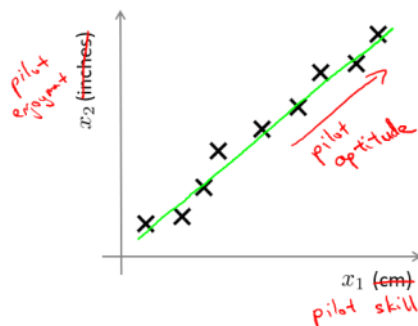


14 降维

2022年11月30日

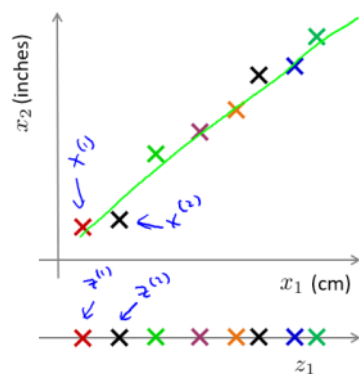
21:12

14.1 目标 I：数据压缩



Reduce data from 2D to 1D

以一个数据集为例，它有两个特征来衡量长度，一个是以厘米为单位的特征，一个是以英寸为单位的特征，那么可以看出它是冗余的，从而可以尝试把特征降低一个维度。



Reduce data from 2D to 1D

$$\begin{aligned} x^{(1)} \in \mathbb{R}^2 &\rightarrow z^{(1)} \in \mathbb{R} \\ x^{(2)} \in \mathbb{R}^2 &\rightarrow z^{(2)} \in \mathbb{R} \\ &\vdots \\ x^{(m)} \in \mathbb{R}^2 &\rightarrow z^{(m)} \in \mathbb{R} \end{aligned}$$

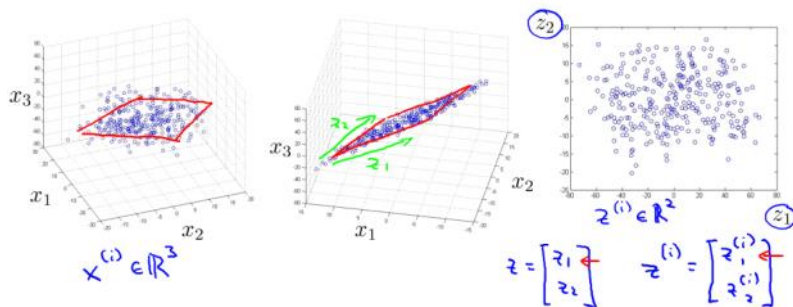
如上图所示通过投影实现特征的映射。

这是将二维降到一维的典型例子，而对于三维降二维：

Data Compression

10000 → 1000

Reduce data from 3D to 2D



三维的数据大致分布于一个平面上，通过将这些数据投影到一个平面实现降维。

14.2 目标 II：可视化

Data Visualization

$$x \in \mathbb{R}^{50} \quad x^{(i)} \in \mathbb{R}^{50}$$

Country	x_1 GDP (trillions of US\$)	x_2 Per capita GDP (thousands of intl. \$)	x_3 Human Development Index	x_4 Life expectancy	x_5 Poverty Index (Gini as percentage)	Mean household income (thousands of US\$)	...
>Canada	1.577	39.17	0.908	80.7	32.6	67.293	...
China	5.878	7.54	0.687	73	46.9	10.22	...
India	1.632	3.41	0.547	64.7	36.8	0.735	...
Russia	1.48	19.84	0.755	65.5	39.9	0.72	...
Singapore	0.223	56.69	0.866	80	42.5	67.1	...
USA	14.527	46.86	0.91	78.3	40.8	84.3	...
...

对于一个非常大的数据集，比如一个国家有50个特征，而且有很多国家，如何可视化？

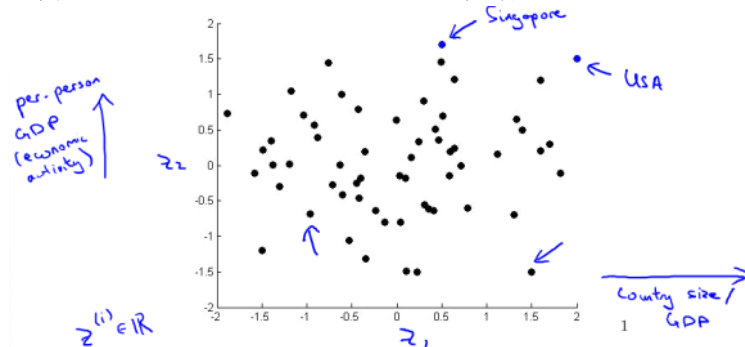
一种办法就是将数据从50维降到2维。

$$z^{(i)} \in \mathbb{R}^2$$

Country	z_1	z_2
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
...

Reduced data from 50D to 2D

降维后的特征似乎并不具有物理意义。对其进行数据绘制：

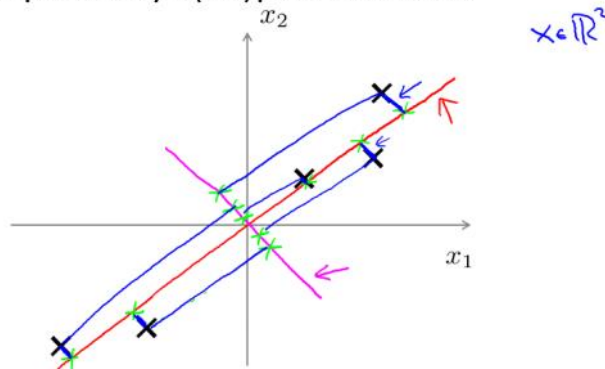


比如此时横轴可能代表国家规模/GDP，而纵轴表示个人GDP。

14.3主成分分析问题规划1

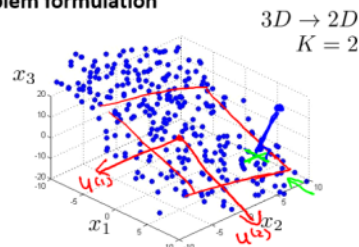
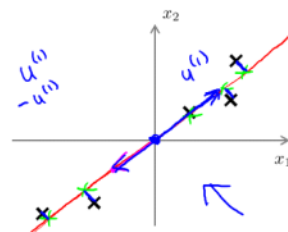
如图所示为一个二维的数据集，

Principal Component Analysis (PCA) problem formulation



为了将其降到一维，找到一条直线，使数据点投影到这条线上，且数据点到投影点的距离最小。而主成分分析(PCA)算法就是找到这样一条可以投影的直线或平面。而如上图的品红色直线，投影的效果非常糟糕，投影误差会很大。

Principal Component Analysis (PCA) problem formulation



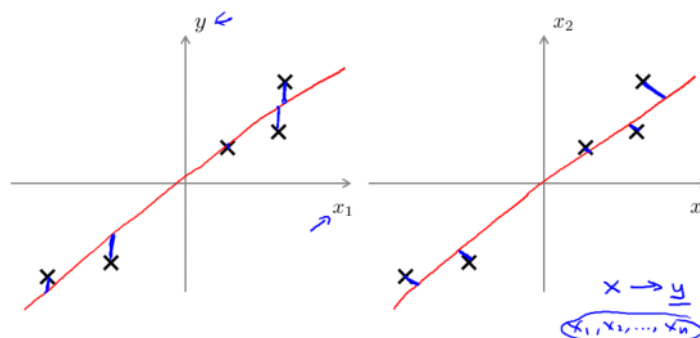
Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \dots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

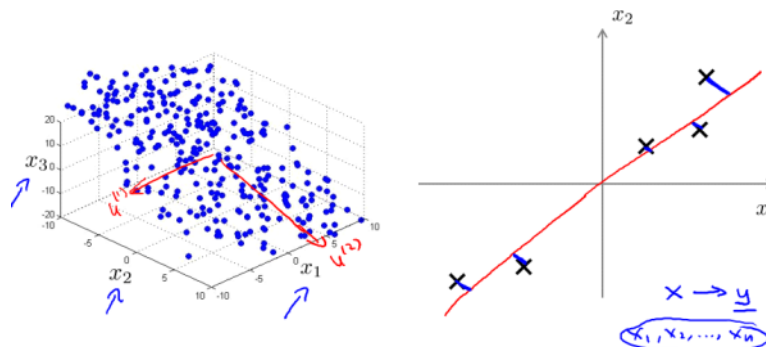
PCA算法会找到一个使投影误差最小的方向，而方向的正负并没有太大影响，因为两个方向都定义了相同的投影直线。而对于更高维的降维处理，也是类似的思想。而用线性代数的思想来描述，就是用一组向量，将数据投影到这个k个向量展开的线性子空间上。

感觉PCA很像线性回归，那么二者有什么关系？

PCA is not linear regression



首先，PCA不是线性回归。二者表示的变量不同，PCA处理的不包括标签，全是特征变量。线性回归的过程是找到数据点到直线的垂直距离的平方和的最小值，而PCA则是找到数据点到直线的投影距离的平方和的最小值。



对于高维度也是同理。PCA没有特殊的变量y需要预测。

14. 4主成分分析问题规划2

Training set: $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ ←

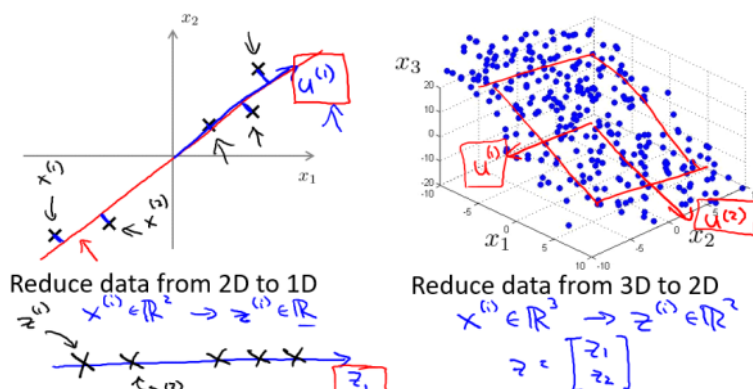
Preprocessing (feature scaling/mean normalization):

$\mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$
Replace each $x_j^{(i)}$ with $x_j^{(i)} - \mu_j$.
If different features on different scales (e.g., x_1 = size of house, x_2 = number of bedrooms), scale features to have comparable range of values.

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

在运行PCA算法之前，首先要做的就是数据预处理。对于数据的集合，进行均值标准化

或者特征缩放。



而后运行PCA算法进行降维处理。

对于具体的实现算法：

Reduce data from n -dimensions to k -dimensions
 Compute "covariance matrix":

$$\Sigma = \frac{1}{m} \sum_{i=1}^m (x^{(i)})(x^{(i)})^T$$
 Compute "eigenvectors" of matrix Σ :

$$[U, S, V] = \text{svd}(\text{Sigma});$$

$$U = \begin{bmatrix} | & | & | & \dots & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \dots & u^{(n)} \\ | & | & | & \dots & | \end{bmatrix}$$

首先计算协方差矩阵。具体地，对于Matlab语言，svd()函数表示奇异值分解，eig()函数也能实现相同的功能，但是svd()函数更加稳定。最终计算的结果是为了得到矩阵U。

From $[U, S, V] = \text{svd}(\text{Sigma})$, we get:

$$U = \begin{bmatrix} | & | & | & \dots & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & | & \dots & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

$$x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^k$$

$$z^{(i)} = \begin{bmatrix} u^{(1)} & u^{(2)} & \dots & u^{(k)} \end{bmatrix}^T x^{(i)} = \begin{bmatrix} -(u^{(1)})^T \\ \vdots \\ -(u^{(k)})^T \end{bmatrix} x^{(i)}$$

然后利用U矩阵的前k列获得k个向量，利用这些向量对数据进行降维。具体的代码实现为：

→ After mean normalization (ensure every feature has zero mean) and optionally feature scaling:

$$\text{Sigma} = \frac{1}{m} \sum_{i=1}^m (x^{(i)})(x^{(i)})^T$$

$$X = \begin{bmatrix} -x^{(1)T} \\ \vdots \\ -x^{(n)T} \end{bmatrix}$$

$$\text{Sigma} = (1/m) \times X' \times X;$$

$$[U, S, V] = \text{svd}(\text{Sigma});$$

$$\text{Ureduce} = U(:, 1:k);$$

$$z = \text{Ureduce}' * x;$$

14.5主成分数量选择

Choosing k (number of principal components)

Average squared projection error: $\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{\text{approx}}^{(i)}\|^2$

Total variation in the data: $\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2$

对于k的选择，首先应该明确的是PCA试图最小化平均平方投影误差。数据的平均变化长度，所谓长度就是训练的样本离距离完全为0到底还有多远。选择k的一个常用的经验法则是选择最小的值：

Typically, choose k to be smallest value so that

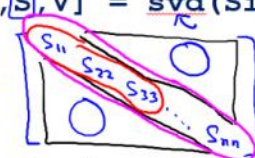
$$\rightarrow \frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2} \leq \frac{0.01}{0.05} \quad \frac{(1\%)}{5\%} \quad \frac{(10\%)}{(10\%)}$$

→ “99% of variance is retained”
95% to 90%

也就是刚刚定义的两式的比率小于0.01。换句话说，平均平方投影误差除以数据的总变化的大小不希望太大。此时也可以说99%的方差被保留。

然后据此进行k的选择。

Algorithm:
Try PCA with $k=1$ ~~$k=2$~~ ~~$k=3$~~ ~~$k=4$~~ ...
Compute $U_{reduce}, z^{(1)}, z^{(2)}, \dots, z^{(m)}, x_{approx}^{(1)}, \dots, x_{approx}^{(m)}$
Check if $\frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2} \leq 0.01?$

→ $[U, S, V] = \text{svd}(\text{Sigma})$
→ $S =$ 
For given k
 $1 - \frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}} \leq 0.01$
→ $\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}} \geq 0.99$

$k=17$

从k=1开始运行PCA算法，检查是否保留了99%的方差。如果这样计算是非常低效的。事实上，在具体的程序实现过程中，当我们调用svd函数，所计算的S矩阵为一个平方矩阵，只有对角线有数值，而其他部分的元素值则为0。那么以k=3为例，此时的方差保留的检查计算可以替换为1减去S矩阵的对角线的前k个元素的和除以所有对角线元素的和。这样会提升算法的效率，不用从k=1开始重复计算PCA。

$$[U, S, V] = \text{svd}(\text{Sigma})$$

Pick smallest value of k for which

$$\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^m S_{ii}} \geq 0.99$$

$k=100$

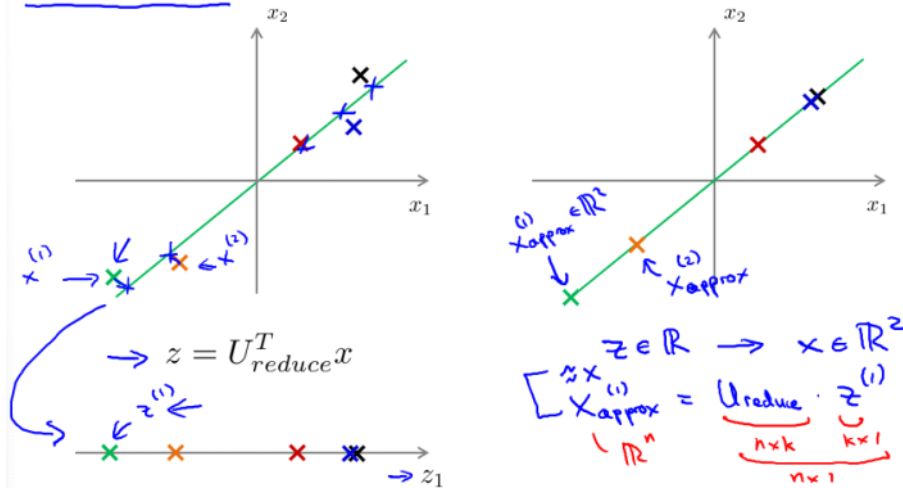
(99% of variance retained)

总结一下，当我们用PCA进行降维处理时，通过计算协方差矩阵，调用svd函数后选择最小的k值。

14.6 压缩重现

对于一组降维的数据，能否把它还原为原来维度的数据？

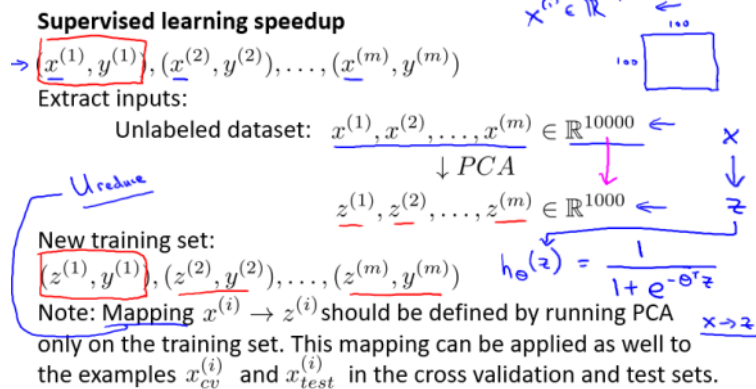
Reconstruction from compressed representation



通过U矩阵可以近似还原到平面中。也将这一过程叫做原始数据重构，通过压缩表示重建原始数据。

15.7应用PCA的建议

大量的特征值会造成算法运行缓慢，通过PCA降低数据维度能够极大地提升算法效率。



对于一个大数据集的监督学习问题，首先检查已经被标记的训练集，并抽取输入，从而得到一个无标签的训练集。接着使用PCA算法得到这些数据的低维表示，得到一个新的训练集。最终，把这个新的训练集输出到机器学习算法中。

最后要注意的是，PCA所要做的是实现一个从x到z的映射，只能通过在训练集上运行PCA来确定。具体而言，就是计算一系列参数进行特征缩放和均值归一化。这种映射也可以被应用到交叉验证集和测试集中。

Application of PCA

- Compression

- Reduce memory/disk needed to store data
- Speed up learning algorithm ←

Choose k by % of variance retain

- Visualization

k=2 or k=3

对于PCA的应用，首先它可以被用于数据压缩，从而减少数据的存储空间和学习算法的运算效率。同时，也可以被用于可视化二维数据或者三维数据。

然而，一种常见的错误使用PCA的应用是尝试通过它防止过拟合。

Bad use of PCA: To prevent overfitting

- Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n$.
Thus, fewer features, less likely to overfit.

Bad!

This might work OK, but isn't a good way to address overfitting. Use regularization instead.

$$\rightarrow \min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2 \leftarrow$$

其背后的一种原因是由于使用了PCA对数据进行压缩使数据的特征数量从原来的n减少到了k。根据前面课程的讨论我们知道，特征量越少，越难以发生过拟合。因此很多人希望通过这种方式来预防过拟合。但是这样做是不合适的，因为它的效果不好。

最好的防止过拟合的办法还是通过增加正则项，而不是减少数据的维度。PCA只是使用低维数据来近似我们的高维数据，它会在压缩的过程中舍掉一些信息，正则化的效果通常会更好，因为不会丢失一些有价值的信息。

另一个对于PCA的错误的应用是在起始设计一个机器学习系统时就将PCA包含在其中。

PCA is sometimes used where it shouldn't be

Design of ML system:

- - Get training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$
 - - ~~Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$~~
 - - Train logistic regression on $\{(z^{(1)}, y^{(1)}), \dots, (z^{(m)}, y^{(m)})\}$
 - - Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$. Run $h_{\theta}(z)$ on $\{(z_{test}^{(1)}, y_{test}^{(1)}), \dots, (z_{test}^{(m)}, y_{test}^{(m)})\}$
- How about doing the whole thing without using PCA?
- Before implementing PCA, first try running whatever you want to do with the original/raw data $x^{(i)}$. Only if that doesn't do what you want, then implement PCA and consider using $z^{(i)}$.

对于一个机器学习系统的初始设计，应该首先要问的是如果不使用PCA会怎么样？所以老师的建议是在做一个项目的时候，先不要考虑使用PCA，而是用原始的数据样本进行操作。当这样所实现的效果没有达到我们的目的或预期时再考虑使用PCA。只有我们有强有力的证据证明原始的数据无法很好地工作时，我们再考虑使用PCA算法对数据进行压缩。