

# Machine Learning (2)

Andrew Ng

## 11. 应用机器学习的建议

### 前言

到目前为止，我们已经介绍了许多不同的学习算法，如果你一直跟着这些视频的进度学习，你会发现自己已经不知不觉地成为一个了解许多先进机器学习技术的专家了。

然而，在懂机器学习的人当中依然存在着很大的差距，一部分人确实掌握了怎样高效有力地运用这些学习算法。而另一些人他们可能对我马上要讲的东西，就不是那么熟悉了。他们可能没有完全理解怎样运用这些算法。因此总是把时间浪费在毫无意义的尝试上。我想做的是确保你在设计机器学习的系统时，你能够明白怎样选择一条最合适、最正确的道路。因此，在这节视频和之后的几段视频中，我将向你介绍一些实用的建议和指导，帮助你明白怎样进行选择。具体来讲，我将重点关注的问题是假如你在开发一个机器学习系统，或者想试着改进一个机器学习系统的性能，你应如何决定接下来应该选择哪条道路？为了解释这一问题，我想仍然使用预测房价的学习例子，假如你已经完成了正则化线性回归，也就是最小化代价函数  $J$  的值，假如，在你得到你的学习参数以后，如果你要将你的假设函数放到一组新的房屋样本上进行测试，假如说你发现在预测房价时产生了巨大的误差，现在你的问题是想要改进这个算法，接下来应该怎么办？

实际上你可以想出很多种方法来改进这个算法的性能，其中一种办法是使用更多的训练样本。具体来讲，也许你能想到通过电话调查或上门调查来获取更多的不同的房屋出售数据。遗憾的是，我看到好多人花费了好多时间想收集更多的训练样本。他们总认为，要是我有两倍甚至十倍数量的训练数据，那就一定会解决问题的是吧？但有时候获得更多的训练数据实际上并没有作用。在接下来的几段视频中，我们将解释原因。

我们也将知道怎样避免把过多的时间浪费在收集更多的训练数据上，这实际上是于事无补的。另一个方法，你也许能想到的是尝试选用更少的特征集。因此如果你有一系列特征比如  $x_1, x_2, x_3$  等等。也许有很多特征，也许你可以花一点时间从这些特征中仔细挑选一小部分来防止过拟合。或者也许你需要用更多的特征，也许目前的特征集，对你来讲并不是很有帮助。你希望从获取更多特征的角度来收集更多的数据，同样地，你可以把这个问题扩展为一个很大的项目，比如使用电话调查来得到更多的房屋案例，或者再进行土地测量来获得更多有关，这块土地的信息等等，因此这是一个复杂的问题。同样的道理，我们非常希望在花费大量时间完成这些工作之前，我们就能知道其效果如何。我们也可以尝试增加多项式特征的方法，比如  $x_1$  的平方， $x_2$  的平方， $x_1, x_2$  的乘积，我们可以花很多时间来考虑这一方法，我们也可以考虑其他方法减小或增大正则化参数  $\lambda$  的值。我们列出的这个单子，上面的很多方法都可以扩展开来扩展成一个六个月或更长时间的项目。遗憾的是，大多数人用来选择这些方法的标准是凭感觉的，也就是说，大多数人的选择方法是随便从这些方法中选择一种，比如他们会说“噢，我们来多找点数据吧”，然后花上六个月的时间收集了一大堆数据，然后也许另一个人说：“好吧，让我们来从这些房子的数据中多找点特征吧”。我很遗憾不止一次地看到很多人花了至少六个月时间来完成他们随便选择的一种方法，而在六个月或者更长时间后，他们很遗憾地发现自己选择的是一条不归路。幸运的是，有一系列简单的方法能让你事半功倍，排除掉单子上的至少一半的方法，留下那些确实有前途的方法，同时也有一种很简单的方法，只要你使用，就能很轻松地排除掉很多选择，从而为你节省大量不必要的花费的时间。最终达到改进机器学习系统性能的目的。假设我们需要用一个线性回归模型来预测房价，当我们运用训练好了的模型来预测未知数据的时候发现有较大的误差，我们下一步可以做什么？

获得更多的训练样本——通常是有效的，但代价较大，下面的方法也可能有效，可考虑先采用下面的几种方法。

- 尝试减少特征的数量
- 尝试获得更多的特征
- 尝试增加多项式特征
- 尝试减少正则化程度
- 尝试增加正则化程度

我们不应该随机选择上面的某种方法来改进我们的算法，而是运用一些机器学习诊断法来帮助我们知道上面哪些方法对我们的算法是有效的。

在接下来的两段视频中，我首先介绍怎样评估机器学习算法的性能，然后在之后的几段视频中，我将开始讨论这些方法，它们也被称为“机器学习诊断法”。“诊断法”的意思是：这是一种测试法，你通过执行这种测试，能够深入了解某种算法到底是否有用。这通常也能够告诉你，要想改进一种算法的效果，什么样的尝试，才是有意义的。在这一系列的视频中我们将介绍具体的诊断法，但我要提前说明一点的是，这些诊断法的执行和实现，是需要花些时间的，有时候确实需要花很多时间来理解和实现，但这样做的确

是把时间用在了刀刃上，因为这些方法让你在开发学习算法时，节省了几个月的时间，因此，在接下来几节课中，我将先来介绍如何评价你的学习算法。在此之后，我将介绍一些诊断法，希望能让你更清楚。在接下来的尝试中，如何选择更有意义的方法。

## 1. 如何选择合适的机器学习算法

- 需要机器学习诊断

## 2. 评估假设

- 如何评估你的假设？
  - 随机选择70%的数据  $\mathbf{x}, \mathbf{y}$  值作为训练集
  - 另外30%的数据作为测试集
- 线性回归训练/测试过程
  - 从训练数据中得到  $\theta$  值 ( $J(\theta)$  最小)
  - 计算测试集是否错误

### - Compute test set error:

$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left( h_{\theta}(x_{test}^{(i)}) - y_{test}^{(i)} \right)^2$$

- logistic回归同理

### - Learn parameter $\theta$ from training data

### - Compute test set error:

$$J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} y_{test}^{(i)} \log h_{\theta}(x_{test}^{(i)}) + (1 - y_{test}^{(i)}) \log (1 - h_{\theta}(x_{test}^{(i)}))$$

### - Misclassification error (0/1 misclassification error):

## 3. 模型选择：training/validation/test sets

- training/test model
  - 首先用，训练集得出  $\Theta^{(i)}$  的值
  - 然后用，测试集得到  $J_{test}(\Theta^{(i)})$  的值，从而得出最好的一组  $h_{\theta}(\mathbf{x})$  方程
  - 最后，当需要知道其普适性时，需要找数据带入  $h_{\theta}(\mathbf{x})$  中验证。如果再用测试集中的数据，所得到的“普适性”就有些作弊嫌疑！是错误的。
- 所以，应当使用 training/validation/test 模型，如下图所示
  - 其和上面模型的区别就是，先用 validation 集选出最好的  $h_{\theta}(\mathbf{x})$ ，方程，然后再用 test 集验证其“普适性”。

## Train/validation/test error

Training error:

$$\rightarrow J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \quad \text{J}(\theta)$$

Cross Validation error:

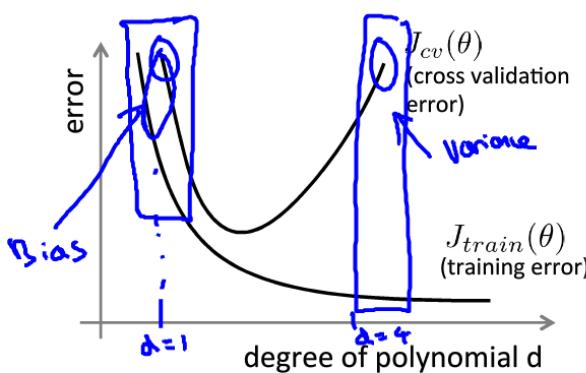
$$\rightarrow J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

Test error:

$$\rightarrow J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

### 4. 诊断偏差与方差

- 误差的来源
  - 偏差
    - 欠拟合: 选取的 $\mathbf{x}$ 数不够多, 多项式不够高
  - 方差
    - 过渡拟合: 高阶项过高, 输入 $\mathbf{x}$ 数量太多
      - 如何识别
      - 偏差Bias (underfit):
    - $J_{train}(\theta) \approx J_{cv}(\theta)$
    - 方差Variance (overfit):
    - $J_{cv}(\theta) \gg J_{train}(\theta)$



Bias (underfit):  
 $\rightarrow J_{train}(\theta) \text{ will be high}$   
 $J_{cv}(\theta) \approx J_{train}(\theta)$

Variance (overfit):  
 $\rightarrow J_{train}(\theta) \text{ will be low}$   
 $J_{cv}(\theta) \gg J_{train}(\theta)$

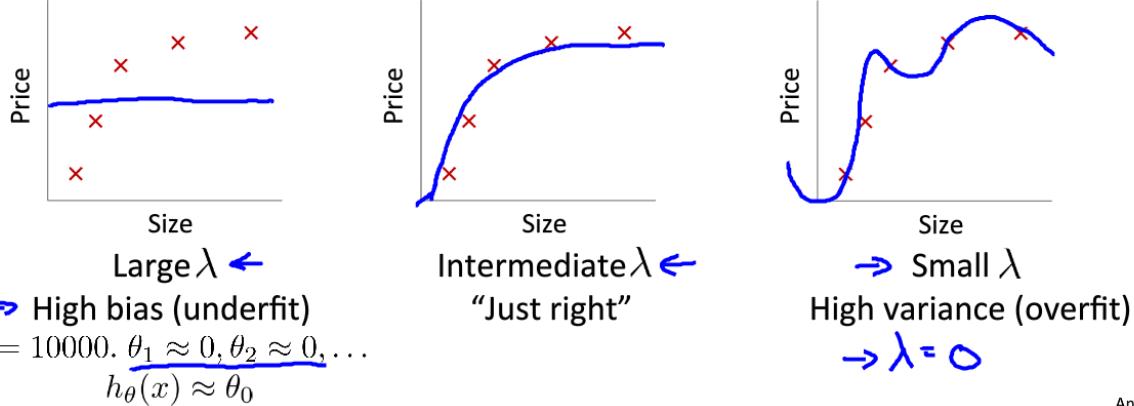
### 5. 正则化和偏差/方差

- 正则化 $\lambda$ 取值大小对于回归模型影响的直观感受

## Linear regression with regularization

Model: 
$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

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



Andrew Ng

- 选择合适的正则参数 $\lambda$ 的步骤

- 测试不同 $\lambda$ 值
- 分别求解 $\min J(\Theta)$  对应的最小 $\Theta^{(i)}$
- 最后求交叉测试误差 $J_{cv}(\Theta^{(i)})$ 大小，选出最小的模型
- 测试 $J_{test}$ 普适性

### Choosing the regularization parameter $\lambda$

Model: 
$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

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

- Try  $\lambda = 0$
  - Try  $\lambda = 0.01$
  - Try  $\lambda = 0.02$
  - Try  $\lambda = 0.04$
  - Try  $\lambda = 0.08$
  - ⋮
  - Try  $\lambda = 10$
- Pick (say)  $\theta^{(5)}$ . Test error:  $J_{test}(\theta^{(5)})$

Andrew Ng

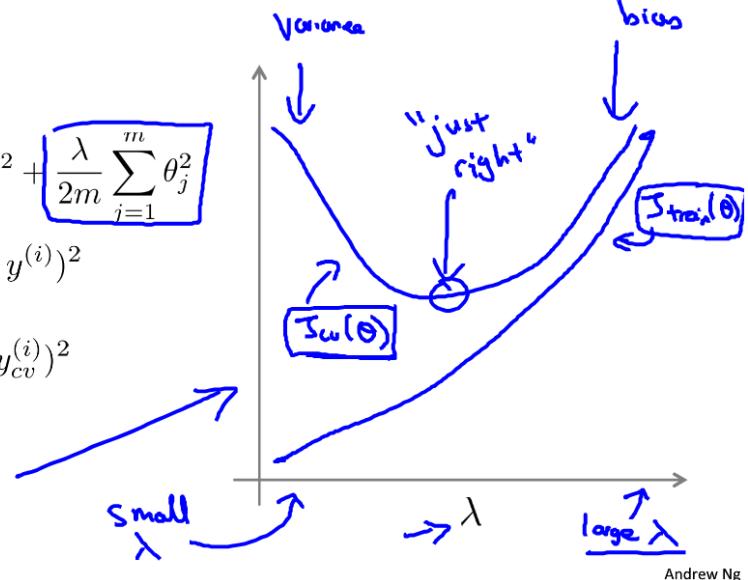
- 找到最合适的 $\lambda$

## Bias/variance as a function of the regularization parameter $\lambda$

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

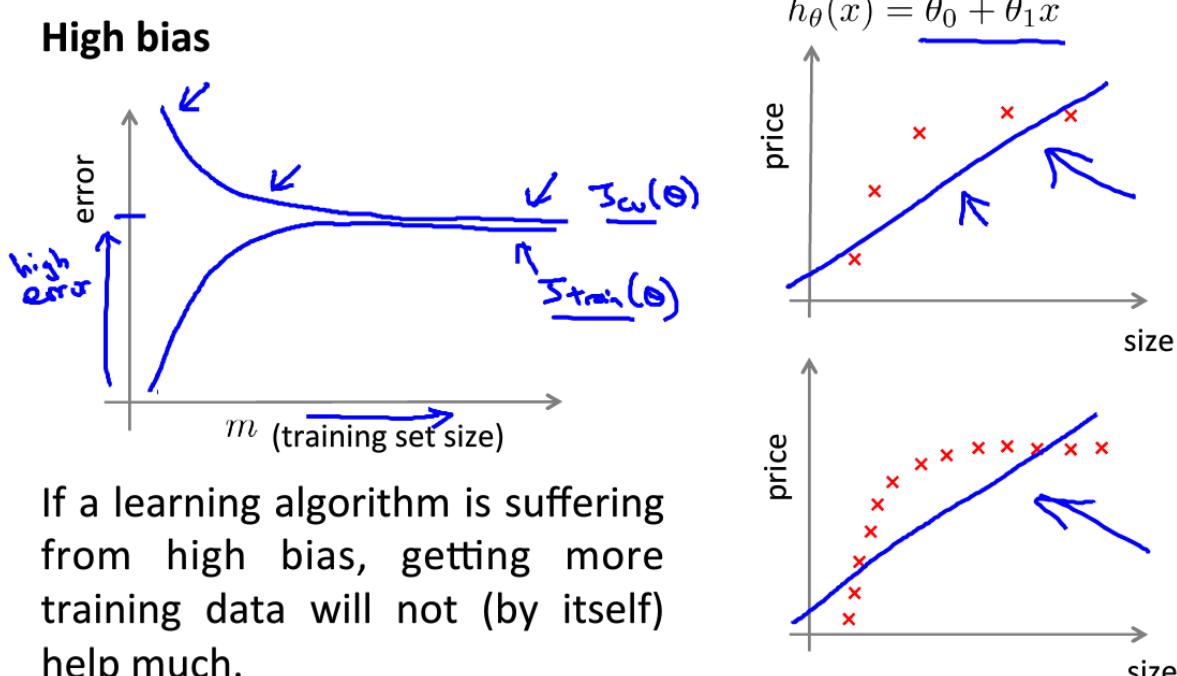
$$\rightarrow J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\rightarrow J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$



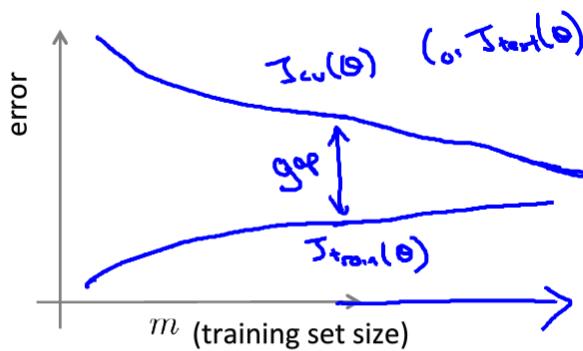
### 6. 学习曲线

- 高偏差模型（欠拟合）
  - 此时增加数据是没有意义的！是公式的问题



- 高方差模型（拟合过渡），公式阶数已经足够。
  - 此时增加数据量是有意义的

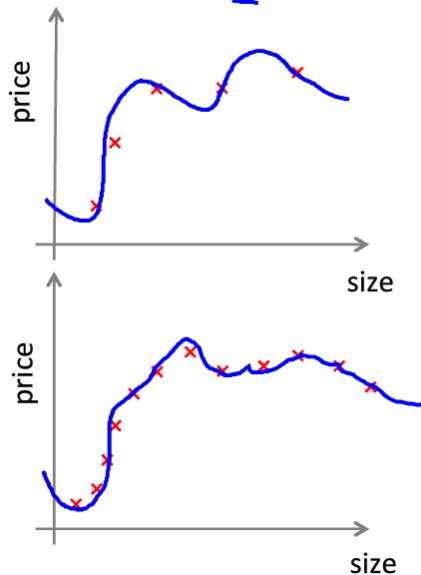
## High variance



If a learning algorithm is suffering from high variance, getting more training data is likely to help. ↫

$$h_{\theta}(x) = \theta_0 + \theta_1 x + \dots + \theta_{100} x^{100}$$

(and small  $\lambda$ )



Andrew Ng

## 7. 决定接下来做什么

- Debugging a learning algorithm
  - 假设你利用了一种学习算法，但是发现它的错误很大，这时候你应该怎么办？
    1. 获得更多的训练数据 → fixes high variance
    2. 尝试减小特征值集合的数量(features) → fixes high variance
    3. 尝试获得更多的特征值 → fixes high bias
    4. 尝试增加polynomial feature( $x_1^2, x_2^2, x_1x_2, etc$ ) → fixes high bias
    5. 尝试减小 $\lambda$  → fixes high bias
    6. 尝试增加 $\lambda$  → fixes high variance
- 神经网络，与过度拟合
  - 过度拟合通常发生在隐藏单元(hiden unit)或者隐藏层(hiden layer)较多的时候
  - 解决的办法应当是加入 $\lambda$  项正则化

## 12. 机器学习系统设计

1. 确定执行的优先级
  - 当面对机器学习模型时，有很多方法可以用来优化，如增加数据量，增加feature数，等等
2. 误差分析
  - 建议方法

## Recommended approach

- Start with a simple algorithm that you can implement quickly. Implement it and test it on your cross-validation data.
- Plot learning curves to decide if more data, more features, etc. are likely to help.
- Error analysis: Manually examine the examples (in cross validation set) that your algorithm made errors on. See if you spot any systematic trend in what type of examples it is making errors on.

- 尽可能快的开始一项任务，用最简单的算法，在24小时内完成
- 画出学习曲线，从而确定是否需要增加数据，feature等
- 误差分析：人工检查被错误分类的数据，是否有系统性的共同点，什么样的数据会容易被错误识别。

- 误差分析

- 手动检查100个错误，识别类型
  1. email的种类（短语，假手表，偷密码等）
  2. 你认为哪种feature的增加可以帮助你优化算法

## Error Analysis

$m_{CV} = 500$  examples in cross validation set

Algorithm misclassifies 100 emails.

Manually examine the 100 errors, and categorize them based on:

- (i) What type of email it is pharma, replica, steal passwords, ...
- (ii) What cues (features) you think would have helped the algorithm classify them correctly.

Pharma: 12	→ Deliberate misspellings: 5
Replica/fake: 4	(m0rgage, med1cine, etc.)
→ Steal passwords: 53	→ Unusual email routing: 16
Other: 31	→ Unusual (spamming) punctuation: 32

- 数值分析的重要性

- 比如确定discounts/discount/discounting等是否为相同词（词干）
- 定量描述，加入词干识别和不加入词干识别优化了多少？5%error, 4%error等。  
尽可能快的写出一种算法，然后再慢慢修正它！！！

- 不对称分类的误差评估

举个例子：假如我们做癌症分析，最后得出该算法只有1%的误差，也就是说准确率达到了99%。这样看起来99%算是非常高的了，但是我们发现在训练集里面只有0.5%的患者患有癌症，那么这1%的错误率就变得那么准确了（比如你可以全部预测没有癌症，那准确率就直接达到99.5%了），那还预测个毛线！

## Cancer classification example

Train logistic regression model  $h_\theta(x)$ .  $y = 1$  if cancer,  $y = 0$  otherwise)

Find that you got 1% error on test set.  
(99% correct diagnoses)

Only 0.50% of patients have cancer.

skewed classes.  
function y = predictCancer(x)  
→  $y = 0$ ; %ignore x!  
return

0.5% error  
→ 99.2% away (0.8% error)  
→ 99.5% away (0.5% error)

- 查准率 (Precision), 召回率 (Recall)

- 查准率：预测有癌症的人中，有多少真的有癌症？查准率高说明，如果医生说你有癌症，你就真的有癌症了。
- 召回率：对于已经得癌症的人，你有多大的可能可以把他们预测出来！召回率越高越好。（这样的话，如果你直接认为所人都没有癌症，虽然你的准确率很高，但是召回率等于0，就太低了）

### Precision/Recall

$y = 1$  in presence of rare class that we want to detect

		Actual class	
		1	0
Predicted 1 class	1	True positive	False positive
	0	False negative	True negative

→ Precision  
(Of all patients where we predicted  $y = 1$ , what fraction actually has cancer?)

$$\frac{\text{True positives}}{\#\text{predicted positive}} = \frac{\text{True positive}}{\text{True pos} + \text{False pos}}$$

→ Recall  
(Of all patients that actually have cancer, what fraction did we correctly detect as having cancer?)

$$\frac{\text{True positives}}{\#\text{actual positives}} = \frac{\text{True positives}}{\text{True pos} + \text{False neg}}$$

Andrew Ng

- 如何用一个值来评估算法的优劣？

- 准确率和召回率大多数情况下不可以同时很高，有相互制衡的趋势

- 理解：比如我想要查准率很高，那么我就要尽可能保证我预测出有癌症 ( $y=1$ ) 的情况是100%准确的，那么我就有尽量减少筛选范围（提高筛选标准）的趋势，这导致的结果就是有一部分虽然有癌症但是没有被筛选出来，即召回率低）

- 当然最理想的情况是查准率和召回率都为1

## Trading off precision and recall

→ Logistic regression:  $0 \leq h_\theta(x) \leq 1$

Predict 1 if  $h_\theta(x) \geq 0.3$  ↗ ↘ 0.9 0.3 ↙ ↖

Predict 0 if  $h_\theta(x) < 0.3$  ↗ ↘ 0.9 0.3 ↙ ↖

→ Suppose we want to predict  $y = 1$  (cancer) only if very confident.

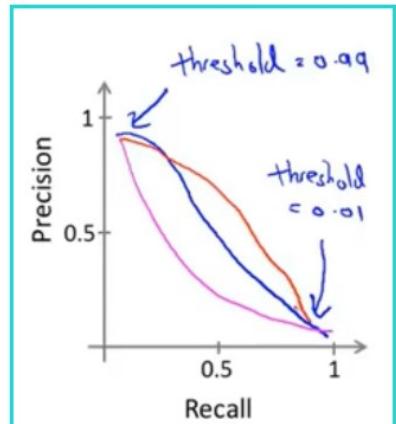
→ Higher precision, lower recall.

→ Suppose we want to avoid missing too many cases of cancer (avoid false negatives).

→ Higher recall, lower precision.

$$\rightarrow \text{precision} = \frac{\text{true positives}}{\text{no. of predicted positive}}$$

$$\rightarrow \text{recall} = \frac{\text{true positives}}{\text{no. of actual positive}}$$



More generally: Predict 1 if  $h_\theta(x) \geq \text{threshold}$ .

Andrew Ng

- 通常采用F1值作为判据：

$$F_1 \text{ Score} = 2 \frac{PR}{P+R}$$

P 指的是 Precision, R 指的是 Recall。

**F1 Score** 的好处就是可以筛除几种特例，比如准确率或召回率为1的情况，他们的平均值都很高，但是瞎预测

## F<sub>1</sub> Score (F score)

How to compare precision/recall numbers?

	Precision(P)	Recall (R)	Average	F <sub>1</sub> Score
→ Algorithm 1	0.5	0.4	0.45	0.444 ↙
→ Algorithm 2	0.7	0.1	0.4	0.175 ↙
Algorithm 3	0.02	1.0 ↙	0.51	0.0392 ↙
Average: <del><math>\frac{P+R}{2}</math></del>				Predict y=1 all the time
$F_1 \text{ Score}: 2 \frac{PR}{P+R}$				$P=0 \text{ or } R=0 \Rightarrow F\text{-score} = 0.$ ↗ $P=1 \text{ and } R=1 \Rightarrow F\text{-score} = 1$ ↗

Andrew Ng

- 补充：

- F1的命名并没有什么意义
- F1值可以作为精确率召回率的权衡值，但是判据标准还需要进一步调节，每一次调节得到一组F1值，最终优化到最大的值。

## 13. 支持向量机SVM

应用于解决复杂非线性方程

到目前为止，你已经见过一系列不同的学习算法。在监督学习中，许多学习算法的性能都非常类似，因此，重要的不是你该选择使用学习算法A还是学习算法B，而更重要的是，应用这些算法时，所创建的大量数据在应用这些算法时，表现情况通常依赖于你的水平。比如：你为学习算法所设计的特征量的选择，以及如何选择正则化参数，诸如此类的事。还有一个更加强大的算法广泛的应用于工业界和学术界，它被称为支持向量机(Support Vector Machine)。与逻辑回归和神经网络相比，支持向量机，或者简称SVM，在学习复杂的非线性方程时提供了一种更为清晰，更加强大的方式。因此，在接下来的视频中，我会探讨这一算法。在稍后的课程中，我也会对监督学习算法进行简要的总结。当然，仅仅是作简要描述。但对于支持向量机，鉴于该算法的强大和受欢迎度，在本课中，我会花许多时间来讲解它。它也是我们所介绍的最后一个监督学习算法。

正如我们之前开发的学习算法，我们从优化目标开始。那么，我们开始学习这个算法。为了描述支持向量机，事实上，我将会从逻辑回归开始展示我们如何一点一点修改来得到本质上的支持向量机。

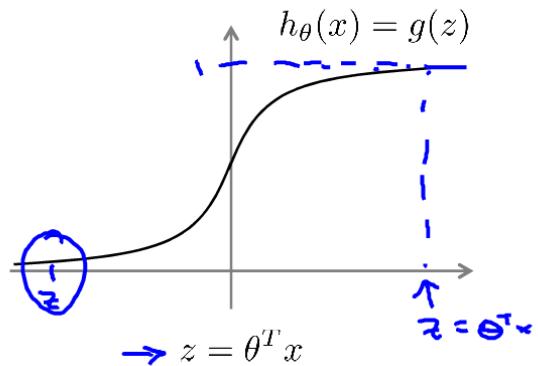
## 1. 优化目标

- 换个角度看Logistic 回归

◦ 理解  $h_{\theta}(x)$

## Alternative view of logistic regression

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



If  $y = 1$ , we want  $h_{\theta}(x) \approx 1$      $\theta^T x \gg 0$   
 If  $y = 0$ , we want  $h_{\theta}(x) \approx 0$      $\theta^T x \ll 0$

- 将  $h_{\theta}(x)$  带入 Cost function

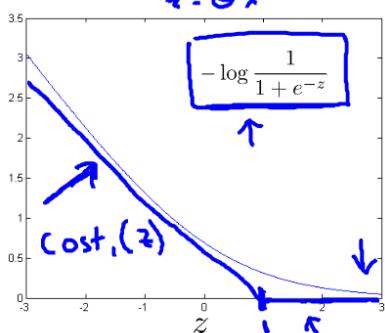
▪ 理解  $-\log \frac{1}{1+e^{-z}}$  的图像

## Alternative view of logistic regression

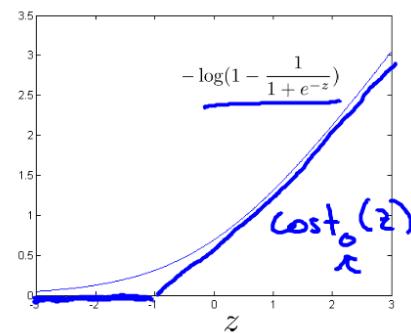
Cost of example:  $-(y \log h_{\theta}(x) + (1 - y) \log(1 - h_{\theta}(x)))$

$$= \boxed{-y \log \frac{1}{1 + e^{-\theta^T x}}} - \boxed{(1 - y) \log(1 - \frac{1}{1 + e^{-\theta^T x}})}$$

If  $y = 1$  (want  $\theta^T x \gg 0$ ):



If  $y = 0$  (want  $\theta^T x \ll 0$ ):



- 支持向量机
  - 将前面的负号放到 $\log$ 函数里面去
  - 简化Logistic回归方程为

$$\begin{cases} A + \lambda \times B \\ CA + B \end{cases}$$

- 其中C可以看作和 $\lambda$ 互为倒数关系
- 省略掉m的值，因为是常数省略对结果无影响

## Support vector machine

Logistic regression:

$$\min_{\theta} \frac{1}{m} \left[ \sum_{i=1}^m y^{(i)} \underbrace{\left( -\log h_{\theta}(x^{(i)}) \right)}_{cost_1(\theta^T x^{(i)})} + (1-y^{(i)}) \underbrace{\left( -\log(1-h_{\theta}(x^{(i)})) \right)}_{cost_0(\theta^T x^{(i)})} \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

Support vector machine:

$$\min_{\theta} C \sum_{i=1}^m y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) cost_0(\theta^T x^{(i)}) + \frac{1}{2} \sum_{j=0}^n \theta_j^2$$

$$\min_u (u-5)^2 + 1 \rightarrow u=5$$

$$\min_u 10(u-5)^2 + 10 \rightarrow u=5$$

$$A + \lambda B \Leftarrow C = \frac{1}{\lambda}$$

$$\rightarrow \min_{\theta} C \sum_{i=1}^m \left[ y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^n \theta_j^2$$

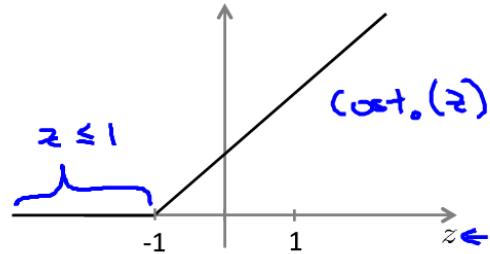
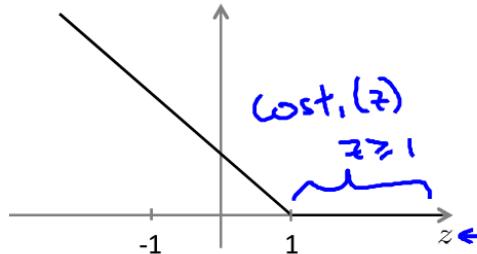
Andrew Ng

## 2. 直观上对大间隔的理解

- 从数学公式上理解大间隔
  - if  $y=1$ , 起作用的就是只有公式左边部分 (右边为0) , 此时如果我们想让 $cost_1(z)=0$ , 就需要 $\theta^T x \geq 1$
  - 同上, if  $y=0$ ,  $cost_0(z)=0$ , 就需要 $\theta^T x \leq -1$
  - 这样就在上述两者中间形成了一个范围

## Support Vector Machine

$$\rightarrow \min_{\theta} C \sum_{i=1}^m \left[ y^{(i)} \underbrace{cost_1(\theta^T x^{(i)})}_{z \geq 1} + (1-y^{(i)}) \underbrace{cost_0(\theta^T x^{(i)})}_{z \leq -1} \right] + \frac{1}{2} \sum_{i=1}^n \theta_j^2$$



→ If  $y=1$ , we want  $\theta^T x \geq 1$  (not just  $\geq 0$ )

→ If  $y=0$ , we want  $\theta^T x \leq -1$  (not just  $< 0$ )

$\theta^T x \geq 1$

$\theta^T x \leq -1$

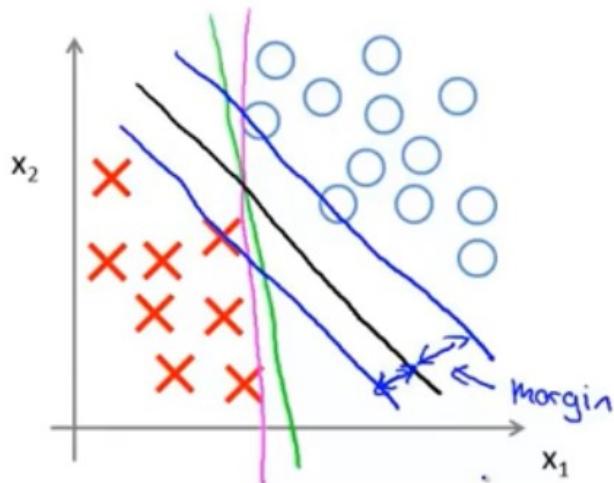
$$C = 100,000$$

## • 从图像上来理解大间隔

- 绿色和蓝色的线代表Logistic回归函数得到的曲线
- 蓝色中间的代表向量机得到的曲线

- 补充：如果  $x_0$  的分布不是这么清晰时，如果  $C$  取得很大的话，向量机得到的线就会随之改变，但是  $C$  取值不是很大的话，部分不规则的排列可以忽略。（ $C$  取值大，对应着  $\lambda$  的取值小，容易过渡拟合）

## SVM Decision Boundary: Linearly separable case



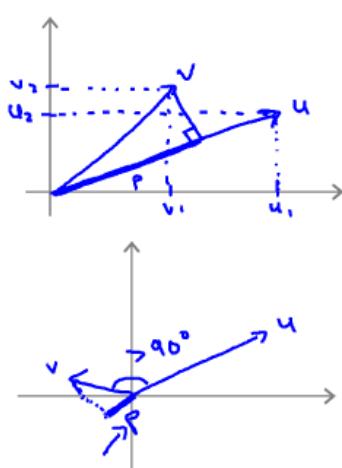
Large margin classifier

### 3. 大间隔分类器的数学原理

- 向量内积

- 两个向量内积， $\vec{u} \cdot \vec{v} = u^T v = v \cdot \cos(\theta) \cdot \|u\| = \rho \cdot \|u\| = u_1 v_1 + u_2 v_2$

#### Vector Inner Product



$$\rightarrow u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \rightarrow v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$\begin{aligned} u^T v &= ? & [u, v] &= \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix} \\ \|u\| &= \text{length of vector } u \\ &= \sqrt{u_1^2 + u_2^2} \in \mathbb{R} \\ p &= \text{length of projection of } v \text{ onto } u. \\ u^T v &= p \cdot \|u\| \leftarrow & = v^T u \\ \text{Signed} &= u_1 v_1 + u_2 v_2 \leftarrow & p \in \mathbb{R} \end{aligned}$$

$$u^T v = p \cdot \|u\|$$

$$p < 0$$

Andrew Ng

- SVM 边界，目的是得到  $\min \frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} \|\theta\|^2$  的最小值，因为此时  $\theta^T x^{(i)} \geq 1 (\leq -1)$  已经把前面一项的最小化  $J=0$  安排妥当了，剩下的目标就是要求最后一项  $\frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} \|\theta\|^2$  最小

- 下面是边界条件

- $\theta^T x^{(i)} \geq 1 \text{ if } y^{(i)} = 1$
- $\theta^T x^{(i)} \leq -1 \text{ if } y^{(i)} = 0$
- 同理上面的  $\theta^T x^{(i)}$  可以简化为  $\theta^T x^{(i)} = \rho^{(i)} \|\theta\| = \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)}$

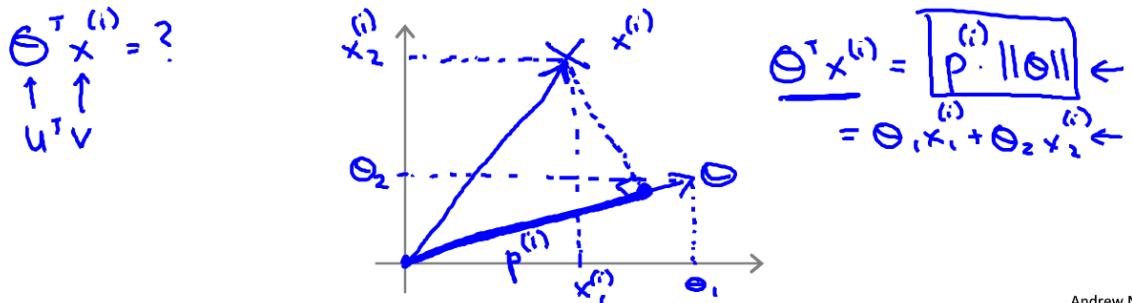
## SVM Decision Boundary

$$\omega = (\sqrt{\omega})'$$

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} (\Theta_1^2 + \Theta_2^2) = \frac{1}{2} (\underbrace{\Theta_1^2 + \Theta_2^2}_{= \|\theta\|^2})^2 = \frac{1}{2} \|\theta\|^2$$

s.t.  $\theta^T x^{(i)} \geq 1$  if  $y^{(i)} = 1$   
 $\theta^T x^{(i)} \leq -1$  if  $y^{(i)} = 0$

Simplification:  $\Theta_0 = 0$ ,  $n=2$



Andrew Ng

- 利用边界条件确定直线

- 假设  $\theta_0 = 0$ , 不等于0也可以
- 左图为例，每一个x值在  $\vec{\theta}$  上的投影  $p$  都很小，这样导致  $\|\theta\|$  值必须取得很大，这就不利于上面的  $\min \frac{1}{2} \sum_{j=1}^n \theta_j^2$  了
- 右图为例，当每一个x的取值在  $\vec{\theta}$  上的投影  $p$  都很大时， $\|\theta\|$  值就会变小，满足  $\min \frac{1}{2} \sum_{j=1}^n \theta_j^2$  要求
- 以上就是向量机大间隔的数学原理

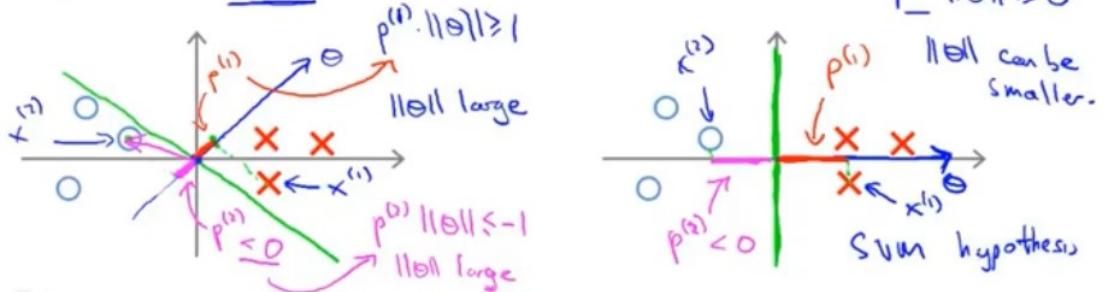
## SVM Decision Boundary

$$\rightarrow \min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} \|\theta\|^2 \leftarrow$$

s.t.  $p^{(i)} \cdot \|\theta\| \geq 1$  if  $y^{(i)} = 1$   
 $p^{(i)} \cdot \|\theta\| \leq -1$  if  $y^{(i)} = -1$

where  $p^{(i)}$  is the projection of  $x^{(i)}$  onto the vector  $\theta$ .

Simplification:  $\theta_0 = 0$



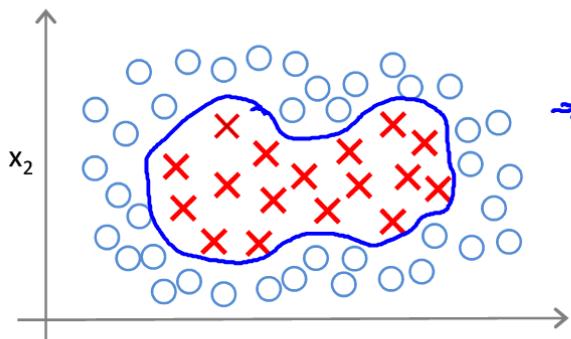
Andrew Ng

## 4. 核(kernel)函数

- 非线性决策边界

- 假设预测  $y=1$  的函数如下，如何更好地选择 feature 是一个问题？

## Non-linear Decision Boundary



Predict  $y = 1$  if

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \dots \geq 0$$

$$h_0(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \dots \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

$$\rightarrow \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \dots$$

$$f_1 = x_1, f_2 = x_2, f_3 = x_1 x_2, f_4 = x_1^2, f_5 = x_2^2, \dots$$

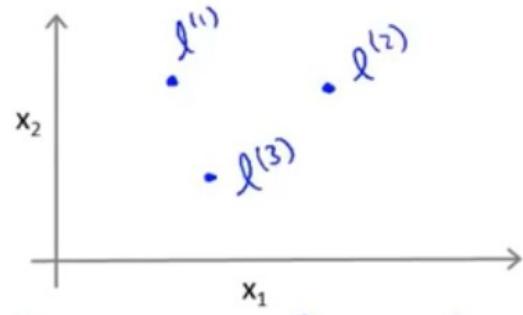
Is there a different / better choice of the features  $f_1, f_2, f_3, \dots$ ?

Andrew Ng

- kernel是什么？

◦ 高斯kernel，此处假设，feature只有 $l^1, l^2, l^3$ 定义 $f_i = \text{similarity}(x, l^{(i)}) = \exp\left(-\frac{\|x-l^{(i)}\|^2}{2\sigma^2}\right)$ ，其中指数部分就为高斯kernel。注：这个函数与正态分布没什么实际上的关系，只是看上去像而已

## Kernel



Given  $x$ , compute new feature depending on proximity to landmarks  $l^{(1)}, l^{(2)}, l^{(3)}$

Given  $x$ :

$$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x-l^{(1)}\|^2}{2\sigma^2}\right)$$

$$f_2 = \text{similarity}(x, l^{(2)}) = \exp\left(-\frac{\|x-l^{(2)}\|^2}{2\sigma^2}\right)$$

$$f_3 = \text{similarity}(x, l^{(3)}) = \exp(\dots)$$

$\nwarrow$  kernel (Gaussian kernels)

$$k(x, l^{(1)})$$

- 高斯kernel的合理性？

◦ 通过改变 $x$ 的值，使之≈或者距离 $l$ 很远得到不同的 $f$ 值，三个 $l$ 值可以得到三个对应的 $f$ 值

- if  $x \approx l^{(1)}$ :  $f_1 \approx 1$
- if  $x$  far from  $l^{(1)}$ :  $f_1 \approx 0$

## Kernels and Similarity

$$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$$

↓ ↓

If  $x \approx l^{(1)}$  :

$$f_1 \underset{\uparrow}{\approx} \exp\left(-\frac{\sigma^2}{2\sigma^2}\right) \approx 1$$

$l^{(1)} \rightarrow f_1$   
 $l^{(2)} \rightarrow f_2$   
 $l^{(3)} \rightarrow f_3$

If  $x$  if far from  $l^{(1)}$  :

$$f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0.$$

↑ ↑

○ 图像中说明：

- $\sigma$ 的取值对于高斯kernel图像的影响，越小y=1的范围越小
- 当x的取值距离中心 $l$ 点距离近时 $f_1=1$ ，反之 $f_1=0$

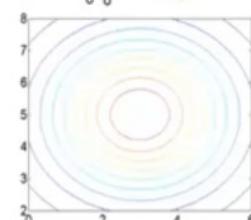
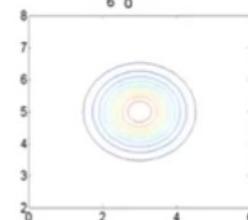
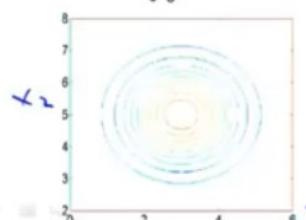
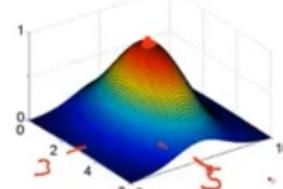
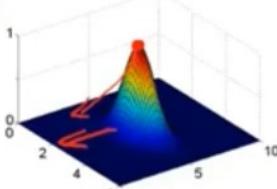
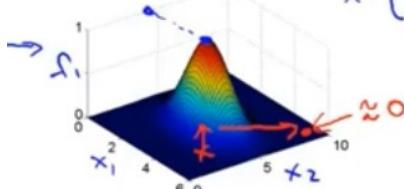
**Example:**

$$\rightarrow l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \quad f_1 = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$$

$$\rightarrow \sigma^2 = 1$$

$$x = \begin{bmatrix} 3 \\ 5 \end{bmatrix} \quad \sigma^2 = 0.5$$

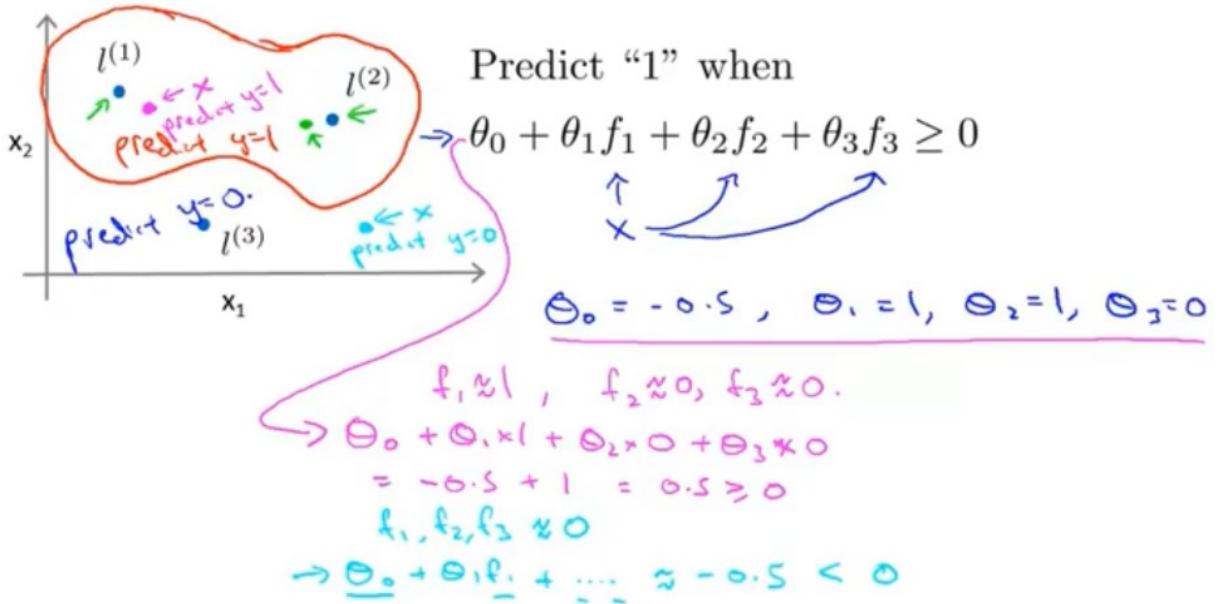
$$\sigma^2 = 3$$



Andrew Ng

○ 回到假设的预测方程中，距离feature  $l_{(i)}$  越近是有可能满足方程， $y=1$ 。

在下图中，当样本处于洋红色的点位置处，因为其离 $l^{(1)}$ 更近，但是离 $l^{(2)}$ 和 $l^{(3)}$ 较远，因此 $f_1$ 接近1，而 $f_2, f_3$ ，接近0。因此  $h_\theta(x) = \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 > 0$ 。因此预测 $y=1$ 。同理可以求出，对于离 $l^{(2)}$ 较近的绿色点，也预测 $y=1$ ，但是对于蓝绿色的点，因为其离三个地标都较远，预测 $y=0$ 。这样对于  $\theta_0 = -0.5, \theta_1 = 1, \theta_2 = 1, \theta_3 = 0$  的例子，就可以得到不同取点时候  $h_\theta$  的取值。



这样，图中红色的封闭曲线所表示的范围，便是我们依据一个单一的训练样本和我们选取的地地标所得出的判定边界，在预测时，我们采用的特征不是训练样本本身的特征，而是通过核函数计算出的新特征 $f_1, f_2, f_3$ 。

## 5. 核函数2

- 如何得到上面的feature  $l^{(1)}, l^{(2)}, l^{(3)} \dots l^{(i)}$  呢？
- 梳理SVM with kernel的计算方法
  - 定义输入输出  $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$ .
  - 选择 $x$ 对应 $l$ :  $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)}$ . 即选择这些点为feature点
  - 接下来对于example  $x$ 可以得到一系列的 $f_m$ 值:
    - $f_1 = \text{similarity}(x, L^{(1)})$
    - $f_2 = \text{similarity}(x, l^{(2)})$
    - ...
- 那么对于训练集 $(x^{(i)}, y^{(i)})$ 来说: 可以得到一系列f值 (f矩阵 $m \times 1$ )

For training example  $(x^{(i)}, y^{(i)})$ :

$$\underline{x^{(i)}} \rightarrow \begin{cases} f_1^{(i)} = \sin(x^{(i)}, l^{(1)}) \\ f_2^{(i)} = \sin(x^{(i)}, l^{(2)}) \\ \vdots \\ f_m^{(i)} = \sin(x^{(i)}, l^{(m)}) \end{cases}$$

$$x^{(i)} \in \mathbb{R}^{n+1} \quad \begin{cases} f^{(i)} = \sin(x^{(i)}, l^{(1)}) \\ f_0^{(i)} = 1 \end{cases} \quad \text{(or } \mathbb{R}^m \text{)}$$

Andrew Ng

- 将上述f值继续作如下处理
  - $m+1$ 个标记点包块 $f_0$
  - $m$ 为标记点个数=f值的个数
  - 将f带入最小化公式如下:

## SVM with Kernels

Hypothesis: Given  $\underline{x}$ , compute features  $\underline{f} \in \mathbb{R}^{m+1}$

$$\rightarrow \text{Predict "y=1" if } \theta^T \underline{f} \geq 0$$

$\Theta \in \mathbb{R}^{n+1}$

Training:

$$\rightarrow \min_{\theta} C \sum_{i=1}^m y^{(i)} \text{cost}_1(\theta^T \underline{f}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T \underline{f}^{(i)}) + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

$n = m$   
 $\rightarrow \Theta_0$

$\rightarrow \left[ \begin{array}{l} - \sum_j \theta_j \\ - \end{array} \right] = \Theta^T \Theta \quad \Theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_m \end{bmatrix} \quad (\text{ignoring } \theta_0)$

$M = 10,000$

Andrew Ng

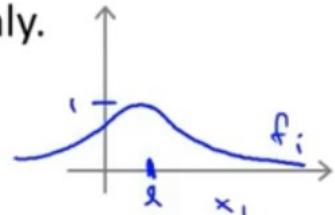
- 这里有些太复杂了！ $\theta$ 的值需要优化， $f$ 的值通过高斯关系由 $\text{sim}(\underline{x}, \underline{l})$ 得到， $\theta^T f > 0$ 的条件下 $y=1$ ，而且需要满足向量机 $\theta^T f \geq 1 \text{ or } \leq -1$ 的条件，此外还有 $\min \sum \|\theta\|$ 的条件需要满足，以上这三个条件，都要用什么样的数学方式来实现？好复杂
- 以上就是向量机学习算法的数学部分
- 就像不建议自己写程序计算矩阵的逆一样，这里也不建议自己写程序计算 $\min$ 函数，求解 $\theta$
- 方差的影响

## SVM parameters:

$C ( = \frac{1}{\lambda} )$ .  $\rightarrow$  Large C: Lower bias, high variance. (small  $\lambda$ )  
 $\rightarrow$  Small C: Higher bias, low variance. (large  $\lambda$ )

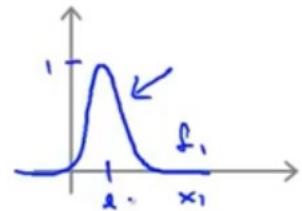
$\sigma^2$   $\rightarrow$  Large  $\sigma^2$ : Features  $f_i$  vary more smoothly.  
 $\rightarrow$  Higher bias, lower variance.

$$\exp\left(-\frac{\|\underline{x} - \underline{l}^{(i)}\|^2}{2\sigma^2}\right)$$



Small  $\sigma^2$ : Features  $f_i$  vary less smoothly.

Lower bias, higher variance.



Andrew Ng

## 6. 使用SVM

### 前言

目前为止，我们已经讨论了SVM比较抽象的层面，在这个视频中我将要讨论到为了运行或者运用SVM。你实际上所需要的一些东西：支持向量机算法，提出了一个特别优化的问题。但是就如在之前的视频中我简单提到的，我真的不建议你自己写软件来求解参数，因此由于今天我们中的很少人，或者其实没有人考虑过自己写代码来转换矩阵，或求一个数的平方根等我们只是知道如何去调用库函数来实现这些功能。同样的，用以解决SVM最优化问题的软件很复杂，且已经有研究者做了很多年数值优化了。因此你提出好的

软件库和好的软件包来做这样一些事儿。然后强烈建议使用高优化软件库中的一个，而不是尝试自己落实一些数据。有许多好的软件库，我正好用得最多的两个是liblinear和libsvm，但是真的有很多软件库可以用来做这件事儿。你可以连接许多你可能会用来编写学习算法的主要编程语言。

- 已经有足够good软件库(eg. liblinear, libsvm,...)去求解 $\theta$ ，但是仍需要注意以下问题：

Need to specify:

→ Choice of parameter C.

Choice of kernel (similarity function):

E.g. No kernel ("linear kernel")

Predict "y = 1" if  $\underline{\theta^T x} \geq 0$

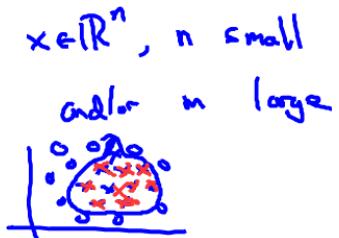
$$\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n \geq 0 \quad \underline{x \in \mathbb{R}^{n+1}}$$

$\rightarrow$  n large, m small

→ Gaussian kernel:

$$f_i = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right), \text{ where } l^{(i)} = x^{(i)}$$

Need to choose  $\frac{\sigma^2}{\pi}$ .



- 高斯kernel的使用形式

- function

- Note: 在做高斯kernel计算之前需要消减一下feature的大小

- 比如 $x_1$ 代表房子的大小,  $x_2$ 代表几件房间等

- 那么房子大小的平方就可能占非常大的比重, 所以要降低一下他的大小 (norm一下)

Kernel (similarity) functions:

$f_i$  function  $f = \text{kernel}(x_1, x_2)$

$$f = \exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right)$$

return

$$\begin{matrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{matrix}$$

→ Note: Do perform feature scaling before using the Gaussian kernel.

$$\begin{aligned} & \rightarrow \|x - l\|^2 \\ & \rightarrow \|v\|^2 = v_1^2 + v_2^2 + \dots + v_n^2 \\ & \quad = \underbrace{(x_1 - l_1)^2}_{1000 \text{ feet}^2} + \underbrace{(x_2 - l_2)^2}_{1-5 \text{ bedrooms}} + \dots + (x_n - l_n)^2 \end{aligned}$$

- 其他kernel的选择, 线性kernel和高斯kernel是最常用的两种svm kernel

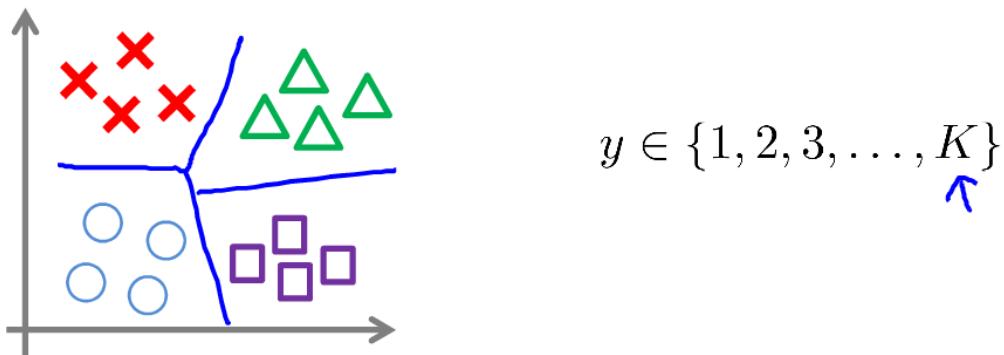
## Other choices of kernel

Note: Not all similarity functions  $\text{similarity}(x, l)$  make valid kernels.  
 (Need to satisfy technical condition called “Mercer’s Theorem” to make sure SVM packages’ optimizations run correctly, and do not diverge).

- Many off-the-shelf kernels available:
- Polynomial kernel:  $k(x, l) = \frac{(x^T l + \text{constant})}{\text{degree}}$
  - More esoteric: String kernel, chi-square kernel, histogram intersection kernel, ...  
 $\text{sim}(x, l)$

- 多种分类的方法在软件包中也都包含

## Multi-class classification



Many SVM packages already have built-in multi-class classification functionality.

Otherwise, use one-vs.-all method. (Train  $K$  SVMs, one to distinguish  $y = i$  from the rest, for  $i = 1, 2, \dots, K$ ), get  $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(K)}$   
 Pick class  $i$  with largest  $\underline{(\theta^{(i)})^T x}$

- Logistic vs. SVMs

## Logistic regression vs. SVMs

$n$  = number of features ( $x \in \mathbb{R}^{n+1}$ ),  $m$  = number of training examples

→ If  $n$  is large (relative to  $m$ ): (e.g.  $n \geq m$ ,  $n = 10,000$ ,  $m = 10 \dots 1000$ )

→ Use logistic regression, or SVM without a kernel ("linear kernel")

→ If  $n$  is small,  $m$  is intermediate: ( $n = 1 - 1000$ ,  $m = 10 - 10,000$ )

→ Use SVM with Gaussian kernel

If  $n$  is small,  $m$  is large: ( $n = 1 - 1000$ ,  $m = 50,000 +$ )

→ Create/add more features, then use logistic regression or SVM without a kernel

→ Neural network likely to work well for most of these settings, but may be slower to train.

- 通常without kernel 和 logistic regression是非常相似的两种算法。



值得一提的是，神经网络在以上三种情况下都可能会有较好的表现，但是训练神经网络可能非常慢，选择支持向量机的原因主要在于它的代价函数是凸函数，不存在局部最小值。

今天的SVM会工作得很好，但是它们仍然会有一些慢。当你有非常非常大的训练集，且用高斯核函数是在这种情况下，我经常会做的是尝试手动地创建，拥有更多的特征变量，然后用逻辑回归或者不带核函数的支持向量机。如果你看到这个幻灯片，看到了逻辑回归，或者不带核函数的支持向量机。在这个两个地方，我把它们放在一起是有原因的。原因是：逻辑回归和不带核函数的支持向量机它们都是非常相似的算法，不管是逻辑回归还是不带核函数的SVM，通常都会做相似的事情，并给出相似的结果。但是根据你实现的情况，其中一个可能会比另一个更加有效。但是在其中一个算法应用的地方，逻辑回归或不带核函数的SVM另一个也很有可能很有效。但是随着SVM的复杂度增加，当你使用不同的内核函数来学习复杂的非线性函数时，这个体系，你知道的，当你有多达1万(10,000)的样本时，也可能是5万(50,000)，你的特征变量的数量这是相当大的。那是一个非常常见的体系，也许在这个体系里，不带核函数的支持向量机就会表现得相当突出。你可以做比这困难得多需要逻辑回归的事情。

最后，神经网络使用于什么时候呢？对于所有的这些问题，对于所有的这些不同体系一个设计得很好的神经网络也很有可能会非常有效。有一个缺点是，或者说是有时可能不会使用神经网络的原因是：对于许多这样的问题，神经网络训练起来可能会特别慢，但是如果你有一个非常好的SVM实现包，它可能会运行得比较快比神经网络快很多，尽管我们在此之前没有展示，但是事实证明，SVM具有的优化问题，是一种凸优化问题。因此，好的SVM优化软件包总是会找到全局最小值，或者接近它的值。对于SVM你不需要担心局部最优。在实际应用中，局部最优不是神经网络所需要解决的一个重大问题，所以这是你在使用SVM的时候不需要太去担心的一个问题。根据你的问题，神经网络可能会比SVM慢，尤其是在这样一个体系中，至于这里给出的参考，看上去有些模糊，如果你在考虑一些问题，这些参考会有一些模糊，但是我仍然不能完全确定，我是该用这个算法还是改用那个算法，这个没有太大关系，当我遇到机器学习问题的时候，有时它确实不清楚这是否是最好的算法，但是就如在之前的视频中看到的算法确实很重要。但是通常更加重要的是：你有多少数据，你有多熟练是否擅长做误差分析和排除学习算法，指出如何设定新的特征变量和找出其他能决定你学习算法的变量等方面，通常这些方面会比你使用逻辑回归还是SVM这方面更加重要。但是，已经说过了，SVM仍然被广泛认为是一种最强大的学习算法，这是一个体系，包含了什么时候一个有效的方法去学习复杂的非线性函数。因此，实际上与逻辑回归、神经网络、SVM一起使用这些方法来提高学习算法，我认为你会很好地建立很有技术的状态。（编者注：当时GPU计算比较慢，神经网络还不流行。）

机器学习系统对于一个宽泛的应用领域来说，这是另一个在你军械库里非常强大的工具，你可以把它应用到很多地方，如硅谷、在工业、学术等领域建立许多高性能的机器学习系统。

来源：<http://www.ai-start.com/ml2014/html/week7.html>

## 14. 无监督学习

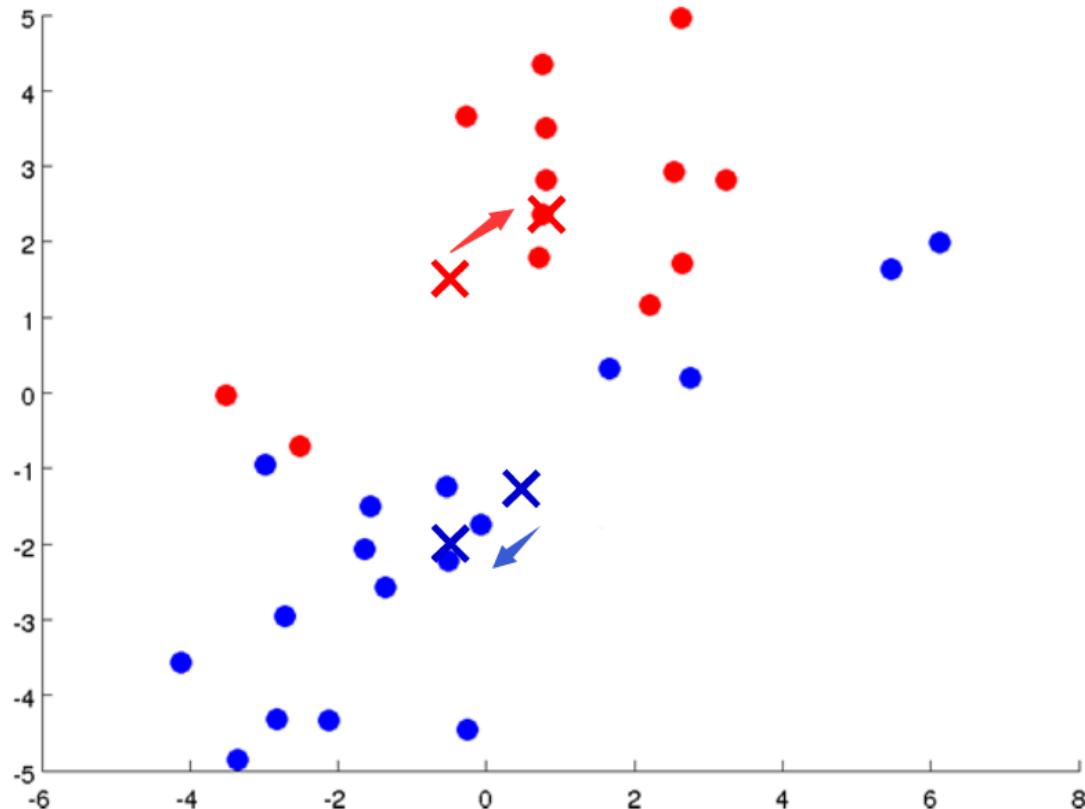
### 1. 无监督学习

- 碱度学习无监督的比较

- 监督学习是有标签的数据
- 无监督是无标签的数据
- 聚类算法 (clustering algo)
  - 举例
    - 市场分割
    - 邮件分类
    - 重新组织计算集群
    - 了解分析星系的组成

## 2. K-Means算法

- 了解K-Means算法 (迭代算法)
  - a. 簇分配
    - 设定红色, 蓝色中心, 然后将数据按照距离远近进行分类
  - b. 移动聚类中心
    - 计算不同簇数据的平均值, 然后更新簇中心的位置为平均值
  - 重复a, b步骤



- 什么是K-means算法 (In math)
  - INPUT:
    - K(number of clusters)
    - Training set(输入训练数据x值)
  - x为n维数据

$x^{(i)} \in \mathbb{R}^n$  (drop  $x_0 = 1$  convention)

- K-Means算法的步骤

- 随机选取总数为K的簇中心  $\mu_1, \mu_2, \dots, \mu_k$

- 迭代 {
  - 簇分类
  - 求平均
  - }

- 簇分类
- 求平均
- }

### K-means algorithm

$$\mu_1 \quad \mu_2$$

Randomly initialize  $K$  cluster centroids  $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

Repeat {

Cluster Assignment step

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

for  $k = 1$  to  $K$   
 $\rightarrow \mu_k$  := average (mean) of points assigned to cluster  $k$

$x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}$        $\rightarrow c^{(1)}=2, c^{(2)}=2, c^{(3)}=2, c^{(4)}=2$

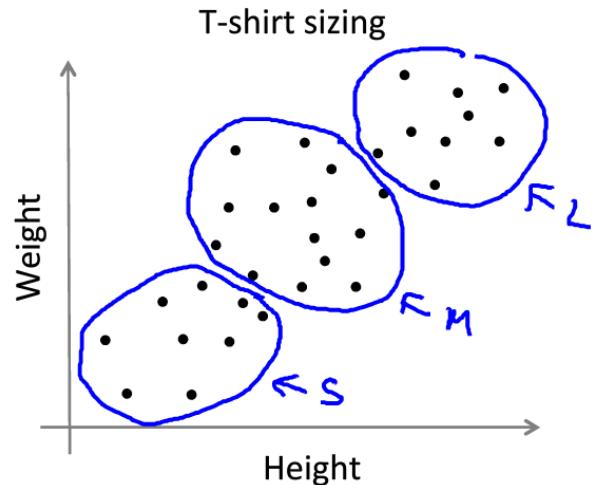
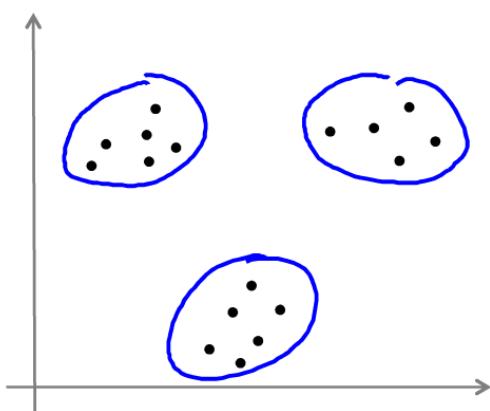
$\mu_2 = \frac{1}{4} [x^{(1)} + x^{(2)} + x^{(3)} + x^{(4)}] \in \mathbb{R}^n$

- K-Means算法的应用

- 在数据并不明显分类上也可以用，比如T-shirt制造商可以按照此种算法设计S, M, L码的衣服

### K-means for non-separated clusters

S, M, L



### 3. 优化目标

- 定义变量意义

- $x^{(i)}$  :  $i$ 代表第几个训练数据
- $c^{(i)} = \text{cluster } (1, 2, 3 \dots K)$ : 表示第*i*个数据 $x^{(i)}$ 属于第几个cluster  $c^{(i)} = k$

- $\mu_k$  : 是第*k*个cluster中心位置

- $\mu_c^{(i)}$  : 表示第*i*个数据 $x^{(i)}$ 在第 $c^{(i)}$ 个cluster, 比如 $x^{(i)} \rightarrow 5$ ,  $c^{(i)} = 5$ ,  $\mu_c^{(i)} = \mu_5$

- 上面的定义搞明白之后就定义出了一个优化目标

- 失真代价函数 (distortion cost function)

## K-means optimization objective

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

Optimization objective:

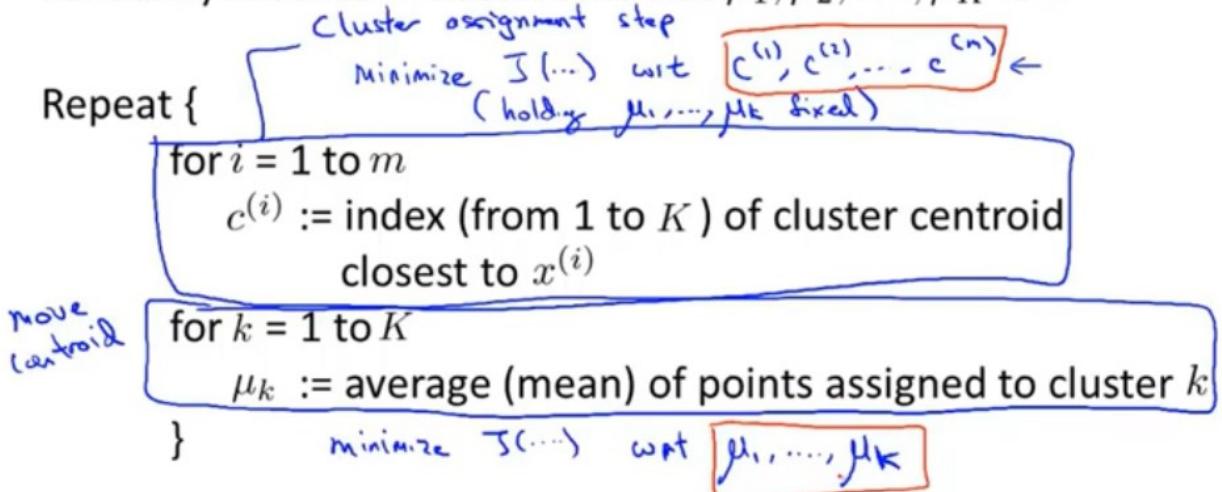
$$\rightarrow J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2 \leftarrow$$

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

- K-means 算法深入理解
  - 注意：第一步计算失真函数是将数据分类（分配红蓝颜色），但是保持  $\mu_k$  值不动 fixed
  - 第二步用失真函数来找到  $\mu_k$  的最优值（令 j 最小，即取平均等方法）

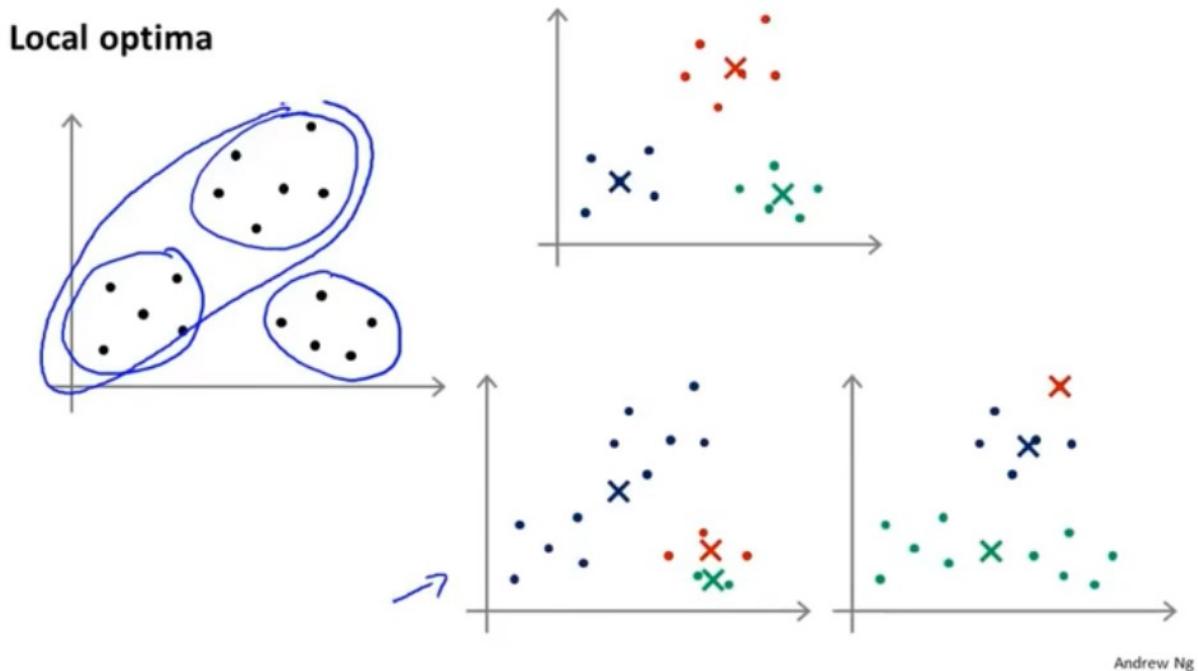
## K-means algorithm

Randomly initialize  $K$  cluster centroids  $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$



### 4. 随机初始化 (及避开局部最优解)

- 随机初始化
  - 选择  $K$  值数目 < 数据的数目
  - 从训练数据中随机选取  $K$  个数值
  - 将  $K$  个数值赋值给  $\mu_k$
- 局部最优
  - 有时当初始数据选取不好的时候就会优化出下图中的局部最优解



Andrew Ng

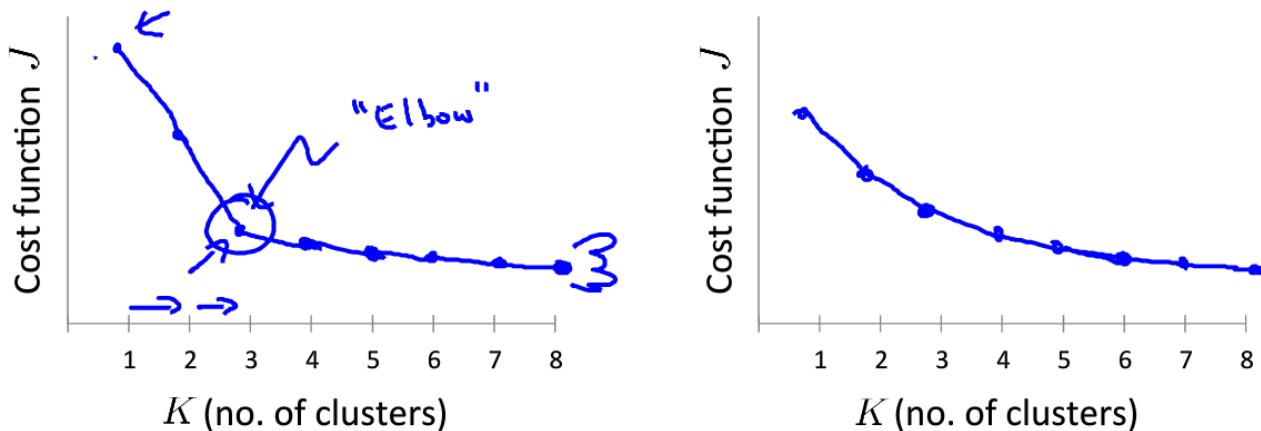
- 解决办法就是多次随机初始化
  - 通常当选取2-10个聚类时，多次随机初始化会有很好的优化结果
  - 但是如果超过10的话就优化不明显。

## 5. 选取聚类的数量

- 肘部法则：选取肘部的K个数如下图所示，但是通常情况下可能会出现右图的结果，没有明显的肘部

### Choosing the value of K

Elbow method:

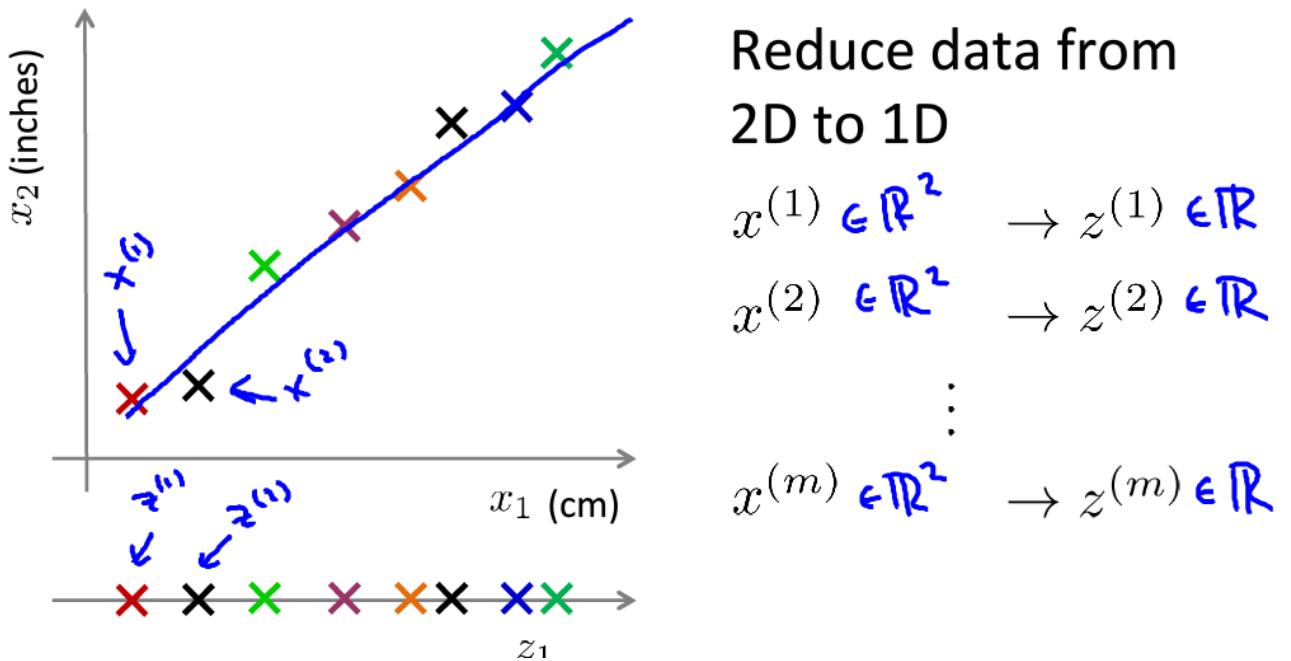


- 通常的方法根据需要手动选择  
比如选择3种T-shirt码数，或者5种T-shirt码数，从上月的角度去考虑是否需要增加或者减少T-shirt的码数。

## 15. 降维（压缩）

### 1. 数据压缩

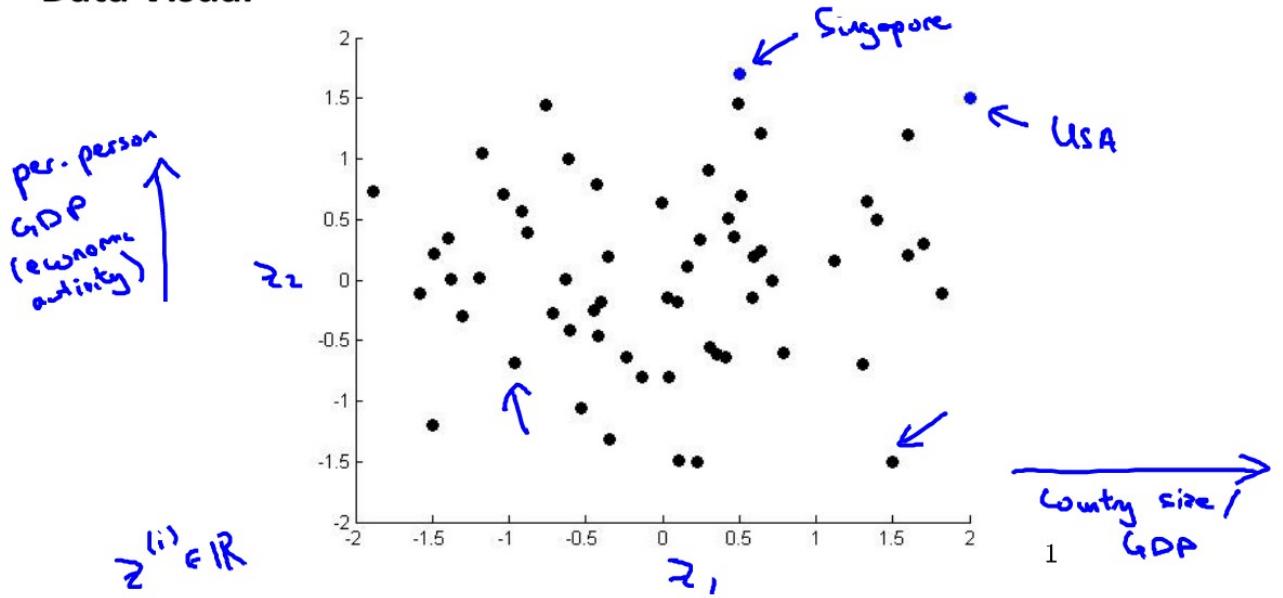
- 将二维数据压缩成一维数据



## 2. 可视化

- 比如将很多维度的数据降低成二维数据就比较容易分析

### Data Visualization



## 3. 主成分分析问题规划

- 主成分分析算法
  - 一种非常困难的降维算法
  - 训练数据 \$x\$，平均化

## 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  $x_j^{(i)} - \mu_j$ .

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.

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

- 算法解释

- 从n维降低到k维
- 计算协方差矩阵
- $[U, S, V] = \text{svd}(\Sigma)$
- 输入是x的值，即Sigma，输出的是U, S, V矩阵。其中U可以理解为一种算法，就像H哈密顿量一样。

## 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*

Compute “eigenvectors” of matrix  $\Sigma$ :

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

*eig (Sigma)*

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

*nxn matrix*

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

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

- Z (x缩放得到的) 可以通过如下求得

- 从U中取k维的向量，乘以x

## Principal Component Analysis (PCA) algorithm

From  $[U, S, V] = \text{svd}(\Sigma)$ , we get:

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

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

$$z^{(i)} = \begin{bmatrix} u^{(1)} \\ u^{(2)} \\ \vdots \\ u^{(k)} \end{bmatrix}^T \quad X^{(i)} = \begin{bmatrix} (u^{(1)})^T \\ \vdots \\ (u^{(k)})^T \end{bmatrix}$$

$U_{\text{reduce}}$

Andrew Ng

- 主成分数量的选择
- PCA算法总结
  - 经过归一 (normalization) 和优化特征缩放 (确保每个feature有0平均值) – 计算Sigma
  - 得到 $[U, S, V] = \text{svd}(\Sigma)$
  - 得到 $U_{\text{reduce}} = U(:, 1:k)$ : nxk维
  - $z = U_{\text{reduce}}' * X$

## Principal Component Analysis (PCA) algorithm summary

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

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

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

$\Rightarrow U_{\text{reduce}} = U(:, 1:k);$

$\Rightarrow z = U_{\text{reduce}}' * x;$

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

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

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

- 补充:

- PCA 减少n维到k:
  - 第一步是均值归一化。我们需要计算出所有特征的均值，然后令 $x_j = x_j - \mu_j$ 。如果特征是在不同的数量级上，我们还需要将其除以标准差  $\sigma^2$ .

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

- 第二步是计算协方差矩阵 (covariance matrix)  $\Sigma$ :

- 第三步是计算协方差矩阵  $\Sigma$  特征向量 (eigenvectors) :

- 在 Octave 里我们可以利用奇异值分解 (singular value decomposition) 来求解,  $[U, S, V] = \text{svd}(\Sigma)$

对于一个  $n \times n$  维度的矩阵，上式中的  $\mathbf{U}$  是一个具有与数据之间最小投射误差的方向向量构成的矩阵。如果我们希望将数据从  $n$  降至  $k$ ，我们只需要从  $\mathbf{U}$  选取前  $k$  向量，获得一个  $n \times k$  度的矩阵，我们用  $\mathbf{U}_{reduce}$  表示，然后通过如下计算获得要求的新特征向量  $\mathbf{z}^{(i)}$

$$\mathbf{z}^{(i)} = \mathbf{U}_{reduce}^T * \mathbf{x}^{(i)}$$

- 其中  $\mathbf{x}$  是  $n \times 1$  维的，因此结果为  $k \times 1$  维度。注，我们不对方差特征进行处理。

#### 4. 主成分数量选择

- 方差保留比例
  - 可以理解为，虽然减少了  $x$  数据的数量，但是保留了 99% 的方差不变

Accepted Doge Bilibili

## 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

$$\begin{aligned} & \rightarrow \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2 \leq \frac{0.01}{0.05} \quad (1\%) \\ & \rightarrow \frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2 \leq \frac{0.10}{0.10} \quad (50\%) \end{aligned}$$

$\rightarrow$  “99% of variance is retained”  
~~98 to 90%~~



- 方差保留的详细算法

- 左图这种选择一种  $k$ ，就需要计算所有的  $\mathbf{U}_{reduce}$ ,  $\mathbf{z}^{(i)}$ ,  $\mathbf{x}_{approx}^{(i)}$  的值，需要很大的计算量
- 可以用  $S$  来计算

- 对于给定的  $k$  值只需要计算  $1 - \frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}} \leq 0.01$  就可以得到方差误差小于 0.01 条件下， $k$  的取值。

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

Algorithm:

Try PCA with  $k = 1$  ~~but less~~ ~~k=4~~

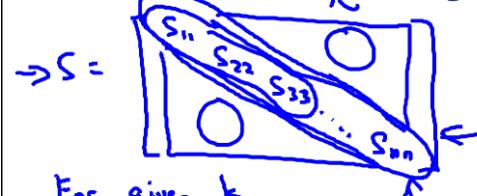
Compute  $\mathbf{U}_{reduce}$ ,  $\mathbf{z}^{(1)}$ ,  $\mathbf{z}^{(2)}$ ,  
 $\dots$ ,  $\mathbf{z}^{(m)}$ ,  $\mathbf{x}_{approx}^{(1)}$ ,  $\dots$ ,  $\mathbf{x}_{approx}^{(m)}$

Check if

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

$$k = 17$$

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



$$k=3$$

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

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

Andrew Ng

# Choosing $k$ (number of principal components)

→  $[U, S, V] = \text{svd}(\Sigma)$

Pick smallest value of  $k$  for which

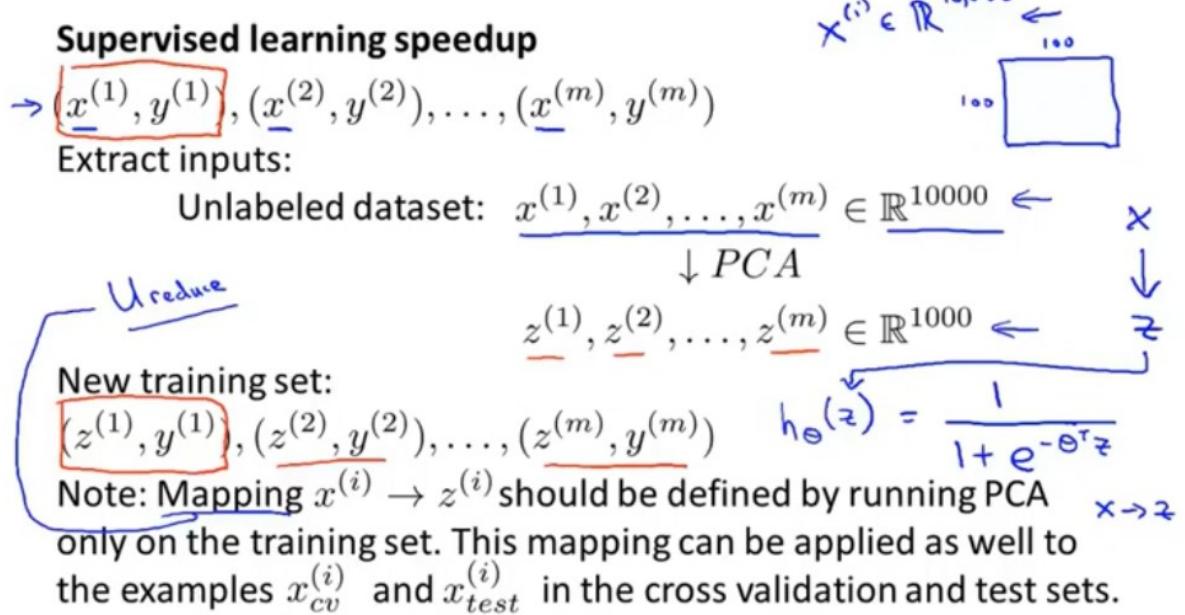
$$\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^m S_{ii}} \geq 0.99$$

$k=100$

(99% of variance retained)

## 5. 应用PCA的建议

- 从  $x$  (1000维) →  $z$  (100维) 的降维
- 然后再从  $z$  到  $y$  值进行无监督学习算法 (比如 logistic 回归)
  - 注意, map 定义的时候只能用 training 集, 应用的时候可以用到交叉和测试  $x$  数据



- PCA 的应用
  - 压缩, 加快计算
  - 可视化
    - 针对  $k=2, k=3$ , 三维图像
- PCA 的错误应用
  - 用 PCA 处理过度拟合
    - 这样做非常不好, 不如尝试正则化处理。原因在于主要成分分析只是近似地丢弃掉一些特征, 它并不考虑任何与结果变量有关的信息, 因此可能会丢失非常重要的特征。然而当我们进行正则化处理时, 会考虑到结果变量, 不会丢掉重要的数据。

## Bad use of PCA: To prevent overfitting

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

Thus, fewer features, less likely to overfit.

Bad!

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

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

- 在用PCA之前，先用原始数据计算，只有原始数据太多才考虑PCA

## PCA is sometimes used where it shouldn't be

Design of ML system:

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

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

## 16. 异常检测

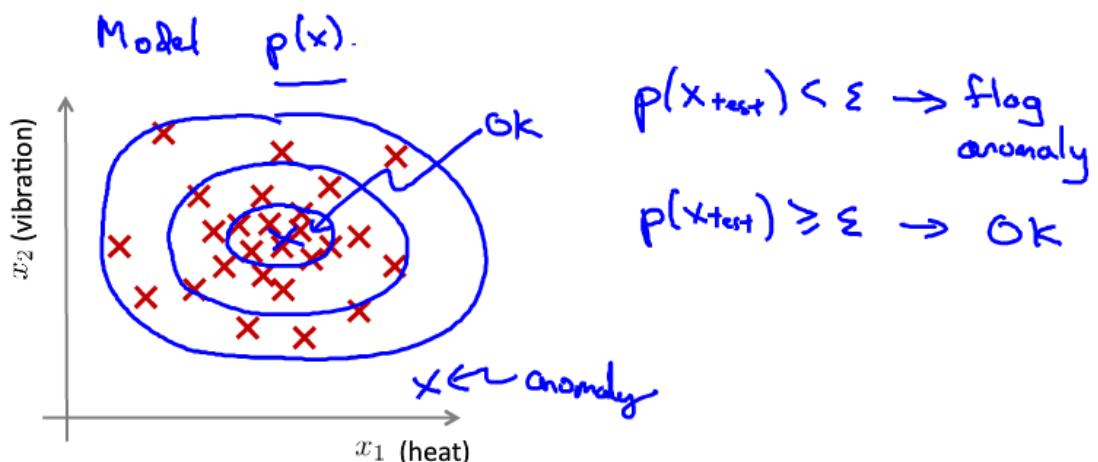
### 1. 问题动机

- 如下图所示，异常检测的目的就是设定一个筛选标准，找出不满足筛选标准的数据，如下图蓝色圈外的数据，为异常数据。

## Density estimation

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

→ Is  $x_{test}$  anomalous?



### 2. 高斯分布回顾

- 高斯是正态函数，积分为1

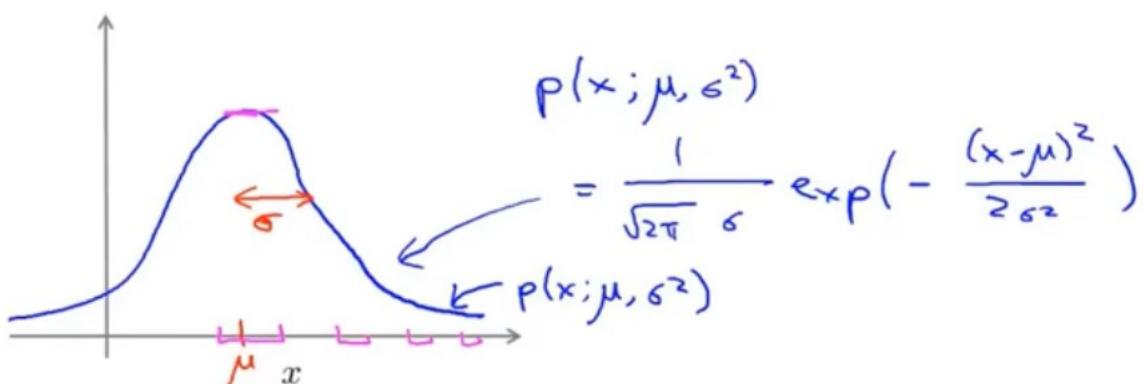
### Gaussian (Normal) distribution

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

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

↖ "distributed as"

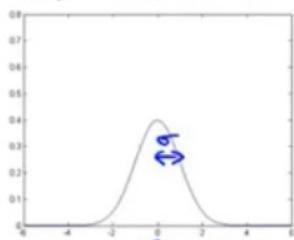
$\sigma$  standard deviation



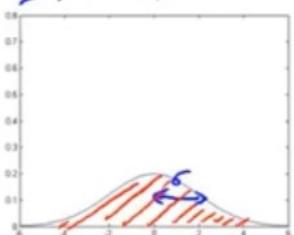
- 宽度等于标准差

## Gaussian distribution example

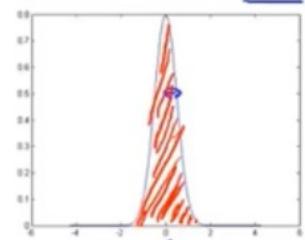
$$\rightarrow \mu = 0, \sigma = 1$$



$$\rightarrow \mu = 0, \sigma = 2$$

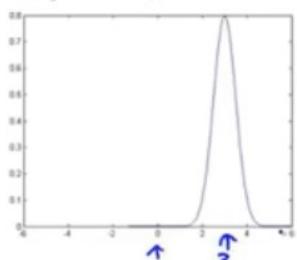


$$\rightarrow \mu = 0, \sigma = 0.5$$



$$\sigma^2 = 0.25$$

$$\rightarrow \mu = 3, \sigma = 0.5$$



Andrew Ng

- $m$  和  $1/m$



## Parameter estimation

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

$$x^{(i)} \sim \mathcal{N}(\mu, \sigma^2)$$

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

$$\rightarrow \sigma^2 = \frac{1}{m-1} \sum_{i=1}^m (x^{(i)} - \underline{\mu})^2$$

Andrew Ng

3. 算法 (密度估计)

## Density estimation

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

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

$$x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$$

$$x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$$

$$x_3 \sim \mathcal{N}(\mu_3, \sigma_3^2)$$

$p(x)$

$$= p(x_1; \mu_1, \sigma_1^2) p(x_2; \mu_2, \sigma_2^2) p(x_3; \mu_3, \sigma_3^2) \dots p(x_n; \mu_n, \sigma_n^2) \leftarrow$$

$$= \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2)$$

$$\sum_{i=1}^n i = 1+2+3+\dots+n$$

$$\prod_{i=1}^n i = 1 \times 2 \times 3 \times \dots \times n$$

- 步骤

## Anomaly detection algorithm

→ 1. Choose features  $x_i$  that you think might be indicative of anomalous examples.  $\{x^{(1)}, \dots, x^{(m)}\}$

→ 2. Fit parameters  $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$

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

$$p(x_j; \mu_j, \sigma_j^2)$$

$$\begin{aligned} &\mu_1, \mu_2, \dots, \mu_n \\ &\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix} = \frac{1}{m} \sum_{i=1}^m x^{(i)} \end{aligned}$$

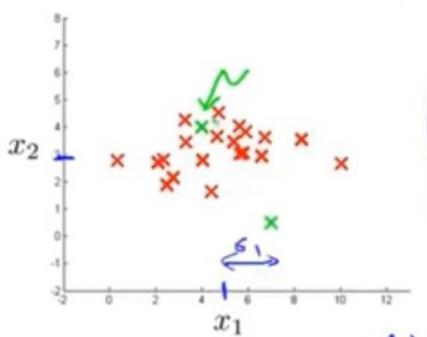
$$\rightarrow \sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2 \leftarrow$$

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

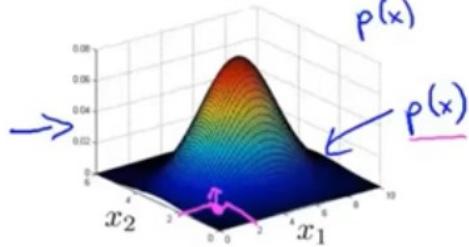
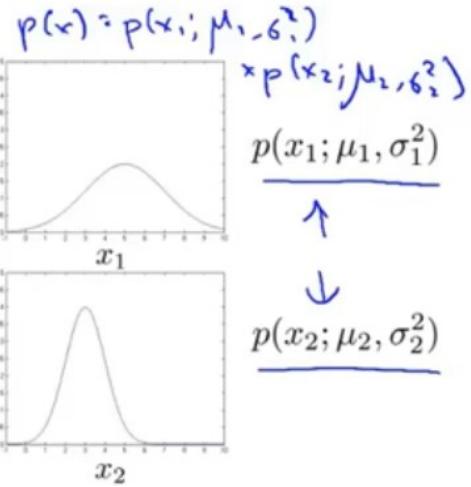
$$p(x) = \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^n \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

Anomaly if  $p(x) < \varepsilon$

## Anomaly detection example



$$\begin{aligned} \mu_1 &= 5, \sigma_1^2 = 2 \\ \mu_2 &= 3, \sigma_2^2 = 1 \end{aligned}$$



Andrew Ng

- 开发和评估异常
- 考虑什么是影响学习算法的原因
- 假设有一些已经被标注的数据，异常或者非异常的例子 ( $y=0$ , if normal,  $y=1$  if anomalous)
- 训练集:  $x^{(1)}, x^{(2)}, \dots, x^{(m)}$  (包括正常或者非正常)
- 交叉验证和测试集合
  - 已经知道是否异常 ( $y$ 的值)

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

- 举例说明 (飞机发动机)
  - 假设有10000个好的 (normal) 发动机x值
    - 其中混有不好的也没关系
    - 通常应用中大于10000
  - 20个不好的发动机 (anomalous)
    - 通常2~50个
  - 进行分配:
    - 训练集: 6000个normal 发动机
      - 目的是求  $\mu_1, \sigma_1^2; \mu_2, \sigma_2^2; \dots; \mu_n, \sigma_n^2$  从而拟合  $p(x) = p(x_1, \mu_1, \sigma_1^2) \cdots p(x_n, \mu_n, \sigma_n^2)$  的值
    - 交叉验证集: 2000个normal发动机 ( $y=0$ ), 10个anomalous ( $y=1$ )
      - 测试集: 2000个normal发动机 ( $y=0$ ), 10个anomalous ( $y=1$ )
  - 错误的分配方式:
    - 比如将交叉验证和测试集用相同的发动机数据训练
    - 或者交叉验证和测试集用相同的anomalous值
- 算法

## Algorithm evaluation

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

- 算法评估
  - 用交叉验证测试集合进行验证
  - 可能的评估方法
    - F1score
  - 取不同的 $\varepsilon$ 以使 $F_1$ 最大

## Algorithm evaluation

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

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

$y=0$

Possible evaluation metrics:

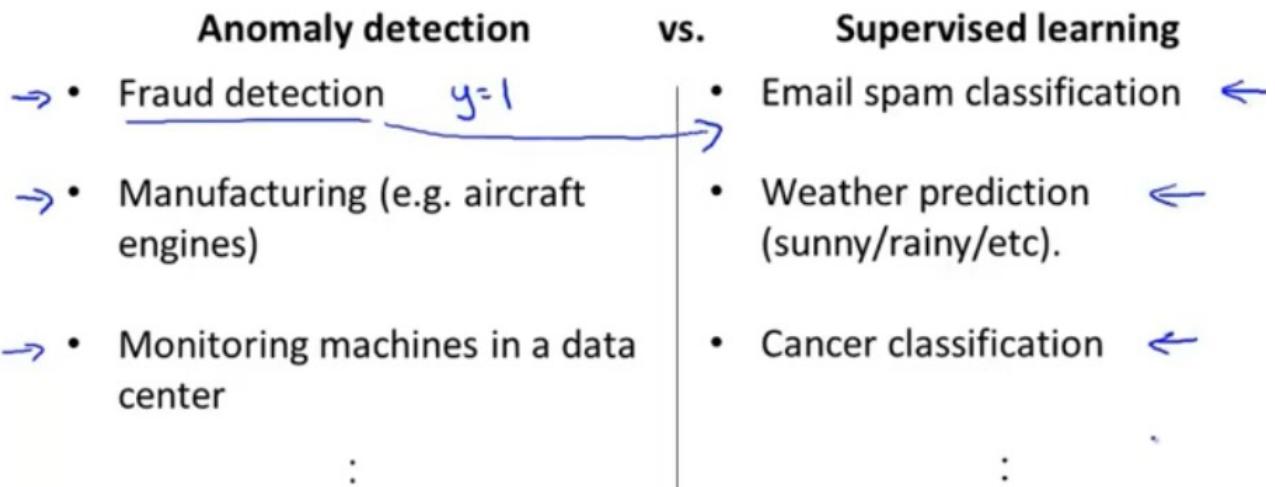
- - True positive, false positive, false negative, true negative
- - Precision/Recall
- -  $F_1$ -score ↙

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

## 4. 异常检测vs. 监督算法

- 什么时候使用异常检测方法?
  - 非常少的positive example ( $y=1$ ) (eg. 0-20)
  - 大量的negative example ( $y=0$ )，拟合 $p(x)$ 只需异常值也可以得到 (此时，用这些少量的positive数据作为交叉验证，测试来使用)
  - 许多类型的异常类型，很难用算法从positive例子中拟合异常情况 (因为positive数据太少)
  - 有可能未来出现其他的完全不同的异常值，无法进行预测
  - 综上需要利用高斯拟合来判断异常
- 什么时候用监督学习方法
  - 大量的positive, negative例子
  - 可以通过以上离子拟合出预测曲线，而且未来出现的positive例子应当是类似的，可预测的！此时才能用监督学习算法

- 比如垃圾邮件的例子，我们有足够的数据来判断一封邮件是否。
- 应用举例

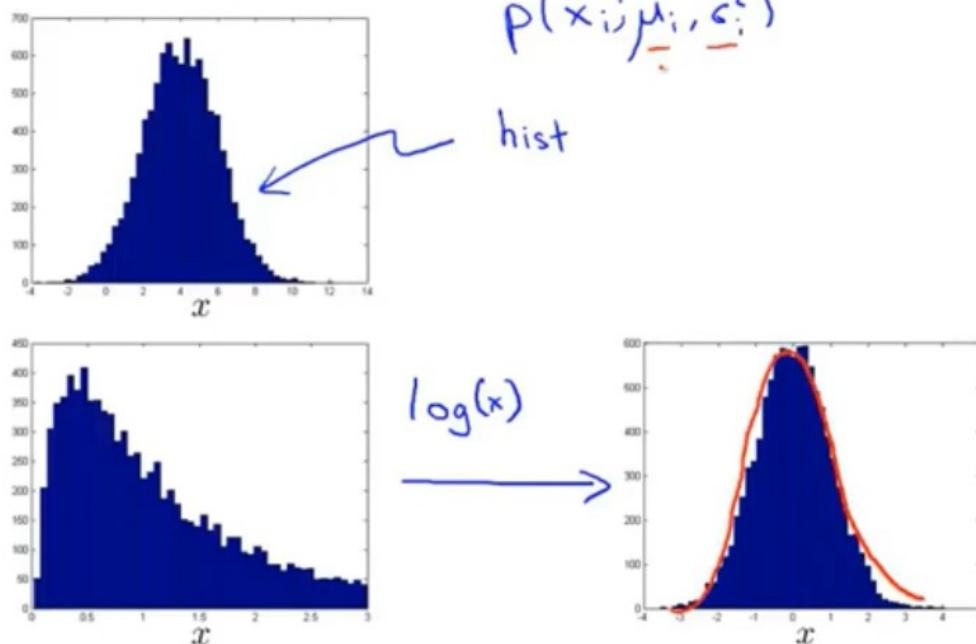


## 5. 选择要使用的功能

如何设计和选择算法特征

- 非高斯特征值
  - 对数据进行转化，使其更加符合高斯分布。比如下图中的  $\log(x) = \ln(x)$  转化方式

### Non-gaussian features

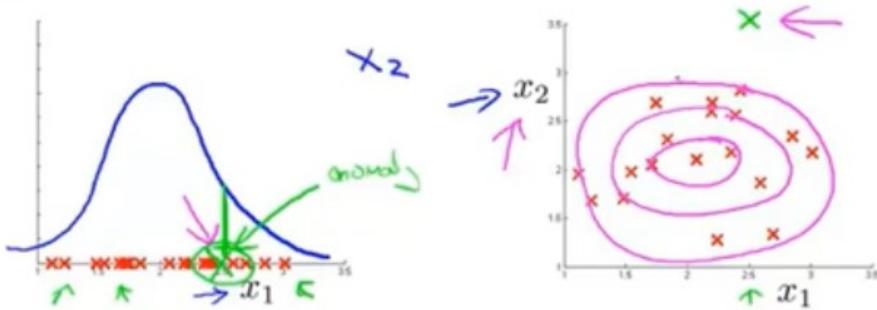


- 其他转化方式，比如在  $\log(x)$  中加入常数  $c$ :  $\log(x + c)$ , 或者  $x \rightarrow x^{1/2}$  等，都可以将特征值  $x, y$  图像的分布更加接近高斯分布

$$\begin{aligned}
 x_1 &\leftarrow \log(x_1) \\
 x_2 &\leftarrow \log(x_2 + 1) \\
 x_3 &\leftarrow \sqrt{x_3} = x_3^{1/2} \\
 x_4 &\leftarrow x_4^{1/3}
 \end{aligned}$$

- 异常值的检测方法分析

- 目的： $p(\mathbf{x})$ 值在 $\mathbf{x}$ 为正常例子时很大  
 $p(\mathbf{x})$ 值在 $\mathbf{x}$ 为异常例子时很小
- 但是大多数情况遇到的问题：  
 $p(\mathbf{x})$ 对于异常或者正常的例子时，相差不大，无法将异常数据找出来。比如左图中所示，异常(绿色) $\mathbf{x}$ 对于的 $p(\mathbf{x})$ 无法识别出来。
- 所以：寻找另一个特征值 $x_2$  ( $y$ 轴)，从而使绿色异常值被区分出来（右图）。



- 常用的异常检测算法（选择特征变量时的思考）
  - 以数据中心检测异常电脑为例  
 选择那些在可能会出现不寻常大，或者不寻常小的特征值
  - 比如如果CPU load很大，如果此时network traffic也很大，说明此时访问量导致的CPU load大，是正常现象。  
 但是，如果此时network traffic并没有增加，说明CPU自己陷入了死循环。  
 所以用两者相除（或者相关变体，右公式）作为增加的特征值！

→ Monitoring computers in a data center

→ Choose features that might take on unusually large or small values in the event of an anomaly.

→  $x_1$  = memory use of computer

→  $x_2$  = number of disk accesses/sec

→  $x_3$  = CPU load ←

→  $x_4$  = network traffic ←

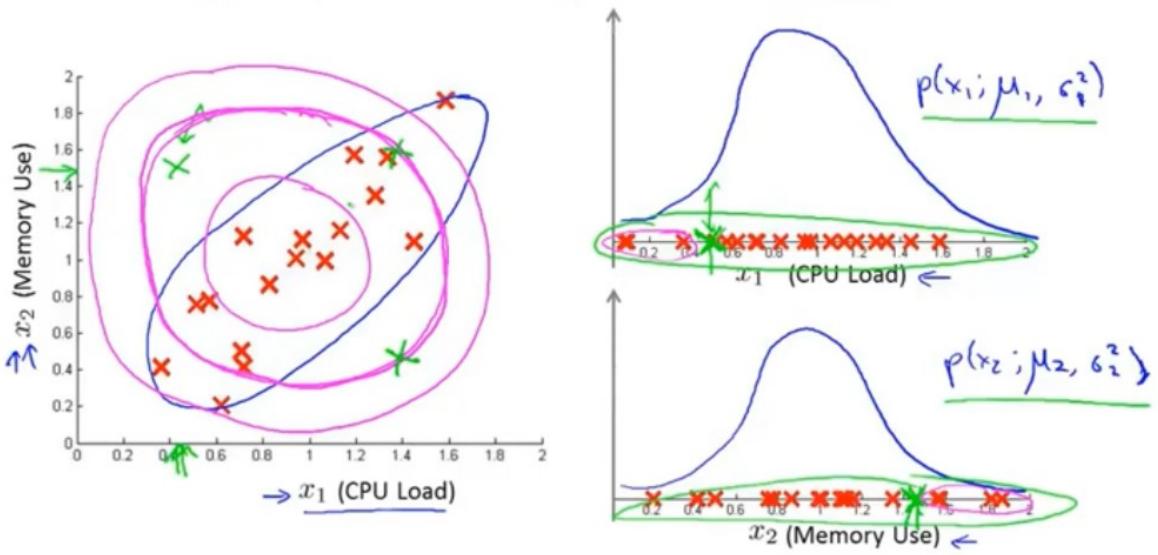
$$x_5 = \frac{\text{CPU load}}{\text{network traffic}}$$

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

## 6. 多元高斯分布

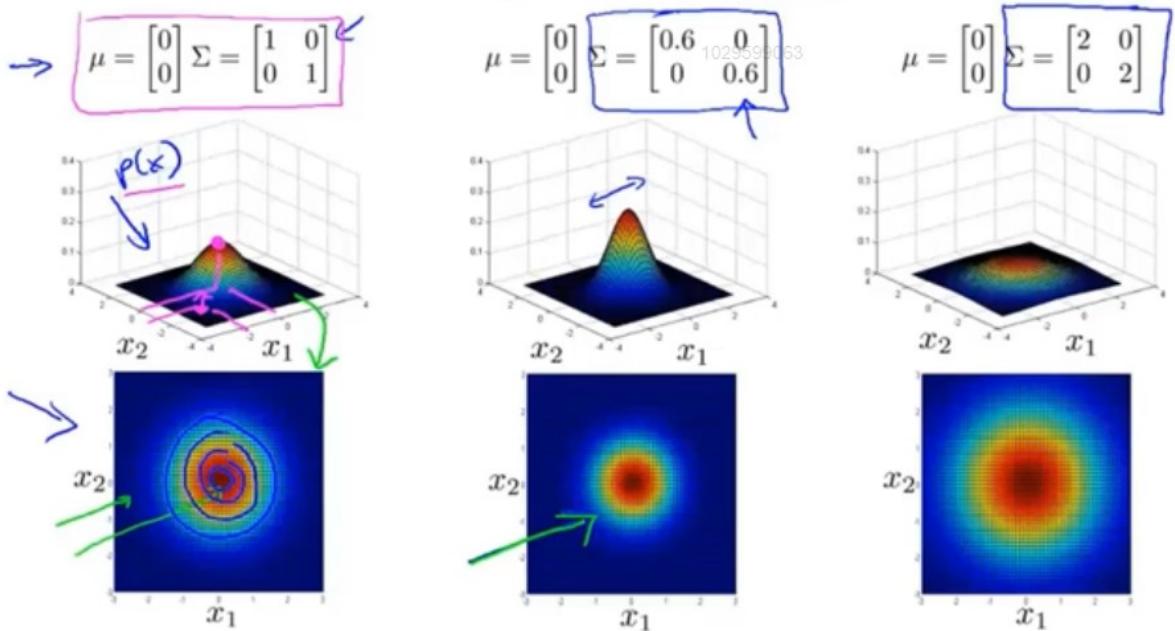
- 为什么使用多元高斯分布
  - 如下左图，绿色点虽然比较异常，但是高斯分布的图像，绿色点和同一半径下(粉色圈)的红色点的 $p$ 值相同。所以无法区分绿色点。

## Motivating example: Monitoring machines in a data center



- 多元高斯分布举例
  - 当多元高斯分布  $\Sigma$  (方差矩阵) 取值相等时

## Multivariate Gaussian (Normal) examples



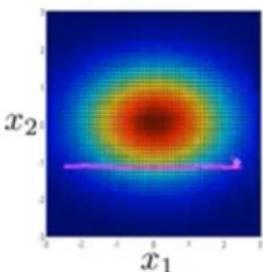
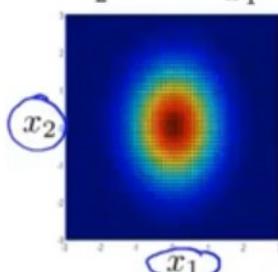
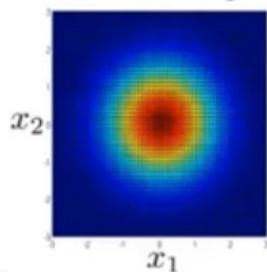
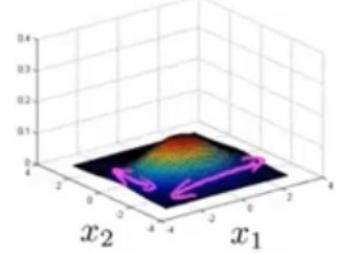
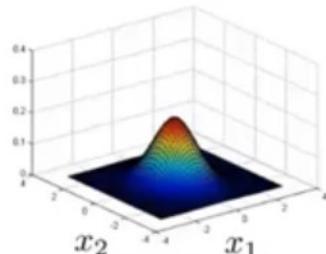
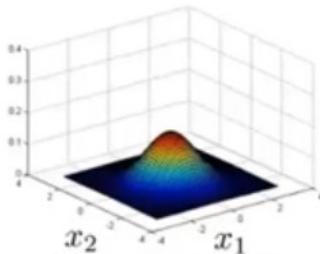
- 当多元高斯分布  $\Sigma$  (方差矩阵) 取值不等时

## Multivariate Gaussian (Normal) examples

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$



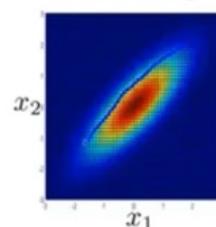
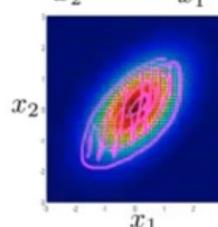
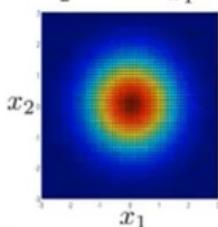
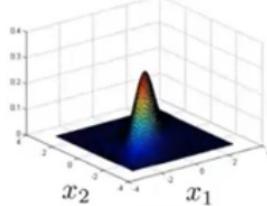
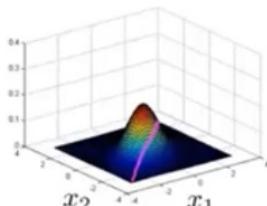
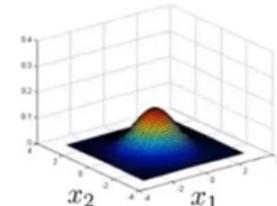
- 当多元高斯分布  $\Sigma$  (方差矩阵) 取值为下列这种形式时, 可以得到x1, 和x2相关性的图像

## Multivariate Gaussian (Normal) examples

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$



- 当多元高斯分布  $\Sigma$  (方差矩阵) 取值为下列这种形式时(负数), 可以得到x1, 和x2相关性的图像

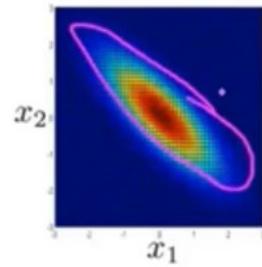
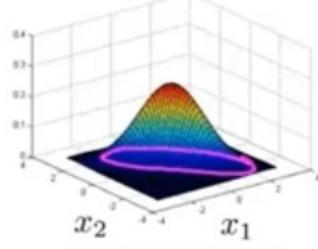
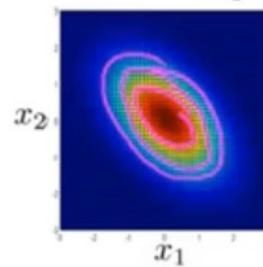
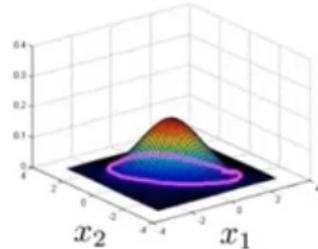
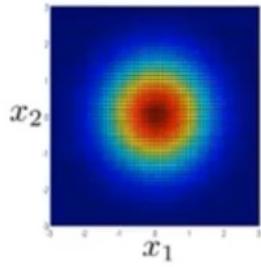
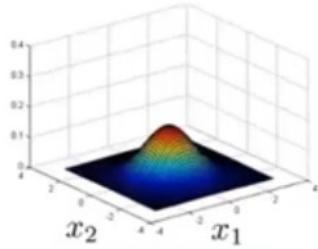
Andrew Ng

## Multivariate Gaussian (Normal) examples

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$$

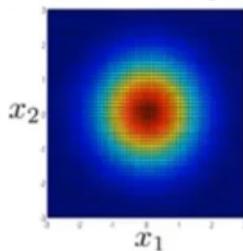
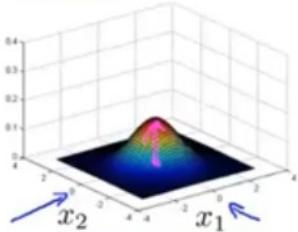
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$$



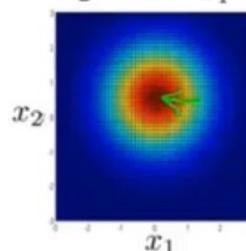
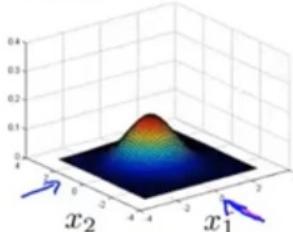
- 当改变  $\mu$  的值，也可以改变中心的位置

## Multivariate Gaussian (Normal) examples

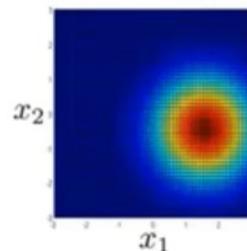
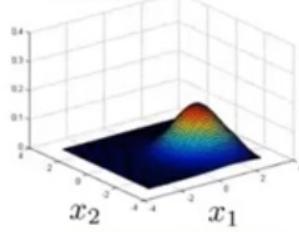
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



Andrew Ng

### 7. 使用多变量高斯分布的异常检测

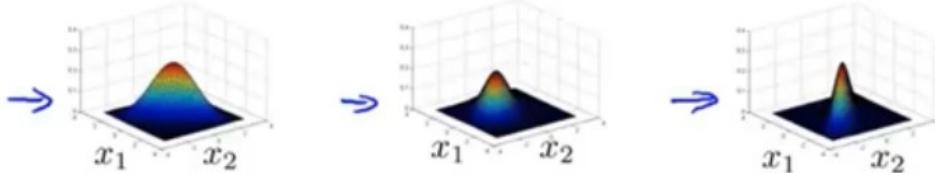
- 所用公式如下

## Multivariate Gaussian (Normal) distribution

Parameters  $\underline{\mu, \Sigma}$

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

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



Parameter fitting:

Given training set  $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\} \leftarrow \text{102599063} \quad x \in \mathbb{R}^n$

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

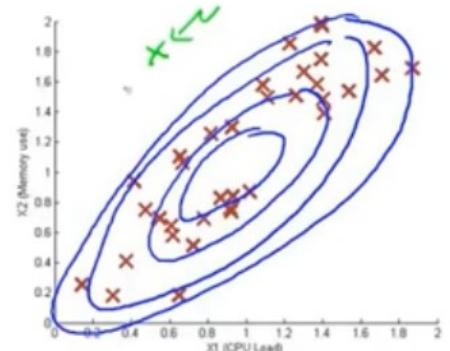
- 步骤：

- 利用  $\mu$  和  $\Sigma$  的公式和训练数据来得到  $p(x)$
- 对于新的数据  $x$ , 利用多元高斯分布公式计算  $p(x)$  的值, 如果  $p(x) < \epsilon$  说明该新数据为异常标记。

## Anomaly detection with the multivariate Gaussian

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

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



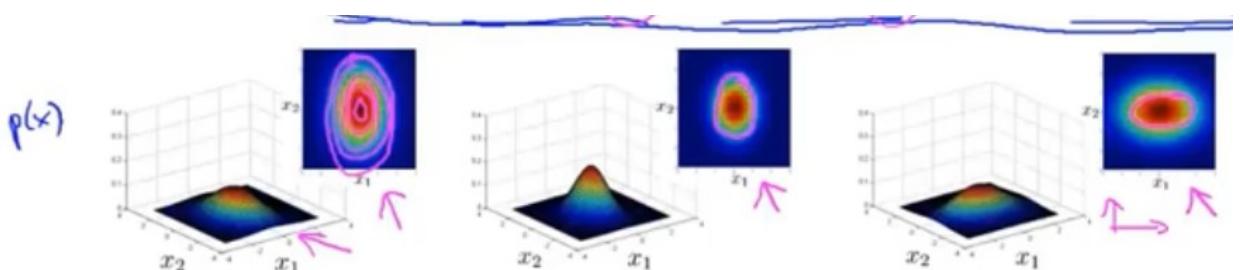
2. Given a new example  $x$ , compute

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

Flag an anomaly if  $\underline{p(x) < \epsilon}$

- 多元高斯分布模型和原始模型有哪些异同呢?

- 从数学公式上来说:



对于上图中的类似形状(椭圆的a, b轴与坐标轴垂直) :

- 用原始高斯模型来表示，是多个  $p(x_1; \mu_1, \sigma_1^2) \times \dots \times p(x_n; \mu_n, \sigma_n^2)$  的乘积。

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

- 用多元高斯模型来表示， $\Sigma$  只有中间轴不为零，中间轴的  $\sigma^2$  就分别对应原始高斯模型中的  $\sigma^2$

$$\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

对于椭圆的 a, b 轴与坐标轴不垂直的情况，如下

此时  $\Sigma$  取值如上图形式

结论：原始高斯模型可以表示的图像，只是多元高斯图像中的一种特殊形态

- 如何在原始高斯分布和多元高斯分布之间做选择？

- 原始高斯分布特点

- 如果可以手动找到所有可能关键特征值，比如 (cpu load/network)，那么使用原始高斯分布就是有效的。
- 计算成本低
- 即使训练集的数量  $m$  很小也可以使用

- 多元高斯分布特点

- 自动找到所有相关特征值，适应于  $m$  很大 ( $n$  不大) 的情况，人为找到关键特征值很难
- 计算成本高，因为有  $\Sigma n \times n$  矩阵的存在。
- 训练集数量  $m$  需要远大于特征值数量  $n$ ，否则  $\Sigma$  不可逆，通常要求  $m \geq 10n$ ，因为需要求解  $n \times n$  的  $\Sigma$  矩阵

- 注意（多元高斯，低概率出现情况）

- 如果遇到  $\Sigma$  矩阵不可逆的情况，有两种原因

- $m$  没有远大于  $n$
- $\Sigma$  矩阵中存在相等特征值  $x_1, x_2$ ；或线性相关  $x_3 = x_4 + x_5$

→ 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.

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

→ 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

$$\Sigma \sim \frac{n^2}{2}$$

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

$$m \geq 10n$$

1029599063

Andrew Ng

## 17. 推荐系统

最常用的机器学习算法之一

### 1. 问题规划

举例，预测电影的评分

- 符号

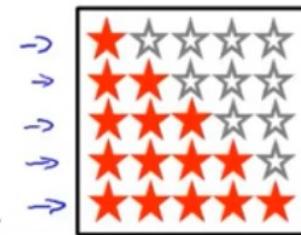
- $n_u = \text{no. users}$  用户的序号, 4
- $n_m = \text{no. movies}$  电影的序号, 5
- $r(i, j) = 1$  if 用户  $j$  对电影  $i$  有评分数据, 表格中问号? 表示不存在评分
- $y^{(i,j)}$  = 用户  $j$  对电影  $i$  的评分分数, 预测目标

### Example: Predicting movie ratings

→ User rates movies using one to five stars  
zero

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	6
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

$$n_u = 4 \quad n_m = 5$$



- $n_u = \text{no. users}$
- $n_m = \text{no. movies}$
- $r(i, j) = 1$  if user  $j$  has rated movie  $i$
- $y^{(i,j)}$  = rating given by user  $j$  to movie  $i$  (defined only if  $r(i, j) = 1$ )

- 对不同电影用特征值  $x_1, x_2$  描述, 然后某影片的特征矩阵乘以某人的参数矩阵

**Content-based recommender systems**

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	$x_1$ (romance)	$x_2$ (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

$n_u = 4, n_m = 5$

$\theta_0 = 1$

$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$

$n = 2$

For each user  $j$ , learn a parameter  $\theta^{(j)} \in \mathbb{R}^3$ . Predict user  $j$  as rating movie  $i$  with  $(\theta^{(j)})^T x^{(i)}$  stars.

$\theta^{(1)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix} \Leftrightarrow \Theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \quad (\Theta^{(1)})^T x^{(3)} = 5 \times 0.99 = 4.95$

- 正式步骤

- $r(i, j) = 1, y^{(i,j)}, \theta^{(j)}$  用户参数矩阵, 矩阵第一行是偏置常数, 第二个表示对第一个特征值的喜爱度, 第三个数字表示对影片第二个特征值的喜爱度,  $x^{(i)}$  电影特征矩阵
- 目标: 对于用户  $j$ , 影片  $i$ , 得出预测评分  $(\theta^{(j)})^T (x^{(i)})$

## Problem formulation

- $r(i, j) = 1$  if user  $j$  has rated movie  $i$  (0 otherwise)
- $y^{(i,j)}$  = rating by user  $j$  on movie  $i$  (if defined)
- $\theta^{(j)}$  = parameter vector for user  $j$
- $x^{(i)}$  = feature vector for movie  $i$
- For user  $j$ , movie  $i$ , predicted rating:  $(\theta^{(j)})^T(x^{(i)})$

- 新定义:  $m^{(j)} = \text{no. of movies rated by user } j$
- 需要得到  $\theta^{(j)}$ :

To learn  $\underline{\theta^{(j)}}$  (parameter for user  $j$ ):

$$\rightarrow \min_{\theta^{(j)}} \frac{1}{2} \sum_{i:r(i,j)=1} \left( (\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^n (\theta_k^{(j)})^2$$

- 对于其他所有的  $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)}$ :

To learn  $\underline{\theta^{(1)}}, \underline{\theta^{(2)}}, \dots, \underline{\theta^{(n_u)}}$ :

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left( (\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

$\Theta^{(1)}, \dots, \Theta^{(n_u)}$

- 个人理解: 机器学习的核心算法,
  - 总有一个自变量, 其中有未知参数  $\Theta$ ,  $x$  是作为已知量 (训练集)
  - 接下来需要求  $\Theta$  值, 利用代价函数  $\min(\Theta^T x - y)^2$  和训练集就可以解方程得到  $\Theta$
- 求解  $\Theta$  的方法: 比如梯度近似 (与线性梯度不同的是  $1/m$  在优化算法中省略了?)

Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} \quad (\text{for } k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left( \sum_{i:r(i,j)=1} \underbrace{((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)}}_{\frac{\partial}{\partial \theta_k^{(j)}} J(\Theta^{(1)}, \dots, \Theta^{(n_u)})} + \lambda \theta_k^{(j)} \right) \quad (\text{for } k \neq 0)$$

比如簇梯度或者 LBFGS 等等

## 2. 协同过滤

在之前的基于内容的推荐系统中, 对于每一部电影, 我们都掌握了可用的特征, 使用这些特征训练出了每一个用户的参数。相反

地，如果我们拥有用户的参数，我们可以学习得出电影的特征。

- 条件

- 每个影片的特征值 $x_1, x_2$ 都不知道，知道的是，每位观众 $\theta$ 对于不同类型的喜好程度已知。(和上文相反，已知 $\theta$ ，求 $x$ )

## Problem motivation

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$	$\downarrow$ $x_1$ (romance)	$\downarrow$ $x_2$ (action)	$\downarrow$ $x_0 = 1$
$\cancel{x^{(1)}}$ Love at last	5	5	0	0	5	0	
Romance forever	5	?	?	0	?	?	
Cute puppies of love	?	4	0	?	?	?	
Nonstop car chases	0	0	5	4	?	?	
Swords vs. karate	0	0	5	?	?	?	

$\rightarrow \theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}, \theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$

$\theta^{(j)}$

$(\theta^{(1)})^T x^{(1)} \approx 5$   
 $(\theta^{(2)})^T x^{(1)} \approx 5$   
 $(\theta^{(3)})^T x^{(1)} \approx 0$   
 $(\theta^{(4)})^T x^{(1)} \approx 0$

Andrew Ng

- 下图显示了该种方法的主要步骤：

## Optimization algorithm

Given  $\theta^{(1)}, \dots, \theta^{(n_u)}$ , to learn  $x^{(i)}$ :

$$\min_{x^{(i)}} \frac{1}{2} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{k=1}^n (x_k^{(i)})^2$$

1029599063

Given  $\theta^{(1)}, \dots, \theta^{(n_u)}$ , to learn  $x^{(1)}, \dots, x^{(n_m)}$ :

$$\min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2$$

- 协同算法，利用初始化的电影特征值 $x$ 来学习用户参数 $\theta$ ，然后再反过来，利用 $\theta$ 估计电影的特征值 $x$ ，从而互相学习优化算法：

## Collaborative filtering

[ Given  $\underline{x^{(1)}, \dots, x^{(n_m)}}$  (and movie ratings),  
can estimate  $\underline{\theta^{(1)}, \dots, \theta^{(n_u)}}$  ]

[ Given  $\underline{\theta^{(1)}, \dots, \theta^{(n_u)}}$ ,  
can estimate  $\underline{x^{(1)}, \dots, x^{(n_m)}}$  ]

Guess     $\Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \dots$

### 3. 协同过滤算法

- 有了电影的特征值  $x$  就可以估计出用户的参数 (特征值)  $\theta$

◦  $i : r(i, j) = 1$  表示对所有用户  $j$  对电影  $i$  有评价的加和

Given  $x^{(1)}, \dots, x^{(n_m)}$ , estimate  $\theta^{(1)}, \dots, \theta^{(n_u)}$ :

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

- 有了用户参数  $\theta$ ，也可以估计出电影的特征值  $x$

◦  $j : r(i, j) = 1$  表示对有评价的加和

Given  $\theta^{(1)}, \dots, \theta^{(n_u)}$ , estimate  $x^{(1)}, \dots, x^{(n_m)}$ :

$$\min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2$$

- 上述需要反复迭代计算  $x$  和  $\theta$ ，下面的公式将上述公式综合起来。此时的  $x$  和  $\theta$  都没有  $x_0=1$  的偏置了。由算法自己寻找

## Collaborative filtering optimization objective $(i, j) : r(i, j) = 1$

→ Given  $x^{(1)}, \dots, x^{(n_m)}$ , estimate  $\theta^{(1)}, \dots, \theta^{(n_u)}$ :

$$\rightarrow \min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

→ Given  $\theta^{(1)}, \dots, \theta^{(n_u)}$ , estimate  $x^{(1)}, \dots, x^{(n_m)}$ :

$$\rightarrow \min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2$$

Minimizing  $x^{(1)}, \dots, x^{(n_m)}$  and  $\theta^{(1)}, \dots, \theta^{(n_u)}$  simultaneously:

$$J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

Andrew Ng

- 总结:

- 初始化  $\mathbf{x}, \Theta$
- 梯度计算
- 用  $(\Theta^T x^{(i)})^T (x^{(i)})$  来得到参数  $\theta$  的用户对特征值为  $x$  的电影的评分

## Collaborative filtering algorithm

→ 1. Initialize  $x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}$  to small random values.

→ 2. Minimize  $J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$  using gradient descent (or an advanced optimization algorithm). E.g. for every  $j = 1, \dots, n_u, i = 1, \dots, n_m$ :

$$x_k^{(i)} := x_k^{(i)} - \alpha \left( \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)} + \lambda x_k^{(i)} \right)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left( \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$

3. For a user with parameters  $\underline{\theta}$  and a movie with (learned) features  $\underline{x}$ , predict a star rating of  $\underline{\theta}^T \underline{x}$ .

$$(\underline{\theta}^{(i)})^T (\underline{x}^{(i)})$$

## 4. 矢量化: 低秩矩阵分解

推荐相关商品给用户

- 协同过滤

- $Y(i, j)$  表示第  $j$  个人对第  $i$  个电压的评分。

- $Y$  可以用右边的评分预测矩阵表示

- 它的计算公式:  $Y = X\Theta^T$

**Collaborative filtering**

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

Predicted ratings:  $\underbrace{\begin{bmatrix} (\theta^{(1)})^T(x^{(1)}) & (\theta^{(2)})^T(x^{(1)}) & \dots & (\theta^{(n_u)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & \dots & (\theta^{(n_u)})^T(x^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ (\theta^{(1)})^T(x^{(n_m)}) & (\theta^{(2)})^T(x^{(n_m)}) & \dots & (\theta^{(n_u)})^T(x^{(n_m)}) \end{bmatrix}}$

$$\Rightarrow X = \begin{bmatrix} -(x^{(1)})^T \\ -(x^{(2)})^T \\ \vdots \\ -(x^{(n_m)})^T \end{bmatrix} \quad \Theta = \begin{bmatrix} -(\Theta^{(1)})^T \\ -(\Theta^{(2)})^T \\ \vdots \\ -(\Theta^{(n_u)})^T \end{bmatrix}$$

*Low rank matrix factorization*

Andrew Ng

- 描述
  - 对于每一部电影都有一个特征矩阵 $x$
  - 如何判断电影j和电影i之间的关系呢？矩阵差值的绝对值。越小说明越接近
  - 这样就可以找出5部电影中最接近的2部

### Finding related movies

For each product  $i$ , we learn a feature vector  $\underline{x^{(i)}} \in \mathbb{R}^n$ .

$\rightarrow x_1 = \text{romance}, x_2 = \text{action}, x_3 = \text{comedy}, x_4 = \dots$

How to find  $\underline{\text{movies } j}$  related to  $\underline{\text{movie } i}$ ?

small  $\|\underline{x^{(i)}} - \underline{x^{(j)}}\| \rightarrow \text{movie } j \text{ and } i \text{ are "similar"}$

5 most similar movies to  $\underline{\text{movie } i}$ :

Find the 5 movies  $j$  with the smallest  $\|\underline{x^{(i)}} - \underline{x^{(j)}}\|$ .

### 5. 均值归一化

- Eve没有任何评分信息
  - 公式上来说  $r(i, j) = 0$  则只剩正则项，正则项=0时最小，所以  $\theta(\text{eve}) = 0$

## Users who have not rated any movies

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)
Love at last	5	5	0	0	?
Romance forever	5	?	?	0	?
Cute puppies of love	?	4	0	?	?
Nonstop car chases	0	0	5	4	?
Swords vs. karate	0	0	5	?	?

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{(i,j): r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

$n=2 \quad \underline{\Theta}^{(s)} \in \mathbb{R}^2 \quad \underline{\Theta}^{(s)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\frac{\lambda}{2} \left[ (\underline{\Theta}_1^{(s)})^2 + (\underline{\Theta}_2^{(s)})^2 \right] \leftarrow$

$(\underline{\Theta}^{(s)})^T \underline{x}^{(i)} = 0$

Andrew Ng

- 均值归一化
  - 对  $Y$  每一行求平均，得到  $\mu$ ，再令  $Y - \mu$  并把他当作新的  $Y$
  - 对于用户  $j$  对电影  $i$  的评分预测： $(\underline{\Theta}^{(i)})^T (\underline{x}^{(i)}) + \mu$
  - 用户 5 Eve 的  $\theta$  就可以得到了

## Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \rightarrow \underline{Y} = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

For user  $j$ , on movie  $i$  predict:

$$\rightarrow (\underline{\Theta}^{(i)})^T (\underline{x}^{(i)}) + \mu_i$$

learn  $\underline{\Theta}^{(i)}, \underline{x}^{(i)}$

User 5 (Eve):

$$\underline{\Theta}^{(s)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\underbrace{(\underline{\Theta}^{(s)})^T (\underline{x}^{(i)})}_{\sim 0} + \boxed{\mu_i}$$

Andrew Ng

## 18. 大规模机器学习

处理海量数据的方法，接下来的内容因为时间关系，所以只有简单的会议笔记

### 1. 随机梯度下降和 Mini-batch 梯度下降法

- 机器学习流程
  - 构建函数  $h(\theta) = \Sigma \theta_j x_j$
  - 获得代价函数  $J_{train}(\theta) = \dots$
  - 梯度下降法：Repeat {
    - $\Theta_j = \theta_j - \alpha 1/m (\Sigma (\theta_j x_j - y_j))$
  - 注意：这样的结果就是每重复一次都要将所有的  $\theta \times x$  做一次方差。如果有一亿个数据，那么就要重复读取一亿的平方次

数据。

- 批量随机下降法
  - 第一步，将数据打乱随机分布
  - 第二步，读取每一个数据，进行一次计算  $cost\ function = \delta(J_{train}(\theta)) / \delta\theta_j$
  - 第三步，读取下一个数据，读一个数据优化一次算法
  - 虽然，可能有些数据读取之后会让算法偏离costfunction减小的方向，但总的来说都是，当数据量足够大，就会正确优化算法。
  - 优点就是，只需要读取一遍数据就可以了！
- Mini-batch梯度下降法
  - 与批量随机下降法的区别就是，他每次选取的数据不是一个，而是 $b$ 个数据为一组（比如10个）。
  - 优点：比普通梯度下降法快速。
  - 比随机下降法慢，但是如果好的向量计算方法，可以比随机更快

## 2. 学习速率 $\alpha$

- 在计算的过程中，需要检查计算是否收敛！
- 对于批量下降法，只有等数据全部遍历结束之后才能检查。
- 但对于随机下降法，可以计算前1000个数据后→画出costfunciton图像观察是否降低
- 小的 $\alpha$ 会使随机下降法收敛更好，但是需要的计算时间更长！
- 所以可以考虑设计 $\alpha$ 的数值随着遍历进行而减小  $\alpha = \frac{const1}{iterationNumber+const2}$  的值，不过需要自己认真处理const1和const2的值

## 3. 在线学习

- 所谓在线学习呢，就是在线收集用户的信息，并随时对模型进行优化
- 比如：快递公司显示价格和留住用户的关系；搜索商品并显示用户最想要的产品
- 该方法与梯度下降一致，唯一的区别就是，对于用户的搜索信息不做保留，用完即丢
- 优点：可以随时间，人群，潮流等不断进化算法

# 19. 应用举例OCR

## 1. Mapreduce

- 作用：减少映射数据，并行计算，用于解决更大规模问题
- 奠基人：Jeffrey Dean (Google) and Sanjay Ghemawat
- 比如400个数据  $m = 400$

$$\left\{ \begin{array}{l} Machine1 : 1 100 \\ Machine2 : 101 200 \\ Machine3 : 201 300 \\ Machine4 : 301 400 \end{array} \right.$$

- 此时  $\Theta_j = \Theta_j - \alpha \frac{1}{400} (M_1 + M_2 + M_3 + M_4)$
- 将上述400个实例分给四个机器并行运算，可以加快计算速度
- 注意：如果使用线性代数库，向量化计算较好的，则不需要使用Mapreduce方法，因为线性代数库本身会实现类似功能

## 2. 照片OCR(Photo Optical Charater Recognition)

,

1. Text detection
2. Character Segmentation
3. Character Classification

,

- 滑动窗口分类器

- 第一步，准备有行人的照片 ***positive example ( $y = 1$ )*** 和无行人的照片 ***negative example ( $y=0$ )***，进行机器学习训练
- 第二步，设定小窗口从左到右，从上到下依次扫描图片，并分析与  $y=1$  相近的概率。逐步增加窗口，重复扫描。最终获得概率图像(高亮表示该处图像有行人的概率大)
- 应用在识别文字上同理
  - 训练机器学习模型 (PE, NE)
  - 滑动窗口找到可能存在字迹的部位 (概率图像)
    - 高亮概率高的位置附近，利用滑动窗口分类器，对该位置的字符进行分割成单个字母
  - 字符分类：将字母识别出来

### 3. 人工数据合成

- 思考如何将已有的数据量扩大十倍？
- 如训练数据不够的话，一种可用的方法：将电脑字体库中的字母随机选取，并将其粘贴到不同的背景图片中去
- 或者对于已有的真实数据，进行变形处理，放大弯曲等。

### 4. 上限分析：合理的对模型的不同功能分配不同的任务量

-以OCR为例

```
\begin{array}{c|c|l}
&&\\
&&\\
\hline
\text{overall system} & \{72\%} & \\
\text{Text detection} & \{89\%} & \{\uparrow 17\%} \\
\text{Character segmental} & \{90\%} & \{\uparrow 1\%} \\
\text{Character classification} & \{100\%} & \{\uparrow 10\%} \\
\end{array}
```

- 步骤是逐步将上述三项调节到100%准确，比如手动选择有字母的部分，那么Text detection一项就是100%准确了。然后观察这样可以将总系统的准确度提升多少。
- 比如第一步可以提升17%，第二步有1%的提升空间，第三步有10%的提升空间，可见，第一步才是最需要努力优化的部分。