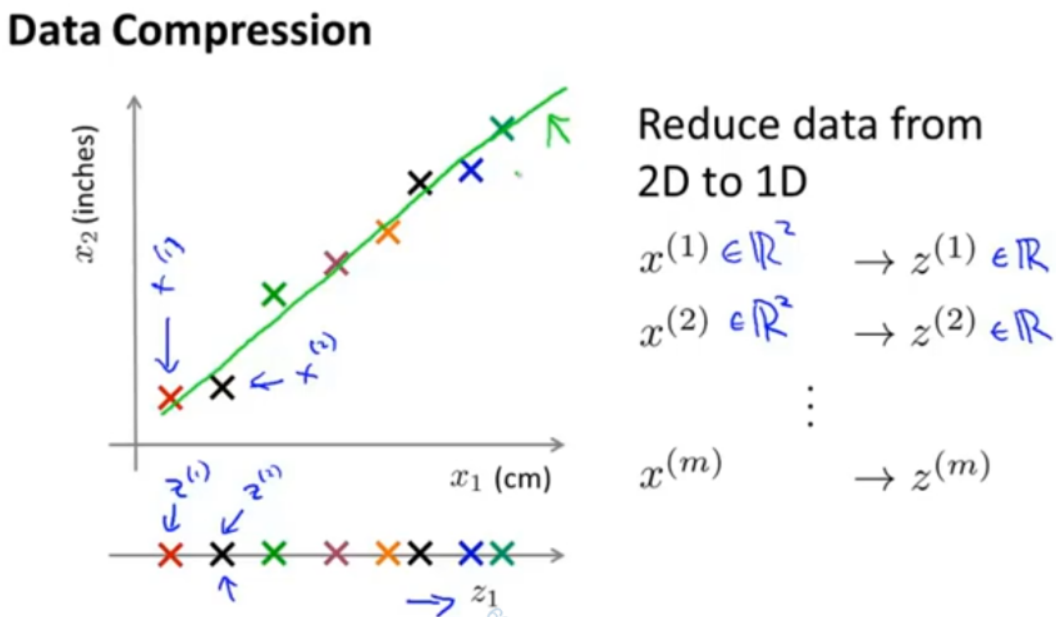
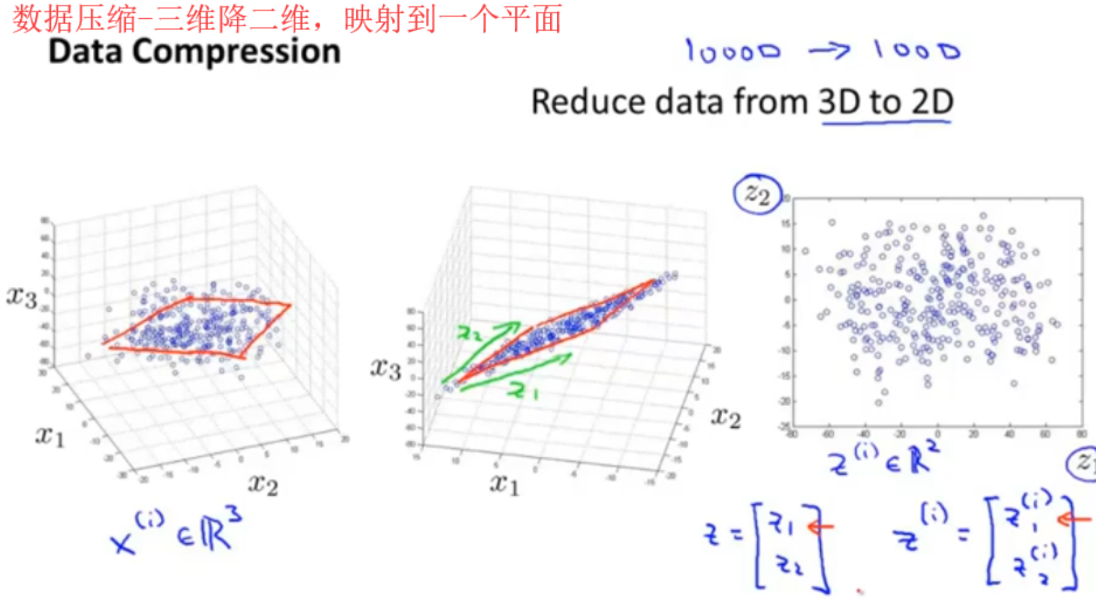
12.吴恩达-机器学习+无监督算法另一类-降维

笔记本: 日常
创建时间: 2019/11/18 9:13 **更新时间:** 2019/11/18 10:31
作者: 296645429@qq.com

数据压缩-二维降一维，映射到一条线上



数据压缩-三维降二维，映射到一个平面



降维-数据可视化：将Z1、Z2分别表示横纵轴查看相关信息

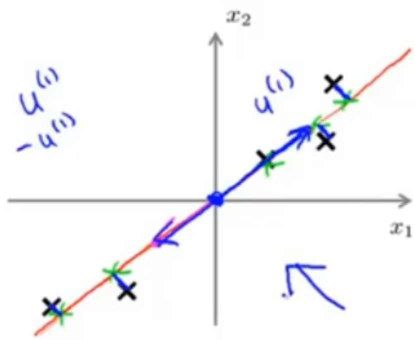
Data Visualization

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

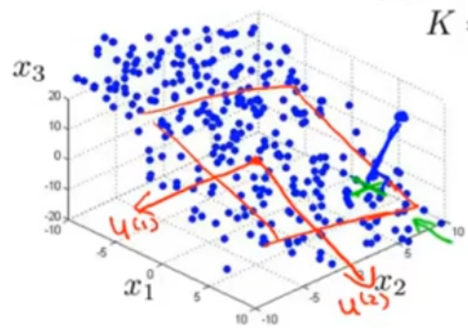
Country	z_1	z_2
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
…	…	…

主成分分析法（PCA）

Principal Component Analysis (PCA) problem formulation



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

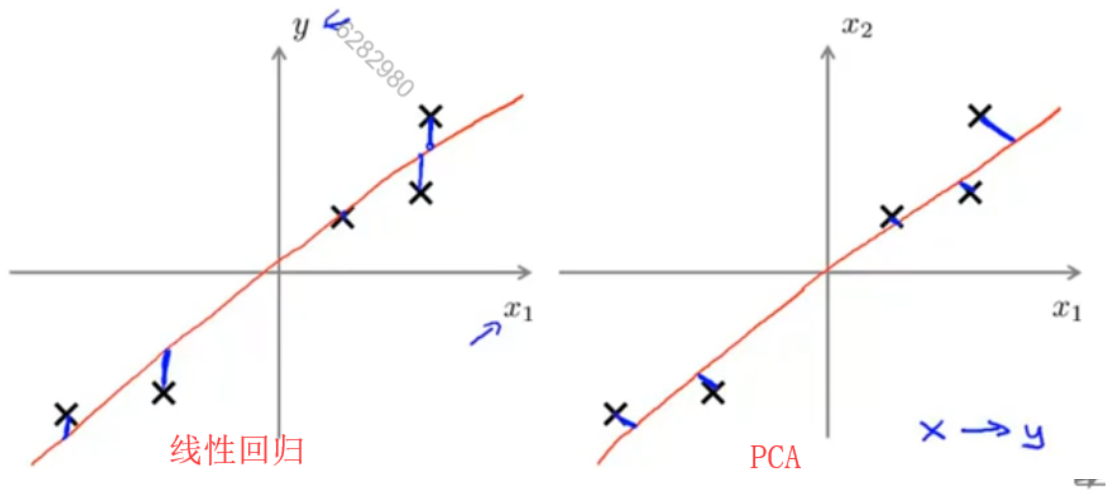


Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

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和线性回归的差异：前者最小化的是垂直线条投影的长度，后者最小化垂直于x投影的长度

PCA is not linear regression



主成分分析流程-数据预处理

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.

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

主成分分析流程-降维（求协方差、奇异值分解）

Principal Component Analysis (PCA) algorithm

Reduce data from n-dimensions to k-dimensions

Compute "covariance matrix":

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

Sigma

Compute "eigenvectors" of matrix Σ:

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

nxn matrix.

→ Singular value decomposition

eig(Sigma)

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

k

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

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

主成分分析-奇异值分解

Principal Component Analysis (PCA) algorithm

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

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

k

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

$$z = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}}_{n\times k}^T x = \underbrace{\underbrace{\begin{bmatrix} — (u^{(1)})^T — \\ \vdots \\ — (u^{(k)})^T — \end{bmatrix}}_{k\times n} \underbrace{x}_{n\times 1}}_{k\times 1}$$

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

U_{reduce}

主成分分析算法流程总结

Principal Component Analysis (PCA) algorithm summary

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

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

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

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

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

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

主成分数量k选择

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

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

~~0.05~~ 0.10 ~~5%~~ (10%)

→ "~~99%~~ of variance is retained"

~~95%~~ 90%

主成分数量k选择

Choosing k (number of principal components)

Algorithm:

Try PCA with ~~$k=1$~~ ~~k=2~~ ~~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,$\boxed{\text{S}}$,V] = svd(Sigma)

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

k=3

For given k

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

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

Andrew

主成分数量k选择

Choosing k (number of principal components)

→ [U,S,V] = svd(Sigma)

Pick smallest value of k for which

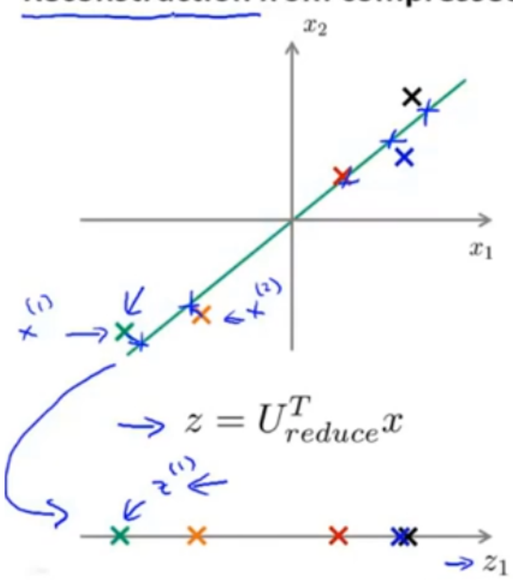
$$\boxed{\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{m} S_{ii}}} \ge 0.99$$

k=100

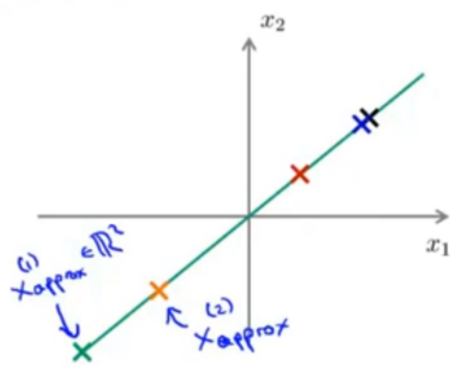
(99% of variance retained)

压缩重现

Reconstruction from compressed representation



$$\rightarrow z = U^T_{reduce}x$$



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

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

PCA-加速学习速度

Supervised learning speedup

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

100

100

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

Extract inputs:

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

$\downarrow PCA$

U_{reduce}

$x \rightarrow z$

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

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.

$x \rightarrow z$

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 $k=3$

Bad use of PCA: To prevent overfitting

Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n$. — 1000, 10000

Thus, fewer features, less likely to overfit.

Bad!

正确的处理过拟合方法

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

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

建议往往不用PCA，只有程序运行不起来是考虑使用PCA降维

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 $\{(\cancel{z^{(1)}}\ x^{(1)}, y^{(1)}), \ldots, (\cancel{z^{(m)}}\ x^{(m)}, y^{(m)})\}$
- 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 $z^{(i)}$.