

目录 - 所有分区页面

- 目录

- 目录 - 所有分区页面
- 目录 - 分区

- 监督学习

- (1) 监督学习
- (2) 线性回归
- (3) 正规方程
- (4) 正规方程与梯度下降算法对比
- (5) 分类
- (6) logistic回归算法
- (7) 简化logistic回归算法中代价函数和梯度下降
- (8) 高级优化
- (9) 多元分类
- (10) 过拟合问题
- (11) 代价函数
- (12) 线性回归的正则化
- (13) 逻辑回归的正则化

- 神经网络

- (01) 神经网络
- (02) 非线性假设
- (03) 神经模型
- (04) 多元分类
- (05) 代价函数
- (06) 反向传播

- [\(07\) 反向传播的直观理解](#)
- [\(08\) 反向传播算法的理解和推导](#)
- [\(09\) 展开参数](#)
- [\(10\) 梯度检测](#)
- [\(11\) 随机初始化](#)
- [\(12\) 总结](#)
- [机器学习算法诊断](#)
 - [\(1\) 机器学习算法诊断](#)
 - [\(2\) 线性回归评估假设](#)
 - [\(3\) 逻辑回归评估假设](#)
 - [\(4\) 模型选择、验证、测试集](#)
 - [\(5\) 诊断偏差、方差](#)
 - [\(6\) 正则化和偏差、方差](#)
 - [\(7\) 学习曲线](#)
 - [\(8\) 总结](#)
- [误差分析](#)
 - [\(1\) 误差分析](#)
 - [\(2\) 不对称性分类的误差评估](#)
 - [\(3\) 精确度和召回率的平衡](#)
 - [\(4\) 机器学习数据](#)
- [支持向量机](#)
 - [\(1\) 支持向量机](#)
 - [\(2\) 直观上对于大间隔的理解](#)
 - [\(3\) 大间隔分类器的数学原理](#)
 - [\(4\) 核函数 1](#)
 - [\(5\) 核函数 2](#)
 - [\(6\) 使用SVM](#)
- [无监督学习](#)
 - [\(1\) 无监督学习](#)

- [\(2\) K-Means\(k-均值\) 算法](#)
- [\(3\) 优化目标](#)
- [\(4\) 随机初始化](#)
- [\(5\) 选取聚类的数量](#)
- [主成分分析法](#)
 - [\(1\) 数据压缩](#)
 - [\(2\) 可视化](#)
 - [\(3\) 主成分分析方法\(PCA\) 1](#)
 - [\(4\) 主成分分析方法\(PCA\) 2](#)
 - [\(5\) 主成分的数量选择](#)
 - [\(6\) 压缩重构\(压缩还原\)](#)
 - [\(7\) 应用PCA建议](#)
 - [\(8\) PCA的数学原理](#)
- [异常检测算法](#)
 - [\(1\) 异常检测问题](#)
 - [\(2\) 高斯分布\(正态分布\)](#)
 - [\(3\) 异常检测算法](#)
 - [\(4\) 评估异常检测系统](#)
 - [\(5\) 异常学习VS监督学习](#)
 - [\(6\) 异常检测算法的特征选择](#)
 - [\(7\) 多变量高斯分布](#)
 - [\(8\) 使用多变量高斯分布的异常检测](#)
- [推荐算法](#)
 - [\(1\) 推荐问题](#)
 - [\(2\) 基于内容的推荐算法](#)
 - [\(3\) 协同过滤](#)
 - [\(4\) 协同过滤算法](#)
 - [\(5\) 矢量化:低秩矩阵分解](#)
 - [\(6\) 均值归一化](#)

- 大数据集

- (1) 学习大数据集
- (2) 随机梯度下降
- (3) Mini-Batch梯度下降
- (4) 随机梯度下降收敛
- (5) 在线学习
- (6) 减少映射与数据并行

- 照片OCR问题

- (1) 照片OCR问题与机器学习流水线
- (2) 滑动窗口

(1) 监督学习

2022年1月22日 18:59

机器学习主要有两种算法即监督学习和无监督学习

监督学习主要是指通过已知的数据去预测已经包括正确答案的数据，主要有以下几种算法

- 线性回归
- 分类
- 神经网络
- 支持向量机

(2) 线性回归

2022年1月4日 13:28

线性回归——通过已知的数据去预测给定的数据对应的键值，如房屋大小-价格

一. 模型构成

1. 假设函数——假设一个函数为其已知数据的函数
2. 代价函数——通常是指实际函数与假设函数的平方差之和
3. 目标函数——是指每个假设函数与其代价函数的最小差

Hypothesis:

$$\underline{h_{\theta}(x) = \theta_0 + \theta_1 x}$$

Parameters:

$$\underline{\theta_0, \theta_1}$$



Cost Function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$

二. 梯度下降算法——求最小代价函数的一种算法，通过不断变幻参数使得找到最小代价函数，定义如下(可推广到多元函数):

Have some function $J(\theta_0, \theta_1)$

Want $\underset{\theta_0, \theta_1}{\text{min}} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$
until we hopefully end up at a minimum

1. 算法概要

Gradient descent algorithm

```
repeat until convergence {  
     $\rightarrow \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$  (for  $j = 0$  and  $j = 1$ )  
}
```

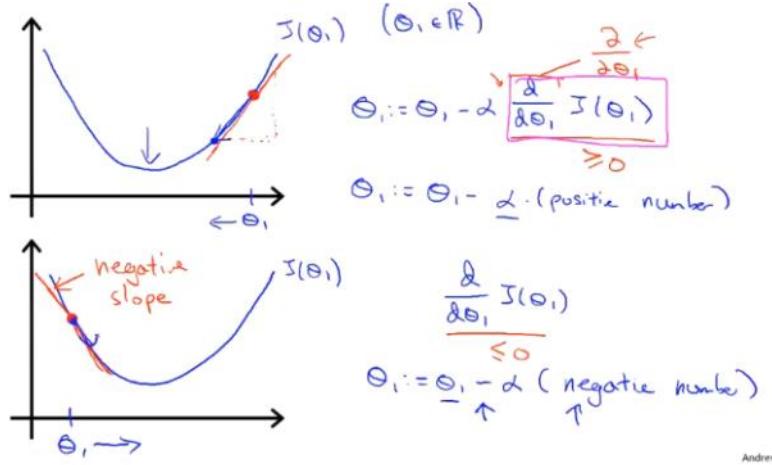
Correct: Simultaneous update

```
temp0 :=  $\theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$   
temp1 :=  $\theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$   
 $\theta_0 := \text{temp0}$   
 $\theta_1 := \text{temp1}$ 
```

注: α 表学习率表明下降得多快

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \text{ 表示偏导数}$$

2. 算法详解



当 θ_1 在右边时，由于导数为 x 轴的 Δx 与 y 轴的 Δy 的比值，又由于 θ_1 在右边，故 Δx 与 Δy 皆为正数所以其导数为正

当 θ_1 在左边时，由于 Δx 为正数与 Δy 为负数，帮其导数为负数

所以上面公式 $\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$ ，无论 θ_1 在其极小值的左边或右边皆向极小值逼近。

此算法通过不断迭代直到极小值，当 θ_1 为极小值点时其导数斜率趋向于某个值，此时斜率发生的变化极小。

三. 梯度下降算法在线性回归中的应用

1. 偏导数如下

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) &= \frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \\ &= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^m (\underline{\theta_0 + \theta_1 x^{(i)}} - y^{(i)})^2 \\ \theta_0, j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) &= \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}), \\ \theta_1, j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) &= \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}\end{aligned}$$

2. 算法概要

Gradient descent algorithm

```

repeat until convergence {
     $\theta_0 := \theta_0 - \alpha \left[ \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \right]$ 
     $\theta_1 := \theta_1 - \alpha \left[ \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)} \right]$ 
}

```

$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

update θ_0 and θ_1 simultaneously

四. 多元回归——指通过多元变量进行线性回归

1. 定义

Hypothesis: $h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \dots, \theta_n$ ⊕ $n+1$ -dimensional vector

Cost function:

$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

Repeat {

$$\rightarrow \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \dots, \theta_n)$$

} (simultaneously update for every $j = 0, \dots, n$)

Andrew Ng

$$\text{其中 } J(\theta) \text{ 是指矩阵向量即 } \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

2. 多元梯度下降算法——把单元梯度下降算法推广到多维

Gradient Descent

Previously ($n=1$):

Repeat {

$$\rightarrow \theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial \theta_0} J(\theta)}$$

$$\rightarrow \theta_1 := \theta_1 - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})}_{\text{(simultaneously update } \theta_0, \theta_1\text{)}} x_j^{(i)}$$

}

New algorithm ($n \geq 1$):

Repeat {

$$\rightarrow \theta_j := \theta_j - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}}_{\text{(simultaneously update } \theta_j \text{ for } j = 0, \dots, n\text{)}}$$

}

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

...

Andrew Ng

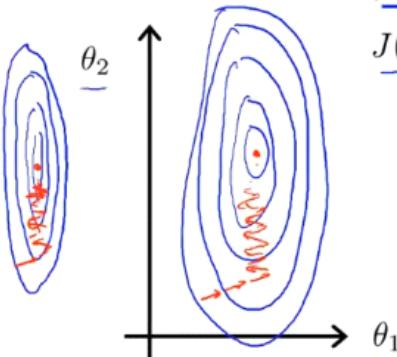
- a) 特征缩放——把不同特征的值都缩放在一个相近的范围，会使得梯度下降算法更加快速的收敛

Feature Scaling

Idea: Make sure features are on a similar scale.

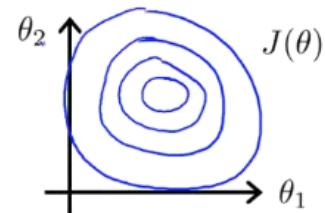
E.g. $x_1 = \text{size (0-2000 feet}^2)$ ←

$x_2 = \text{number of bedrooms (1-5)}$ ←



$$\Rightarrow x_1 = \frac{\text{size (feet}^2)}{2000}$$

$$\Rightarrow x_2 = \frac{\text{number of bedrooms}}{5}$$



Andrew

- b) 均值归一化——把不同特征减去它们特征的平均值，使之缩放在一个相近的范围

Mean normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean
(Do not apply to $x_0 = 1$).

E.g. $x_1 = \frac{\text{size}-1000}{2000}$

Average size = 100

$x_2 = \frac{\#\text{bedrooms}-2}{5}$

1-5 bedrooms

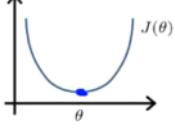
$$-0.5 \leq x_1 \leq 0.5, -0.5 \leq x_2 \leq 0.5$$

(3) 正规方程

2022年1月5日 16:45

一. 定义——线性回归的一种算法，用于解决多元变量函数求代价函数中最小代价函数的问题，其原理是用于微积分中的多元微分函数中求偏导，联立方程式求最小值问题。

Intuition: If 1D ($\theta \in \mathbb{R}$)
 $J(\theta) = a\theta^2 + b\theta + c$
 $\frac{\partial}{\partial \theta} J(\theta) = \dots \text{ set } 0$
 Solve for θ



$\theta \in \mathbb{R}^{n+1}$ $J(\theta_0, \theta_1, \dots, \theta_m) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$
 $\frac{\partial}{\partial \theta_j} J(\theta) = \dots \text{ set } 0 \quad (\text{for every } j)$
 Solve for $\theta_0, \theta_1, \dots, \theta_n$

二. 例子

Examples: $m = 4$.

x_0	Size (feet ²)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}_{m \times (n+1)}$

$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}_{m \text{-dimensional vector}}$

$\theta = (X^T X)^{-1} X^T y$

如上图所示通过训练集构建X矩阵，通过结果集构建Y矩阵暂时不清楚为什么要加 x_0 通过 $\theta = (x^T x)^{-1} x^T y$ 公式求得最小值。

三. 求证 $\theta = (x^T x)^{-1} x^T y$

设多变量线性回归代价函数为 $j(\theta_0 \ \theta_1 \ \dots \ \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$

其中 $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n$

又由正规方程通过求解下面方程来找出使得代价函数的最小参数

$$\frac{\partial}{\partial \theta_j} J(\theta_j) = 0$$

设有m个训练实例，每个n实例有特征，则训练实例集为：

$$X = \begin{pmatrix} x_0^{(1)} & \dots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_0^{(m)} & \dots & x_n^{(m)} \end{pmatrix}$$

其中 $x_j^{(i)}$ 表示第i个实例第j个特征

特征参数为

$$\theta = [\theta_0, \theta_1, \dots, \theta_n]^T$$

输出变量为

$$Y = [y^1, y^2, \dots, y^n]^T$$

故代价函数为

$$j(\theta_0 \ \theta_1 \ \dots \ \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

由 $\pi = \begin{pmatrix} \pi_0^{(1)} & \dots & \pi_n^{(1)} \\ \vdots & \ddots & \vdots \\ \pi_0^{(m)} & \dots & \pi_n^{(m)} \end{pmatrix}$ 生产工时数 $\theta = [\theta_0, \theta_1, \dots, \theta_n]^T$

车间出货量为 $\gamma = \begin{pmatrix} \gamma^{(1)} \\ \gamma^{(2)} \\ \vdots \\ \gamma^{(m)} \end{pmatrix} \Rightarrow \theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{pmatrix}$

$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (\hat{h}_0(\pi^{(i)}) - \gamma^{(i)})^2$

$= \frac{1}{2m} \sum_{i=1}^m (\hat{h}_0(\pi^{(i)}) - \gamma^{(i)}) (\hat{h}_0(\pi^{(i)}) - \gamma^{(i)})$

其中 $\hat{h}_0(\pi^{(i)}) = \theta^T \pi^{(i)} = \theta_0 \pi_0 + \theta_1 \pi_1 + \dots + \theta_n \pi_n$

其中 $\pi^{(i)}$ 表示第 i 个样本的 n 个特征。

故 $J = \frac{1}{2m} \sum_{i=1}^m (\theta^T \pi^{(i)} - \gamma^{(i)}) (\theta^T \pi^{(i)} - \gamma^{(i)})$

$\therefore J = \frac{1}{2m} \left[(\theta^T \pi^{(1)} - \gamma^{(1)})^2 + (\theta^T \pi^{(2)} - \gamma^{(2)})^2 + \dots + (\theta^T \pi^{(m)} - \gamma^{(m)})^2 \right]$

有 $J = \frac{1}{2m} \left[((\theta_0, \theta_1, \dots, \theta_n) \begin{pmatrix} \pi_0^{(1)} \\ \vdots \\ \pi_n^{(1)} \end{pmatrix} - \gamma^{(1)})^2 + \dots + ((\theta_0, \theta_1, \dots, \theta_n) \begin{pmatrix} \pi_0^{(m)} \\ \vdots \\ \pi_n^{(m)} \end{pmatrix} - \gamma^{(m)})^2 \right]$

$= \frac{1}{2m} \left[((\theta_0, \theta_1, \dots, \theta_n) \begin{pmatrix} \pi_0^{(1)} \\ \vdots \\ \pi_n^{(1)} \end{pmatrix} - \gamma^{(1)})^2 + ((\theta_0, \theta_1, \dots, \theta_n) \begin{pmatrix} \pi_0^{(2)} \\ \vdots \\ \pi_n^{(2)} \end{pmatrix} - \gamma^{(2)})^2 + \dots + ((\theta_0, \theta_1, \dots, \theta_n) \begin{pmatrix} \pi_0^{(m)} \\ \vdots \\ \pi_n^{(m)} \end{pmatrix} - \gamma^{(m)})^2 \right]$

Güneş

$$\text{X} \in \mathbb{C}[\theta_0, \theta_1, \dots, \theta_n] \left(\begin{array}{c} x_0^{(0)} \\ \vdots \\ x_n^{(0)} \end{array} \right) - Y^{(0)} \text{ 为数.}$$

$$\Rightarrow A_1$$

$$\Re \left[\left(\theta_0, \theta_1, \dots, \theta_n \right) \left(\begin{array}{c} x_0^{(1)} \\ \vdots \\ x_n^{(1)} \end{array} \right) - Y^{(1)} \right]^2 = A_1^2.$$

$$\Re \left[\Im \left[\left(\theta_0, \theta_1, \dots, \theta_n \right) \left(\begin{array}{c} x_0^{(1)} \\ \vdots \\ x_n^{(1)} \end{array} \right) - Y^{(1)} \right]^2 + \dots \right. \\ \left. \dots \left(\left(\theta_0, \theta_1, \dots, \theta_n \right) \left(\begin{array}{c} x_0^{(n)} \\ \vdots \\ x_n^{(n)} \end{array} \right) - Y^{(n)} \right)^2 \right]$$

$$\Rightarrow \Re \left[A_1^2 + \dots + A_n^2 \right].$$

$$\Rightarrow \frac{1}{2n} \left(A_1, A_2, \dots, A_n \right) \underbrace{\begin{pmatrix} A_1 \\ \vdots \\ A_n \end{pmatrix}}_{\rightarrow V^T} \underbrace{\begin{pmatrix} A_1 \\ \vdots \\ A_n \end{pmatrix}}_{\rightarrow V}$$
~~$$\Re \left[V^T \left(\left(\theta_0, \theta_1, \dots, \theta_n \right) \left(\begin{array}{c} x_0^{(0)} \\ \vdots \\ x_n^{(0)} \end{array} \right) - Y^{(0)} \right) \dots \right. \\ \left. \dots \left(\theta_0, \theta_1, \dots, \theta_n \right) \left(\begin{array}{c} x_0^{(n)} \\ \vdots \\ x_n^{(n)} \end{array} \right) - Y^{(n)} \right) \right]$$~~
~~$$\cdot \left(\theta_0, \theta_1, \dots, \theta_n \right) \left(\begin{array}{c} x_0^{(n)} \\ \vdots \\ x_n^{(n)} \end{array} \right) - Y^{(n)} \right)^T$$~~

$$\Re \left[V = \begin{pmatrix} A_1 \\ \vdots \\ A_n \end{pmatrix} = \left(\begin{array}{c} (\theta_0, \theta_1, \dots, \theta_n) \left(\begin{array}{c} x_0^{(0)} \\ \vdots \\ x_n^{(0)} \end{array} \right) - Y^{(0)} \\ \vdots \\ (\theta_0, \theta_1, \dots, \theta_n) \left(\begin{array}{c} x_0^{(n)} \\ \vdots \\ x_n^{(n)} \end{array} \right) - Y^{(n)} \end{array} \right) \right]$$
~~$$\Re \left[V = X \otimes \right]$$~~

$$= \begin{pmatrix} \theta_0 \cdot x_0^{(0)} + \theta_1 \cdot x_1^{(0)} + \dots + \theta_n \cdot x_n^{(0)} - Y^{(0)} \\ \vdots \\ \theta_0 \cdot x_0^{(n)} + \theta_1 \cdot x_1^{(n)} + \dots + \theta_n \cdot x_n^{(n)} - Y^{(n)} \end{pmatrix}$$

Given $\forall j \quad Y_j = X\theta - Y$

(ii) $J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} (\mathbf{X}\theta - \mathbf{Y})^T (\mathbf{X}\theta - \mathbf{Y})$

$$\times (\mathbf{X}\theta - \mathbf{Y})^T = \theta^T \mathbf{X}^T - \mathbf{Y}^T$$

(iv) $J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} [(\theta^T \mathbf{X}^T - \mathbf{Y}^T) (\mathbf{X}\theta - \mathbf{Y})]$
 $= \frac{1}{2m} [\mathbf{Y}^T \mathbf{Y} - \mathbf{Y}^T \mathbf{X}\theta - \theta^T \mathbf{X}^T \mathbf{Y} + \theta^T \mathbf{X}^T \mathbf{X}\theta]$

(v) $(\mathbf{X}\theta - \mathbf{Y})^T = \theta^T \mathbf{X}^T - \mathbf{Y}^T$

$$\mathbf{X}\theta - \mathbf{Y} = \begin{pmatrix} x_0^{(1)} & x_1^{(1)} & \dots & x_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_0^{(m)} & x_1^{(m)} & \dots & x_n^{(m)} \end{pmatrix} \begin{pmatrix} \theta_0 \\ \vdots \\ \theta_n \end{pmatrix} - \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{pmatrix}$$

Given $\mathbf{X}\theta - \mathbf{Y} = \begin{pmatrix} x_0^{(1)} & x_1^{(1)} & \dots & x_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_0^{(m)} & x_1^{(m)} & \dots & x_n^{(m)} \end{pmatrix} \begin{pmatrix} \theta_0 \\ \vdots \\ \theta_n \end{pmatrix} - \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{pmatrix}$

$$= \begin{pmatrix} x_0^{(1)}\theta_0 + x_1^{(1)}\theta_1 + \dots + x_n^{(1)}\theta_n - y^{(1)} \\ \vdots \\ x_0^{(m)}\theta_0 + x_1^{(m)}\theta_1 + \dots + x_n^{(m)}\theta_n - y^{(m)} \end{pmatrix}$$

(vi) $(\mathbf{X}\theta - \mathbf{Y})^T =$

$\therefore \forall j \quad Y_j = \gamma_0^{(j)}\theta_0 + \gamma_1^{(j)}\theta_1 + \dots + \gamma_n^{(j)}\theta_n$

\vdots

$\gamma_m = \gamma_0^{(m)}\theta_0 + \gamma_1^{(m)}\theta_1 + \dots + \gamma_n^{(m)}\theta_n$

Cue

$$\text{Right side} = \begin{pmatrix} x_1 - y^{(1)} \\ x_2 - y^{(2)} \\ \vdots \\ x_m - y^{(m)} \end{pmatrix}$$

$$\Rightarrow (x\theta - y)^T = (x_1 - y^{(1)}, x_2 - y^{(2)}, \dots, x_m - y^{(m)})$$

$$x \cdot \theta = \begin{pmatrix} \theta_0 \\ \vdots \\ \theta_n \end{pmatrix}, x^{(i)} = \begin{pmatrix} x_0^{(i)} \\ \vdots \\ x_n^{(i)} \end{pmatrix}$$

$$x_1 = x_0^{(1)} \theta_0 + \dots + x_n^{(1)} \theta_n = \theta^T x^{(1)}$$

$$\Rightarrow (x\theta - y)^T = (\theta^T x^{(1)} - y^{(1)}, \dots, \theta^T x^{(m)} - y^{(m)})$$

$$= \underbrace{\begin{pmatrix} 1 & \dots & 1 \end{pmatrix}}_{(1 \times m)} \underbrace{\begin{pmatrix} \theta^T x^{(1)} - y^{(1)} \\ \vdots \\ \theta^T x^{(m)} - y^{(m)} \end{pmatrix}}_{(\theta^T x)}$$
~~$$= \underbrace{\begin{pmatrix} 1 & \dots & 1 \end{pmatrix}}_{(1 \times m)} \left[\begin{pmatrix} \theta^{(1)} \\ \theta^{(2)} \\ \vdots \\ \theta^{(m)} \end{pmatrix} - \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{pmatrix} \right]$$~~

$$= \underbrace{\begin{pmatrix} 1 & \dots & 1 \end{pmatrix}}_{(1 \times m)} \left[\begin{pmatrix} \theta^T x^{(1)} \\ \theta^T x^{(2)} \\ \vdots \\ \theta^T x^{(m)} \end{pmatrix} - \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{pmatrix} \right]$$

$$= (\theta^T x^{(1)}, \dots, \theta^T x^{(m)} - (y^{(1)} \dots y^{(m)}))$$

$$= \theta^T x^T - y^T$$

$$x \frac{\partial \mathbf{J}}{\partial \theta} = \frac{1}{m} \left(\frac{\partial \mathbf{y}^T \mathbf{y}}{\partial \theta} - \frac{\partial \mathbf{y}^T \mathbf{x}\theta}{\partial \theta} - \frac{\partial \theta^T \mathbf{A}^T \mathbf{y}}{\partial \theta} + \frac{\partial \theta^T \mathbf{A}^T \mathbf{x}\theta}{\partial \theta} \right)$$

$$\begin{aligned} \text{Cue} & \quad \text{第三项} \\ \theta^T \gamma^T Y &= (\theta_0, \theta_1, \dots, \theta_n) \begin{pmatrix} \gamma_0^{(1)} & \dots & \gamma_n^{(1)} \\ \vdots & \ddots & \vdots \\ \gamma_0^{(m)} & \dots & \gamma_n^{(m)} \end{pmatrix}^T \begin{pmatrix} Y^{(1)} \\ Y^{(2)} \\ \vdots \\ Y^{(m)} \end{pmatrix}^T \\ &= (\gamma_0^{(1)}\theta_0 + \dots + \gamma_n^{(1)}\theta_n) Y^{(1)} + (\gamma_0^{(2)}\theta_0 + \dots + \gamma_n^{(2)}\theta_n) Y^{(2)} \\ & \quad + \dots + (\gamma_0^{(m)}\theta_0 + \dots + \gamma_n^{(m)}\theta_n) Y^{(m)} \end{aligned}$$

1 2 3 4

又第1项 $\frac{\partial \gamma^T \gamma}{\partial \theta} =$

第二项

$$\gamma^T \gamma \theta = [\gamma^{(1)}, \gamma^{(2)}, \dots, \gamma^{(m)}] \begin{pmatrix} \gamma_0^{(1)} & \dots & \gamma_m^{(1)} \\ \vdots & \ddots & \vdots \\ \gamma_0^{(m)} & \dots & \gamma_m^{(m)} \end{pmatrix} (\theta_0, \theta_1, \dots, \theta_n)^T$$

$$= (\gamma_0^{(1)} y^{(1)}) + \dots + \gamma_0^{(m)} y^{(m)} \theta_0 + (\gamma_1^{(1)} y^{(1)} + \dots + \gamma_1^{(m)} y^{(m)}) \theta_1 + \dots + (\gamma_m^{(1)} y^{(1)} + \dots + \gamma_m^{(m)} y^{(m)}) \theta_n$$

$$\Rightarrow \frac{\partial \gamma^T \gamma \theta}{\partial \theta} = \begin{pmatrix} \frac{\partial \gamma^T \gamma \theta}{\partial \theta_0} \\ \frac{\partial \gamma^T \gamma \theta}{\partial \theta_1} \\ \vdots \\ \frac{\partial \gamma^T \gamma \theta}{\partial \theta_n} \end{pmatrix} = \begin{pmatrix} \gamma_0^{(1)} y^{(1)} + \dots + \gamma_0^{(m)} y^{(m)} \\ \gamma_1^{(1)} y^{(1)} + \dots + \gamma_1^{(m)} y^{(m)} \\ \vdots \\ \gamma_m^{(1)} y^{(1)} + \dots + \gamma_m^{(m)} y^{(m)} \end{pmatrix} = \gamma^T \gamma.$$

即对上面式子分别偏导

$$\begin{aligned} & \left(\gamma_0^{(1)} \dots \gamma_m^{(m)} \right) / \dots \\ & = (\gamma_0^{(1)} \theta_0 + \dots + \gamma_m^{(1)} \theta_m) \gamma^{(1)} + (\gamma_0^{(2)} \theta_0 + \dots + \gamma_m^{(2)} \theta_m) \gamma^{(2)} \\ & \quad + \dots + (\gamma_0^{(m)} \theta_0 + \dots + \gamma_m^{(m)} \theta_m) \gamma^{(m)} \\ & \Rightarrow \frac{\partial \gamma^T \gamma \theta}{\partial \theta} = \begin{pmatrix} \frac{\partial \gamma^T \gamma \theta}{\partial \theta_0} \\ \frac{\partial \gamma^T \gamma \theta}{\partial \theta_1} \\ \vdots \\ \frac{\partial \gamma^T \gamma \theta}{\partial \theta_n} \end{pmatrix} = \begin{pmatrix} \gamma_0^{(1)} \gamma^{(1)} + \dots + \gamma_0^{(m)} \gamma^{(m)} \\ \gamma_1^{(1)} \gamma^{(1)} + \dots + \gamma_1^{(m)} \gamma^{(m)} \\ \vdots \\ \gamma_m^{(1)} \gamma^{(1)} + \dots + \gamma_m^{(m)} \gamma^{(m)} \end{pmatrix} = \gamma^T \gamma. \end{aligned}$$

第四项

$$\theta^T \gamma^T \gamma \theta = (\theta_0, \theta_1, \dots, \theta_n) \underbrace{\begin{pmatrix} \gamma_0^{(1)} & \dots & \gamma_m^{(1)} \\ \vdots & \ddots & \vdots \\ \gamma_0^{(m)} & \dots & \gamma_m^{(m)} \end{pmatrix}}_{m \times m} \underbrace{\begin{pmatrix} \gamma_0^{(1)} & \dots & \gamma_m^{(1)} \\ \vdots & \ddots & \vdots \\ \gamma_0^{(m)} & \dots & \gamma_m^{(m)} \end{pmatrix}}_{m \times m} \cdot \begin{pmatrix} \theta_0 \\ \vdots \\ \theta_n \end{pmatrix}$$

$$\text{有 } \gamma^T \gamma = \underbrace{\begin{pmatrix} \gamma_0^{(1)} & \gamma_0^{(2)} & \dots & \gamma_0^{(m)} \\ \vdots & \ddots & \ddots & \vdots \\ \gamma_m^{(1)} & \gamma_m^{(2)} & \dots & \gamma_m^{(m)} \end{pmatrix}}_{n \times m} \underbrace{\begin{pmatrix} \gamma_0^{(1)} & \dots & \gamma_m^{(1)} \\ \vdots & \ddots & \vdots \\ \gamma_0^{(m)} & \dots & \gamma_m^{(m)} \end{pmatrix}}_{m \times m}$$

$$\Rightarrow \gamma^T \gamma = (\gamma_0^{(1)} \gamma_1^{(1)} \dots \gamma_m^{(1)})^T$$

$$\Rightarrow \gamma^T \gamma = (\theta_1, \theta_2, \dots, \theta_m) \begin{pmatrix} \frac{\partial}{\partial \theta_1} \\ \frac{\partial}{\partial \theta_2} \\ \vdots \\ \frac{\partial}{\partial \theta_m} \end{pmatrix}$$

由 $\theta^T \gamma^T \gamma \theta = (\theta_0, \theta_1, \dots, \theta_n) (\partial_0 \partial_1^T + \partial_2 \partial_2^T + \dots + \partial_m \partial_m^T) \theta_0 + \theta_1 (\partial_1 \partial_1^T + \partial_2 \partial_2^T + \dots + \partial_m \partial_m^T) \theta_1 + \dots + \theta_n (\partial_n \partial_n^T + \partial_1 \partial_1^T + \dots + \partial_m \partial_m^T) \theta_n.$

$$\Rightarrow \frac{\partial \theta^T \gamma^T \gamma \theta}{\partial \theta} = \begin{pmatrix} \frac{\partial \theta^T \gamma^T \gamma \theta}{\partial \theta_0} \\ \frac{\partial \theta^T \gamma^T \gamma \theta}{\partial \theta_1} \\ \vdots \\ \frac{\partial \theta^T \gamma^T \gamma \theta}{\partial \theta_n} \end{pmatrix} = 2(\gamma^T \gamma) \begin{pmatrix} \theta_0 \\ \vdots \\ \theta_n \end{pmatrix} = 2\gamma^T \gamma \theta$$

由 $J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2n} [\gamma^T \gamma - \gamma^T \gamma \theta - \theta^T \gamma^T \gamma + \theta^T \gamma^T \gamma \theta].$

~~且 $\frac{\partial J}{\partial \theta} = 0$~~

$$\Rightarrow \frac{\partial J}{\partial \theta} = \frac{1}{2n} (-2\gamma^T \gamma + 2\gamma^T \gamma \theta) = 0.$$

$\frac{1}{2n} \neq 0$. 故 $-2\gamma^T \gamma + 2\gamma^T \gamma \theta = 0.$

$$\Rightarrow 2\gamma^T \gamma \theta = 2\gamma^T \gamma$$

$$\gamma^T \gamma \theta = \gamma^T \gamma$$

$$\boxed{\theta = (\gamma^T \gamma)^{-1} \gamma^T \gamma.}$$

当 $\theta = (\gamma^T \gamma)^{-1} \gamma^T \gamma$, 代价函数 $J(\theta)$ 取得最小值.

(4) 正规方程与梯度下降算法对比

2022年1月7日 13:12

一. 两种算法对比

m training examples, n features.

Gradient Descent

- Need to choose α .
- Needs many iterations.
- Works well even when n is large.

Normal Equation

- No need to choose α .
- Don't need to iterate.
- Need to compute $(X^T X)^{-1}$ $n \times n$ $O(n^3)$
- Slow if n is very large.

梯度下降算法：

特点

- 需要挑选学习率 θ
- 需要多次迭代
- 当特征很大时，能够很好的运行

正规方程法：

特点

- 不需要挑选学习率 θ
- 不需要迭代
- 需要计算 $(X^T X)^{-1}$ ，由于 $X^T X$ 是一个 $n \times n$ 的矩阵，所以要计算 $(X^T X)$ 的逆矩阵所以计算这个逆矩阵大约需要 $O(n^3)$

总结：当 n 超过 10000 时应选择梯度下降，否则选择正规方程

二. $X^T X$ 不可逆的情况

What if $X^T X$ is non-invertible?

- Redundant features (linearly dependent).

E.g. $x_1 = \text{size in feet}^2$ $l_m = 3.28 \text{ feet}$

$x_2 = \text{size in m}^2$

$x_1 = (3.28)^2 x_2$

$m = 100$

$n = 100$

$O \in \mathbb{R}^{100 \times 100}$

- Too many features (e.g. $m \leq n$).

- Delete some features, or use regularization.

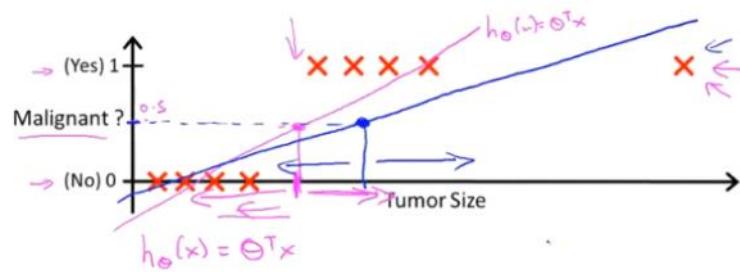
1. 有多余的特征，即两个特征之间具有某种线性相关的关系。

2. 特征过多，需要样本 m 过少，然后特征 n 过多可能会造成不可逆

(5) 分类

2022年1月4日 16:34

一. 定义——通过已知数据对给定的数据进行数据的分类问题



→ Threshold classifier output $h_\theta(x)$ at 0.5:

→ If $h_\theta(x) \geq 0.5$, predict "y = 1"

If $h_\theta(x) < 0.5$, predict "y = 0"

有时线性回归算法并不能准确的对样本集进行分类，如上图所示这是一个差的线性回归

(6) logistic回归算法

2022年1月8日 13:09

一. 起源

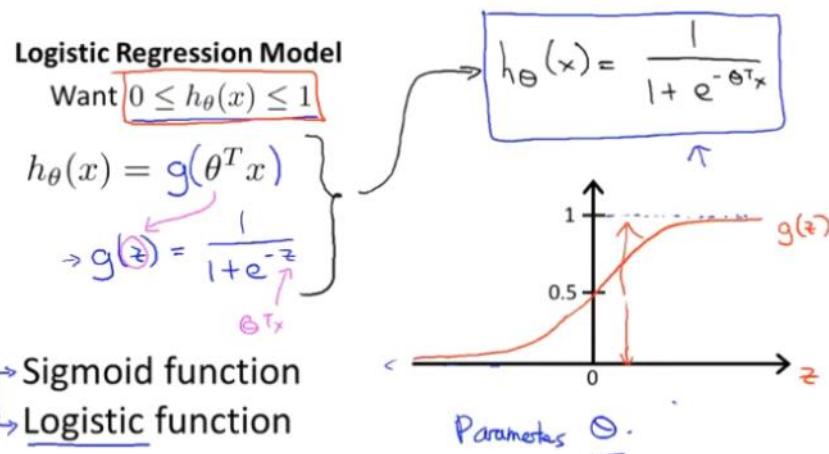
Classification: $y = \underbrace{0}_{\nwarrow} \text{ or } \underbrace{1}_{\uparrow}$

$h_{\theta}(x)$ can be $\underbrace{> 1}_{\nwarrow} \text{ or } \underbrace{< 0}_{\uparrow}$

Logistic Regression: $0 \leq h_{\theta}(x) \leq 1$
Classification

如上图所示， y 的值为0或1，表示对样本进行分类的两种情况，假设函数 $h_0(x)$ 可能会大于1或者小于0，而logistic回归算法可以使得 $0 \leq h_0(x) \leq 1$

二. 假设函数定义



要使得 $0 \leq h_{\theta}(x) \leq 1$ 使得原来的 $h_{\theta}(x) = \theta^T x$ 变为

$h_{\theta}(x) = g(\theta^T x)$ 使得 $g(z) = \frac{1}{1+e^{-z}}$, 故

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

使得 $0 \leq h_{\theta}(x) \leq 1$

三. 解释假设输出

Interpretation of Hypothesis Output

$$h_{\theta}(x)$$

$h_{\theta}(x)$ = estimated probability that $y = 1$ on input x

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$$\underline{h_{\theta}(x) = 0.7} \quad \underline{y=1}$$

Tell patient that 70% chance of tumor being malignant

$$h_{\theta}(x) = p(y=1|x; \theta)$$

"probability that $y = 1$, given x , parameterized by θ "

$$\underline{y = 0 \text{ or } 1}$$

$$\rightarrow P(y=0|\underline{x}; \theta) + P(y=1|\underline{x}; \theta) = 1$$

$$\rightarrow P(y=0|x; \theta) = 1 - P(y=1|x; \theta)$$

Andrew Ng

假设函数 $h_{\theta}(x)$ 的输出等价于

$$h_{\theta}(x) = p(y=1|x; \theta)$$

表示对于参数 x 和 θ , 使得 $y=1$ 的概率为

$$p(y=1|x; \theta)$$

这是并非固定为 $y=1$ 的概率, 也可以是 $y=0$,

又有

$$p(y=1|x; \theta) + p(y=0|x; \theta) = 1$$

故

$$p(y=0|x; \theta) = 1 - p(y=1|x; \theta)$$

四. 决策界限

1. 原理

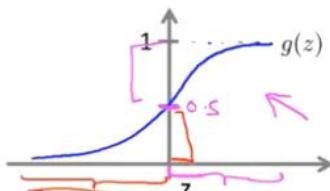
Logistic regression

$$\rightarrow h_{\theta}(x) = g(\theta^T x) = p(y=1|x; \theta)$$

$$\rightarrow g(z) = \frac{1}{1+e^{-z}}$$

Suppose predict " $y = 1$ " if $h_{\theta}(x) \geq 0.5$

$$\rightarrow \theta^T x \geq 0$$



$$g(z) \geq 0.5$$

when $z \geq 0$

$$h_{\theta}(x) = g(\theta^T x) \geq 0.5$$

when $\theta^T x \geq 0$

predict " $y = 0$ " if $h_{\theta}(x) < 0.5$

$$h_{\theta}(x) = g(\theta^T x)$$

$$g(z) < 0.5$$

$$\rightarrow \theta^T x < 0$$

Andrew Ng

假设预测 $y=1$ if $h_{\theta}(x) \geq 0.5$

$$y=0 \text{ if } h_{\theta}(x) < 0.5$$

有函数 $g(z) = g(\theta^T x) = \frac{1}{1+e^{-z}}$ 使得

当 $z \geq 0$ 时, $g(z) = g(\theta^T x) \geq 0.5$

当 $z < 0$ 时, $g(z) = g(\theta^T x) < 0.5$

1) 证明

设有函数 $F(x) = e^{-x}$

$$\text{有 } F(x) = \begin{cases} e^{|x|} & x < 0 \\ e^{-x} & x \geq 0 \end{cases}$$

对 $F(x)$ 求导有

$$F'(x) = \begin{cases} e^{|x|} & x < 0 \\ -e^{-x} & x \geq 0 \end{cases}$$

有当 $x < 0$ 时 $F'(x) > 0$ 单增又 $F(0) = 1$ 所以 $e^{|x|} > 1$

有当 $x \geq 0$ 时 $F'(x) < 0$ 单减，又 $F(0) = 1$ 所以 $e^{-x} < 1$

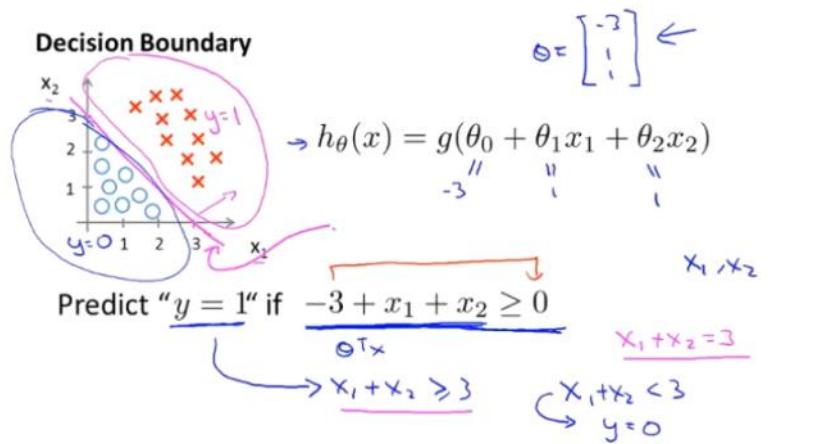
所以有

当 $z \geq 0$ 时, $g(z) = g(\theta^T x) \geq 0.5$

当 $z < 0$ 时, $g(z) = g(\theta^T x) < 0.5$

五. 决策界限定义

1. 定义

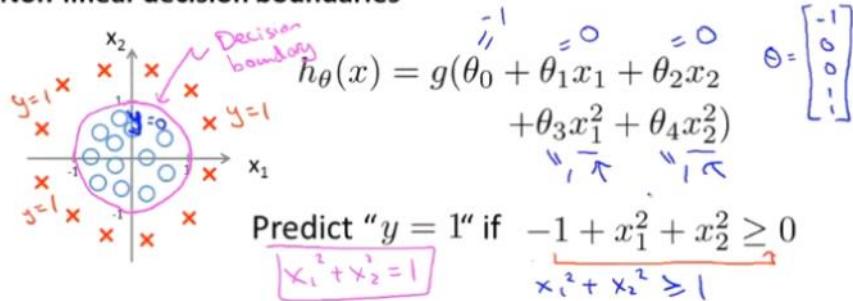


Andrew Ng

如上图所示如若有一个假设函数使得两个不种类型的样本进行分割，则对于这个假设函数所表示的界限称为决策界限。

2. 非线性决策界限定义

Non-linear decision boundaries



如上图所示，有一假设函数，当 θ 为上图所示的值时，有使得方程 $x_1^2 + x_2^2 \geq 1$ 划分样本的不同类别

六. 代价函数

1. 代价函数的定义

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

m examples $x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$ $x_0 = 1, y \in \{0, 1\}$

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

How to choose parameters θ ?

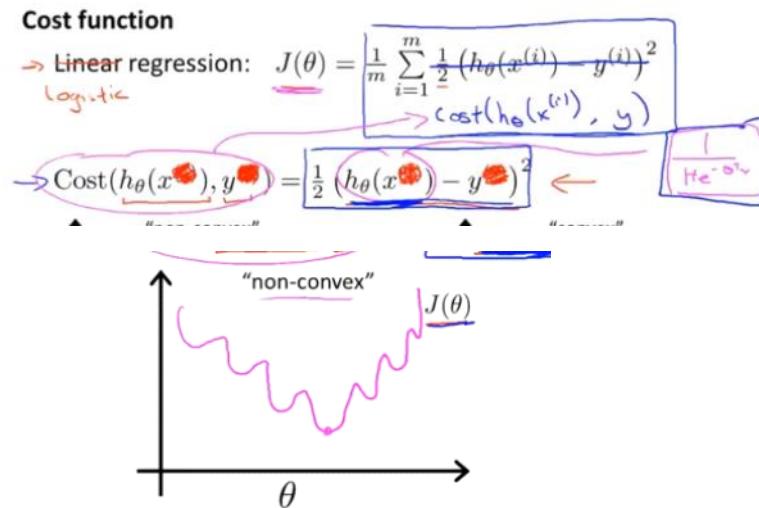
Andrew

Cost function

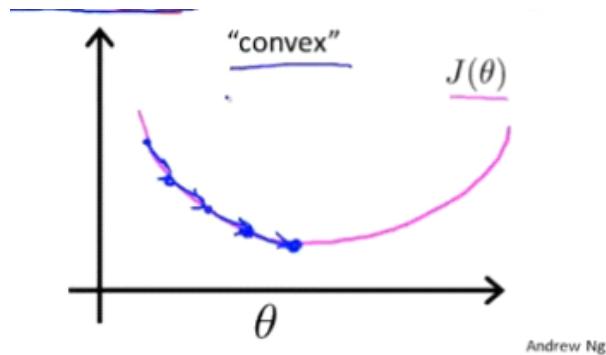
→ Linear regression: $J(\theta) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (h_{\theta}(x^{(i)}) - y^{(i)})^2$

$$\text{Cost}(h_{\theta}(x^{(i)}), y^{(i)}) = \frac{1}{2} (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

当采用上图的这个代价函数时会导致，有多个局部极小值，不好进行梯度下降法求最小值如下图所示



我们需要函数 $J(\theta)$ 变成如下图所示，



Andrew Ng

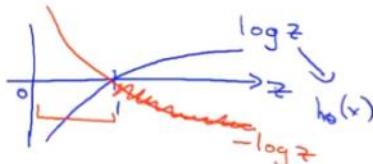
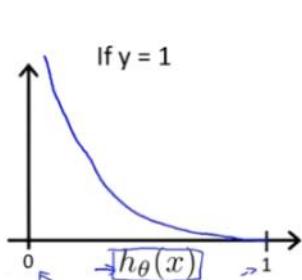
故我们定义代价函数有

$$\bar{J}(\theta) = \frac{1}{m} \sum_{i=1}^m \text{cost}(h_\theta(x^{(i)}), y^{(i)})$$

其中 cost 函数如下

Logistic regression cost function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

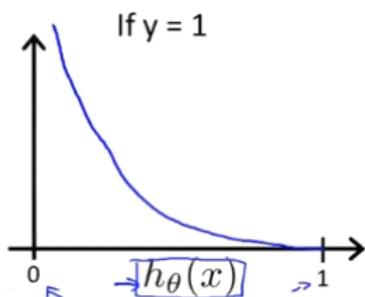


Andrew Ng

有函数

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

使得当 $y=1$ 时，函数图像如下



如下图所示，当 $y=1$ 和 $h_0(x) = 1$ ，即上图的 x 轴为 1 时，cost 为 0，即上图中的 y 由趋向于 0。

但当 $h_0(x) \rightarrow 0$ 时即上图的 x 轴趋向于 0 时，cost 函数趋向 1。

$\text{Cost} = 0$ if $y = 1, h_\theta(x) = 1$

But as $h_\theta(x) \rightarrow 0$

$\text{Cost} \rightarrow \infty$

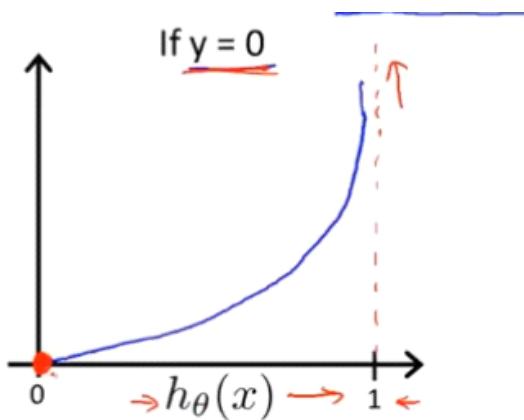
Captures intuition that if $h_\theta(x) = 0$,
(predict $P(y=1|x; \theta) = 0$), but $y = 1$,
we'll penalize learning algorithm by a very
large cost.

如果实际由函数 $h_0(x)$ 预测的为0，但实际上的y值为1，就会使得 $\text{cost} \rightarrow \infty$
这样会使函数得到一个非常大的惩罚值

当 $y=0$ 时，函数 cost

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

有下图



总结：

cost 函数的特点

当 $y=0$ 时，函数 $h_0(x)$ 也预测为0时，其 cost 函数为0，即代价为0，然后随着函数 $h_0(x)$ 的增大，其 cost 函数，即代价也随之增大。

当 $y=1$ 时，函数 $h_0(x)$ 预测也为1时，其代价为0，但随着 $h_0(x)$ 的减少，而代价也随之增大

(7) 简化logistic回归算法中代价函数和梯度下降

2022年1月10日 19:33

Logistic regression cost function

$$\rightarrow J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$\Rightarrow \text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or 1 always

我们可以把cost函数合并到一项，不进行分段，就有如下函数

Logistic regression cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})) \right]$$

- 如何最小化关于 θ 的代价函数 $J(\theta)$? 即寻找 $\text{Min}(J(\theta))$

我们可以运用梯度下降法，不断重复对函数进行求导，使得其 θ 不再变化为止

Gradient Descent

$$\rightarrow J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})) \right]$$

Want $\min_{\theta} J(\theta)$:

$$\left[\begin{array}{l} \text{Repeat } \{ \\ \quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \\ \quad \} \qquad \qquad \text{(simultaneously update all } \theta_j \text{)} \end{array} \right]$$

即恒等于下图的等式

Gradient Descent

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})) \right]$$

Want $\min_{\theta} J(\theta)$:

Repeat {

$$\theta_j := \theta_j - \alpha \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

} (simultaneously update all θ_j)

$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$ for $i = 0 \dots n$

$h_\theta(x) = \theta^\top x$

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

Algorithm looks identical to linear regression!

Andrew Ng

使用梯度下降算法来同步更新 θ ，在实现此算法时不必使用for循环之类的来同步更新 θ 。考虑使用矩阵来使得运行这个算法有更快的速度。在使用梯度下降算法的过程中，可能需要进行一定范围的特征缩放。

- 证明： $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$ 等价于 $\theta_j := \theta_j - \alpha \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

$$\text{考慮: } h_\theta(x^{(i)}) = \frac{1}{1 + e^{-\theta^T x^{(i)}}}$$

$$\text{則: } y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$

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

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

$$\text{所以: } \frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} \left[-\frac{1}{m} \sum_{i=1}^m [-y^{(i)} \log(1 + e^{-\theta^T x^{(i)}}) - (1 - y^{(i)}) \log(1 + e^{\theta^T x^{(i)}})] \right]$$

$$= -\frac{1}{m} \sum_{i=1}^m \left[-y^{(i)} \frac{-x_j^{(i)} e^{-\theta^T x^{(i)}}}{1 + e^{-\theta^T x^{(i)}}} - (1 - y^{(i)}) \frac{x_j^{(i)} e^{\theta^T x^{(i)}}}{1 + e^{\theta^T x^{(i)}}} \right]$$

$$= -\frac{1}{m} \sum_{i=1}^m y^{(i)} \frac{x_j^{(i)}}{1 + e^{\theta^T x^{(i)}}} - (1 - y^{(i)}) \frac{x_j^{(i)} e^{\theta^T x^{(i)}}}{1 + e^{\theta^T x^{(i)}}}$$

$$= -\frac{1}{m} \sum_{i=1}^m \frac{y^{(i)} x_j^{(i)} - x_j^{(i)} e^{\theta^T x^{(i)}} + y^{(i)} x_j^{(i)} e^{\theta^T x^{(i)}}}{1 + e^{\theta^T x^{(i)}}}$$

$$= -\frac{1}{m} \sum_{i=1}^m \frac{y^{(i)} (1 + e^{\theta^T x^{(i)}}) - e^{\theta^T x^{(i)}}}{1 + e^{\theta^T x^{(i)}}} x_j^{(i)}$$

$$= -\frac{1}{m} \sum_{i=1}^m \left(y^{(i)} - \frac{e^{\theta^T x^{(i)}}}{1 + e^{\theta^T x^{(i)}}} \right) x_j^{(i)}$$

$$= -\frac{1}{m} \sum_{i=1}^m \left(y^{(i)} - \frac{1}{1 + e^{-\theta^T x^{(i)}}} \right) x_j^{(i)}$$

$$= -\frac{1}{m} \sum_{i=1}^m [y^{(i)} - h_\theta(x^{(i)})] x_j^{(i)}$$

$$= \frac{1}{m} \sum_{i=1}^m [h_\theta(x^{(i)}) - y^{(i)}] x_j^{(i)}$$

注: 虽然得到的梯度下降算法表面上看上去与线性回归的梯度下降算法一样, 但是这注:

由 $x^{(i)}$ 表示第 i 个样本，又

$$x^{(i)} = \begin{pmatrix} x_0^{(i)} \\ \vdots \\ x_n^{(i)} \end{pmatrix}$$

$$x\theta = \begin{pmatrix} \theta_0 \\ \vdots \\ \theta_n \end{pmatrix}$$

故 $\theta^T = (\theta_0, \dots, \theta_n)$.

$$\text{有 } \cancel{\theta^T} \theta^T x^{(i)} = (\theta_0, \dots, \theta_n) \begin{pmatrix} x_0^{(i)} \\ \vdots \\ x_n^{(i)} \end{pmatrix}$$

$$= \theta_0 \cdot x_0^{(i)} + \dots + \theta_n \cdot x_n^{(i)}.$$

$$x \frac{\partial}{\partial \theta_j} (\theta^T x^{(i)}) = x_j^{(i)}.$$

(8) 高级优化

2022年1月11日 12:59

Optimization algorithm

Given θ , we have code that can compute

$$\begin{aligned} & - J(\theta) \\ & - \frac{\partial}{\partial \theta_j} J(\theta) \quad (for j = 0, 1, \dots, n) \end{aligned}$$

Optimization algorithms:

- > Gradient descent
- [- Conjugate gradient
- BFGS
- L-BFGS

Advantages:

- No need to manually pick α
- Often faster than gradient descent.

Disadvantages:

- More complex

在获取最小 θ 过程中除了梯度下降算法之外，还有三种高级算法

- 共轭梯度算法
- BFGS 算法
- L-BFGS 算法

这三种算法有以下特点

一、优点

- 不需要手动挑选学习率 θ
- 常常比梯度下降算法收敛得更快

二、缺点

- 比梯度下降算法要更加复杂

这些算法中间有一个智能内循环称之为线搜索算法，它可以尝试不同的学习率 θ ，并自动选择一个好的学习率 θ ，它可以为每次迭代选择不同的学习速率，可以进行调库使用。

使用 Octave 调用高级算法如下

Example: $\min_{\theta} J(\theta)$

$$\rightarrow \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \quad \theta_1 = 5, \theta_2 = 5.$$

$$\rightarrow J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$$

$$\rightarrow \frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$

$$\rightarrow \frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

```

function [jVal, gradient]
    = costFunction(theta)
jVal = (theta(1)-5)^2 + ...
       (theta(2)-5)^2;
gradient = zeros(2,1);
gradient(1) = 2*(theta(1)-5);
gradient(2) = 2*(theta(2)-5);

options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] ...
    = fminunc(@costFunction, initialTheta, options);

```

(9) 多元分类

2022年1月11日 15:21

多元分类是指对样本数据分多个类别

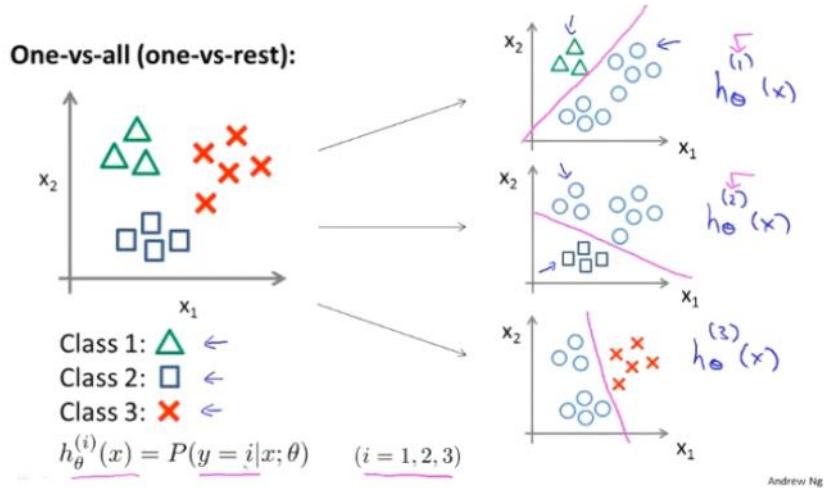
Multiclass classification

Email foldering/tagging: Work, Friends, Family, Hobby
 $y=1$ $y=2$ $y=3$ $y=4$

Medical diagrams: Not ill, Cold, Flu
 $y=1$ $y=2$ $y=3$

Weather: Sunny, Cloudy, Rain, Snow
 $y=1$ $y=2$ $y=3$ $y=4$ ←

如下图所示，对三种类别进行分类，我们先把三角形的类别筛选出来，并确定决策边界，具体先把三角形的作为一类，把类别样本作为另一类，其它的以此类推，确定三个决策边界，这样就可以进行多元分类



训练一个逻辑回归类别器，对于每个类别，去预测 y 为当前运行类别器的概率。对于一个输入 x ，去做出一个预测，提取类别*i*的最大值，即在上个例子中，对于输入 x ，分别运行三个类别器，取三个类别器中的最大概率的值，使得 x 为最大类别的类型

One-vs-all

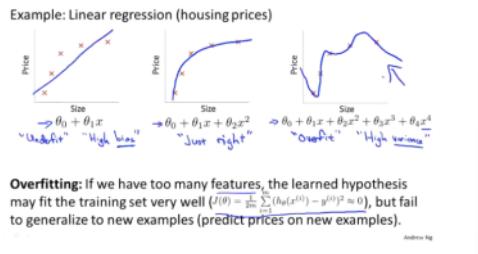
Train a logistic regression classifier $h_{\theta}^{(i)}(x)$ for each class i to predict the probability that $y = i$.

On a new input x , to make a prediction, pick the class i that maximizes

$$\max_i \underbrace{h_{\theta}^{(i)}(x)}_{\pi}$$

(10) 过拟合问题

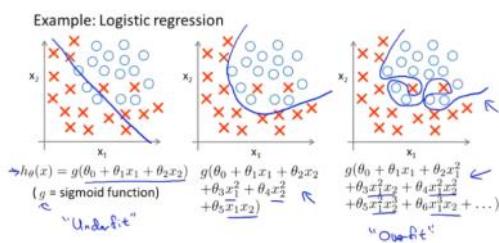
2022年1月11日 21:17



在上图中第一张图表示欠拟合，即拟合度不足，中间的拟合的比较好，然后最右一张属于过拟合，过拟合会导致无法泛化到新的样本中，无法预测新样本的价格。

“泛化”是指一个假设模型应用于新样本上的能力

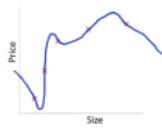
逻辑回归时的过拟合情形



过拟合问题产生于，当我们有过多的特征的数量，而只有极少的样本数量时，会产生过拟合问题。

Addressing overfitting:

- x_1 = size of house
- x_2 = no. of bedrooms
- x_3 = no. of floors
- x_4 = age of house
- x_5 = average income in neighborhood
- x_6 = kitchen size
- ⋮
- x_{100}



解决过拟合问题的两种方法：

一、减少特征的数量，有两种方案

- 手动挑选哪些特征保留，哪些特征减少，以此来减少特征的数量
- 模型选择算法，该算法自动选择哪些特征保留，哪些特征减少（见后面）

二、正则化

保留所有特征，但减少参数 θ_j 的数量级/值

Addressing overfitting:

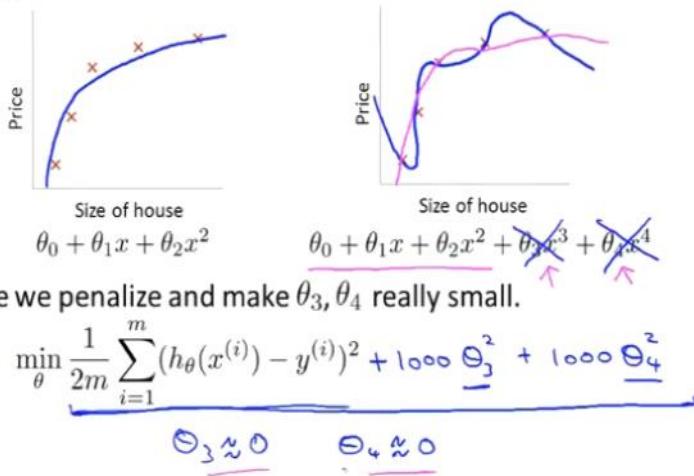
- Options:
1. Reduce number of features.
 - Manually select which features to keep.
 - Model selection algorithm (later in course).
 2. Regularization.
 - Keep all the features, but reduce magnitude/values of parameters θ_j .
 - Works well when we have a lot of features, each of which contributes a bit to predicting y .

Andrew Ng

(11) 代价函数

2022年1月11日 21:30

Intuition



Andrew Ng

在上个式子中二次形的式子足够拟合数据样本，而如果提高到四次幂的式子会存在过拟合问题。所以我们需要把式子 θ_3 和 θ_4 都趋向于0使得在代价函数中 θ_3 和 θ_4 对于整个代价函数的影响趋向于0

证明下面式子，会使得 θ_3 和 θ_4 会趋向于0

$$\rightarrow \min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + 1000 \underline{\theta_3^2} + 1000 \underline{\theta_4^2}$$

$\underline{\theta_3 \approx 0} \quad \underline{\theta_4 \approx 0}$

我们在寻找最小化 θ 的过程中，我们若采用梯度下降算法，则可得到如下过程，由于我们的公式是

$$\frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + 1000 \underline{\theta_3^2} + 1000 \underline{\theta_4^2}$$

$\underline{\theta_3 \approx 0} \quad \underline{\theta_4 \approx 0}$

与下面的公式不尽相同，但运行过程是一样的，都是要对其分别求导，当然也以采用矩阵化进行加速计算，这里只是演示其证明过程

<p>Gradient Descent</p> <p>Previously ($n=1$):</p> <p>Repeat {</p> <p>$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})$</p> <p style="text-align: center;">$\boxed{\frac{\partial}{\partial \theta_0} J(\theta)}$</p> <p>$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$</p> <p style="text-align: center;">(simultaneously update θ_0, θ_1)</p> <p>}</p>	<p>New algorithm ($n \geq 1$):</p> <p>Repeat {</p> <p style="text-align: right;">$\downarrow \frac{\partial}{\partial \theta_j} J(\theta)$</p> <p>$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$</p> <p style="text-align: center;">(simultaneously update θ_j for $j = 0, \dots, n$)</p> <p>}</p> <hr/> <p>$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$</p> <p>$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$</p> <p>$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$</p> <p>...</p>
---	--

Andrew Ng

我们有公式

$$\underbrace{\frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2}_{\Theta_3 \approx 0} + \underbrace{1000 \Theta_3^2}_{\Theta_4 \approx 0} + 1000 \Theta_4^2,$$

我们在其梯度下降的过程中，我们对 θ_3 和 θ_4 求导有如下公式

$$\theta_3 := \theta_3 - \alpha \frac{1}{2m} \left(\sum_{i=1}^m 2(h_\theta(x^{(i)}) - y^{(i)}) x_3^{(i)} + 2000\theta_3 \right)$$

$$\theta_4 := \theta_4 - \alpha \frac{1}{2m} \left(\sum_{i=1}^m 2(h_\theta(x^{(i)}) - y^{(i)}) x_4^{(i)} + 2000\theta_4 \right)$$

这样会使得 θ_3 和 θ_4 趋向于 0

正则化

Regularization.

Small values for parameters $\theta_0, \theta_1, \dots, \theta_n$

— “Simpler” hypothesis

— Less prone to overfitting

$$\rightarrow \boxed{\theta_3, \theta_4} \approx 0$$

Housing:

— Features: x_1, x_2, \dots, x_{100}

— Parameters: $\theta_0, \theta_1, \theta_2, \dots, \theta_{100}$

在这个特征中我们无法确定哪个特征是相关度比较低的，所以我们

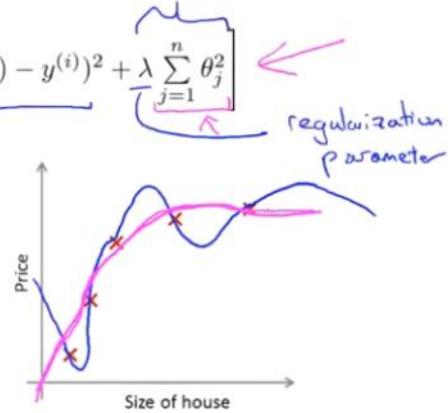
直接把所有除 θ_0 之外的所有特征都进行正则化

所有我们有正则化之后的代价函数

Regularization.

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

$\min_{\theta} J(\theta)$



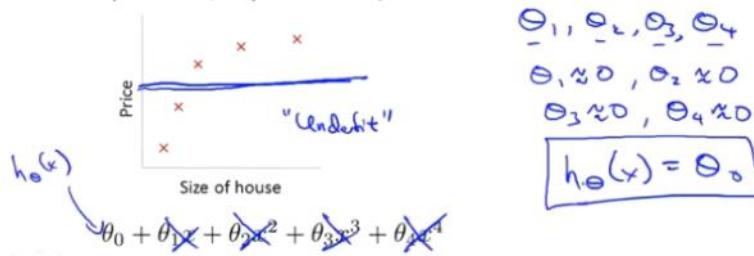
Andrew

若我们把所有特征都正则化，且没有选择一个合适的 λ ,会出现下图的欠拟合的情况，所以要选择一个合适的 λ

In regularized linear regression, we choose θ to minimize

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

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?



Andrew Ng

(12) 线性回归的正则化

2022年1月12日 15:44

线性回归有两种算法，一种梯度下降法，一种正规方程法

对于梯度下降法我们有如下图

Gradient descent

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$
$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right] \quad (j = 1, 2, 3, \dots, n)$$
$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$1 - \alpha \frac{\lambda}{m} < 1$ 0.99 $\theta_j \times 0.99$

Andrew Ng

在上图中我们的更新公式变成

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \left(\sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right)$$

$1 - \alpha \frac{\lambda}{m} < 1$ 可能趋向于0.99, 这样进行逐步迭代

在正规矩阵中求最小的 θ , 证明如下

Non-invertibility (optional/advanced).

Suppose $m \leq n$,
(#examples) (#features)

$$\theta = (X^T X)^{-1} X^T y$$

If $\lambda > 0$,

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \\ & & & 1 \end{bmatrix} \right)^{-1} X^T y$$

在上式中，如果样本数m小于等于特征数n，就会产生不可逆的情况

(13) 逻辑回归的正则化

2022年1月12日 16:42

在逻辑回归正则化中使用梯度下降算法如下

Gradient descent

Repeat {

$$\rightarrow \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\rightarrow \theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right] \leftarrow$$

} $j = 1, 2, 3, \dots, n$
 $\theta_0, \dots, \theta_n$

$$\frac{\partial}{\partial \theta_j} J(\theta)$$

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

Andrew Ng

通过Octave使参数正则化，并通过高级算法使最小化 $\min(\theta)$ 有

Advanced optimization

\rightarrow function [jVal, gradient] = costFunction(theta)

$jVal =$ [code to compute $J(\theta)$];

 $\rightarrow J(\theta) = \left[-\frac{1}{m} \sum_{i=1}^m y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$

\rightarrow gradient(1) = [code to compute $\frac{\partial}{\partial \theta_0} J(\theta)$];

 $\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$

\rightarrow gradient(2) = [code to compute $\frac{\partial}{\partial \theta_1} J(\theta)$]; $J(\theta)$

 $\left(\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \right) + \frac{\lambda}{m} \theta_1$

\rightarrow gradient(3) = [code to compute $\frac{\partial}{\partial \theta_2} J(\theta)$];

 $\vdots \quad \left(\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)} \right) + \frac{\lambda}{m} \theta_2$

gradient(n+1) = [code to compute $\frac{\partial}{\partial \theta_n} J(\theta)$];

Andrew Ng

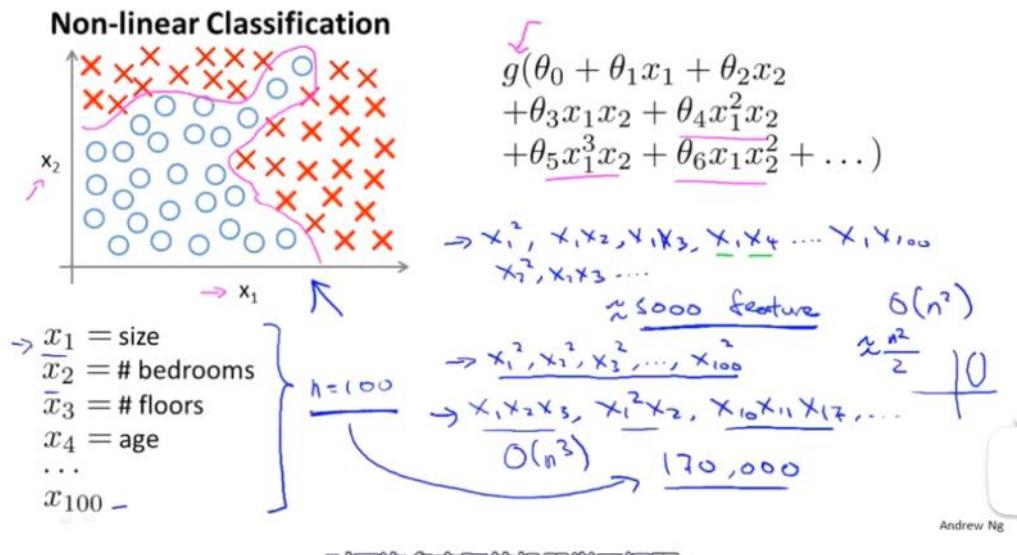
(01) 神经网络

2022年1月12日 17:13

(02) 非线性假设

2022年1月12日 17:22

我们在变量进行分类的时候，可能会存在非线性分类，如果使用过多的特征会出现过拟合或使得特征空间过大的情况

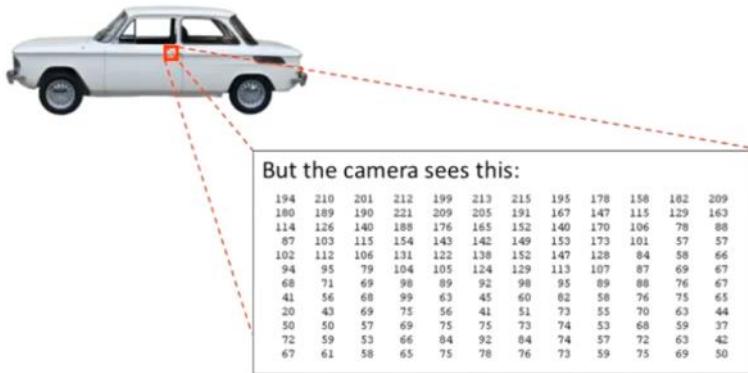


对于许多实际的机器学习问题

例如有下面的情形

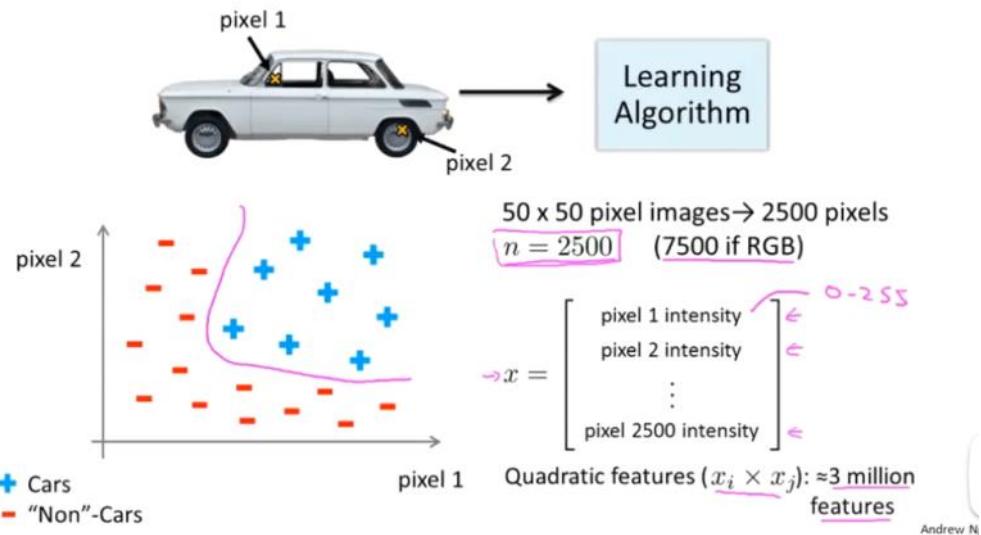
What is this?

You see this:



Andrew

在有些计算机视觉的问题中，根据这个像素矩阵，告诉我们这个矩阵代表门把手，具体来说，当我们用机器学习算法构造一个汽车识别器时，我们要做的是提供一个带标签的样本集，其中一些是各类汽车，另一部分样本不是车。将这个样本集输入给学习算法以训练出另一个分类器，然后我们进行测试，输入一张新图片，让分类器判定这是什么东西

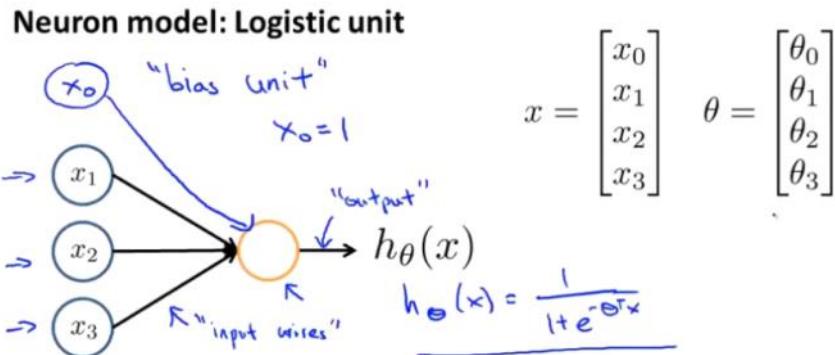


在上图中我们假设中取两个像素，并通过这两个像素来代表是汽车或者非汽车。然而如果有一张50*50像素的图片，那么会导致有过多的特征值，且导致计算成本过高

因此，只是包括平方项或者是立方项特征的简单logistic回归算法，并不是在一个n很大时，学习复杂的非线性假设的好办法，因为特征过多。

(03) 神经模型

2022年1月12日 19:17



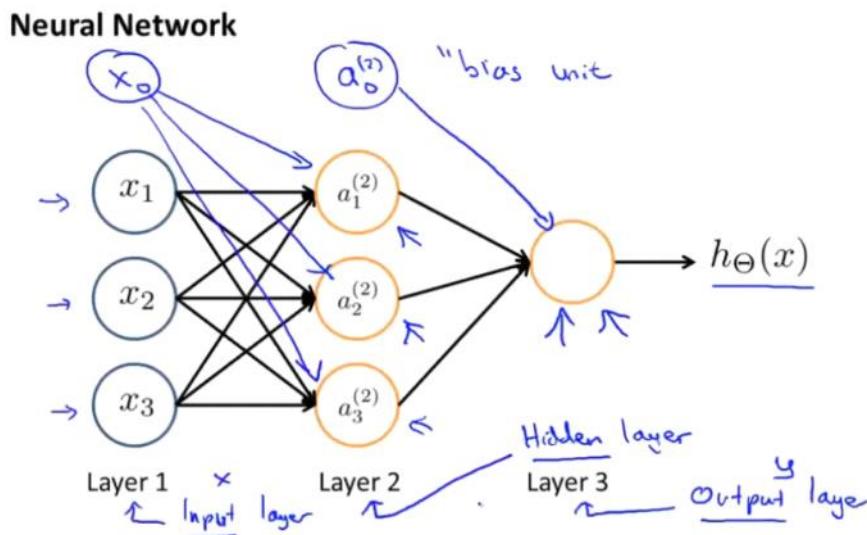
Sigmoid (logistic) activation function.

$$g(z) = \frac{1}{1 + e^{-z}}$$

Andrew Ng

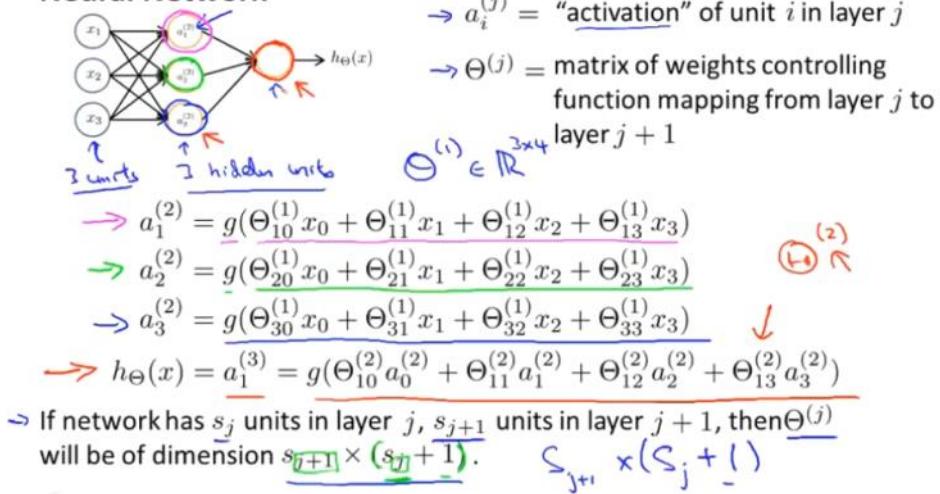
神经模型中的单个逻辑单元如上图所示，通过输入 x_0, x_1, \dots, x_n 得到一个逻辑激活函数，这称之为逻辑激活函数。

逻辑激活函数即 $g(z) = \frac{1}{1 + e^{-z}}$



在上图中是由一组神经元组合而成的神经网络，其中最左边的为输入层，中间为隐藏层，最右边的为输出层

Neural Network



Anc

在上图中表示神经网络共有三层，第一层表示输入，中间层表示隐藏层，第三层表示输出层。其中 $a_i^{(j)}$ 表示第 j 层的第 i 个激活单元， $\theta^{(j)}$ 表示权重控制的矩阵函数从第 j 层映射到第 $j+1$ 层。

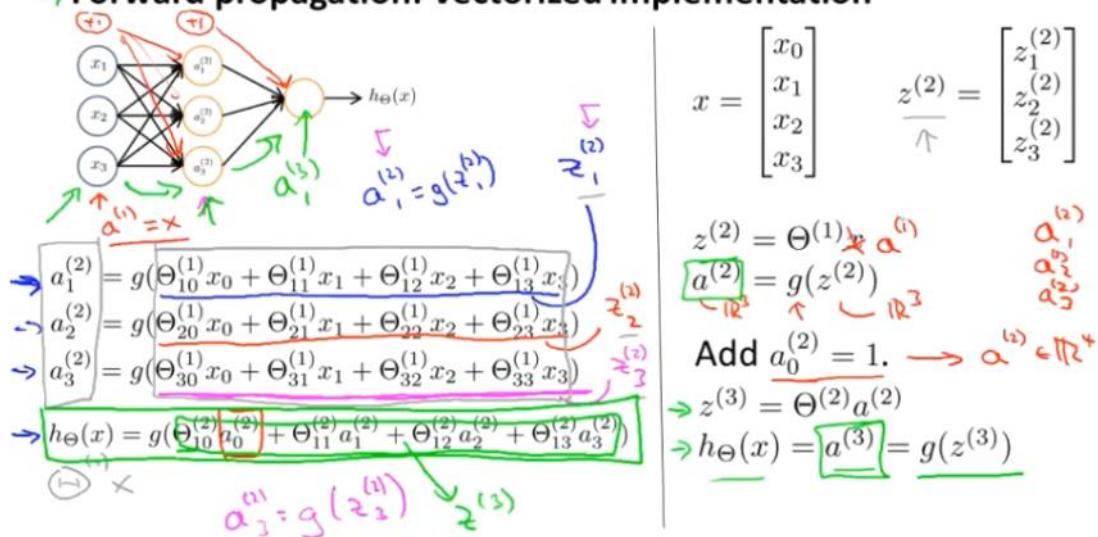
如果神经网络在第 j 层有 s_j 单元在第 $j+1$ 层有 s_{j+1} 单元然后 $\theta^{(j)}$ 有维度

$$s_{j+1} \times (s_j + 1)$$

在上图中 $\theta^{(1)}$ 为一个 3×4 的矩阵

向前传播是指从第一层传播到第三层，以这样的方式进行传播，即从输入层传播到输出层

Forward propagation: Vectorized implementation



我们对向前传播进行向量化，我们把变化为

$$z_1^{(2)} = \theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3$$

$$z_2^{(2)} = \theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3$$

$$z_3^{(2)} = \theta_{30}^{(1)}x_0 + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3$$

其中

$$z^{(2)} = \theta^{(1)}x = \theta^{(1)}a^{(1)}$$

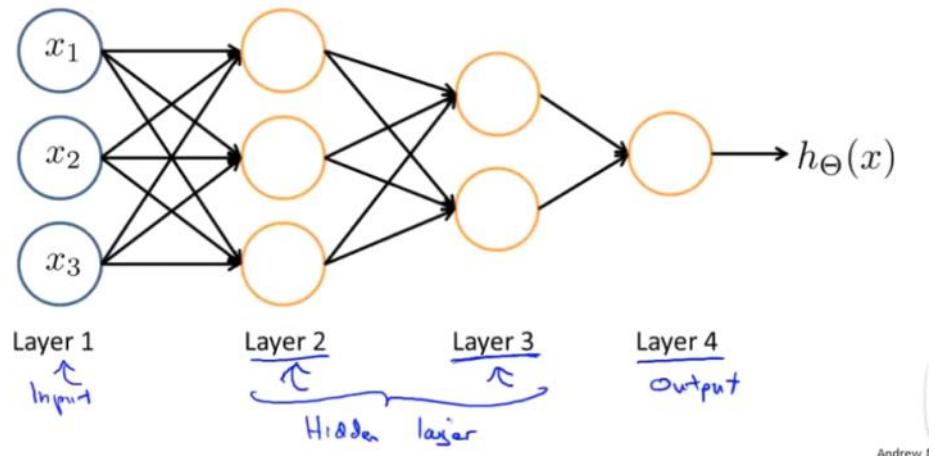
$$a^{(2)} = g(z^{(2)})$$

$$z^{(3)} = \theta^{(2)}x = \theta^{(2)}a^{(2)}$$

$$h_\theta(x) = a^{(3)} = g(z^{(3)})$$

神经网络的架构如下图，基本上有三部分组成，一是输入层，二是隐藏层，三是输出层。

Other network architectures



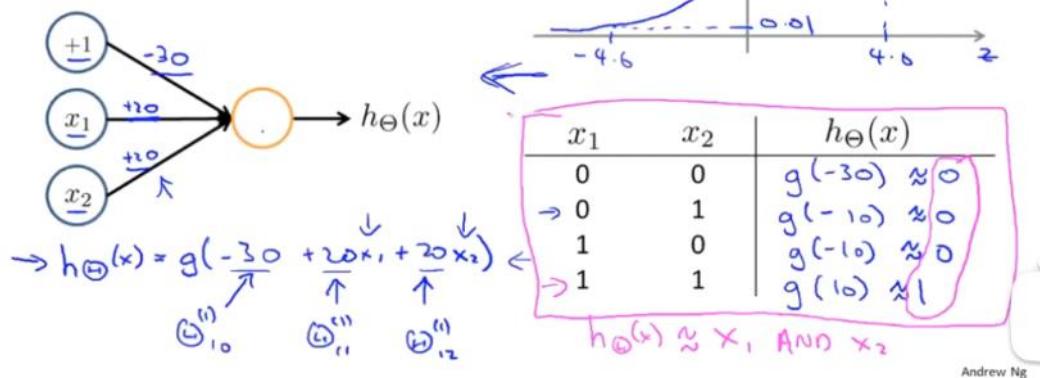
Andrew I

下面是一个神经网络的简单例子

Simple example: AND

$$\rightarrow x_1, x_2 \in \{0, 1\}$$

$$\rightarrow y = x_1 \text{ AND } x_2$$

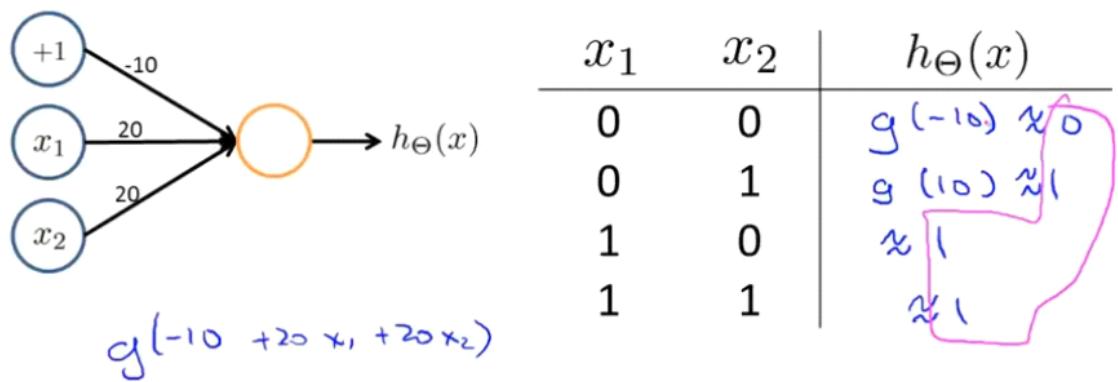


在上张图的例子中，我们对于改变 x_1 和 x_2 的值来改变 $h_\theta(x)$ 的来模拟逻辑符号

and

下面这张图的模拟逻辑符号or的例子

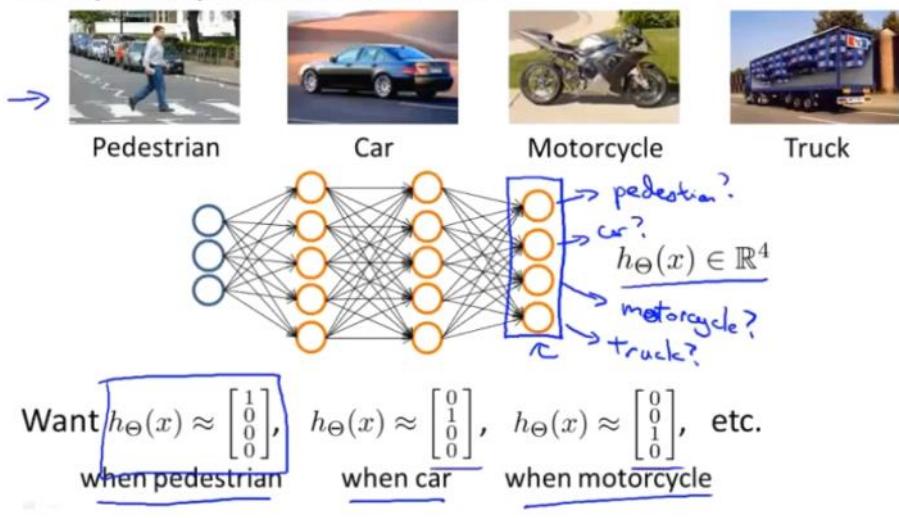
Example: OR function



(04) 多元分类

2022年1月13日 14:23

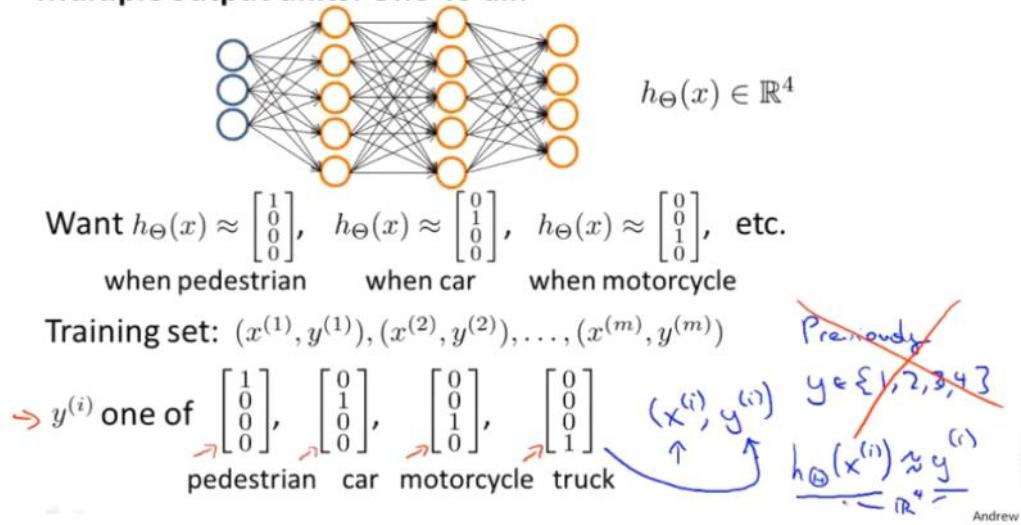
Multiple output units: One-vs-all.



Ans

在上图中表示了神经网络分类的方法，我们可以通过 $h_{\theta}(x)$ ，输出不同的矩阵，来实行多元分类。

Multiple output units: One-vs-all.

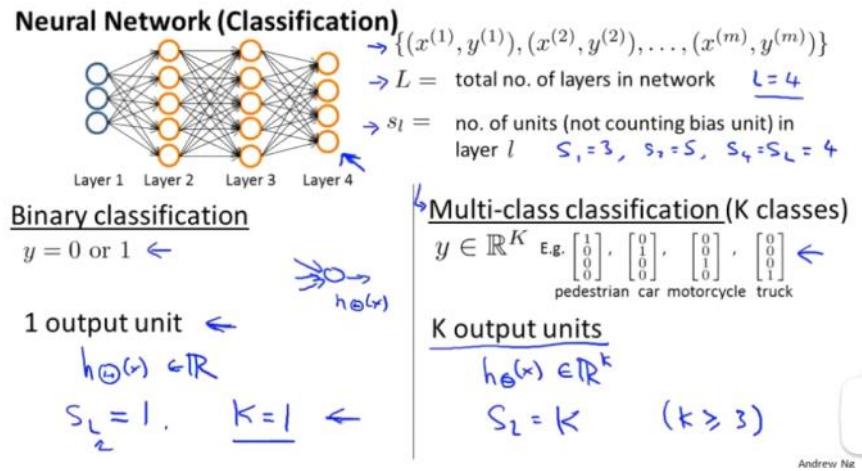


Andrew

在神经网络中，我们若要实行多元分类，我们需要通过输入样本数据 $x^{(i)}$ ，然后输出数据 $y^{(i)}$ 分类的矩阵我们需要使得函数 $h_{\theta}(x)$ 来近似于 $y^{(i)}$ 来对不同的数据进行分类

(05) 代价函数

2022年1月13日 15:00



在卷积神经网络中我们既需要进行二元分类也需要进行多元分类

Cost function

Logistic regression:

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

Neural network:

$$\rightarrow h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{\text{th}} \text{ output}$$

$$\rightarrow J(\Theta) = -\frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k) \right]$$

$$\boxed{\frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2}$$

Andrew Ng

神经网络的代价函数如下：

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log((h_\theta(x^{(i)}))_k) + (1 - y_k^{(i)}) \log(1 - h_\theta(x^{(i)}))_k \right] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\theta_{ji}^{(l)})^2$$

因为在神经网络中有多个输入项，所以有多个代价函数，在这个式子中代价函数数为K，即K个输出函数(项)，所以第一项为上式所表示，故有第一项为

$$-\frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log((h_\theta(x^{(i)}))_k) + (1 - y_k^{(i)}) \log(1 - h_\theta(x^{(i)}))_k \right]$$

在第二项中，表示有L层，而因为 θ 由上一层与连接下一层而形成的，所以个数也为两层的乘积，而后有从1到L-1，故有第二项为

$$\frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\theta_{ji}^{(l)})^2$$

(06) 反向传播

2022年1月19日 19:15

Gradient computation

$$\begin{aligned} \rightarrow J(\Theta) = & -\frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log h_\theta(x^{(i)})_k + (1 - y_k^{(i)}) \log(1 - h_\theta(x^{(i)})_k) \right] \\ & + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_j^{(l)})^2 \end{aligned}$$

$$\rightarrow \min_{\Theta} J(\Theta)$$

Need code to compute:

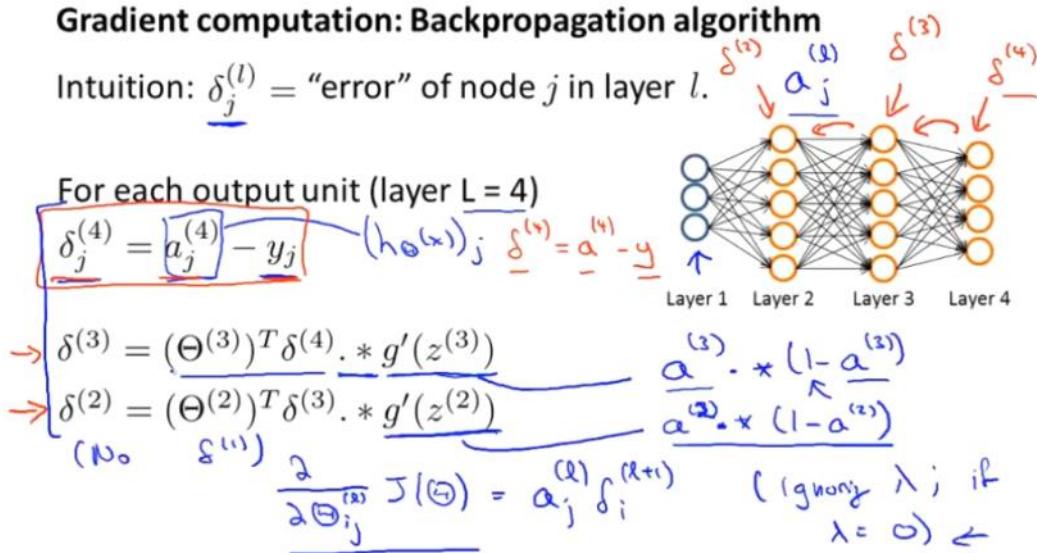
$$\begin{aligned} \rightarrow -J(\Theta) \\ \rightarrow -\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) \end{aligned}$$

$$\Theta_{ij}^{(l)} \in \mathbb{R}$$

在上图中我们的最终目的是寻找最小化 $J(\Theta)$

Gradient computation: Backpropagation algorithm

Intuition: $\delta_j^{(l)}$ = "error" of node j in layer l .



反射传播从直观上来说就是计算每一层的每个节点的误差，在上图中就是计算第三层的各个节点的误差，而后计算第二层的各个节点的误差，最后计算第一层的各个节点的误差。

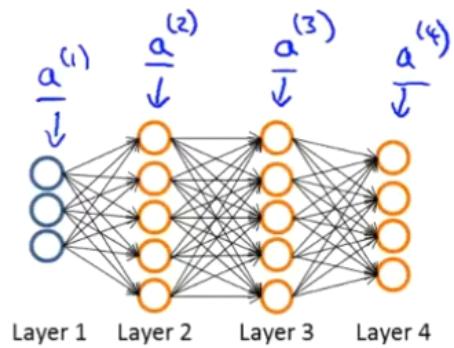
我们有梯度下降的正向传播过程以及涉及的参数如下

Gradient computation

Given one training example (x, y) :

Forward propagation:

$$\begin{aligned} \underline{a^{(1)}} &= x \\ \rightarrow z^{(2)} &= \Theta^{(1)} a^{(1)} \\ \rightarrow a^{(2)} &= g(z^{(2)}) \quad (\text{add } a_0^{(2)}) \\ \rightarrow z^{(3)} &= \Theta^{(2)} a^{(2)} \\ \rightarrow a^{(3)} &= g(z^{(3)}) \quad (\text{add } a_0^{(3)}) \\ \rightarrow z^{(4)} &= \Theta^{(3)} a^{(3)} \\ \rightarrow a^{(4)} &= \underline{h_\Theta(x) = g(z^{(4)})} \end{aligned}$$



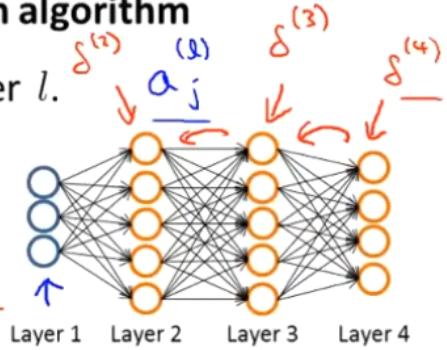
梯度下降算法中的反向传播算法过程如下

Gradient computation: Backpropagation algorithm

Intuition: $\underline{\delta_j^{(l)}}$ = “error” of node j in layer l .

For each output unit (layer $L = 4$)

$$\delta_j^{(4)} = \underline{a_j^{(4)} - y_j} \quad (h_\Theta(x))_j \quad \underline{\delta^{(4)}} = \underline{a^{(4)} - y}$$



$$\rightarrow \delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} \cdot * g'(z^{(3)})$$

$$\rightarrow \delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} \cdot * g'(z^{(2)})$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)}$$

反向传播算法如下：

Backpropagation algorithm

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

Set $\Delta_{ij}^{(l)} = 0$ (for all l, i, j). (use to compute $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$)

For $i = 1$ to m ← $(x^{(i)}, y^{(i)})$.

Set $a^{(1)} = x^{(i)}$

→ Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \dots, L$

→ Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$

→ Compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$

$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}$ if $j \neq 0$

$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)}$ if $j = 0$

$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + \delta^{(l+1)} (a^{(l)})^T$

$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$

其中符号 Δ , 其中 l 表示神经网络的层数, i 表示样本数, j 表示神经网络中某层的第几个节点。

算法步骤解释如下:

一、设置所有符号 $\Delta_{ij}^{(l)}$ 均为0.

二、设置从样本1到m的循环, 即遍历所有样本

三、在循环中设置输入参数为 $a^{(1)}$

四、执行正向传播算法, 求得各层的 $a^{(l)}$

五、使用结果 $y^{(i)}$ 来计算输出层的差值 $\delta^{(l)}$

六、计算隐藏层的差值 $\delta^{(l-1)}, \delta^{(l-2)} \dots$

七、计算每个样本的各层的各个节点的 $\Delta_{ij}^{(l)}$, 其中式子

$$\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$$

其中, $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\theta) = a_j^{(l)} \delta_i^{(l+1)}$, 此处的 $J(\theta)$ 为 $m=1$ 时, 且无正则项里的情况

八、我们计算各样本各层各节点的 $D_{ij}^{(l)}$, $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\theta) = D_{ij}^{(l)}$

此处的函数 $J(\theta)$ 为有正则项的分两种情况, 有式子

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \theta_{ij}^{(l)} \quad \text{if } j \neq 0 \text{ (非偏置项)}$$

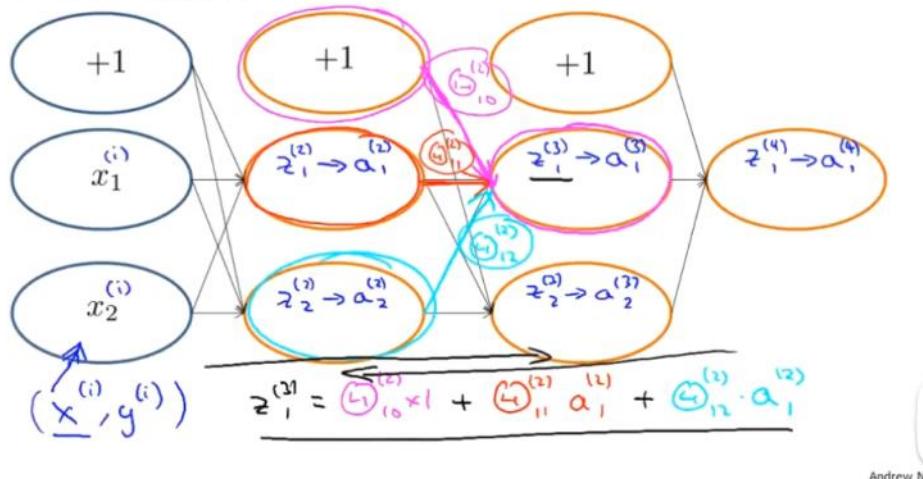
$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} \quad \text{if } j = 0 \text{ (偏置项)}$$

在此处为什么是 $\lambda \theta_{ij}^{(l)}$ 不是 $\frac{\lambda}{m} \theta_{ij}^{(l)}$ 暂时还不清楚

(07) 反向传播的直观理解

2022年1月19日 20:47

Forward Propagation



上图给出了正向传播的计算方式，即从左往右进行传播，即举出的例子，对于正向传播而言，我们先需要求出各个 θ 和 a 的值，然后根据各个 θ 和 a 求出 $z_1^{(3)}$ 的值。而对于反向传播而言，我们是先根据 $z_1^{(3)}$ 的值再求出各个各个 θ 和 a 。

What is backpropagation doing?

$$J(\Theta) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h_\Theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\Theta(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2$$

Focusing on a single example $x^{(i)}, y^{(i)}$, the case of 1 output unit, and ignoring regularization ($\lambda = 0$),

$$\text{cost}(i) = y^{(i)} \log h_\Theta(x^{(i)}) + (1 - y^{(i)}) \log h_\Theta(x^{(i)})$$

(Think of $\text{cost}(i) \approx (h_\Theta(x^{(i)}) - y^{(i)})^2$)

I.e. how well is the network doing on example i?

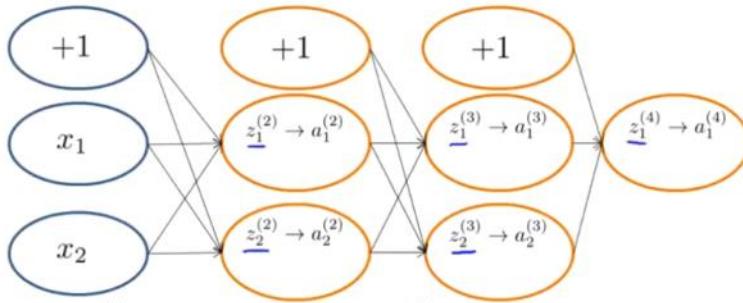
我们聚焦在对于样本 $x(i)$ 给出的假设 $h_\theta(x^{(i)})$ 和实际的值 $y^{(i)}$ ，我们忽略其正则，即把 λ 视为0。又有图中的cost函数表示机器预测的样本给出的结果与实际结果是否一致的程度，然后式子

$$(h_\theta(x^{(i)}) - y^{(i)})^2$$

表示预测的结果与实际的结果之间的方差其也是用于反应预测结果与实际结果是否一致的程度故有式子

$$\text{cost}(i) \approx (h_\theta(x^{(i)}) - y^{(i)})^2$$

Forward Propagation



$\rightarrow \delta_j^{(l)}$ = “error” of cost for $a_j^{(l)}$ (unit j in layer l).

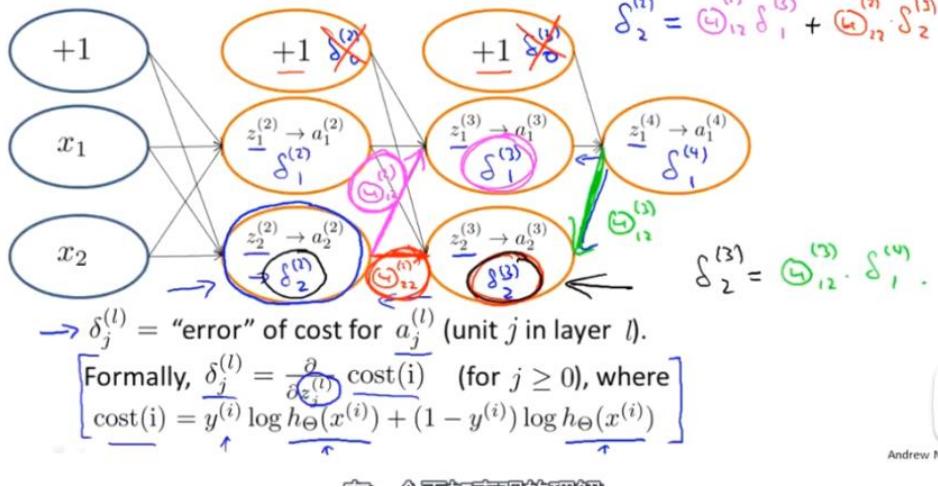
$$\left[\text{Formally, } \delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \underbrace{\text{cost}(i)}_{\text{cost}(i) = y^{(i)} \log h_\Theta(x^{(i)}) + (1 - y^{(i)}) \log h_\Theta(x^{(i)})} \quad (\text{for } j \geq 0), \text{ where} \right]$$

符号 $\delta_j^{(l)}$ 表示激活函数 cost 对于 $a_j^{(l)}$ (表示在第 l 层的单元节点) 的误差。

进而推出当 $\text{cost}(i) = y^{(i)} \log h_\Theta(x^{(i)}) + (1 - y^{(i)}) \log h_\Theta(x^{(i)})$

$$\text{就有 } \delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(i)$$

Forward Propagation



如上图所示，我们得出三个式子，分别为

$$\delta_1^{(4)} = y^{(4)} - a_1^{(4)}$$

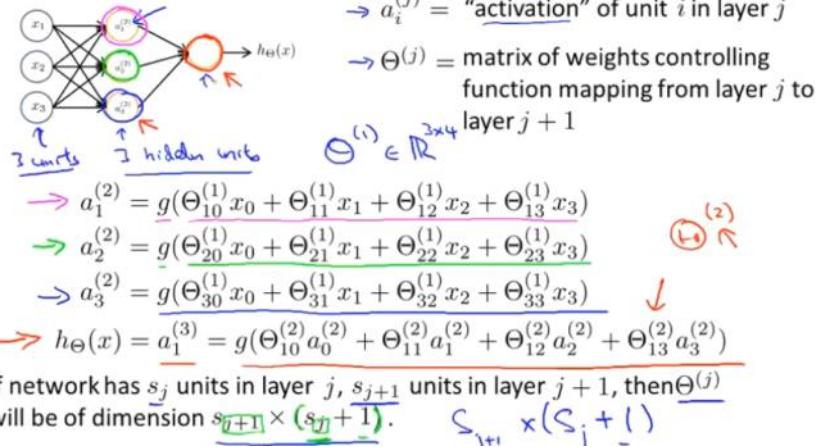
$$\delta_2^{(3)} = \theta_{12}^{(3)} \delta_1^{(4)}$$

$$\delta_2^{(2)} = \theta_{12}^{(2)} \delta_1^{(3)} + \theta_{22}^{(2)} \delta_2^{(3)}$$

我们可以从上面三个式子中得到，从最后输出的假设函数和实际的值之的误差得到一个值 $\delta_1^{(4)}$ ，而后根据初步得到的误差 $\delta_1^{(4)}$ 给予权重 $\theta_{12}^{(3)}$ 得到误差 $\delta_2^{(3)}$ ，而后进行反向传播最后得到最终的误差。

注：

Neural Network



$\theta^{(j)}$ 表示从第 j 层映射到第 $j+1$ 层的控制函数的矩阵权重，具体而言， $\theta_i^{(j)}$ 表示第 j 层的第 i 个节点映射到第 $j+1$ 层的权重，而 $\theta_{ik}^{(j)}$ 表示第 j 层的第 i 个节点的，对于输入参数 k 所配的权重

$a_i^{(j)}$ 表示第 j 层的单元 i 的活动，具体而言有

$$a_1^{(2)} = g(\theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3)$$

表示第 2 层的单元 1 在给出输入参数和节点权重之后的概率值

(08) 反向传播算法的理解和推导

2022年1月20日 13:55

在前两节中我们给出了一些反向传播算法的一些概念和解释，在这里梳理一下反向传播算法及其推导过程，我们有反向传播算法描述。

Backpropagation algorithm

- Training set $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$
- Set $\Delta_{ij}^{(l)} = 0$ (for all l, i, j). (use to compute $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$)
- For $i = 1$ to m ← $(x^{(i)}, y^{(i)})$.
- Set $a^{(1)} = x^{(i)}$
- Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \dots, L$
- Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
- Compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$
- $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$
- $D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}$ if $j \neq 0$
- $D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)}$ if $j = 0$
- $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + \delta^{(l+1)} (a^{(l)})^T$
- $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$

反向传播算法描述：

神经网络的向前传播并构建出了损失函数，我们从输入层开始一层一层向前进行计算，计算从第一层到最后一层的 $h_\theta(x)$ 。

现在为了计算梯度下降法就要计算偏导数 $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\square)$ ，因为要各层各个节点的偏导数我们需要先计算 $\delta_i^{(l)}$ ，继而计算出 $\Delta_{ij}^{(l)}$ 最后根据 $\Delta_{ij}^{(l)}$ ，计算出偏导数 $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\square)$ 。因为要计算 $\delta_i^{(l)}$ ，故我们需要运用反向传播的算法，计算各个 $\delta_i^{(l)}$ 。其原理为先计算最后一层的误差，然后再一层层反向求出各层的误差，直到倒数第二层(应倒数第一层为输入层，所以不存在任何误差)。

反向传播算法步骤：

一、对神经网络的权值进行随机初始化。

二、遍历所有样本

1、运用正向传播算法，得到预测值 $a^L = h_\theta(x)$

2、运用反向传播算法，从输出层开始计算每层的误差，以此来求取偏导，输出层的误差即预测值与真实值的差值。 $\delta^L = \delta^L - y$ ，对于隐藏层中的每一层的误差，都通过上一层的误差来计算，得到

$$\delta^{(l)} = (\theta^{(l)})^T \delta^{(l+1)} * a^{(l)} * (1 - a^{(l)}), \quad a_0^{(l)} = 1$$

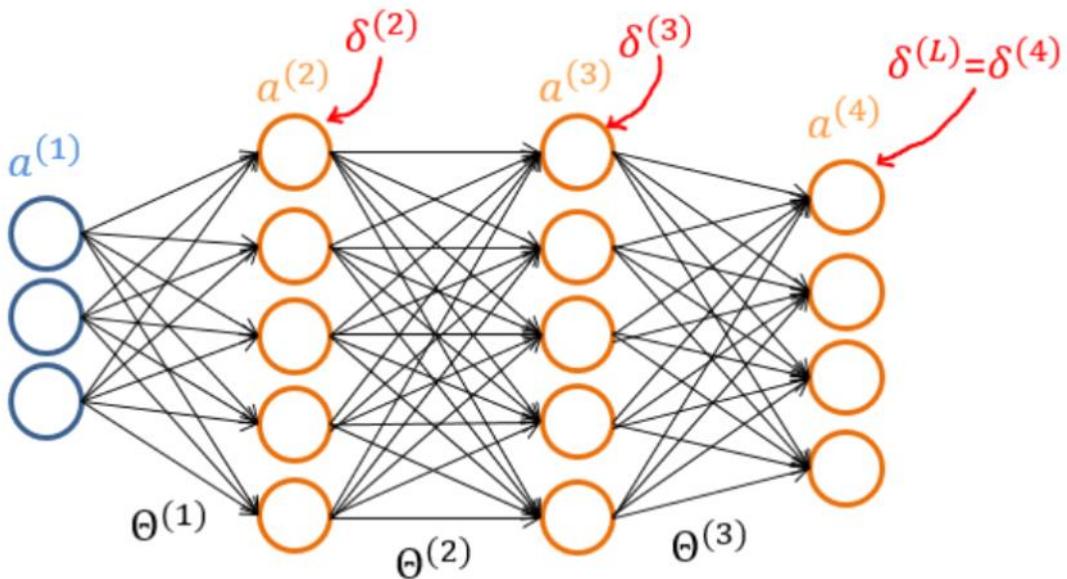
$$\text{对于式子 } \Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$$

三、遍历完所有样本后，求得偏导为 $\frac{\partial}{\partial \theta_{ij}^{(l)}} \bar{J}(\theta) = D_{ij}^{(l)}$

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \theta_{ij}^{(l)} \quad \text{if } j \neq 0$$

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} \quad \text{if } j = 0$$

反射传播算法推导



在这里我们假设样本只有一个，则：

先令

$$\delta = \frac{\partial}{\partial z^{(L)}} J(\theta)$$

故

$$\frac{\partial}{\partial \theta^{(3)}} J(\theta) = \frac{\partial J(\theta)}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial \theta^{(3)}} = \delta^{(4)} \frac{\partial z^{(4)}}{\partial \theta^{(3)}}$$

因 $z^{(4)} = \theta^{(3)} a^{(3)}$, 故 $\frac{\partial z^{(4)}}{\partial \theta^{(3)}} = a^{(3)}$,

所以证得

$$\frac{\partial}{\partial \theta^{(3)}} J(\theta) = a^{(3)} \delta^{(4)}$$

代价函数无正则项时有

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log(h_0(x^{(i)})) + (1-y^{(i)}) \log(1-h_0(x^{(i)})) \right]$$

$$h_\theta(x) = a^{(L)} = g(z^{(L)}), g(z) = \frac{1}{1+e^{-z}}, \text{ 带入到 J 中 (为方便计算令 m=1)}$$

$$\begin{aligned} J(\theta) &= -y \log\left(\frac{1}{1+e^{-z}}\right) - (1-y)(1-\frac{1}{1+e^{-z}}) \\ &= y \log(1+e^{-z}) - (1-y)(\frac{1}{1+e^{-z}}) \\ &= y \log(1+e^{-z}) + (1-y)(1+e^z) \end{aligned}$$

$$\begin{aligned} \delta^{(4)} &= \frac{\partial}{\partial z^{(4)}} J(\theta) = \frac{\partial}{\partial z^{(4)}} \left[y \log(1+e^{-z^{(4)}}) + (1-y) \log(1+e^{z^{(4)}}) \right] \\ &= y \frac{-e^{-z^{(4)}}}{1+e^{-z^{(4)}}} \\ &+ (1-y) \frac{e^{z^{(4)}}}{1+e^{z^{(4)}}} \frac{-ye^{(-z^{(4)})} - ye^{-z^{(4)}} e^{z^{(4)}} + 1 - y + e^{z^{(4)}} - ye^{z^{(4)}}}{(1-e^{(-z^{(4)})})(1+e^{(z^{(4)})})} \\ &= \frac{(1-y)e^{z^{(4)}} - (1+e^{z^{(4)}})ye^{z^{(4)}} + (1-y)}{(1-e^{(-z^{(4)})})(1+e^{(z^{(4)})})} \\ &= \frac{(1-y)(1+e^{z^{(4)}}) - (1+e^{z^{(4)}})ye^{z^{(4)}}}{(1-e^{(-z^{(4)})})(1+e^{(z^{(4)})})} \\ &= \frac{(1-y - ye^{-z^{(4)}})(1+e^{z^{(4)}})}{(1-e^{(-z^{(4)})})(1+e^{(z^{(4)})})} \\ &= \frac{1-y(1-e^{-z^{(4)}})}{(1-e^{(-z^{(4)})})} \\ &= g(z^{(4)}) - y \\ &= a^{(4)} - y \end{aligned}$$

即证得 $\delta^{(4)} = a^{(4)} - y$

故对于输出层 L

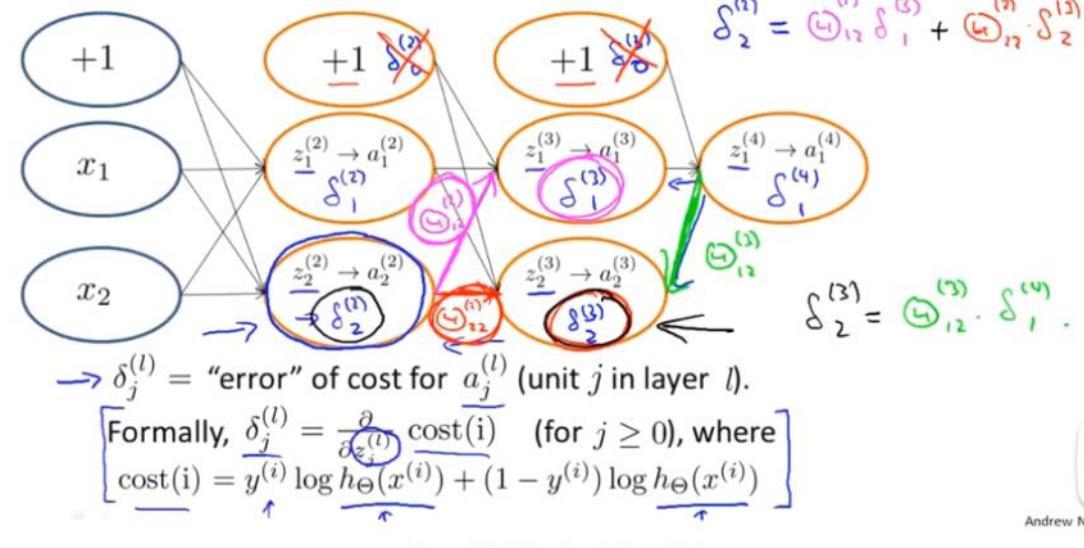
$$\frac{\partial}{\partial \theta^{(L-1)}} J(\theta) = a^{(L-1)} \delta^{(L)}$$

$$\delta^{(L)} = a^{(L)} - y$$

对于非输出层有

$$\frac{\partial}{\partial \theta^{(2)}} J(\theta) = \frac{\partial J(\theta)}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial \theta^{(2)}} = \delta^{(3)} \frac{\partial z^{(3)}}{\partial \theta^{(2)}} = a^{(2)} \delta^{(3)}$$

Forward Propagation



Andrew Ng

由上图可知

$$\begin{aligned}\delta^{(3)} &= \frac{\partial J(\theta)}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}} \\ &= \delta^{(4)} \theta^{(3)} \frac{\partial a^{(3)}}{\partial z^{(3)}}\end{aligned}$$

对 sigmoid 函数求导可得

有函数

$$g(z) = \frac{1}{1 + e^{-z}}$$

对其导，可得

$$g'(z) = \frac{e^{-z}}{(1 + e^{-z})^2}$$

$$\begin{aligned}&= \frac{1}{1 + e^{-z}} \frac{e^{-z}}{1 + e^{-z}} \\ &= \frac{1}{1 + e^{-z}} \left(\frac{1 + e^{-z}}{1 + e^{-z}} - \frac{1}{1 + e^{-z}} \right) \\ &= g(z)(1 - g(z))\end{aligned}$$

由于 $a^{(3)} = g(z^{(3)})$ 添加偏置项 $a_0 = 1$ ，则

$$\frac{\partial a^{(3)}}{\partial z^{(3)}} = g(z^{(3)}) (1 - g(z^{(3)})) = a^{(3)} \cdot (1 - a^{(3)})$$

所以

$$\delta^{(3)} = (\theta^{(3)})^T \delta^{(4) \cdot *} a^{(3) \cdot *} (1 - a^{(3)})$$

故对于隐藏层：

$$\frac{\partial}{\partial \theta^{(L)}} J(\theta) = a^{(L)} \delta^{(L+1)}$$
$$\delta^{(L)} = (\theta^{(l)})^T \delta^{(l+1) \cdot *} a^{(l) \cdot *} (1 - a^{(l)})$$

(09) 展开参数

2022年1月21日 15:57

Advanced optimization

```
function [jVal, gradient] = costFunction(theta)
    ...
optTheta = fminunc(@costFunction, initialTheta, options)
```

Neural Network ($L=4$):

→ $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$ - matrices (Theta1, Theta2, Theta3)

→ $D^{(1)}, D^{(2)}, D^{(3)}$ - matrices (D1, D2, D3)

“Unroll” into vectors

对于一个四层的神经网络来说，有一个参数矩阵 $\theta^{(1)}, \theta^{(2)}, \theta^{(3)}$.
有计算出来的梯度矩阵 $D^{(1)}, D^{(2)}, D^{(3)}$.

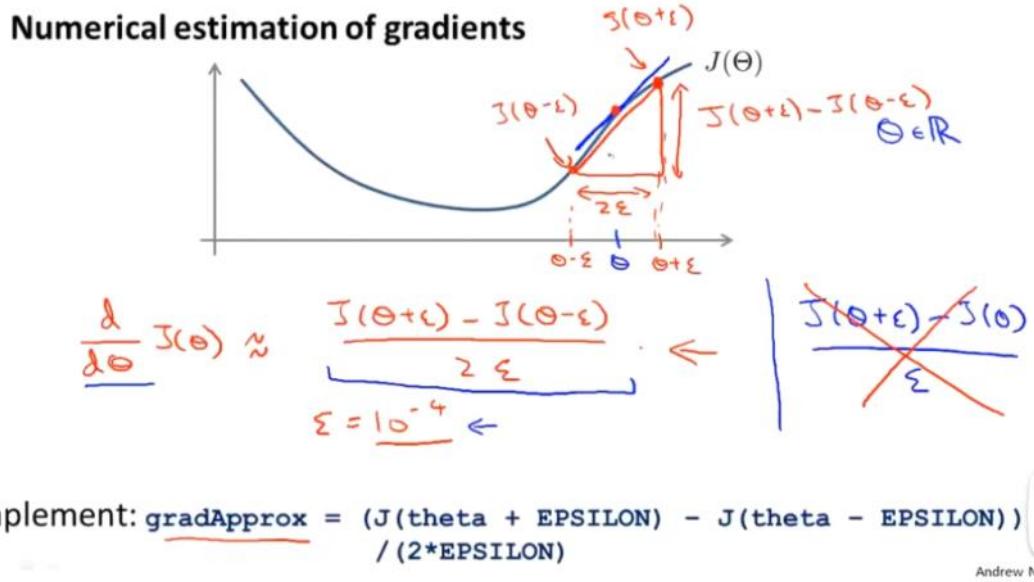
因为我们costFunction函数是需要给定输入参数，而该函数会返回一个梯度，所以我们需要把参数和梯度进行矩阵化，方便传入更多的参数和返回更多的梯度。

在octave中使用方法，以及如何得到梯度矩阵。

(10) 梯度检测

2022年1月21日 16:05

这里有梯度检测的原因是因为反向传播算法含有许多细节，所以实现起来比较困难。并且反向传播算法有一个不好的特征容易产生一些微妙的bug，当反向传播算法和梯度下降算法或其它算法一起运行里，可能会看起来能正常运行，并且代价函数 $J(\theta)$ 在每次梯度下降的迭代中也在不断的减小，虽然在反向传播的实现中存在一些bug，但是运行情况看起来不错，虽然代价函数 $J(\theta)$ 在不断缩小，但最后得到的神经网络其误差会比在无bug的情况下高出一个量级。所以我们需要运用梯度检测查看是否反向传播算法运行正确。



在上图中我们进行数字的梯度估量，如上图所示，我们通常采用右边的双侧差分进行估量函数 $J(\theta)$ 的导数，因为右边的双侧差分比左边的单侧差分要估计的更加准确。

Parameter vector θ

$\rightarrow \theta \in \mathbb{R}^n$ (E.g. θ is “unrolled” version of $\underline{\Theta^{(1)}}, \underline{\Theta^{(2)}}, \underline{\Theta^{(3)}}$)

$$\rightarrow \theta = [\theta_1, \theta_2, \theta_3, \dots, \theta_n]$$

$$\frac{\partial}{\partial \theta_1} J(\theta) \approx \frac{J(\theta_1 + \epsilon, \theta_2, \theta_3, \dots, \theta_n) - J(\theta_1 - \epsilon, \theta_2, \theta_3, \dots, \theta_n)}{2\epsilon}$$

$$\frac{\partial}{\partial \theta_2} J(\theta) \approx \frac{J(\theta_1, \theta_2 + \epsilon, \theta_3, \dots, \theta_n) - J(\theta_1, \theta_2 - \epsilon, \theta_3, \dots, \theta_n)}{2\epsilon}$$

⋮

$$\frac{\partial}{\partial \theta_n} J(\theta) \approx \frac{J(\theta_1, \theta_2, \theta_3, \dots, \theta_n + \epsilon) - J(\theta_1, \theta_2, \theta_3, \dots, \theta_n - \epsilon)}{2\epsilon}$$

Andi

我们可以采用上一张图的方法来估测所有偏导的值

```

for i = 1:n,
    thetaPlus = theta;
    thetaPlus(i) = thetaPlus(i) + EPSILON;
    thetaMinus = theta;
    thetaMinus(i) = thetaMinus(i) - EPSILON;
    gradApprox(i) = (J(thetaPlus) - J(thetaMinus))
                    / (2*EPSILON);
end;

```

Check that $\text{gradApprox} \approx \text{DVec}$

\uparrow
From backprop.

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_i + \epsilon \\ \vdots \\ \theta_n \end{bmatrix} \xrightarrow{\theta_i - \epsilon}$$

$$\frac{\partial}{\partial \theta_i} J(\theta)$$

在Octave的实现方式如上图所示，我们通过此方法计算出所有的偏导记为gradApprox，而我们获取由反向传播得到的偏导，然后把两个相比较，如何相差不大，就表示反向传播算法运行无误。

Implementation Note:

- - Implement backprop to compute DVec (unrolled $D^{(1)}, D^{(2)}, D^{(3)}$).
- - Implement numerical gradient check to compute gradApprox.
- - Make sure they give similar values.
- - Turn off gradient checking. Using backprop code for learning.

Important:

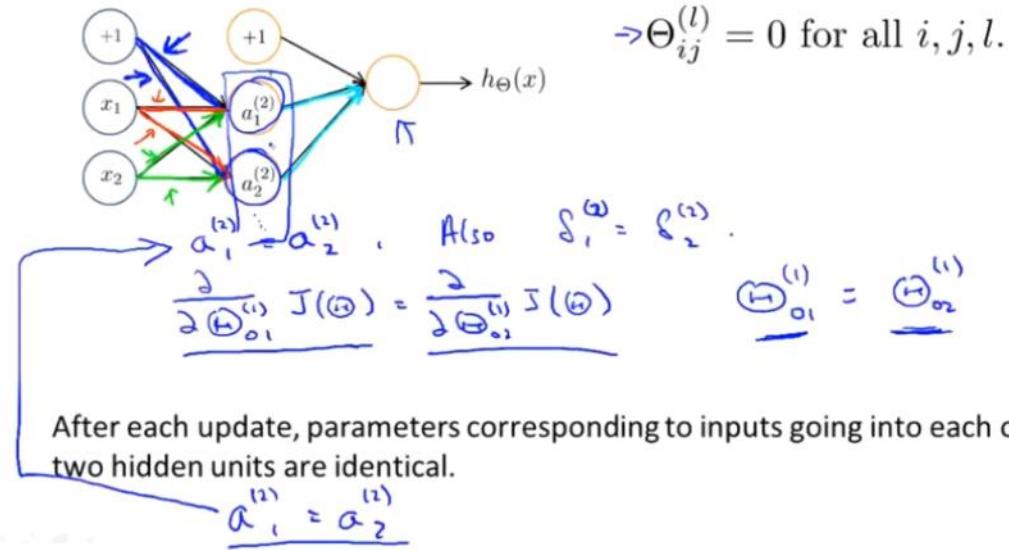
- - Be sure to disable your gradient checking code before training your classifier. If you run numerical gradient computation on every iteration of gradient descent (or in the inner loop of `costFunction(...)`) your code will be very slow.

注意，在使用反向传播的时候，需要关闭梯度检测，因为梯度检测代码计算导数比较缓慢，而反向传播计算导数比较快，所以如果同时运行，会拉低反向传播的计算速度。

(11) 随机初始化

2022年1月21日 21:04

Zero initialization



Anc

若我们把 θ_{ij}^l 全部初始化为 0，可以得到之后的 $a_1^{(2)} = a_2^{(2)}$ … 还有会使得各梯度可能会相同，所以我们需要把 θ_{ij} 进行随机初始化。

Random initialization: Symmetry breaking

→ Initialize each $\Theta_{ij}^{(l)}$ to a random value in $[-\epsilon, \epsilon]$
 (i.e. $-\epsilon \leq \Theta_{ij}^{(l)} \leq \epsilon$)

E.g.

Random 10×11 matrix (betw. 0 and 1)

→ Theta1 = $\boxed{\text{rand}(10, 11) * (2 * \text{INIT_EPSILON})}$
 $- \underline{\text{INIT_EPSILON}}$; $[-\epsilon, \epsilon]$

→ Theta2 = $\boxed{\text{rand}(1, 11) * (2 * \text{INIT_EPSILON})}$
 $- \underline{\text{INIT_EPSILON}}$;

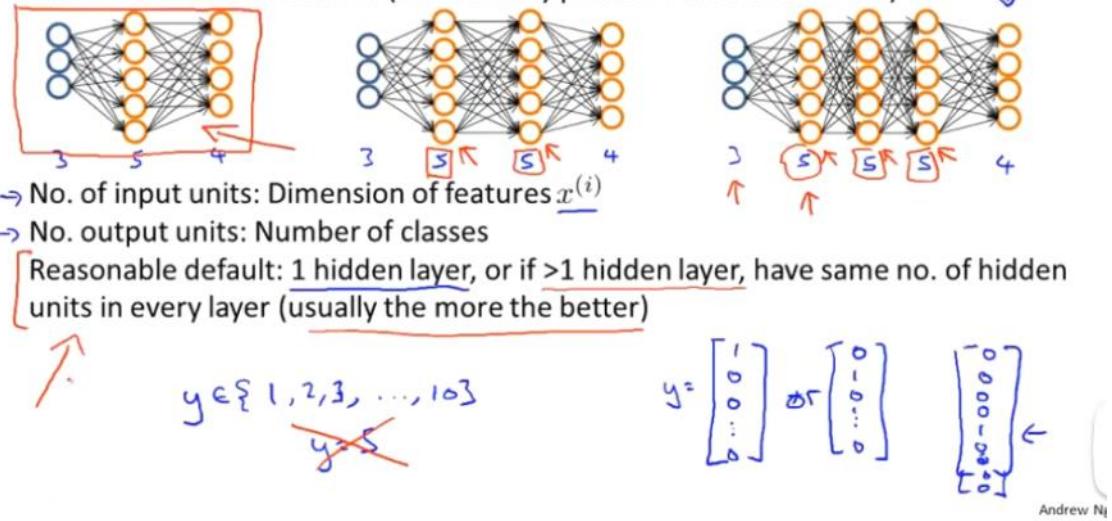
所以我们把 θ_{ij}^l 随机初始化值在范围 $[-\epsilon, \epsilon]$ ，而 ϵ 为趋近于 0 的值

(12) 总结

2022年1月21日 21:39

Training a neural network

Pick a network architecture (connectivity pattern between neurons)



我们在选择神经网络架构时需要考虑的东西。

第一、我们需确定样本 $x^{(i)}$ 的特征数量即输入单元的个数

第二、我们需要确定需要分类的数量，是二元分类还是多元分类，如果是多元分类我们需要考虑用矩阵进行输出来表示类别。

第三、神经网络的隐藏层数量，我们默认是一个隐藏层，若需要多个隐藏层，通常把隐藏层每一层的节点数目弄成相同往往效果要好。

神经网络的步骤：

Training a neural network

- 1. Randomly initialize weights
 - 2. Implement forward propagation to get $h_{\Theta}(x^{(i)})$ for any $x^{(i)}$
 - 3. Implement code to compute cost function $J(\Theta)$
 - 4. Implement backprop to compute partial derivatives $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$
- for $i = 1:m$ { $(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$, ..., $(x^{(m)}, y^{(m)})$
- Perform forward propagation and backpropagation using example $(x^{(i)}, y^{(i)})$
- (Get activations $a^{(l)}$ and delta terms $\delta^{(l)}$ for $l = 2, \dots, L$).
- $\Delta^{(2)} := \Delta^{(2)} + \delta^{(2)} (\alpha^{(2)})^T$
- compute $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$.
-
- And

Training a neural network

- 5. Use gradient checking to compare $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$ computed using backpropagation vs. using numerical estimate of gradient of $J(\Theta)$.
→ Then disable gradient checking code.
 - 6. Use gradient descent or advanced optimization method with backpropagation to try to minimize $J(\Theta)$ as a function of parameters Θ
- $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$
- $J(\Theta) - \text{non-convex.}$

第一、随机初始化权重，即随机初始化 θ_{ij}^l

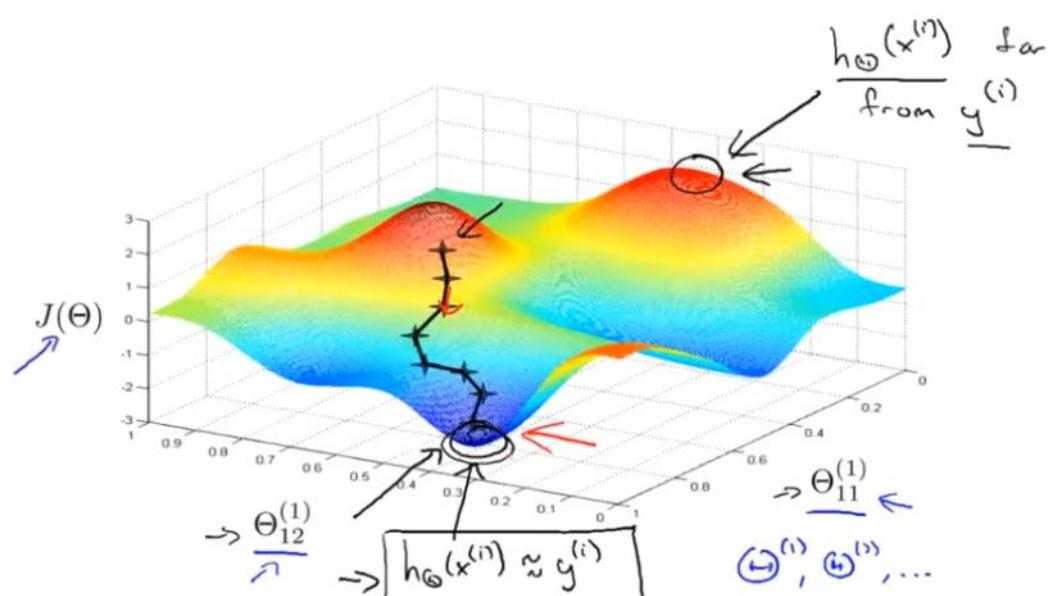
第二、实现正向传播，即把样本 $x^{(i)}$ 从正向传播的输入端给出，然后经过正向传播算法得到 $h_\Theta(x^{(i)})$ (即算法认为的概率)。

第三、实现代码去计算代码函数 $J(\theta)$

第四、实现反向传播代码去计算 $\frac{\partial}{\partial \Theta_{jk}^{(L)}} J(\Theta)$ 的部分计算。

第五、实现梯度检测算法去检验反向传播算法是否正确实现。

第六、使用梯度下降算法或高级算法与反向传播算法一起去寻找关于最小化 $J(\Theta)$ 作为关于 Θ 的函数。



Andrew Ng

总结：反向传播与梯度下降算法一起，最根本就是从随机初始化参数后的点不

断往局部最小点进行靠近，当到达局部最小点时 $h_{\theta}(x^{(i)}) \approx y^{(i)}$ 表示预测的值等于实际的值。

(1) 机器学习算法诊断

2022年1月22日 19:13

当你在学习算法或都试着改进一个机器学习系统的性能，如何进行选择或者接下来应该选择哪条道路

Debugging a learning algorithm:

Suppose you have implemented regularized linear regression to predict housing prices.

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

However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

- - Get more training examples
- Try smaller sets of features $x_1, x_2, x_3, \dots, x_{100}$
- - Try getting additional features
- Try adding polynomial features (x_1^2, x_2^2, x_1x_2 , etc.)
- Try decreasing λ
- Try increasing λ

假设你已经实现了线性回归去预测房价，但实际上的值与你所预测的值差距较大。

我们有如下几种方式去重新调整

- 一、 获取更多的训练数据
- 二、 试着设置更小的特征数目
- 三、 试着获取更多的特征数目(如增加土地范围等)
- 四、 试着添加更多的多项式特征
- 五、 尝试增加 λ
- 六、 尝试减少 λ

有方法可以将上面所尝试的方式排除一半，而首先要做就是怎样评估机器学习算法的性能。

Machine learning diagnostic:

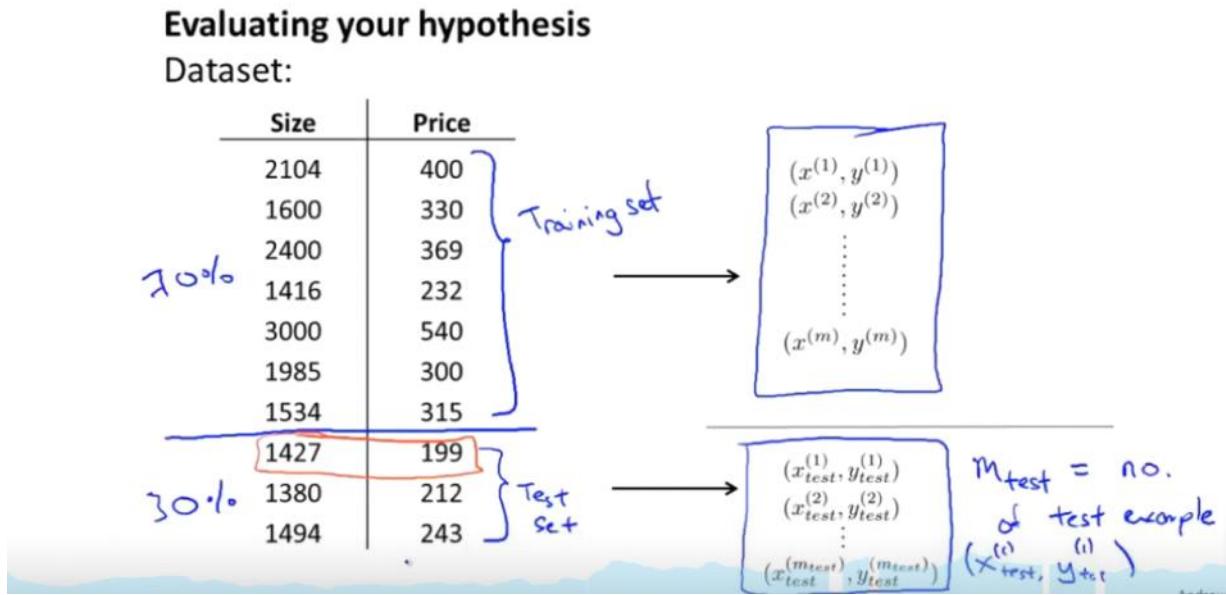
Diagnostic: A test that you can run to gain insight what is/isn't working with a learning algorithm, and gain guidance as to how best to improve its performance.

Diagnostics can take time to implement, but doing so can be a very good use of your time.

机器学习诊断法即一种测试法，通过执行这种测试法能够了解算法在哪里出了问题，即改进算法，什么样的尝试才有意义。

(2) 线性回归评估假设

2022年1月23日 11:20



我们需要随机从样本集中，随机抽取70%的数据当作训练集，把30%的数据当作测试集。

Training/testing procedure for linear regression

- - Learn parameter θ from training data (minimizing training error $J(\theta)$) $\underset{70\%}{}$

- Compute test set error:

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

我们通过70%的训练集得到最小化 $J(\theta)$ 的参数 θ 。我们把得到的参数 θ 计算出测试集的误差，由公式

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

得出

(3) 逻辑回归评估假设

2022年1月23日 17:36

Training/testing procedure for logistic regression

- - Learn parameter θ from training data m_{test}
- Compute test set error:
 - $J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} y_{test}^{(i)} \log h_\theta(x_{test}^{(i)}) + (1 - y_{test}^{(i)}) \log h_\theta(x_{test}^{(i)})$
 - Misclassification error (0/1 misclassification error):
$$\text{err}(h_\theta(x), y) = \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5, \quad y = 0 \\ & \text{or if } h_\theta(x) < 0.5, \quad y = 1 \\ 0 & \text{otherwise} \end{cases} \quad \text{error}$$
$$\text{Test error} = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \text{err}(h_\theta(x_{test}^{(i)}), y_{test}^{(i)}).$$

逻辑回归的也是从70%的训练集中得到参数 θ ，而后计算30%的测试集数据的误差。

而式子

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

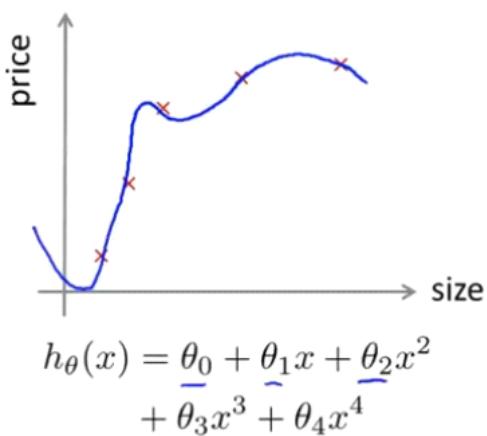
等价于下面形式

- Misclassification error (0/1 misclassification error):
$$\text{err}(h_\theta(x), y) = \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5, \quad y = 0 \\ & \text{or if } h_\theta(x) < 0.5, \quad y = 1 \\ 0 & \text{otherwise} \end{cases} \quad \text{error}$$
$$\text{Test error} = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \text{err}(h_\theta(x_{test}^{(i)}), y_{test}^{(i)}).$$

(4) 模型选择、验证、测试集

2022年1月23日 19:20

Overfitting example



Once parameters $\theta_0, \theta_1, \dots, \theta_4$ were fit to some set of data (training set), the error of the parameters as measured on that data (the training error $J(\theta)$) is likely to be lower than the actual generalization error.

可能会存在过拟合问题，如该参数的选择只对训练集的数据具有很好的拟合作用，对于新的样本没有很好的拟合，即存在对于训练集的误差并不能很好的估计出实际的泛化误差。

Model selection

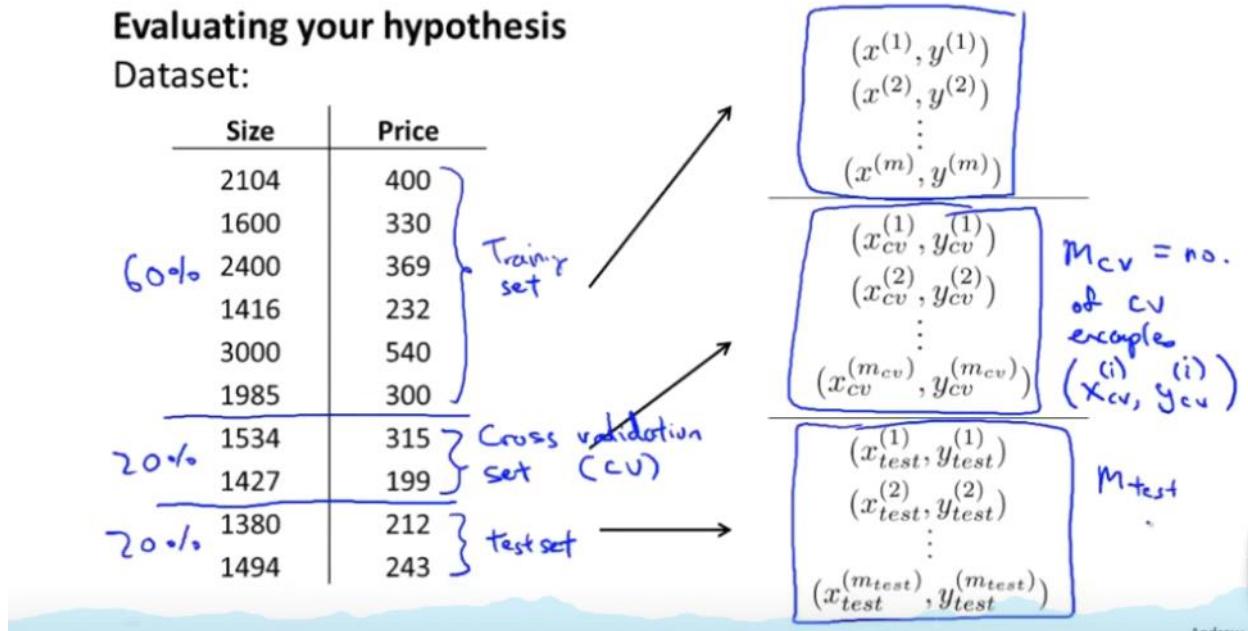
- $d=1$ 1. $\rightarrow h_{\theta}(x) = \theta_0 + \theta_1 x \rightarrow \Theta^{(1)} \rightarrow J_{test}(\Theta^{(1)})$
- $d=2$ 2. $\rightarrow h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \rightarrow \Theta^{(2)} \rightarrow J_{test}(\Theta^{(2)})$
- $d=3$ 3. $\rightarrow h_{\theta}(x) = \theta_0 + \theta_1 x + \dots + \theta_3 x^3 \rightarrow \Theta^{(3)} \rightarrow J_{test}(\Theta^{(3)})$
- \vdots \vdots
- $d=10$ 10. $\rightarrow h_{\theta}(x) = \theta_0 + \theta_1 x + \dots + \theta_{10} x^{10} \rightarrow \Theta^{(10)} \rightarrow J_{test}(\Theta^{(10)})$

Choose $\boxed{\theta_0 + \dots + \theta_5 x^5}$ ↩

How well does the model generalize? Report test set error $\underline{J_{test}(\theta^{(5)})}$.

Problem: $J_{test}(\theta^{(5)})$ is likely to be an optimistic estimate of generalization error. I.e. our extra parameter (d = degree of polynomial) is fit to test set.

在上图中，我们分别给出测试集在模型从1到10中选中最优的模型即寻找最小 $J(\theta)$ ，然而可能会存在过拟合问题，因为拟合了一个额外的参数d（即多项式的指数）所以可能存在过拟合问题。



上图表示我们把整个样本集分为训练集、验证集、测试集三部分

Train/validation/test error

Training error:

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

Cross Validation error:

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

Test error:

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

我们有上图的估计误差的公式

Model selection

$$\begin{aligned}
 & \text{d=1} \quad 1. \quad h_{\theta}(x) = \theta_0 + \theta_1 x \rightarrow \min_{\theta} J(\theta) \rightarrow \Theta^{(1)} \rightarrow J_{cv}(\Theta^{(1)}) \\
 & \text{d=2} \quad 2. \quad h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \rightarrow \Theta^{(2)} \rightarrow J_{cv}(\Theta^{(2)}) \\
 & \text{d=3} \quad 3. \quad h_{\theta}(x) = \theta_0 + \theta_1 x + \dots + \theta_3 x^3 \rightarrow \Theta^{(3)} \rightarrow J_{cv}(\Theta^{(3)}) \\
 & \vdots \\
 & \text{d=10} \quad 10. \quad h_{\theta}(x) = \theta_0 + \theta_1 x + \dots + \theta_{10} x^{10} \rightarrow \Theta^{(10)} \rightarrow J_{cv}(\Theta^{(10)})
 \end{aligned}$$

$d=4$ ↑

Pick $\theta_0 + \theta_1 x_1 + \dots + \theta_4 x^4$ ↘

Estimate generalization error for test set $J_{test}(\Theta^{(4)})$ —

上图中我们表示，首先要使用验证集去合适的模型，进而得到合适的参数d,然后使用得到的参数去放入测试集中去得到误差

(5) 诊断偏差、方差

2022年1月24日 17:25

当运行一个算法时，如果这个算法运行的不理想，会出现两种情况，要么是偏差比较大，要么是方差比较大。即要么是过拟合问题要么是欠拟合问题。

注：

符号的定义：

符号	涵义
\mathbf{x}	测试样本
D	数据集
y_D	\mathbf{x} 在数据集中的标记
y	\mathbf{x} 的真实标记
f	训练集 D 学得的模型
$f(\mathbf{x}; D)$	由训练集 D 学得的模型 f 对 \mathbf{x} 的预测输出
$\bar{f}(\mathbf{x})$	模型 f 对 \mathbf{x} 的 期望预测输出 @Microstrong

如对于回归任务来说，学习算法的期望预测为：

$$\bar{F}(x) = E_D[f(x; D)]$$

这里的期望预测是针对不同数据集 D ，模型 f 对样本 \mathbf{x} 的预测值取其期望，也叫平均预测。

(1) 方差的定义：

使用样本数相同的不同训练集产生的方差为

$$var(x) = E_d[(f(x; D) - \bar{F}(x))^2]$$

方差的含义：方差度量了同样大小的训练集的变动所导致的学习性能的变化，即刻画了数据扰动所造成的影响。

(2) 偏差的含义

期望输出与真实标记的差别称为偏差，即：

$$bias^2(x) = (\bar{F}(x) - y)^2$$

偏差的含义：偏差度量了学习算法的期望预测与真实结果的偏离程度，即刻画了学习算法本身的能力。

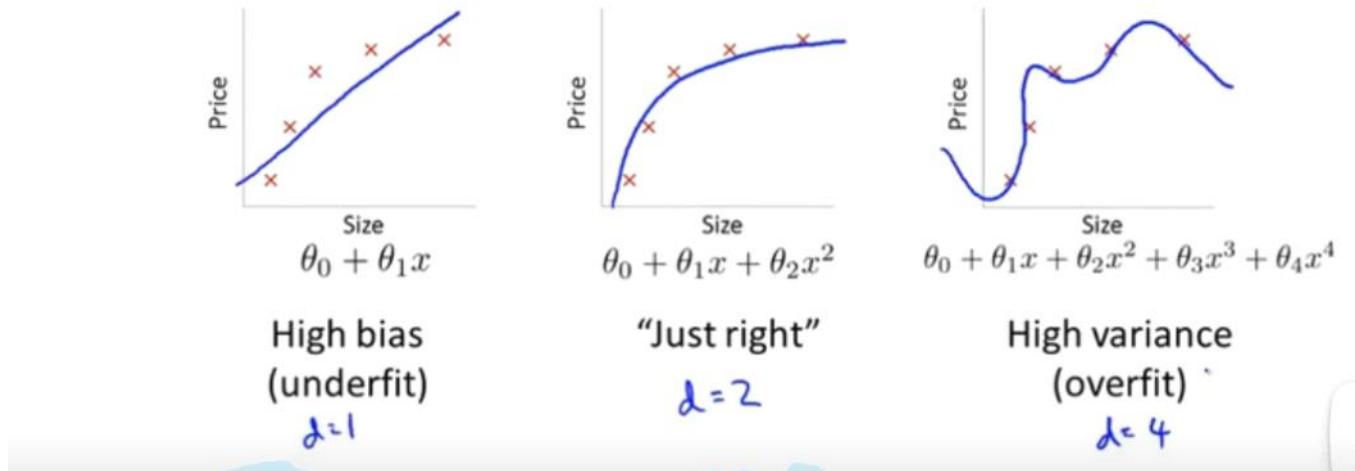
方差和偏差的概念

偏差 (Bias)：预测值和真实值之间的误差

方差 (Variance)：预测值之间的离散程度，也就是离其期望值的

距离。方差越大，数据的分布越分散。

Bias/variance

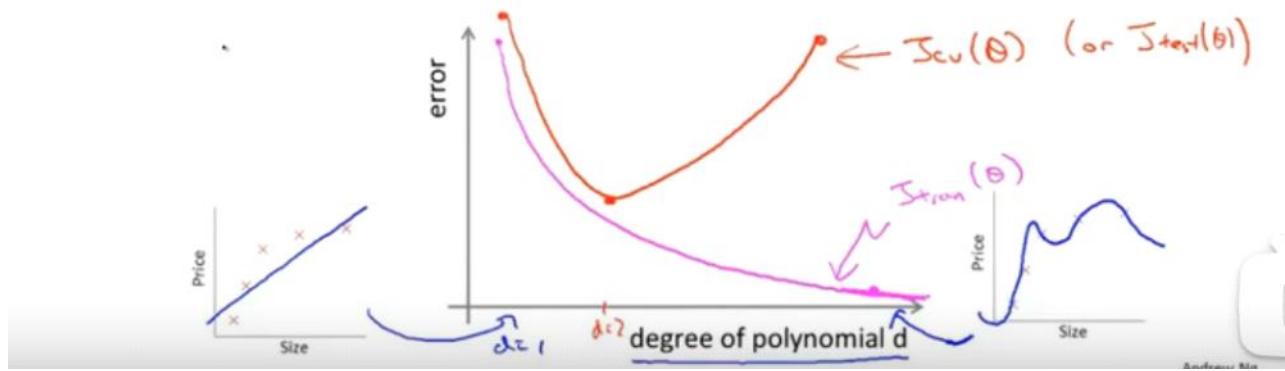


如上图所示，左边表示高偏差即欠拟合，右边表示高方差，即过拟合。

Bias/variance

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

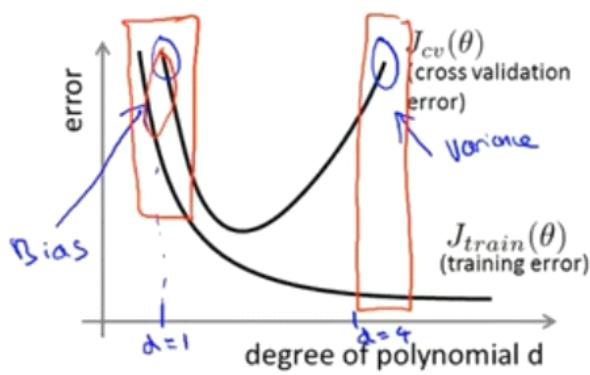
$$\text{Cross validation error: } J_{\text{cv}}(\theta) = \frac{1}{2m_{\text{cv}}} \sum_{i=1}^{m_{\text{cv}}} (h_\theta(x_{\text{cv}}^{(i)}) - y_{\text{cv}}^{(i)})^2 \quad (\text{or } J_{\text{test}}(\theta))$$



上图的训练误差公式和交叉验证误差公式如上图所示

Diagnosing bias vs. variance

Suppose your learning algorithm is performing less well than you were hoping. ($J_{cv}(\theta)$ or $J_{test}(\theta)$ is high.) Is it a bias problem or a variance problem?



Bias (underfit):

$$\left. \begin{array}{l} J_{train}(\theta) \text{ will be high} \\ J_{cv}(\theta) \approx J_{train}(\theta) \end{array} \right\}$$

Variance (overfit):

$$\left. \begin{array}{l} J_{train}(\theta) \text{ will be low} \\ J_{cv}(\theta) \gg J_{train}(\theta) \end{array} \right\}$$

⇒

An

所上图所示，

偏差过大，所显示出来的特征是 $J_{train}(\theta)$ 非常高时，那么 $J_{cv}(\theta) \approx J_{train}(\theta)$ 。
当方差过大时，所显示出来的特征是 $J_{train}(\theta)$ 非常低时，那么 $J_{cv}(\theta)$ 远大于 $J_{train}(\theta)$ 。

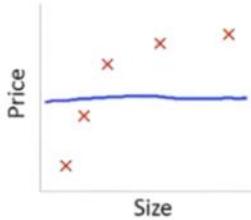
(6) 正则化和偏差、方差

2022年1月25日 16:37

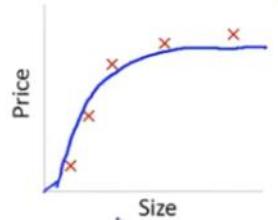
Linear regression with regularization

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

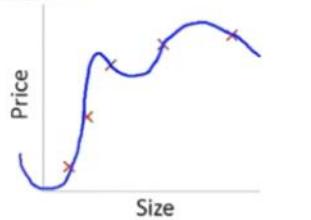
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^m \theta_j^2 \quad \leftarrow$$



Large $\lambda \leftarrow$
 → High bias (underfit)
 $\rightarrow \lambda = 10000, \theta_1 \approx 0, \theta_2 \approx 0, \dots$
 $h_{\theta}(x) \approx \theta_0$



Intermediate $\lambda \leftarrow$
 "Just right"



Small $\lambda \rightarrow$
 High variance (overfit)
 $\rightarrow \lambda = 0$

在模型中我们有如上公式，当 λ 的选择过大时，我们会出现高偏差（欠拟合），当 λ 的选择过小时，我们会出现高方差（过拟合）。

Choosing the regularization parameter λ

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

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

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

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

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

在挑选规则化的参数 λ 时，我们有以上公式

Choosing the regularization parameter λ

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

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

1. Try $\lambda = 0$ $\rightarrow \min_{\theta} J(\theta) \rightarrow \theta^{(0)} \rightarrow J_{cv}(\theta^{(0)})$
 2. Try $\lambda = 0.01$ $\rightarrow \min_{\theta} J(\theta) \rightarrow \theta^{(1)} \rightarrow J_{cv}(\theta^{(1)})$
 3. Try $\lambda = 0.02$ $\rightarrow \min_{\theta} J(\theta) \rightarrow \theta^{(2)} \rightarrow J_{cv}(\theta^{(2)})$
 4. Try $\lambda = 0.04$
 5. Try $\lambda = 0.08$
 :
 12. Try $\lambda = 10$ $\rightarrow \theta^{(12)} \rightarrow J_{cv}(\theta^{(12)})$
 ↑ 10 · 24 Pick (say) $\theta^{(5)}$. Test error: $J_{test}(\theta^{(5)})$

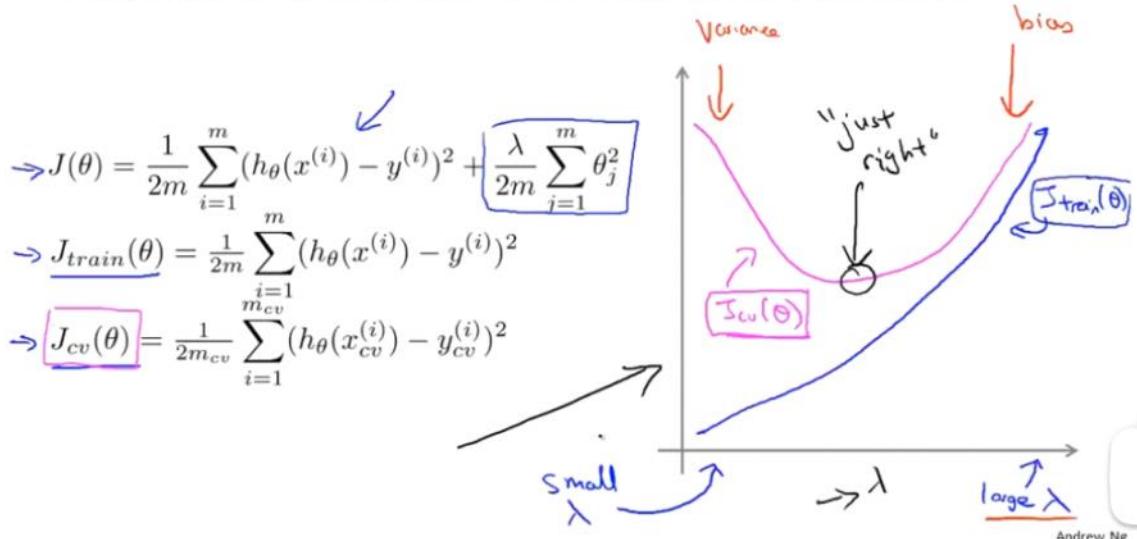
Andrew Ng

在上图中我们需要挑选合适的参数 λ ，我们需要不断依次尝试对于参数 λ 的调整，可以把每次步长设置成 2 倍，依次递增。在这种模型中我们选择误差最小的那部分模型。

注：

我们这部分数据在验证集中测试，而在测试集中测试选择的参数 λ 的表现。

Bias/variance as a function of the regularization parameter λ

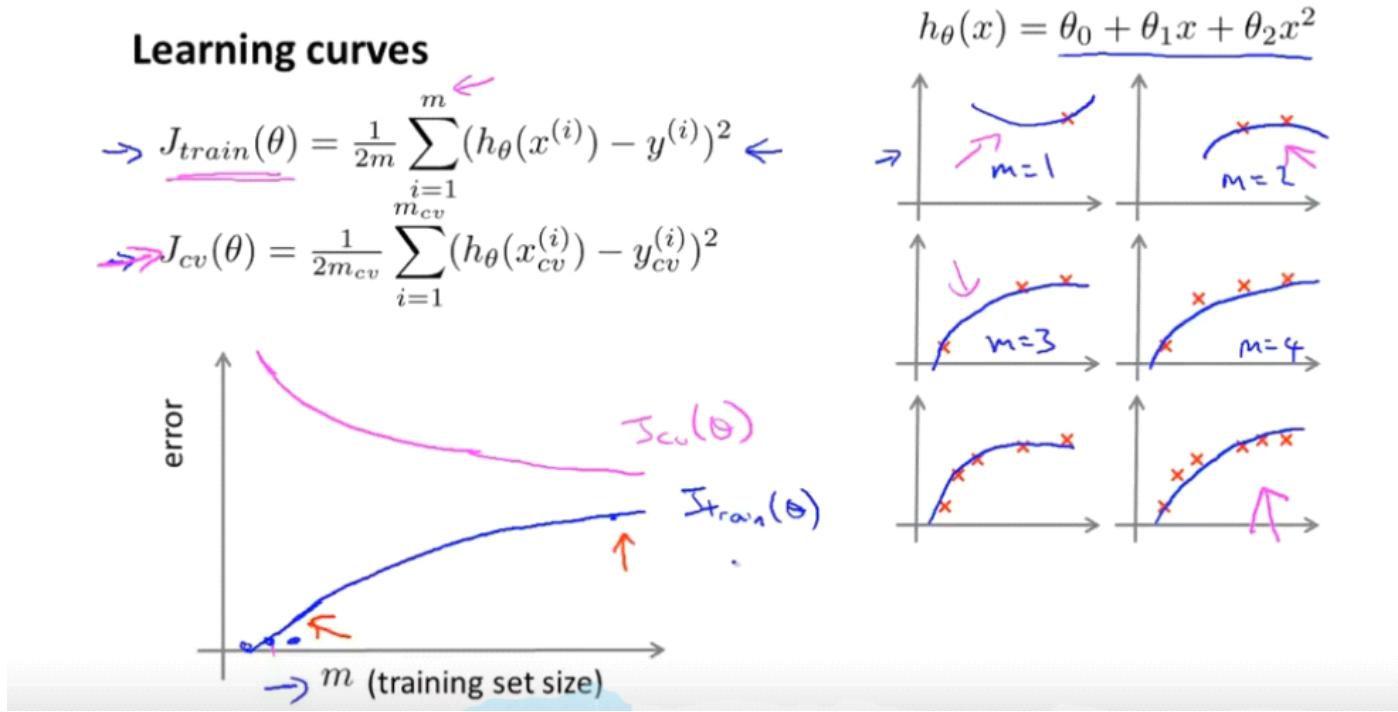


偏置和方差的权衡

在上图中，首先可以看到训练集和验证集都没有进行正则化，他们在图中的表现如右图所示，当参数 λ 过小时，会出现方差问题，而参数 λ 过大时，会出现偏差问题，所示我们需要找到一个合适的参数 λ ，使得恰好存在一个参数 λ ，使得方差和偏差都没有距离过大。

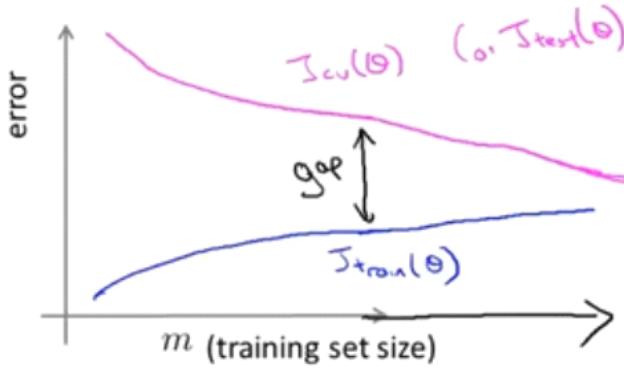
(7) 学习曲线

2022年1月25日 17:01



我们从上图中可以看到，在样本较小时非常好拟合，而随机样本数据的增大，想要较好的拟合会变得比较困难，我们可以挑选一些样本集，来绘制误差曲线。

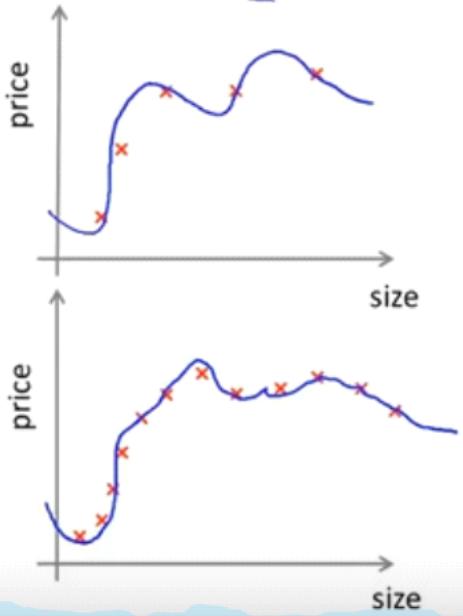
High variance



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

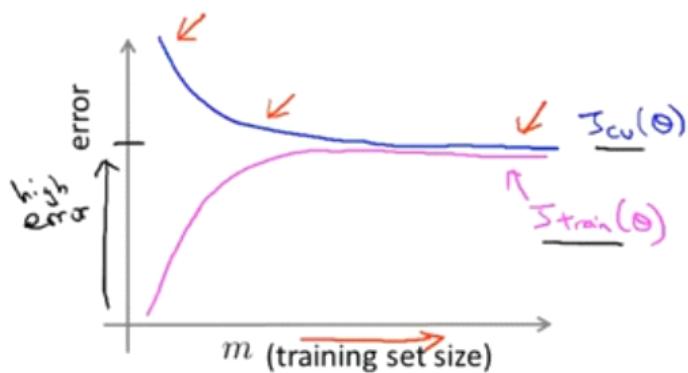
$$h_{\theta}(x) = \theta_0 + \theta_1 x + \dots + \theta_{100} x^{100}$$

(and small λ) ↗

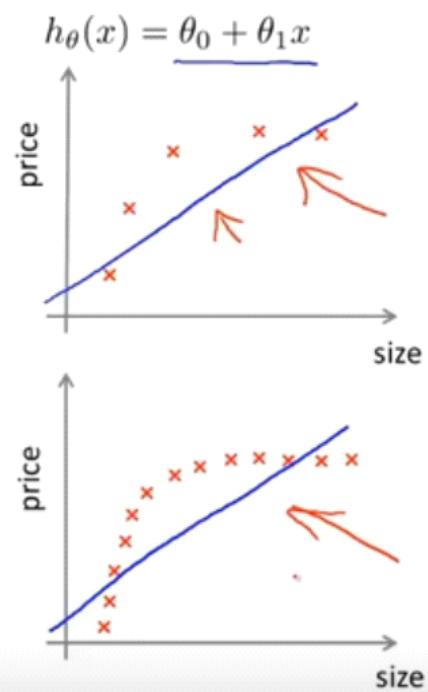


如上图所示，当训练集的 $J_{train}(\theta)$ 函数的学习曲线不断上升时，而验证集或测试集的 $J_{cv}(\theta)$ 的学习曲线不断下降时，这两条曲线之间有较大的误差，存在过拟合状态，这是表示算法有高的方差，此时使用更多的样本去训练是有帮助的。

High bias



If a learning algorithm is suffering from high bias, getting more training data will not (by itself) help much.



Andrew

如上图所示，存在着高偏差，即预测值与真实值之间存在一定的差距，此时加大样本的练习力度无法得到提高

(8) 总结

2022年1月25日 20:03

Debugging a learning algorithm:

Suppose you have implemented regularized linear regression to predict housing prices. However, when you test your hypothesis in a new set of houses, you find that it makes unacceptably large errors in its prediction. What should you try next?

- Get more training examples → fixes high variance
- Try smaller sets of features → fixes high variance
- Try getting additional features → fixes high bias
- Try adding polynomial features (x_1^2, x_2^2, x_1x_2 , etc) → fixes high bias.
- Try decreasing λ → fixes high bias
- Try increasing λ → fixes high variance .

学习算法的调试：

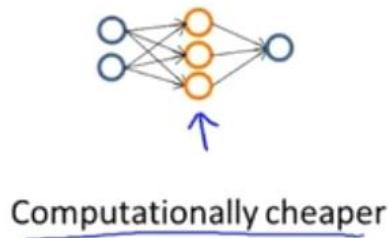
假设你已经实现了已经正则化的线性回归去预测房屋的价格，然而，当你在一个新的房屋集合去测试你的假设时，你发现在这个预测中有着不可接受的较大误差，你的下一次应该怎样去尝试？

- 1、获取更多的训练样本——修复高方差
- 2、尝试更少的特征集合——修复高方差
- 3、尝试获取额外的特征——修复高偏差
- 4、尝试添加多项式特征——修复高偏差
- 5、尝试增加参数 λ ——修复高偏差
- 6、尝试减少参数 λ ——修复高方差

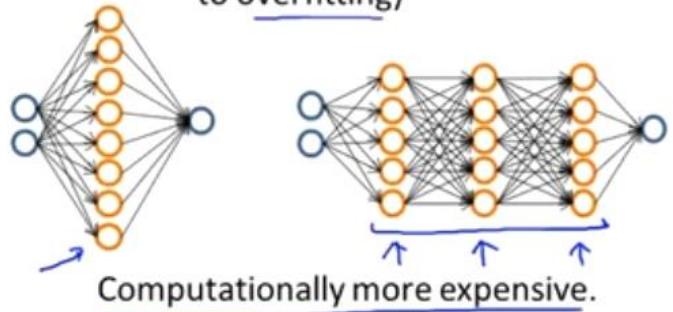
注：因为如果存在高偏差，那么代表这是一个欠拟合的状态，根据正则化的含义，当出现欠拟合时，我们可以增加参数 λ ，使其去修复高偏差。如果存在高方差时，代表这是一个过拟合的状态，根据正则化的含义，减少参数 λ ，使其去修改高方差。

Neural networks and overfitting

→ “Small” neural network
(fewer parameters; more prone to underfitting)



→ “Large” neural network
(more parameters; more prone to overfitting)



Use regularization (λ) to address overfitting.

在上图中，左图表示一个小型神经网络，该网络存在的特征为，有较少的参数，但容易出现欠拟合状态。计算量小。

右图表示一个大型神经网络，该网络存在的特征为，有更多的参数，容易出现过拟合状态，计算量大，还要去选择网络的架构是三层网络或四层网络以及更多，使用正则化可以去解决过拟合的状态。

(1) 误差分析

2022年1月26日 19:59

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.

对于误差分析的推荐方法是

- 一、开始一个简单的算法并快速去实现它，然后用验证集去测试数据。
- 二、绘制学习曲线来看是需要更多的数据或者是需要更多的特征等。
- 三、进行错误分析，手动去测试在样本中，设计算法的误差的特征。

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

Replica/fake: 4

→ Steal passwords: 53

Other: 31

→ Deliberate misspellings: 5

(mOrgage, med1cine, etc.)

→ Unusual email routing: 16

→ Unusual (spamming) punctuation: 32

在上图中是一个邮件分类算法，如出现了错100个错误的分析情况，有如下方法去分析。

一、察看邮件的类别

二、邮件都具有什么特征，这能帮助你的算法能正确的分类。

The importance of numerical evaluation

Should discount/discounts/discounted/discounting be treated as the same word?

Can use “stemming” software (E.g. “Porter stemmer”)
universe/university.

Error analysis may not be helpful for deciding if this is likely to improve performance. Only solution is to try it and see if it works.

Need numerical evaluation (e.g., cross validation error) of algorithm’s performance with and without stemming.

Without stemming: 5% error With stemming: 3% error

Distinguish upper vs. lower case (Mom/mom): 3.2%

在进行误差分析时，一个技巧是保证自己对学习算法有一种数值估计的方法，即在改进学习算法时，如果算法能够返回一个数值评价的指标一定要估计算法执行的效果将会对于改进算法有帮助。

如在上图中，表示我们是否在识别一些单词时，只看前几个字母是否相同，我们可以进行单词全识别进行一个误差的评估，再只进行前几个字母相同进行一个误差的评估，把两者进行相比，看哪个算法更加适合。

(2) 不对称性分类的误差评估

2022年1月26日 20:29

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% accy (0.8% error)
→ 99.5% accy (0.5% error)

在上图中，我们用逻辑回归训练模型，设 $y=1$ 表示阳性， $y=0$ 表示其它。我们发现在测试集中有99%的正确诊断，有1%的错误。而实际上只有0.5%的病人表阳性。这表示有0.5%的误差，而若我们有一个函数把所有输入参数 x 都预测为阴性，这样也有0.5%的误差。若我们有未改进算法的正确率为99.2%(0.8%的误差)，而改进算法之后有99.5%的正确率(0.5%)，此时我们需要弄清这部分的正确率究竟是真正的改进了，还是一段直接把输入参数设为0的函数。

注：在上图中存在一个偏斜类问题，即正样本和负样本的比例非常接近于一个极端的情况，在该例子中阴性的比例非常少，有偏斜类问题，比较喜欢出现上图所显示的情况。

Precision/Recall

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

		Actual class		→ Precision
		1	0	(Of all patients where we predicted $y = 1$, what fraction actually has cancer?)
Predicted class	1	True positive	False positive	$\frac{\text{True positive}}{\#\text{predicted positive}} = \frac{\text{True positive}}{\text{True pos} + \text{False pos}}$
	0	False negative	True negative	→ Recall (Of all patients that actually have cancer, what fraction did we correctly detect as having cancer?)
$y = 0$		$\text{Recall} = 0$		$\frac{\text{True positives}}{\#\text{actual positives}} = \frac{\text{True positives}}{\text{True pos} + \text{False neg}}$

查准率/召回率

我们一般把极少数的情况设为 $y=1$ 。

查准率(在我们预测y=1的所有病人中，其实上有癌症的点比多少?)

$$\frac{\text{True positives}}{\#\text{predict positive}} = \frac{\text{true positive}}{\text{true pos} + \text{false pos}}$$

召回率(在实际上有癌症的所有病人中，我们正确预测癌症的占比多少?)

$$\frac{\text{True positives}}{\#\text{actual positive}} = \frac{\text{true positive}}{\text{true pos} + \text{false neg}}$$

注：若查准率较高，而召回率较低，表示此时设置y=1的条件阈值过高，导致虽然预测y=1较准，但预测y=1的数据规模较小，就导致在实际上有癌症的所有病人中，能正确预测癌症的占比较少。而把y=1的条件阈值设置过低，出会导致召回率虽然高，但是查准率较低。

(3) 精确度和召回率的平衡

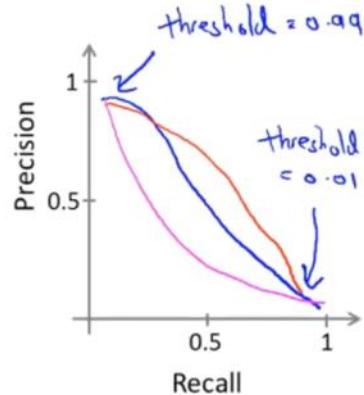
2022年1月28日 13:51

Trading off precision and recall

- Logistic regression: $0 \leq h_\theta(x) \leq 1$
Predict 1 if $h_\theta(x) \geq 0.5$ ~~0.7~~ ~~0.9~~ ~~0.3~~ ↗
Predict 0 if $h_\theta(x) < 0.5$ ~~0.7~~ ~~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.

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

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



Andrew Ng

我们在设置阈值时，不同阈值的设置会导致不同的精准度和召回率的变化。如上图所示若我们预测函数 $h_\theta(x) \geq 0.7$ 时我们预测 $y=1$ (即患有癌症)，当函数 $h_\theta(x) < 0.7$ 时，我们预测 $y=0$ (即病人没有患有癌症)。如若这样做，除非把握十足否则后果十分严重。以这种方式来预测，会导致模型具有高的准确病，较低的召回率。所以当函数 $h_\theta(x)$ 大于某个阈值时预测为 1 显得尤为重要。如右上方的图表所示召回率和查准率的图像显示。

注：若看不懂上面的内容可以参考上一章的注解。

F₁ Score (F score)

How to compare precision/recall numbers?

	Precision(P)	Recall (R)	Average	F ₁ 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: ~~$\frac{P+R}{2}$~~

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

$P=0 \text{ or } R=0 \Rightarrow F\text{-score} = 0.$

$P=1 \text{ and } R=1 \Rightarrow F\text{-score} = 1.$

Predict $y=1$ all the time

如何去比较查准率/召回率的比值?

如上图所示, 若我们采用公式 $\frac{P+R}{2}$ 来表示评估数字, 但这个方法并不合适。因为若一方的数字远大于另一方的数字, 也会导致评估的数字较大。因为我们需要查准率和召回率都不会过大, 这才是一个比较好的模型。所以此方法并不适用。帮我们采用公式 $2 \frac{PR}{P+R}$ 来进行评估。

(4) 机器学习数据

2022年1月28日 15:30

在之前的章节中有讲授过不要盲目的收集过多的数据，这是因为大量的数据，只是在一定的情况下起作用。但事实证明，在一定的条件下，收集大量的数据是有效的。这里给出这些条件是什么，在这些条件下得到大量的数据存在某种类型的学习算法中进行训练，可以是一些有效的方法来获得一个具有良好性能的学习算法，这种情况往往出现在这些条件对于你的问题都成立，并且你可能获得大量数据的情况下，这可以是一个很好的方式来获得非常高性能的学习算法。

Designing a high accuracy learning system

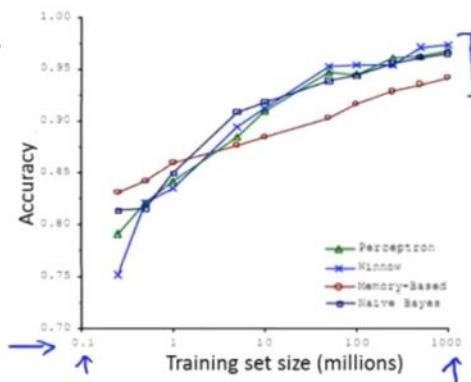
E.g. Classify between confusable words.

{to, two, too} {then, than}

→ For breakfast I ate two eggs.

Algorithms

- - Perceptron (Logistic regression)
- - Winnow
- - Memory-based
- - Naïve Bayes



"It's not who has the best algorithm that wins.

It's who has the most data."

[Banko and Brill, 2001]

在上图的例子中，想要设计一个高准确的学习系统，并进行混淆单词之间的分类，并且采用了四种学习算法，如上图中右边的图表所示，随着数据的加大，各学习算法的准确用词也在不断的提高。

Large data rationale

- Assume feature $x \in \mathbb{R}^{n+1}$ has sufficient information to predict y accurately.
- Example: For breakfast I ate two eggs. ↙ ~~too, too~~ ←
- Counterexample: Predict housing price from only size (feet²) and no other features. ←
- Useful test: Given the input x , can a human expert confidently predict y ?

大数据的内在逻辑

假设特征x有足够的信息去预测标签y。

例子：for breakfast I ate two eggs.

在这个例子中，因为有空格前后的信息，所以我们才可以去预测此空应该填two，即在该例子中对于y，我们有足够的信息去预测。

反例：只有房子的大小而没有其它特征去预测房屋的价格。

在这个例子中，因为只有房子的大小而没有地段等情况，所以无法去预测房屋的价格。

Large data rationale

- Use a learning algorithm with many parameters (e.g. logistic regression/linear regression with many features; neural network with many hidden units). low bias algorithms. ←

→ $J_{train}(\theta)$ will be small.

Use a very large training set (unlikely to overfit) low variance ←

→ $J_{train}(\theta) \approx J_{test}(\theta)$

→ $J_{test}(\theta)$ will be small

我们在使用具有大量参数的学习算法时，这种算法具有低偏差的特征。因为算

法具有大量参数，所以会导致 $J_{train}(\theta)$ 很小，即对于训练集而言所预测的值与实际的值误差很小。又当我们使用非常大的训练集时，此时不可能产生过拟合的状态，即具有低方差的特征，故会导致 $J_{train}(\theta) \approx J_{test}(\theta)$ ，所以 $J_{test}(\theta)$ 的误差也会比较小，所以这是一个非常好的理想情况。

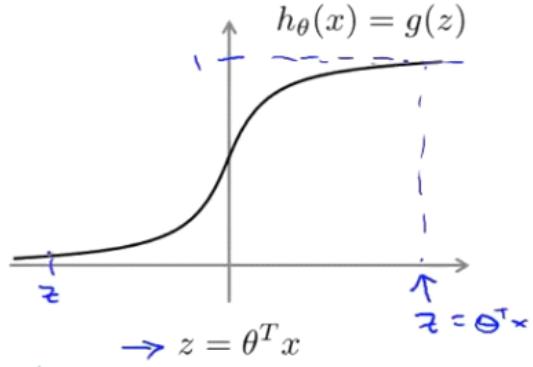
(1) 支持向量机

2022年1月28日 16:00

支持向量机与logistic回归与神经网络相比，支持向量机（support vector machine,SVM）在学习复杂的非线性方程时能够提供一种更为清晰和更加强大的方式。

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 $h_{\theta}(x) \approx 0$, $\theta^T x \ll 0$

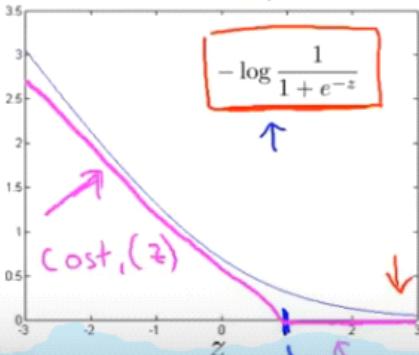
上图以logistic回归的函数构成，在logistic回归中，若 $y=1$ ，则我们需要使函数 $h_{\theta}(x) \approx 1$ ，并要使参数 $\theta^T x$ 远大于0。而若要 $y=0$ ，则我们需要使函数 $h_{\theta}(x) \approx 0$, $\theta^T x$ 远小于0。

Alternative view of logistic regression

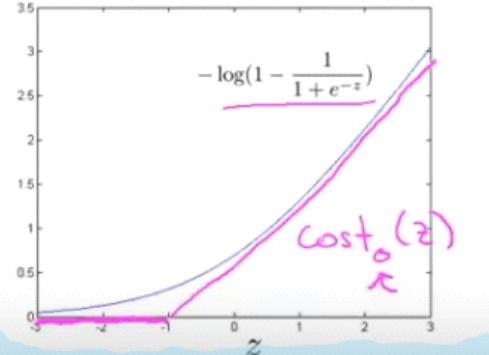
$$\text{Cost of example: } -(y \log h_\theta(x) + (1-y) \log(1 - h_\theta(x))) \leftarrow$$

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

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



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



在上图中我们给出了logistic回归的代价函数以及图像，而向量机的代价函数与logistic回归的代价函数较为相似，其中图像左边粉红色的线条表示支持向量机在 $y=1$ 时的代价函数，支持向量机在 $y=1$ 时，会使 $z < 1$ 呈直线， $z \geq 1$ 呈水平，我们把该函数称为 $cost_1(z)$ ，支持向量机在 $y=0$ 时的代价函数图像如图中右边所示，支持向量机在 $y=0$ 时，会使 $z < -1$ 呈水平， $z \geq -1$ 呈直线，我们把该函数称为 $cost_0(z)$ 。

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:

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

$$\min_u \frac{(u-s)^2 + 1}{2} \rightarrow u=5$$

$$\min_u |u(s-u)| + 1 \rightarrow u=5$$

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

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

支持向量机的代价函数与logistic回归的代价函数相似。

首先支持向量机的代价函数对比于logistic回归的代价函数而言去掉了常量¹这是因为就算去掉了常量，对于最小化代价函数所得的结果与未去掉结果不会改变。

例：

有 $\min_u (u-5)^2 + 1$ 我们可以得出当 $u=5$ 时，函数 $(u - 5)^2 + 1$ 最小。

但当 $\min_u ((u - 5)^2 + 1) \times 10$ 时，我们仍可以得出 $u=5$ ，使得函数最小。

其次我们若把logistic回归的代价函数分成两部分。

我们把

$$\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} (-\log h_\theta(x^{(i)})) + (1 - y^{(i)}) (-\log(1 - h_\theta(x^{(i)}))) \right]$$

当作A。

我们把

$$\frac{1}{2M} \sum_{j=1}^n \theta_j^2$$

当作B

则logistic回归的代价函数由 $A + \lambda B$ 组成，即前部分A不变，B的权重由正则化参数 λ 控制B在公式的权重的大小。

而在支持向量机的代价函数中，由 $CA + B$

组成，即B不变，A的权重由正则化参数 λ 控制A在公式的权重大小。

在logistic回归中若想要A在公式中占比较大，可以令参数 λ

极小使得B可以忽略不计，使得A在公式中占比较大，也可以令参数 λ

极大使得A可以忽略不计，使得B在公式中占比较大。

支持向量机也是同样的原理，只不过把参数改到A上，也可以调整得到A或B在公式上的占比。

另外我们可以使得 $C = \frac{1}{\lambda}$ ，这在某种程度会使得logistic和支持向量机相等，但参数C并不总等于 $\frac{1}{\lambda}$ ，在此处只是一个举例。

在支持向量机的公式

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

我们通过上面的公式寻找到合适的参数 θ

SVM hypothesis

$$\Rightarrow \min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^n \theta_j^2$$

Hypothesis:

$$h_{\theta}(x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

*

支持向量机的假设

我们通过最小化公式得到合适的参数。

支持向量机的假设不是给出一个预测输出，而是直接通过公式

$$h_{\theta}(x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

得到

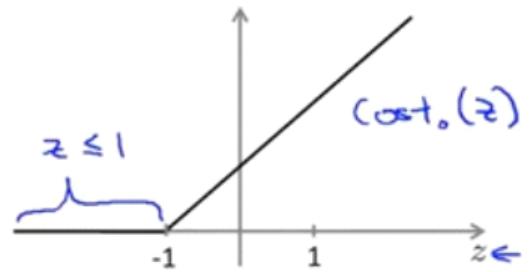
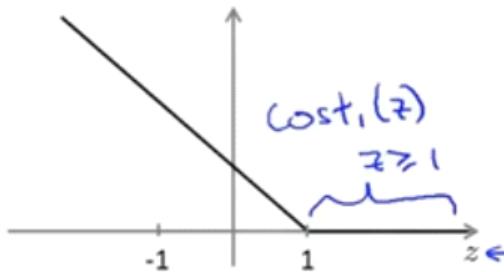
(2) 直观上对于大间隔的理解

2022年1月30日 9:19

支持向量机有时也会称为大间隔

Support Vector Machine

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



\rightarrow If $y = 1$, we want $\underline{\theta^T x \geq 1}$ (not just ≥ 0)

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

$\theta^T x \geq \alpha 1$
 $\theta^T x \leq \alpha -1$

$$C = 100,000$$

如上图所示，图中为支持向量机的假设函数图像，我们需要当 $y=1$ 时 $\theta^T x \geq 1$ 。而若 $y=0$ 时， $\theta^T x \leq -1$ 。因为支持向量机的假设为当参数 $\theta^T x \geq 0$ 时，就表示假设函数为1，而当参数 $\theta^T x \leq 0$ 时，就表示假设函数为0，所以这是当 $y=1$ 时，令 $\theta^T x \geq 1$ 表示 y 与假设函数所得结果一致，故没有差误，所以可以看到此时整个代价函数为0， $y=0$ 也是相同的结果。至于要令 $\theta^T x \geq 1$ 和 $\theta^T x \leq -1$ 而不是0，是要保持一定的安全间隔(见后面)。

SVM Decision Boundary

$$\min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

Whenever $y^{(i)} = 1$:

$$\theta^T x^{(i)} \geq 1$$

$$\min_{\theta} C \sum_{i=1}^m \left[\text{cost}_1(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

$$\text{s.t. } \theta^T x^{(i)} \geq 1 \quad \text{if } y^{(i)} = 1$$

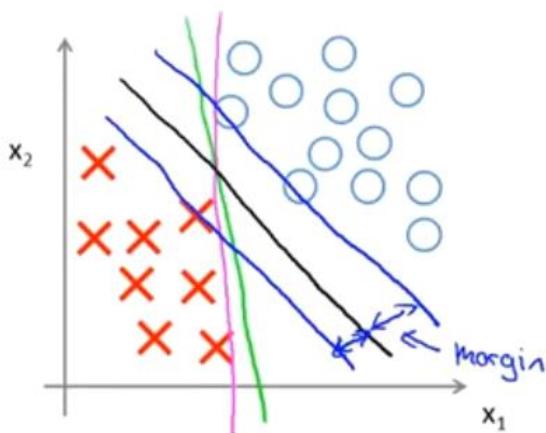
$$\theta^T x^{(i)} \leq -1 \quad \text{if } y^{(i)} = 0.$$

Whenever $y^{(i)} = 0$:

$$\theta^T x^{(i)} \leq -1$$

支持向量机的决定边策如上图所示。

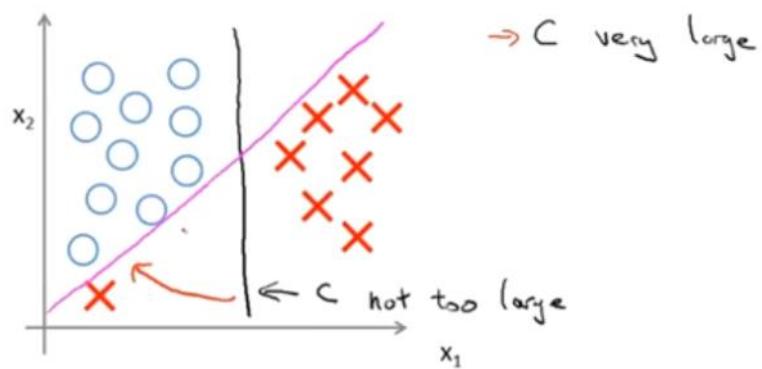
SVM Decision Boundary: Linearly separable case



Large margin classifier

如上图所示，黑线为支持向量机所描绘出来用来分割正负样本的决策边界，

Large margin classifier in presence of outliers



如上图所示，当公式中常数C若被设置得足够大的话，那么它的决策边界将从黑线变为紫线，而若C没有被设置得足够大的话，它的决策边界将为黑线。

(3) 大间隔分类器的数学原理

2022年1月31日 19:38

从直观上理解，假定SVM的Loss中，C值非常大(比如是10000000)，那么此时为了使Loss最小化，优化过程就会倾向于使C后面一项非常趋近于0(这样才能使加号前边这项变小)，那么就会到最

$$\min_{\theta} \left(0 + \frac{1}{2} \sum_{j=1}^n \theta_j^2 \right)$$

即

$$\min_{\theta} \left(\frac{1}{2} \sum_{j=1}^n \theta_j^2 \right) \quad (1)$$

同时，根据我们改造的Loss，如果C后面的那一项非常趋近于0，就一定会有

$$\min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

Whenever $y^{(i)} = 1$: $\theta^T x^{(i)} \geq 1$

Whenever $y^{(i)} = 0$: $\theta^T x^{(i)} \leq -1$

即对于任意 $y^{(i)} = 1$, 都要满足 $\theta^T x \geq 1$ (大于1时 cost_1 才为0); 对任意的 $y^{(i)} = 0$, 都要满足 $\theta^T x \leq -1$ (小于-1时 cost_0 才为0)。而这样，就相当于是同时给优化问题(1)增加了这个约束条件:

s.t.

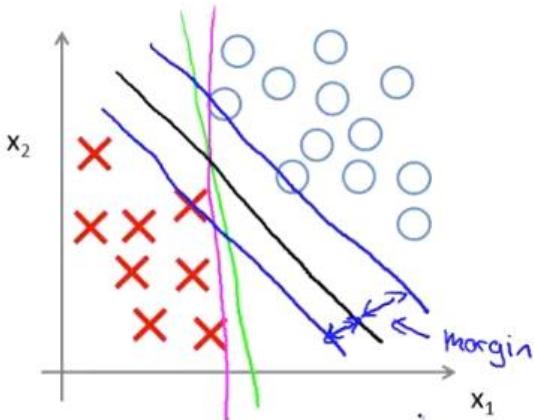
$$\theta^T x^{(i)} \geq 1, y = 1$$

$$\theta^T x^{(i)} \leq -1, y = 0$$

这个约束条件的作用，同样也就是经过我们修改的SVM的Loss的作用，也是最大间隔分类器的核心所在。下面我们就一步一步的揭示其原理和意义。

以二维特征为例，求解这个优化问题(1)，会得到这样一个有趣的分类结果(假设数据线性可分)：得到的分类边界一定是距离正负类样本点最短距离最大的边界(下图黑线)。

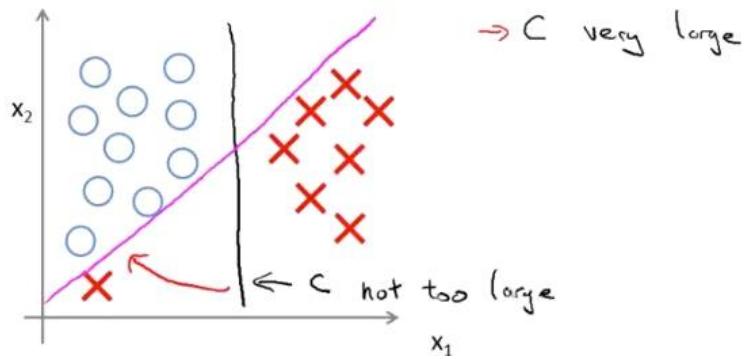
SVM Decision Boundary: Linearly separable case



Large margin classifier

当然，这只是一个直观解释，因为C值非常大时，SVM总是会学习到一个与所有的正负样本间距都最大的决策边界，那么如果出现如下图的异常点，为了跟它距离最大，反而会产生过拟合(如粉线的形式，而不是黑线)。或者从另一个角度理解，如果C非常大，那么就相当于LR中 $\frac{1}{\lambda}$ 非常大，即 λ 非常小，那么就相当于Loss后边的正则项就失效了。因此在实际应用中C会设置成一个不是很大的值。

Large margin classifier in presence of outliers

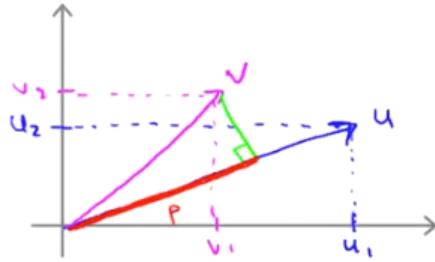


那么，为什么我们会得到上面的结果呢？即，为什么通过求解上边这个最优化问题，就会得到一个与正负样本最小距离尽可能大的决策边界呢？下面我们就对其进行证明。首先要对向量内积的概念做一下回顾：

对于两个列向量 $u = (\blacksquare(u_1 @ u_2))$, $v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$ 线性代数的知识告诉我们，他们的内积是

$$u * v = u^T v = u_1 v_1 + u_2 v_2$$

但是在几何意义上，向量 u, v 的内积是这么定义的：



将v在向量u的方向上进行投影，如上图所示，得到投影长度为p，则u,v的内积为。

$$u \cdot v = u^T v = p * \|u\| \quad (2)$$

其中u的范数 $\|u\|$ ，也就是长度为

$$\|u\| = \sqrt{u_1^2 + u_2^2}$$

注意，这里将v在向量u的方向上投影或者将u在向量v的方向上进行投影再求内积，得到的结果是没有区别的，最终得到的是同一个实数，有 $u^T v = v^T u$ 。如果uv夹角大于90度，则v的投影p的方向就会与u相反，此时p就是负值，此时内积也是负值。同时，如果两个向量内积为0，说明两个向量相互垂直。

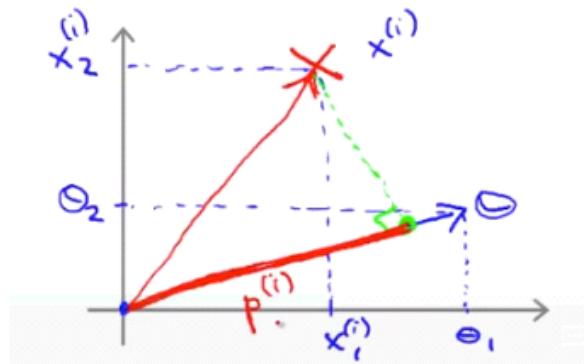
那么回到最大间隔分类器的Loss最小化问题上来，为了简化推导过程，需要假设截 $\theta_0 = 0$ ，此时 θ 就变成了仅由 θ_1, θ_2 构成的列向量，且此时仍然是二维特征。

则可以对式(1)做以下变换：

$$\begin{aligned} & \min_{\theta} \left(\frac{1}{2} \sum_{j=1}^n \theta_j^2 \right) \\ &= \min_{\theta} \frac{1}{2} (\theta_1^2 + \theta_2^2) \\ &= \min_{\theta} \frac{1}{2} (\sqrt{\theta_1^2 + \theta_2^2})^2 \\ &= \min_{\theta} \frac{1}{2} (\|\theta\|)^2 \quad (3) \end{aligned}$$

可以看到，最终结果里 $\|\theta\|$ 就是参数向量的长度，也就是说，在C值很大的时候，求解SVM做的唯一一件事情，就是最小化参数向量 θ 的长度(范数)(的平方)。

接下来，我们看，在公式(1)的两个约束条件中， $\theta^T x^{(i)}$ 的实际意义很明确，参数向和样本的乘积用以预测样本所属类别，这和神经网络中的正向传播完全相同。



但是如果我们从向量内积的几何意义来看 $\theta^T x^{(i)}$, 如上图, 那么它代表着第*i*个样本向量 $x^{(i)}$ 在参数向量 θ 上的投影 $p^{(i)}$ 与 θ 的范数(长度)之间的乘积, 即

$$\theta^T x^{(i)} = p^{(i)} * \|\theta\|$$

那么(1)的约束条件就可以被更换为

$$p^{(i)} * \|\theta\| \geq 1, y = 1 \quad (4)$$

$$p^{(i)} * \|\theta\| \leq -1, y = 0 \quad (5)$$

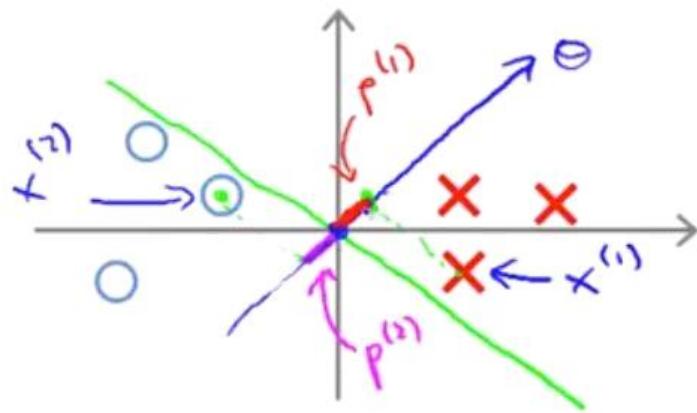
我们知道, 参数向量 θ 与决策边界是正交的, 这是因为, 由LR的定义的可知(SVM与LR在求预测值时形式一致), 对于一个二分类问题, 如果我们认为输出概率>0.5为正类1, 反之为反类0, 则当预测结果 $y=1$ 时, 代表Sigmoid函数的输出大于0.5, 由Sigmoid的形状可知, 此时

$$\theta^T x^{(i)} \geq 0$$

反过来 $y=0$, 则有

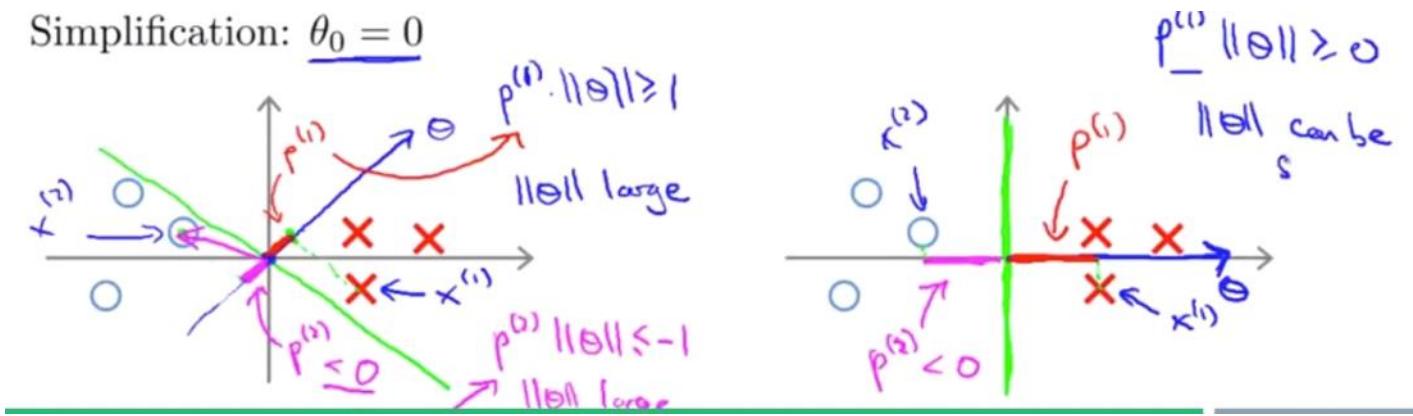
$$\theta^T x^{(i)} < 0$$

那么, 如果我们根据 θ 值在样本平面上画一条直线, 落在这条直线上方的都认为是正类, 落在这条直线下方的都认为是负类, 那么这条直线就是决策边界。既然落在直线上方的都是正类(即 $y=1$), 那对任意样本 $x^{(i)}$ 在这条直线上方就有 $\theta^T x^{(i)} \geq 0$, 同理这条直线下方就有 $\theta^T x^{(i)} < 0$ 。那么不考虑边界条件, 在这条决策边界直线上, 就一定有 $\theta^T x^{(i)} = 0$, 或者说决策边界的直线方程就是 $\theta^T x^{(i)} = 0$, 由内积为0的定义可知, 既然 θ 与决策边界上的任意一个样本向量的内积都为0, 就说明 θ 与决策边界上的任意向量都正交, 也就是说 θ 与决策边界是正交的。



既然 θ 正交于决策边界，那么样本 $x^{(i)}$ 在 θ 上的投影也就正交于决策边界，如上图所示，这时一个有趣的事事实现了：样本 $x^{(i)}$ 在 θ 上的投影长度 $p^{(i)}$ ，也恰好就是 $x^{(i)}$ 到决策边界的距离！也就是说，SVM的margin，就是样本点到参数向量 θ 上的投影长度。

那么上边说的一大堆是怎么决定SVM会选择一个最大间隔分类边界的呢？我们看优化过程中可能会出现的以下左右两个决策边界：



左边的情况，决策边界严重歪斜，距离一些样本非常近，这会导致许多样本 x 到 θ 的投影长度 p 非常短。如果 x 是正类，在分类边界上方，那么 p 就会是一个很小的正值；若 x 是负类则 p 是一个小负值。则为了满足约束(4)大于1的要求以及(5)小于-1，就需要 $\|\theta\|$ 的值较大，即需要 θ 的有很大的长度，而由式(3)，我们的优化目标是最小化 θ 的长度，因此直观理解，左边的这个决策边界不是一个好的选择。

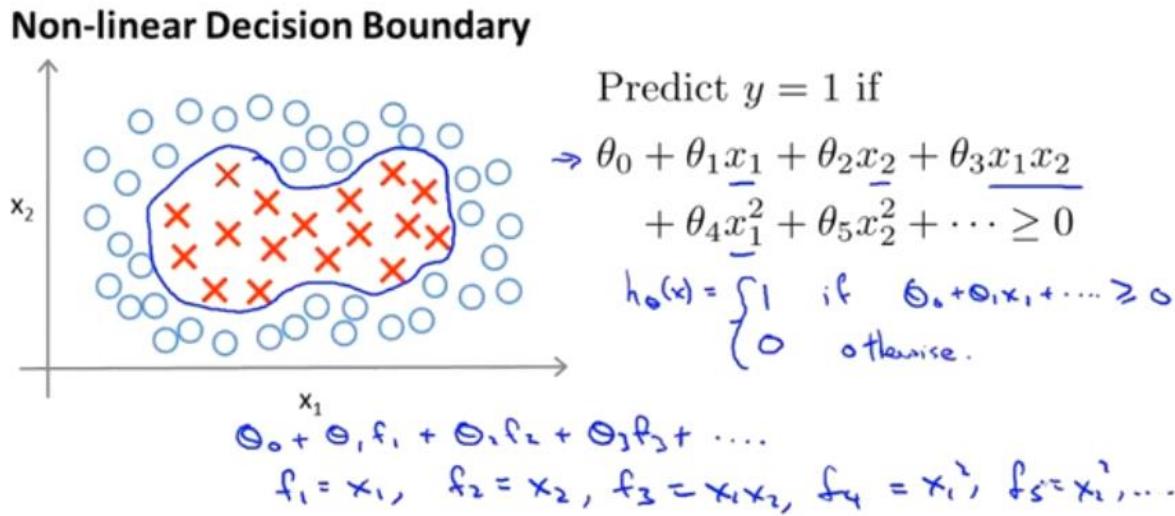
而右边的情况，决策边界距离各样本都较远，即 p 值相对较高，此时较小的 $\|\theta\|$ 的值就可以满足约束条件(4)(5)，这样与优化(3)的方向是一致的。

因此直观上来讲，SVM在优化过程中会更倾向于形成右侧的决策边界，实际上梯度的方向就是使决策边界倾向于与样本点距离最大的方向。我们反过来理解也可以，因为在求解(3)时，会尽量选择较小的 $\|\theta\|$ ，那么为了满足约束(4)(5)，就会要使 $p^{(i)}$ 倾向于更大的值，才能满足大于1或小于-1的约束条件，而我们已经说了在 θ 上的投影长度 $p^{(i)}$ 就是样本 $x^{(i)}$ 到决策边界的距离。

上述推导我们是假定截距 θ_0 为 0，实际上 θ_0 不为 0 时，过程和结果是一样的，最大间距分类器同样会朝着使决策边界与样本点距离最大的方向去优化。

(4) 核函数 1

2022年2月1日 19:51



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

如上图所示，在一个训练集合。如何进行正负样本的非线性边界的判定。

一个办法是构造一个复杂多项式特征的集合，也就是像这样的特征变量的集合于是我们假设函数 $h_\theta(x)$ 。

当多项式大于等于0时，假设函数 $h_\theta(x) = 1$

当多项式小于0时，假设函数 $h_\theta(x) = 0$

又我们令多项式为

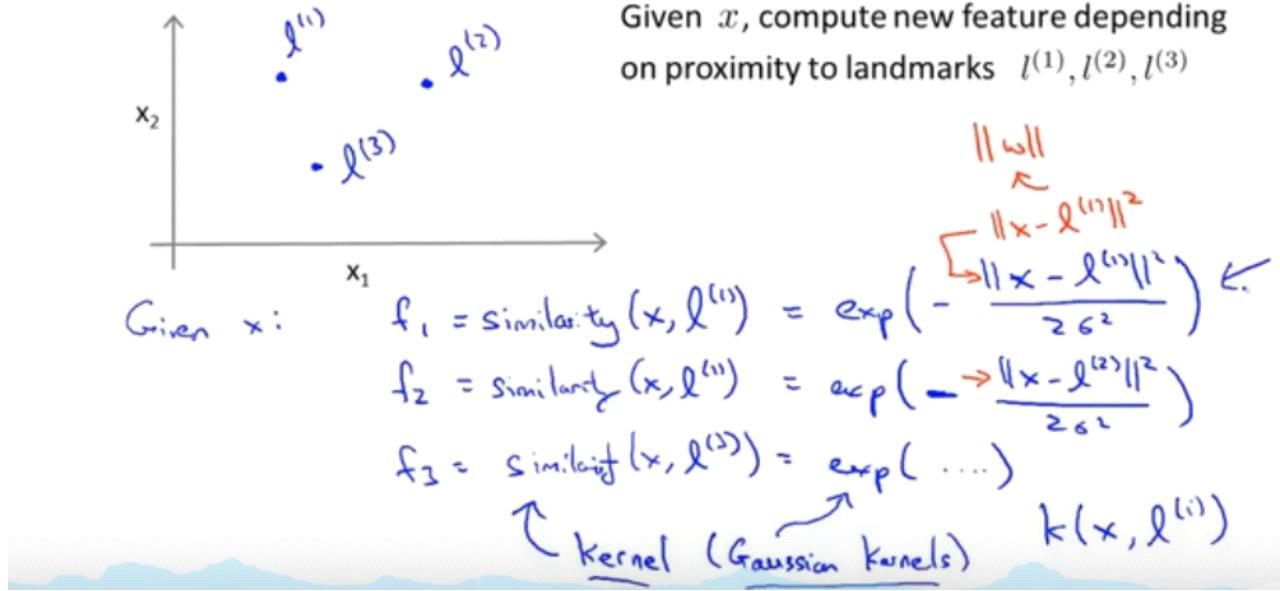
$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \dots$$

令

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

这些是一种得到更多特征的方式，但问题是我们可以有很多不同的特征选择或都可能会存在比这些高阶多项式更好的特征。因为我们并不清楚地知道这些高阶项是不是我们真正需要的。

Kernel



在这里我们有 x_1 和 x_2 两个特征值，并且从这两个特征值中获得三个取值，记为 $l^{(1)}$ (标记1), $l^{(2)}$ (标记2), $l^{(3)}$ (标记3)。

我们记 f_1 为一种相似的度量，即度量训练样本 x 与第一个标记的相似度。记公式为

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

其中函数 $\text{similarity}(x, l^{(1)})$ 叫做高斯核函数(相似函数)。

Kernels and Similarity

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

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

$$f_1 \approx \exp\left(-\frac{0}{2\sigma^2}\right) \approx 1$$

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

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

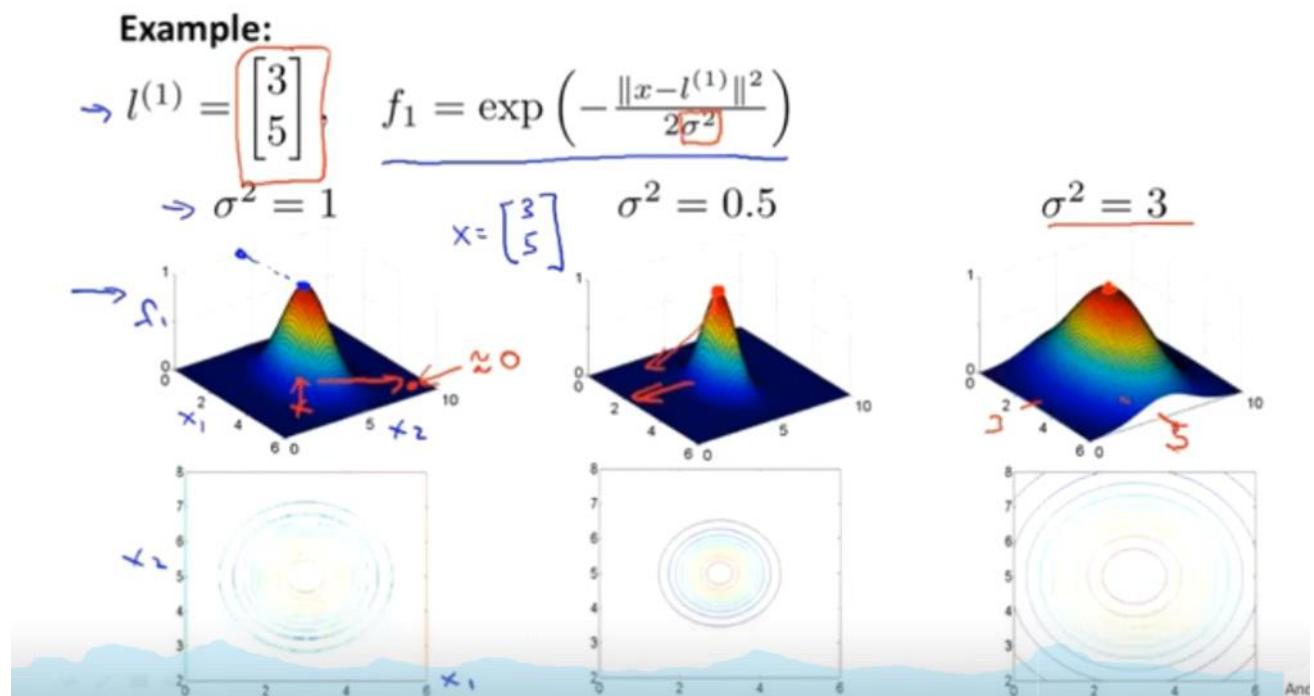
$$f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0.$$

我们有核函数 $\text{similarity}(x, l^{(1)})$ 当 $x \approx l^{(1)}$

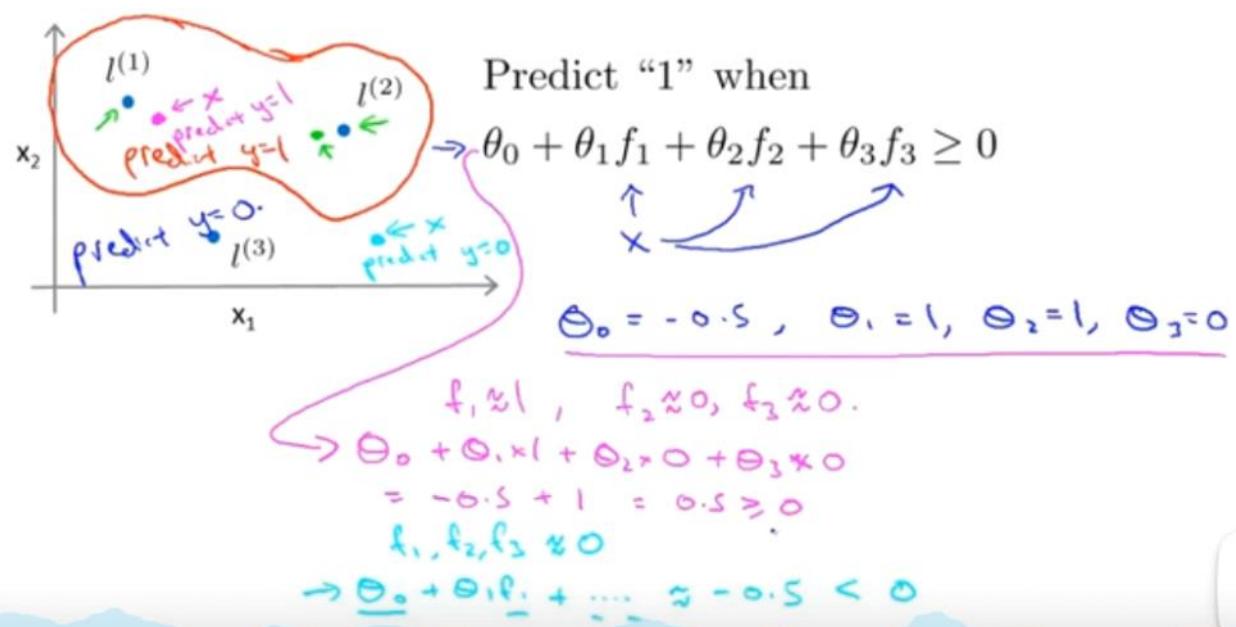
$$f_1 \approx e^{\left(-\frac{0^2}{2\sigma^2}\right)} \approx 1$$

若当 x 远大于 $l^{(1)}$ 时

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



当参数 θ^2 变化时，图像的变化。

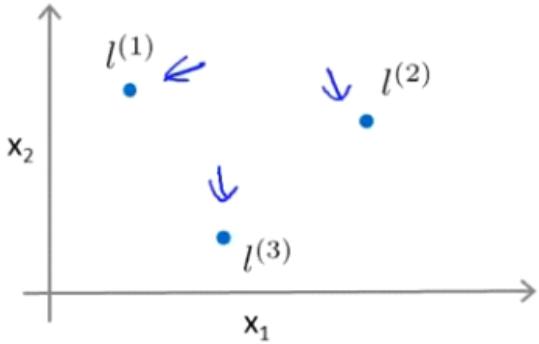


我们有特征 x_1 和 x_2 再从样本 x 中取三个标记点，得到函数 f_1, f_2, f_3 ，假定我们通过学习算法得到参数 $\theta_0, \theta_1 \dots$ 的值，我们就可以通过这些参数若所得结果大于0.5则为1，否则为0。

(5) 核函数 2

2022年2月2日 21:09

Choosing the landmarks

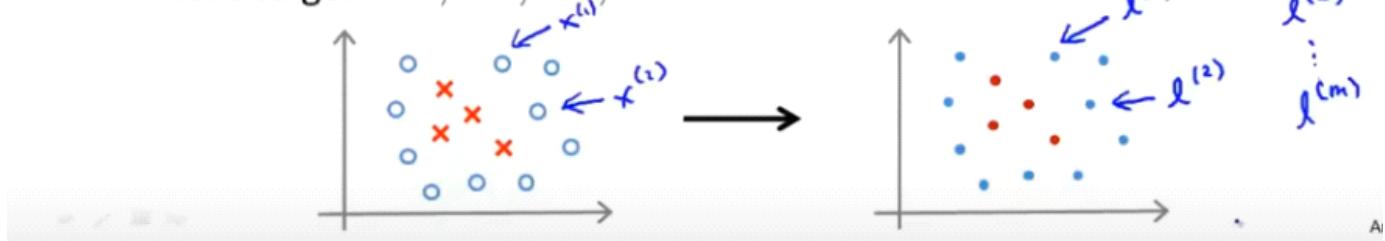


Given x :

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

Predict $y = 1$ if $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$

Where to get $l^{(1)}, l^{(2)}, l^{(3)}, \dots$?



我们如何挑选合适的 $l^{(1)}, l^{(2)}, l^{(3)}, \dots$?

我们拥有的样本集 x , 直接将训练样本作为标记点, 若我们拥有训练样本 $x^{(1)}$, 那么我们将在这个样本完全相同的位置上选作我的第一个标记点。如果我有另一个训练样本 $x^{(2)}$, 那么第二个标记点就在与第二个样本点一致的位置上。

利用这个方法, 最终能得到 m 个标记 $l^{(1)}, l^{(2)}, \dots, l^{(m)}$ 一直到 $l^{(m)}$, 即每一个标记点的位置都与每一个样本点的位置相对应。这表明特征函数基本上是在描述每一个样本距离, 样本集中其他样本的距离。

SVM with Kernels

- \rightarrow Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$,
- \rightarrow choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)}$.

Given example x :

$$\rightarrow f_1 = \text{similarity}(x, l^{(1)})$$

$$\rightarrow f_2 = \text{similarity}(x, l^{(2)})$$

$$\dots$$

$$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \quad f_0 = 1$$

SVM with Kernels

- Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$,
- choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)}$.

Given example x :

$$\begin{aligned} &\rightarrow f_1 = \text{similarity}(x, l^{(1)}) \\ &\rightarrow f_2 = \text{similarity}(x, l^{(2)}) \\ &\dots \end{aligned}$$

$$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \quad f_0 = 1$$

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

$$\begin{aligned} x^{(i)} \rightarrow & f_1^{(i)} = \text{sim}(x^{(i)}, l^{(1)}) \\ & f_2^{(i)} = \text{sim}(x^{(i)}, l^{(2)}) \\ & \vdots \\ & f_i^{(i)} = \text{sim}(x^{(i)}, l^{(i)}) = \exp(-\frac{\alpha}{2\pi}) = 1 \\ & f_m^{(i)} = \text{sim}(x^{(i)}, l^{(m)}) \end{aligned}$$

$$\begin{aligned} x^{(i)} \in \mathbb{R}^{n+1} \quad (\text{or } \mathbb{P}^n) \\ f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} \\ f_0^{(i)} = 1 \end{aligned}$$

Andrew Ng

由公式 $f_1 = \text{similarity}(x, l^{(i)})$, 我们可以组成特征矩阵。

$$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \dots \\ f_m \end{bmatrix}$$

若我们有样本 $x^{(i)}$, 则我们有

$$f_1^{(i)} = \text{sim}(x^{(i)}, l^{(1)})$$

$$f_2^{(i)} = \text{sim}(x^{(i)}, l^{(2)})$$

$$f_3^{(i)} = \text{sim}(x^{(i)}, l^{(3)})$$

我们可以得到特征向量

$$f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ f_2^{(i)} \\ \dots \\ f_m^{(i)} \end{bmatrix}$$

SVM with Kernels

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

\rightarrow Predict "y=1" if $\underline{\theta^T f} \geq 0$ $\theta_0 f_0 + \theta_1 f_1 + \dots + \theta_m f_m$

Training:

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

$\rightarrow \theta_0$

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

$\rightarrow \theta^T M \theta \quad \| \theta \|^2 \quad M = 10,000$

Andrew

我们假设 x , 若 $\theta^T f \geq 0$ 则预测 $y=1$ 。我们训练样本得到最小化参数 θ 而我们有公式

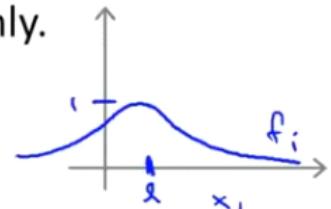
$$\min_{\theta} C \left[\sum_{i=1}^m y^{(i)} (\text{cost}_1(\theta^T f^{(i)})) + (1 - y^{(i)}) (\text{cost}_0(\theta^T f^{(i)})) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

SVM parameters:

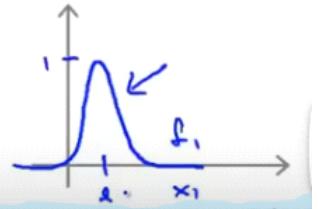
C ($= \frac{1}{\lambda}$). \rightarrow Large C : Lower bias, high variance. (small λ)
 \rightarrow Small C : Higher bias, low variance. (large λ)

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

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



Small σ^2 : Features f_i vary less smoothly.
Lower bias, higher variance.



SVM 参数

当我们有较大的数C时，具有低偏差，高方差的特征(过拟合)。

当我们有较小的数C时，具有高偏差，低方差的特征(欠拟合)。

我们有参数 θ^2

在有较大参数 θ^2 时，特征 f_i 表现为比较平滑(欠拟合)。

在有较小参数 θ^2 时，特征 f_i 表现为具有较少的平滑(过拟合)。

(6) 使用SVM

2022年2月4日 9:36

Use SVM software package (e.g. liblinear, libsvm, ...) to solve for parameters θ .
↑

Need to specify:

→ Choice of parameter C.

Choice of kernel (similarity function):

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

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

$$\Theta_0 + \Theta_1 x_1 + \dots + \Theta_n x_n \geq 0 \quad x \in \mathbb{R}^{n+1}$$

→ 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 $\underline{\sigma^2}$

$$x \in \mathbb{R}^n, n \text{ small}$$

and/or m large



Andrew N

我们一般使用软件库去调用函数去解决参数 θ

另外我们在使用SVM时需要去

一、挑选合适的参数C

二、挑选合适的核函数

核函数有两种

(1) 无核函数(线性核函数)

如果 $\theta^T x \geq 0$, 则预测 $y=1$ (当特征n很大时, 而样本m很小时, 这是个合理的选择)

(2) 高斯核函数

当 $l^{(i)} = x^{(i)}$ 时, $f_i = e^{-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}}$

我们也要挑选合适的参数 σ^2

若参数 σ^2 过大, 会导致较大的偏差和较低的方差的分类器。

若参数 σ^2 过小, 会导致较小的偏差和较高的方差的分类器。

那什么时候选择高斯函数呢?

若n很小, 且m值很大, 用核函数来拟合相当复杂非线性决策边界, 高斯核函数是个不错的选择。

Kernel (similarity) functions:

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

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

return

$x \rightarrow$
 f_1
 f_2
 \vdots
 f_m

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

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

定义核函数如上图所示。

注意：在使用高斯核函数之前需要先执行特征缩放。

即如上图所示，如第一个特征为房屋大小，第二个特征为卧室数量，第一特征相差所得到的结果会远大于第二特征相差，故需要做好特征缩放。

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: $k(x, l) = (x^T l + \text{constant})^{\text{degree}}$

- More esoteric: String kernel, chi-square kernel, histogram intersection kernel, $\text{sim}(x, l)$

核函数的其它选择

注意：不是所有相似函数 $\text{similarity}(x, l)$ 都是有效核函数。

需要满足被称为“默塞尔定理”技术条件去确保SVM包最优化正确地运行，确保不会产生偏差。即确保SVM软件包能够用大类的优化方法并从而迅速得到参数 θ 。

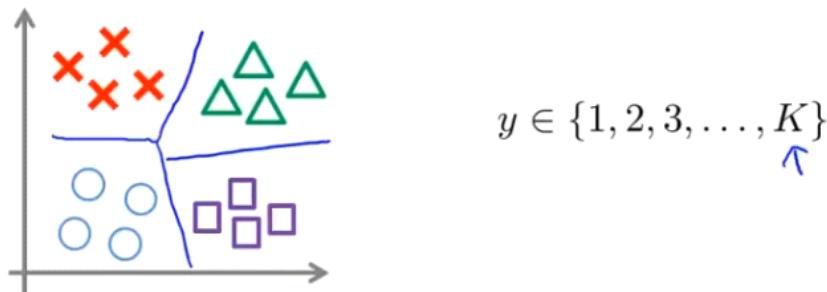
大多数现成的可用核函数有：

多项式核函数：

形式如下：

$$k(x, l) = (x^T l + contact)^{degree}$$

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

大量的SVM包已经内置了多元分类函数。

另外，使用一个one-vs-all的方法，所要做的是训练KSVM，如果你有 K 个类别，用以将每个类别从其他的类别中区分开来这个会给你 K 参数的向量，这个会给你 $\theta^{(1)}$ 这会尝试从所有其他类别中，识别出 $y=1$ 的类别，之后你得到第二个参数，向量 $\theta^{(2)}$,然后得到 $y=2$ 作为正类别，其他的作为负类别时，得到的以此类推参数向量 $\theta^{(k)}$ 是用以识别最后一个类别参数向量，最后，这就于我们在逻辑回归中用到的一对多的方法一样，在逻辑回归中我们只是用最大的 $(\theta^{(i)})^T x$ 来预测类别 i

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")

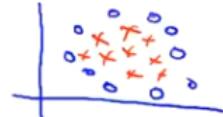
→ If n is small, m is intermediate: ($n = 1-1000$, $m = 10-10,000$) ←

→ 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.



逻辑回归 VS SVM

n 为特征数量 m 为训练样本数

一、若 n 非常大时, m 较小时, 即 $n \geq m$, 例 $n=10000, m=10-1000$

使用逻辑回归或 SVM(线性核函数)

二、若 n 非常小时, m 适中, 即 $(n=1-1000, m=10-10,000)$

使用带有高斯核函数的 SVM。

三、若 n 非常小时, m 较大时, $(n=1-1000, m=50,000+)$

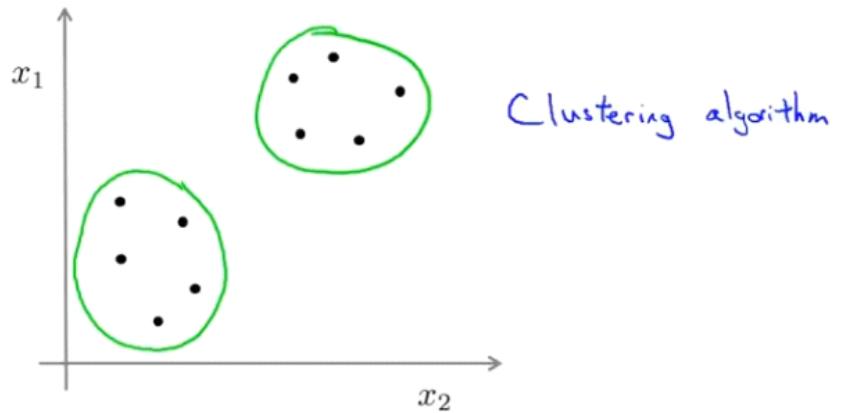
创建/添加更多的特征, 然后使用逻辑回归或不带核函数的 SVM,
使用带高斯核函数的 SVM 会比较慢。

神经网络可能对于上述场景都可能运行的非常好, 但这可能会使训练更加缓慢。

(1) 无监督学习

2022年2月4日 15:09

Unsupervised learning



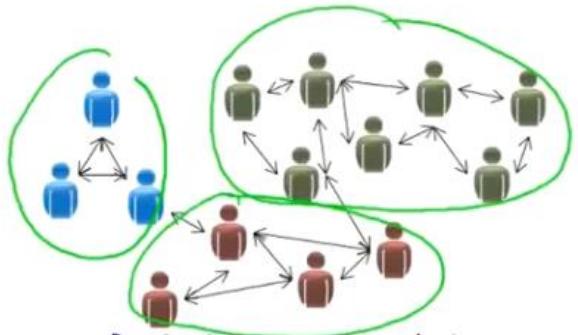
Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(m)}\}$ ← .

在监督学习中，数据并不带有任何的标签，得到的数据如上图所示，图上有一系列的点，但是它们并没有任何标签，因此我们的训练集可以写成 $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ 。我们要将这系列的无标签的数据输入到算法中，然后我们要让算法找到一些隐含在数据中的结构，通过图中这些数据，我们能通过算法找到的一个结构就是这个数据集中的点，可以分成两组分开的点集。这种能够找出所圈出的这些簇的算法被称为“聚集算法”（这是无监督算法中的一种算法）。

Applications of clustering



→ Market segmentation



→ Social network analysis



→ Organize computing clusters



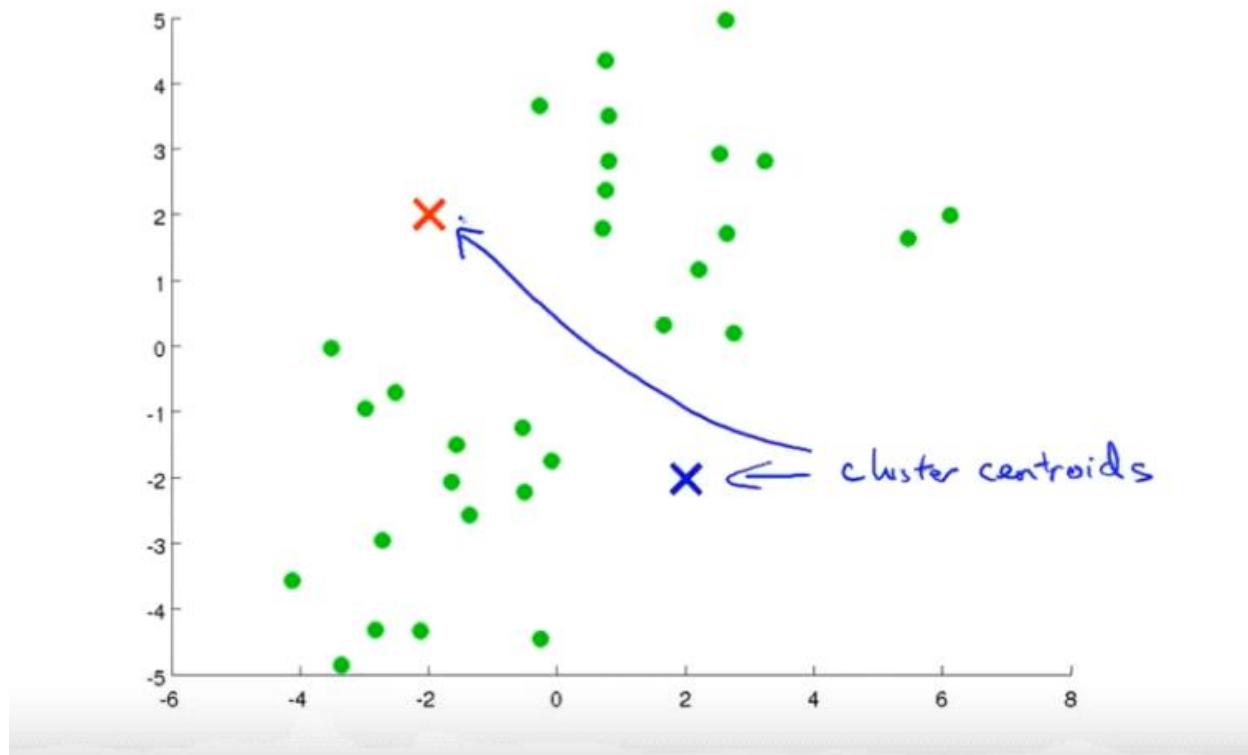
→ Astronomical data analysis

聚集算法的应用：

- 一、市场的划分
- 二、社会网络的分析
- 三、组成计算聚集
- 四、天文数据分析

(2) K-Means(k-均值)算法

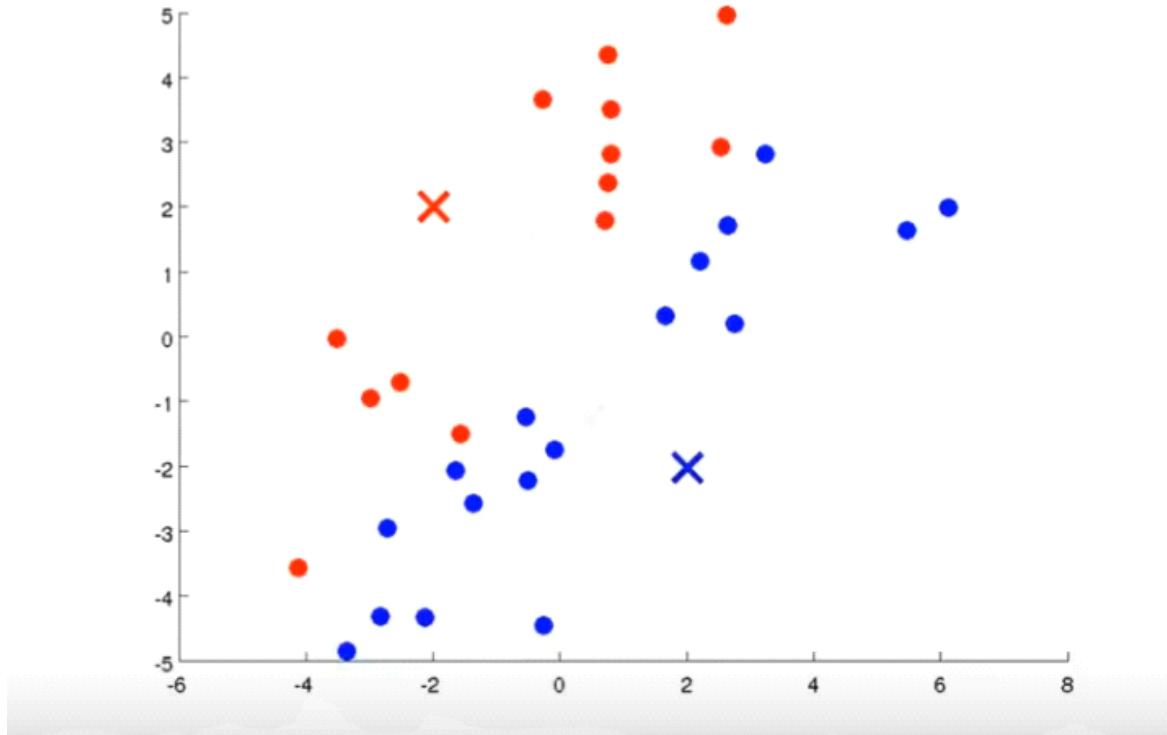
2022年2月4日 17:08



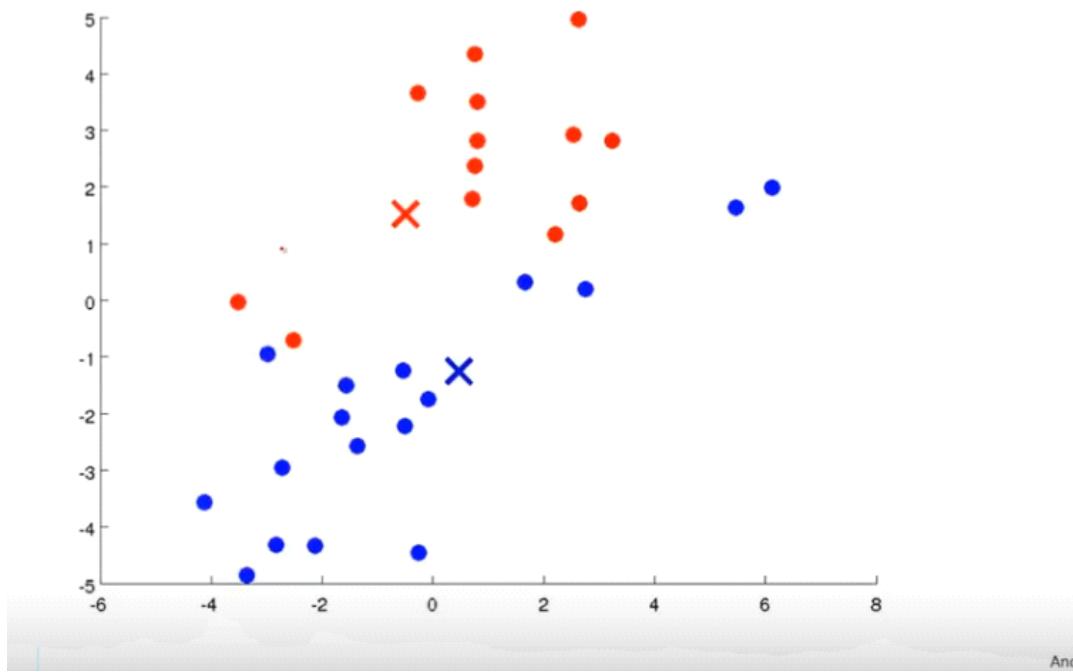
K-均值算法：

假设我有一个无标签的数据集，如上图所示，并且想要将其分成两个簇，现在执行K-均值算法，具体操作如下：

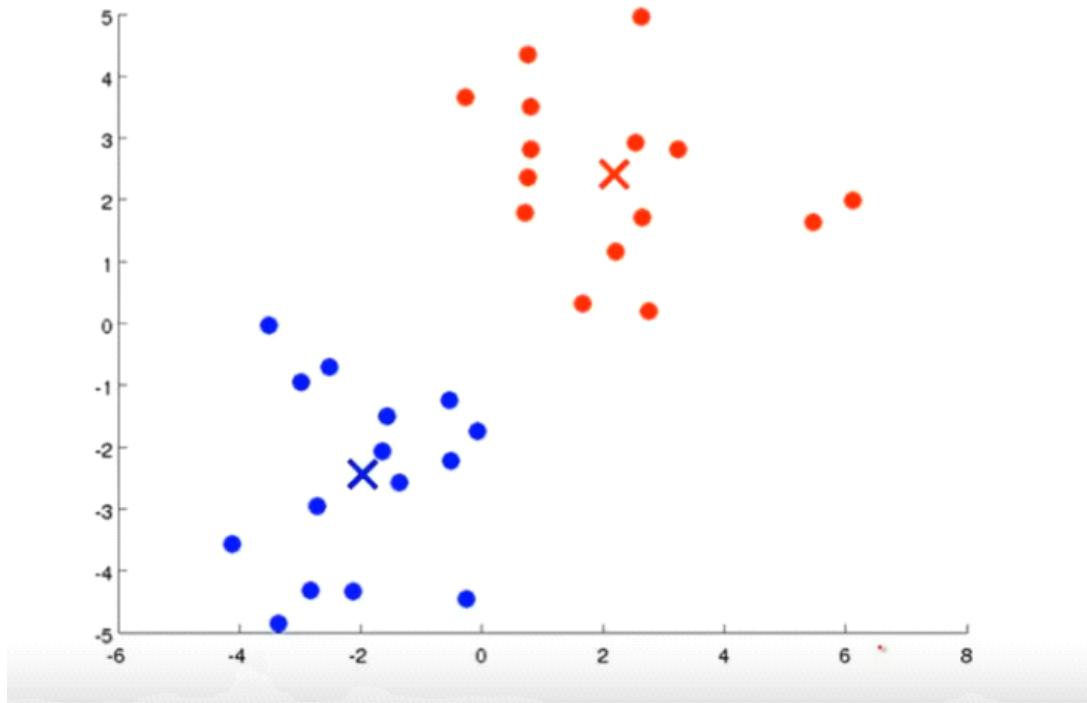
随机生成两个点(图中两个叉处)，这两个点叫做聚集中心，选取这两个点的原因是，因为想要将这些数据聚成两类，K均值算法是一个迭代算法，它会做两件事。第一个是簇分配，第二个是移动聚集中心。K均值中每一次内循环是要进行簇分配，即遍历每一个样本，然后根据每一个点是与红色聚集中心更近还是蓝色聚集中心更近，来将每个数据点分配给两个聚集中心之一。



具体来说，就是遍历数据集，然后将每个点染成红色或蓝色，这取决于某个点是更接近红色聚集中心还是蓝色聚集中心。K-均值算法的第二步就是移动聚集中心，即将两个聚集中心将其移动到同色的点的均值处。要做的就是找到所有红色的点然后找到这些点的均值(即所有红色点的平均位置)，然后移动聚集中心，蓝色的点亦如此。



再次重复上面的步骤，计算均值，移动聚集中心。



再次移动。

K-means algorithm

Input:

- K (number of clusters) ←
- Training set $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ ←

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

K-均值算法

输入：参数K(簇的数量)

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

其中 $x^{(i)}$ 为n维实数向量， $x_0 \neq 1$

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

Move
centroid

for $i = 1$ to m

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

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

for $k = 1$ to K

$\rightarrow \mu_k$:= average (mean) of points assigned to cluster k
 $x^{(1)}, x^{(5)}, x^{(6)}, x^{(10)}$ $\rightarrow c^{(1)}=2, c^{(5)}=2, c^{(6)}=2, c^{(10)}=2$

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

K-均值算法：

一、随机初始化 K 个聚类中心, $u_1, u_2, \dots, u_k \in R^n$

二、重复进行簇聚类中心的计算和进行簇聚类中心的移动。利用公式

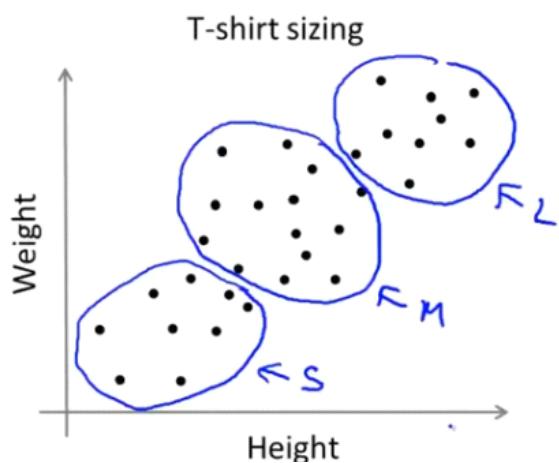
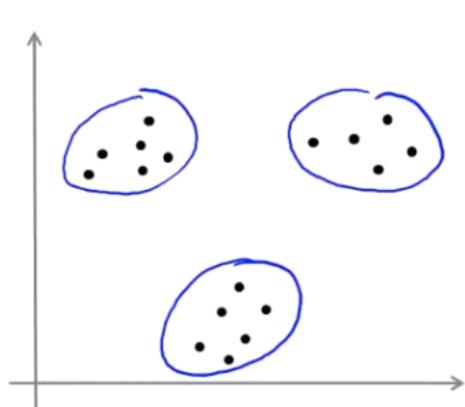
$$c^{(i)} = \min_k \|x^{(i)} - u_k\|^2$$

进行探测点 $x^{(i)}$ 与哪一簇聚类中心距离最短，即点 $x^{(i)}$ 属于哪一簇。

再通过平均公式得到新的聚类中心。

K-means for non-separated clusters

S, M, L



对于无分离簇的K-均值算法。

K-均值算法除了可以对左图进行簇分离，也可对右图进行簇分离。

(3) 优化目标

2022年2月4日 19:07

K-means optimization objective

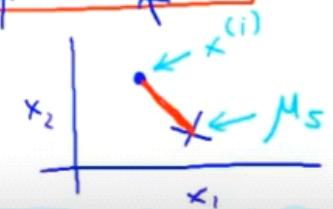
- $c^{(i)}$ = index of cluster ($1, 2, \dots, K$) to which example $x^{(i)}$ is currently assigned
- μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$) K $k \in \{1, 2, \dots, K\}$
- $\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned $x^{(i)} \rightarrow S$ $c^{(i)}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

Optimization objective:

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

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

Distortion



K-均值算法的优化目标

$c^{(i)}$ 表示当前样本 $x^{(i)}$ 所指定的簇的索引。

μ_k 表示簇聚集中心 k

$\mu_c^{(i)}$

表示样本 $x^{(i)}$ 所指定簇索引的簇的位置。

优化目标公式

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

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

其中 $\square \square (\square)$ 代表与 $\square (\square)$ 最近的聚类中心点。我们的优化目标便是找出使得代价函数最小的 $\square (1), \square (2), \dots, \square (\square)$ 和 $\square 1, \square 2, \dots, \square \square$ 。

这个过程与公式

$$c^{(i)} = \min_k \|x^{(i)} - u_k\|^2$$

具有相同的效果。

第一个循环是用于减小 $J(C)$ 引起的代价，表与聚集中心相近的点不断变为与就近聚集中心相同的标签，这个迭代表示聚集中心的点都有相同的标签，在不断减少 $C(i)$ 引起的代价。

而第二个循环则是用于减小 $J(C)$ 引起的代价。不断移动聚集中心，不断减少代价。迭代的过程一定会是每一次迭代都在减小代价函数。

K-means algorithm

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

Repeat { [Cluster assignment step]
 Minimize $J(\dots)$ wrt $(c^{(1)}, c^{(2)}, \dots, c^{(n)})$ ←
 (holding μ_1, \dots, μ_K fixed) }

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

} minimize $J(\dots)$ wrt (μ_1, \dots, μ_K)

move
centroid

K-均值算法

第一个内循环，最小化 J 函数的参数 $C^{(i)}$

第二个内循环，最小化 J 函数的参数 u_k

(4) 随机初始化

2022年2月5日 10:40

随机初始化的目的是使K-均值算法避开局部最优

K-means algorithm

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

```
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  $k$   
}
```

K-均值算法如上图所示。

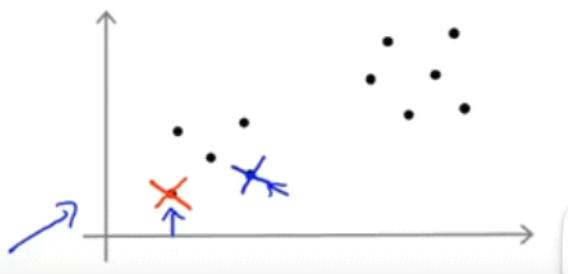
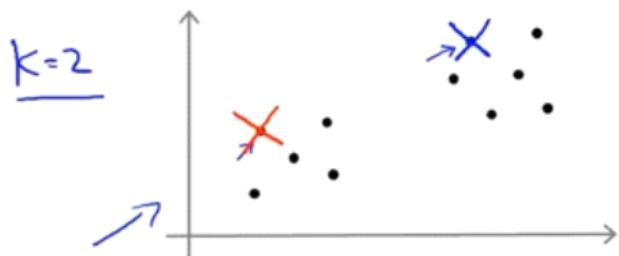
Random initialization

Should have $K < m$

Randomly pick K training examples.

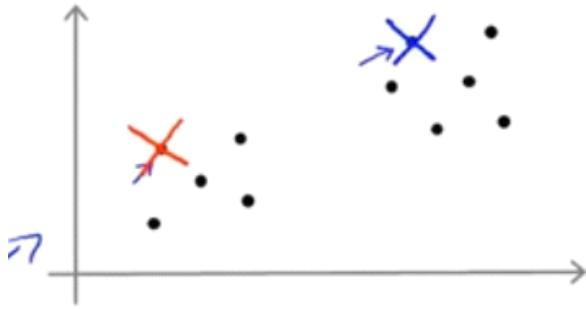
Set μ_1, \dots, μ_K equal to these K examples.

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

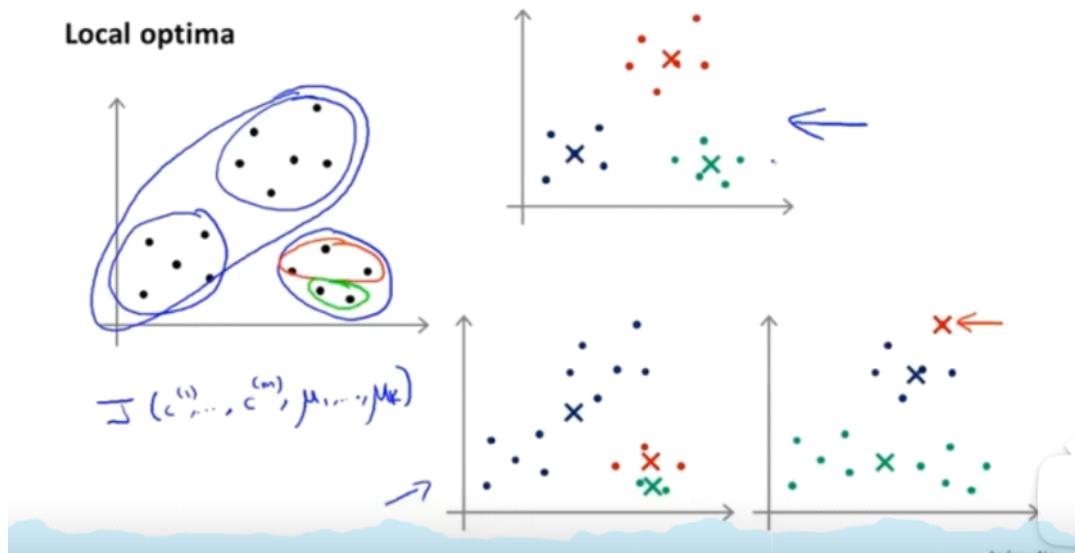
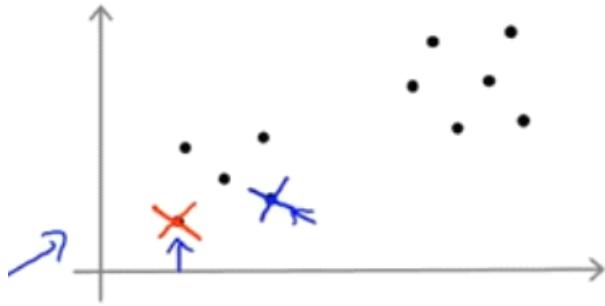


随机初始化的过程，

首先 $k < m$ ，随机挑选 K 个训练样本，设置聚类中心 $\mu_1, \mu_2, \mu_3, \dots, \mu_k$ 为这些样本的值。但因为是随机初始化，故有些情况会分成比较均衡的簇。



而有些情况会分成比较靠拢的簇。



如上图所示，上图靠右边的两张图的情况，是代价函数J的局部最优情况，非全局最优情况，应对这种情况，我们需要多次随机初始化，求得全局最优的情况。

Random initialization

For i = 1 to 100 { 50 - 1000

 → Randomly initialize K-means.
 Run K-means. Get $c^{(1)}, \dots, c^{(m)}$, μ_1, \dots, μ_K .
 Compute cost function (distortion)
 → $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$
}

Pick clustering that gave lowest cost $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

$k=2-10$

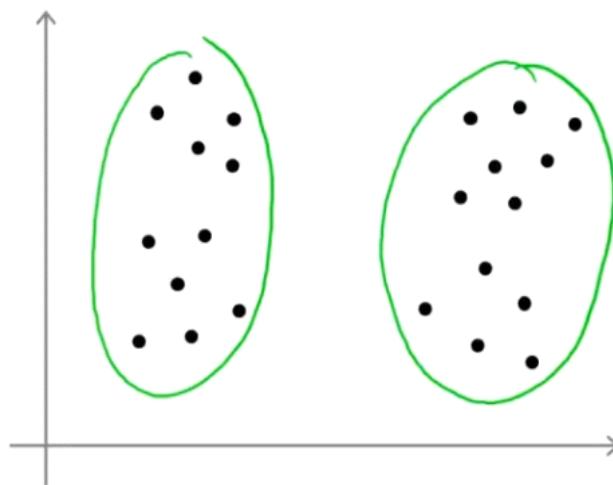
上图为随机初始化算法，其经典的运行次数为20-1000次，这个算法在2-10个簇较为有效。

(5) 选取聚类的数量

2022年2月5日 11:21

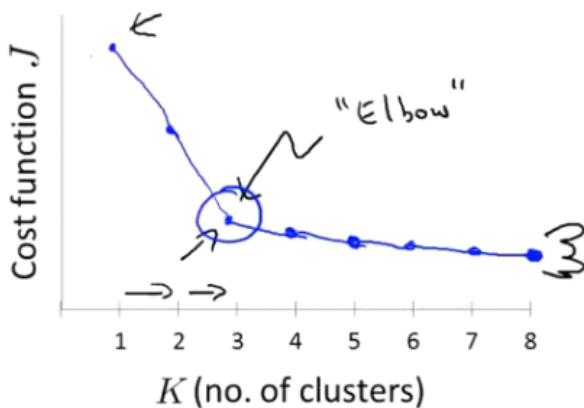
解决选择聚类数量问题仍然是通过观察可视化的图或者通过观察聚类算法的输出等。

What is the right value of K?



上图中我们即可选取四簇也可以选取二簇，这个比较不确定，所以用自动化算法来选取簇是不合理的。

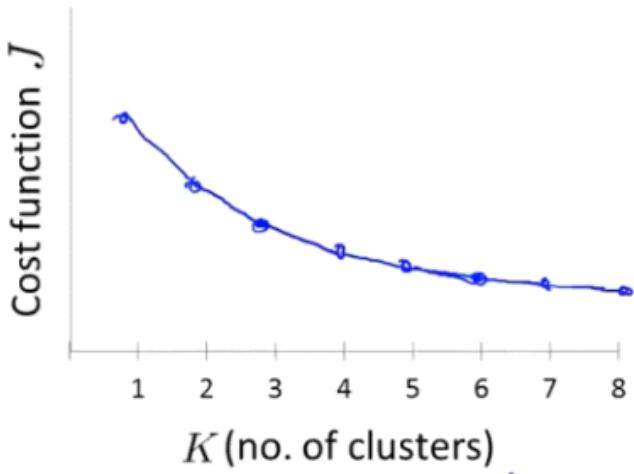
Elbow method:



对于选取合适的簇的方法，有"肘部方法"。

即选取 $K=1$ ，得代价函数J，依次类推，然后观察整个线条走势。

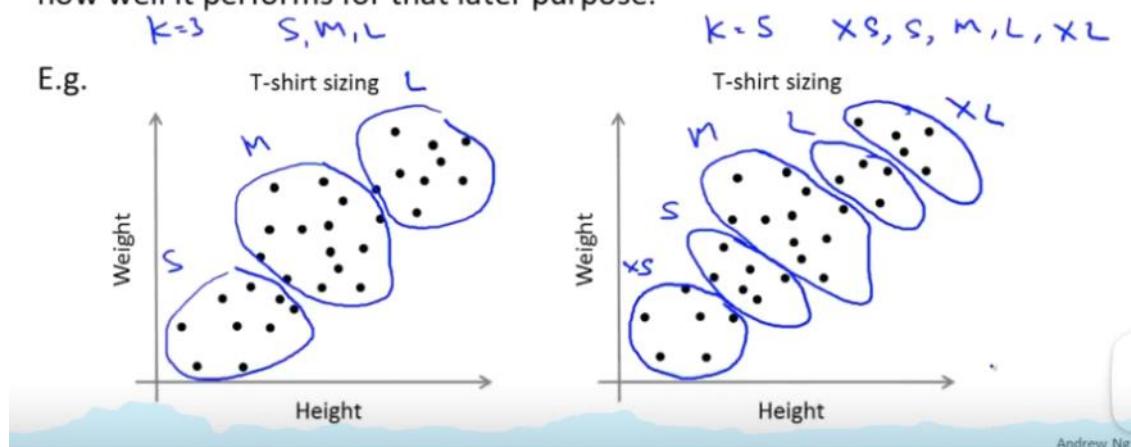
所上图所示，选择 $K=3$ 是比较合理的，因为有 $k=2$ 到 $k=3$ 有一个比较快速的下降趋势，从 $k=3$ 到 $k=4$ 有一个比较平缓的趋势，所以选择 $k=3$ 是比较合理的。



但有时图像会呈现出上图的走势，无法选定簇的数量。

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.



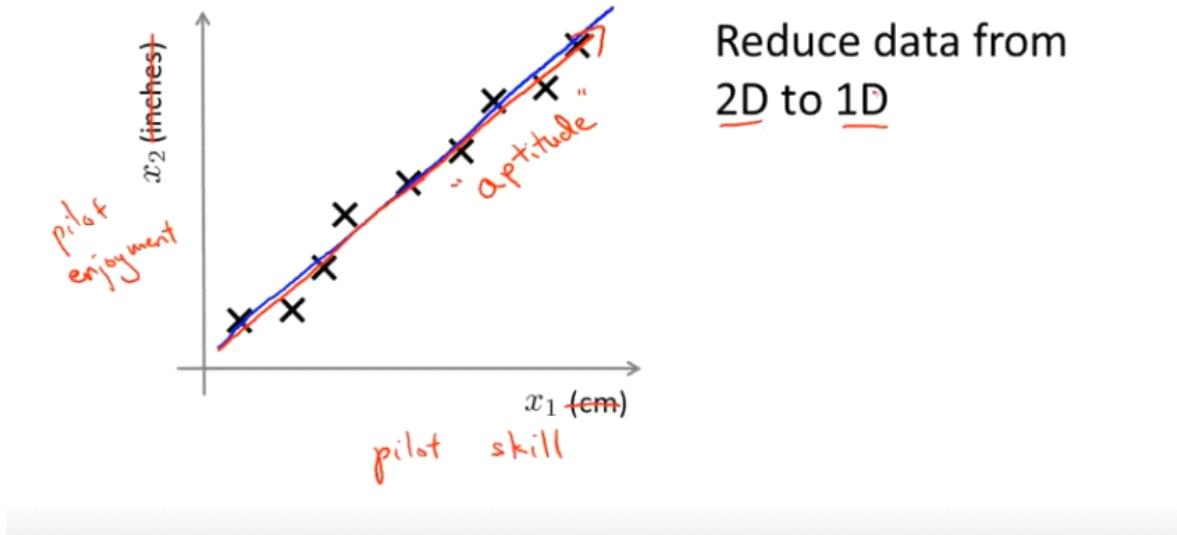
有时选取簇K的数量，可能会由业务层面所决定，所左图它把T恤分成了小中大三码，而右图则分成了五类，这时需要人工决定选取多少个码号。

(1) 数据压缩

2022年2月6日 11:17

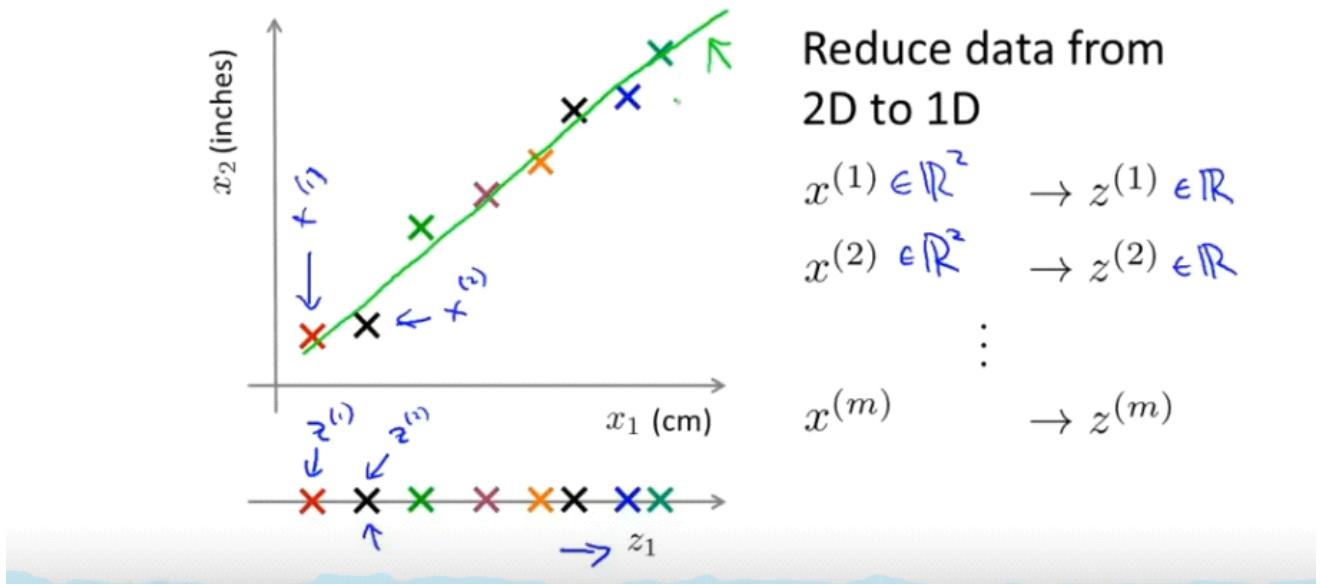
降维的第一个应用是数据压缩，数据压缩不仅能让对我们对数据进行压缩使得数据占用较少的内存和硬盘空间，它还让我们能对学习算法进行加速。

Data Compression



如上图所示，我们把2维数据压缩成1维数据。我们采用降维的方式进行数据的压缩，把二维的数据压缩在一条直线上，即从 $(x_1, x_2) \rightarrow z_1$ 。

Data Compression

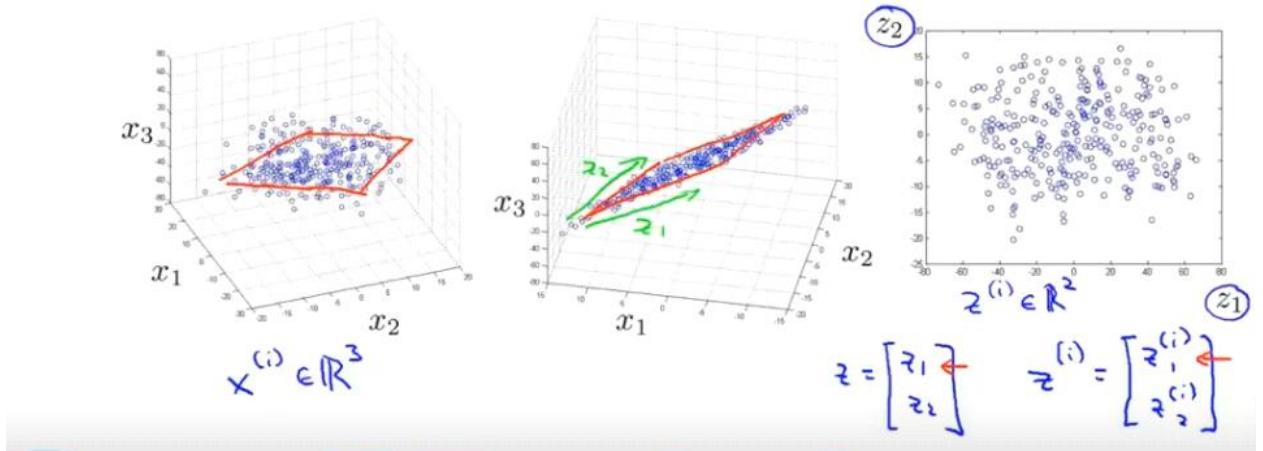


上图的产颜色不同的点代表不同的样本的点。

Data Compression

$10000 \rightarrow 1000$

Reduce data from 3D to 2D



上图是一个把3维压缩成2维的例子，首先将三维数据压缩在一个平面上，然后再进行二维显示。

(2) 可视化

2022年2月6日 11:25

降维的第二个应用是可视化数据，可视化数据在许多机器学习的应用可以帮助我们对学习算法进行优化，让我们更好地了解我们的数据这可以帮助我们更多地对数据进行可视化。

Data Visualization

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

Country	x_1 GDP (trillions of US\$)	x_2 Per capita GDP (thousands of intl. \$)	x_3 Human Development Index	x_4 Life expectancy	x_5 Poverty Index (Gini as percentage)	x_6 Mean household income (thousands of US\$)	...
Canada	1.577	39.17	0.908	80.7	32.6	67.293	...
China	5.878	7.54	0.687	73	46.9	10.22	...
India	1.632	3.41	0.547	64.7	36.8	0.735	...
Russia	1.48	19.84	0.755	65.5	39.9	0.72	...
Singapore	0.223	56.69	0.866	80	42.5	67.1	...
USA	14.527	46.86	0.91	78.3	40.8	84.3	...
...

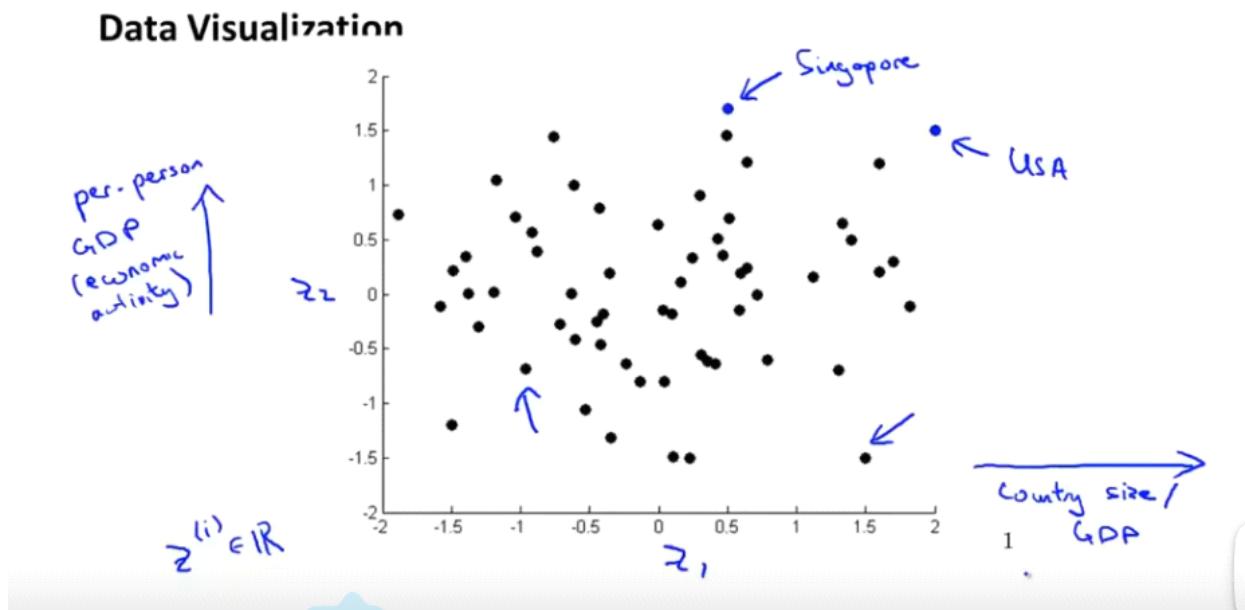
如上图所示，这是一个与国家有关的50维数据。

Data Visualization

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

Country	z_1	z_2	
Canada	1.6	1.2	
China	1.7	0.3	Reduce data from 50D to 2D
India	1.6	0.2	
Russia	1.4	0.5	
Singapore	0.5	1.7	
USA	2	1.5	
...	

我们把50维变成2维的数据。

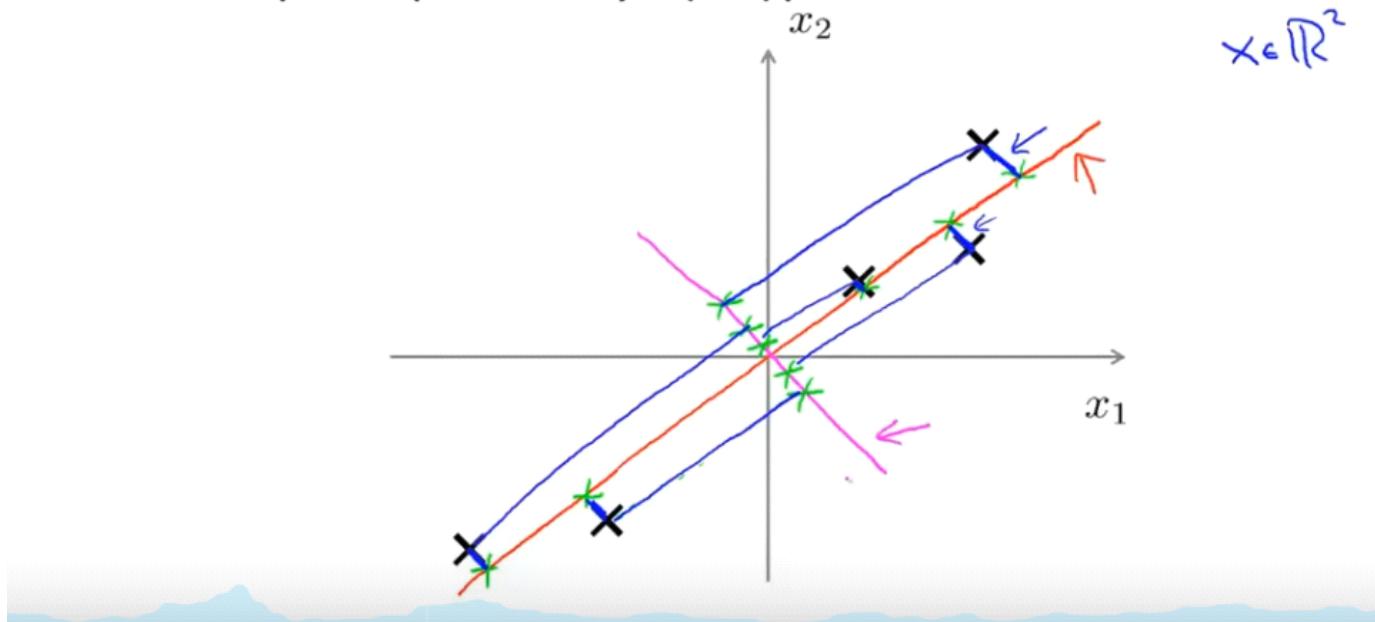


我们把数据用 z_1 和 z_2 来表示，其中 z_1 表示国家的GDP， z_2 表示国家的幸福指数。

(3) 主成分分析方法(PCA) 1

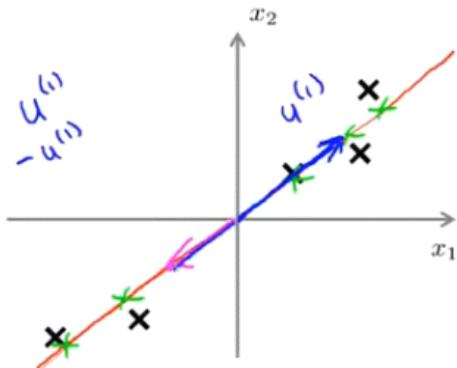
2022年2月6日 13:51

Principal Component Analysis (PCA) problem formulation



我们需要把二维数据变成一维数据，即把 x_1, x_2 上的叉点投影到红色线条上，如图所示，叉点上的数据位置与直线投影点之间的距离非常小。即PCA的做法是找一个低维平面，在该例子中是红色线条，然后将数据投影在上面，使得蓝色线条长度的平方最小，这些蓝色线条被称为“投影误差”。所以PCA所做的是，会试图寻找一个投影平面对数据的投影，使能最小化这个距离。另外在应用PCA之前需要均值归一法和特征规范化，使得特征 x_1 和 x_2 其均值为0，并且其数值在可比较范围内。在上图中还有一条品红色的线条，其 x_1, x_2 在品红色线条上的投影距离有较大的距离，即蓝色线条过多，所以PCA算法会选择红色线条而不是品红色线条。

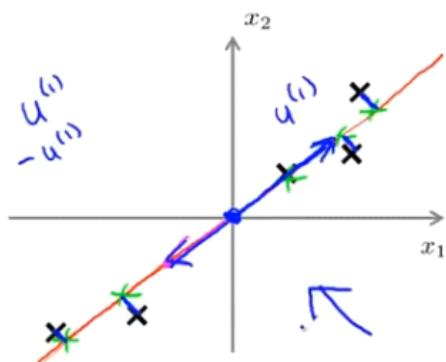
Principal Component Analysis (PCA) problem formulation



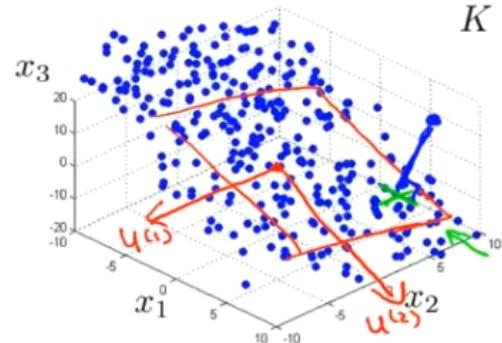
Reduce from 2-dimension to 1-dimension: Find a direction (a vector $\underline{u^{(1)} \in \mathbb{R}^n}$) onto which to project the data so as to minimize the projection error.

在该例子中，PCA想做的是将二维降到一维，我们要试着找到一个向量，假设向量 $u^{(i)}$ 它是属于 $u^{(n)}$ 空间中的向量。在这个例子中，就是 $\mathbb{R}^{(2)}$ 空间。我们需要找到一个数据投影后能够最小化投影误差的方向。所以在这个例子中，我们需要把数据投影到图中的直线上，最后会得到非常小的重构误差。

Principal Component Analysis (PCA) problem formulation



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



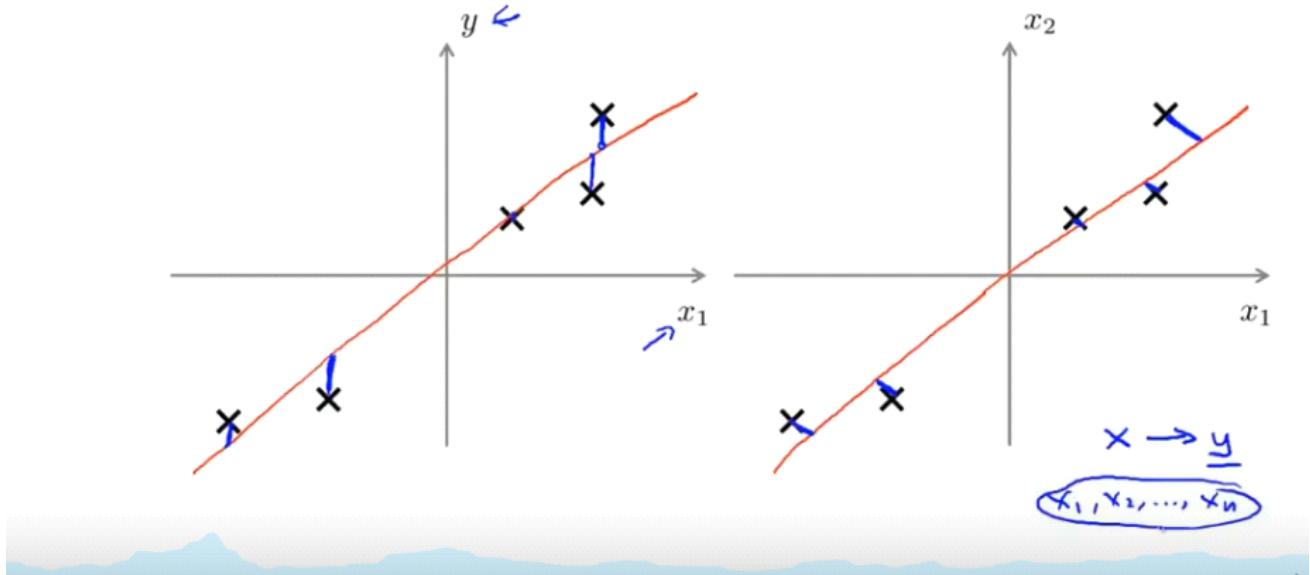
Reduce from 2-dimension to 1-dimension: Find a direction (a vector $\underline{u^{(1)} \in \mathbb{R}^n}$) onto which to project the data so as to minimize the projection error.

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

更为一般时，我们需要把N维空间投影到K维空间中，并且需要保证最小化投影误差。上图有一个将3维投影到2维平面的例子，给点一个三维点把它投影到二维平面上，到完成时，投影误差就是这个点

到二维平面上对应的投影点的距离，故PCA做的是试图找出一条直线或一个平面或其它维的空间然后对数据进行投影以最小化平方投影、90度投影或下次投影的误差。

PCA is not linear regression



PCA不是线性回归

线性回归是通过 X 去预测 Y ，而PCA是每个特征向量都是平等的，另外，线性回归是样本 X 与直线的垂直距离，而PCA是样本与直线的距离（可能垂直，也可能斜的）。

(4) 主成分分析方法(PCA) 2

2022年2月6日 14:33

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.

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

在使用PCA之前，我们需要对数据进行预处理，即我们需要执行均值标准化，依据数据可能也需要进行特征缩放。

均值标准化，对未标记数据做的，首先我们需要计算每个特征的均值，采用公式，

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

进行计算。对于每个样本 $x_j^{(i)}$ ，我们使用每个样本的特征减去特征的均值，即

$$x_j = x_j^{(i)} - \mu_j$$

这将使每个特征的均值正好为0。

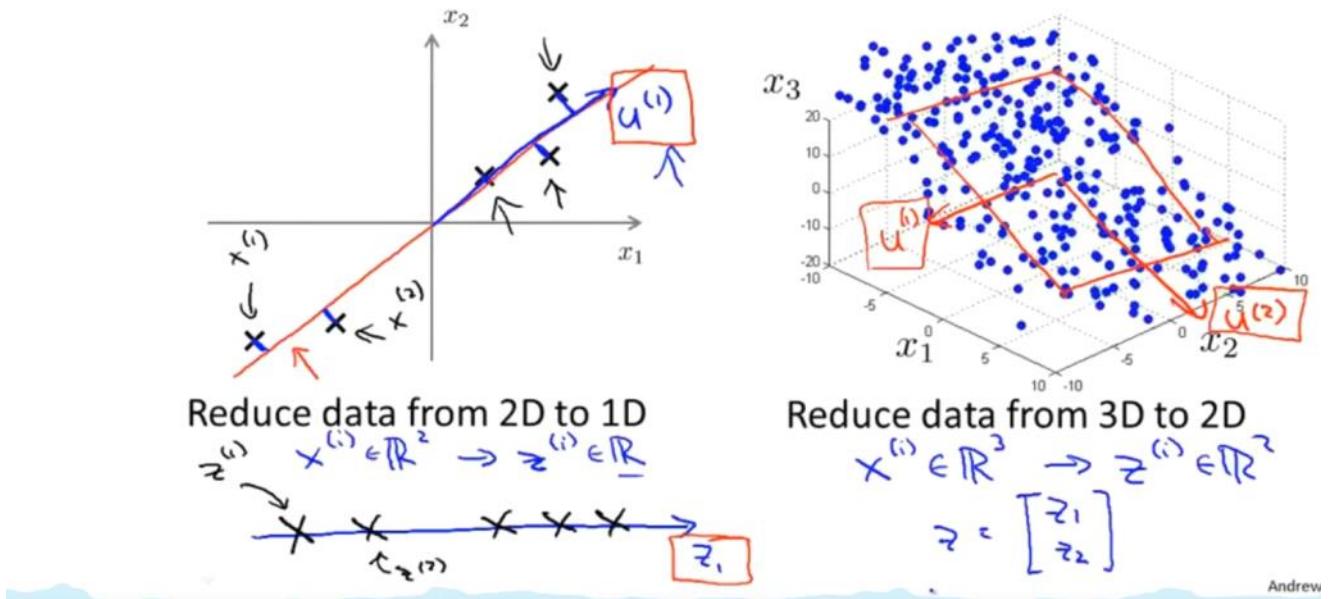
若有不同的特征具有不同的缩放，如 x_1 为房屋大小， x_2 为卧室数量，我们缩放特征在一个相对的价值范围内。

即公式

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

μ_j 是特征的均值， s_j 是特征 j 的一些测量值，它是最大值减去最小值或更普遍的是一个特征 j 的标准偏差。

Principal Component Analysis (PCA) algorithm



将从二维减少数据到一维，从三维减少数据到二维。

Principal Component Analysis (PCA) algorithm

Reduce data from n -dimensions to k -dimensions

Compute "covariance matrix":

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

Compute "eigenvectors" of matrix Σ :

$$\rightarrow [U, S, V] = \underline{\text{svd}}(\text{Sigma}) ; \quad \rightarrow \underline{\text{Singular value decomposition}}$$

$\underbrace{\qquad\qquad\qquad}_{n \times n \text{ matrix.}}$ $\underbrace{\qquad\qquad\qquad}_{\text{eig}(\text{Sigma})}$

$$U = \left[\begin{array}{c|c|c|c|c} | & | & | & | & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \dots & u^{(n)} \\ | & | & | & & | \end{array} \right] \quad U \in \mathbb{R}^{n \times n}$$

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

Andrew Ng

PCA 算法

从N维减少数据到K维。

第一步计算协方差，公式

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

其中 $x^{(i)}$ 为 $n \times 1$ 的矩阵， $(x^{(i)})^T$ 为 $1 \times n$ 的矩阵，故 Σ 为一个 $n \times n$ 的矩阵。

第二步计算 Σ 的特征向量，有公式

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

svd 代表奇异值分解，Svd(Sigma) 表示一个 $n \times n$ 的矩阵，这个表达式输出三个矩阵分别是 U, S, V ，我们只需要了解 U 矩阵。 U 矩阵是一个 $n \times n$ 的矩阵即 $u = [u^{(1)} \ u^{(2)} \ u^{(3)} \dots u^{(n)}]$

而我们需要做的是提取前 K 个向量，我们将提取 $u^{(1)}, \dots, u^{(k)}$

Principal Component Analysis (PCA) algorithm

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

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

$\underbrace{\quad\quad\quad}_{K}$

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

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

$\underbrace{\quad\quad\quad}_{n \times k}$

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

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

U_{reduce}

Andrew Ng

从 $svd(\text{Sigma})$ 中我们将得到

$$u = [u^{(1)} \ u^{(2)} \ u^{(3)} \dots u^{(n)}] \in R^{n \times n}, \text{ 我们需要把 } u \in R^n \rightarrow z \in R^k \text{ 有}$$

我们先从矩阵 U 中提取 K 个向量有

$$U_{reduce} = [u^{(1)} \ u^{(2)} \ u^{(3)} \dots u^{(k)}]$$

U_{reduce} 为一个 $n \times k$ 的矩阵，我们把矩阵 U_{reduce} 乘以样本 $x^{(i)}$

即完成了样本 $x^{(i)}$ 从m维空间降维到K维空间的过程。即有公式

$$z^{(i)} = [u^{(1)} \ u^{(2)} \ u^{(3)} \dots u^{(k)}]^T x^{(i)} = \begin{bmatrix} (u^{(1)})^T \\ (u^{(2)})^T \\ \vdots \\ (u^{(k)})^T \end{bmatrix} x^{(i)}$$

其中 $z^{(i)}$ 为一个 $k*1$ 的矩阵。

Principal Component Analysis (PCA) algorithm summary

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

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

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

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

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

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

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

↑ ↑

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

PCA算法总结

我们首先进行预处理，即对样本进行均值归一化和特征规范化，再计算sigma，通过svd进行分解，得到u矩阵，再提取U矩阵前K个向量

组成 U_{reduce} 矩阵，再把 U_{reduce} 矩阵乘以样本 $x = \begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \vdots \\ (x^{(k)})^T \end{bmatrix}$ ，最后得到 z 。这样就完成了从N维到K维的转换。

注：类似于K均值，在应用PCA算法时，即有 $x \in R^n$ ，所以 x_0 不能为1

(5) 主成分的数量选择

2022年2月6日 16:47

在PCA算法中，我们将N维特征减少为某K维特征，数字K为PCA算法中的一个参数，也被称为主成分数字。

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

$$\rightarrow \frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2} \leq \frac{0.01}{0.05}$$

~~(1%)~~
~~5%~~
~~(10%)~~

Because for most real life

~~“99% of variance is~~ 因为对于真实世界的数据

~~gets to 90%.~~

PCA试图去减少投影误差平方的平均值，即则有平均投影误差公式

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

其中 $x^{(i)}$ 为原始样本数据， $x_{approx}^{(i)}$ 为原始样本数据的投影。

即试图去减少 x 和 x 在低维上的投影之间的误差。

数据总方差公式，

$$\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2$$

即数据的总方差为训练集中每个训练样本的平均长度，这个表明平均来说训练样本距离全零向量的距离或者说，训练样本距离原点有多远。

Typically, choose k to be smallest value so that

$$\begin{aligned} &\rightarrow \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2 \leq \frac{0.01}{0.05} \quad (1\%) \\ &\rightarrow \frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2 \leq \frac{0.10}{0.05} \quad (5\%) \\ &\rightarrow \frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2 \leq \frac{0.10}{0.01} \quad (100\%) \end{aligned}$$

\rightarrow "99% of variance is retained"
~~95 to 90%~~

通常来说，在我们选择K时，一个经验是选择较小的值，使得两者之间的比值小于0.01，即一个常见的方法有

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

即平均投影误差/数据总方差，这表示数据的波动程度。

这个比例不定是0.01也可能是0.02或者其它值。当这个比值小于等于0.01时，表示99%的方差性会被保留。

Choosing k (number of principal components)

Algorithm:

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

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

Check if

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

$$k = 17$$

选择主成分数量

算法：

试图令 $k=1$ ，然后计算 $u_{reduce}, z^{(1)}, \dots, z^{(m)}, x_{approx}^{(1)}, \dots, x_{approx}^{(m)}$ ，然后校正平均投影误差/数据总方差是否小于0.01，如果不小于0.01，则令 $k=2$ ，再重复这个算法。

上述算法比较慢，故有算法，

.....

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

$\rightarrow S =$

For given k

$$1 - \frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}} \leq 0.01$$
$$\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}} \geq 0.99$$

Andrew Ng

利用SVD(Sigma)得到U,S,V矩阵，

其中有S矩阵

$$S = \begin{pmatrix} S_{11} & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & S_{nn} \end{pmatrix}$$

即该矩阵只有对角线有元素，其它元素皆为0。

我们首先令 $k=1$ ，计算公式

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

其公式 S_{ii} 为矩阵S对角线上的元素。

即探测公式

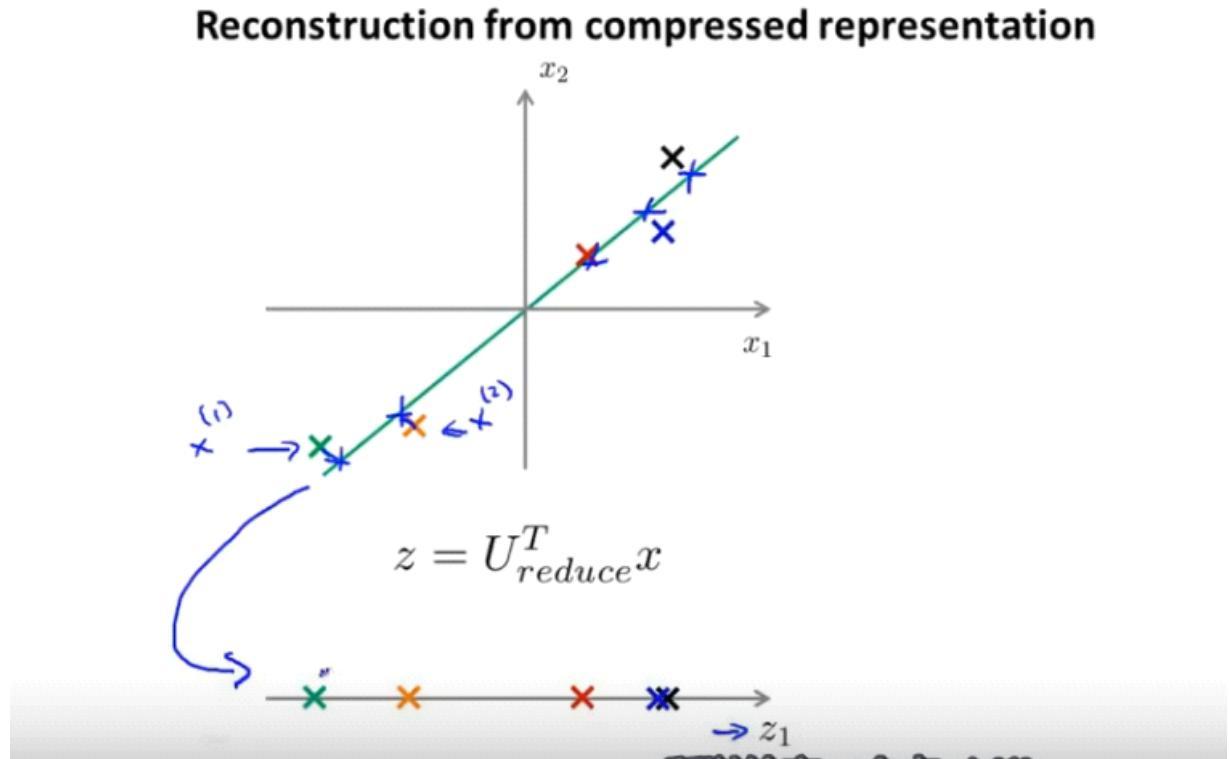
$$\frac{\sum_{i=1}^k s_{ii}}{\sum_{i=1}^n s_{ii}} \geq 0.99$$

是否成立，若不成立，则令k=2，再重复上述过程，依次类推。

(6) 压缩重构(压缩还原)

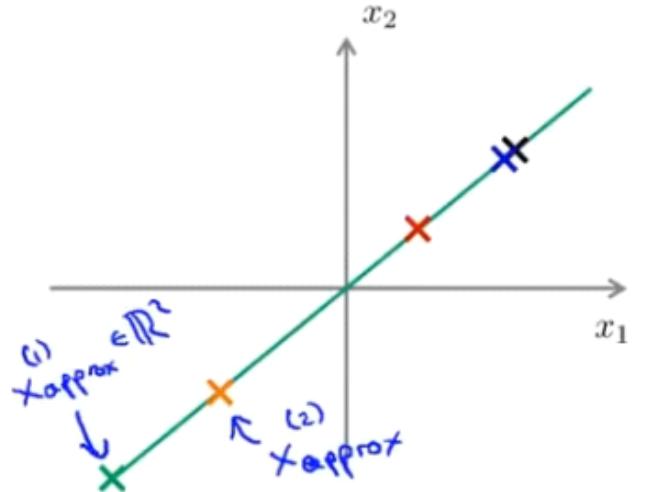
2022年2月7日 10:48

通过PCA我们可以把一个1000维的数据压缩成100维的特征向量，或者把一个三维压缩成二维，既然可以把高维数据压缩成低维数据，那么也有方法可以把低维还原为高维数据的一种近似表示。



这里是一个将二维数据压缩到一维的例子，即将 $x_1, x_2 \rightarrow z_1$ 从平面压缩到直线。

Dimensionality reduction



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

$$\boxed{x_{approx}^{(1)} \in \mathbb{R}^n = U_{reduce} \cdot z^{(1)}}$$

Andrew Ng

这是一个经由二维压缩成一维，然后再从一维还原于二维的例子。我们有一维数据集 $Z \in R$ ，有二维数据 $x \in R^2$ 。有公式

$$x_{approx}^{(i)} = U_{reduce} * z^{(i)}$$

其中 $x_{approx}^{(i)}$ 为一个近似值， $x_{approx}^{(i)} \in R^n$ 。

U_{reduce} 为一个 $n*k$ 矩阵， $z^{(i)}$ 为一个 $k*1$ 的矩阵。

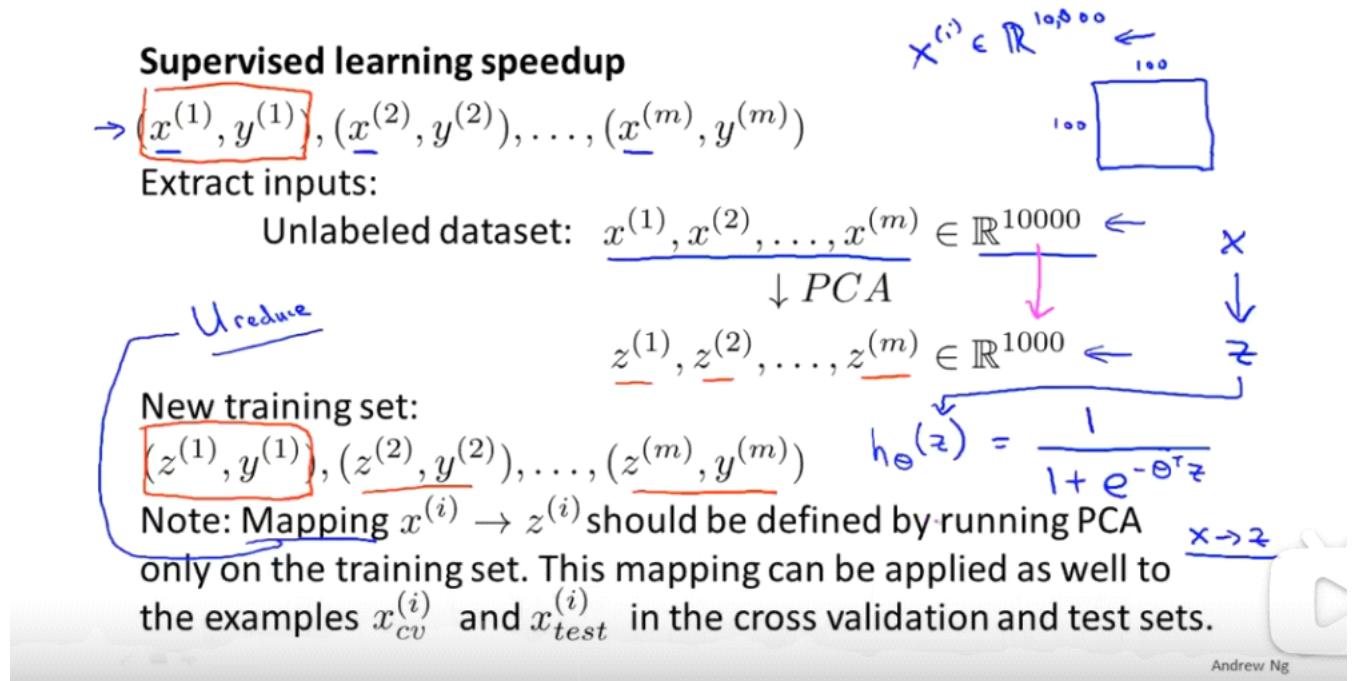
注：公式有以下公式推导

$$z^{(i)} = [u^{(1)} \ u^{(2)} \ u^{(3)} \dots u^{(k)}]^T \ x^{(i)} = \begin{bmatrix} (u^{(1)})^T \\ (u^{(2)})^T \\ \vdots \\ (u^{(k)})^T \end{bmatrix} x^{(i)}$$

$$U_{reduce} = [u^{(1)} \ u^{(2)} \ u^{(3)} \dots u^{(k)}]$$

(7) 应用PCA建议

2022年2月7日 11:49



PCA应用之一是为监督学习进行加速，如我们有数据集 $x^{(1)}, \dots, x^{(m)}$ ，和结果集 $y^{(1)}, \dots, y^{(m)}$ 。我们有未有标签集 $x^{(1)}, \dots, x^{(m)} \in R^{10000}$ ，因为这个数据集太大，导致算法进行缓慢。所以我们运用PCA算法进行加速。使得 $x^{(1)}, \dots, x^{(m)} \in R^{10000}$ 映射为 $z^{(1)}, \dots, z^{(m)} \in R^{1000}$ 。故我们有新的训练集： $(z^{(1)}, y^{(1)}), \dots, (z^{(m)}, y^{(m)})$ 。

注：映射 $x^{(i)} \rightarrow z^{(i)}$ 应该只有训练集上运行PCA，当映射完成后应用在交互验证集和测试集。

Application of PCA

- Compression

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

Choose k by % of variance retain

- Visualization

$k=2$ or $k=3$

PCA应用

- 压缩

- 1) 减少内存/硬盘的存储空间。

- 2) 学习算法的加速

注：以上两种应用需要保留方差率在99%

- 可视化

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.

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

PCA的滥用：防止过拟合

使用PCA从 $x^{(i)} \rightarrow z^{(i)}$ 使得特征的数量从n减少到k。因此， $z^{(i)}$ 具有更少的特征，所以有更少可能过拟合。这种应用是错误的，尽管它可能运行的很好，但PCA从x映射到z中间会丢弃掉一些特征，这里面可能会有重要的信息，所以最好使用正则化。

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 $z^{(i)}$~~
- - Train logistic regression on $\{(\cancel{z^{(1)}} \cancel{y^{(1)}}, \dots, (\cancel{z^{(m)}} \cancel{y^{(m)}})\}$
- - Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$. Run $h_\theta(z)$ on $\{(z_{test}^{(1)}, y_{test}^{(1)}), \dots, (z_{test}^{(m)}, y_{test}^{(m)})\}$
- 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 $x^{(i)}$. Only if that doesn't do what you want, then implement PCA and consider using $z^{(i)}$.

PCA的滥用：有时PCA不应该被使用

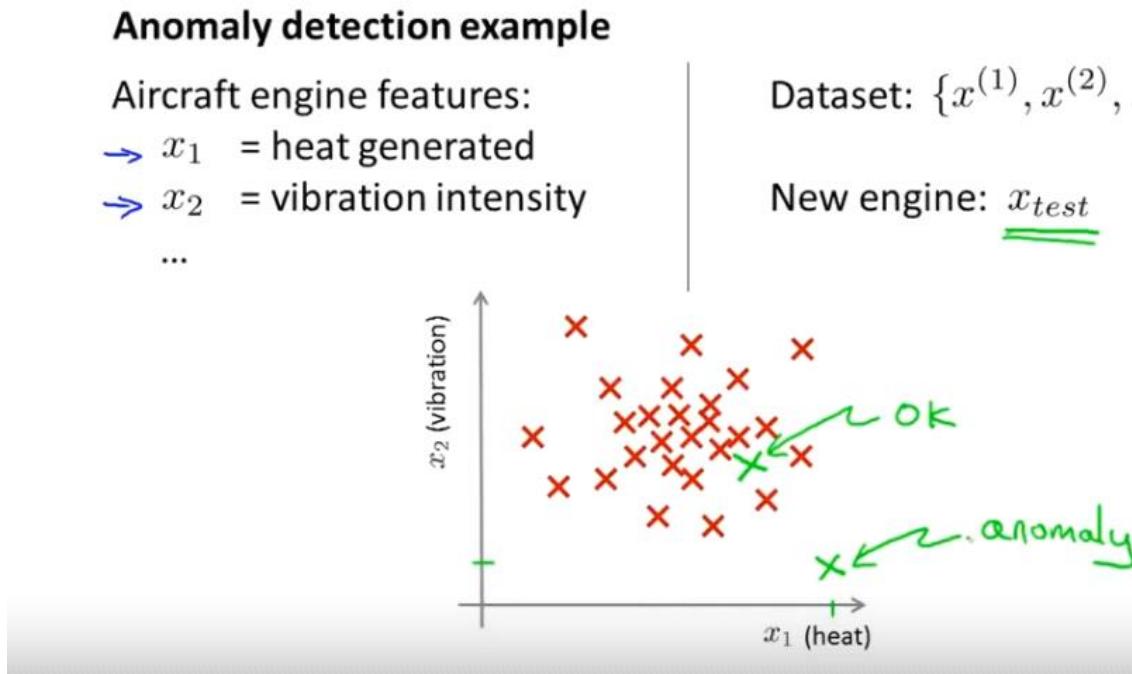
如你想要设计一个机器学习系统，计划如上图所示，这里的错误是你不应该一开始就把PCA考虑进来，应该先使用原始数据 $x^{(i)}$ 除非因为算法运行太慢或者占用硬盘空间太大或 $x^{(i)}$ 不可用，这些情况才应该考虑使用 $z^{(i)}$ 。

(8) PCA的数学原理

2022年2月8日 15:53

(1) 异常检测问题

2022年2月8日 16:00



异常检测问题：

假设你在生产飞机引擎时，你需要进行质量控制测试，而作为这个测试的一部分，已经测试了飞机部件的一些特征，如 x_1 高热量的产生， x_2 引擎的振动等。如何画成图就是如上图所示，这里的每个带叉的点都是无标签的数据。

异常检测问题定义如下：

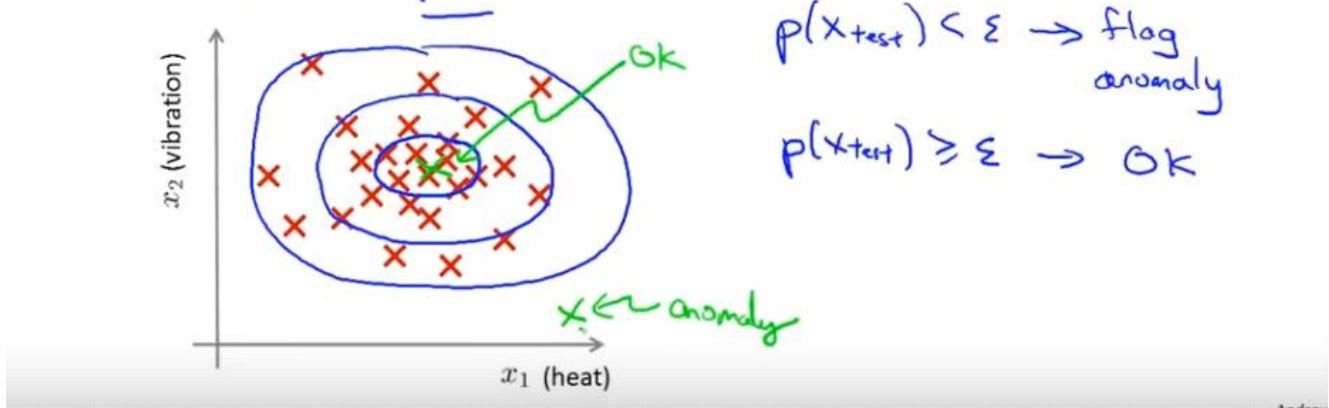
假设你有一个新的飞机引擎，你的飞机引擎有一个特征变量集 x_{test} , 所谓的异常检测问题就是我们想知道新的飞机引擎是否有某种异常，即有点落到一堆带叉的数据里，我们就认为这个引擎正常，而若有点落入离带叉的点很远的距离，就代表这点异常。

Density estimation

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

→ Is x_{test} anomalous?

Model $p(x)$.



密度评估

若我们有数据集合 $\{x^{(1)}, \dots, x^{(m)}\}$ ，我们需要判别测试集 X_{test} 是否异常，我们要采用的方法是给定无标签的训练集，对数据建模即 $p(x)$ 。即我们将对 X 的分布概率建模，其中 X 是这些特征变量，因此，当我们建立了 X 的概率模型之后，我们就有对于测试集 x_{test} ，

$$p(x_{test}) < \epsilon \rightarrow \text{flag anomaly}$$

即给入训练集的数据到 P 函数里，若函数 $p(x_{test})$ 低于阈值 ϵ ，就说它异常，反之，若公式

$$p(x_{test}) \geq \epsilon \rightarrow \text{ok}$$

成立，即函数 $p(x_{test})$ 大于等于阈值 ϵ ，就说它正常。

给定图中，这样的训练集，建立了一个模型，你很有可能发现模型 $p(x_{test})$ ，将这些中心区域的点概率相当的大，而那些稍微远离中心区域的点概率会小些，更远地方的点，概率会更小，而外面的点将会成为异常点，而在中心区域的点为正常点。

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 $\underline{p(x) < \varepsilon}$

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.

...

$p(x) < \varepsilon$

异常点控制例子

一、欺诈检测

二、制造业

参考飞机引擎例子。

三、数据中心监视计算机

(2) 高斯分布(正态分布)

2022年2月8日 16:55

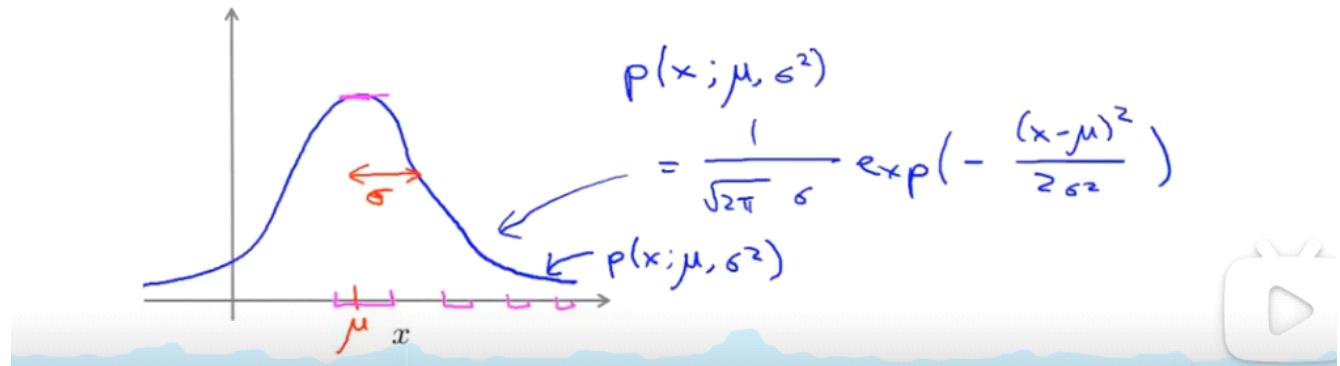
Gaussian (Normal) distribution

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

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

↖ "distributed as"

σ standard deviation



高斯分布

若有实数 x ，服从高斯分布，其中均值为 u ，方差为 θ^2 。记作，
 $X \sim N(u, \theta^2)$

其中高斯分布的图形如上图所示，参数 u 在图形的中间位置，参数 θ 控制宽度。

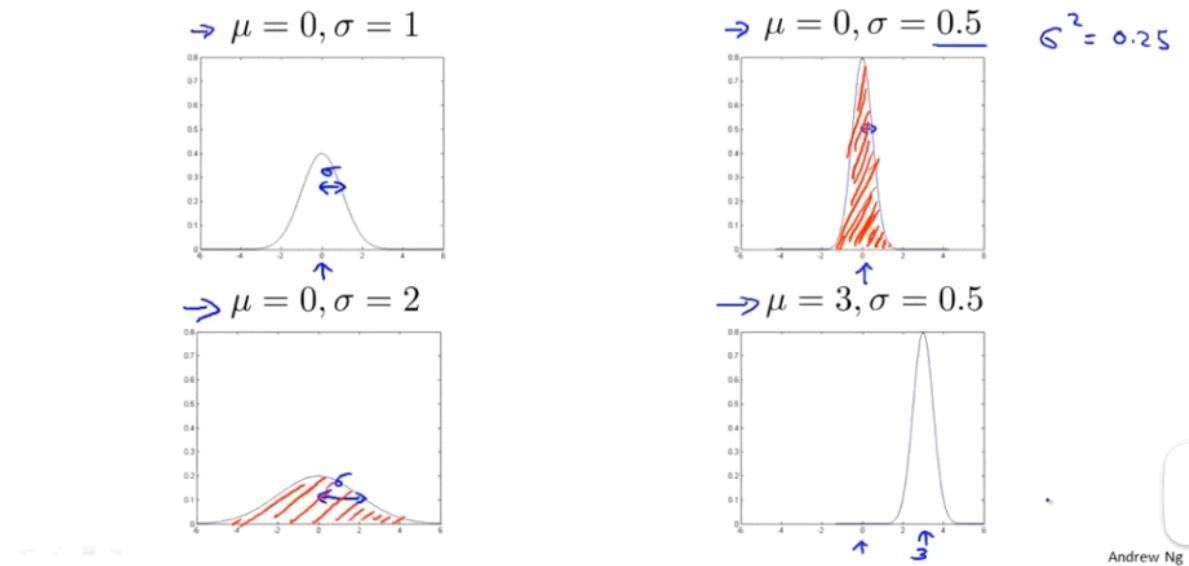
高斯分布的数字公式，

$$p(x; u, \theta^2)$$

x 的概率分布，由参数 u, θ^2 决定。有公式

$$p(x; u, \theta^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-u)^2}{2\sigma^2}}$$

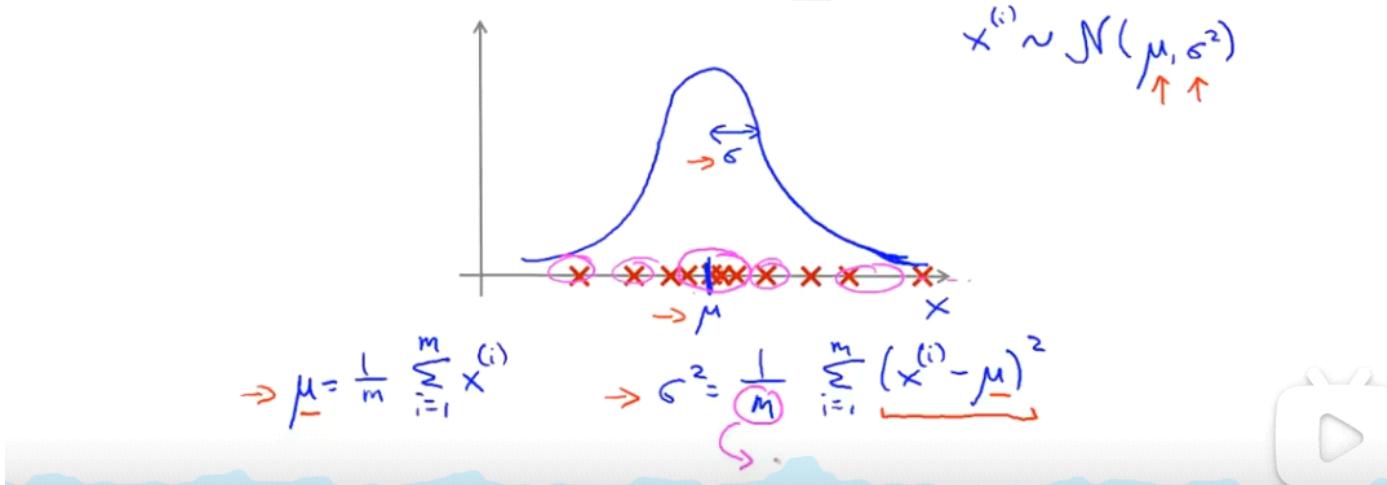
Gaussian distribution example



几个高斯分布的例子。

Parameter estimation

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



参数估计：

我们有数据集 $\{x^{(1)}, \dots, x^{(m)}\}$ $x^{(i)} \in R$ ，其中图形如上图所示，参数估计问题是假设所猜测的这些样本来自一个高斯分布的总体，假设所猜测的每一个样本 $x^{(i)}$ 服从某个分布，有公式。

$$X \sim N(\mu, \theta^2)$$

这个公式由参数 μ, θ^2 决定，但是所不知道的是参数 μ, θ^2 的值。参数估计问题就是给定数据集，我们希望找到能够估算出 μ, θ^2 的值。

如上图所示，那个高斯分布线条很好地拟合了数据，其中 u 为中值， θ 控制宽度，并且保证了在中间区域 u 的概率最大，然后依次递减，我们有估计参数 u, θ^2 公式，

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

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

对于估计参数 σ^2 ，有公式

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

这两个公式，在样本较大的情况下区别不大，对于结果没有很大的影响。

(3) 异常检测算法

2022年2月9日 9:41

→ Density estimation

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

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

→ $p(x)$

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

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

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

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

密度评估

我们有训练集 $\{x^{(1)}, \dots, x^{(m)}\}$, 对于每个样本都有 $x \in \mathbb{R}^n$ 。对于每个样本我们有特征从 1 到 n 。且每个特征符合高斯分布, 有公式

$$x_1^{(i)} \sim N(\mu_1, \sigma_1^2), \dots, x_n^{(i)} \sim N(\mu_n, \sigma_n^2)$$

故有公式

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

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

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 $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$
 - $\rightarrow \mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$ $p(x_j; \mu_j, \sigma_j^2)$ $\mu_1, \mu_2, \dots, \mu_n$
 - $\rightarrow \sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2$ $\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix} = \frac{1}{m} \sum_{i=1}^m x^{(i)}$
- 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 $p(x) < \varepsilon$

异常检测算法

一、挑选特征 x_i ，使得当出现异常样本时，特征值会异常地特别的大或者异常的特别的小。通常来讲，还是选择哪些能够描述你所想的一般特征的特征。

二、给出特征集，就是 m 个未做标记的样本 $\{x^{(1)}, \dots, x^{(m)}\}$ ，之后我们进行参数拟合 $\{\mu_1, \dots, \mu_n\}, \{\sigma_1^2, \dots, \sigma_n^2\}$ ，我们有参数公式

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

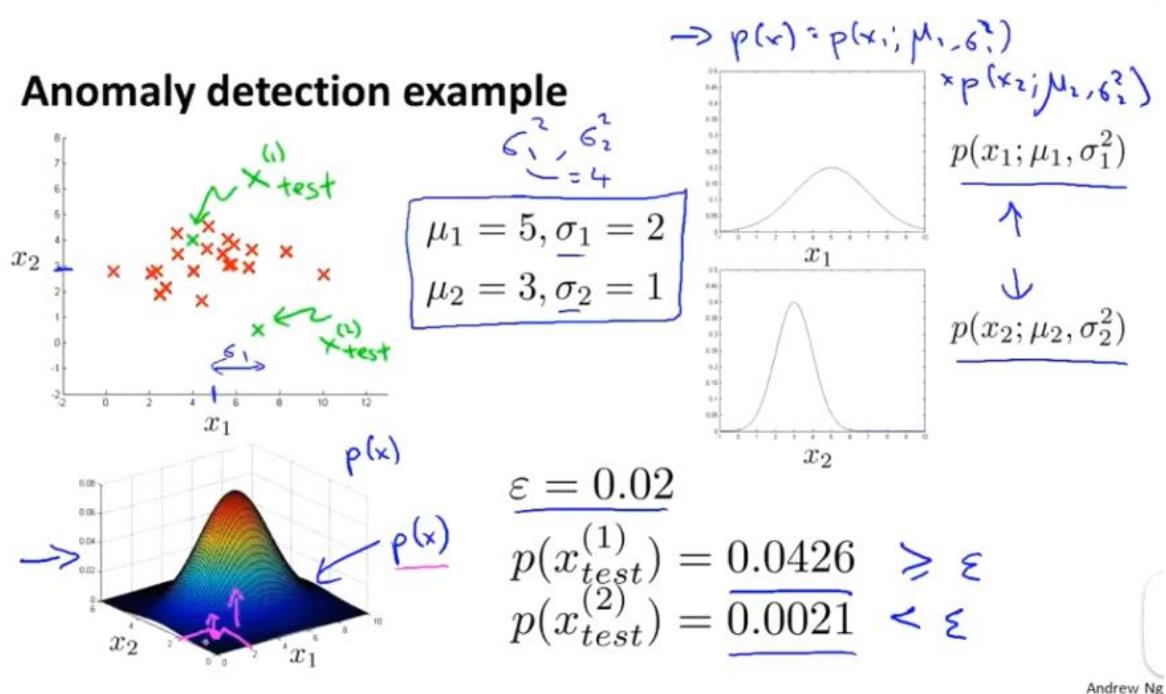
$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2$$

三、对于新样本 $x^{(i)}$ ，计算 $p(x^{(i)})$

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

经过计算后，若 $p(x^{(i)}) < \varepsilon$ ，则称这个样本异常。

Anomaly detection example



Andrew Ng

如上图所示，当样本的概率大于阈值时就认为是异常点。

(4) 评估异常检测系统

2022年2月9日 15:04

The importance of real-number evaluation

When developing a learning algorithm (choosing features, etc.), making decisions is much easier if we have a way of evaluating our learning algorithm.

- Assume we have some labeled data, of anomalous and non-anomalous examples. ($y = 0$ if normal, $y = 1$ if anomalous).
- Training set: $\underline{x^{(1)}, x^{(2)}, \dots, x^{(m)}}$ (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: $(x_{test}^{(1)}, y_{test}^{(1)}), \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

. $y=1$

实数评估的重要性

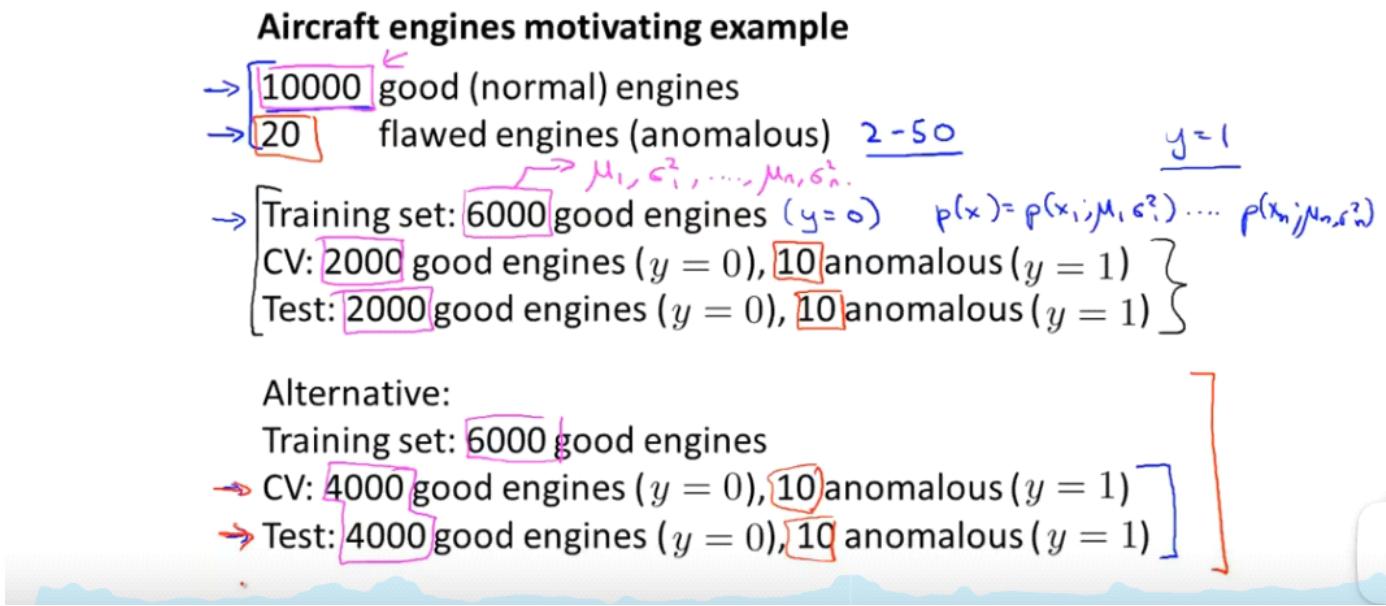
当你开发学习算法时，比如你需要选择某个特征等，如果你有某种方法能够返回一个实数来评估你的算法。那么多这些做出选择要容易的多，比如，你有一个新特征该不该纳入这个特征？如果你分别在纳入该特征和不纳入该特征的情况下运行算法，然后算法返回一个数字来告诉你这个特征对算法的影响是好还是坏。这样的话能更简单决定要不要纳入这个特征。

为了评估这个异常检测系统，假设我们有一组带标签的数据，现在我们把异常检测看作一个无监督学习问题，因为用的是无标签数据，但是如果你有一些带标签的数据，来指明哪些是异常样本，哪些是无异常样本，这就是我们认为能够评估异常检测的系统。

假设我们有一些带标签的数据代表异常样本，同时我们还有一些无标签的数据代表正常样本，用 $y=0$ 来代表正常样本，用 $y=1$ 来代

表异常样本。

我们假设有一组正常无标签的样本集 $\{x^{(1)}, \dots, x^{(m)}\}$, 我们假设交叉验证集 $\{(x_{cv}^{(1)}, y_{cv}^{(1)}), \dots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})\}$, 测试样本集 $\{(x_{test}^{(1)}, y_{test}^{(1)}), \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})\}$ 。



如我们有10000个无标签的正常样本，20个异常样本，我们设计6000个到训练集中，2000个到交叉验证集，2000个到训练集，设置10个异常样本到交叉验证集，10个到测试集。

注：切不可把交叉验证集和测试集设置为同一数据集。

Algorithm evaluation

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

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

Possible evaluation metrics:

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

CV

Test

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

Andrea

估计算法：

采用训练集 $\{x^{(1)}, \dots, x^{(m)}\}$ 的高斯分布来拟合函数 $p(x)$, 使用样本在交互验证集/测试集上，去预测 y 值。

可能的评估度量有：

1) 正阳性，假阳性，假阴性，真阴性。

2) 准确度和召回率

具体而言，我们使用交互验证集去挑选参数 ε ，挑选参数 ε 的方法是尝试去使用许多不同的 ε 值，然后从中选择一个 ε 值，该值能够最大化 F_1 - score 或者在其它方面有良好表现的 ε 值。

一般而言，使用交叉验证集和测试集的方法是，当我们进行决定要不要纳入某个特征或定义参数 ε 作为阈值时，然后我们就能在交叉验证集中评估算法，然后做出决策。例如使用哪些特征，怎么设置 ε 值，用这个方法在交叉验证集中评估算法，当我们选择好特征以后或当我们选择好合适的 ε 值时，我们就能使用最后的模型来评估算法，然后在测试集中进行最后的评估。

(5) 异常学习VS监督学习

2022年2月9日 16:44

之前有一些例子，这些例子要么正常，要么异常，相应地，我们用 $y=1$ 或 $y=0$ 标记，因此，这就产生了问题，假如我们有带标签的数据，如果我们有已知的是异常的例子，还有一些已知正常的例子。那么我们为什么用监督学习算法，用逻辑回归或神经网络来尝试直接学习我们的带标签数据来预测 $y=1$ 还是 $y=0$ 。

Anomaly detection	vs.	Supervised learning
<ul style="list-style-type: none">→ Very small number of positive examples ($y = 1$). (0-20 is common).→ 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.		<p>Large number of positive and negative examples.</p> <p>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.</p> <p>Spam</p>

异常检测情形：

- 一、当有非常少的异常样本时(0-20比较常见)
- 二、有较多的正常样本时
- 三、有许多不同类型的异常时，因为任何算法很难从异常样本中学习到异常是什么
- 四、未来出现的异常可能与已有的截然不同

使用监督学习情形：

- 一、在合理的范围内有大量的正常样本和异常样本时

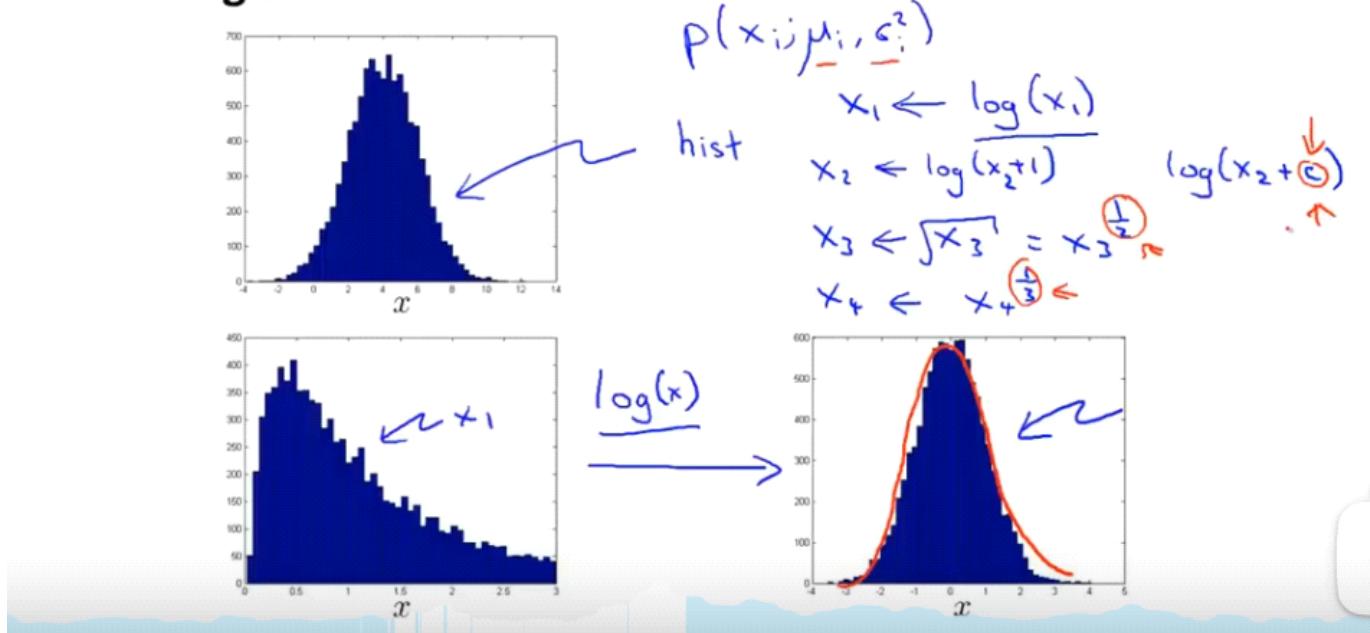
二、对于算法有足够的异常样本能够认识到异常样本是什么样，尤其时，未来可能出现的异常样本与你当前训练集出现的异常样本类似，那么在这种情况下使用监督学习算法更加合理，它能够观察大量的正常样本和异常样本来学到相应的特征，并且能够尝试区别正常和异常样本。

二都不同的关键是在异常检测中我们只有少量的异常样本，因此对于一个学习算法而言是不可能从这些异常样本中学习到足够的知识的，所我们要做的是使用大量的正常样本，让这个算法学习到足够的内容

(6) 异常检测算法的特征选择

2022年2月9日 19:24

Non-gaussian features



非高斯特征

若我们的样本特征用图形化表示出来接近高斯分布，那么可以直接应用，若样本特征用图形化表示出来与高斯分布不同，我们可以通过

$$x_1 \leftarrow \log(x_1 + a)$$

$$x_2 \leftarrow x_2^{1/b}$$

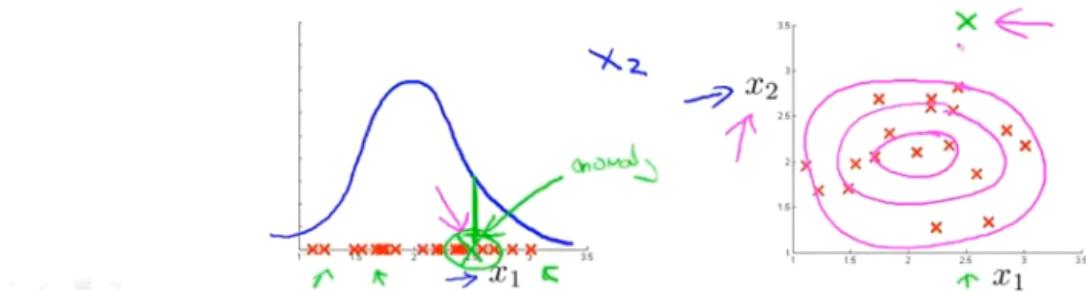
其中a,b皆为可变量，通过这种调整可以变数据更加接近高斯分布。

→ Error analysis for anomaly detection

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

Most common problem:

$p(x)$ is comparable (say, both large) for normal
and anomalous examples



如何得到异常检测算法的特征?

通过异常检测的误差分析步骤来得到。

异常检测的误差分析:

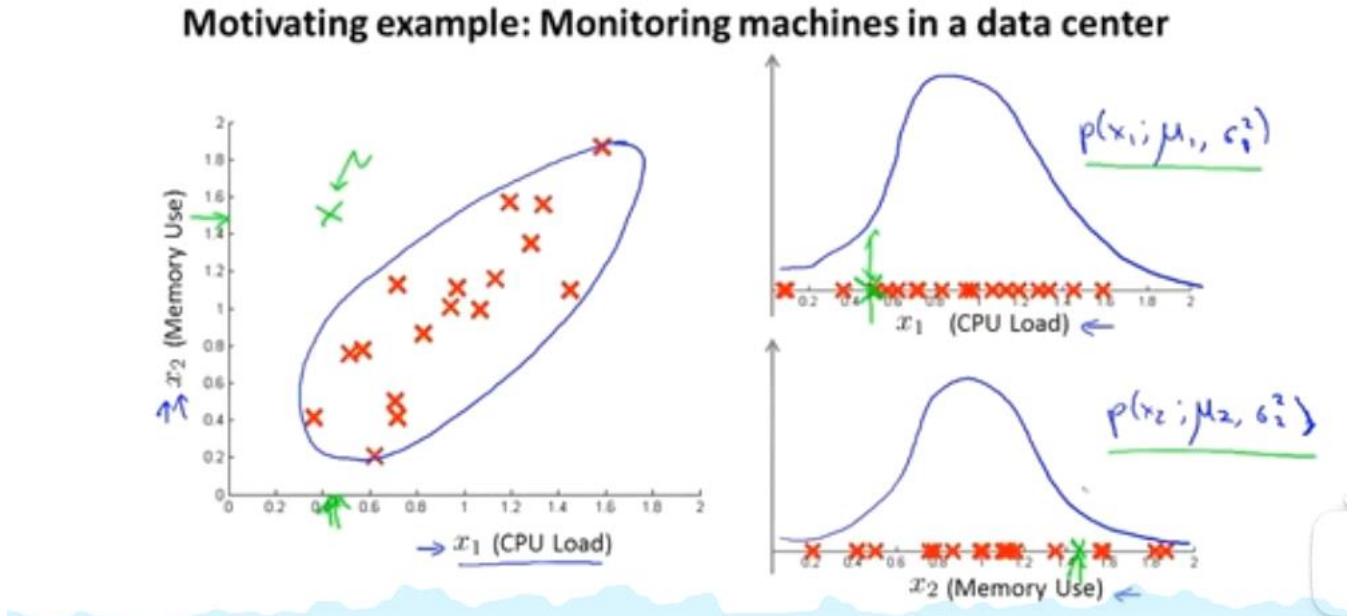
我们想要 $p(x)$ 在正常样本时概率大，在异常样本时概率小。然而更加常见的是，正常情况和异常情况一般大。

我们有异常样本在 x_1 特征里与正常样本无异，我们需要通过观察这个异常样本看能不能启发 x_2 特征，使得异常样本在 x_2 与正常样本不同，使得在 $p(x_2)$ 时概率更小。

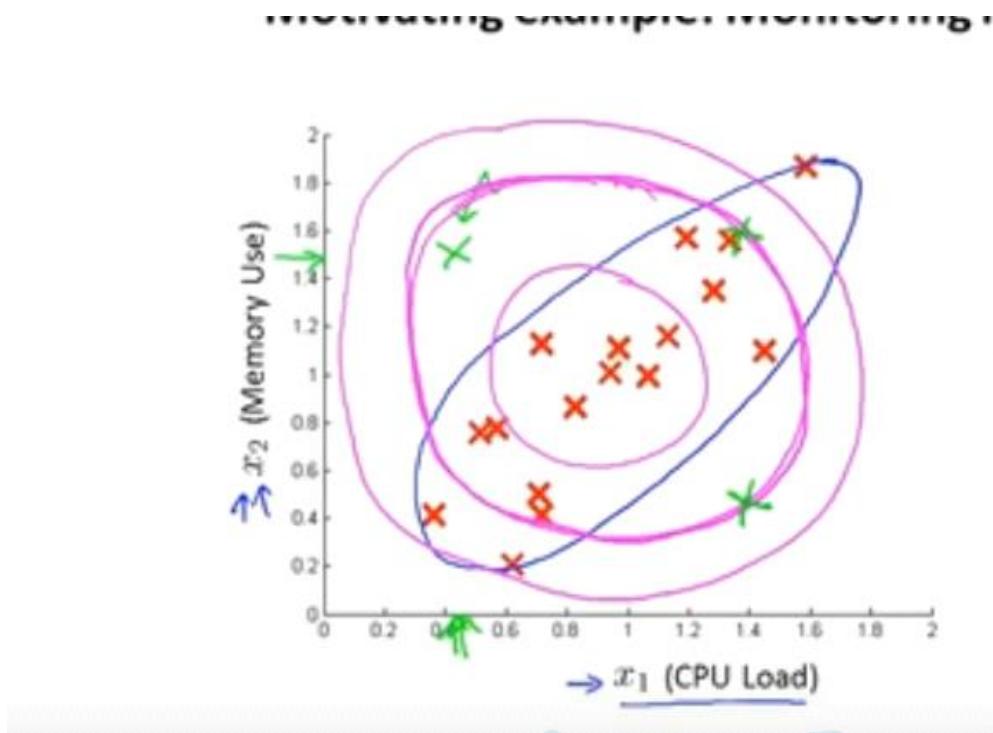
(7) 多变量高斯分布

2022年2月11日 20:58

多变量高斯分布可以捕捉到一些之前的算法检测不出来的异常。



一个数据中心监视机器的例子，左图中的横轴为CPU加载，纵轴为内存使用情况。右图中可以看到我们分别对CPU加载特征与内存使用情况特征进行高斯分布，可以直观的观察到两个绿色的叉样本(异常样本)在图中的概率并不小，即位置并不差，所以异常检测算法并不会把这个两点标记为异常。



我们发现异常检测算法并不能意识到蓝色的椭圆所表示的好样本概率高的范围，相反他认为的范围为粉红色所画的圈，他认为绿色样本是好样本的概率比较高，并且他认为最上方的绿色样本与第二上的绿色样本概率一样。

Multivariate Gaussian (Normal) distribution

$\Rightarrow x \in \mathbb{R}^n$. Don't model $p(x_1), p(x_2), \dots$, etc. separately.

Model $p(x)$ all in one go.

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

$$p(x; \mu, \Sigma) =$$

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

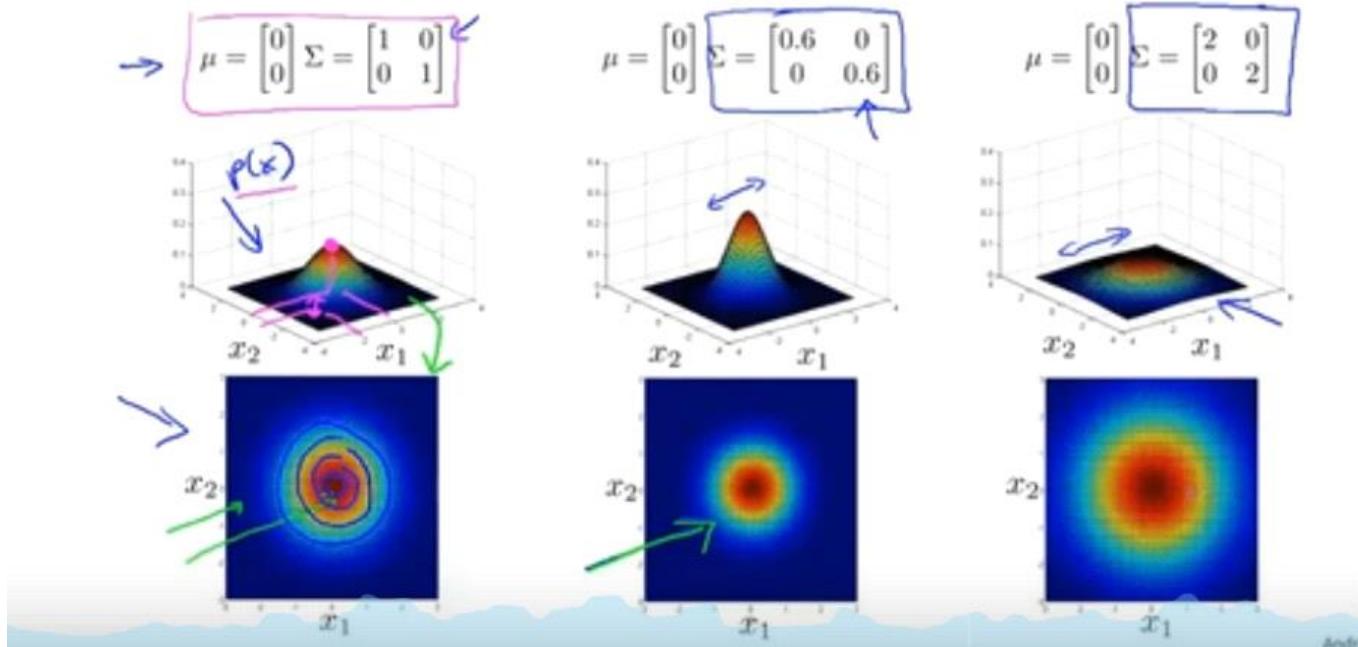
$|\Sigma| = \text{determinant of } \Sigma$ | $\det(\Sigma)$

改良版的异常检测算法需要用到多变量的高斯分布，具体做法。我们有 $x \in R^n$ ，不为 $p(x_1), p(x_2), \dots$ 分别建模。而是要建立一个整体的 $p(x)$ 模型，即为所有的 $p(x)$ 建立统一的模型。

多元高斯分布的参数是 $\mu \in R^n, \Sigma \in R^{n \times n}$, 其中符号 Σ 为协方差矩阵。其中 $p(x)$ 为

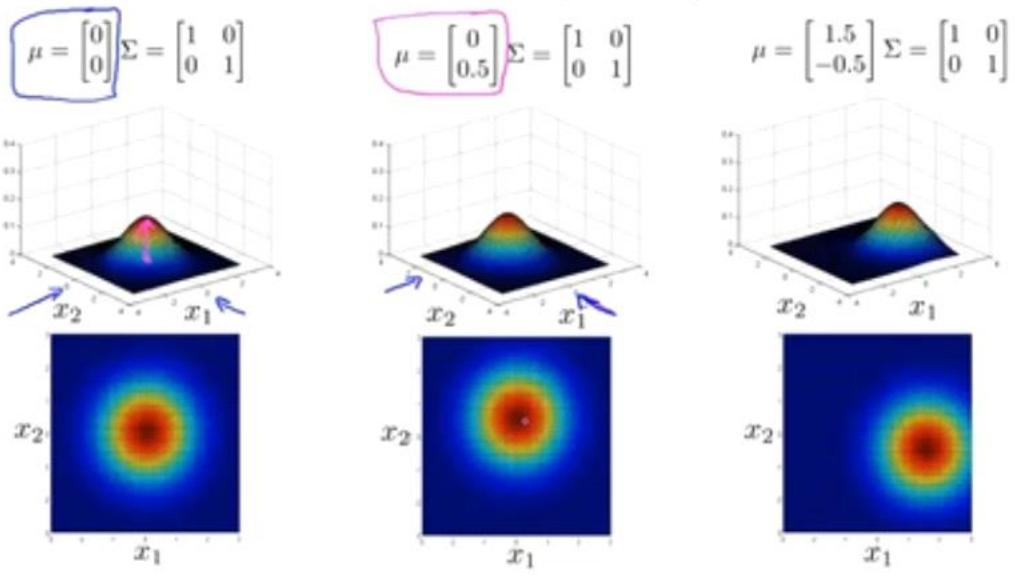
$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

Multivariate Gaussian (Normal) examples



通过改变 Σ , 改变的是方差。

Multivariate Gaussian (Normal) examples



改变u改变的是峰值。

(8) 使用多变量高斯分布的异常检测

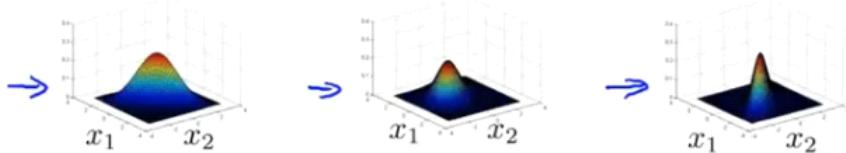
2022年2月11日 21:56

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

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

多变量高斯分布有参数 u, Σ , 其中

$$p(x; u, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{(-\frac{1}{2}(x-u)^T \Sigma^{-1} (x-u))}$$

我们有训练集合 $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\} x \in \mathbb{R}^n$

其中我们对于多变量高斯分布参数 u, Σ , 有公式

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

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

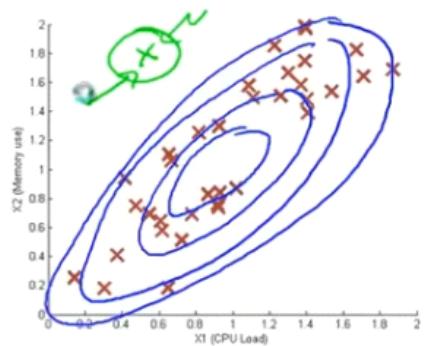
Anomaly detection with the multivariate Gaussian

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 $p(x) < \varepsilon$

多变量高斯分布的异常检测算法：

一、我们对于模型 $p(x)$ ，设置参数 μ, Σ

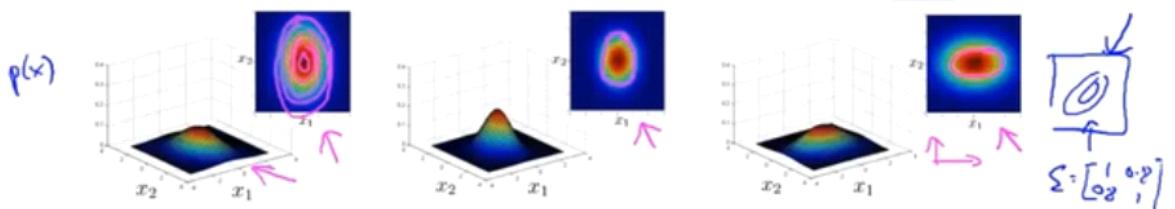
二、对于新模型 x ，我们计算公式

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

如果模型 $p(x)$ 小于 ε ，就标记为异常。

Relationship to original model

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



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

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

Andrew Ng

我们有原始模型：

$$p(x) = p(x_1; u_1, \sigma_1^2) \times p(x_2; u_2, \sigma_2^2) \times \cdots \times p(x_n; u_n, \sigma_n^2)$$

我们有高斯分布模型：

$$p(x; u, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{(-\frac{1}{2}(x-u)^T \Sigma^{-1} (x-u))}$$

当高斯分布模型中协方差矩阵：

$$\Sigma = \begin{pmatrix} \sigma_1^2 & & & & \\ & \sigma_2^2 & & & 0 \\ & & \sigma_3^2 & & \\ & & & \sigma_4^2 & \\ 0 & & & & \dots \\ & & & & \sigma_n^2 \end{pmatrix}$$

→ Original model

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

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

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

$$\Sigma^{-1}$$

Computationally more expensive

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

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

Andrew Ng

原始模型：

一、假设你在进行异常检测，假设你有一些不同特征 x_1, x_2 等，你想通过异常组合来捕捉异常样本，比如在之前的例子中，我们有这样的样本，它出现异常通常是在CPU负载以及内存使用有着异常组合时，如果用原始模型来捕捉这一点，就需要创建一个额外的特征，如上图的 x_3 ，如果花时间去创建 x_3 特征，原始模型会运行的很好。

二、原始模型，它的计算成本比较低，即这能适应巨大规模的 n ，即适

应数量巨大的特征

三、当样本数据较小时，使用原始模型会运行的比较好。

高斯模型：

一、能自动捕捉不同特征之间的关系

二、高斯模型，计算成本会比较昂贵，因为符号 $\Sigma \in \mathbb{R}^{n*n}$ ，所以在计算 Σ^{-1} 时，计算的逆矩阵会比较耗费时间。

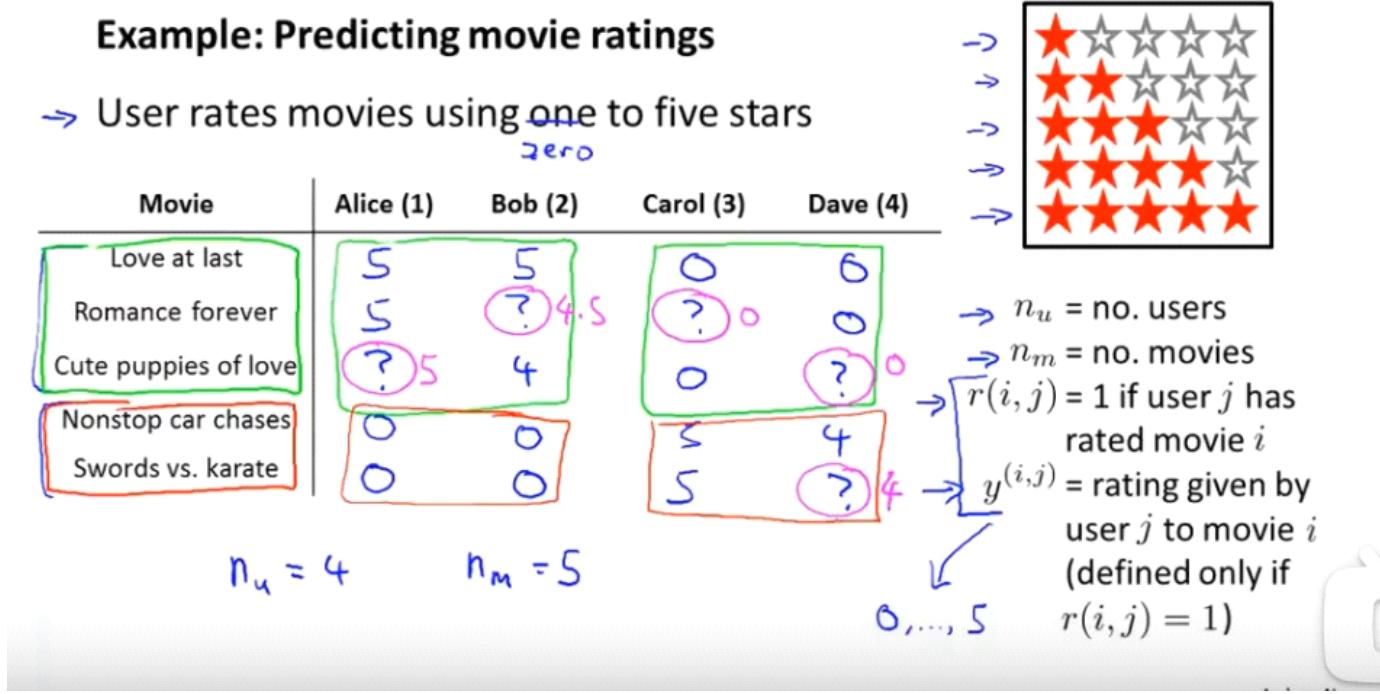
三、必须要使得m远大于n时，如m大于10n时高斯分布模型才运行的比较好，或者 Σ 不可逆时，即非线性相关。

注：需要捕捉组合异常时，需要使用原始模型，当样本数量很大时，使用高斯模型是一个比较好的选择。

(1) 推荐问题

2022年2月12日

20:53



预测电影评级：

用户评级电影使用0-5星

我们有五部电影和四位用户对电影进行评级，如上图所示，Alice和Bob通过评级表现出对于爱情片，他们有很高的评分，对于动作片却没有多少兴趣，而用户Carol和Dave对于爱情片没有多少兴趣，对于动作片却有很高的评分，具体来说推荐系统的问题是，给定以下数据，这些数据组成如下，其中

n_u 表用户的数量

n_m 表电影的数量

当 $R(i,j)=1$ ，表示用户 j 给电影 i 给出了评级

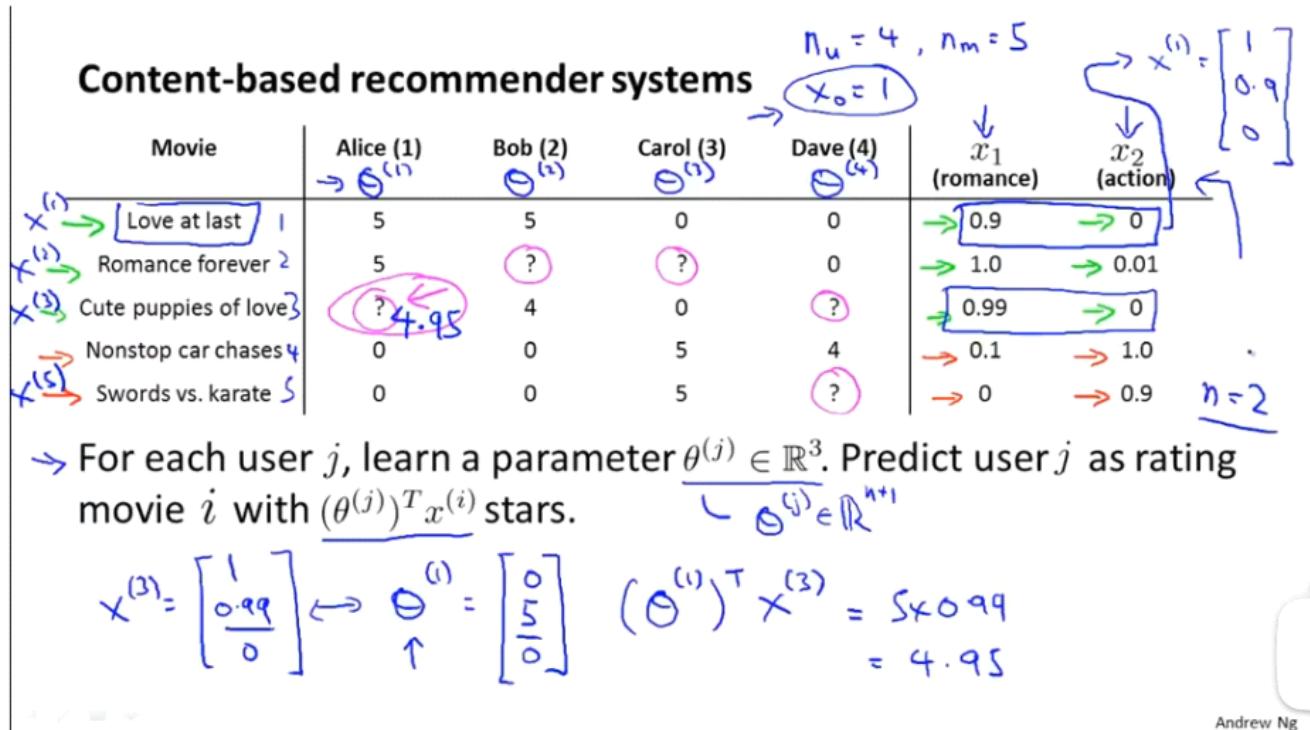
当用于 j 给电影 i 给出评级后，我们会得到 $y(i, j)$ ，它表示用户 j 对电影 i 所给出的评分，因此 $y(i, j)$ 是一个0-5的数字表示。

所以推荐系统的问题是给出了 $R(i, j)$ 和数据 $y(i, j)$ ，然后去查找那些没有评级的电影，并试图预测这些电影的评级，因此若我们想要开发出一个推荐系统，那我们的任务是想出一种学习算法，一个能

自启动为我们能填充这些缺失值的算法，这样我们就能看到，该用户还有哪些电影没看，并推荐电影给新用户，可以去预测什么样的用户会感兴趣的内容。

(2) 基于内容的推荐算法

2022年2月12日 21:16



如上图所示，我们有四位用户对五部电影做出的评价，其中有两个特征， x_1 表示电影中爱情的饱和度， x_2 表示电影中动作的饱和度，其中我们有 $n=2$ ，表示有两个特征，又有参数

$$x^{(i)} = \begin{pmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \end{pmatrix}$$

其中 $x_0^{(i)} = 0$ ， $x_1^{(i)}$ 为第 i 部电影的特征 x_1 的数值， $x_2^{(i)}$ 为第 i 部电影的特征 x_2 的数值。我们可以把每个用户的评价预测值看做是一个线性回归问题，特别规定，对于每一个用户 j ，我们要学习参数向量 $\theta^{(j)} \in \mathbb{R}^{n+1}$ ，其中 n 不包括 x_0 。然后我们要预测用户 j 评价电影 i 的值，也就是参数 $\theta^{(j)}$ 与 $x^{(i)}$ 的内积。我们有对用户 Alice 预测它对电影 Cute puppies of love 的评级，我们有电影特征

$$x^{(3)} = \begin{pmatrix} 1 \\ 0.99 \\ 0 \end{pmatrix}$$

和Alice的参数向量

$$\theta^{(1)} = \begin{pmatrix} 0 \\ 5 \\ 0 \end{pmatrix}$$

最终，我们有对于电影的预测评级为

$$(\theta^{(1)})^T x^{(3)} = 5 * 0.99 = 4.95$$

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

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

$\underline{m^{(j)}}$ = no. of movies rated by user j

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

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

对于用户 j , 电影 i , 我们预测评级为 $(\theta^{(j)})^T(x^{(i)})$

$m^{(j)}$ 表示用户 j 已经对电影进行评级

为了去学习参数 $\theta^{(j)}$

我们有最小化代价函数

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

我们去掉项 $\frac{1}{m^{(j)}}$, 去掉之后, 这并不影响我们最小化代价函数的结果, 所以我们有简化后的代价函数。

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

其中 $\frac{\lambda}{2} \sum_{k=1}^n (\theta_k^{(j)})^2$ 为正则项。

Optimization objective:

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

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

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

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

优化目标：

我们去学习参数 $\theta^{(j)}$, 故我们有最小化目标函数

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

我们不仅要学习单个用户的参数，而是要学习所有用户的参数，故有公式。

对于参数 $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)}$:

我们有公式

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

其中从 j 到 n_u 表示从参数 $\theta^{(1)}, \theta^{(2)}$ 到 $\theta^{(n_u)}$, 有 n_u 个用户.

这个式子表示对所有目标进行求和，并且最小化，即最小化这个代价函数。

Optimization algorithm:

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

↙

$J(\theta^{(1)}, \dots, \theta^{(n_u)})$

Gradient descent update:

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

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \frac{\lambda \theta_k^{(j)}}{2} \right) \quad (\text{for } k \neq 0)$$

$\frac{\partial}{\partial \theta_k^{(j)}} J(\theta^{(1)}, \dots, \theta^{(n_u)})$

对这个函数进行最小化，即需要进行梯度下降法，求出函数
 $J(\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)})$ 。

(3) 协同过滤

2022年2月12日 22:39

这一节，我们将要学习特征学习，这种算法能够自行学习所要使用的特征。

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)	$x_0 = 1$
Love at first	5	5	0	0	1.0	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	?	[?]	[?]	

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

$\Theta^{(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

$(\Theta^{(1)})^T \times^{(1)} \approx 5$
 $(\Theta^{(2)})^T \times^{(1)} \approx 5$
 $(\Theta^{(3)})^T \times^{(1)} \approx 0$
 $(\Theta^{(4)})^T \times^{(1)} \approx 0$

这里的数据集，我们假定每一部电影都一些人来评价并告诉我们，这部电影的爱情成分是多少(x_1)，又包含多少动作成分(x_2)，但是很难使得每个人的看完一部电影告诉你这部电影包含多少爱情成分，多少动作成分，而且通常你还想要这两个特征之外的其它特征，那么该怎样得到这些特征呢？

从另一个角度看，假设我们有一个数据集，但我们不知道 x_1, x_2 的特征值是多少，比如我们得到一些关于电影的数据，不同的用户对电影的评分，也不知道每部电影的爱情指数以及每部电影的动作指数，所以我们把特征 x_1, x_2 都换成问号。现在我们改变一下假设，假设我们采访了每一位用户，而且每一位用户都告诉我们它们喜欢爱情电影的程度以及喜爱动作电影的程度，这样Alice就有了对应的参数 $\theta^{(1)}$ ，Bob的对应参数 $\theta^{(2)}$ ，Carol的对应参数 $\theta^{(3)}$ ，Dave的对应

参数 $\theta^{(4)}$ 。从参数中可以看出Alice和Bob比较喜欢爱情片，比较讨厌动作电影，而Carol和Dave比较喜欢动作电影，讨厌爱情电影。这说明参数 $\theta^{(j)}$ 表示了他们对不同题材电影的喜爱程度。我们从用户上得到参数 θ ，那么我们理论上就能推测出每部电影的 x_1 和 x_2 的值，假设我们观察电影1，所以电影1对应的是特征向量 $x^{(1)}$ ，我们从参数 θ 可以看出Alice和Bob比较喜欢爱情电影，而Caro和Dave比较喜欢动作电影，因此我们可以从以上特征推测出对于电影1，为爱情电影的概率比较大，而动作电影的概率比较小，所以我们可以猜测电影1的特征 x_1, x_2 分别为1.0, 0.0。其中这个的推导过程如下，我们有各个用户对这部电影的评分以及用户参数，故有公式。我们有Alice对于电影1的评分公式：

$$(\theta^{(1)})^T x^{(1)} \approx 5$$

Bob对于电影1的评分公式：

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

Caro对于电影1的评分公式：

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

Dave对于电影1的评分公式：

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

其中 $x^{(1)} = \begin{bmatrix} 1 \\ 1.0 \\ 0 \end{bmatrix}$ ，其中 x_0 为偏置单元， $x_0=1$ 。

Optimization algorithm

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

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

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

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

我们有参数向量 $\theta^{(1)}, \dots, \theta^{(n_u)}$, 去学习 $x^{(i)}$, 所以有优化目标:

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

这个公式表示我们要把对于电影i, 已经对电影i的进行评分过的用户的值 $y^{(i,j)}$ 与我们预测的用户j对电影i的评分 $(\theta^{(j)})^T x^{(i)}$ 之间的最小差值, 其中 $\frac{\lambda}{2} \sum_{k=1}^n (x_k^{(i)})^2$ 为正则化项。

我们有参数向量 $\theta^{(1)}, \dots, \theta^{(n_u)}$, 去学习 $x^{(1)}, \dots, x^{(n_m)}$, 所以有优化目标:

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

即使得所有用户对于用户已经评价过的电影的评分与推荐系统的预测评分之间的差值最小。

Collaborative filtering

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

$\sigma^{(i,j)}$
 $y^{(i,j)}$

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

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

协同过滤：

对于给予参数 $x^{(1)}, \dots, x^{(n_m)}$, 我们能评估参数 $\theta^{(1)}, \dots, \theta^{(n_u)}$.

对于给予参数 $\theta^{(1)}, \dots, \theta^{(n_u)}$, 我们能评估参数 $x^{(1)}, \dots, x^{(n_m)}$.

即通过参数 $x^{(1)}, \dots, x^{(n_m)}$, 我们能学习参数 $\theta^{(1)}, \dots, \theta^{(n_u)}$.

通过参数 $\theta^{(1)}, \dots, \theta^{(n_u)}$, 我们能学习参数 $x^{(1)}, \dots, x^{(n_m)}$.

我们通过随机地猜取一些 θ 的值, 然后我们就能去学习参数 x ,
然后再通过 x 去更好的学习 θ , 如此反复迭代可以使得参数 θ 和 x 都可以得到一个不错的值。

(4) 协同过滤算法

2022年2月13日 14:57

Collaborative filtering optimization objective

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

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

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

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

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

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

$$\min_{\substack{x^{(1)}, \dots, x^{(n_m)} \\ \theta^{(1)}, \dots, \theta^{(n_u)}}} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$$

协同过滤的优化目标

我们有公式如下：

我们有参数向量 $x^{(1)}, \dots, x^{(n_m)}$, 去学习 $\theta^{(1)}, \dots, \theta^{(n_u)}$, 所以有优化目标:

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

我们有参数向量 $\theta^{(1)}, \dots, \theta^{(n_u)}$, 去学习 $x^{(1)}, \dots, x^{(n_m)}$, 所以有优化目标:

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

即通过 θ 解得 x , 再通过 x , 解得 θ , 如此反复迭代, 解得一个最优值, 但实际上, 有一个更加有效率的算法, 让我们不再需要不停的计算 θ 和 x , 而是能够将 θ 和 x 同时计算出来, 下面的算法就是将上面两个优化目标合在一起, 所以我们有如下公式。

最小化 $x^{(1)}, \dots, x^{(n_m)}$ 和对 $\theta^{(1)}, \dots, \theta^{(n_u)}$ 进行评估, 定义新的优化目标 J 是一个关于特征 x 和参数 θ 的代价函数。

$$\begin{aligned}
& J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) \\
&= \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2 \\
&+ \frac{\lambda}{2} \sum_{j=1}^{n_m} \sum_{i=1}^n (x_k^{(j)})^2 \quad (3)
\end{aligned}$$

即 $\min_{x^{(1)}, x^{(2)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$

其中式(1)的第一项表中第一个求和运算即j从1到 n_u 表示所有用户j的总和，又求和项 $i: r(i,j) = 1$ 表为被用户j所有评分过的电影的总和，其中 $r(i,j) = 1$ 表示为每项对应被某一用户评分过的某一电影，关于J的求和的意思是，对于每个用户，对该用户评分的所有电影的评分求和，式子

$$\sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)})^2$$

表示从1到 n_u 的所有用户，对于其中第j个用户，他对于已经评分过的所有的电影，推荐系统预测的值与用户真实的评价的差值之和。

而式子

$$\sum_{j=1}^{n_m} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)})^2$$

表示对于每部电影i将所有对它评分过的用户j求和。

这两个求和运算都是对所有 $r(i,j) = 1$ 的i,j求和，就是对所有有评分的用户-电影对进行求和，因此这两个式子合起来就是式子

$$\sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)})^2$$

所以我们求解式子(3)就能同时解出 θ 和x。

按照惯例我们是 $x_0=1$, $x \in R^n$, 而不是之前的 $x \in R^{n+1}$ 。同样的参数 θ 也具有同样的维度，故有 $\theta \in R^n$ 。

注：我们放弃这个惯例是因为我们现在是在学习所有的特征，我们没有必要将一个特征值硬编码为1，因为如果算法真的需要一个特征永远为1，它可以选择靠自己去获得1这个数值，如果这算法想要的话，它可以将特征值 x_1 设为1，所以没必要将去掉1，这个特征定死，现在算法有了灵活性去自行学习。

注2：上面式子中最小化参数 x 和最小化参数 θ 的意义可以参考上几节的内容。

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 $j = 1, \dots, n_u, i = 1, \dots, n_m$:

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

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

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

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

协同过滤算法：

第一步：我们将会把 x 和 θ 初始化为小的随机值。

第二步：使用梯度下降法去最小化函数 $J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$

第三步：对于用户有参数 θ 和已有电影评分 x ，我们可以预测评分 $\theta^T x$

(5) 矢量化: 低秩矩阵分解

2022年2月13日 17:17

本节将介绍协同过滤算法的向量化实现以及该算法可以实现的一些功能，比如说给定一个商品，你可以找到与之相关的商品，比如一个用户最近一直寻找一个商品，有没有一些相关的其他商品能够推荐给这个用户。

Collaborative filtering				
Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
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	?

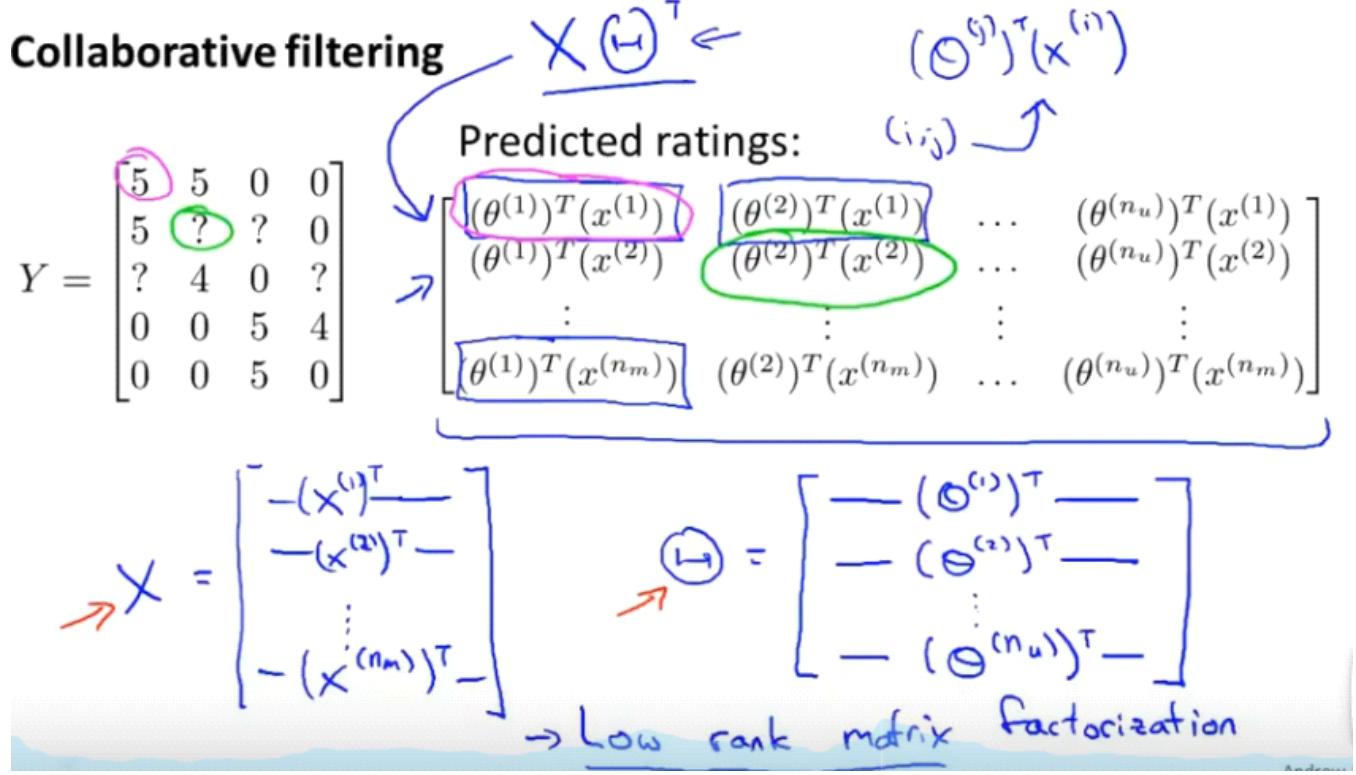
$n_m = 5$
 $n_u = 4$

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

$\uparrow \quad \uparrow$
 $y_{(i,j)}$

首先这是我们的数据集，包括五部电影，我们所要做的是得到所有用户对所有电影的评价，然后把它们分组写入矩阵，我们这里有五部电影和四位用户共同构成矩阵Y

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



给定矩阵 Y 包含所有已知的矩阵，我们写出这个算法的所有预测评分：

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

现在对于这个矩阵，我们可以有一个比较简单的或者向量化的方法来写出它们，我们有

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

我们有矩阵 X ，把每部电影看作一个样本，将不同电影的所

有属性都按行写入矩阵。我们又有

$$\theta = \begin{bmatrix} -(\theta^{(1)})^T & - \\ -(\theta^{(2)})^T & - \\ \vdots & \\ -(\theta^{(n_u)})^T & - \end{bmatrix}$$

我们有矩阵 θ ，我们把用户参数向量按行写入，得到矩阵 θ 。这样令 $X * \theta$ 就可以得到预测矩阵，我们把这个过程叫做低秩矩阵分解。

Finding related movies

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

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

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

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

5 most similar movies to movie i :

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

寻找相关电影问题：

具体来说，对于每个商品*i*，比如电影*i*我们已经学到一个属性向量 $x^{(i)}$ ，当你学习某一组特征时，你之前并不知道该选取哪些不同的特征，但是如果你运行这个算法的一些特征，将捕捉到有关电影和商品的重要方面，这些重要方面将导致一些用户喜欢某些电影，导致一些用户喜欢另一些电影，可能你最后学习到一些特征，比如 x_1 表示浪漫爱情， x_2 表示动作，在你学习完特征之后可能很难理解这些被学习到的特征，并对这些特征给出人类可以理解的解释，但是实际上，即使这些特征难以可视化人类难以

理解这些特征的含义，但是通常算法将学习一些有意义的特征，它们可以捕捉到一部电影的最重要的特征，这些特征将导致了你喜欢或不喜欢这部电影，现在再看下一个问题

比如你有一部电影i, 你想要找另一部电影j, 它可i相关联，为什么你要这么做呢？

假设你有一个用户正在浏览电影，它们正在看电影j, 那么在他们看完电影j后，推荐给他们哪一部电影比较合理呢？

我们可以权衡两部电影的相似度，比如电影i有一个特征向量 $x^{(i)}$, 如果能够找到另一部电影 $x^{(j)}$, 使得 $x^{(i)}$ 和 $x^{(j)}$ 之间的距离很小，即公式

$$\text{Small } ||x^{(i)} - x^{(j)}||$$

如你的用户正在看某个电影i, 如果你想要找五个与电影最，相似的电影，即推荐给用户五部相似的电影，我们所要做的使这些电影的特征向量与电影i的特征向量有最小距离。

(6) 均值归一化

2022年2月13日 17:47

均值归一化可以让协同过滤算法运行的更好。

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	?	?

↓

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

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

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

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

Andrew Ng

我们有一个没有给任何电影评级的用户Eve，我们来看看协同过滤算法会做些什么，假设 $n=2$ ，我们要学习两个特征变量，我们要学习出用户参数 $\theta^{(5)} \in R^2$ ，这是一个二维向量，这个是 n 维向量非 $n+1$ 维向量，我们要学习Eve的参数 $\theta^{(5)}$ ，我们来看公式

$$\sum_{(i,j): r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 \quad (1)$$

用户Eve没有给任何电影打分，所以对于Eve来说没有电影满足 $r(i,j) = 1$ ，所以上式公式完全不影响 $\theta^{(5)}$ ，所以式子

$$\frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2 \quad (2)$$

是唯一影响 $\theta^{(5)}$ 的，我们想要选一个向量 $\theta^{(5)}$ 使得最后的正则项，尽可能的小，即我们想要最小化式子

$$\frac{\lambda}{2} [(\theta_1^{(5)})^2 + (\theta_2^{(5)})^2]$$

如果上式为我们最小化的目标，那么我们最终得到

$$\theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

这是因为正则化项会让你的参数接近0，如果没有数据使得参数远离0，因为式子(1)不影响参数，所以参数 $\theta^{(5)}$ 最终会为0，所以当我们预测用户5要如何给电影打分时，我们有

$$(\theta^{(5)})^T x^{(i)} = 0$$

因此我们会预测到用户5给所有电影的评分都是零颗星。但是这个结果没有什么用，因为Eve给所有电影零分，我们还是没有任何比较好的办法推荐电影给她，因为所有的这些电影都会被Eve给出一样的预测评分，所以没有一部电影拥有高一点的评分，能让我们有电影推荐给她，所以这种结果不太好，均值归一化的思想可以让我们解决这个问题。

Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? & 2.5 \\ 5 & ? & ? & 0 & ? & 2.5 \\ ? & 4 & 0 & ? & ? & 2 \\ 0 & 0 & 5 & 4 & ? & : \\ 0 & 0 & 5 & 0 & ? & \end{bmatrix}$$

$$\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow Y =$$

$$\begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

For user j , on movie i predict:

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

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

User 5 (Eve):

$$\underline{\theta}^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\underbrace{(\underline{\theta}^{(5)})^T (\underline{x}^{(i)})}_{=0} + \boxed{\mu_i}$$

Andrew Ng

均值归一化：

我们有实际的评分矩阵

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

以及均值矩阵

$$u = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$$

我们观察一下这些电影的评分，然后减去均值，有矩阵

$$Y_{new} = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

我们把矩阵 Y_{new} 当作数据，来学习我们的 x 和参数 θ 。

对于用户j在电影i上的预测是

$$(\theta^{(j)})^T(x^{(i)}) + u_i$$

对于Eve，我们有

$$(\theta^{(5)})^T(x^{(i)}) + u_i$$

$$\theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

所以Eve的电影预测，

$$\begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$$

我们也可以尝试均值归一化的其它版本，可以对不同的列进行归一化使得它们的均值为0，而不是对行进行归一化

(1) 学习大数据集

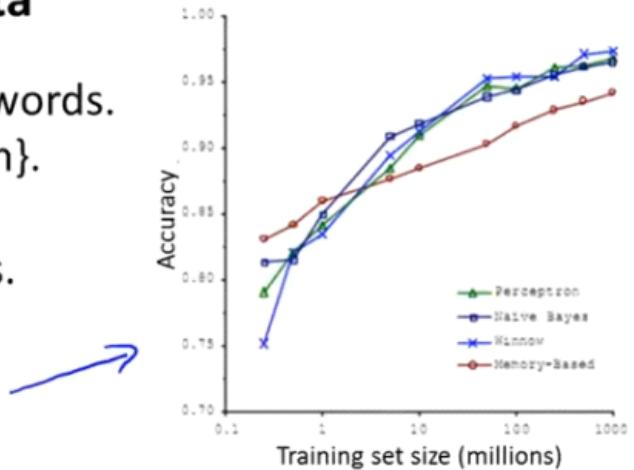
2022年2月14日 12:50

在这节中，我们将学习大规模机器学习以及处理大数据集的算法。

Machine learning and data

Classify between confusable words.
E.g., {to, two, too}, {then, than}.

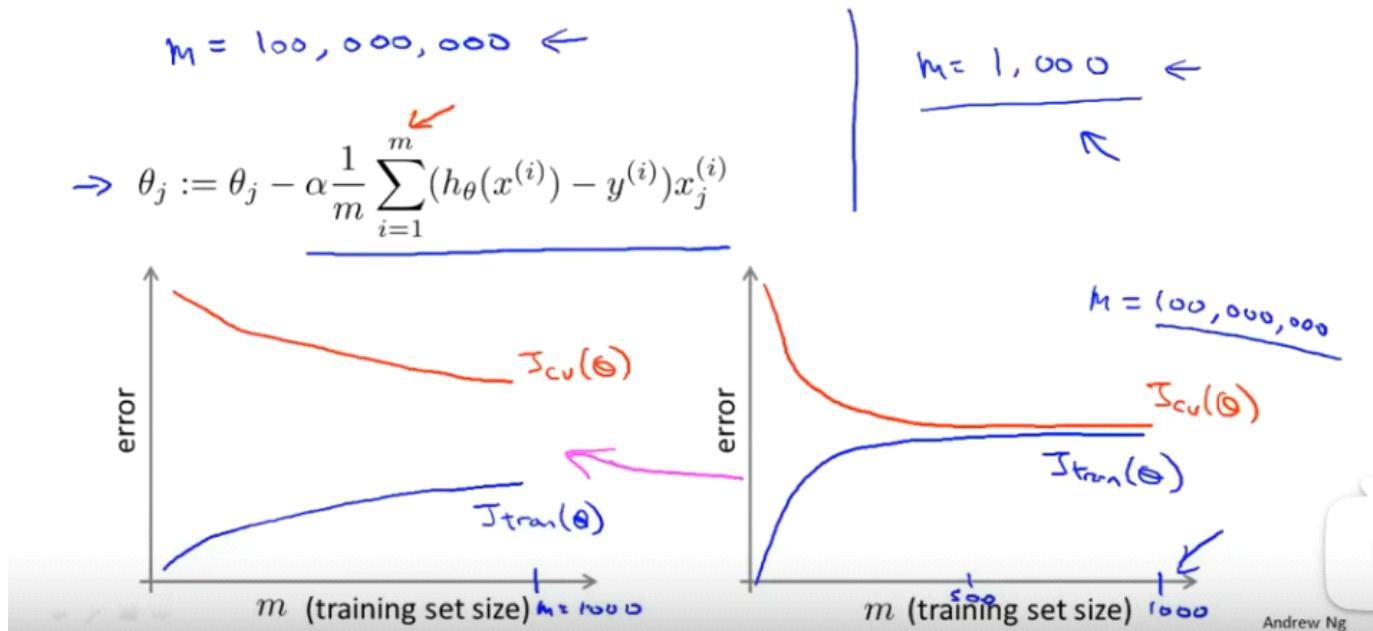
For breakfast I ate two eggs.



→ “It's not who has the best algorithm that wins.
It's who has the most data.”

我们为什么要学习大规模的数据，我们已知晓的一种获取高性能的机器学习系统的途径是采用低偏差的学习算法，并用大数据进行训练，一个我们之前已经了解过的例子，这个对易混淆单词进行分类的例子，在这个例子中，只要你用大量数据进行训练，它的效果看起来非常好，从类似的结果可以看出，在机器学习中，通常情况下决定因素往往不是最好的算法，而是谁的训练数据最多。

Learning with large datasets



不过大规模数据的训练，也有其问题，就是计算的问题，假定训练集大小为一亿，其中我们要计算公式

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

即我们需要对一亿项进行求和，计算导数项以及演算单步下降，因为计算超过一亿的代价太大了，我们需要寻找更加高效的算法去计算。

我们也可进行选择其中的1000项数据，然后使用算法，通过观察图像做出适当的调整。

(2) 随机梯度下降

2022年2月14日 13:18

随机梯度下降算法，可以适应更大的训练集，使得算法比普通梯度下降算法运行的要更快。

Linear regression with gradient descent

$$h_{\theta}(x) = \sum_{j=0}^n \theta_j x_j$$

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

Repeat {

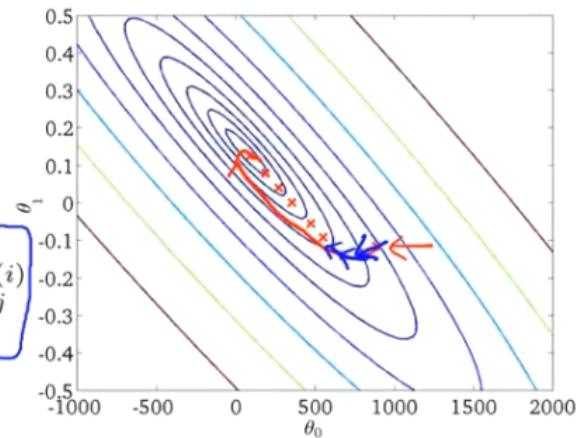
$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(for every $j = 0, \dots, n$)

}

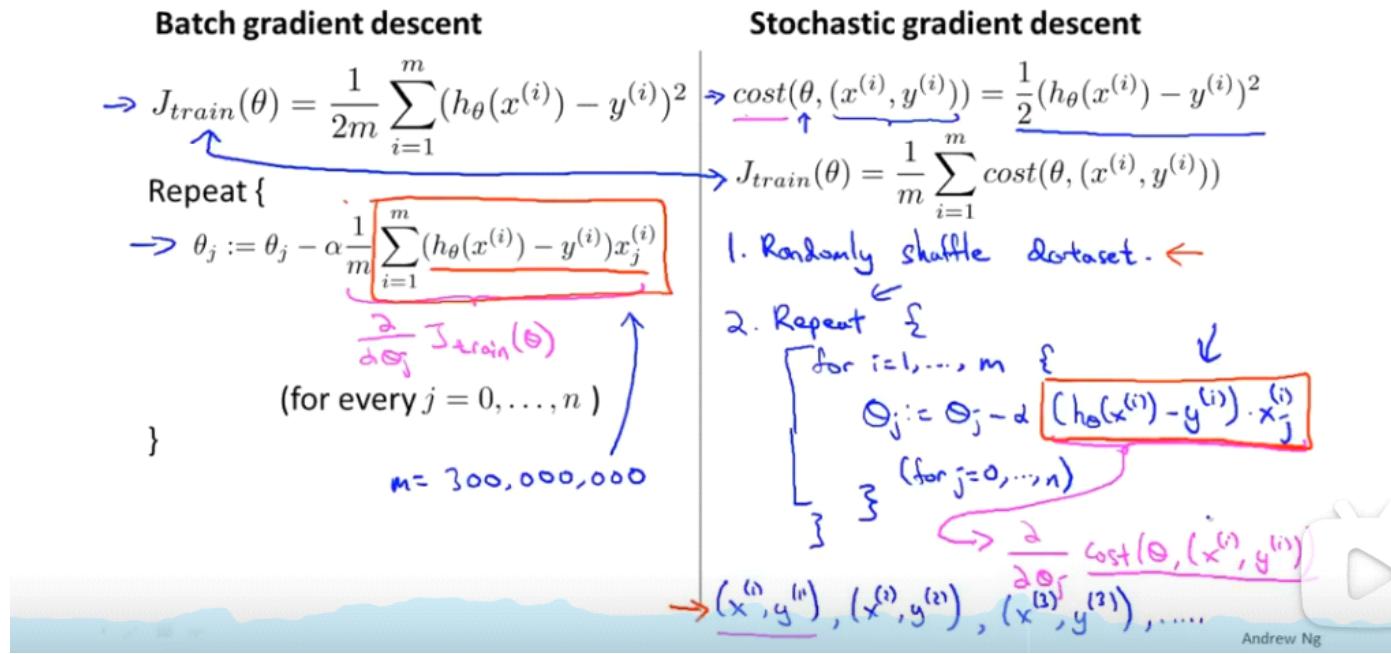
$$M = 300,000,000$$

Batch gradient descent



线性回归的梯度下降：

我们有假设函数和代价函数如上图所示，我们在运行梯度下降算法里，会从初始位置沿着红色一直迭代，直到迭代到一个全局最小的近似值为止，但问题是当数据集很大时，计算微分项，计算量会变得很大，因为需要对 m 个样本进行求和运算，假设你有三亿数据，你需要对这三亿数据的微分项进行相加运算，然后再次迭代，直到找到全局最小的相似值为止，这种方法会消耗很多时间。



批量梯度下降与随机梯度下降对比：

批量梯度下降法：

有代价函数如上图所示，批量梯度下降法是对所有样本进行求导，然后对于所有的样本的求导与原始的值相减之后再与所有的样本进行操作后所得到的值进行相加，再不断的进行迭代。

随机梯度下降法：

我们有代价函数：

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$

第一步：随机打乱所有数据

第二步：

```
Repeat {
For i=1,...,m {
For j=1,..,n{
 $\theta_j = \theta_j - \alpha (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$ 
}
}
```

}

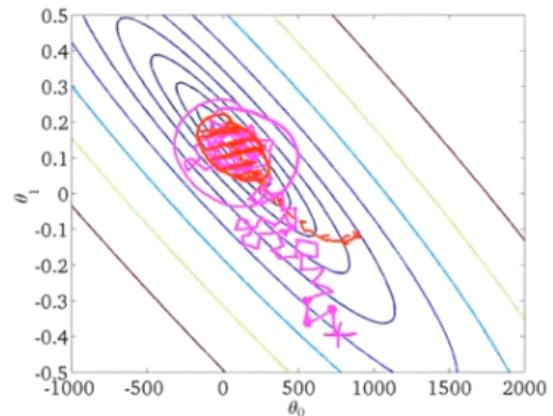
随机梯度下降算法是先对第一个样本 $(x^{(1)}, y^{(1)})$ 进行求导完成，再对第二个样本 $(x^{(2)}, y^{(2)})$ 进行求导，直到所有样本完成，然后再进行迭代进行最终逼近全局最小值。

随机梯度下降算法的过程：

Stochastic gradient descent

- 1. Randomly shuffle (reorder) training examples
- 2. Repeat {
 - $\underbrace{\text{for } i := 1, \dots, m \{}$
 - $\rightarrow \theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$
 - (for every $j = 0, \dots, n$)
 - }
}

 $\rightarrow \underline{m = 300,000,000}$



随机梯度下降算法如上图中的鲜红色线条进行运动，随机梯度下降算法的迭代次在1-10之间都是比较合理的。

(3) Mini-Batch梯度下降

2022年2月14日 16:35

我们在上节中讨论了随机梯度下降算法以及它比梯度下降法更快的原因，我们将讲述另一种算法Mini-Batch梯度下降算法，它有时候会比随机梯度下降算法要更快。

Mini-batch gradient descent

→ Batch gradient descent: Use all m examples in each iteration

→ Stochastic gradient descent: Use 1 example in each iteration

Mini-batch gradient descent: Use b examples in each iteration

$$\begin{aligned} b &= \text{mini-batch size.} & b &= 10. & 2 - 100 \\ \text{Get } & b = 10 \text{ examples} & & (x^{(i)}, y^{(i)}), \dots, (x^{(i+9)}, y^{(i+9)}) \\ & \rightarrow \theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) \cdot x_j^{(k)} \\ & \quad i := i + 10 \end{aligned}$$

Mini-batch梯度下降算法：

批量梯度下降算法：在每次迭代中使用所有样本

随机梯度下降算法：在每次迭代中使用一个样本

Mini-Batch梯度下降算法：在每次迭代中使用 b 个样本

Mini-Batch梯度下降算法与批量梯度下降算法类似，只不过它会选取 B 个样本来进行迭代，介于批量梯度下降算法与随机梯度下降算法之间，对于Mini-Batch而言 B 在2到100都是一个比较合理的值。

Mini-batch gradient descent

Say $b = 10$, $m = 1000$.

$\rightarrow b$ examples
 $\rightarrow 1$ example

Repeat {

Vectorization

\rightarrow for $i = 1, 11, 21, 31, \dots, 991$ {

$\rightarrow \theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$
(for every $j = 0, \dots, n$)

}

}

$$m = \underbrace{300, 000, 000}_{\uparrow}$$

$$b = \frac{10}{\uparrow}$$

图中是B为10的一个例子，Mini-batch在数据存取和求导的过程中使用向量化，进行并行计算，这样可以加快计算速度，而Mini-batch的一个缺点是需要确定参数B的大小。

(4) 随机梯度下降收敛

2022年2月14日 16:48

在运行随机梯度算法时如何保证高度过程已经完成，并且已经收敛到合适的位置呢？还有就是如何调整梯度下降中学习率 α 的值呢？

Checking for convergence

→ Batch gradient descent:

→ Plot $J_{train}(\theta)$ as a function of the number of iterations of gradient descent.

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

M = 300, 500, 800

→ Stochastic gradient descent:

$$\rightarrow cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$\Rightarrow (x^{(i)}, y^{(i)}) , (x^{(i+1)}, y^{(i+1)}) \dots$

→ During learning, compute $cost(\theta, (x^{(i)}, y^{(i)}))$ before updating θ using $(x^{(i)}, y^{(i)})$.

$\uparrow \uparrow$



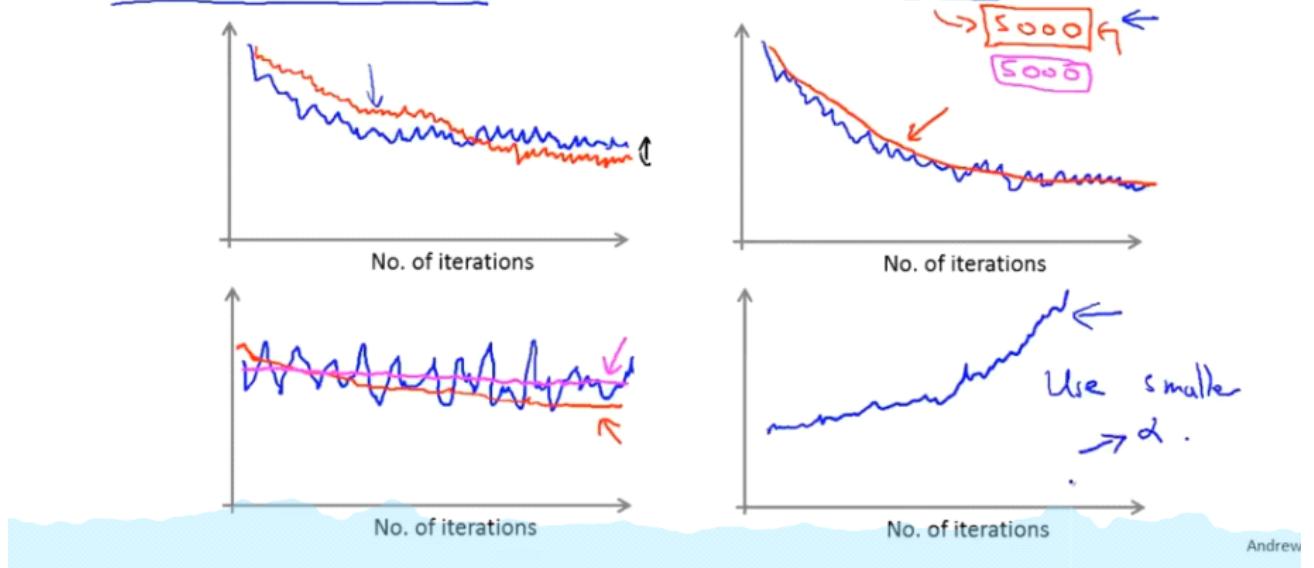
→ Every 1000 iterations (say), plot $cost(\theta, (x^{(i)}, y^{(i)}))$ averaged over the last 1000 examples processed by algorithm.

Andrew Ng

我们在批理梯度算法中，可以对每次梯度下降的迭代进行图像的绘制，然后我们对图像进行观察。而在随机梯度下降中，我们可以以每1000个样本(1000次迭代)为样例，对每1000个样本进行图像的绘制并观察是否收敛。

Checking for convergence

Plot $\text{cost}(\theta, (x^{(i)}, y^{(i)}))$, averaged over the last 1000 (say) examples



假设我们已经画出了前1000组样本的cost函数的平均值，由于它们只是前1000组样本的平均值，因此看起来会有很多的噪声，它可能不是每一步迭代都在下降。

图1：

蓝色线条，一个不错的下降过程，可以看出代价函数的值在下降，从这幅图像中看出代价函数已经开始收敛了。

红色线条——蓝色线条的更小学习率

线条线条的学习效率开始变得缓慢，所以代价函数也开始变得缓慢，因为使用了更小的学习率，所以它让结果收敛到了一个更好的值。

图2：

使用随机梯度下降算法，

蓝色线条为1000个样本的均值下降过程

红色线条为5000个样本的均值下降过程，缺点是每隔5000个样本才会得到一个点，因此图像显示比较缓慢。

图3：

有时运行随机梯度下降算法，会遇到如下情况

蓝色线条：可以看到代价函数没有丝毫的减小，看起来算法没有

在学习，若我们增加更多的样本，如1000变成5000，可能会变成红色线条。

红色线条：线条在缓慢下降，代价函数在正逐渐减小。

粉色线条：线条出现这种情况，代表算法需要调整。

图4：

蓝色线条：线条在正逐渐向上，正逐渐发散，代表我们需要使用更小的 α 学习率。

Stochastic gradient descent

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_{\theta}(x^{(i)}) - y^{(i)})^2$$

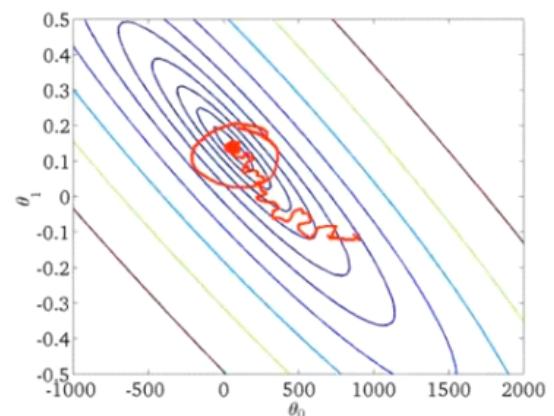
$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m cost(\theta, (x^{(i)}, y^{(i)}))$$

1. Randomly shuffle dataset.

2. Repeat {

```

    for i := 1, ..., m      {
         $\theta_j := \theta_j - \alpha(h_{\theta}(x^{(i)}) - y^{(i)})x_j^{(i)}$ 
        (for j = 0, ..., n)
    }
}
```



Learning rate α is typically held constant. Can slowly decrease α over time if we want θ to converge. (E.g. $\alpha = \frac{\text{const1}}{\text{iterationNumber} + \text{const2}}$) $\alpha \rightarrow 0$

关于学习速率 α :

我们已经知道了，当运行随机梯度下降时，算法会从某个点开始，然后曲折的到达最小值，但它不会完全收敛，而是一直在最小值附近徘徊，因此得到的最终参数是一个接近全局最小值的数，而不是真正的最小值，在大多数，随机梯度下降算法的典型应用中学习速率 α 通常是一个不变的常量，如果想要随机梯度下降算法更好的收敛全局最小值，我们可以做的是，让学习速率 α 的值随时间变化逐渐减小，所以一种典型设置 α 的值的公式如下：

$$\alpha = \frac{\text{const1}}{\text{iterationNumber} + \text{const2}}$$

iterationNumber 是指随机梯度下降的迭代次数。

因为学习率随着迭代次数正逐渐减小，所以它能更小的收敛到全局最小值。

(5) 在线学习

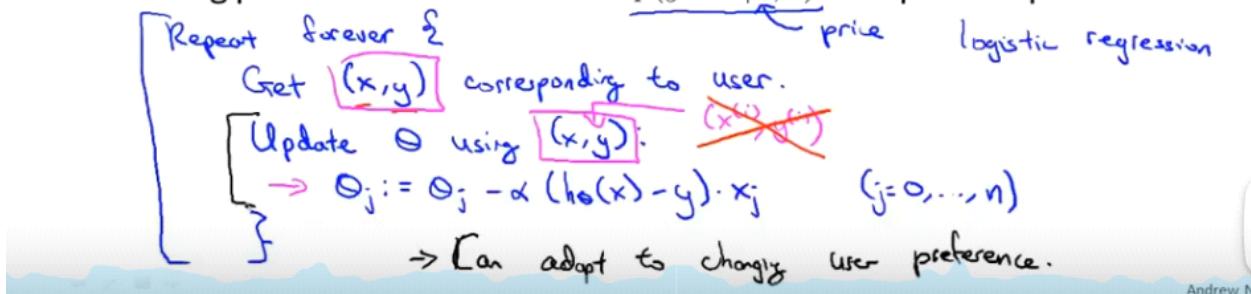
2022年2月14日 18:32

本节中将会讨论到一种新的大规模机器学习机制，若我们有一个不断进行网站的用户流所产生的连续数据流，我们就可以使用在线学习机制，从数据流中学习用户的偏好，然后使用这些信息在进行优化关于网站的决策。

Online learning

Shipping service website where user comes, specifies origin and destination, you offer to ship their package for some asking price, and users sometimes choose to use your shipping service ($y = 1$), sometimes not ($y = 0$).

Features x capture properties of user, of origin/destination and asking price. We want to learn $p(y = 1|x; \theta)$ to optimize price.



假定我们提供运输服务，用户们来询问把包裹从A地运到B地的服务，同时假定我们有一个网站，用户们登陆网站告诉我们他们想从哪里寄出包裹以及寄到哪里去，然后在我们的网站开出运输包裹的服务价格，然后我们给用户开出的这个价格，用户有时会接受这个运输服务，那么这个就是正样本 $y=1$, 有时他们会走掉 $y=0$ 。

假定我们想要一个学习算法来帮助我们，优化我们想给用户开出的价格，具体来说，假设我们获取了描述用户特点的特征，我们想要做的是用这些特征学习，他们将会选择我们的服务来运输包裹的机率，这些特征中包含了我们开出的价格，如果我们可

能估计出在每种价格下用户选择使用我们服务的概率，那么我们可以选择一个价格使得用户有很大的可能性选择我们网站，并且同时能够保证一个合适的回报，保证我们能获得合适的利润，所以如果我们可以学习在任何给定价格和其他特征下 $y=1$ 的概率，我们就可以利用这一些信息，在新用户来的时候选择合适的价格，所以为了获得 $y=1$ 的概率的模型，我们能做的是用logistic回归或神经网络或其它一些类似的算法，现在我们考虑logistic回归。

假定我们有一个连续运行的网站，以下是在线学习算法所做的操作

Repeat {

Get $(x^{(i)}, y^{(i)})$ corresponding to user. // 我们从网站中获取了一个用户的相关信息， $x^{(i)}$ 表示起始地和目的地以及服务价格， $y^{(i)} = 1$ 表示使用了我们的运输服务， $y^{(i)} = 0$ 则没有

Update θ using $(x^{(i)}, y^{(i)})$ // 我们使用 $x^{(i)}, y^{(i)}$ 来更新参数 θ

$\theta_j = \theta_j - \alpha(h_{\theta}(x^{(i)}) - y^{(i)})x_j^{(i)}$ // 我们一个样本，一个样本的更新
($j=0, \dots, n$)

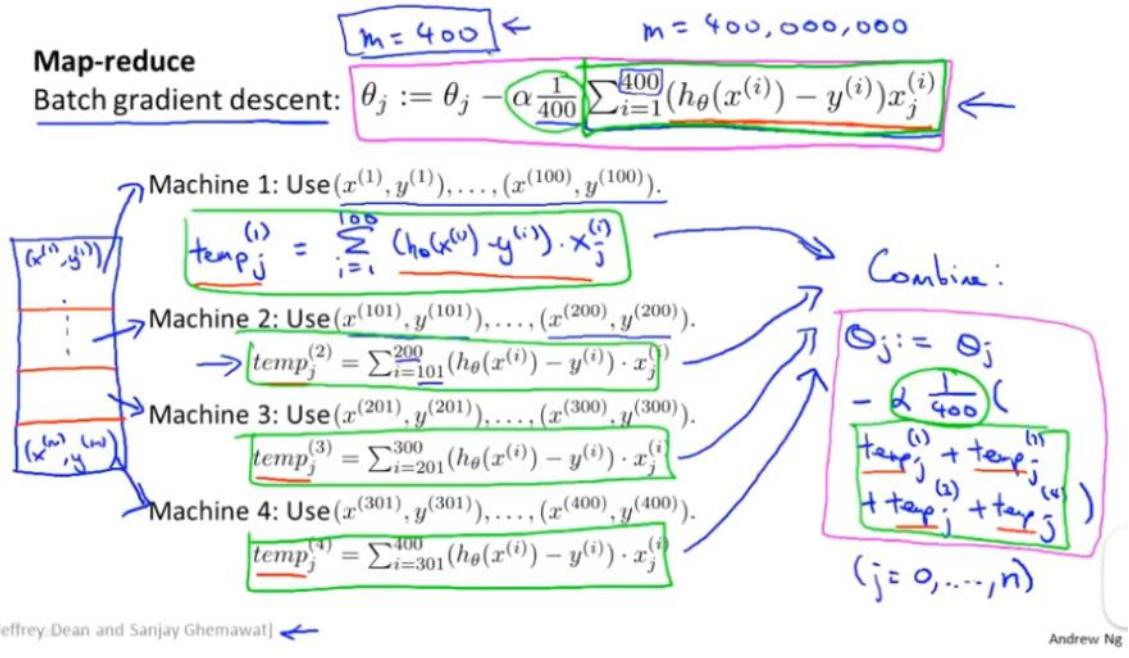
}

这种算法可以很好的适用于不断变化的用户数量的规模。

(6) 减少映射与数据并行

2022年2月14日 18:56

减少映射与数据并行可以让学习算法应用于随机梯度不能解决的更大规模的问题。



映射减少：

我们有批量梯度下降

$$\theta_j = \theta_j - a \frac{1}{400} \sum_{i=1}^{400} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

我们可以把数据集分成四个部分，每个部分运算100个样本数，都我们有

Machine 1: 使用 $(x^{(1)}, y^{(1)}), \dots, (x^{(100)}, y^{(100)})$

$$temp_j^{(1)} = \sum_{i=1}^{100} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Machine 2: 使用 $(x^{(101)}, y^{(101)}), \dots, (x^{(200)}, y^{(200)})$

$$temp_j^{(2)} = \sum_{i=101}^{200} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Machine 3: 使用 $(x^{(201)}, y^{(201)}), \dots, (x^{(300)}, y^{(300)})$

$$temp_j^{(3)} = \sum_{i=201}^{300} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

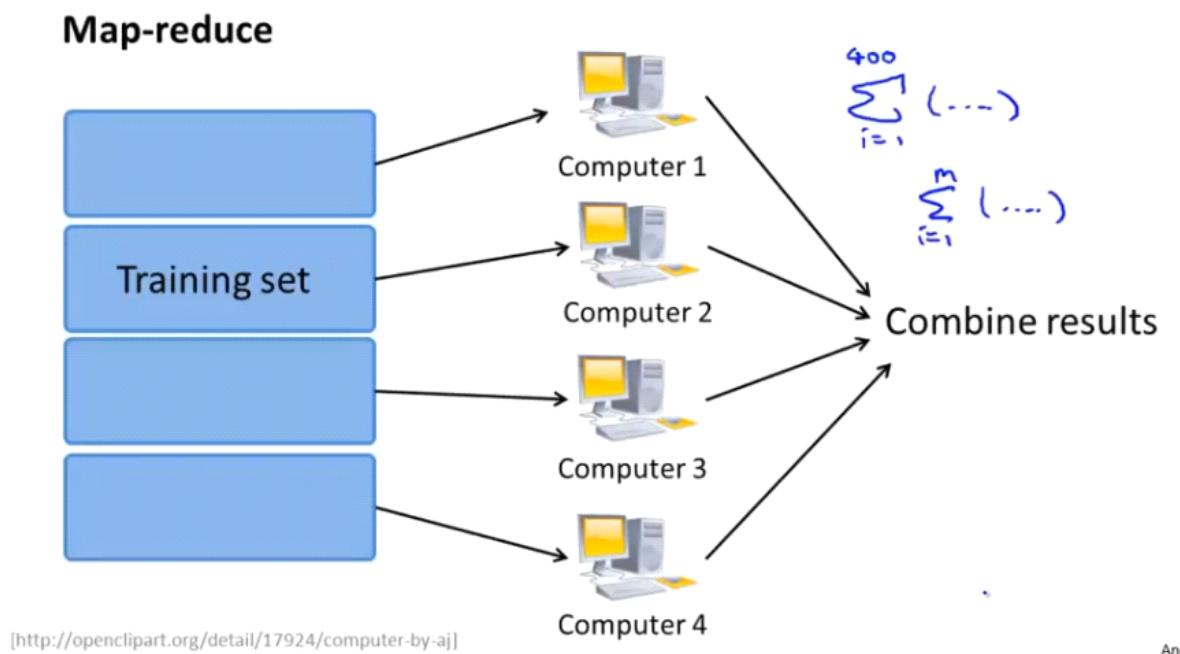
Machine 4: 使用 $(x^{(301)}, y^{(301)}), \dots, (x^{(400)}, y^{(400)})$

$$temp_j^{(4)} = \sum_{i=301}^{400} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

我们把数据通过这个台计算机进行运行，并在中央服务器进行汇总，有公式

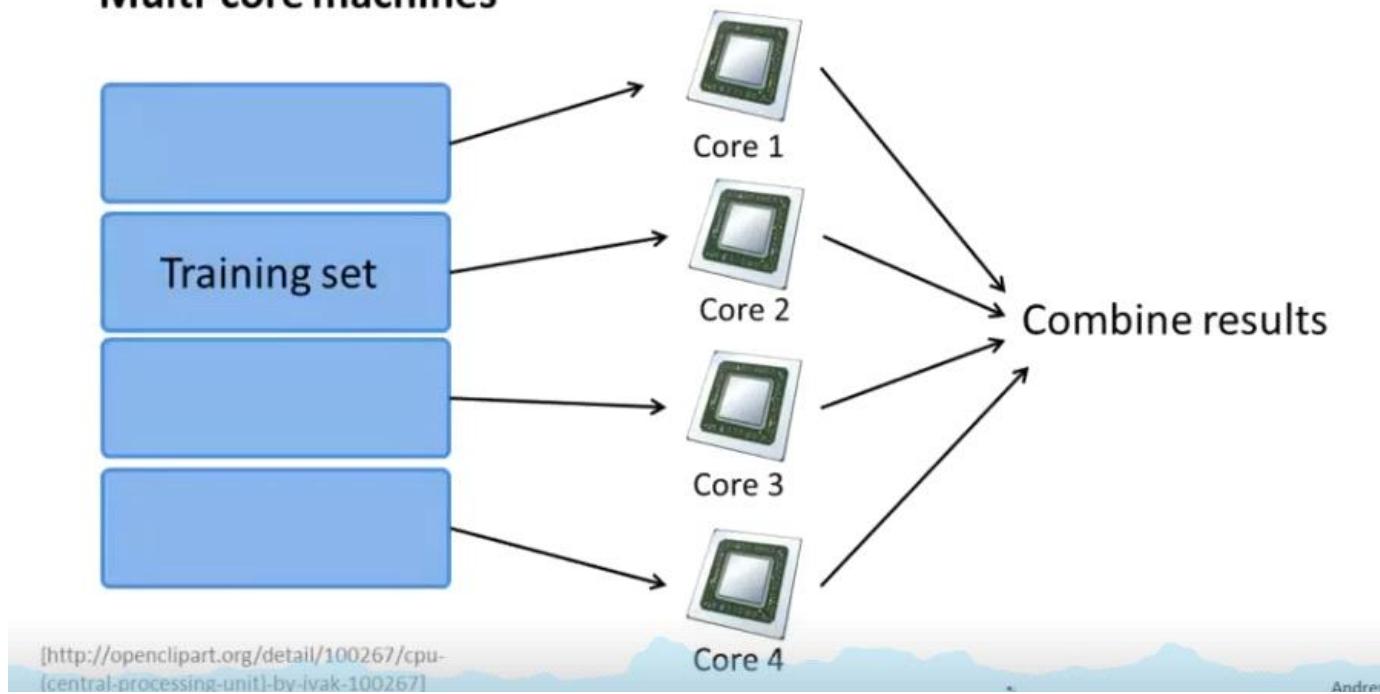
$$\theta_j = \theta_j - a \frac{1}{400} (temp_j^{(1)} + temp_j^{(2)} + temp_j^{(3)} + temp_j^{(4)})$$

这样做可以通过并行计算来加快计算速度。



映射减少与并行计算示意图。

Multi-core machines



在多核计算机中，我们可以使用多核来加速运算，如图我们计算机有四核，我们可以把数据集分成四个集合，然后再每一个核心运行一部分，最后再进行汇总。

(1) 照片OCR问题与机器学习流水线

2022年2月15日 18:25

The Photo OCR problem



照片OCR全称为照片光学字符识别，照片OCR问题注意的是如何让计算机读出图上中的文字信息，假设现在有一张照片，我们希望计算机能够读出照片的文字。

Photo OCR pipeline

→ 1. Text detection



→ 2. Character segmentation



→ 3. Character classification



Cleaning → Cleaning

Andre

照片OCR流水线

为了实现照片OCR，我们需要实现

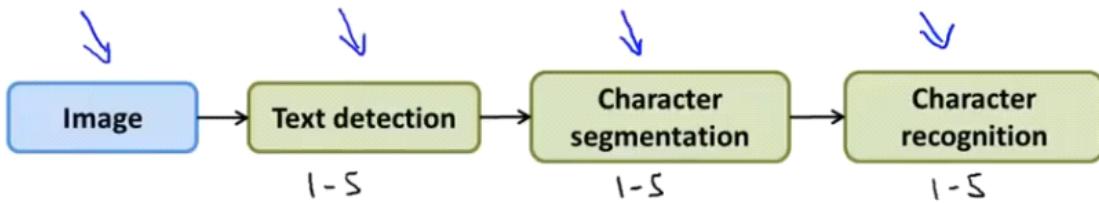
第一、扫描图像，并找出有文字的图像区域。

第二、我们得到一个字符矩形，我们需要这个区域分割成一个个的独立字符区域。

第三、分割好这些字符区域后，我们要用一个分类器，它会对这些可见的字符进行识别，一些复杂的分类器还具有纠正的功能。

像以上这个程序我们称之为机器学习流水线。

Photo OCR pipeline



我们有图像->文字检测->字符区域->字符确认四个阶段。我们有设计机器学习系统时，我们一个比较关键的问题是如何设计流水线的各个模块。

(2) 滑动窗口

2022年2月15日 18:38