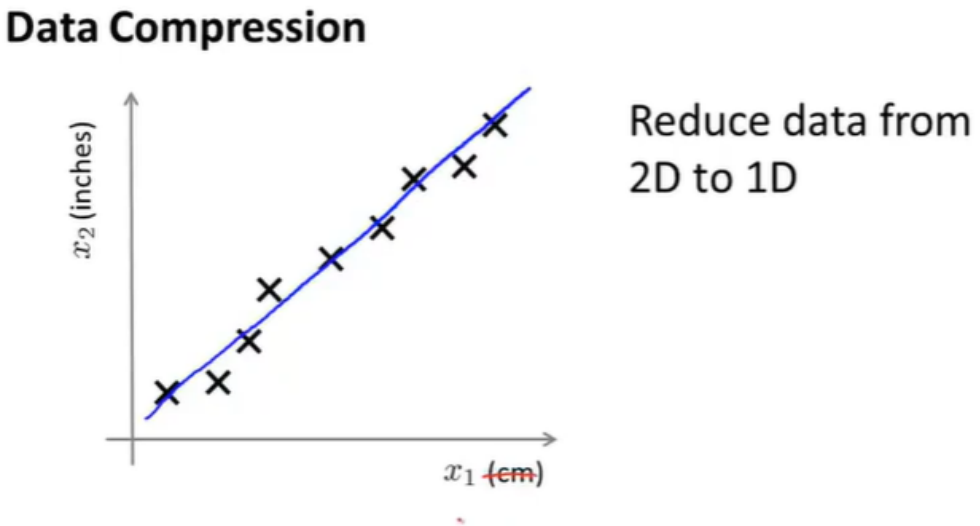
降维

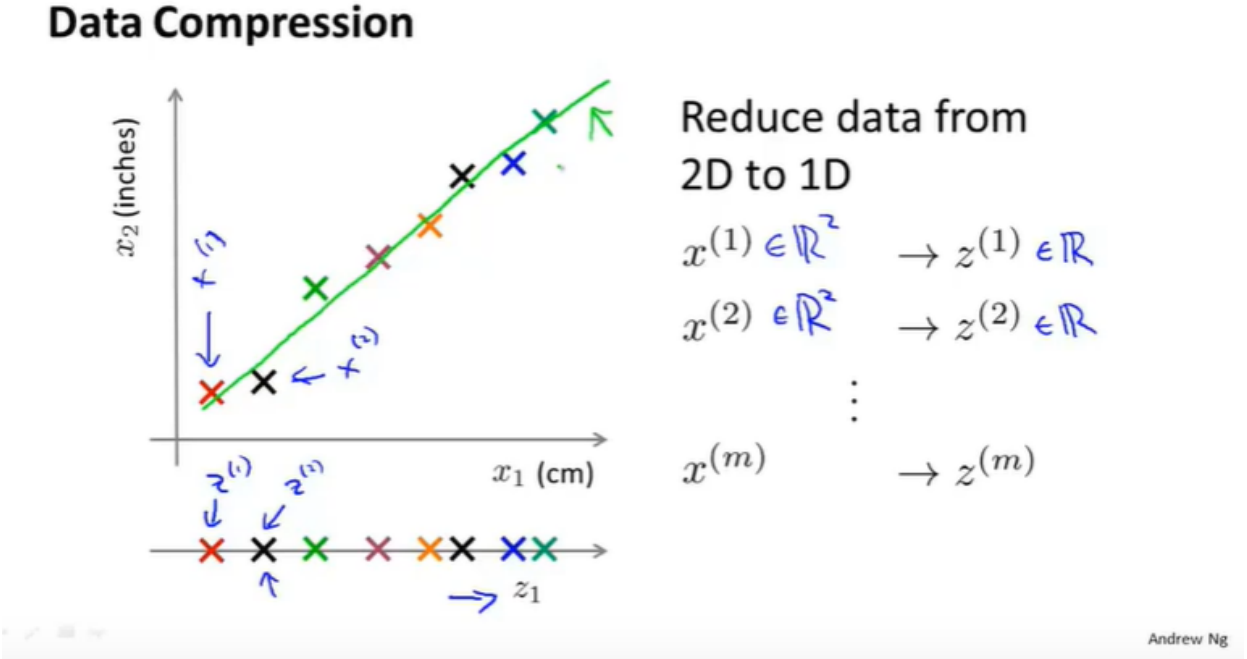
应用1.数据压缩

比如一个特征是长度(cm),另一个是长度（inches），这样特征就高度冗余，两个特征可以降维为一个。

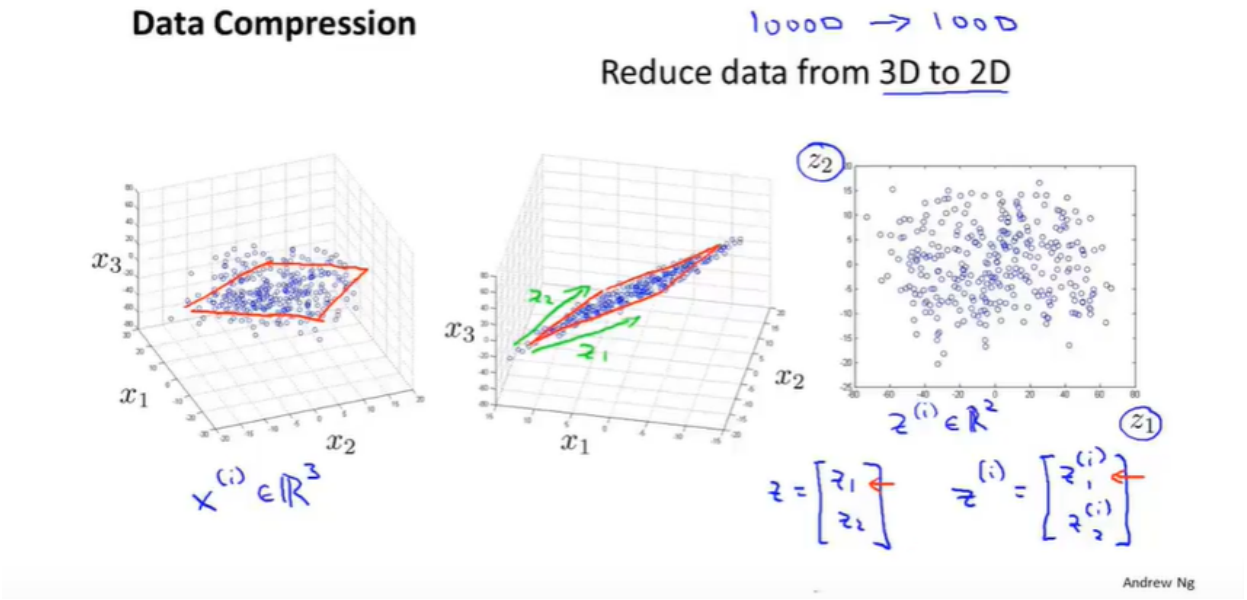
或者一个特征表示飞行员的技能，另一个是他享受飞行的程度，这两个特征也是高度相关的，这种情况也需要降维。



我们将坐标投影到z1这条线上，本来用二维向量表示的一个数据点，现在用一个实数就可以表示，可以将内存需求减半，另外这也可以让学习算法运行地更快。



三维降维到两维：将三维空间内的数据点投影到一个二维平面上。这样只需要z1，z2两个点就可以确认一个点的位置。



应用2.可视化数据

比如有50个指标数据来衡量一个国家，我们不想用50个数字去代表一个国家。

Data Visualization $x \in \mathbb{R}^{50}$

Country	x_1 GDP (trillions of US$)	x_2 Per capita GDP (thousands of intl. $)	x_3 Human Develop-ment Index	x_4 Life expectancy	x_5 Poverty Index (Gini as percentage)	x_6 Mean household income (thousands of US$)	...
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	...
...	...	...	...	...	...	...	

[resources from en.wikipedia.org]

Andrew Ng

我们可以尝试用一个二维的向量z来表示一个国家。相当于将50D降维到2D。可以在一个二维图中表示每个国家。

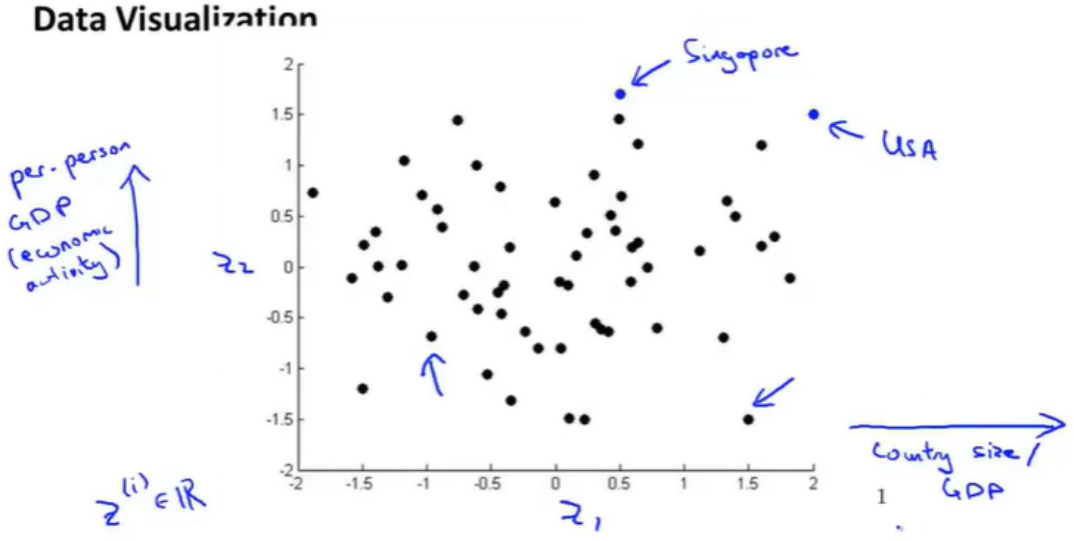
Data Visualization

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

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
...	...	...

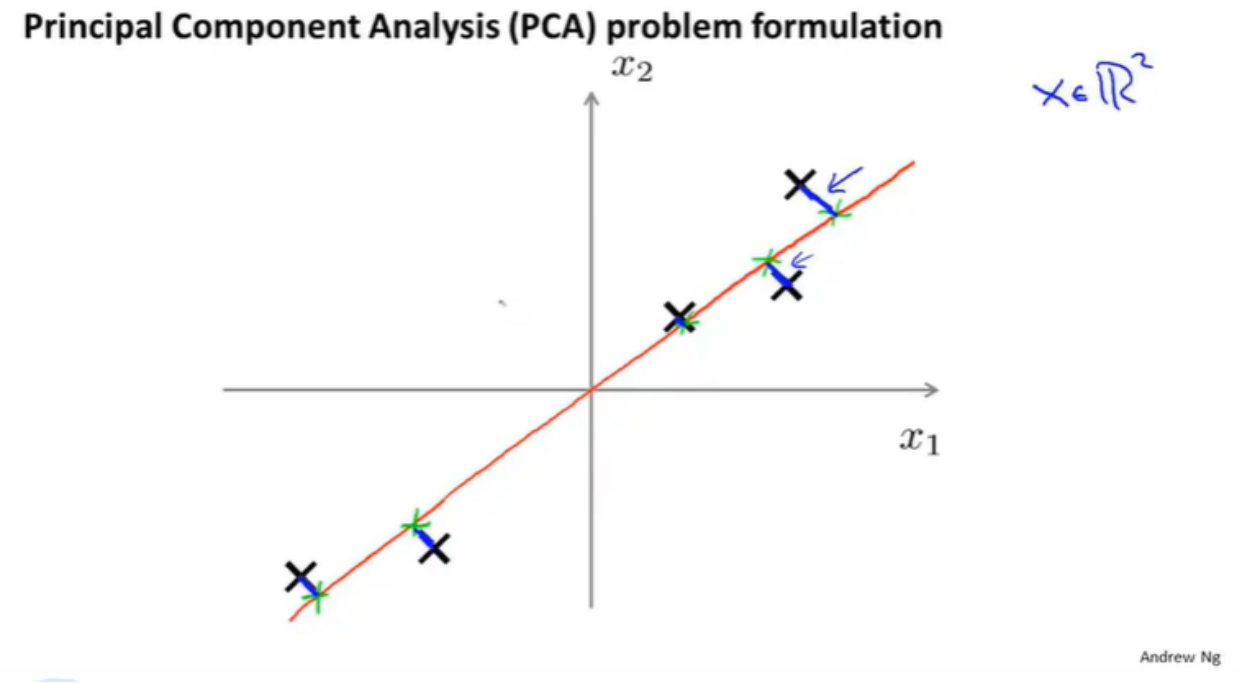
Andrew

50个特征可以用国家面积和GDP来表示，并很好的理解这俩这两个数据。

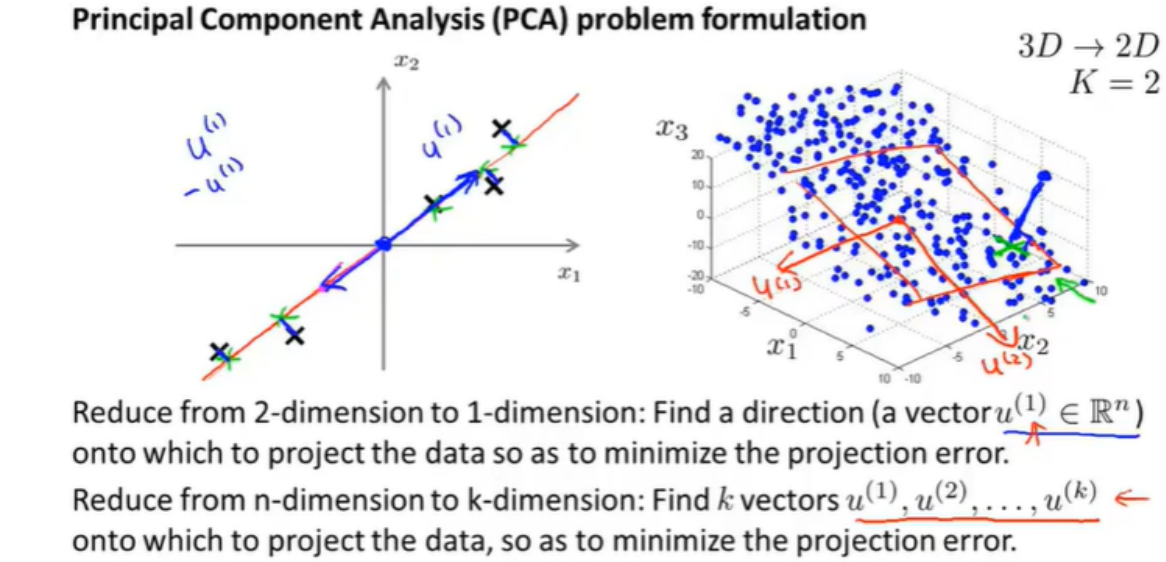


PCA(主成分分析方法)降维（1）

在二维降维到一维过程中，需要找到一条直线，将各个点投影到这条直线上，投影的蓝色线段长度称为投影误差。PCA需要找到一条投影误差最小的线。

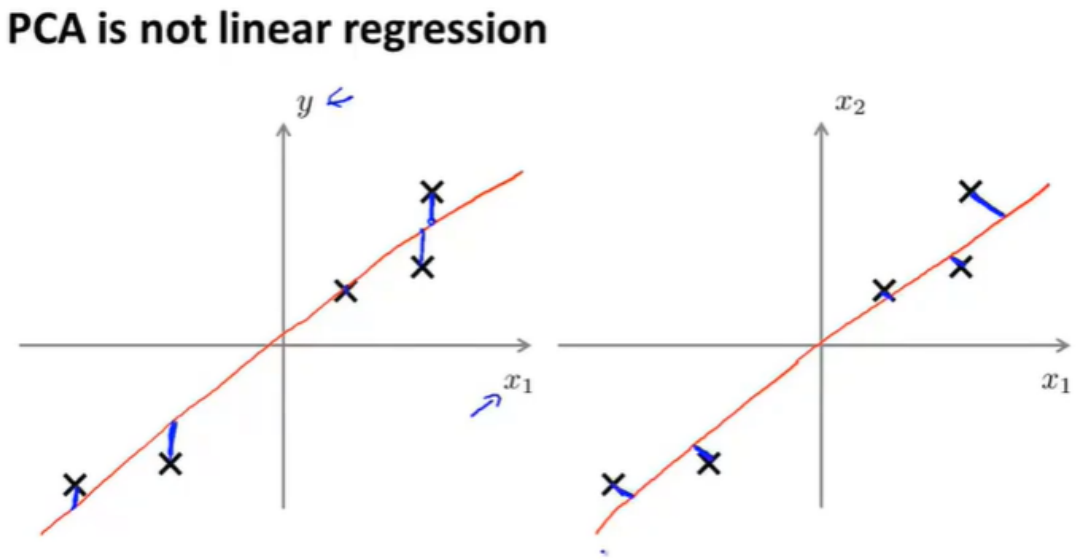


PCA就是找出一条直线，一个平面，或者其他维空间，来对数据样本点进行投影。



PCA不是线性回归

左图是线性回归，右图是PCA，线性回归目标是最小化蓝色线段是y值的差值，PCA最小化的是正交距离。



线性回归试图预测变量y，PCA中我们同等对待一系列数据集中的x

PCA(主成分分析方法)降维（2）

数据预处理：

均值标准化：用每个值减去均值得到的数来代替每个数，这样使得每个数据样本的均值为0

特征缩放：（数据样本特征值-平均值）/ 标准偏差

Data preprocessing

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

Preprocessing (feature scaling/mean normalization):

$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$

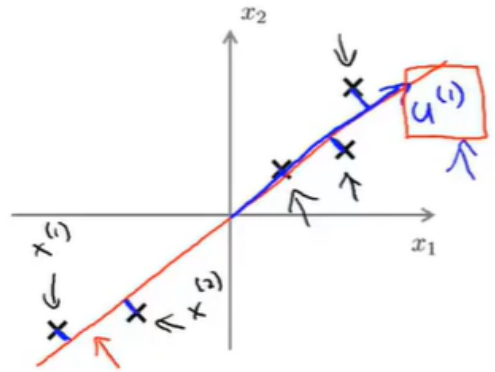
Replace each $x_j^{(i)}$ with $x_j - \mu_j$.

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.

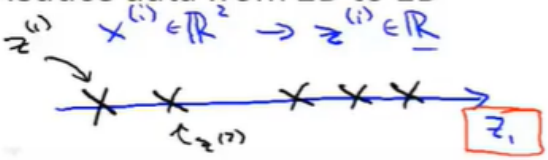
$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$

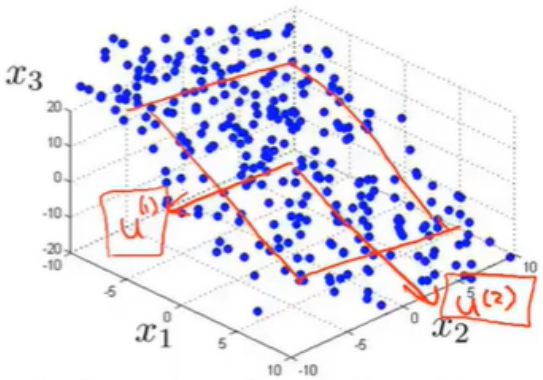
降维：

Principal Component Analysis (PCA) algorithm



Reduce data from 2D to 1D





Reduce data from 3D to 2D

$x^{(i)} \in \mathbb{R}^3 \rightarrow z^{(i)} \in \mathbb{R}^2$

$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$

Andrew Ng

PCA分析：

协方差矩阵： sigma符号，用来表示一个矩阵（看起来很像求和符号）

svd 奇异值分解

Principal Component Analysis (PCA) algorithm

Reduce data from n-dimensions to k-dimensions

Compute "covariance matrix":

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

Sigma

$n \times 1$ $1 \times n$ $n \times n$

Compute "eigenvectors" of matrix Σ: → Singular value decomposition

```
[U,S,V] = svd(Sigma);
```

eig (Sigma)

n×n matrix.

$$U = \begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \cdots & u^{(n)} \\ | & | & | & & | \end{bmatrix} \quad U \in \mathbb{R}^{n \times n}$$

$u^{(1)}, \ldots, u^{(k)}$

k

取U矩阵(n*n)的前k列

U矩阵(n*n)的前k列的转置，再乘以n*1维的x(i)向量，即可得到k维的向量z

Principal Component Analysis (PCA) algorithm

From `[U,S,V] = svd(Sigma)`, we get:

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

k

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

$$z^{(i)} = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{bmatrix}^T x^{(i)} \qquad x^{(i)} = \begin{bmatrix} — (u^{(1)})^T — \\ \vdots \\ — (u^{(k)})^T — \end{bmatrix} x^{(i)}$$

$z \in \mathbb{R}^k$

$n \times k$

U_{reduce}

$k \times n$ $n \times 1$

$k \times 1$

Andrew N

Principal Component Analysis (PCA) algorithm summary

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

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

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

Sigma = (1/m) * X' * X;

```
[U,S,V] = svd(Sigma);
Ureduce = U(:,1:k);
z = Ureduce'*x;
```

$x \in \mathbb{R}^n$ ~~$x_0 = 1$~~

PCA最终得到了最小平方差投影误差