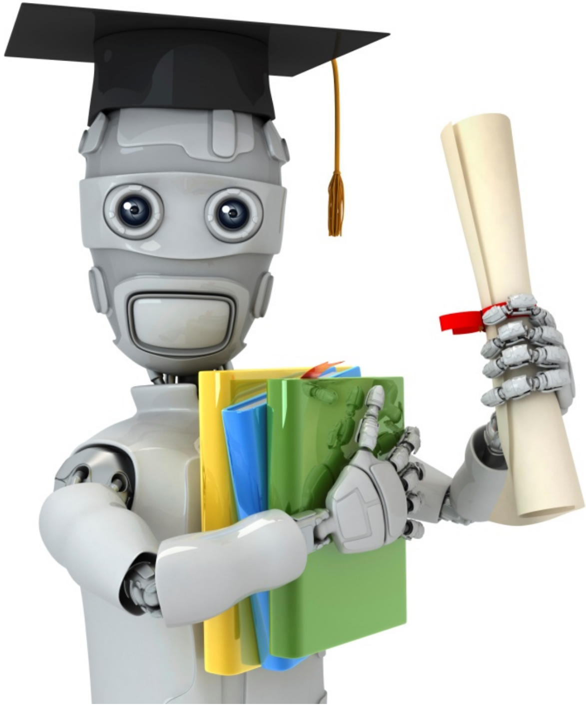
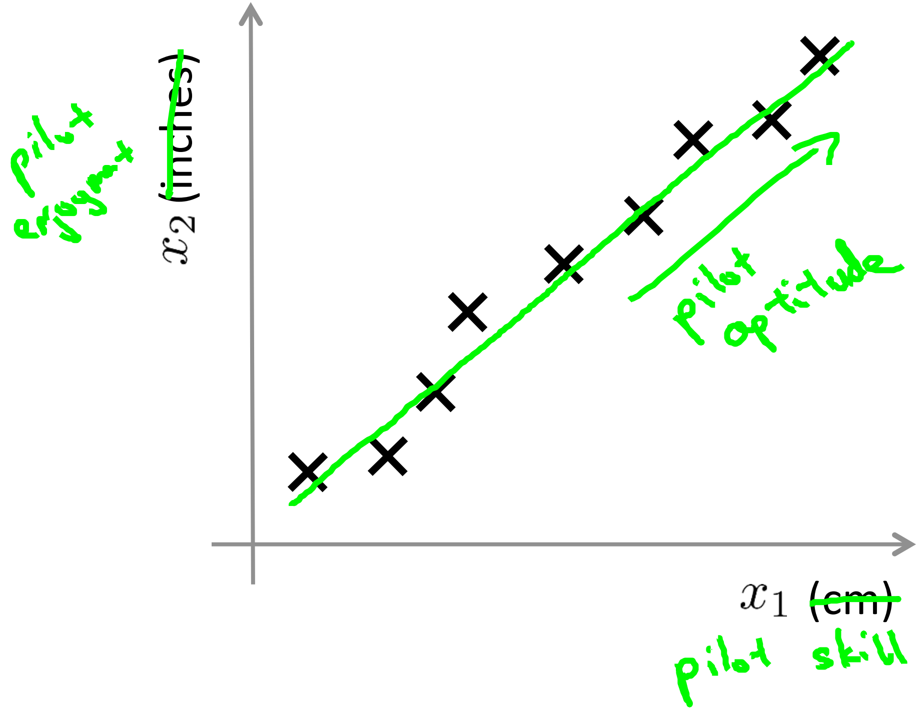


Machine Learning

Dimensionality Reduction

Motivation I: Data Compression

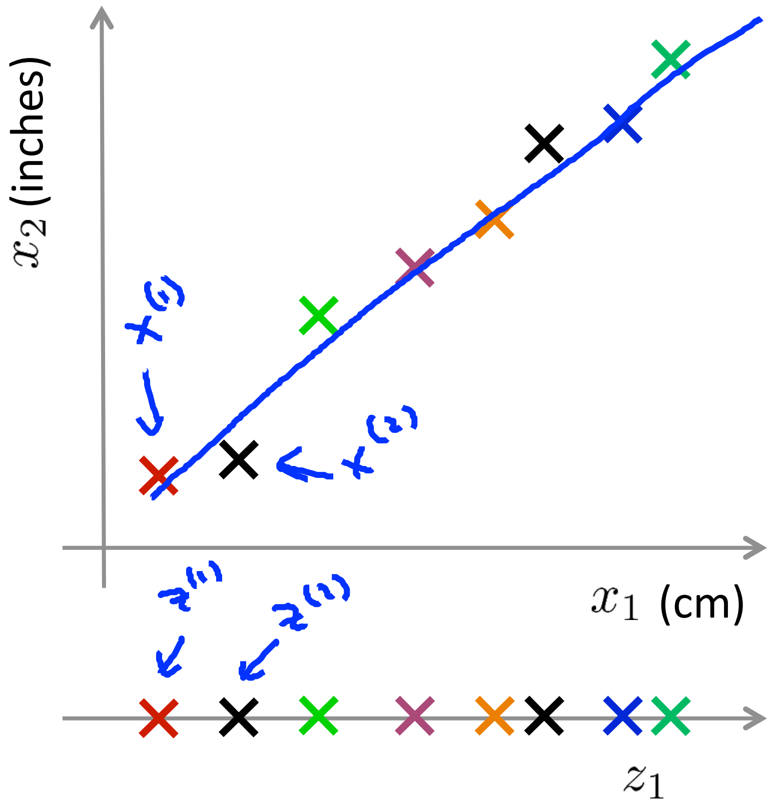
Data Compression



Reduce data from 2D to 1D

Andrew Ng

Data Compression



Reduce data from 2D to 1D

$x^{(1)} \in \mathbb{R}^2 \rightarrow z^{(1)} \in \mathbb{R}$

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

$\vdots$

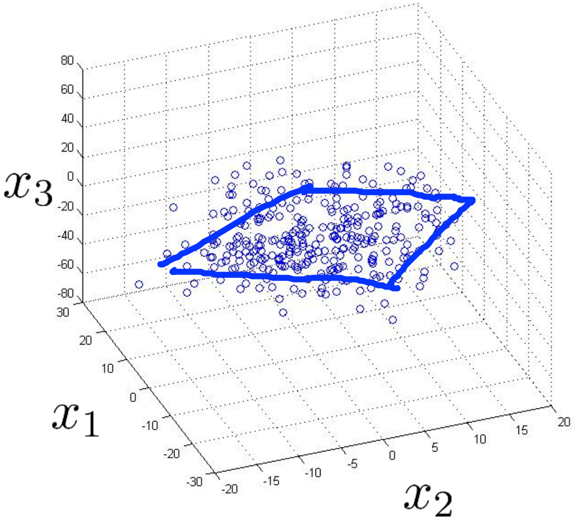
$x^{(m)} \in \mathbb{R}^2 \rightarrow z^{(m)} \in \mathbb{R}$

Andrew Ng

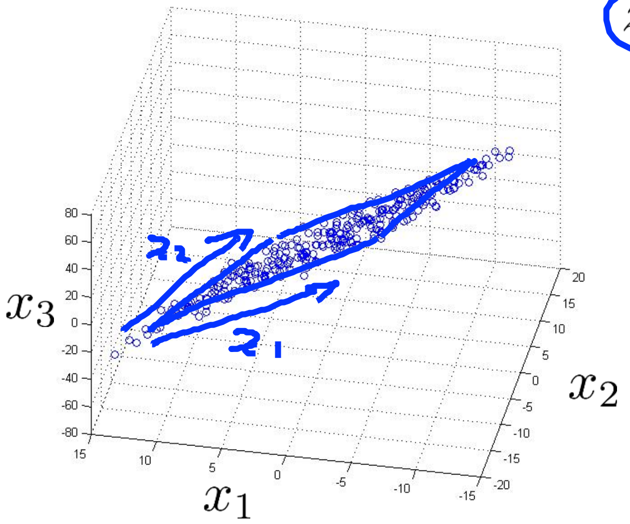
Data Compression

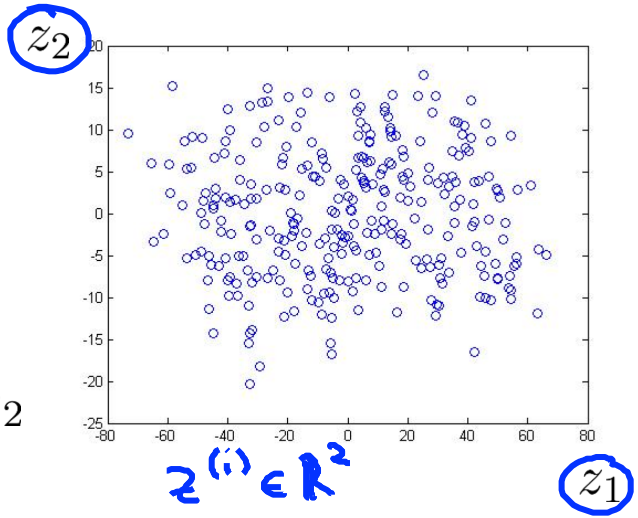
$1000D \rightarrow 100D$

Reduce data from 3D to 2D



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

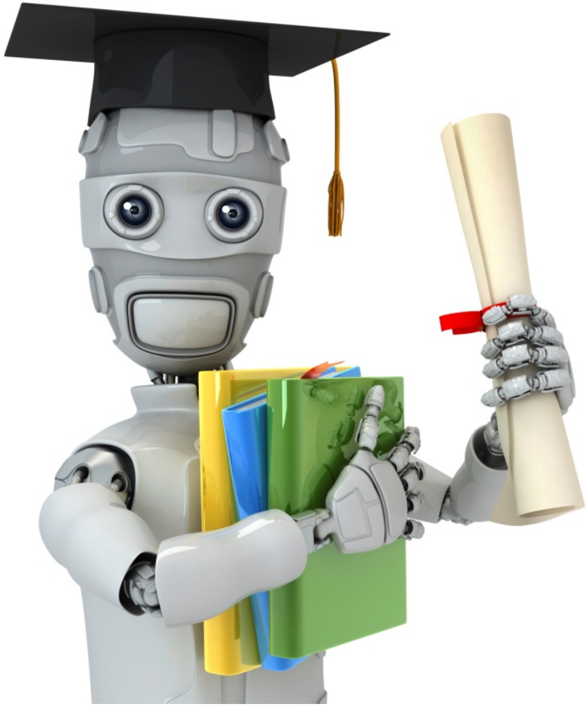




$z^{(i)} \in \mathbb{R}^2$

$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ $\quad z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$

Andrew Ng



Machine Learning

Dimensionality Reduction

Motivation II: Data Visualization

Data Visualization

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

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]

我们可以使用一对数字 z1和z2 从某种程度来说 这两个数总结了50个数

Data Visualization

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

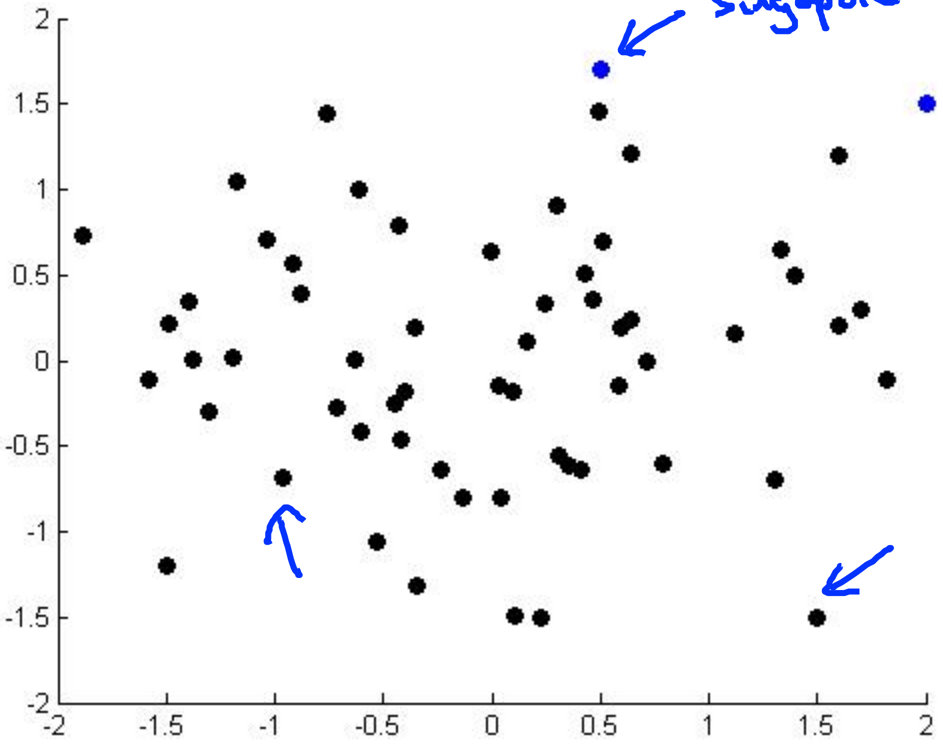
Country	z_1	z_2
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
…	…	…

Reduce data from 50D to 2D

Data Visualization

per. person GDP (economic activity)

z_2

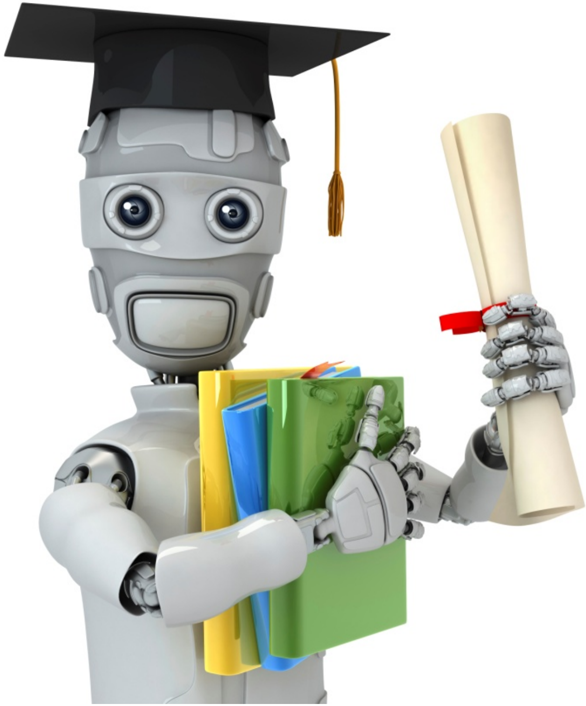


z_1

Country size / GDP

1

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

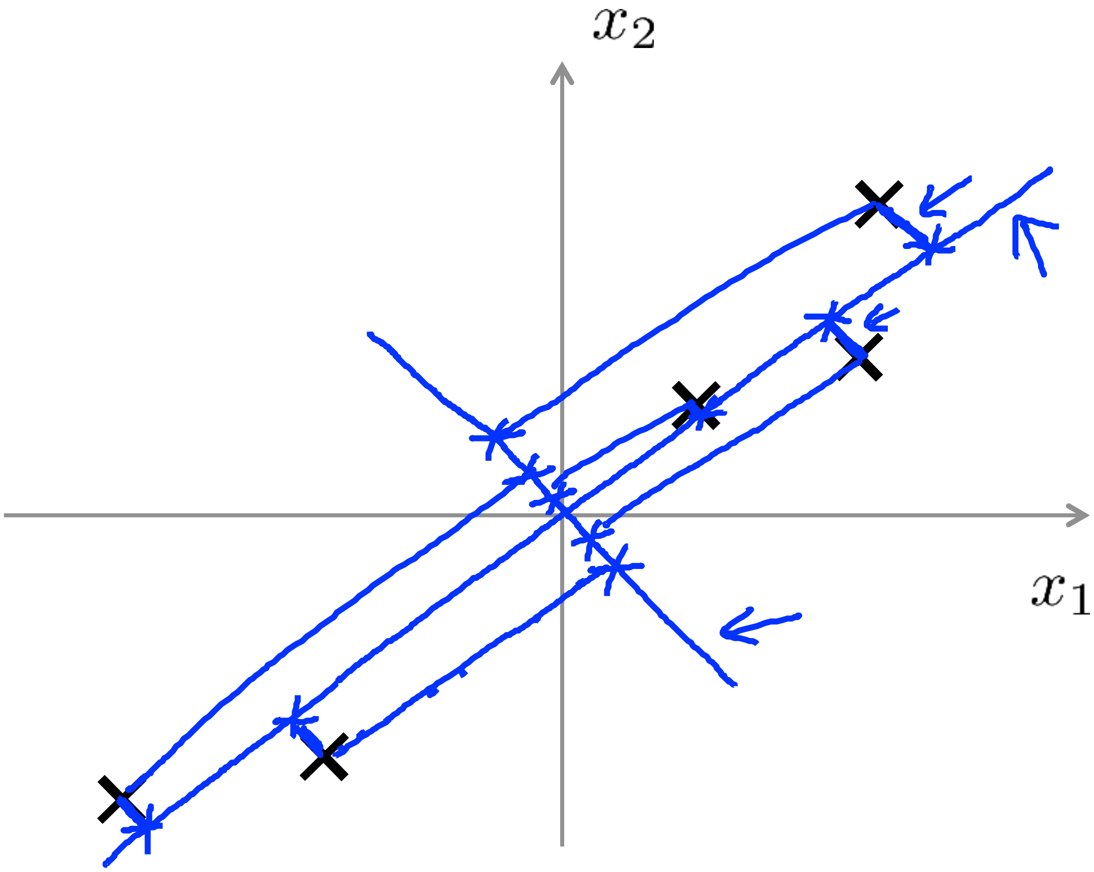


Machine Learning

Dimensionality Reduction

Principal Component Analysis problem formulation

Principal Component Analysis (PCA) problem formulation

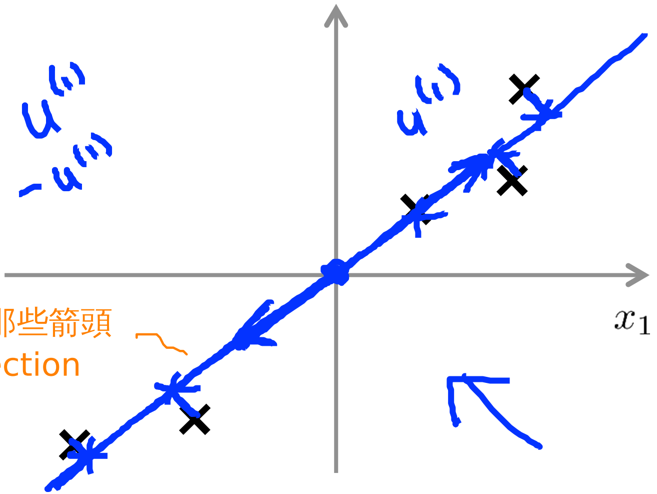


$x \in \mathbb{R}^2$

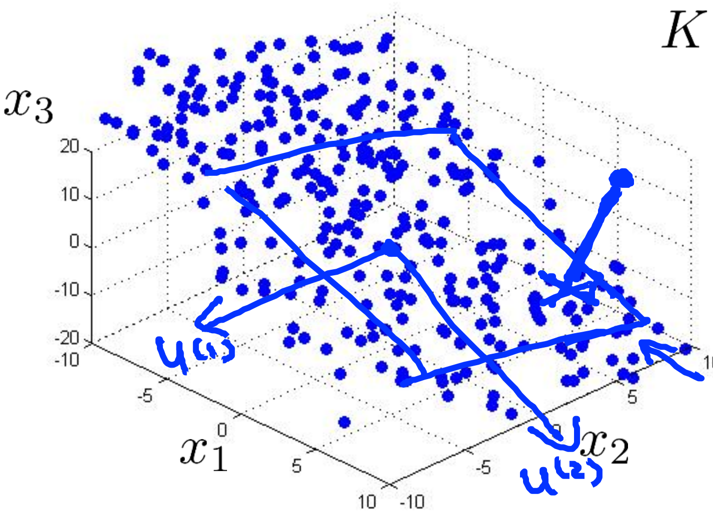
在应用PCA之前 通常的做法是 先进行均值归一化和 特征规范化 使得 特征 x1 和 x2 均值为0 数值在可比较的范围之内(後會詳講)

Principal Component Analysis (PCA) problem formulation

$$3D \rightarrow 2D$$
$$K = 2$$



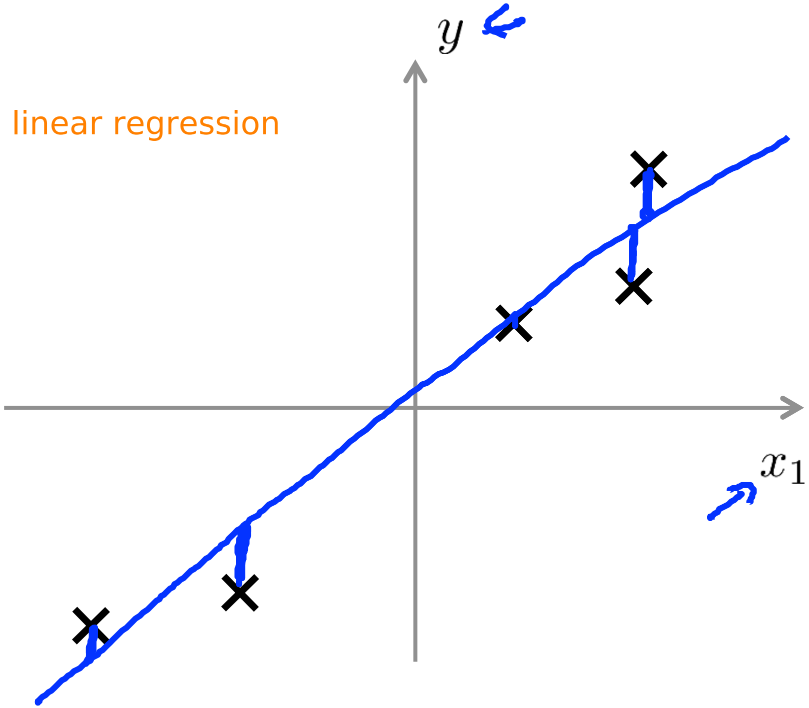
該線上的那些箭頭
就是projection

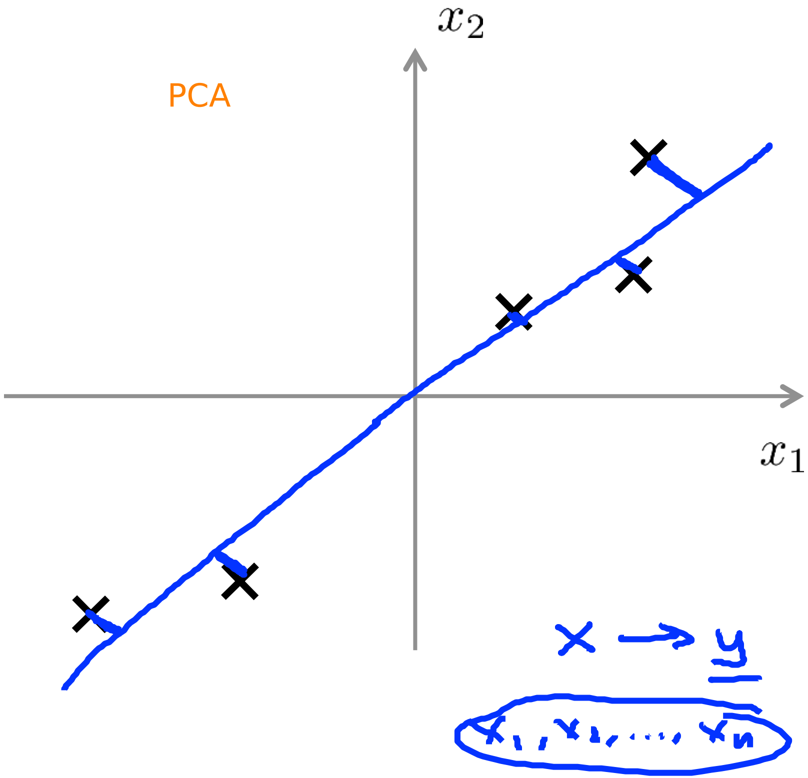


Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

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

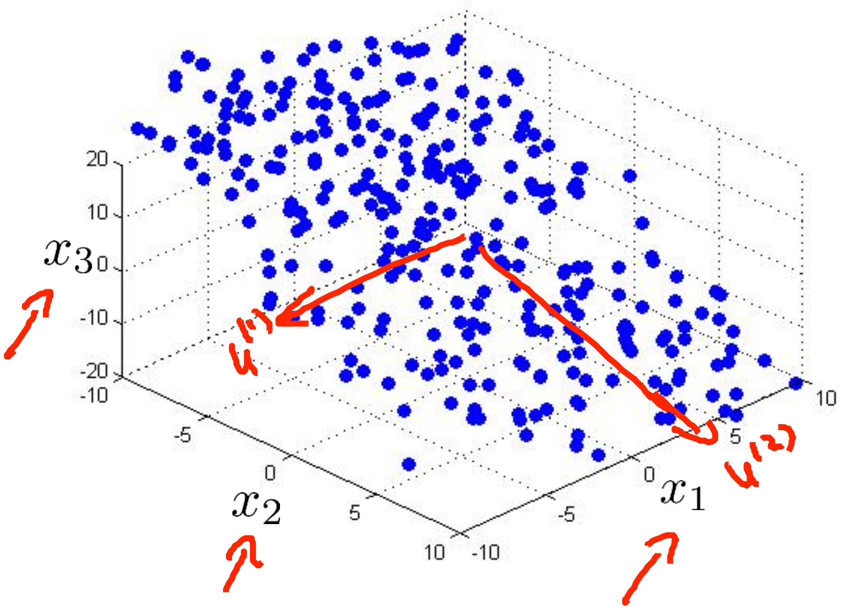
PCA is not linear regression

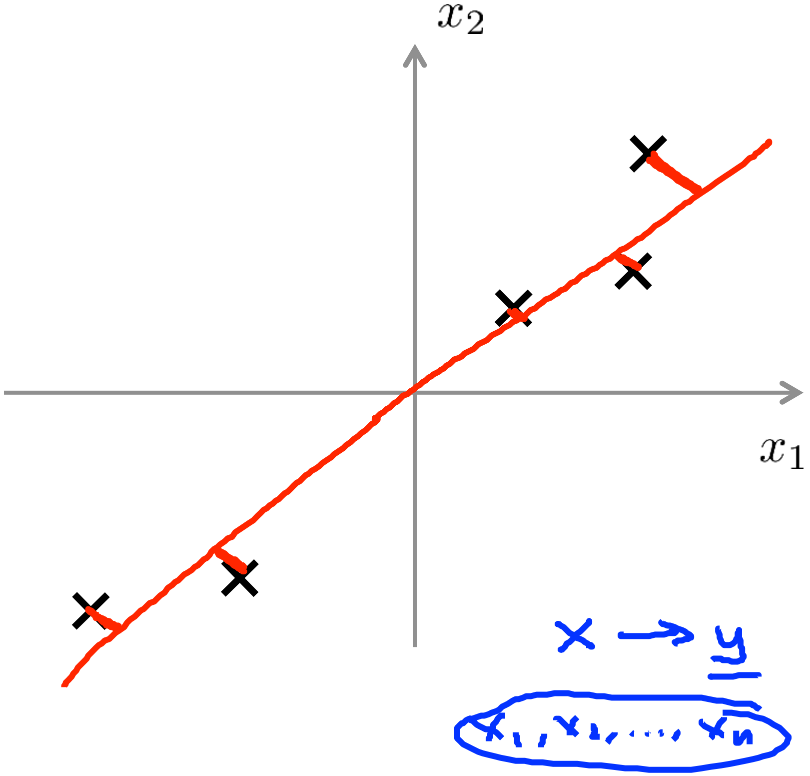




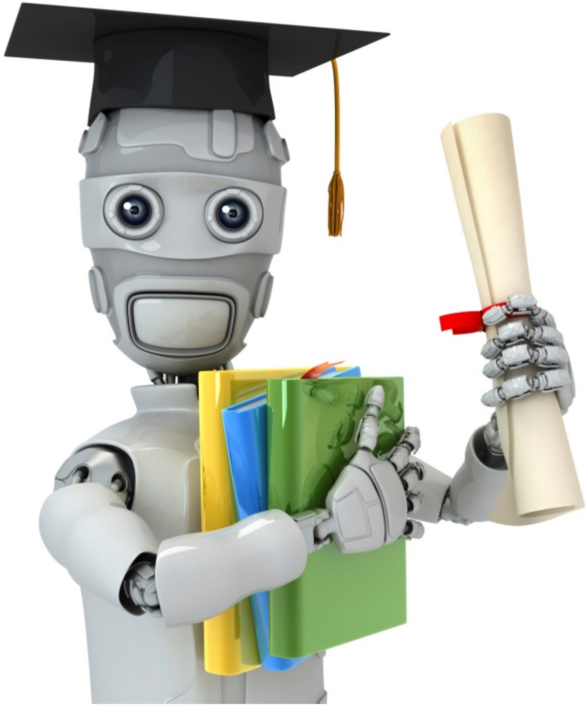
Andrew Ng

PCA is not linear regression





Andrew Ng



Machine Learning

Dimensionality Reduction

Principal Component Analysis algorithm

Data preprocessing

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

Preprocessing (feature scaling/mean normalization):

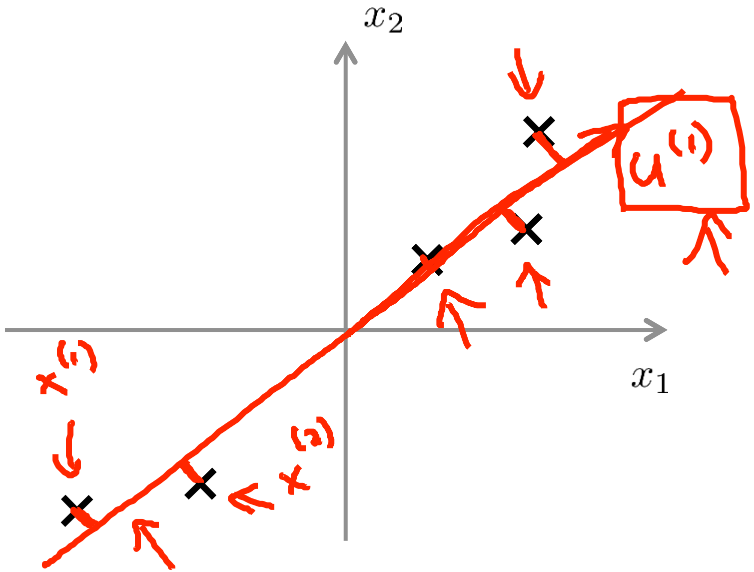
$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

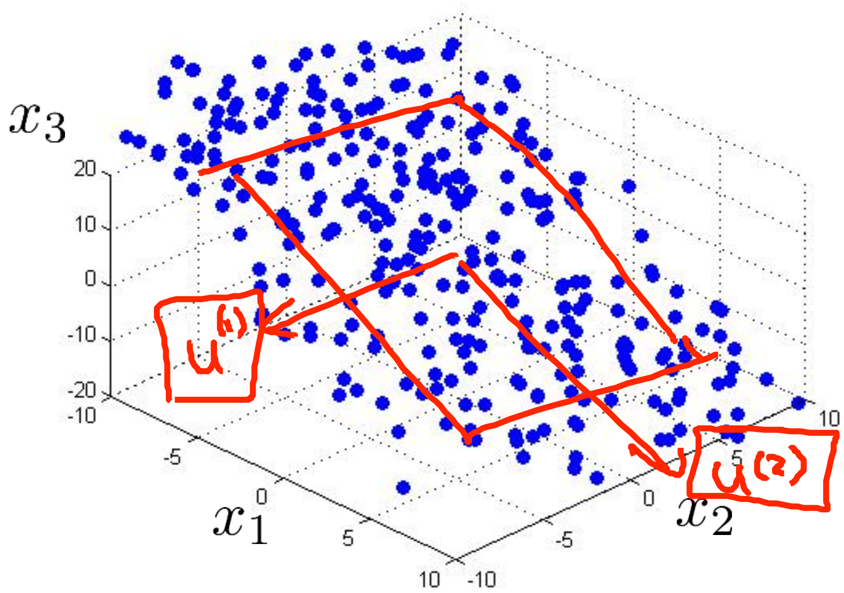
Replace each $x_j^{(i)}$ with $x_j - \mu_j$.

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.

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

Principal Component Analysis (PCA) algorithm



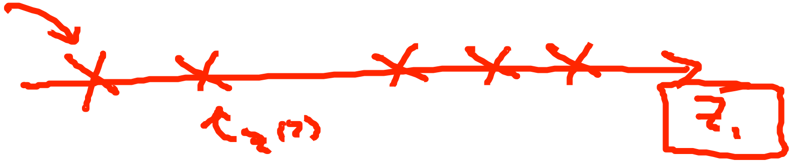


Reduce data from 2D to 1D

Reduce data from 3D to 2D

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

$z^{(i)}$



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

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

Principal Component Analysis (PCA) algorithm

Reduce data from n-dimensions to k-dimensions

Compute "covariance matrix":

$$\Sigma = \frac{1}{m}\sum_{i=1}^{n} \boxed{\underbrace{(x^{(i)})}_{n\times 1}\underbrace{(x^{(i)})^T}_{1\times n}} \quad n\times n$$

Sigma

Compute "eigenvectors" of matrix Σ :

→ Singular value decomposition

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

eig(Sigma)

n×n matrix

svd 将输出三个矩阵
分别是 U S V
你真正需要的是 U 矩阵
U 矩阵也是一个
n×n 矩阵
U 矩阵的列就是
我们需要的 u(1) u(2) 等等
如果我们想 将数据的维度从
n 降低到 k 的话
我们只需要提取前 k 列向量

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

k

$U \in \mathbb{R}^{n\times n}$

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

Andrew Ng

Principal Component Analysis (PCA) algorithm

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

$$\rightarrow U = \left[\begin{array}{cccc} | & | & & | \\ u^{(1)} & u^{(2)} & \ldots & u^{(n)} \\ | & | & & | \end{array}\right] \in \mathbb{R}^{n\times n}$$

$$\underbrace{u^{(1)} \; u^{(2)}}_{k}$$

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

$$z^{(i)} = \underbrace{\underbrace{\left[\begin{array}{cccc} | & | & & | \\ u^{(1)} & u^{(2)} & \ldots & u^{(k)} \\ | & | & & | \end{array}\right]^T}_{n\times k}}_{U_{reduce}} x^{(i)} = \underbrace{\underbrace{\left[\begin{array}{c} —(u^{(1)})^T— \\ \vdots \\ —(u^{(k)})^T— \end{array}\right]}_{k\times n} \underbrace{x^{(i)}}_{n\times 1}}_{k\times 1}$$

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

Principal Component Analysis (PCA) algorithm summary

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

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

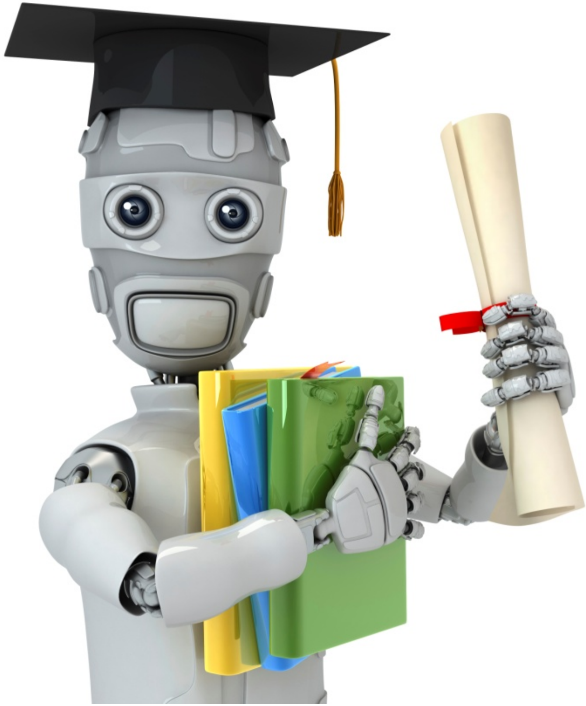
$$X = \begin{bmatrix} — & x^{(1)T} & — \\ & \vdots & \\ — & x^{(m)T} & — \end{bmatrix}$$

$$\boxed{\texttt{Sigma = (1/m) * X' * X;}}$$

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

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

我没有证明确实能够使得 平方投影误差最小化
证明已经超出了这门课的范围



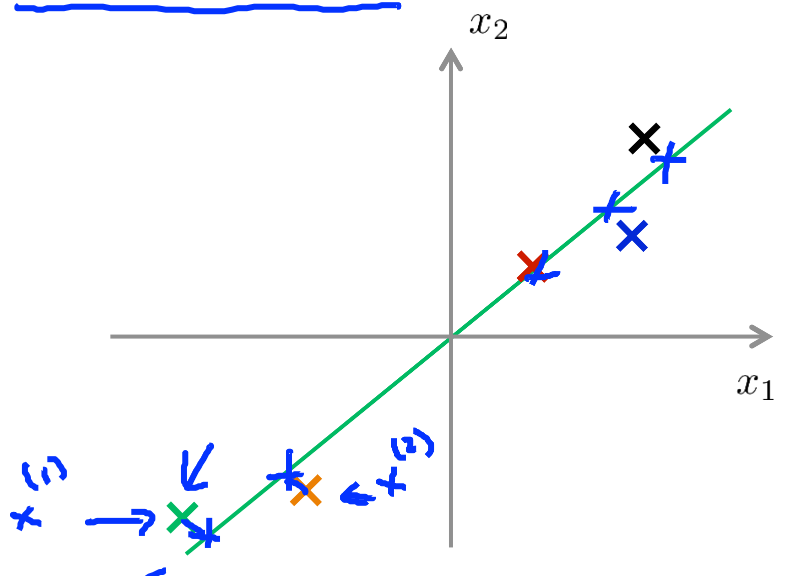
Machine Learning

Dimensionality Reduction

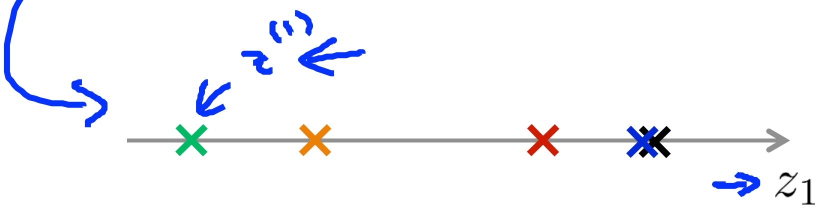
Reconstruction from compressed representation

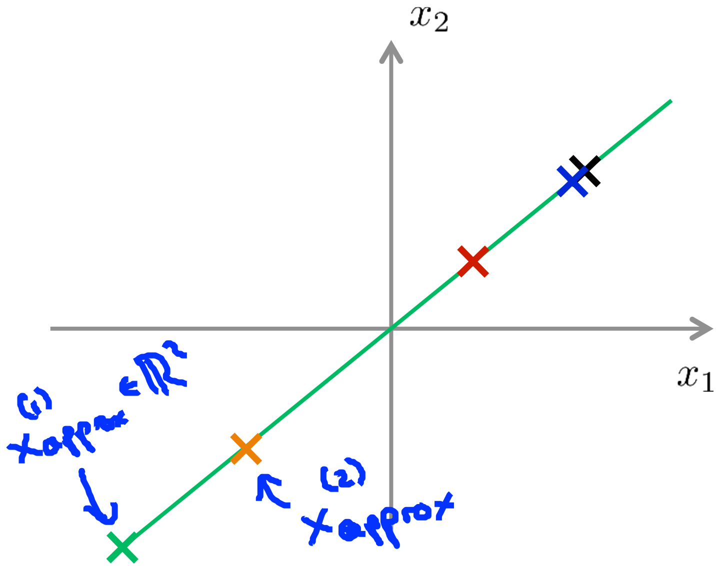
可以从压缩过的数据 近似地回到原始高维度的数据

Reconstruction from compressed representation



$$z = U_{reduce}^T x$$



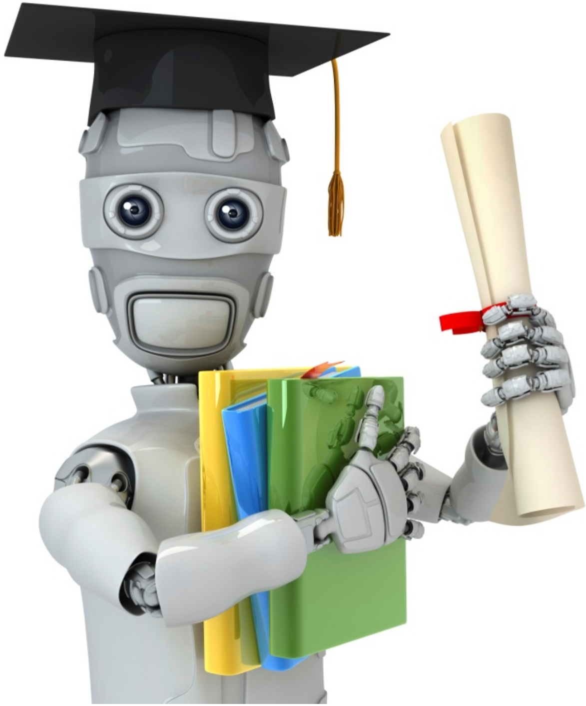


$z \in \mathbb{R} \rightarrow x \in \mathbb{R}^2$

$\tilde{x}$

$$\underbrace{X_{approx}^{(1)}}_{\mathbb{R}^n} = \underbrace{\underbrace{U_{reduce}}_{n \times k} \cdot \underbrace{z^{(1)}}_{k \times 1}}_{n \times 1}$$

Andrew Ng



Machine Learning

Dimensionality Reduction

Choosing the number of principal components

Choosing k (number of principal components)

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

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

Typically, choose k to be smallest value so that

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

$\cancel{0.05}$ $\cancel{5\%}$

0.10 (10%)

→ "~~99%~~ of variance is retained"

~~95%~~ 90%

一个常见的 选择K值的经验法则是 选择能够使得它们之间的比例 小于等于0.01

Choosing k (number of principal components)

Algorithm:

Try PCA with $\cancel{k = 1}$ $\cancel{k=2}$ $\cancel{k=3}$ $k=4$ ...

Compute $U_{reduce}, z^{(1)}, z^{(2)}, \ldots, z^{(m)}, x^{(1)}_{approx}, \ldots, x^{(m)}_{approx}$

Check if

$$\boxed{\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x^{(i)}_{approx}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2}} \le 0.01?$$

$k = 17$

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

$$S = \begin{bmatrix} S_{11} & & & & 0 \\ & S_{22} & & & \\ & & S_{33} & & \\ & & & \ddots & \\ 0 & & & & S_{nn} \end{bmatrix}$$

$k=3$

For given k

$$1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \le 0.01$$

$$\boxed{\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \ge 0.99}$$

Choosing k (number of principal components)

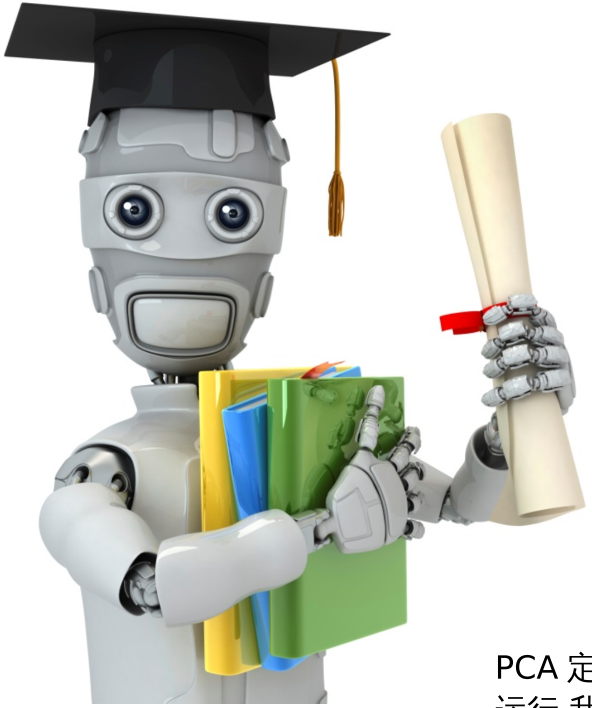
```
[U,S,V] = svd(Sigma)
```

Pick smallest value of k for which

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

$k=100$

(99% of variance retained)



Machine Learning

Dimensionality Reduction

Advice for applying PCA

PCA 定义了从 x到z的对应关系 这种从 x 到 z的对应关系只可以通过 在训练集上
运行 我们需要 使我们的参数唯一地适应 这些训练集 而不是 适应我们的交叉验证
或者测试集 因此Ureduce矩阵中的数据 就应该 只通过对训练集运行PCA来获得
在训练集中找到了 所有这些参数后 就可以 将同样的对应关系应用到其他样本中了
可能 是交叉验证数集样本 或者 用在你的测试数据集中

$x^{(i)} \in \mathbb{R}^{10{,}000}$

100

100

Supervised learning speedup

$\boxed{(x^{(1)}, y^{(1)})}, (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

Extract inputs:

Unlabeled dataset: $x^{(1)}, x^{(2)}, \ldots, x^{(m)} \in \mathbb{R}^{10000}$

$$\downarrow PCA$$

U_{reduce}

$z^{(1)}, z^{(2)}, \ldots, z^{(m)} \in \mathbb{R}^{1000}$

$x \downarrow z$

New training set:

$\boxed{(z^{(1)}, y^{(1)})}, (z^{(2)}, y^{(2)}), \ldots, (z^{(m)}, y^{(m)})$

$h_\theta(z) = \frac{1}{1+e^{-\theta^T z}}$

Note: Mapping $x^{(i)} \rightarrow z^{(i)}$ should be defined by running PCA only on the training set. This mapping can be applied as well to the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross validation and test sets.

$x \to z$

Application of PCA

(Avado)就分类的精确度而言 数据降维后对学习算法 几乎没有什么影响
如果我们将降维 用在低维数据上 我们的学习算法会运行得更快

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

Choose k by % of variance retain

- Visualization

k=2 or k=3

Bad use of PCA: To prevent overfitting

→ Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n$. — 10000

1000

Thus, fewer features, less likely to overfit.

如果你仔细想想PCA是如何工作的 它并不需要使用数据的标签
PCA做了什么呢 它把某些信息舍弃掉了

Bad!

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

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

同时正则化 效果也会比PCA更好
总之 使用PCA的目的是 加速 学习算法的时候是好的
但是用它来避免过拟合 却并不是一个好的PCA应用

PCA is sometimes used where it shouldn't be

Design of ML system:

- Get training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$
- ~~Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$~~
- Train logistic regression on $\{(\cancel{z^{(1)}}\, x^{(i)}, y^{(1)}), \ldots, (\cancel{z^{(m)}}\, x^{(m)}, y^{(m)})\}$
- Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$. Run $h_\theta(z)$ on $\{(z_{test}^{(1)}, y_{test}^{(1)}), \ldots, (z_{test}^{(m)}, y_{test}^{(m)})\}$

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 $\boxed{x^{(i)}}$. Only if that doesn't do what you want, then implement PCA and consider using $\underline{z^{(i)}}$.