

# Unsupervised learning

## 目录

一.Clustering.....	1
范数 norm.....	1
1 K-Means.....	2
1.1 K-Means Algorithm.....	2
Optimization Objective.....	4
cost function.....	4
Random Initialization & Randomly initialize centroid.....	5
Local Optima.....	5
Choosing the Number of Clusters.....	6
二.Dimensionality Reduction (Data Compression) PCA.....	7
PCA Principal Component Analysis (就是线性代数投影到基向量! ).....	9
feature scaling /mean normalization.....	10
Reconstruction from Compressed Representation.....	12
Choose the K for PCA.....	14
Applying PCA.....	15
三.Anomaly Detection 异常检测 算法: .....	16
anomaly detect example: .....	16
Gaussian (normal)Distribution.....	18
Parameter estimation.....	18
Algorithm.....	19
Developing and Evaluating an Anomaly Detection System(重点是 how to evaluate)....	20
Anomaly Detection vs. Supervised Learning (两种方法对比).....	21
Choosing Features.....	22
Multivariate Gaussian Distribution(改进之前每个 feature 建模的缺点).....	23
Anomaly Detection using the Multivariate Gaussian Distribution.....	26
多元高斯分布和之前的 feature 单独建模的关系.....	27

## 一.Clustering

### 范数 norm

**范数 norm:** 向量的范数就是向量的长度

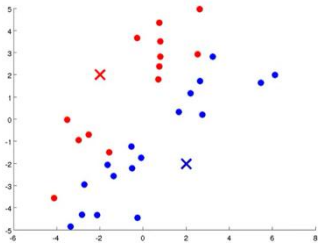
$||x||$  为  $x$  向量各个元素平方和的  $1/2$  次方

两个点的距离，就是两个向量之差的范数  $\|x-y\|$

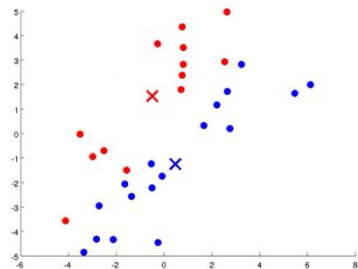
## 1 K-Means

### 1.1 K-Means Algorithm

- 1 first randomly 选  $K$  个 cluster centroids
- 2 iterater algorithm with 2 steps
  - 1 cluster assigment : go through all examples ,assign each data to one cluster, base on 哪一个 close



- 2 move centroid: 将 centroid move 到 the 每个 cluster 的 mean point



一直循环 直到 centroid 不再移动

#### K-means algorithm

Randomly initialize  $K$  cluster centroids  $\underline{\mu}_1, \underline{\mu}_2, \dots, \underline{\mu}_K \in \mathbb{R}^n$

Repeat {

  for  $i = 1$  to  $m$

$c^{(i)} :=$  index (from 1 to  $K$ ) of cluster centroid  
    closest to  $x^{(i)}$

  for  $k = 1$  to  $K$

$\mu_k :=$  average (mean) of points assigned to cluster  $k$

}

2 个变量:  $c$  和  $u$

$c$ : index of **cluster** of  $x$  assigned

$u$ : 类坐标

$K$  是 total number of clusters centroid,

$k$  是第几个 centroid

1 遍历每个  $x$ , 计算欧氏距离, 选择最小的 cluster  $c$

$$\min_k \|x^{(i)} - \mu_k\|^2$$

$\swarrow$   
 $c^{(i)}$

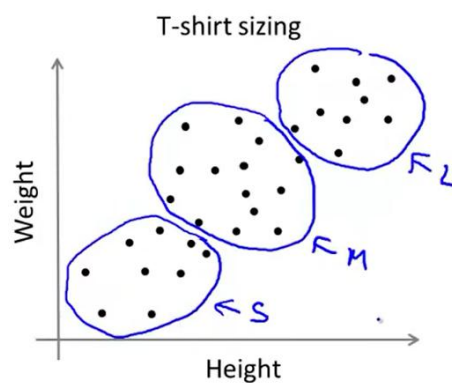
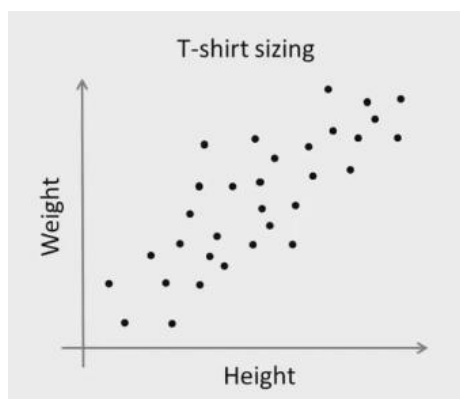
2 求 average point

$$\mu_2 = \frac{1}{4} [x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}] \in \mathbb{R}^n$$

如果有些  $c$  没有 point, 需要删除该 cluster 或从新 randomly 找个 centroid

K-Mean for **non-separated clusters**

看起来没有明显类别的数据



也会帮我们分成三类, 也就是衣服的尺寸  
market segmentation 市场细分

## Optimization Objective

### K-means optimization objective

- $c^{(i)}$  = index of cluster (1,2,...,K) to which example  $x^{(i)}$  is currently assigned
- $\mu_k$  = cluster centroid  $\underline{k}$  ( $\mu_k \in \mathbb{R}^n$ )  $K$   $k \in \{1, 2, \dots, K\}$
- $\mu_{c^{(i)}}$  = cluster centroid of cluster to which example  $x^{(i)}$  has been assigned  $x^{(i)} \rightarrow \underline{5}$   $\underline{c^{(i)} = 5}$   $\underline{\mu_{c^{(i)}} = \mu_5}$

$c^{(i)}$  是数字，当前第  $i$  个 data 的类编号 (1,2...K)

$\mu_k$  是  $k$  个 cluster centroid 是向量

$c$  是当前每个 data 的类向量

### cost function

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

$$\min_{\substack{c^{(1)}, \dots, c^{(m)}, \\ \mu_1, \dots, \mu_K}} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

cost fun 参数是  $c$  和  $\mu$  (最终是为了确定每个 data 的  $\mu$ ，以及  $k$  个 cluster centroid 向量)

使得所有 data 到各自 centroid 的距离和最小，不可能最小，需要设置个 threshold

## Random Initialization & Randomly initialize centroid

### Random initialization

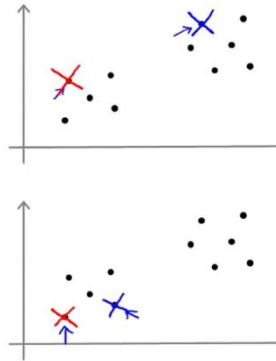
Should have  $K < m$

$K=2$

Randomly pick  $K$  training examples.

Set  $\mu_1, \dots, \mu_K$  equal to these  $K$  examples.

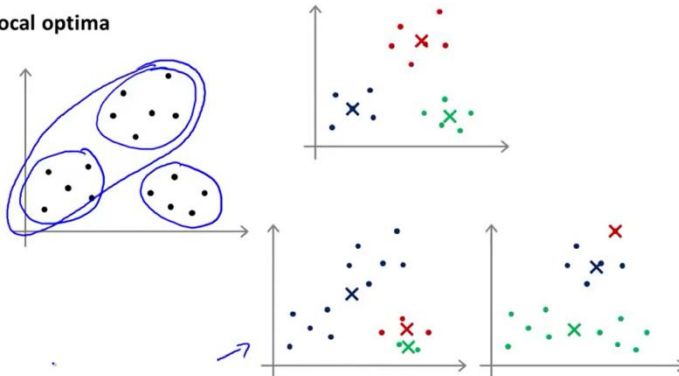
$$\begin{aligned}\mu_1 &= x^{(i)} \\ \mu_2 &= x^{(j)} \\ &\vdots\end{aligned}$$



初始类中心：在 data 里随机 pick  $K$  个数据点作为类中心

## Local Optima

### Local optima



造成不好的分类。不能很好的 minimize cost function  
解决: try multiple 多次 random initialization

### Random initialization

For  $i = 1$  to 100 {

Randomly initialize K-means.

Run K-means. Get  $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K$ .

Compute cost function (distortion)

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

}

最后 pick the lowest cost function 聚类

对于  $K=2-10$  的聚类，上面的尝试会得到更好的聚类结果

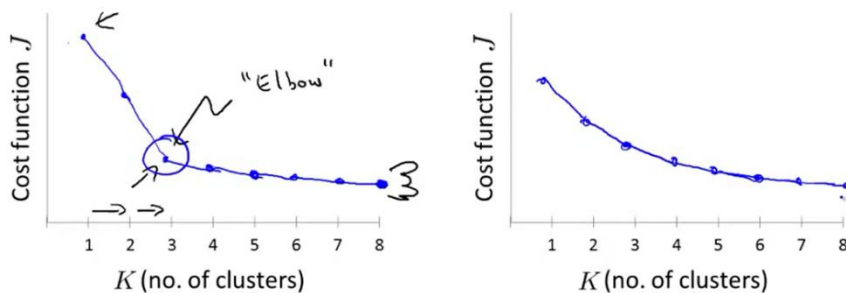
但对于很多分类，影响不会很大，和第一次结果差不多。

## Choosing the Number of Clusters

有时候很难区分 data 有多少类，ambiguous 歧义的

### Choosing the value of K

Elbow method:

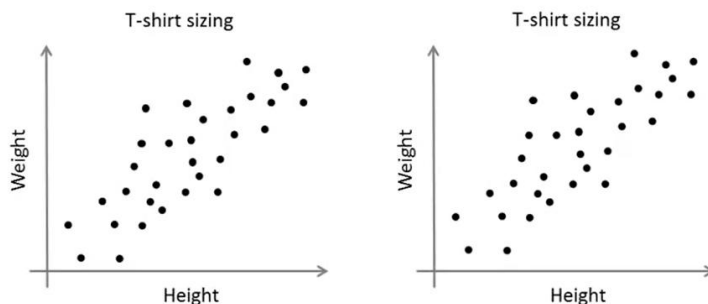


但更多是右边情况，难以区分

### Choosing the value of K

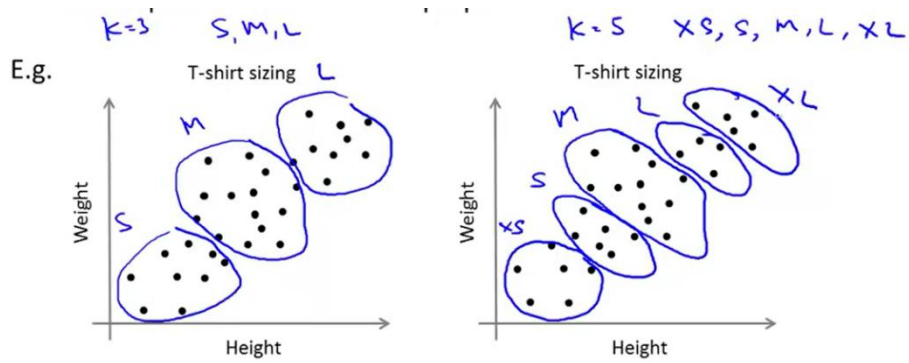
Sometimes, you're running K-means to get clusters to use for some later/downstream purpose. Evaluate K-means based on a metric for how well it performs for that later purpose.

E.g.



see later purpose(以我们的目的来做判断)，看 K 等于几，回归实际作用效果好

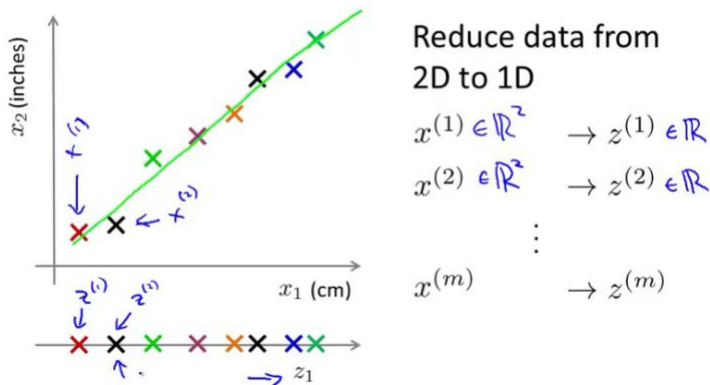
T 恤衫：看市场反应，看成本，看销售成本



左边是为了按体型划分  
 右边是划分更加精细  
 根据实际用途选择  $k$ !!!

## 二 .Dimensionality Reduction (Data Compression) PCA

### Data Compression

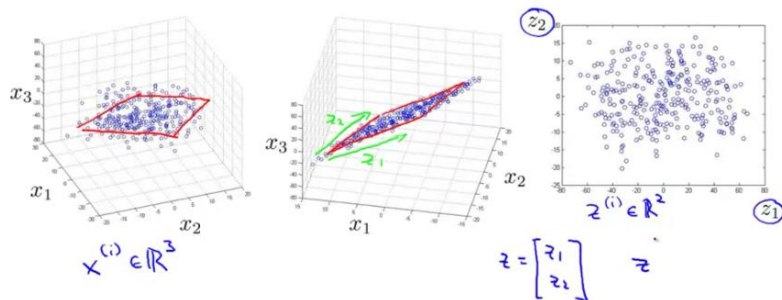


用一维代表两维

## Data Compression

10000 → 1000

Reduce data from 3D to 2D



三维数据大致分布在一个平面上，投影至二维，用二维平面去代替。

## Data Visualization

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

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

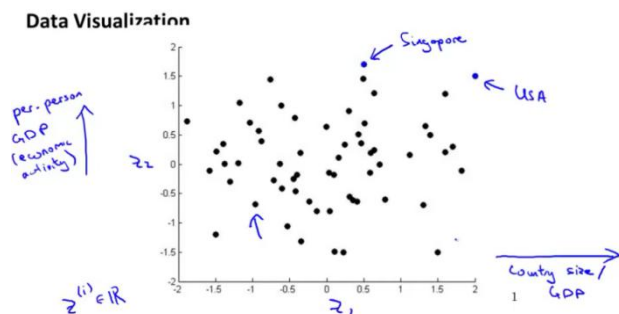
Country	$x_1$ GDP (trillions of US\$)	$x_2$ Per capita GDP (thousands of intl. \$)	$x_3$ Human Develop- ment Index	$x_4$ Life expectancy	$x_5$ Poverty Index (Gini as percentage)	$x_6$ Mean household income (thousands of US\$)	...
Canada	1.577	39.17	0.908	80.7	32.6	67.293	...
China	5.878	7.54	0.687	73	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	...

50 个 features 变 2 个

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

此时可以 visualize

两个 main deviation 主要方差:

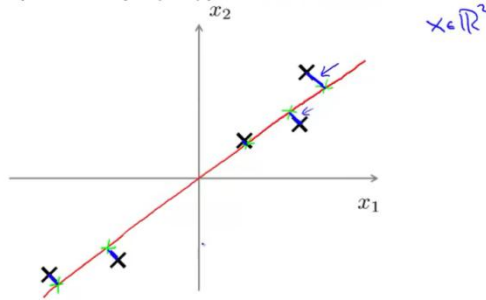


横轴 国际 size gdp 纵轴 person per gdp



# PCA Principal Component Analysis (就是线性代数投影到基向量! )

Principal Component Analysis (PCA) problem formulation

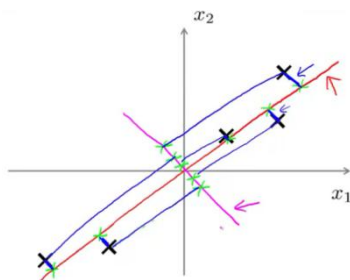


Find a **plane** 使得每个点到平面的垂直投影距离的平方和 (projection error(垂直误差)和最小二乘 error 不同!! ) 最小

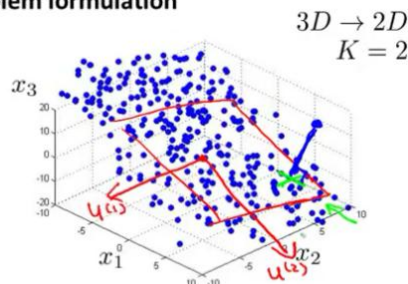
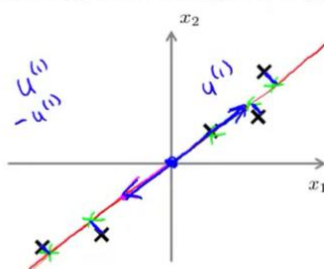
说明垂直方向数据没有什么方差，而 plane 方向方差最大！  
也就是寻找数据方差最大的方向的 plane

PCA 之前先要做 feature scaling 标准化

垂直方向：距离最大，最能代表数据的波动，方差最大的方向



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.

先找到这个 **hyperplane** (data projection 距离平方和最小), 假如是  $K$  维!

这个平面是由  $k$  个基向量  $u$  决定!! 每个向量代表一个方向  
 **$k$  个向量可以代表这个平面!!!**

project data onto subspace **spanned by  $k$  vectors**

so to minimize the project 距离

投影到基向量空间

基向量  $u$  就是我们得到的新低维 feature

## feature scaling / mean normalization

### Data preprocessing

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  $x_j - \mu_j$ .

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.

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

scaling 是单位, 统一单位就是除以标准差或 range  
已经标准化的  $x$   $\mu=0$  方差 1

### Principal Component Analysis (PCA) algorithm

Reduce data from  $n$ -dimensions to  $k$ -dimensions

Compute "covariance matrix":

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

$\Sigma$  is  $n \times n$  matrix.  $\Sigma$  is Sigma

Compute "eigenvectors" of matrix  $\Sigma$ :

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

$\rightarrow$  Singular value decomposition  $\text{eig}(\Sigma)$

$U$  is  $n \times n$  matrix.

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

$U \in \mathbb{R}^{n \times n}$   
 $u^{(1)}, \dots, u^{(k)}$

$k$

## 1 先计算 x 的协方差矩阵

协方差：

设  $X_1, X_2, \dots, X_n$  为一组随机变量，这些随机变量构成随机向量  $X = [X_1, X_2, \dots, X_n]^T$ ，每个随机变量有  $m$  个样本，则有样本矩阵

单随机变量间的协方差：

随机变量  $X_i, X_j$  之间的协方差可以表示为

$$c_{ij} = E\{[X_i - E(X_i)][X_j - E(X_j)]\}$$

根据已知的样本值可以得到协方差的估计值如下：

$$c_{ij} = \frac{1}{m} \sum_{k=1}^m [(M_{ik} - \frac{1}{m} \sum_{a=1}^m M_{ia})(M_{jk} - \frac{1}{m} \sum_{b=1}^m M_{jb})]$$

可以进一步地简化为：

$$\begin{aligned} c_{ij} &= \frac{1}{m} \sum_{k=1}^m M_{ik} M_{jk} - \frac{1}{m^2} \sum_{a=1}^m M_{ia} \sum_{b=1}^m M_{jb} \\ &= \frac{1}{m} \bar{\alpha}_i^T \cdot \bar{\alpha}_j - \frac{1}{m^2} \sum_{a=1}^m M_{ia} \sum_{b=1}^m M_{jb} \end{aligned}$$

协方差矩阵：

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix}$$

## 相关系数矩阵：

又相关系数公式，**经标准化的样本数据的协方差矩阵就是原始样本数据的相关矩阵**。这里所说的标准化指正态化，即将原始数据处理成均值为 0，方差为 1 的标准数据。

计算相关系数矩阵的公式：

$x^{(1)}(x^{(1)})^T$  是  $n \times n$  维矩阵，最后加总：

Compute "covariance matrix":

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

$n \times n$

还是  $n \times n$  维矩阵。

向量实现;

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

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

$X = \begin{bmatrix} x^{(1)T} \\ \vdots \\ x^{(m)T} \end{bmatrix}$   
 $\text{Sigma} = (1/m) \times X' \times X$

加总总是可以由矩阵乘法实现。构造矩阵  $X$

## 2 计算协方差矩阵 $\text{sigma}$ 的特征向量

$\text{sigma}$  covariance matrix 满足正定矩阵  $n \times n$  的矩阵

利用 SVD 得到  $U$   $n \times n$  的矩阵

矩阵  $U$  每个列向量就是我们要找的方向向量也是  $n \times n$  矩阵

Compute "eigenvectors" of matrix  $\Sigma$ :

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

$n \times n$  matrix

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

只需取前  $K$  个, 得到  $U_{\text{reduce}}$   $n \times k$  矩阵

$U(\text{reduce})$  转置是  $k \times n$  来乘以每一个数据点  $n \times 1$  向量, 得到  $k \times 1$  向量, 就将我们的  $n$  维转成  $k$  维

**直接  $X(m, n)$  乘  $U(n \times k)$  得到  $Z:(m, k)$ .  $m$  个  $K$  维向量!!**

## Reconstruction from Compressed Representation

如何从低维数据  $Z$  还原高维数据  $X$ ?

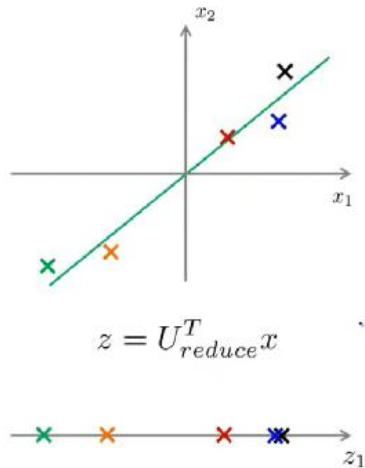
$$z = U_{\text{reduce}}^T x$$

$z$  是  $(m, k)$

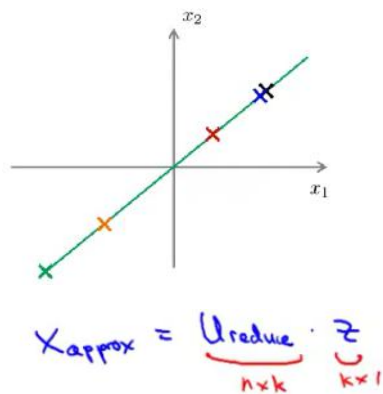
U reduce 乘 z 转置得 (n\*m) 维的 X 近似数据

$$X_{\text{approx}} = \underbrace{U_{\text{reduce}}}_{n \times k} \cdot \underbrace{z}_{k \times 1}$$

图解：



原始点不在平面上，现在投影在平面上，得到 1 维数据



还原时，只能将二维还原到平面上的点，不再是原始点，因此是 approximate 近似等于 x

## Choose the K for PCA

### 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

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

"99% of variance is retained"

1 projection error:  $\|x - x_{approx}\|$

2 total variation of data

让误差远远小于数据的总方差

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

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

$k=17$

matlab 简化算法:

### Choosing $k$ (number of principal components)

Algorithm:

Try PCA with  $k=1$   ~~$k=2$~~   ~~$k=3$~~   $k=4$

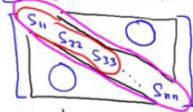
Compute  $U_{reduce}, z^{(1)}, z^{(2)}, \dots, z^{(m)}, x_{approx}^{(1)}, \dots, x_{approx}^{(m)}$

Check if

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

$k=17$

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

$S =$  

For given  $k$   $k=3$

$$1 - \frac{\sum_{i=1}^k s_{ii}}{\sum_{i=1}^n s_{ii}} \leq 0.01$$

$$\frac{\sum_{i=1}^k s_{ii}}{\sum_{i=1}^n s_{ii}} \geq 0.99$$

Andrew

## Applying PCA

### 1 speed up supervised learning algorithm:

#### Supervised learning speedup

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

Extract inputs:

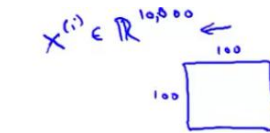
Unlabeled dataset:  $x^{(1)}, x^{(2)}, \dots, x^{(m)} \in \mathbb{R}^{10000}$

$\downarrow PCA$

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

New training set:

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



$$h_{\theta}(z) = 1$$

先对 **x train** 降维，降维度后 **y** 不变

注意不可以所有 **data(train,vc,test)** 一起 **pca**，得到的特征会用到 **test** 和 **vc** 的 **data** 的信息，**train** 建的模型包含了 **test** 信息！

所有要先 **split train, vc, test**

Note: Mapping  $x^{(i)} \rightarrow z^{(i)}$  should be defined by running PCA  $x \rightarrow z$  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.

用 **train** 来做 PCA，得到 **U reduce**

然后用 **U reduce**，对 **cv** 以及 **test** 数据降维

#### 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.

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



不应该用 PCA 来降维 feature, 防止过拟合

因为 PCA 没有用到  $y(\text{label})$  的信息, 会丢失 data 的信息!!!

什么时候用 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  $z^{(i)}$ .

一定要先用 raw data!! 不要滥用 PCA

只是因为:

- 1 运行速度慢或内存不足时再用 PCA
- 2 visualize data

### 三.Anomaly Detection 异常检测 算法:

anomaly detect example:

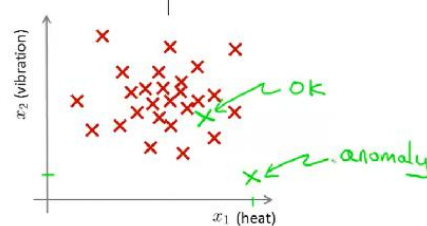
#### Anomaly detection example

Aircraft engine features:

- $x_1$  = heat generated
- $x_2$  = vibration intensity
- ...

Dataset:  $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

New engine:  $x_{test}$



unsupervised learning

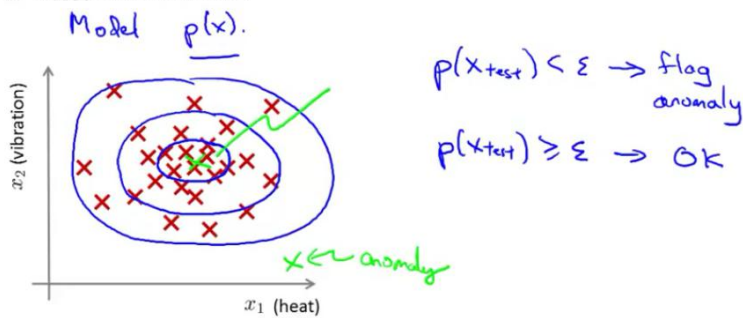


不需要知道 label

### Density estimation

→ Dataset:  $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

→ Is  $x_{test}$  anomalous?



建立一个 model:  $p(x)$  是概率函数(数据出现的概率), 当概率很小(小于一个 **small threshold** 值)是就是异常值

应用:

异常用户状态检测

机器异常运行检测

### Anomaly detection example

→ Fraud detection:

→  $x^{(i)}$  = features of user  $i$ 's activities

→ Model  $p(x)$  from data.

→ Identify unusual users by checking which have  $p(x) < \varepsilon$

$x_1$   
 $x_2$   
 $x_3$   
 $x_4$        $p(x)$

→ Manufacturing

→ Monitoring computers in a data center.

→  $x^{(i)}$  = features of machine  $i$

$x_1$  = memory use,  $x_2$  = number of disk accesses/sec,

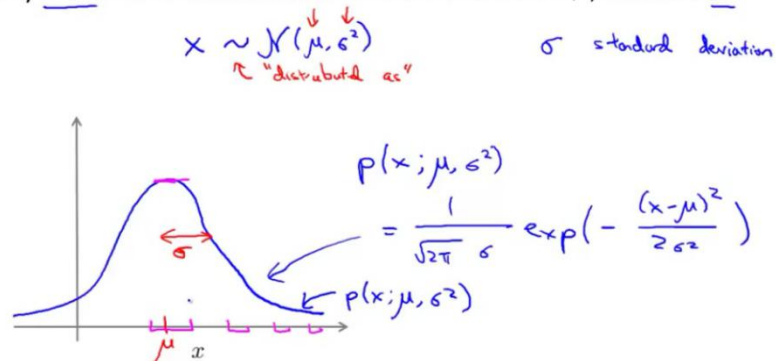
$x_3$  = CPU load,  $x_4$  = CPU load/network traffic.

...       $p(x) < \varepsilon$

## Gaussian (normal) Distribution

### Gaussian (Normal) distribution

Say  $x \in \mathbb{R}$ . If  $x$  is a distributed Gaussian with mean  $\mu$ , variance  $\sigma^2$ .



$\mu$  是对称轴,  $\sigma$  是高度

## Parameter estimation

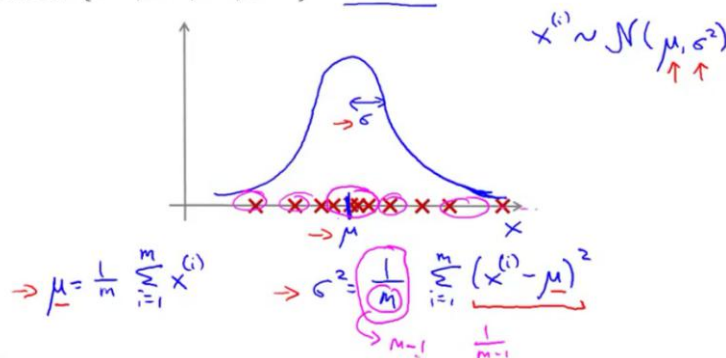
假设我们的数据服从正态分布

我们 **需要从数据中 estimate  $\mu$  和  $\sigma$**

用 **最大似然估计** 可以得到:

### Parameter estimation

Dataset:  $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$   $x^{(i)} \in \mathbb{R}$



最大似然估计的结果其实就是 mean 和 方差

## Algorithm

数据 independent assumption: 即使不成立, 模型 work well too

### Density estimation

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

Each example is  $x \in \mathbb{R}^n$

$$\begin{aligned} p(x) &= p(x_1; \mu_1, \sigma_1^2) p(x_2; \mu_2, \sigma_2^2) p(x_3; \mu_3, \sigma_3^2) \dots p(x_n; \mu_n, \sigma_n^2) \leftarrow \\ &= \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2) \end{aligned}$$

$$\begin{aligned} x_1 &\sim \mathcal{N}(\mu_1, \sigma_1^2) \\ x_2 &\sim \mathcal{N}(\mu_2, \sigma_2^2) \\ x_3 &\sim \mathcal{N}(\mu_3, \sigma_3^2) \end{aligned}$$

每个 feature 都是随机变量, 假设服从不同的正态分布, 且独立

一个 example 出现的概率  $p(x)$ , 就是每个 feature 概率乘积  
 $p(x)$  很小就说明是在分布的尾部, 属于异常值。

### Anomaly detection algorithm

1. Choose features  $x_i$  that you think might be indicative of anomalous examples.
2. Fit parameters  $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$

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

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2$$

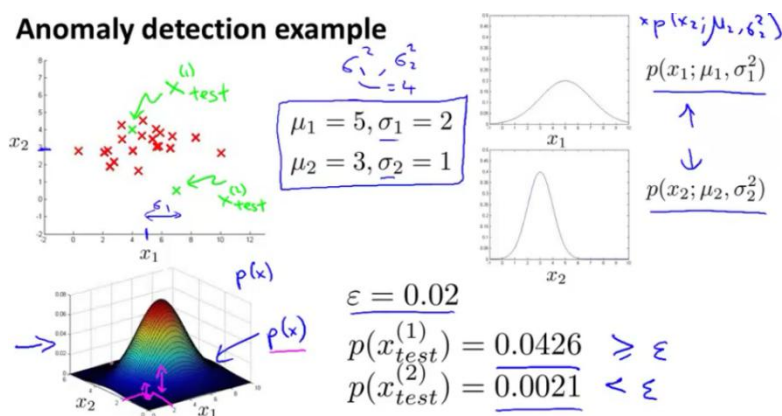
3. Given new example  $x$ , compute  $p(x)$ :

$$p(x) = \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^n \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

Anomaly if  $p(x) < \varepsilon$

先选择对诊断异常数据重要的 feature, 然后 fit 高斯模型

然后计算新的 example 的概率, 判断是否异常!!



## Developing and **Evaluating** an Anomaly Detection System(重点是 how to evaluate)

### 另一种方法 supervised learning

Assume we have some labeled data, of anomalous and non-anomalous examples. ( $y = 0$  if normal,  $y = 1$  if anomalous).

Training set:  $x^{(1)}, x^{(2)}, \dots, x^{(m)}$  (assume normal examples/not anomalous)

Cross validation set:  $(x_{cv}^{(1)}, y_{cv}^{(1)}), \dots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$

Test set:  $(x_{test}^{(1)}, y_{test}^{(1)}), \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

$y=1$

假设已经有一些 labeled 的 data, 将其分成 train 和 test set。

supervised learning train  $y=1$  是不正常, classifier 结果是 **不正常的概率 (0-1)** 其实也是  $P(x)$ , 是  $x$  不正常的概率

例子 : 正负比例比较悬殊

### Aircraft engines motivating example

10000 good (normal) engines  
20 flawed engines (anomalous) 2-50

- Training set: 6000 good engines ( $y = 0$ )
- CV: 2000 good engines ( $y = 0$ ), 10 anomalous ( $y = 1$ )
- Test: 2000 good engines ( $y = 0$ ), 10 anomalous ( $y = 1$ )

同样划分 train, test

### Algorithm evaluation

- Fit model  $p(x)$  on training set  $\{x^{(1)}, \dots, x^{(m)}\}$
- On a cross validation/test example  $x$ , predict

$$y = \begin{cases} 1 & \text{if } p(x) < \varepsilon \text{ (anomaly)} \\ 0 & \text{if } p(x) \geq \varepsilon \text{ (normal)} \end{cases}$$

Possible evaluation metrics:

- True positive, false positive, false negative, true negative
- Precision/Recall
- $F_1$ -score

Can also use cross validation set to choose parameter  $\varepsilon$

给他一个 threshold, 上面写错了: 当  $p(x) > \text{threshold}$   $y=1$

又因为我们的数据很 skew, 需要其他 metrics 来评估

## Anomaly Detection vs. Supervised Learning (两种方法对比)

Anomaly detection	vs.	Supervised learning
Very small number of positive examples ( $y = 1$ ). (0-20 is common).		Large number of positive and negative examples. ←
Large number of negative ( $y = 0$ ) examples. $p(x)$ ←		
Many different "types" of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far.		Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples likely to be similar to ones in training set. ←

左边 unsupervised learning 右边 supervised learning  
1 当很少的 positive examples 时, 很多 negative example,  
用 negative example fit  $p(x)$  高斯分布建模!!

如果用 supervised learning, 很难学到少量的正样本的 pattern, 并且以后可能会出现新的 type 的正样本, 算法就不行了

2 当正负样本很多时, (而且正样本错误的类型很全时) 可以用监督学习

总结:

错误类型很多时情况很复杂时, 而我们的数据不足时, 用 unsupervised!

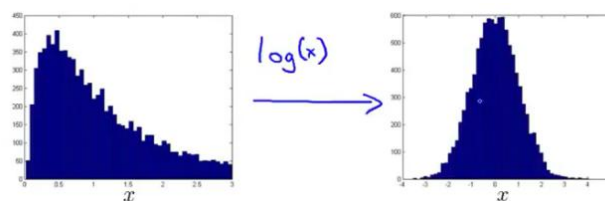
对于错误类型很少, 有 data 时用 supervised!

## Choosing Features

### non-gaussian feature 转高斯分布

对于 non-gaussian feature, 需要转成 gaussian feature 为了满足 Anomaly Detection 的假设

plot 直方图 the feature, 如果是偏态, 就需要处理



有时是:  $\log(x+c)$  或  $x^{\frac{1}{c}}$

c 需要我们自己调整, 看哪个好

通过 Anomaly Detection 错误分析, choose feature !

### Error analysis for anomaly detection

Want  $p(x)$  large for normal examples  $x$ .

$p(x)$  small for anomalous examples  $x$ .

有时候对于 anomal 和 normal,  $p(x)$  都很大

重点查看错误 data, 看能否 inspire me to create, 看能否 create 新的 feature, 就可以检测出这种错误, 例如:

$x_1$  = memory use of computer

$x_2$  = number of disk accesses/sec

$x_3$  = CPU load ←

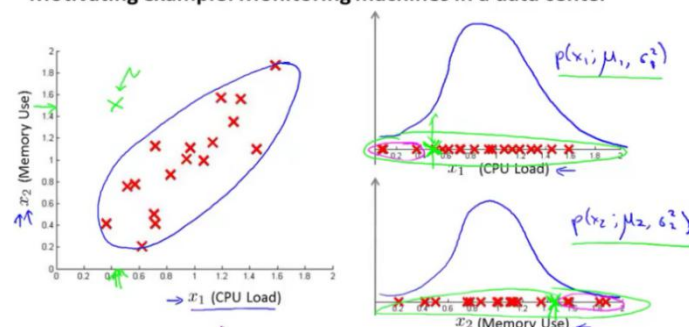
$x_4$  = network traffic ←

$$x_5 = \frac{\text{CPU load}}{\text{network traffic}}$$

$$x_6 = \frac{(\text{CPU load})^2}{\text{network traffic}}$$

Multivariate Gaussian Distribution(改进之前每个 feature 建模的缺点)

Motivating example: Monitoring machines in a data center

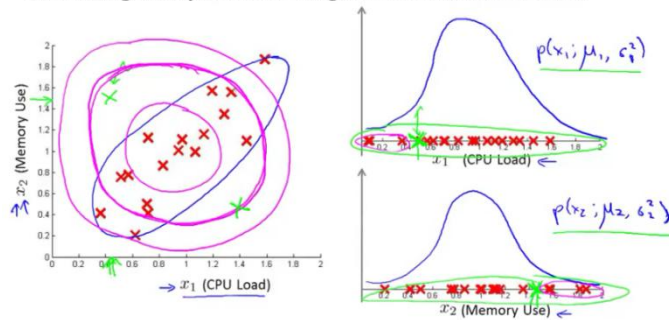


对两个 feature  $x_1, x_2$  分别建模, 正态曲线  
看左图: 有个绿色异常点, 找右图中的异常点的概率  
最终相乘的  $p(x)$  值大, 不会被诊断成异常

**需要用多元正态分布!**



### Motivating example: Monitoring machines in a data center



它的规律是一圈一圈的  $p(x)$  减少，因此这个算法不太适合了  
 $x \in \mathbb{R}^n$ . Don't model  $p(x_1), p(x_2), \dots$ , etc. separately.  
 Model  $p(x)$  all in one go.

多元正态分布：

不需要每个 feature 分别建模了，一起建立多元正态分布  
 此时模型的参数  $\mu (1 \times n)$  是均值向量， $\Sigma (n \times n)$  是协方差矩阵

Parameters:  $\mu \in \mathbb{R}^n, \Sigma \in \mathbb{R}^{n \times n}$  (covariance matrix)

### Multivariate Gaussian (Normal) distribution

Parameters  $\mu, \Sigma$

$\mu \in \mathbb{R}^n$      $\Sigma \in \mathbb{R}^{n \times n}$

$$\rightarrow p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

example:

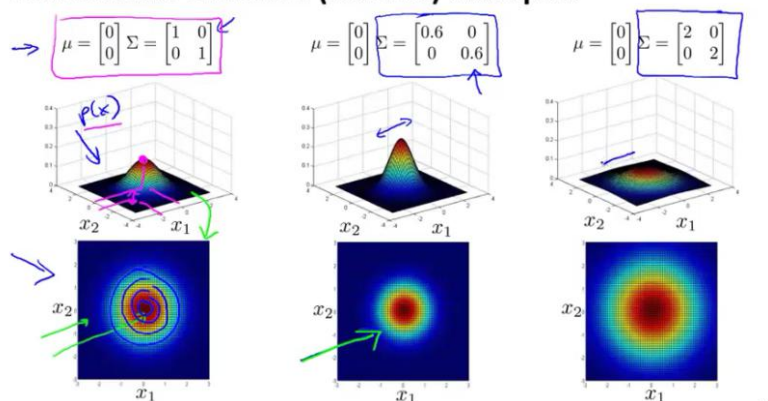
方差越大，变量的 range 越长

对角线是方差，第一个元素是第一个变量方差，第二个是第二个变量方差

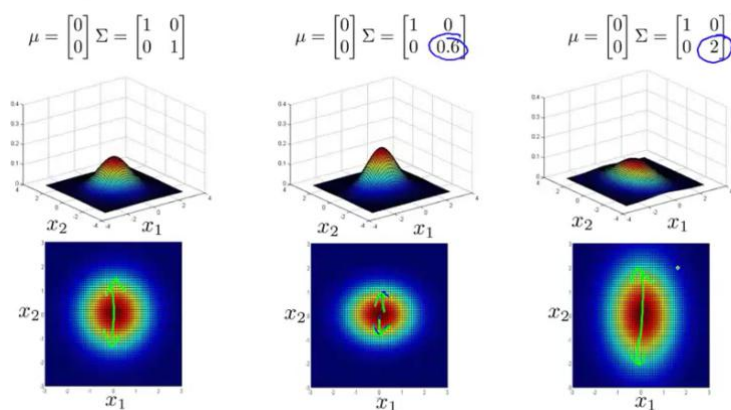
当协方差矩阵对角线相等，是圆的高斯！



## Multivariate Gaussian (Normal) examples



不等时，不是圆的而是椭圆的

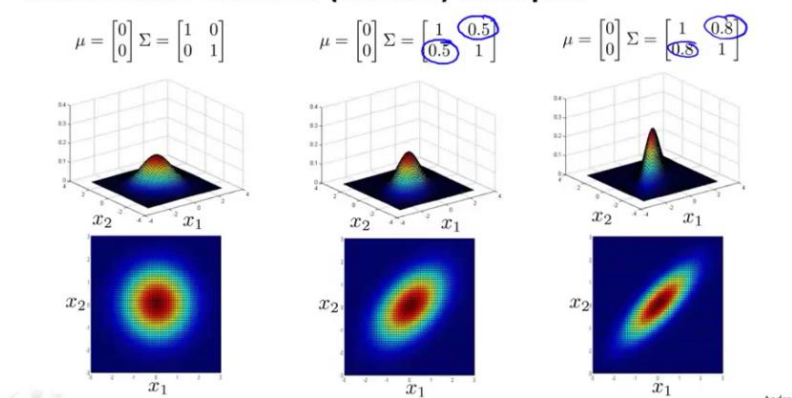


对角线以外的元素是相关系数 correlation

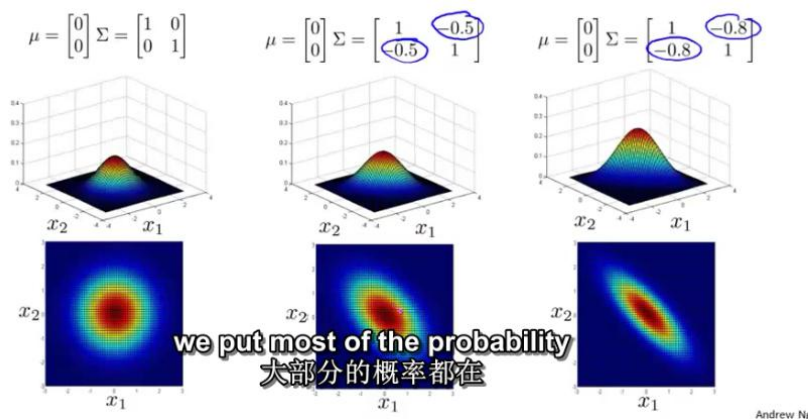
对角线以外的元素越大，圆越平行于坐标对角线

当对角线以外的元素全是 0 时，contour 和坐标轴平行!!

## Multivariate Gaussian (Normal) examples

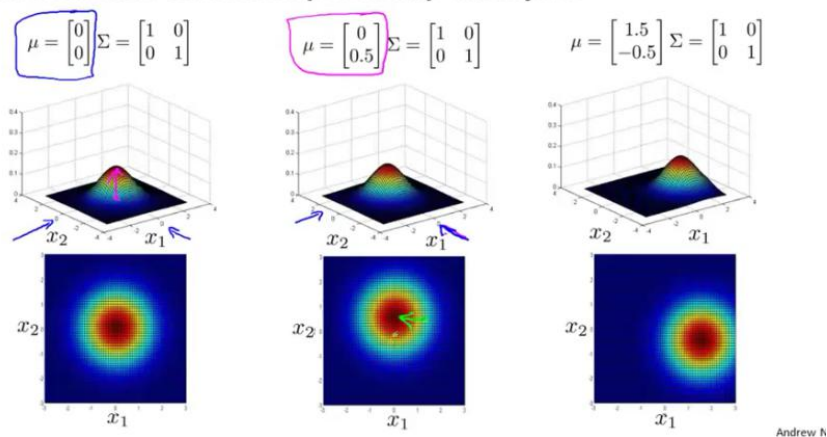


对角线元素正负，决定椭圆方向



均值向量决定位置

### Multivariate Gaussian (Normal) examples



## Anomaly Detection using the Multivariate Gaussian Distribution

parameter estimate:

Parameter fitting:

Given training set  $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\} \leftarrow x \in \mathbb{R}^n$

$$\rightarrow \mu = \frac{1}{m} \sum_{i=1}^m x^{(i)} \quad \rightarrow \Sigma = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

m 个 example n 个 feature:

均值向量每个元素是每个 feature 的均值(共 n 个)

协方差矩阵，每个 feature 跟上面的 u 元素计算方差或协方差，共  $n*n$  个组合！！

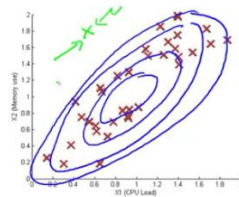
算法：

#### Anomaly detection with the multivariate Gaussian

1. Fit model  $p(x)$  by setting

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

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



2. Given a new example  $x$ , compute

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

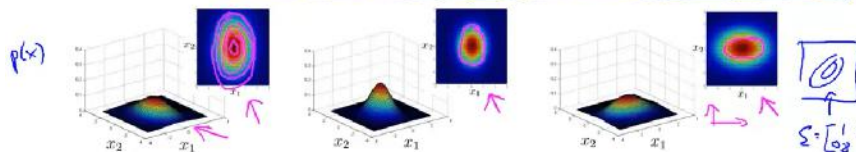
Flag an anomaly if  $p(x) < \varepsilon$

我们建立了二维的高斯分布，椭圆等高线围绕我们的数据点  
因此绿色的点的  $p(x)$  一定很低！！

#### 多元高斯分布和之前的 feature 单独建模的关系

##### Relationship to original model

Original model:  $p(x) = p(x_1; \mu_1, \sigma_1^2) \times p(x_2; \mu_2, \sigma_2^2) \times \dots \times p(x_n; \mu_n, \sigma_n^2)$



Corresponds to multivariate Gaussian

$$\rightarrow p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

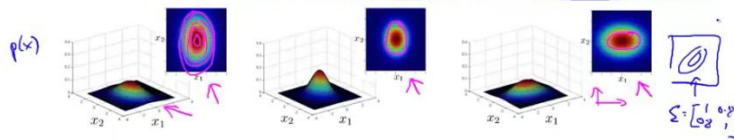
where

每个 original model 都会对应一个 multivariate gaussian model (这个 model 的条件是 the contour of gaussian always axis aligned 轴线对齐，椭圆形平行于  $x_1, x_2$  轴) **其实 original model 的值和这种特殊的多元高斯分布的值一样！**

也就是当多元高斯分布的 sigma 是对角矩阵!!

#### Relationship to original model

Original model:  $p(x) = p(x_1; \mu_1, \sigma_1^2) \times p(x_2; \mu_2, \sigma_2^2) \times \dots \times p(x_n; \mu_n, \sigma_n^2)$



Corresponds to multivariate Gaussian

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$

where

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n^2 \end{bmatrix}$$

Andrew Ng

original model 是个多元高斯的 special case!!!

original model 会用的更多

#### → Original model

$$p(x_1; \mu_1, \sigma_1^2) \times \dots \times p(x_n; \mu_n, \sigma_n^2)$$

Manually create features to capture anomalies where  $x_1, x_2$  take unusual combinations of values.

$$\rightarrow x_3 = \frac{x_1}{x_2} = \frac{\text{CPU load}}{\text{memory}}$$

Computationally cheaper (alternatively, scales better to large  $n$ )  $n=10,000, m=100,000$

OK even if  $m$  (training set size) is small

#### vs. → Multivariate Gaussian

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$

→ Automatically captures correlations between features

$$\Sigma \in \mathbb{R}^{n \times n} \quad \Sigma^{-1}$$

Computationally more expensive

Must have  $m > n$  or else  $\Sigma$  is non-invertible.  $m \geq 10n$

左边 original model 需要我们手动 create feature 来捕捉 feature combination(cross) 关系, 两种 feature 组合用于检测异常

计算量小, 可以用于大数据

但右边多元分布, 会自动捕捉上面的信息, 因为有协方差但计算量太大, 计算协方差矩阵的时候太大

还有一个区别:

左边当  $m$  很小时, 也适用

1 右边当  $m$  小于  $n$  时

协方差矩阵 singular, 不能 invert

当  $m$  远大于  $10*n$  时再用右边方法

2 当包含 redundant feature(feature 之间线性依赖: 一个 feature 是其他 feature 线性组合)

协方差矩阵也会 singular, 不能 invert