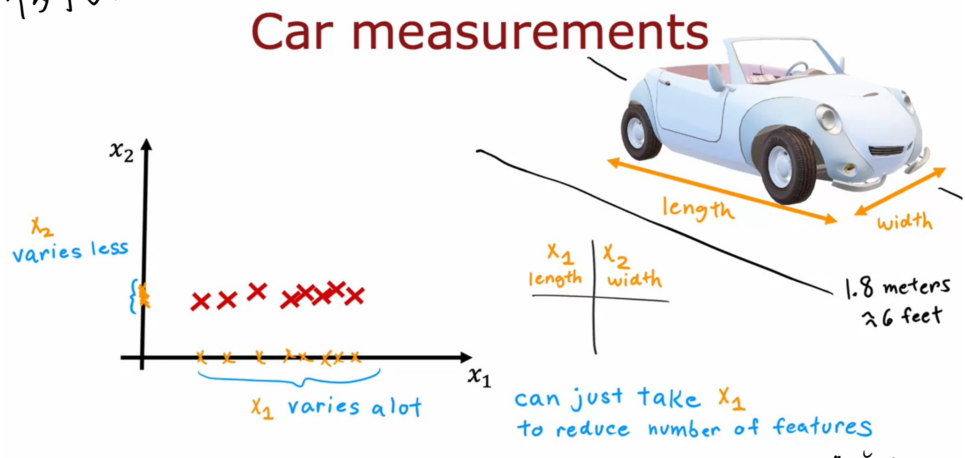
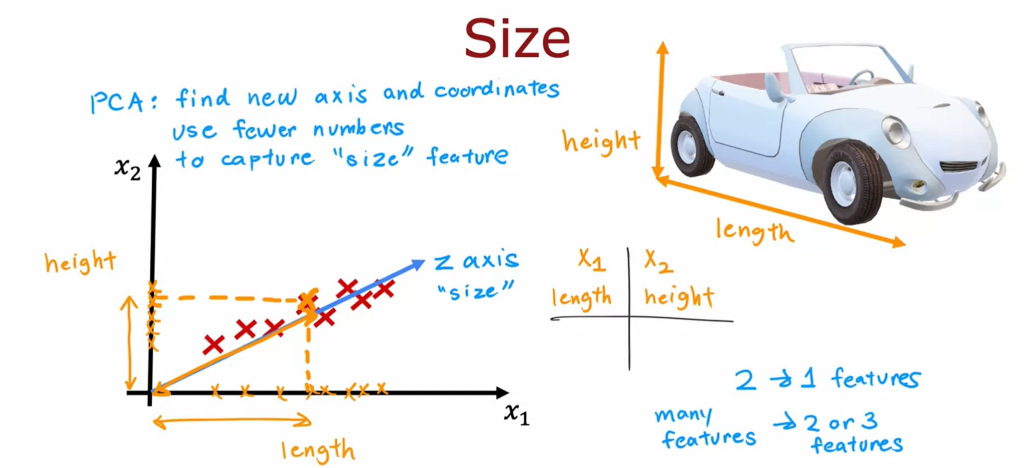
Reducing the number of feature.

例子1:



X_2 没变 就可以不用考虑它.
X_1 变化大，要考虑.

例子2:



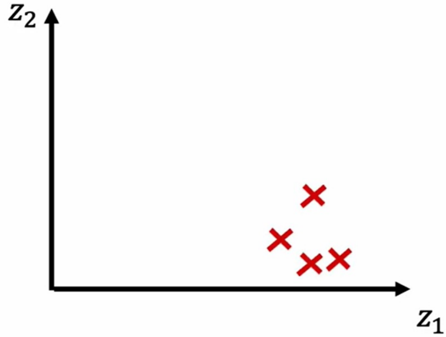
size 一个量 能代表 2个 feature.

例子3:

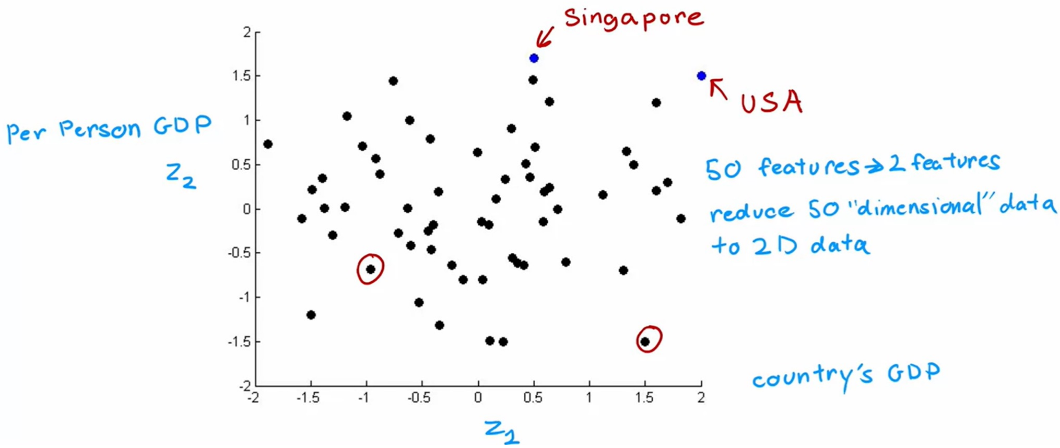
50 features

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

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



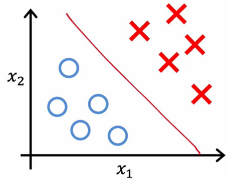
Data visualization



PCA algorithm

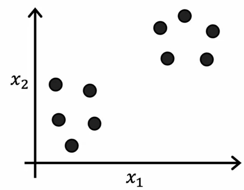
处理..

Supervised learning



Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(m)}, y^{(m)})\}$?

Unsupervised learning



Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

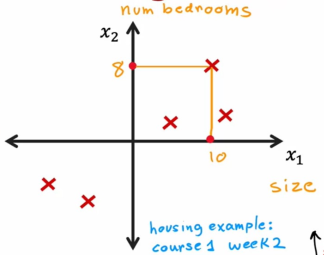
没有 target label y.

PCA algorithm

coordinates

$x_1 = 10$ $x_2 = 8$

Can we choose a different axis?



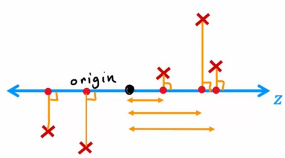
Preprocess features

Normalized to have zero mean

feature scaling

Choose an axis

"project" examples onto the axis

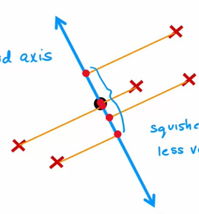


variance is large

capturing info of original data

Choose an axis

not a good axis



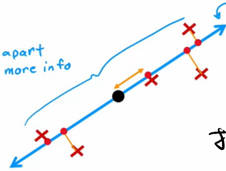
squished together

less variance

Choose an axis

far apart

capturing more info

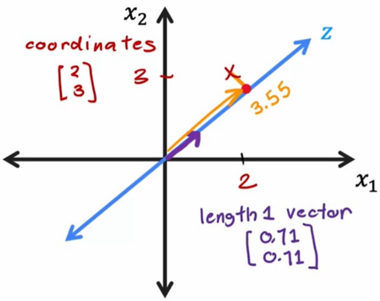


principal component

max variance

投影越疏越好.

Coordinate on the new axis

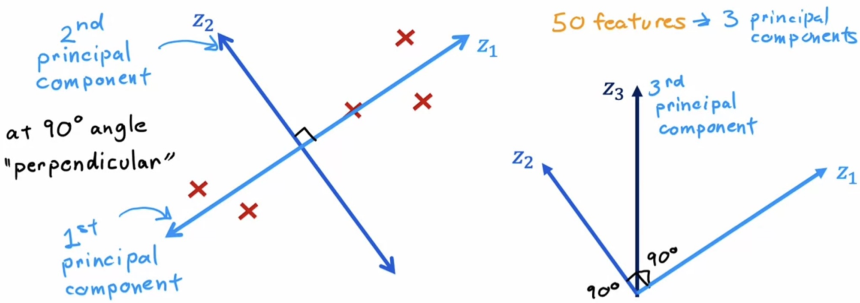


dot product

$$\begin{bmatrix}2\\3\end{bmatrix}\cdot\begin{bmatrix}0.71\\0.71\end{bmatrix}$$

$$2\times 0.71 + 3\times 0.71 = 3.55$$

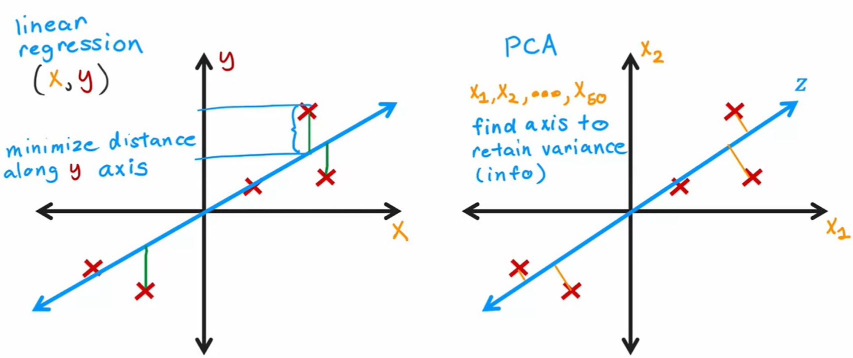
More principal components



如果有3个轴

3个轴相互垂直.

PCA is not linear regression

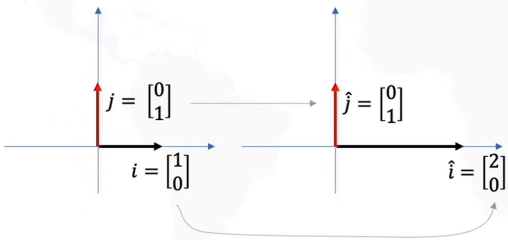


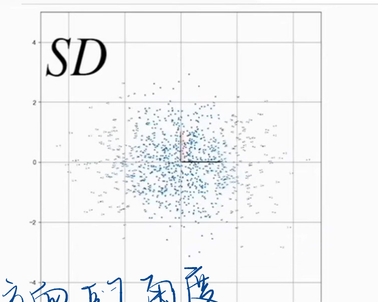
数据线性变换

拉伸 → 拉伸决定了方差最大的方向是横向还是纵向。

$$D = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{bmatrix} \qquad S = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \qquad SD = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{bmatrix} = \begin{bmatrix} 2x_1 & 2x_2 & 2x_3 & 2x_4 \\ y_1 & y_2 & y_3 & y_4 \end{bmatrix}$$

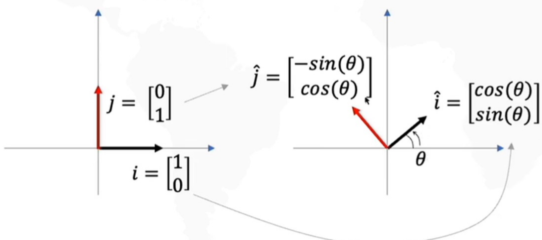
只在对角线上有数字

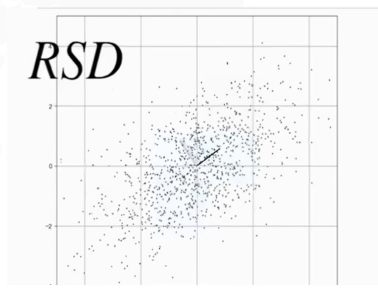




旋转 → 旋转决定了方差最大方向的角度。

$$D = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{bmatrix} \qquad R = \begin{bmatrix} \cos(\theta) & -sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \qquad RD = \begin{bmatrix} \cos(\theta) & -sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{bmatrix}$$



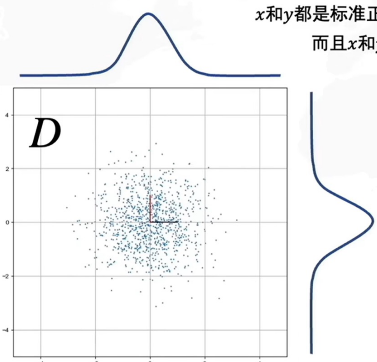


白数据

白数据（white data）

x和y都是标准正态分布

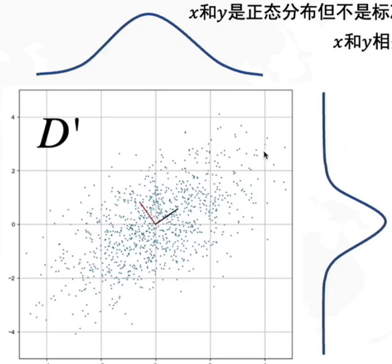
而且x和y不相关



我们手上的数据(去中心化后)

x和y是正态分布但不是标准

x和y相关

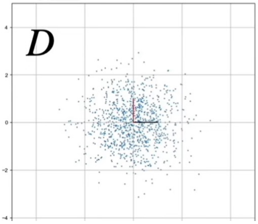


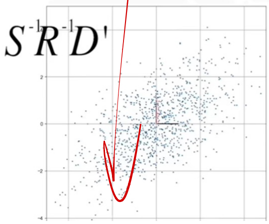
白数据

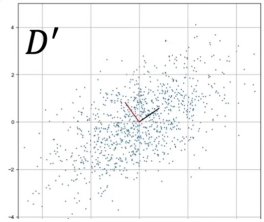
$S = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$　　$R = \begin{bmatrix} \cos(\theta) & -sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$

D	$S^{-1}R^{-1}D'$	$R^{-1}D'$	D'
白数据（white data） ←	拉伸 ←	旋转 ←	我们手上的数据

$D = S^{-1}R^{-1}D'$　　$S^{-1} = \begin{bmatrix} 1/a & 0 \\ 0 & 1/b \end{bmatrix}$　　$R^{-1} = \begin{bmatrix} \cos(-\theta) & -sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} = R^{\mathrm{T}}$







协方差的特征向量就是R

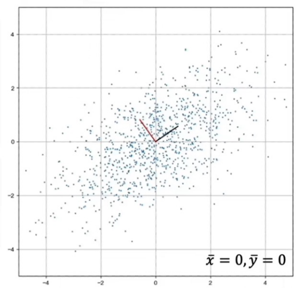
协方差：

$$cov(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

协方差表示的是：

两个变量在变化过程中是同方向变化？还是反方向变化？同向或反向程度如何？

$x\uparrow \rightarrow \quad y\uparrow \rightarrow \quad cov(x,y) > 0$



$cov(x,y) = \frac{\sum_{i=1}^{n} x_i y_i}{n-1}$　　$cov(x,x) = \frac{\sum_{i=1}^{n} {x_i}^2}{n-1}$

方差

协方差矩阵：

$$C = \begin{bmatrix} cov(x,x) & cov(x,y) \\ cov(x,y) & cov(y,y) \end{bmatrix}$$

对称阵
对角线是方差

协方差矩阵：

对称阵
对角线是方差

$$C = \begin{bmatrix} \frac{\sum_{i=1}^{n} {x_i}^2}{n-1} & \frac{\sum_{i=1}^{n} x_i y_i}{n-1} \\ \frac{\sum_{i=1}^{n} x_i y_i}{n-1} & \frac{\sum_{i=1}^{n} {y_i}^2}{n-1} \end{bmatrix}$$

$cov(x,y) = \frac{\sum_{i=1}^{n} x_i y_i}{n-1}$

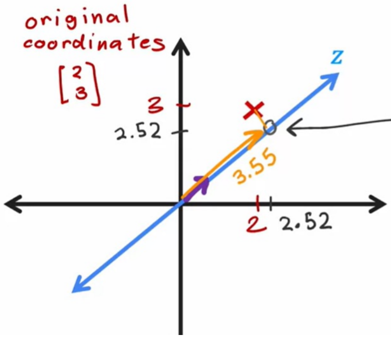
$cov(x,x) = \frac{\sum_{i=1}^{n} {x_i}^2}{n-1}$

$$= \frac{1}{n-1} \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}$$

$= \frac{1}{n-1} DD^{\mathrm{T}}$　　　D —数据

Reconstruction.

Approximation to the original data

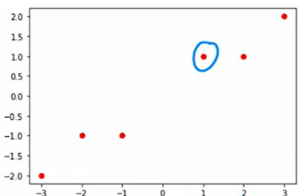


given $z = 3.55$,
find original (x_1, x_2) (approximately)
"reconstruction"

$$3.55 \times \begin{bmatrix} 0.71 \\ 0.71 \end{bmatrix} = \begin{bmatrix} 2.52 \\ 2.52 \end{bmatrix}$$

PCA in code.

1D

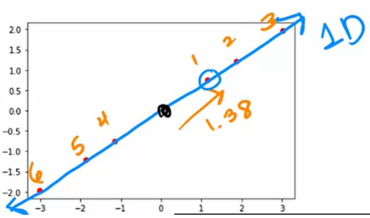


Example

```
X = np.array([[1, 1], [2, 1], [3, 2],
              [-1, -1], [-2, -1], [-3, -2]])
```

2D

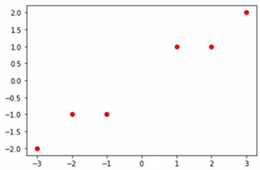
```
pca_1 = PCA(n_components=1)
pca_1.fit(X)
pca_1.explained_variance_ratio_   0.992
X_trans_1 = pca_1.transform(X)
X_reduced_1 = pca.inverse_transform(X_trans_1)
```




```
   array([
1  [ 1.38340578],
2  [ 2.22189802],
3  [ 3.6053038 ],
4  [-1.38340578],
5  [-2.22189802],
6  [-3.6053038 ]])
```

27.

Example



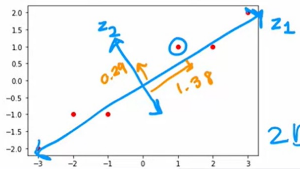
```
X = np.array([[1, 1], [2, 1], [3, 2],
              [-1, -1], [-2, -1], [-3, -2]])
```

2D

6

```
pca_2 = PCA(n_components=2)
pca_2.fit(X)
pca_2.explained_variance_ratio_
X_trans_2 = pca.transform(X)
X_reduced_2 = pca.inverse_transform(X_trans_2)
```

z_1 0.992 z_2 0.008



2D

z_1 z_2

```
array([
[ 1.38340578,  0.2935787 ],
[ 2.22189802, -0.25133484],
[ 3.6053038 ,  0.04224385],
[-1.38340578, -0.2935787 ],
[-2.22189802,  0.25133484],
[-3.6053038 , -0.04224385]])
```