征向量间互相垂直这么重要。现在我们已经把我们的数据从跟x轴和y轴相关改为2个特征向量组成的坐标系相关。如果说我们已经通过降维构造了新的数据集，也就是说我们扔掉了一些特征向量，那么新数据只跟我们留下的特征向量相关。

为了展示我们的数据，我已经把两种可能的特征的向量都对数据做了变换。我已经对每种情况的结果进行的了转置，这样我就把数据恢复成表结构的组织形式。同时，我也把最终的数据点画了出来，这样我们就可以观察这些数据点与这些成分之间的关系。

两个特征向量都保留的情况转换后的结果见图3.3。这个图其实就是原始数据旋转后，这样特征向量就成了坐标轴。这种情况很好理解，因为我们在分解的过程中并没有丢失任何信息。

另外一种变换，我们只保留有最大特征值的特征向量，我们可以从图3.4中看的数据的结果。和预期的一样，这个数据只有一维。如果你用这份数据与两维特征向量都用变换后的数据对比，你会注意到，这个数据就是另一份数据的第一列。所以，如果你画出这个数据的图，这份数据只有一维，那么结果其实就是图3.3数据点x的坐标点。我们其实就是高效的抛弃了其他的坐标轴，也就是其他的特征向量。

那么我究竟做了什么呢? 本质上我们把数据进行了变换，使之可以用相关的模式进行表示，这些模式就是一些最适合描述这些数据之间关系的线。这么做非常有用，因为我们现在已经把数据点对每条线的贡献进行分类，然后进行组合。首先，我们仅仅有x轴和y轴，这还不错，但是每个x和y的数据点其实并无法告诉我们，每个数据点和其他数据点之间的关系。现在数据点的值可以精确告诉我们数据点处于趋势线的位置（上面或者下面）。如果是两个特征向量都用的情况，我们仅仅是把数据转换以便于我们使这些数据与特征向量相关，而不是x轴和y轴。但是只留一维特征向量的分解移除了较小特征向量的贡献，是我们的数据只与保留的一维数据相关。

3.1.1 把旧数据找回来

显然，如果你用PCA对数据进行压缩，你一定想把原始数据恢复回来。（下一章我们看到例子）这些内容来自于这里。

转换后的数据 =

x	y
−0.827970186	−0.175115307
1.77758033	0.142857227
−0.992197494	0.384374989
−0.274210416	0.1304117207
−1.67580142	−0.209498461
−0.912949103	0.17528282444
0.0991094375	−0.349824698
1.14457216	0.0464172582
0.438046137	0.0177646297
1.22382056	−0.162675287

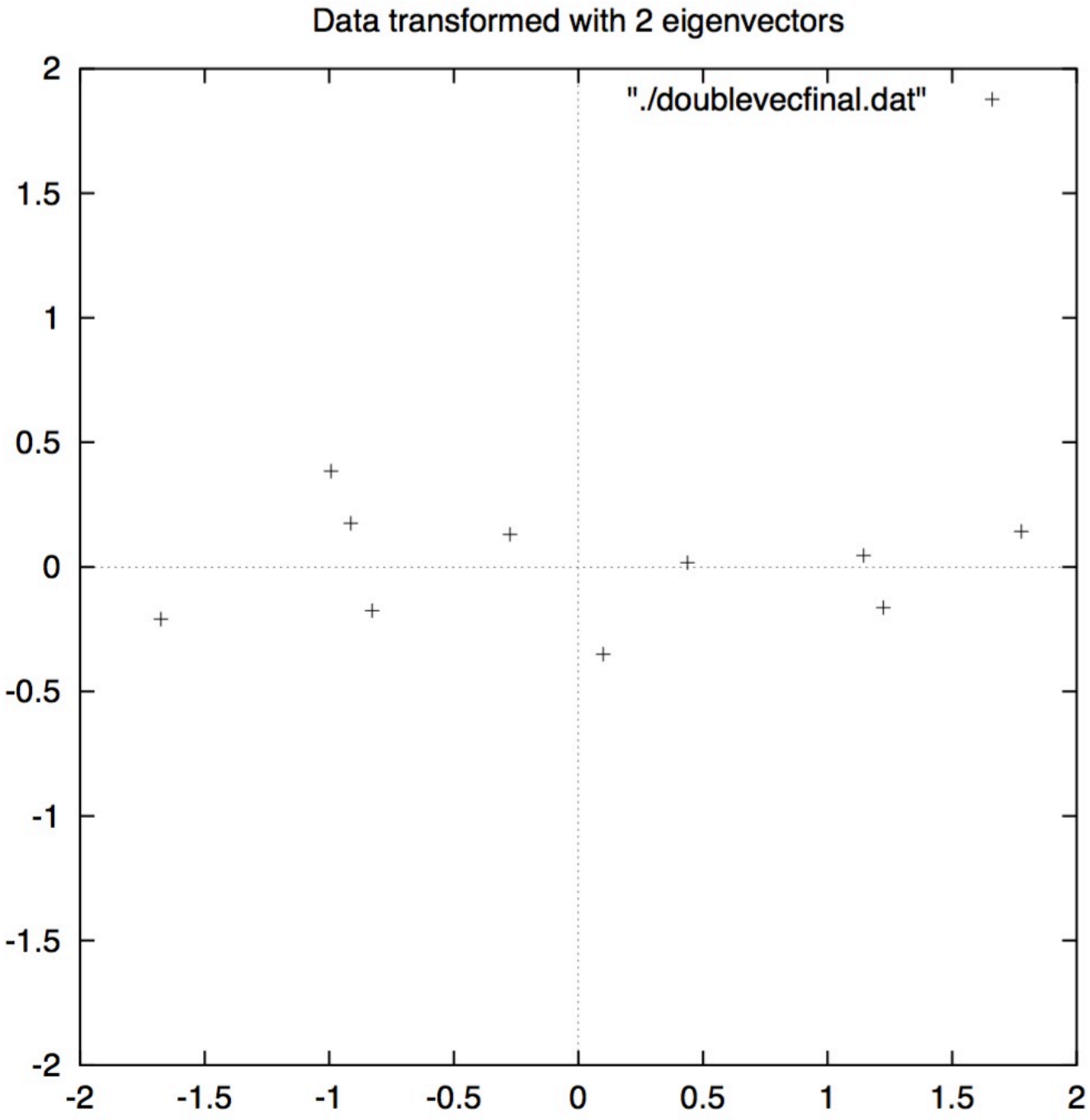

图3.3：应用了*PCA*分析后并使用了两个特征向量的数据表以及绘制的新数据点

转换后的数据(单一特征向量)

x
−0.827970186
1.77758033
−0.992197494
−0.274210416
−1.67580142
−0.912949103
1.14457216
0.438046137
1.22382056

图3.4：只用最重要的特征向量数据转换的数据

所以，我们怎么把原来数据恢复回来？在我们进行恢复原始数据之前，回忆只有我们将所有特征向量进行转换才能精确的把数据恢复回来。如果我们在最后转换时减少特征向量，那么恢复的数据已经失去很多信息。
回想一下，最后的变换是：

$$\textbf{最终数据} = \textbf{行特征的向量} \times \textbf{行调整后的数据}$$

我们可以把公式反转过来，进而得到原始数据，

$$\textbf{行调整后的数据} = \textbf{行特征的向量}^{-1} \times \textbf{最终数据}$$

这里，$\textbf{行特征的向量}^{-1}$是的**行特征的向量**的逆。由于我们讨论的是特征向量，组成的特征的向量，所以，$\textbf{行特征的向量}^{-1}$其实就是**行特征的向量**的转置。当然，只有在矩阵中的所有元素是由单位特征向量组成是才成立。这样，恢复原始数据又变得容易了很多，现在公式变成了：

$$\textbf{行调整后的数据} = \textbf{行特征的向量}^{\mathrm{T}} \times \textbf{最终数据}$$

这个公式在我们只保留部分特征向量的情况下仍然成立。也就是说，就算你扔掉了一下特征向量，上面的公式仍然成立。

我不会演示用所有特征向量恢复原始数据，因为这样计算的结果和开始的数据一模一样。但是，我们一起来看一下只保留了一维特征向量的情况下，是怎样损失信息的。图3.5展示了丢失信息的情况。我们把图中的数据点与图3.1对比一下就会发现，沿着主特征向量的变化被保留下来了（见图3.2特征向量及数据）沿着其他成分（另一个特征向量被扔掉了）的变化丢失了。

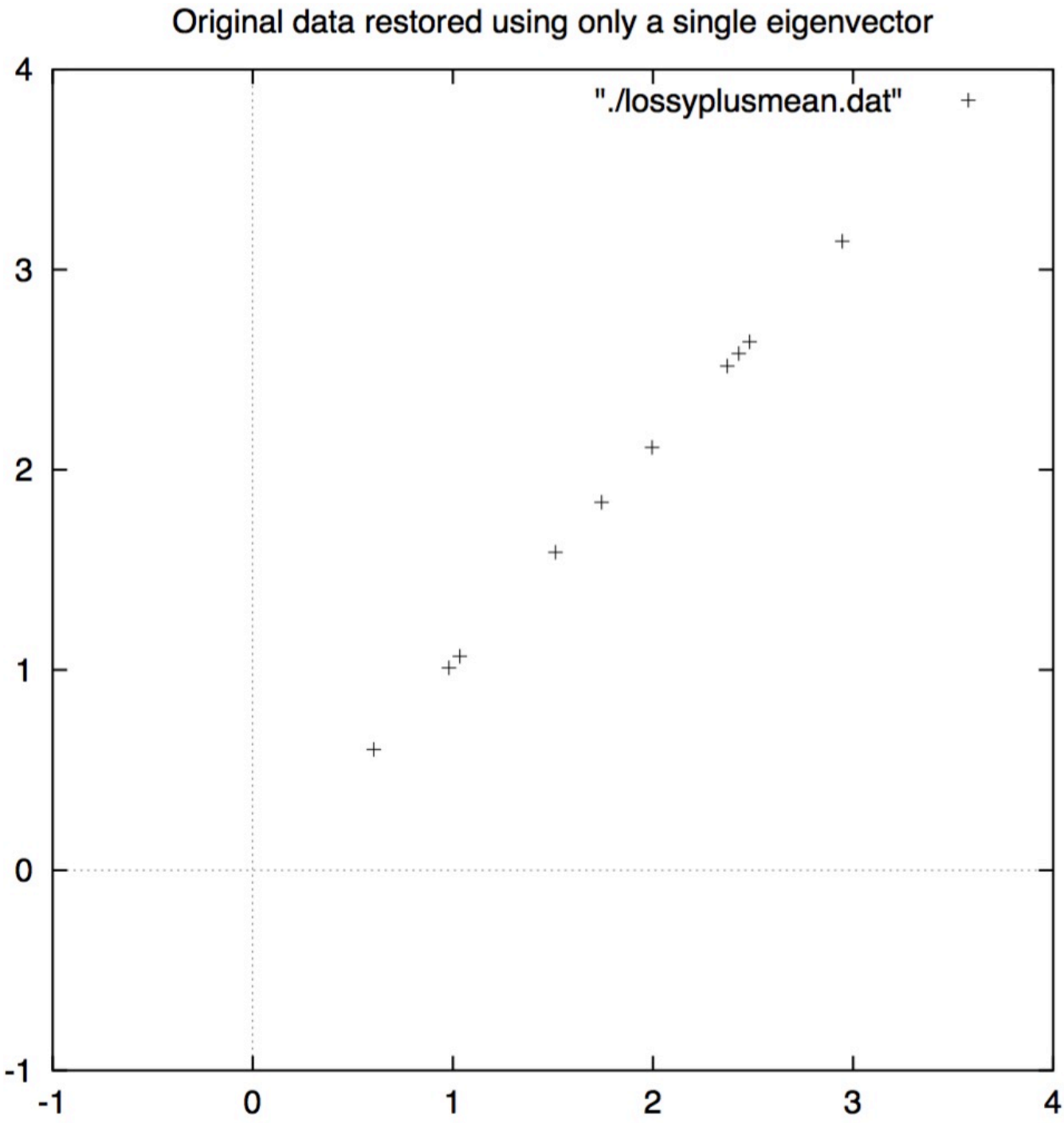

图3.5从单独一维特征向量重新构造的数据

练习

1. 协方差矩阵的特征向量为我们提供了什么呢?
2. 我们在PCA计算的过程中，哪一步可以决定压缩数据，压缩可以起到什么效果呢?
3. 举例说明PCA在图像处理怎样用主成分表示，同时调研一下人脸识别中“特征脸”(Eigenfaces)主题。

第四章 计算机视觉应用

本章我们将简单的PCA在计算机视觉领域的应用，首先我们看一下图像是怎么表示的，然后我们我们看看能怎样用PCA处理这些图像。本章关于人脸识别的的信息主要来自于1997年IEEE 9月Vol 85, No. 9《Face Recognition: Eigenface, Elastic Matching, and Neural Nets》。图像表示来自于爱迪生-韦斯利出版社1987年出版的由Rafael C. Gonzalez 和Paul Wintz合著的《Digital Image Processing》想了解更多信息，KL变换相关知识也是非常好的参考。图像压缩相关知识来自于[这里][5]，此网站还提供了大量用不同数量特征向量重新构造图像的方法。

4.1 表示

在我们把一系列矩阵技术应用于计算机视觉上时，我们必须考虑图像的表示方法。一个正方形，$N \times N$的图像可以被表示成N^2维的向量。

$$X = (x_1, x_2, x_3, \ldots, x_{N^2})$$

这里，第一行前N个$(x_1 - -x_n)$一个挨着一个的像素点组成了1维的图像，下N个元素是下一行，以此类推。每个像素点的值代表图像三原色的亮度，也可能是只是灰度图像，那么只需要1个单独的值即可表示。

4.2 PCA寻找模式

假设我们有20个图像。每个图像的像素非常高。对每个图像，我们都建立一个图像向量表示相应图像。接着我们就可以把所有的图像放到一个像这样的大矩阵中：

$$\text{图像矩阵} = \begin{pmatrix} ImageVec_1 \\ ImageVec_2 \\ \vdots \\ ImageVec_{20} \end{pmatrix}$$

我们现在就可以开始以这条图像矩阵为基始，应用PCA，先构造协方差矩阵，然后得到原始数据相关的特征向量。为什么用PCA分析有用呢？假设我们要做人脸识别，那我们的原始数据就是很多人脸。接下来的问题是，给一张我新的图片，那么这是原始人脸数据中谁的人脸呢（注意，这新的图片不是我们开始给的20个人脸图片）？计算机视觉的处理方法是衡量新的图片和原始图片的差别，但并不是在原始坐标系进行对比，而是在PCA分析的生成的坐标系下衡量。

在实际应用中，PCA生成的坐标系下识别人脸会好非常多，因为PCA分析已经提供了原始图片中不同和相似等相关性。主成分分析已经识别出了数据中的统计学模式。

因为所有的向量都是N^2维的，我们最后会得到N^2个特征向量，在实践中，我们可以扔掉其中不重要的一些特征向量，识别效果仍然非常好。

4.3 PCA图像压缩

使用PCA做图像压缩常常也被称作霍特林变换或者是KL变换（Karhunen-Leove transform）. 如果我们有20个图像，每个图像有N^2个像素，所以我们就可以构造N^2个向量，每个向量20维。每个向量由每个图片的相同像素点的图片亮度值组成。这与我们之前的例子不同，因为之前我们是有一个图像的向量，向量里的每项都是不同的像素。然而我们现在是有一个每个像素的向量，向量里的每项是都是来自于不同的图片。

如果现在我们在一个数据集上应用PCA，那么，我们将会得到20个特征向量，因为，每个向量都是20维的。如果想要压缩数据，我们可以选择只用其中一部分特征向量变换，假设是15个特征向量。这样我得到的最终数据只有15维，达到了节省空间的目的。但是，当要恢复原始数据是，图像已经丢了一些信息。这种压缩技术叫做有损压缩，因为解压后的图片已经不是和原始图片完全一样的图片了，一般来说会变差。

附录 A

实现代码

这份代码用于可替换Matlab的自由软件Scilab。我用这份代码生成了文章的所有例子。除了第一个宏，剩下的都是我(原文作者)写的。

```
1 // This macro taken from
```

```
// http://www.cs.montana.edu/ ~harkin/courses/cs530/scilab/macros/cov.sci // No alterations made
// Return the covariance matrix of the data in x, where each column of x
// is one dimension of an n-dimensional data set.  That is, x has x columns
// and m rows, and each row is one sample.
//
// For example, if x is three dimensional and there are 4 samples.
// x=[123;456;789;101112]
// c=cov(x)
function [c]=cov (x)
// Get the size of the array
sizex=size(x);
// Get the mean of each column
meanx = mean (x, "r");
// For each pair of variables, x1, x2, calculate
// sum ((x1 - meanx1)(x2-meanx2))/(m-1)
for var = 1:sizex(2),
        x1 = x(:,var);
        mx1 = meanx (var);
        for ct = var:sizex (2),
                x2 = x(:,ct);
                mx2 = meanx (ct);
                v = ((x1 - mx1)' * (x2 - mx2))/(sizex(1) - 1);
end, c=cv;
end,
cv(var,ct) = v;
cv(ct,var) = v;
// do the lower part of c also.
// This a simple wrapper function to get just the eigenvectors
// since the system call returns 3 matrices
function [x]=justeigs (x)
// This just returns the eigenvectors of the matrix
[a, eig, b] = bdiag(x);
x= eig;
// this function makes the transformation to the eigenspace for PCA
// parameters:
// adjusteddata = mean-adjusted data set
// eigenvectors = SORTED eigenvectors (by eigenvalue)
// dimensions  = how many eigenvectors you wish to keep
//
// The first two parameters can come from the result of calling
// PCAprepare on your data.
// The last is up to you.
function [finaldata] = PCAtransform(adjusteddata,eigenvectors,dimensions) finaleigs = eigenvectors(:,1:dimensions);
prefinaldata = finaleigs'*adjusteddata';
finaldata = prefinaldata';
// This function does the preparation for PCA analysis
// It adjusts the data to subtract the mean, finds the covariance matrix,
// and finds normal eigenvectors of that covariance matrix.
// It returns 4 matrices
// meanadjust = the mean-adjust data set
// covmat = the covariance matrix of the data
// eigvalues = the eigenvalues of the covariance matrix, IN SORTED ORDER
// normaleigs = the normalised eigenvectors of the covariance matrix,
// IN SORTED ORDER WITH RESPECT TO
```

```
// THEIR EIGENVALUES, for selection for the feature vector.
//
// NOTE: This function cannot handle data sets that have any eigenvalues
// equal to zero. It’s got something to do with the way that scilab treats
// the empty matrix and zeros.
//
function [meanadjusted,covmat,sorteigvalues,sortnormaleigs] = PCAprepare (data) //
Calculates the mean adjusted matrix, only for 2 dimensional data
means = mean(data,"r");
meanadjusted = meanadjust(data);
covmat = cov(meanadjusted);
eigvalues = spec(covmat);
normaleigs = justeigs(covmat);
sorteigvalues = sorteigvectors(eigvalues’,eigvalues’);
sortnormaleigs = sorteigvectors(eigvalues’,normaleigs);
// This removes a specified column from a matrix
// A = the matrix
// n = the column number you wish to remove
function [columnremoved] = removecolumn(A,n)
inputsize = size(A);
numcols = inputsize(2);
temp = A(:,1:(n-1));
for var = 1:(numcols - n)
        temp(:,(n+var)-1) = A(:,(n+var));
columnremoved = temp;
// This finds the column number that has the
// highest value in it’s first row.
function [column] = highestvalcolumn(A)
inputsize = size(A);
numcols = inputsize(2);
maxval = A(1,1);
maxcol = 1;
for var = 2:numcols
        if A(1,var) > maxval
                maxval = A(1,var);
end,
        end,
column = maxcol
maxcol = var;
25
end,
// This sorts a matrix of vectors, based on the values of
// another matrix
//
// values = the list of eigenvalues (1 per column)
// vectors = The list of eigenvectors (1 per column)
//
// NOTE:  The values should correspond to the vectors
// so that the value in column x corresponds to the vector
// in column x.
function [sortedvecs] = sorteigvectors(values,vectors)
inputsize = size(values);
numcols  = inputsize(2);
highcol = highestvalcolumn(values);
sorted = vectors(:,highcol);
remainvec = removecolumn(vectors,highcol);
```

```
remainval = removecolumn(values,highcol);
for var = 2:numcols
        highcol = highestvalcolumn(remainval);
        sorted(:,var) = remainvec(:,highcol);
        remainvec = removecolumn(remainvec,highcol);
        remainval = removecolumn(remainval,highcol);
end,
sortedvecs = sorted;
// This takes a set of data, and subtracts
// the column mean from each column.
function [meanadjusted] = meanadjust(Data)
inputsize = size(Data);
numcols = inputsize(2);
means = mean(Data,"r");
tmpmeanadjusted = Data(:,1) - means(:,1);
for var = 2:numcols
        tmpmeanadjusted(:,var) = Data(:,var) - means(:,var);
meanadjusted = tmpmeanadjusted
end,
```