瞧这里！完完全全说透PCA算法原理外加代码实现！

AI派 昨天

以下文章来源于海边的拾遗者 ，作者Jones

海边的拾遗者

一名既爱生活也爱算法的修行者。 不定期更心得，生活，学习笔记(包括 ML/GL/NLP/...

众所周知，PCA(principal component analysis)是一种数据降维的方式，能够有效的将高维数据转换为低维数据，进而降低模型训练所需要的计算资源。

以上是比较官方的说法，下面是人话(正常人讲的话)版。

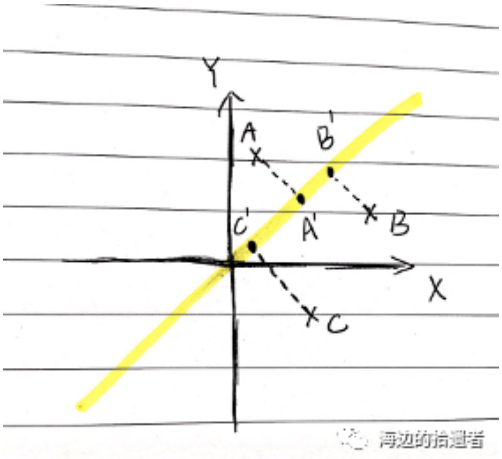
pca就是一种能够有效压缩数据的方法！打个不太准确的例子：我们现在有一个描述人的健康状况的数据：(年龄，身高，体重，地区，血压，心率，肺活量，体温，体脂），也就是说(年龄，身高，体重，地区，血压，心率，肺活量，体温，体脂）这些东西是特征，而这些特征对应的标签是健康状况：健康/亚健康/不健康。那么pca就是通过一些方法，将这9个特征压缩到只有4个，3个甚至更少的特征(暂且称之为x1，x2，x3，x4)，但是我们仍能用这些特征来准确预测它们对应的健康状况。

在这里补充一下，在数学/机器学习中，我们会把特征抽象为向量，比如上面提到的健康状况的数据：(年龄，身高，体重，地区，血压，心率，肺活量，体温，体脂），我们会抽象为（18，180，70，广东，100，80，5000，37，0.1），其中地区这一项显得与众特别，毕竟其他的维度都是数值，就它是文字。我们把这样的维度称为类别，因为它是在有限的选项中选出来的（从世界上所有的地区中取一个），在计算机中表示这样的信息，我们可以有很多方式，但是为了简化难度，这边我就暂且不搞，直接把这一列删掉。于是乎我们的数据（其实就是向量！）就是（18，180，70，100，80，5000，37，0.1），值得一提的是，这样的一个数据是属于一个实体的（也就是说这是描述一个人的健康状况的），在机器学习中，我们倾向于将一个实体的数据排成一列，也就是（18，180，70，100，80，5000，37，0.1）^T（转置）。

本文要介绍的目录为:

- 使用PCA的必要性
- PCA的本质
- 前置知识的介绍
- PCA的数学原理
- PCA的思想
- PCA的实现

使用PCA的必要性



前面说了，pca就是将高维（很多列属性）数据转换为低维（较少列）数据的方法，同时保留大部分信息（可以用保留的信息准确预测）。但是我们可能会想：如果我不压缩的话，那我不就可以有100%的数据吗？我闲着没事干压缩干哈？其实我一开始使用的时候也有这样的疑惑，因为我一开始是用在图像上的，而一个图像只有500多个维度（列）的数据，使用pca压缩到100列可以保存原始数据95%的信息，但是我发现我用压缩的数据和不压缩的数据对模型的训练速度并没有什么影响。。。但是后来我做其他一些有500000维度的数据的时候，发现使用pca将维度降到5000就能保存接近98%的数据，而且训练速度可以提升数十倍！于是我就成了pca的脑残粉了。。。所以pca在应对高维度数据的时候是有奇效的！它不仅可以有效减少训练时间而且还可以防止过拟合，前面一点上文已给出原因，防止过拟合在下文给出。

PCA的本质

其实pca的本质很简单，上面也有说，就是将高维度数据转换到低维度，不过在这里为了让大家能够有所体会，我使用2维数据降到1维在解释这点。

如上图所示，假设我们的原始数据A，B，C是在直角坐标系中的三个点，它们的坐标分别为A(x_a，y_a)，B(x_b，y_b)，C(x_c，y_c)，那么我们现在想要使用pca，将这三个在平面上的点降维到直线上(也就是上图中黄色的线上)。那么现在的问题就是：

- 平面中的A，B，C点（高维数据）可以通过怎样的映射关系降维到黄线上（也就是高维的数据如何在低维中表示）。
- 这条黄线（就是低维）怎么求/确定？

前置知识的介绍

对于上面提到的题一个问题（如何将高维度数据映射到低维度中），我们需要先知道数据点如何被表示。

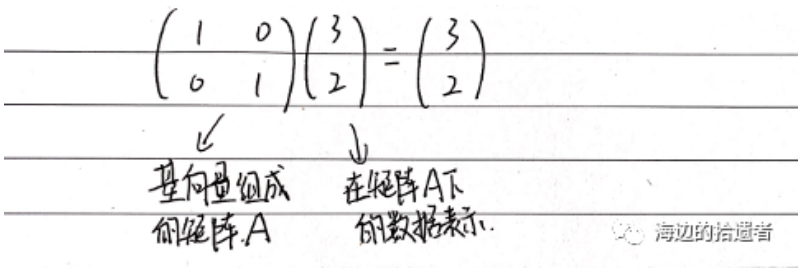
这看起来似乎是一个很蠢的问题，因为答案貌似很简单，比如图xx中的点ABC不就是A(x1，y1)，B(x2，y2)，C(x3，y3)吗？对滴！这个答案是没有问题的，但是这样的答案并不具有普遍性，也就是说如果我们的坐标系发生了变化（类比直角坐标系变化到极坐标系），那就不能再用（x，y）这样的形式进行表示了，那么我们这里需要更加普遍的方法。

我先说答案，再说为什么是这个答案～。答案就是通过坐标系的基向量来表示数据（向量）。如图所示，我们取i和j作为基向量（在这里i的坐标为(1,0)，j的坐标为(0，1)），那么数据A(1，2),就可以表示为(1*i，2*j)。于是我们把这个问题拓展开来，二维上的数据点可以通过(基向量i*数据点在基向量i上的投影长度，基向量j*数据点在基向量j上的投影长度)表示，那么三维上的数据点也可以用这样的方式，于是乎n(n>=2)维上的点可以表示为：(基向量i*数据点在基向量i上的投影长度,基向量j*数据点在基向量j上的投影长度,…,基向量n * 数

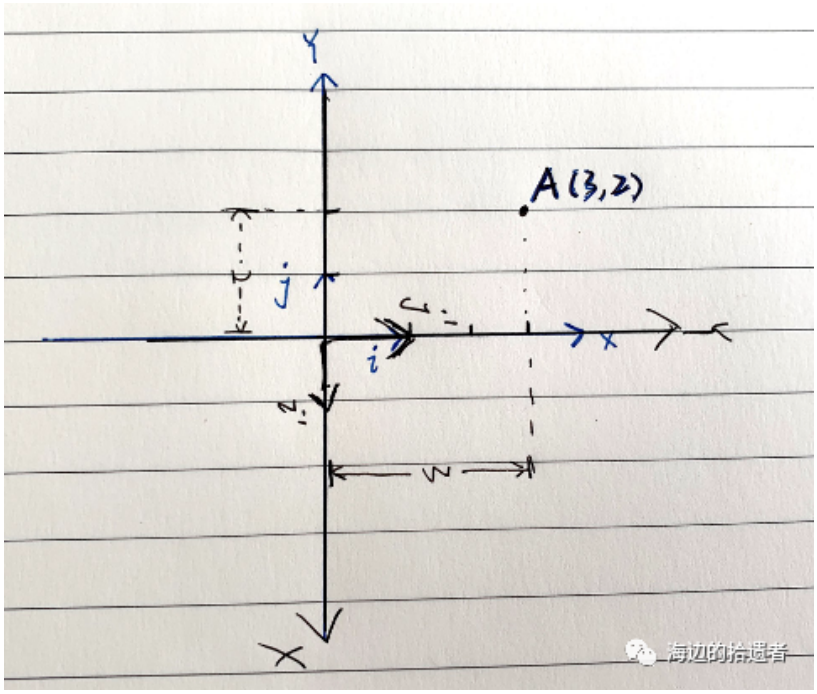
据点在基向量n上的投影长度)，于是乎我们这个子问题就解决了，即找到了一种在不同维度坐标系下表示数据的方法。

PCA的数学原理

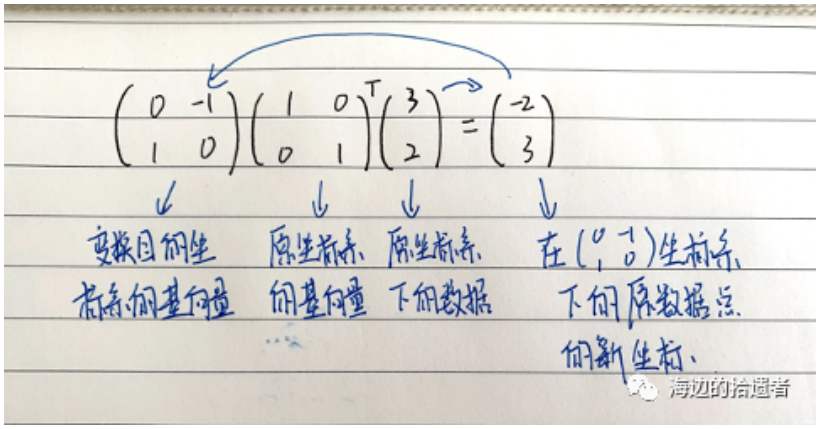
那么接下来的问题就是，我们如何把一个数据点从一个维度转变到另一个维度。



在解决这个问题之前，我们先用矩阵描述一下上一个问题，比如我们现在基向量为(0,1)和(1,0)的坐标中表示(3,2)，则可以写作为：



假设我们现在有一个新的坐标系，这个坐标系的基向量 i 和 j 在普通平面直角坐标系中的表示是(0，-1)和(1，0)，(其实就是普通直角坐标系顺时针旋转90度)，如下图所示(黑色为新的坐标系)：



A点在普通直角坐标系中为(3，2)，在新的直角坐标系中为(-2，3)。新的坐标(-2，3)可以通过以下方式计算：

于是乎我们找到了二维空间下数据变换的方式：

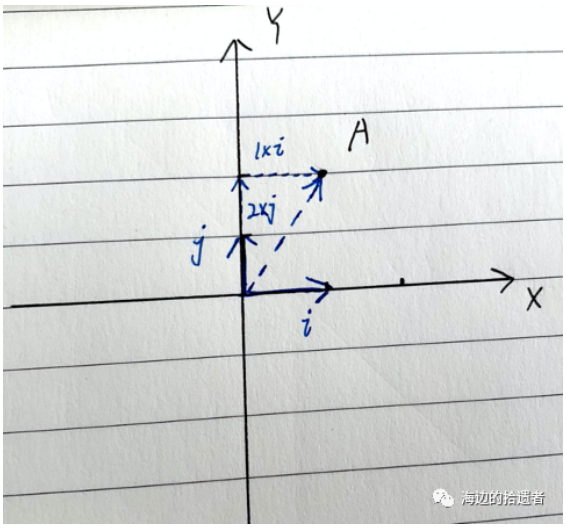
新的基向量矩阵 * 原基向量矩阵的转置 * 原数据向量 = 新的数据向量

也就是说我们想要将高维数据转换为低维数据可以通过：

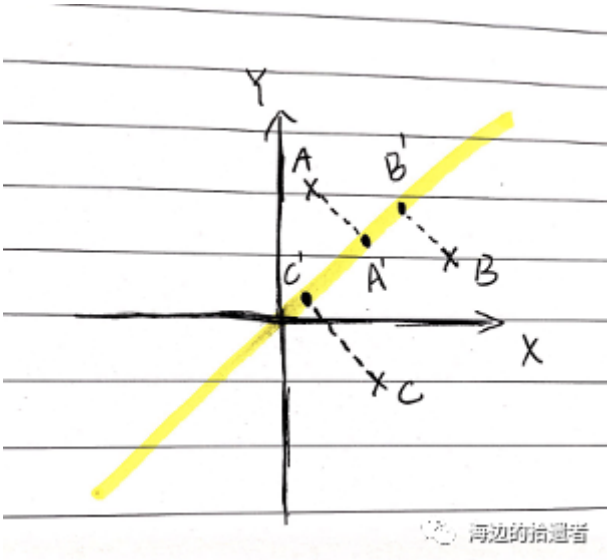
低维空间的基向量矩阵 * 高维空间的基向量矩阵的转置 * 高维数据向量 = 低维数据向量

而参考上图，我们可以知道‘高维空间的基向量矩阵的转置 * 高维数据向量’是等于高维数据向量本身的，于是乎可以得到：

低维空间的基向量矩阵 * 高维数据向量 = 低维数据向量(此处应有数学公式)



接下来我们解决第二个大问题，也就是如下这条黄线（就是低维）怎么求/确定？



PCA的思想

这样要确定低维，要么就要给出标准，也就是什么样的低维是好的？

在这里先确定两个标准，稍后解释为什么确定这两标准：

- 不同特征之间的方差尽可能大。
- 不同特征之间的协方差等于0。

设立这两个标准的原因是这样的：

先做一个前提假设哈！

假设我们现在有两个样本(在这个例子中就是两个人)，他们的健康状况如下：

年龄	身高	体重	地区	血压	心率	肺活量	体温	体脂	健康水平	姓名
18	180	70	广东	100	80	5000	37	0.1	健康	小王
38	175	80	北京	120	70	3000	36.4	0.3	亚健康	老丁

好了，我要来解释了。

第一个标准的解释其实不算太难，假设我们现在要处理上面的数据，也就是要将小王和老丁的数据的进行降维，而他们的健康数据包含9个特征（健康水平是算作label而不是特征X，相当于y=f(X)中的y），理想状态是每个特征描述的东西都是完全不同的（因为特征描述的是对象的特性，如果两个特征描述的东西很类似甚至可以被代替，那就浪费了大把的计算资源了。比如说现在有这样两个特征，第一个特征是：是否为男性，第二个特征是：是否为女性。如果第一个特征为真，则第二个特征必定为假，也就是这两个东西描述的都是同一个特性，就是性别），也就是说在原始数据中，不同的特征它们之间的方差应该是很大的（可以理解为方差越大，这两个东西越不同）。而每个特征之间我们希望降维之后它们也和原来的数据一样，不同的特征之间保持有大的方差，于是乎就有了第一个标准：不同特征之间的方差尽可能大。

第二个标准的解释其实和第一个标准是类似的，只不过形式不同。上面说过了，我们是希望原数据中不同的特征降维后还是不同，而希望它们不同就等价于说它们之间不相关，而协方差就是用来衡量两个特征之间的相关程度的，当协方差等于0的时候，就说明这两个特征之间是无关的。所以就有了这个标准：不同特征之间的协方差等于0。

好了！现在我们已经有了处理标准了，接下来我们就要把这个标准给抽象化（就是写成数学公式）方便我们计算！

我重新上面的数据贴出来：

年龄，	身高，	体重，	地区，	血压，	心率，	肺活量，	体温，	体脂，	健康水平，	姓名
18	，180	，70,	广东，	100，	80，	5000,	37,	0.1，	健康	小王
38	，175	，80,	北京，	120，	70，	3000，	36.4,	0.3，	亚健康	老丁

写成向量的形式

(18, 180 ,70, 100, 80, 5000, 37, 0.1)
(38 ,175 ,80 ,120 ,70 ,3000 ,36.4 ,0.3)

让我们先写出标准1的公式吧：

$$Var(X) = \frac{1}{m}\sum_i^m (X_i - \mu)^2$$

其中X就是一个特征的数据，用我们上面的例子来说，假设X是身高，则X为(180, 175)，则/mu就是177.5，m等于2，于是就求得Var(X) = xxx，同样的道理可以用来算年龄呀，血压呀，心率呀啥的。

这里有个技巧，就是我们先对X进行处理，就是将其减去它的均值：X = X - /mu，于是乎我们的公式就变成了：

$$Var(X) = \frac{1}{m}\sum_i^m (X_i)^2$$

标准2的公式如下：

$$Cov(X,Y) = \frac{1}{m}\sum_i^m (X_i - \mu_X)(Y_i - \mu_Y)$$

我们用上面的技巧，于是乎我们的公式就变成了：

$$Cov(X,Y) = \frac{1}{m}\sum_i^m (X_i Y_i)$$

现在我们把这两个标准给写成数学的公式了，这样我们就可以用计算机来算了。但是这两个公式是分开的。。（就是说他们是两个公式），这样并不太方便于我们计算，我们要像个办法把他们组合起来，这样优化起来才能”联动”。在这里我介绍一个矩阵，叫做协方差矩阵：

$$\frac{1}{m}XX^T = \begin{pmatrix} \frac{1}{m}\sum_i^m a_i^2 & \frac{1}{m}\sum_i^m a_i b_i \\ \frac{1}{m}\sum_i^m a_i b_i & \frac{1}{m}\sum_i^m b_i^2 \end{pmatrix}$$

可以发现，这个矩阵的正对角线就是我们的标准1，也就是方差，而其他的位置则为协方差。所以我们的目的就是使得协方差的位置为0，然后选取方差最大的值（选取方差最大的值可能有点疑惑，是这样的：我们的特征矩阵X是可以有很多特征的，比如年龄，血压，心率，肺活量等等，而方程右边的a和b就是这些特征中的一个，那么势必有些特征的方差比较大，有些特征的方差比较小。而那些方差比较小的特征我们就觉得它们没有那么好的区分效果，而那些方差大的就觉得它们有很好的区分效果，于是乎就把它们选中）。

于是乎我们的目标就变成了，通过优化方法，使得协方差矩阵对角化（就是非对角线的位置值为0）。接下来是很简单的数学推导了。

假设我们最终的协方差矩阵（就是上面说的对角化后的矩阵）为D，X为我们的特征矩阵，C为我们特征矩阵X的协方差矩阵，我们要找到一个矩阵P，使得我们的X特征矩阵可以变成D矩阵。

$$\begin{aligned} D &= \frac{1}{m}(PX)(PX)^T \\ &= \frac{1}{m}PXX^TP^T \\ &= P(\frac{1}{m}XX^T)P^T \\ &= PCP^T \end{aligned}$$

也就是说，我们现在的目标就变成了找到一个矩阵P，使得矩阵以上等式成立。这里需要一丢丢线性代数的知识，主要是关于实对称矩阵的知识，但是这里就不说啦！

最后我们就可以得到矩阵P，这个矩阵P是由我们的特征X矩阵找到的，你也可以理解为它蕴含着我们X矩阵的信息，而这些信息的重要性是越往上的越重要，比如：

$$P = \begin{pmatrix} 0.2 & 0.3 \\ 0.4 & 0.2 \end{pmatrix}$$

则第一行中的(0.2 0.3)的重要性要高于第二行的(0.4 0.2)，然后我们想将我们的数据降到一维度，则：

$$newX = \begin{pmatrix} 0.2 & 0.3 \end{pmatrix} X$$

其中X是原始特征，newX是降维后的特征，而(0.2 0.3)就是我们P矩阵的第一列。从之前的知识可以知道，我们是将X矩阵降维到一维。

PCA的实现

```
1 import numpy as np
2 import pandas as pd
```

```
import matplotlib.pyplot as plt
```

定义一个均值函数。

```
#计算均值,要求输入数据为numpy的矩阵格式，行表示样本数，列表示特征
def meanX(dataX):
    return np.mean(dataX,axis=0)#axis=0表示依照列来求均值。假设输入li
```

开始实现pca的函数：

```
def pca(XMat, k):
    """
    XMat：传入的是一个numpy的矩阵格式，行表示样本数，列表示特征
    k：表示取前k个特征值相应的特征向量
    finalData：指的是返回的低维矩阵
    reconData：相应的是移动坐标轴后的矩阵
    """
    average = meanX(XMat)
    m, n = np.shape(XMat)
    data_adjust = []
    avgs = np.tile(average, (m, 1))
    data_adjust = XMat - avgs
    covX = np.cov(data_adjust.T)   #计算协方差矩阵
    featValue, featVec=  np.linalg.eig(covX)  #求解协方差矩阵的特征
    index = np.argsort(-featValue) #依照featValue进行从大到小排序
    finalData = []
    if k > n:
        print("k must lower than feature number")
        return
    else:
        #注意特征向量时列向量。而numpy的二维矩阵(数组)a[m][n]中，a[1]表
        selectVec = np.matrix(featVec.T[index[:k]]) #所以这里须要
        finalData = data_adjust * selectVec.T
        reconData = (finalData * selectVec) + average
    return finalData, reconData
```

到这里整个流程基本就结束了~