



Table of Contents

1. [Introduction](#) 1.1
2. [机器学习](#) 1.2
 1. [第一课 介绍](#) 1.2.1
 2. [第二课 单一特征量的线性回归](#) 1.2.2
 3. [第三课 线性代数基本](#) 1.2.3
 4. [第四课 多特征量线性回归](#) 1.2.4
 5. [第五课 Octave介绍](#) 1.2.5
 6. [第六课 逻辑回归](#) 1.2.6
 7. [第七课 正规化](#) 1.2.7
 8. [第八课 神经网络](#) 1.2.8
 9. [第九课 神经网络学习](#) 1.2.9
 10. [第十课 机器学习的优化方法](#) 1.2.10
 11. [第十一课 系统设计](#) 1.2.11
 12. [第十二课 支持向量机\(SVM\)](#) 1.2.12
 13. [第十三课 聚类](#) 1.2.13
 14. [第十四课 降维](#) 1.2.14
 15. [第十五课 异常检测](#) 1.2.15
 16. [第十六课 推荐系统](#) 1.2.16
 17. [第十七课 大规模机器学习](#) 1.2.17
 18. [第十八课 应用举例: 图片 OCR](#) 1.2.18
 19. [第十九课 总结](#) 1.2.19
 20. [机器学习作业](#) 1.2.20
3. [NG机器学习 Python实践](#) 1.3
4. [Scikit-learn实践](#) 1.4
 1. [数据集](#) 1.4.1
 2. [决策树](#) 1.4.2
 3. [逻辑回归](#) 1.4.3
 4. [朴素贝叶斯] 1.4.4
 5. [GBDT] 1.4.5
 6. [随机森林] 1.4.6
 7. [交叉验证集] 1.4.7
 8. [网格搜索] 1.4.8
 9. [模型持久化] 1.4.9
 10. [Pipline] 1.4.10
5. [Pandas实践](#) 1.5
6. [Tesorflow实践 0.8](#) 1.6
 1. [环境搭建](#) 1.6.1
 2. [卷积神经网络 CNN](#) 1.6.2
7. [机器学习实践](#) 1.7
 1. [第二章 k邻近算法](#) 1.7.1
 2. [第三章 决策树](#) 1.7.2
 3. [第四章 朴素贝叶斯分类](#) 1.7.3
 4. [第五章 逻辑回归](#) 1.7.4
8. [Python](#) 1.8
 1. [语法](#) 1.8.1
 2. [相关环境搭建](#) 1.8.2
 3. [Python核心编程](#) 1.8.3
 1. [第二章 快速入门](#) 1.8.3.1
 2. [第三章 Python基础](#) 1.8.3.2

- 3. [第四章 Python对象](#) 1.8.3.3
 - 4. [第五章 数字](#) 1.8.3.4
 - 5. [第六章 序列:字符串、列表和元组](#) 1.8.3.5
 - 6. [第七章 映像和集合类型](#) 1.8.3.6
 - 7. [第八章 条件和循环](#) 1.8.3.7
 - 8. [第八章 条件和循环](#) 1.8.3.8
 - 9. [第九章 文件输入和输出](#) 1.8.3.9
 - 10. [第十章 错误和异常](#) 1.8.3.10
 - 11. [第十一章 函数和函数式编程](#) 1.8.3.11
 - 12. [第十二章 模块](#) 1.8.3.12
 - 13. [第十三章 类](#) 1.8.3.13
 - 14. [第十四章 执行环境](#) 1.8.3.14
 - 15. [第十五章 正则表达式](#) 1.8.3.15
 - 16. [第十六章 网络编程](#) 1.8.3.16
 - 17. [第十七章 网络客户端编程](#) 1.8.3.17
 - 18. [第十八章 多线程编程](#) 1.8.3.18
 - 19. [实战技巧](#) 1.8.3.19
- 4. [用Python做科学计算](#) 1.8.4.
 - 1. [数值计算要点1](#) 1.8.4.1
 - 2. [pandas要点](#) 1.8.4.2
- 9. [量化](#) 1.9.
 - 1. [统计套利](#) 1.9.1.
 - 1. [第一章 蒙特卡罗的谬误](#) 1.9.1.1
 - 2. [第二章 统计套利](#) 1.9.1.2
 - 3. [第三章 结构模型](#) 1.9.1.3
 - 10. [机器学习课题](#) 1.10.
 - 1. [课题一 基于神经网络的股票研究](#) 1.10.1
 - 2. [课题二 对自己的邮件进行自动分类整理](#) 1.10.2
 - 3. [课题三 最个性网络阅读器](#) 1.10.3
 - 4. [课题四 知识分享型机器人](#) 1.10.4
 - 11. [概率论](#) 1.11.
 - 1. [第一周](#) 1.11.1
 - 2. [第二周](#) 1.11.2
 - 3. [小结](#) 1.11.3
 - 4. [第九讲 随机变量](#) 1.11.4
 - 5. [第十讲 离散型随机变量](#) 1.11.5
 - 6. [第十一讲 分布函数](#) 1.11.6
 - 7. [第十三讲 连续分布和指数分布](#) 1.11.7
 - 8. [第十四讲 正态分布](#) 1.11.8
 - 9. [概率分布小结](#) 1.11.9
 - 10. [第十六讲 二元随机变量，离散型随机变量分布规律](#) 1.11.10
 - 11. [第十七讲 二元离散型随机变量边际分布率和条件分布律](#) 1.11.11
 - 12. [第十八讲 二元随机变量分布函数，条件分布函数](#) 1.11.12
 - 13. [第二十三讲 随机变量的独立性](#) 1.11.13
 - 14. [二元概率分布小结](#) 1.11.14
 - 15. [第二十九讲 数学期望的性质](#) 1.11.15
 - 16. [概率全息知识小结](#) 1.11.16
 - 17. [第三十五讲 依概率收敛和切比雪夫不等式](#) 1.11.17
 - 12. 线性代数 1.12
 - 13. 高等数学 1.13

Introduction

梦想的阶梯

这本书是希望能够以成体系的方式来介绍自己所做的项目和知识总结以及心得体会。

梦想是要有的万一实现了呢?

机器学习

NG的机器学习笔记

记录了NG机器学习的课程笔记。

第一课 介绍

Lesson1 介绍

- E:经验，也就是过去的数据，用作训练使用的数据。
- T:任务，也就是你要完成的是什么任务
- P:胜率，就是成功的概率

例子：点击“垃圾邮件”按钮过滤垃圾邮件，这里的E：找出那些是垃圾还不是垃圾的邮件的工作.T:完成过滤垃圾邮件的工作。P:是否是垃圾邮件的概率

两种算法

分别是监督学习和非监督学习。监督学习，就是我们指定策略让计算机去执行；而非监督学习是让计算机自己去学习如何分析。

监督学习

给定一组数据集，我们给出针对每一个数据给出正确的表现，通过预测接下来的结果。这种方式是监督学习，因为数据是有标签的。

在机器学习中可能会有基于多个attribute也就是属性进行预测的方式。例如癌症和肿块的大小以及年龄的大小，两个属性来进行预测。可能有些时候，你想要使用的是无穷多个数据维度进行分析和预测。

这里讨论两种方式，分别是回归还是分类。回归就是得到连续的结果；而分类得到的是离散的。

非监督学习

聚类

聚类，通过对一组数据的分析和分组，找出共性，然后进行分类。这是聚类算法。在股票方便的研究，我在想，可以分出强势股，弱势股等。

例如：新闻分类，基因组分类，计算机集群分类（将相互协作的计算机放在一起），客户数据分析，天文数据分析（例如星系诞生）。

鸡尾酒宴会算法

通过鸡尾酒宴会算法来进行语音识别与分析。通过将宴会上两个人通过两个麦克风的录制声音，能够通过非监督学习，将背景和人的声音分离开来。

鸡尾酒算法使用了SVD。

开发工具与环境

Octave 作为机器学习的原型开发会更加快速，更加方便。开发出原型之后，再转换成java, C, python等语言。

第二课 单一特征量的线性回归

Lesson2 单一特征量的线性回归

2-1

- Training Set: 训练集，表示所有训练的数据总和。
- m: 表示测试的数据集的总量。例如有47条数据作为数据集。
- x: 表示输入，例如目前预测是不同尺寸的房子的价钱，所以x表示的就是房子的大小。
- y: 表示输出，例如房子的最终价钱就是输出。

(x,y)表示一个训练的例子。对于第几个使用的是上标。标注在x的右上角，在markdown中很难表示所以就正常的写了。(x(i),y(i))>对应一组数据。

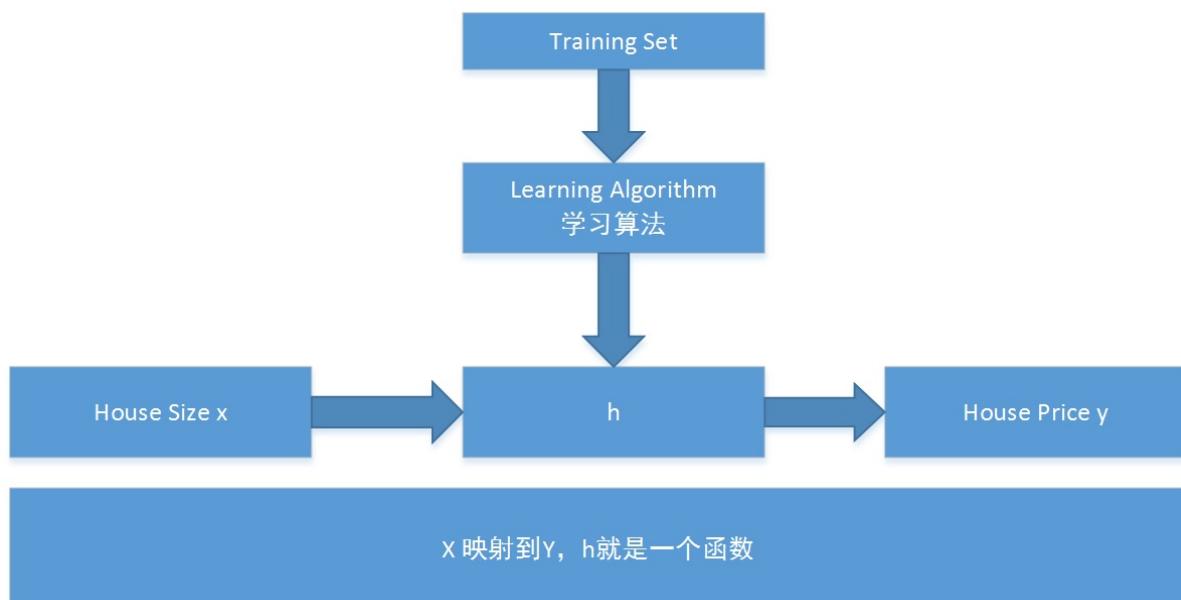
例如，在视频中的房价的案例：

房子大小 房价

2104	460
1416	232
1534	315

对于x来说，就是 $x(1) = 2104, x(2) = 1416, x(3)=1534$; 对于y来说, $y(1)=460, y(2)=232, y(3)=315$ 。所以对于(x,y)来说就组成了一个3*3的矩阵用作计算。

监督算法的流程



针对这个流程就是搞清楚h是什么，来设计一个学习算法。所以使用了另外一个字符: θ . 所以这个学习算法应该如下表述：

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

经常为了表述方便写成 $h(x) = \theta_0 + \theta_1 x$

对于房价的走势关系图，我们假设是一个线性的方程，也许这个并不是正确的，没关系，这作为一个基础会进行一点点的优化。这种线性函数的推测，称之为：线性回归

2-2

为了找到正确的 θ 的数值，所以需要一个成本函数来进行计算。原理上就是找到这样的 θ 使得， $h\theta(x)-y$ 的值最小。那么，这样的代价函数，使用的是平方差代价函数：

$$J(\theta_0, \theta_1) = (1/2m) \sum (h\theta(x^{(i)}) - y^{(i)})^2$$

$\text{minimize}(J(\theta_0, \theta_1))$ 就是代价函数的最小值，满足这样的 θ_0 和 θ_1 就是要求的。

$h\theta(x)$: 假设函数

J : 代价函数

2-3

代价函数式如何进行工作的呢？例如，简化上面的 $h(x) = \theta_1 x$ ，也就是 $\theta_0 = 0$ 这种简单情况。这时候的数据集类似如下：

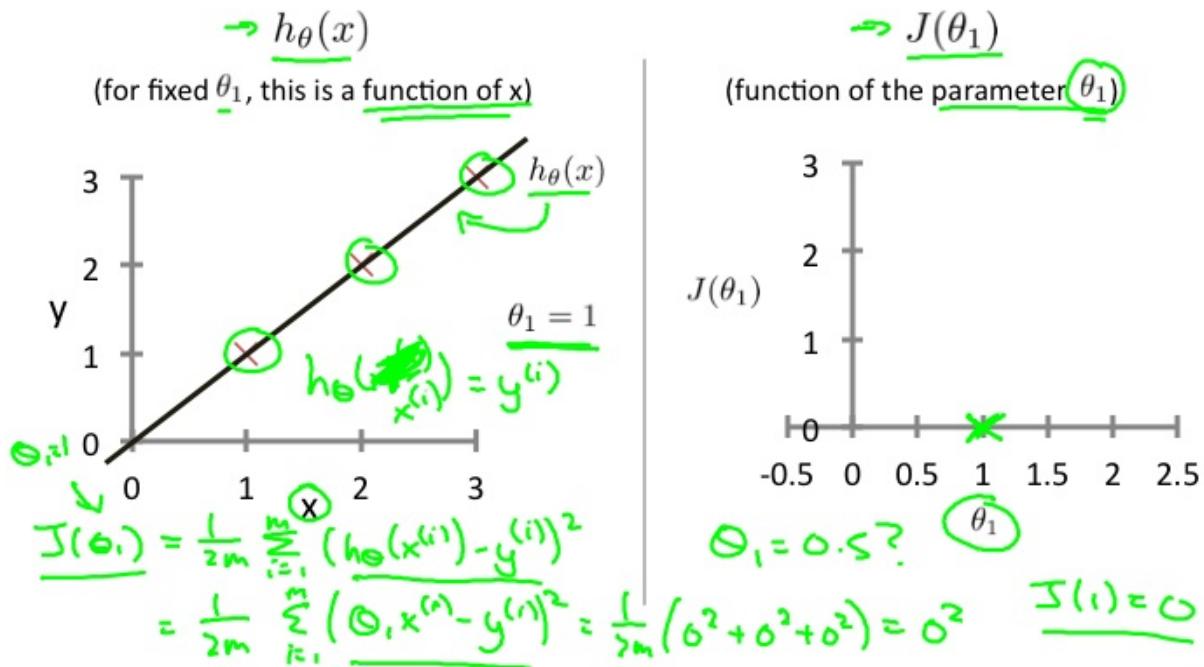
x y

1 1

2 2

3 3

在这里我们需要建立两个坐标系，分别是 x,y 坐标系 和 $J(\theta)$, θ 坐标系。如下图

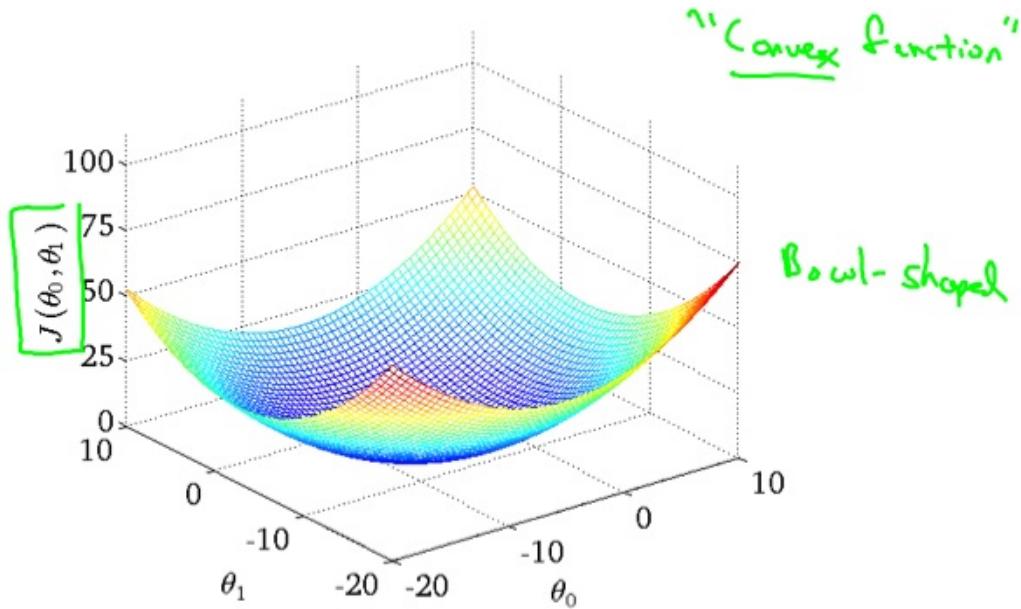


通过对 θ 的不断取值，来计算 J ，当取值到一定数量之后，找出其中的最小值，也就是 J 的最小值，就是我们要拟合计算出来的 θ 。

Andrew Ng

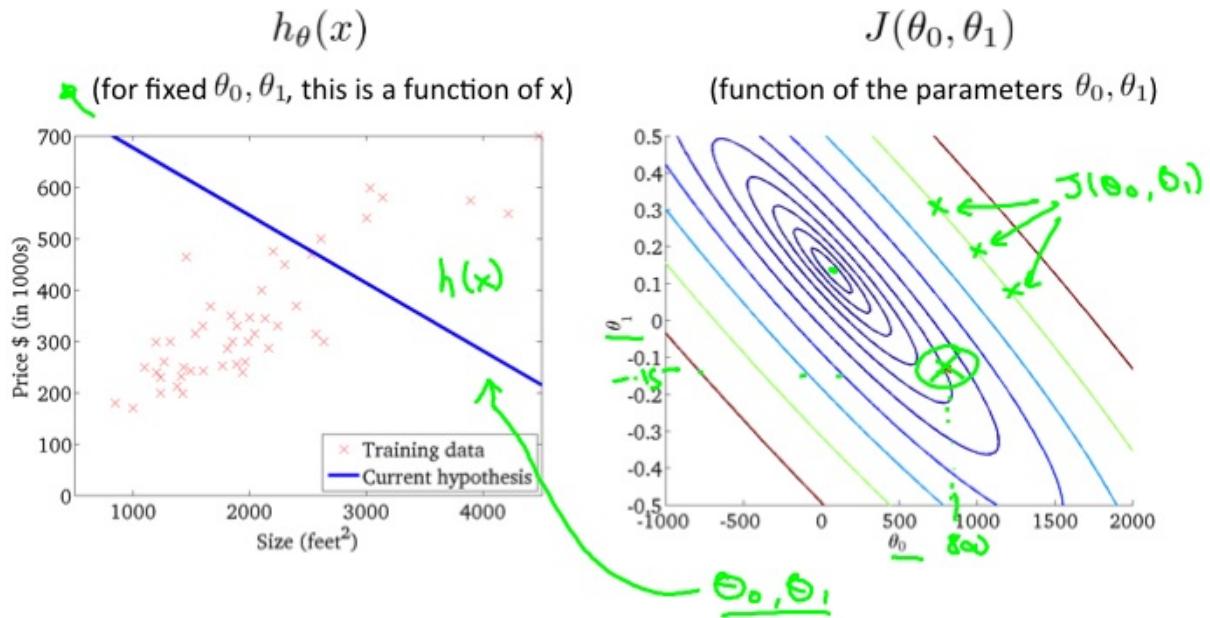
2-4

在前面的一节中通过二维的坐标就可以描述 J 和 θ_1 的关系，可是，如果还有 θ_0 ，那么就是三个变量，那该如何处理呢？对于我们来说最直观的想法自然是计算一个三维曲面。大概会是下面的样子。



Andrew Ng

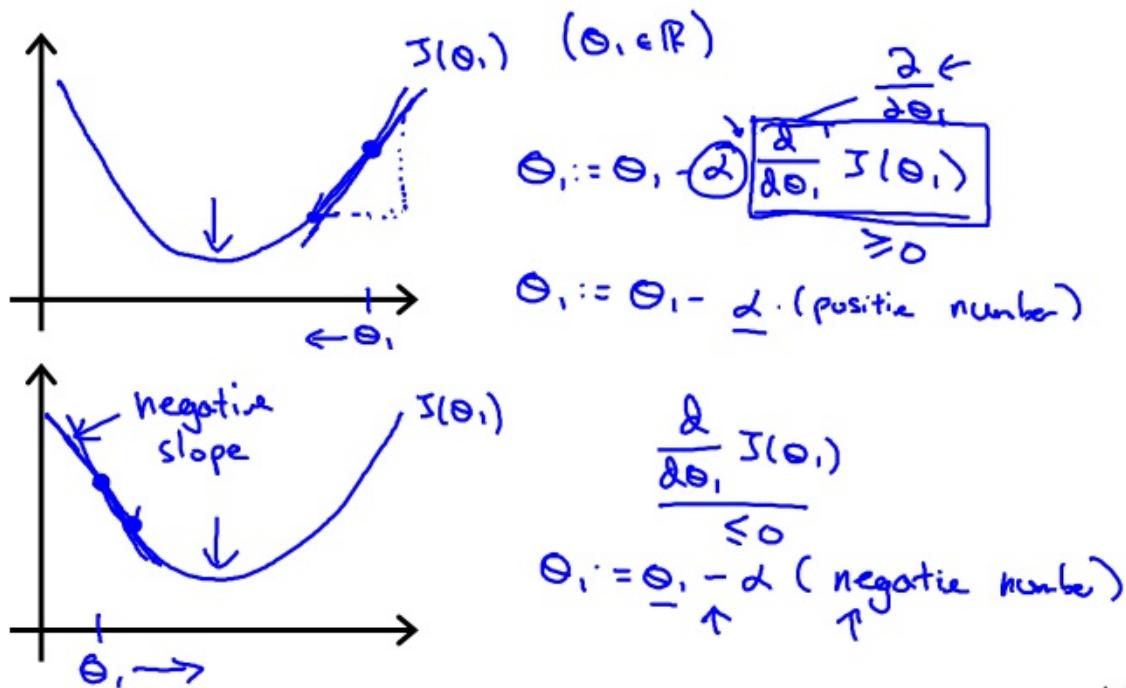
但事实上我们并不使用这种方式来表现。使用另外一种称作等高线的方法来表现。也就是说相同的J会在同一条线上，而最中心的线就是最小值。



Andrew Ng

2-5

关于 $J(\theta_0, \theta_1)$ 的选取，我们的目标是选取到最小的 J ，那么如何来找到最小的 J 呢？对于 J 的函数来说，找到最小极值点，就是全部或者局部的最小值。所以为了找到极值点，从数学的角度来说，如果只有一个参数，那么直接求导数，令其等于0，即可。目前是二维的，所以我们需要选取一个初始的值 (θ_0, θ_1) ，然后逐步的逼近最小值。对于，这个原理我们可以明白，以一定的步长沿着切线的方向，变化 θ_0 或者 θ_1 ，就会到达这个最小值。如下图：



从这张图上，我们看到在最低点的 J 。如果斜率大于0，那么 θ 需要逐渐变小；如果斜率小于0，那么需要 θ 逐渐变大。所以，能够得出如下的 θ 的逼近函数：

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

所以对于 θ 来说的逼近，是分别对 θ_0 和 θ_1 求偏导数的计算。

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

注意，这里的 θ_0 和 θ_1 ，必须同时计算，“同时”这个词非常关键，因为不同时，偏导数就是有问题的。

2-6

在理解了梯度下降的算法的工作原理和公式之后，我们可以代入具体的先前的 $h\theta(x)$ 来计算。

Gradient descent algorithm

$$\text{repeat until convergence} \{$$

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

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

$$\}$$

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

update θ_0 and θ_1 simultaneously

这是最终的求导之后的计算结果。通过这个计算结果就可以对 θ_0 和 θ_1 进行逼近计算。

Andrew Ng

当进入了局部最低点， $d(J(\theta))/d(\theta)=0$ ，事实上梯度下降最到局部最低，就会产生这个结果，这时候 θ 将不会改变，也就是达到了最终的收敛。

对于线性回来说，得到的局部最优解就是全局最优解，因为线性回归的代价函数是一个弓形，也就是一个凸函数，所以只有一个极值点，也就是全局最优解。

目前每次进行梯度下降的时候使用的是全部数据，所以叫做“批量梯度下降”，"Batch 梯度下降". 到最后，你会看到“随机梯度下降”和“小批量梯度下降”，来对这种大量数据的优化。

2-7

在前面我们看到的是只有两个维度，那么有多个维度的时候，该如何处理呢？需要应用到线性代数的知识。这是第三章主要阐述的内容，线性代数基础。

第三课 线性代数基本

Lesson3

矩阵和向量相乘的代数意义。在计算方程的时候是非常好的方法。这也是，我们需要扭转的思想，通过线性代数的思想去思考问题。例如：

$$h\theta(x) = \theta_0 + \theta_1 x$$

对于x取值为 $x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 4$ ，分别求 h_1, h_2, h_3, h_4 ，那么，针对这样的求解，就可以转换成矩阵的乘法来计算。 $h(x) = \theta_0 1 + \theta_1 x$ 。具体乘法如下：

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} \times \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix} = \begin{pmatrix} \theta_0 + \theta_1 \\ \theta_0 + 2\theta_1 \\ \theta_0 + 3\theta_1 \\ \theta_0 + 4\theta_1 \end{pmatrix} = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix}$$

使用这种方法来计算多个数据的方程式效率更高，而不是对x进行循环然后逐个去求解。因为对于计算机来说，进行矩阵运算是可以进行优化的。

针对一种 θ 的取值，可以这样做，显然，当进行多组计算的时候，这种计算就更加方便。例如：同时对四组 $h(x)$ 进行计算。 $h(x)$ 的方程组如下：

$$\begin{cases} h(x) = 10 - 10x \\ h(x) = 20 - 20x \\ h(x) = 30 - 30x \\ h(x) = 40 - 40x \end{cases}$$

针对 $x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 4$ ，这四种取值对应 $h(x)$ 的四个函数，一共会有16组 h 的计算结果，可以使用矩阵同时求出。看上面的解释就很容易明白，对 θ 向量再增加机组即可。所以，最终的矩阵如下：

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} \times \begin{bmatrix} 10 & 20 & 30 & 40 \\ -10 & -20 & -30 & -40 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix}$$

这样非常完美，一个矩阵相乘能够计算出16组测试结果。与上面的分析是一致的。

什么样的矩阵有逆矩阵呢？必须满足两个条件 1 必须是方阵。也就是说必须是 $n \times n$ 2 存在着 $AA^{-1} = I$

第四课 多特征量线性回归

Lesson4

4-1

前面所研究的都是基于单变量的预测模型，即通过房子的大小这一个特征量来拟合放假的曲线。然而，我们常常需要多个变量来进行拟合。现在的方程如下：

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

事实上这样一种表述方法，却可以通过向量的乘法来表示出来。

$$\begin{aligned} h_{\theta}(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n \\ &= [\theta_0 \ \theta_1 \ \theta_2 \ \dots \ \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}^T \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \theta^T x \end{aligned}$$

通过这里可以看到，所有的系数 θ 变量， x 是所有 x 的向量。通过这里知道，因为引入了 $x_0=1$ ，所以 x 的维度是 $n+1$ ，也就是课程中常写的 R^{n+1} 。

同时，对于多特征量的线性回归的数据结构是这样的 $(x_1, x_2, \dots, x_n, y)$ 而不是 $[(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)]$ 这种表述方式是单变量的线性回归。

4-2

再次回到梯度下降算法，

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

求 $\min(J)$ 的梯度递归下降逼近算法，是

Repeat {

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

}

when $j = 0, x^0 = 1,$

Repeat {

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

}

4-3

特征值缩放。对于 x_1, x_2, \dots, x_n 这些特征值的取值可能范围偏差很大，例如房屋的大小 1000-2000 的范围，而房间的数量 1-5. 这样会导致梯度线的不均匀，使得梯度下降算法比较慢。所以正确的做法，是使得所有的特征值都落在 [-0.5, 0.5] 之间，这样进行归一化会更好的进行梯度下降。那么如何归一化， x 在 $[a, b]$ 区间取值，那么，可以使用下面的方法，相当于将该区间的中心移动到 0，所以就是 $(a+b)/2$ 移动到 0，所以就是 $x - (a+b)/2$ 同时将大小变化，所以最终就是: $(x - (a+b)/2) / (b-a)$ 这就是归一化的方法。假设 $a = 100, b = 200$ ，那么就是 $(x-150)/100$ ，当 $x = 100$ 时，是 -0.5，当 $x = 200$ ，是 +0.5，当 $x = 130$ 时 -0.2，当 $x = 150$ 是 0. 所以恰好归一化特征变量。

$\text{normalize}(x) = (x - (a+b)/2) / (b-a), \quad x \in [a, b]$

4-4

对特征值的优化可以加速迭代。那么同时 α 的选取也是非常重要的。如果选取过大，会使得 $J(\theta)$ 无法下降收敛；如果选取过小那么则会收敛的速度过慢。所以一般从: 0.00001, 0.00003 开始，同时观察 $J(\theta)$ 的函数图像看看是否是不停的收敛变小，趋于平缓，然后再进行调优 α . 目前关于 $J(\theta)$ 的观察，仅仅限于肉眼观测。因为机器观察也是有很多问题。目前的做法，依然是通过 α 和 $J(\theta)$ 的图像进行观察，来得知是否收敛。

对于 α 的取值，是从 0.001, 0.003, 0.01 ... 3 倍速增加，来逐步观察得到最好的 α 。

4-5

非线性，而是多项式回归。

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

如果是这样，那么特征的归一化就非常的重要，例如 $x \in [1, 10]$ 但是 $x^2 \in [1, 100]$ $x^3 \in [1, 1000]$. 所以本质上还是对 x_1, x_2, x_3 这种特征量的归一，只是这时候变成了 $x_1 = x, x_2 = x^2, x_3 = x^3$.

关于特征量的选取，上面可能 x^3 的变化太快，那么可以考虑 $x^{1/2}$ ，这样可以使得变化没有那么快，以便得到更好的拟合。

不论是否是多项式对于 θ 的偏导数是没有影响的，只需要将每个 θ 的系数当做 x 即可。但是特征值的选取是个问题。

4-6

在这部分主要讲述的是数学的求解方法--标准方程法。例如， $J(\theta)$ 的最值问题，是可以通过对 θ 的求导，来得到最值。同时，如果 $h\theta(x)$ 是线性的，也可以直接通过线性代数求解。

Examples: $m = 4$.

x_0	Size (feet ²)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)
	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$$

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$m \times (n+1)$

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

Andrew Ng

$$X_{m \times n} \times \theta = y$$

$$\Rightarrow X_{m \times n}^T X_{m \times n} \times \theta = X_{m \times n}^T y$$

$$\Rightarrow (X_{m \times n}^T X_{m \times n})^{-1} X_{m \times n}^T X_{m \times n} \times \theta = (X_{m \times n}^T X_{m \times n})^{-1} X_{m \times n}^T y$$

$$\Rightarrow I \times \theta = (X_{m \times n}^T X_{m \times n})^{-1} X_{m \times n}^T y$$

$$\Rightarrow \theta = (X_{m \times n}^T X_{m \times n})^{-1} X_{m \times n}^T y$$

这里我们注意到 X 是非对角矩阵，也就是说 X 的逆矩阵是不存在，那么该如何计算呢？有个技巧，先将 X 变成对角矩阵，也就是乘以 X 的转置。所以，计算过程如下：

$$X_{m \times n} \times \theta = y$$

$$\Rightarrow X_{m \times n}^T X_{m \times n} \times \theta = X_{m \times n}^T y$$

$$\Rightarrow (X_{m \times n}^T X_{m \times n})^{-1} X_{m \times n}^T X_{m \times n} \times \theta = (X_{m \times n}^T X_{m \times n})^{-1} X_{m \times n}^T y$$

$$\Rightarrow I \times \theta = (X_{m \times n}^T X_{m \times n})^{-1} X_{m \times n}^T y$$

$$\Rightarrow \theta = (X_{m \times n}^T X_{m \times n})^{-1} X_{m \times n}^T y$$

通过这样的一种数学方式也可以直接求出 θ 。我的疑惑？就是 θ 的个数是 n ，所以要求解需要 n 个方程即可，可是 m 是大于 n 的，那么就是无法求解出 θ ，因为方程冗余了？这个需要使用OCTAVE进行验证一下，奇怪的现象。

那么究竟是选择梯度下降还是使用标准方程呢？结论是：如果特征小于 10^6 ，使用标准方程；否则，因为逆矩阵的计算较慢，需要使用梯度下降。另外，标准方程只适用于线性回

归，也就是线性的 $h(x)$ 表达式，所以并非适用于所有，在线性回归中速度会很快。

4-7

通过上面的计算，可以看到求 θ 的时候 X 乘以 X 的转置的逆矩阵不存在的问题。如果过是这样，需要做两点：

- 1 检查 X 矩阵是否有线性相关的向量。
- 2 可能是特征值过多，减少特征值的数量试试。

第五课 Octave介绍

Lesson5

主要讲述Octave语法以及Octave的应用。所以对于数据的理解，变成另外一个层面的理解，那就是向量化理解。我们之前在处理很多问题的时候，事实是程序思维，但是现在需要进化成更加抽象的向量思维。看下面一段代码：

```
int sum = 0;
int[] a = {1, 2, ..., 100};
for (int i = 0; i < 100; i++) {
    sum += a[i];
}
```

上面这段代码是非常典型的求和算法，那么，可以通过向量化的思维来使得这个操作更加快速。代码如下：

```
sum = [1, 1, 1 ... 1] * [a[0], a[1], ..., a[n]]'
```

所以通过上面的例子，我们看出使用一行向量相乘就能处理循环的问题。那么，向量化方法的特点是什么呢？通过观察矩阵相乘，我们知道需要有如下特征：

- Σ 符号，需要有来产生加法云轩
- 需要有乘积

有了以上两点，就可以考虑使用向量或者矩阵相乘来进行优化处理。看下面的梯度下降算法的向量化计算。里面有 Σ 符号有相乘。那么，我们现在就来进行 θ 的向量化计算。

Repeat {

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

}

when j = 0, x⁰ = 1,

Repeat {

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

}

进行向量化计算的步骤如下：

$$\begin{aligned}
\theta_j &= \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x^j \\
\omega^{(i)} &= h_\theta(x^{(i)}) - y^{(i)}, \\
\Rightarrow \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x^j &= \sum_{i=1}^m \omega^{(i)} x^j \\
&= [w_1 \ w_2 \ \dots \ w_m] \begin{bmatrix} x^j \\ x^j \\ x^j \\ x^j \end{bmatrix} \\
\Rightarrow \theta_j &= \theta_j - \alpha \frac{1}{m} [w_1 \ w_2 \ \dots \ w_m] \begin{bmatrix} x^j \\ x^j \\ x^j \\ x^j \end{bmatrix} \\
\Rightarrow \theta &= \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dots \\ \theta_n \end{bmatrix}
\end{aligned}$$

第六课 逻辑回归

Lesson6

6-1

第六课，开始介绍分类算法，logistic 分类算法。对于前面学习的线性回归算法来说，是无法适应在分类算法的。因为线性回归算法是连续的数据集，而对于分类算法来说，是对在同一个分类中不同的分类产生的概率。

为什么不能使用线性回归来进行分类测试呢？是因为分类取值 y 属于 $\{0, 1\}$ 而对于 $h(x)$ 来说，可能产生 $h(x)>1$ 和 $h(x)<0$ ，这两种情况，那么这并不是合理的。所以线性回归是不能进行分类的。而应该使用 $h(x)$ 属于 $[0, 1]$ 的假设函数。

6-2

分类算法的数据原型，是癌症的良性和恶性计算。

良性/恶性 0/1 肿瘤大小

0	2
1	4
1	6
0	1

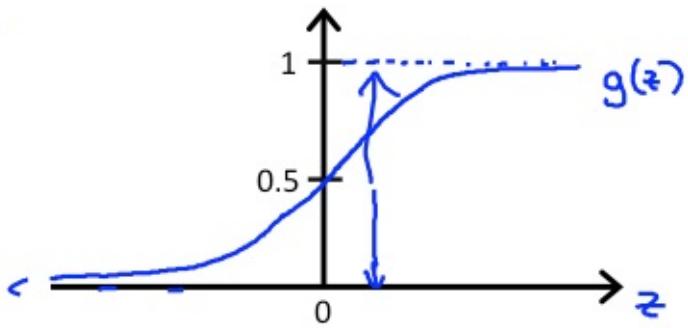
对于分类算法来说，是通过肿瘤的大小来计算良性还是恶性的概率。所以对于 $h_\theta(x)$ 是计算概率也就是：

$$h_\theta(x) \in [0, 1]$$

所以这是问题的关键，通过这一点就明白为什么线性回归在这里无法工作的原因了。因为线性回归是没有区间限制也就是概率的处理.但是，可以通过对线性回归的函数进行变换，形成在 $[0, 1]$ 之间。

$$\begin{cases} h_\theta(x) = \theta^T x \\ g(x) = \frac{1}{1 + e^{-\theta^T x}} \end{cases} \Rightarrow h_\theta(x) = g(h_\theta(x)) = \frac{1}{1 + e^{-\theta^T x}}$$

通过上面的 $g(x)$ 可以知道， $g(x) \in [0, 1]$ 。



所以对于概率上的表示就是

$$P\{y=1 \mid x; \theta\} + P\{y=0 \mid x; \theta\} = 1$$

$h(x)$ 的含义是什么？ $h(x)=0.7$ 表示的是 $y=1$ 的概率是70%。 $h(x)$ 的数值表示的是出现 $y=1$ 的概率，也就是 $P\{y=1 \mid h(x)\} = P\{y=1 \mid x, \theta\}$ 。也就是说当我们求出了逻辑回归的假设函数 $h(x)$ 我们就能够知道他为1的概率。用来预测股票的涨和跌，可以使用。计算出来 $h(x)$ ，就计算出了明天上涨的概率，然后通过凯利公式计算头寸。

6-3

决策边界。这一节主要介绍的是决策边界的问题。

预测 $y=1$, 如果 $h(x) >= 0.5$

预测 $y=0$, 如果 $h(x) < 0.5$

注意这里的 $h(x)$ 的取值，事实上我们可以取值为任意数值，例如 $y=1$, 如果 $h(x) >= 0.7$ ，这样做好处是增加成功率。所以本质上逻辑回归就是在计算概率，根据概率大于阈值来确定一个 y 究竟是不是1. 这在股票的预测中尤为有用。

对于 $g(x)$ 函数来说，通过图像我们可以知道，对于最终的计算结果来说，要么是0要么是1，那么认为 $g(x) >= 0.5$ 和 $g(x) < 0.5$ 两个区间进行分析。对于 x 的取值来说就是 $x >= 0$ 和 $x < 0$ 。所以对于 $x = 0$ 来说，就是边界条件。对应到前面的函数就是：

$$\theta^T x = \theta$$

这个就是边界条件。对于两个变量的函数如下：

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$$

所以可见，对于 x_1, x_2 来说是一个直线。当然在进行拟合的时候，也可以使用多项式拟合，例如：

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 = 0$$

取 $\theta_0 = -1, \theta_1 = 0, \theta_2 = 0, \theta_3 = 1, \theta_4 = 1$ ，那么， $x_1^2 + x_2^2 = 1$ ，显然这是一个圆。所以决策边界就是一个圆的区域。

逻辑回归的边界绘制

通过 $\theta^T X = 0$, 得出 x_1, x_2 的关系, 取 $x_1 = [0.0, \dots]$ 一个序列, 计算出 x_2 , 然后将 (x_1, x_2) 所有的点连接起来, 就是边界。

6-4

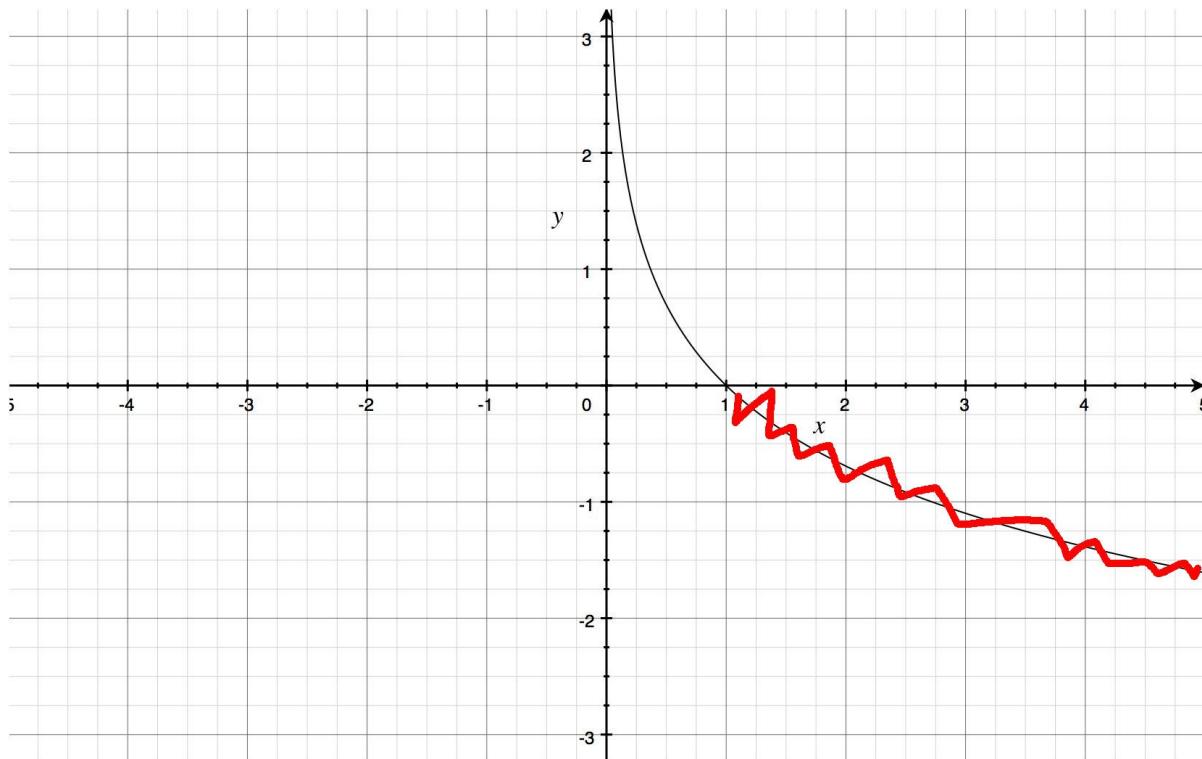
代价函数的计算。使用线性回归的代价函数式无法计算的, 因为现在的函数是指数函数, 所以在收敛上是个问题, 换句话说, 不是凸函数。那么, 如何转换成凸函数呢?

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

什么是代价函数? 代价函数式对 θ 的一种拟合方式, 也就是讨论 $h_\theta(x)$ 和 y 的关系的最贴近的拟合方法。在线性回归上面的方法是, $h_\theta(x)$ 和 y 的方差来决定的, 是所有的 $h_\theta(x)$ 与 y 的方差最小的函数。通过这一点, 我们可以得知, 代价函数就是对不同的 θ 的取值, 然后不停的迭代所有的 x , 使得所有的 $h_\theta(x)$ 与 y 的值最小的一种方法。代价函数式对“所有”数据的一种拟合函数。

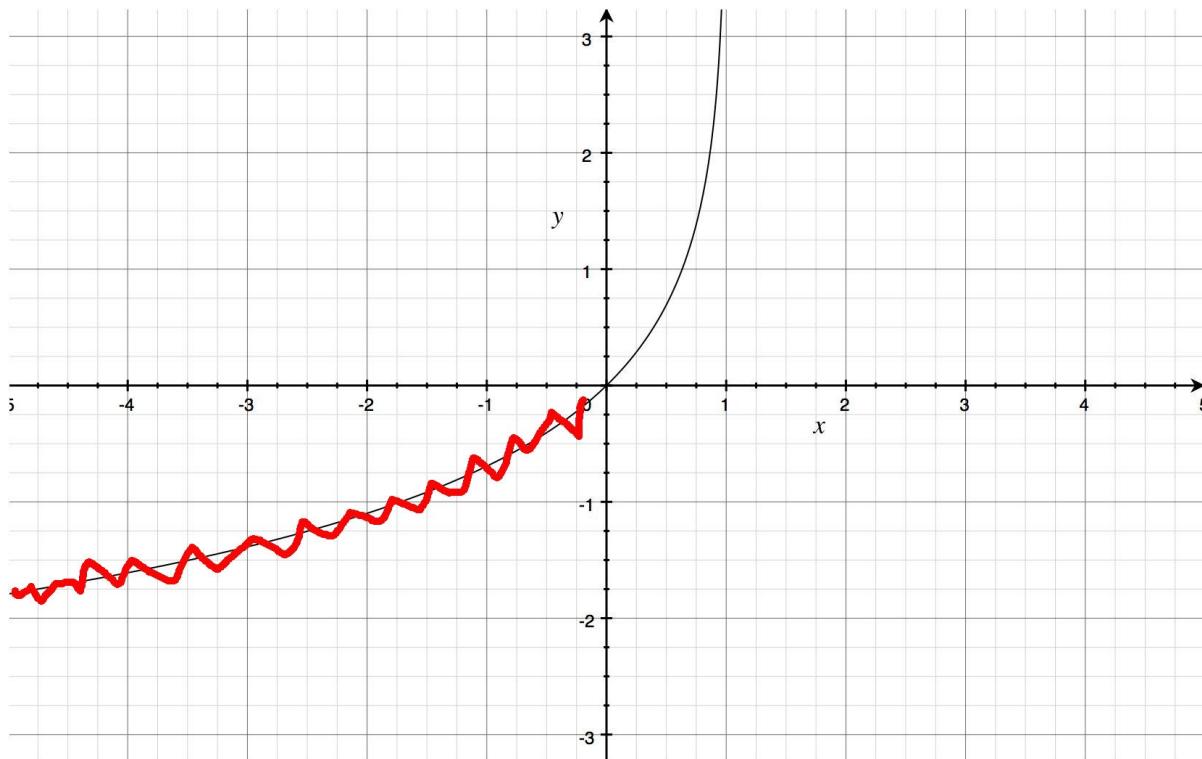
所以对于现在来说, 既然 $y=1$ 或者 $y=0$, 所以就分两种拟合方法, 对所有的 $y=1$, 使用 $-\log(h_\theta(x))$ 来处理 $y=1$ 的情况。

对于两者的代价函数的图像如下:



因为 $h(x)$ 的取值范围是 $[0, 1]$ ，所以只关注这一个区间的图像即可。通过图像我们看到当 $h(x)$ 趋近于 1 的时候，代价函数趋近于 0，这里的含义就是 $y=1$ 的时候代价函数最小，这正是我们需要的；而当 $h(x)$ 趋近于 0 的时候，代价函数趋近于无穷大，也就是当 $y=0$ 的时候，代价函数无穷大，这是正确的结论，也就是。对于概率还说正是如此。

同样对于 $y=0$ 的函数图像如下图，依然是 $h(x) \in [0, 1]$.



这里告诉我们，如果你的代价函数，不是一个凸函数，那么将无法进行梯度下降的收敛。所以无论如何要通过函数的变换将其变成凸函数。

6-5

代价函数的最小化问题。首先，需要对代价函数的分段表示方式变成统一的表达式。如下：

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

通过将Cost函数变成统一的表达式，剩下的就容易多了。求 $\min(J(\theta))$ ，具体的思想与线性回归是差不多的。使用梯度下降法。推导如下：

$$\begin{aligned}
\theta_j &= \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \\
\frac{\partial}{\partial \theta} J(\theta) &= -\frac{1}{m} \left[\sum_{i=1}^n y^{(i)} \frac{1}{h_\theta(x^{(i)})} \frac{\partial}{\partial \theta_j} (h_\theta(x^{(i)})) + (1 - y^{(i)}) \frac{1}{1 - h_\theta(x^{(i)})} \left(-\frac{\partial}{\partial \theta} (h_\theta(x^{(i)})) \right) \right] \\
h &= h_\theta(x^{(i)}), y = y^{(i)}, k = \frac{\partial}{\partial \theta_j} (h_\theta(x^{(i)})) \\
\frac{\partial}{\partial \theta} J(\theta) &= -\frac{1}{m} \left[\sum_{i=1}^n y \frac{1}{h} k - (1 - y) \frac{1}{1 - h} k \right] \\
&= -\frac{1}{m} \left[\sum_{i=1}^n k \left(\frac{y - h}{h(1 - h)} \right) \right] \\
h_\theta(x^{(i)}) &= \frac{1}{1 + e^{-\theta^T x}} \\
k &= \frac{\partial}{\partial \theta_j} (h_\theta(x^{(i)})) = \frac{\partial}{\partial \theta_j} \left(\frac{1}{1 + e^{-\theta^T x}} \right) = (-1) (1 + e^{-\theta^T x})^{-2} e^{-\theta^T x} (-x_j) = h^2 e^{-\theta^T x} x_j \\
e^{-\theta^T x} &= \frac{1 - h}{h} \Rightarrow k = h^2 e^{-\theta^T x} x_j = h^2 \frac{1 - h}{h} x_j = h(1 - h) x_j \\
\frac{\partial}{\partial \theta} J(\theta) &= -\frac{1}{m} \left[\sum_{i=1}^n k \left(\frac{y - h}{h(1 - h)} \right) \right] \\
&= -\frac{1}{m} \left[\sum_{i=1}^n (h(1 - h) x_j) \left(\frac{y - h}{h(1 - h)} \right) \right] \\
&= \frac{1}{m} \left[\sum_{i=1}^n (h - y) x_j \right] \\
&= \frac{1}{m} \left[\sum_{i=1}^n (h_\theta(x^{(i)}) - y) x_j \right] \\
\theta_j &= \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) = \theta_j - \alpha \frac{1}{m} \left[\sum_{i=1}^n (h_\theta(x^{(i)}) - y) x_j \right]
\end{aligned}$$

我们看到最终推导迭代结果与线性回归的是一样的。

6-6

这一部分主要是教会我们使用更快速和更高级的方法来优化梯度算法的收敛。主要函数是：

fminunc
参数包括： **optimset**

通过fminunc函数来自动的进行梯度下降，来自动的使用不同的 θ 进行迭代来，找到最小的 $J(\theta)$ 。所以到这里，你明白了，在数学上推导的梯度下降的 θ_j 的迭代下降，是通过这里的fminunc函数来自动进行处理的，省得我们自己写循环来进行比较。书写的标准格式如下：

```

theta 初始向量 [00, 01, ..., 0n] 的转至
function [jVal, gradient] = constFunction(theta)
    jVal = [code to compute J(theta)]
    gradient(1) = [code to compute 偏导θ]
    gradient(2) = [code to compute 偏导θ]
    ...
    gradient(n+1) = [code to compute 偏导θ]

```

使用OCTAVE或者MATLAB的库函数进行梯度下降，比我们先前自己完成的更快，因为

会有很多自动优化，例如 α 的自动选择。所以，以后就使用 fminunc 函数来自动进行梯度下降，而不是自己再次书写。

6-7

前面我们所进行的分析都是基于0，1的两个数据的分类，在这部分来解决处理多种分类的问题，例如1，2，3的分类问题。计算的思想是：

h1: 1, (2, 3) 计算出h1;

h2: (1, 2), 3 计算出h2

h3: 2, (1, 3) 计算出h3

最后计算 $h = \max(h)$

所以总体的逻辑还是基于0，1分类的计算。

第七课 正规化

Lesson7

7-1

这里介绍了拟合中的两个问题，分别是欠拟合和过拟合。所谓的欠拟合，是指选取的特征变量过少，而使得不能很好的拟合大部分数据。而所谓的过拟合，是指追求对每一个样本数据的精准拟合而导致特征变量过于复杂。这两种都是不对的做法。对于过拟合来说，可以考虑减少某些特征量，但是，减少了意味着丢掉某些信息。

7-2

正规化来对欠拟合和过拟合的处理。引入正规化处理的目的是使得，很多的特征量能够对影响很小的特征量处理掉，这样能够使得代价函数更加简洁，拟合的更好。

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

从公式上看出，对于 λ 项的引入是对 $J(\theta)$ 整体的调整，而不是对每一个样本的调整。

在这里的最后一项是代价函数没有项，通过最后一项来对所有的 θ 进行惩罚，将关联度低的进行消除。一般会设置 λ 为一个比较大的数，例如 10^{10} 以便对 θ 对比较大的惩罚，产生较小的 θ 。注意上面的代价函数中 正规化参数中 θ 是从1开始的而不是0，对于 θ_0 的惩罚意义不大。

那么，惩罚的原理又是什么呢？是如何做到惩罚的呢？

因为，我们需要 $\text{Minimize}(J(\theta))$ ，所以如果 λ 的值很大，要想 $J(\theta)$ 很小，那么一定是 θ

那么，又是如何反应到假设函数 $h(\theta)$ 上的呢？

当，通过梯度下降来求解 $\text{Minimize}(J(\theta))$ 的时候，自然会将 λ 计算在内，因为这个因素的

7-3

基于上面的方程，对于梯度下降法来说，计算结果如下：

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

Repeat {

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

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

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

}

对于 θ_0 来说，是没有 λ 项的。对于其他的 θ 来说在进行下降的时候使用了 $1 - \alpha \lambda / m$ 作为系数，所以对于 θ 来说的下降来说是逐渐变小的。

对于非梯度下降的正规方程法来说，

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

$\lambda > 0.$

这个正规化的好处是，逆矩阵是一定存在的，这是可以证明的。小遗憾，这个方程的具体推导我还没有搞清楚。

7-4

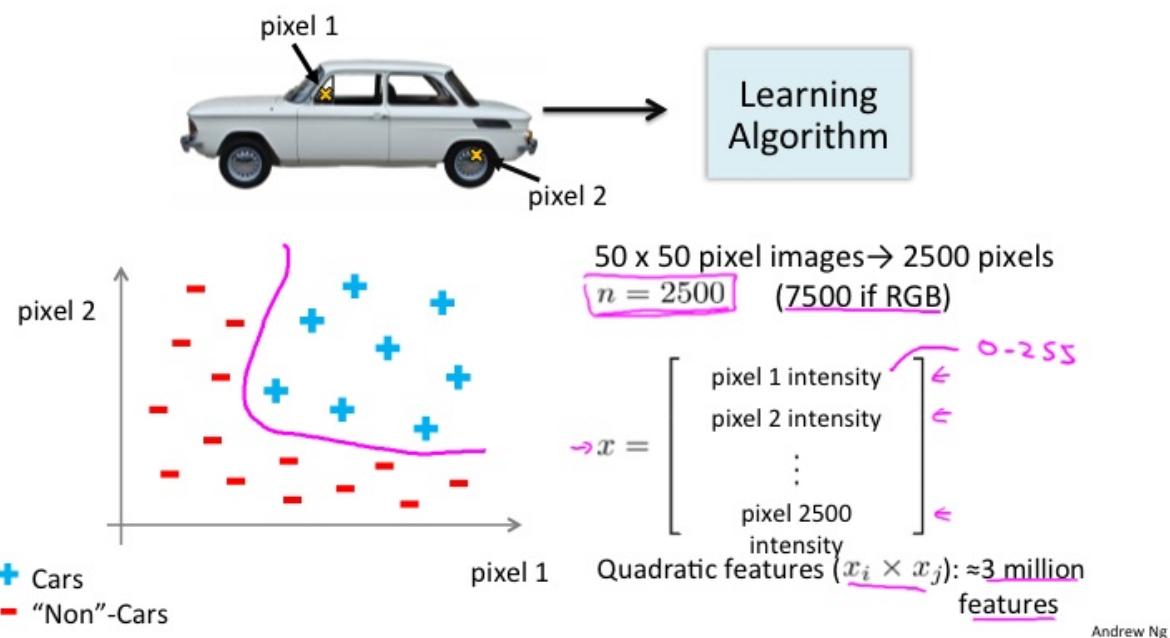
对于逻辑回归来说也是同样的表达式。

第八课 神经网络

Lesson8

8-1

神经网络的机器学习。在使用逻辑回归或者线性回归的时候如果参与的特征量过于多，那么基于多项式的函数将会出现及其复杂的现象。所以引入神经网络来解决这个问题。例如，在图像识别中的问题。

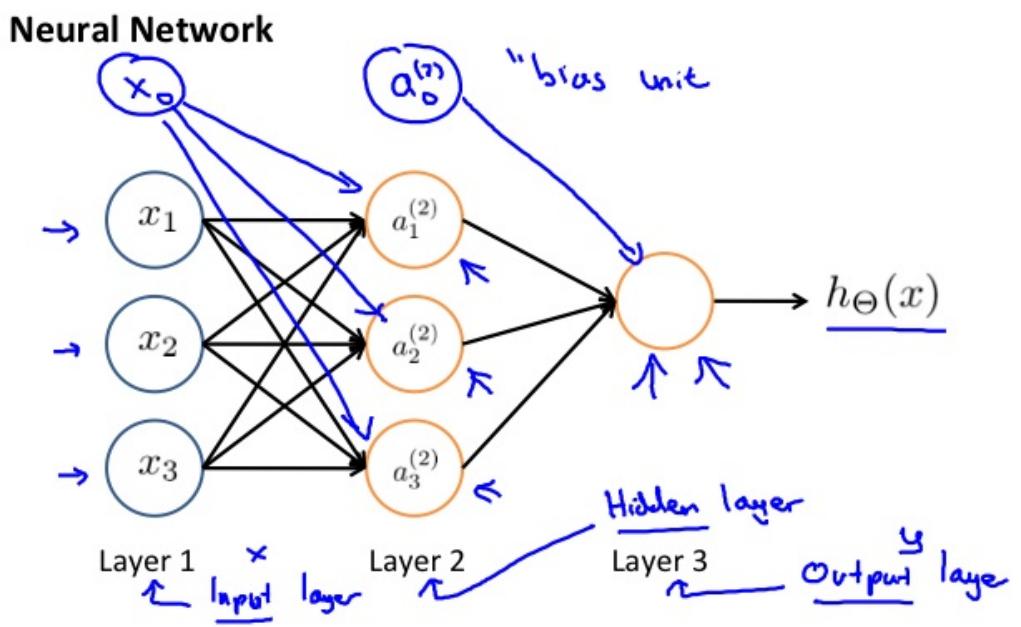


如图，将汽车图片的每一个像素的位置，是汽车用加号表示不是汽车用减号表示，那么有多少个像素相当于有多少个特征值，所以对于这个问题来说特征值就是图片的大小所包含的像素数。例如 $50 \times 50 = 2500$ 个像素，如果每个像素用3个字节来表示，那么就是 $2500 \times 3 = 7500$ 个字节，每一个字节作为一个特征量。换句话说是否是一个汽车是通过所有像素点来决定的，所以特征量就是所有像素点。

8-2

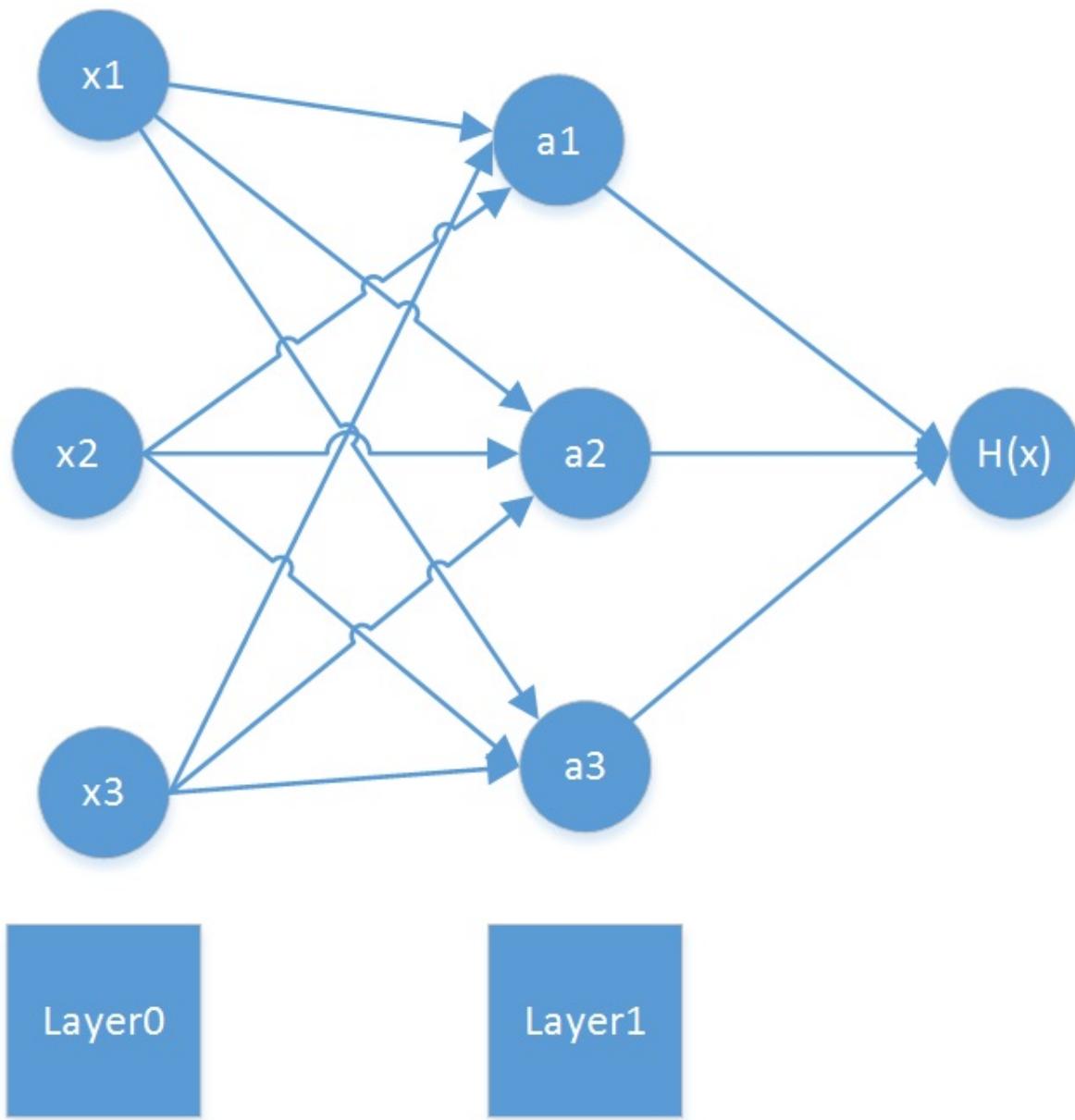
神经网络是模拟大脑的算法。任何一种传感器接入大脑，大脑都会自己学习如何处理这些数据。这就是神经网络的理论基础。

8-3



Andrew Ng

神经网络模型，是根据不同的层进行处理的。



具体的数学公式如下：

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

$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$$h_{\Theta}(x) = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

层级：1

a 小标：i 第几个单元

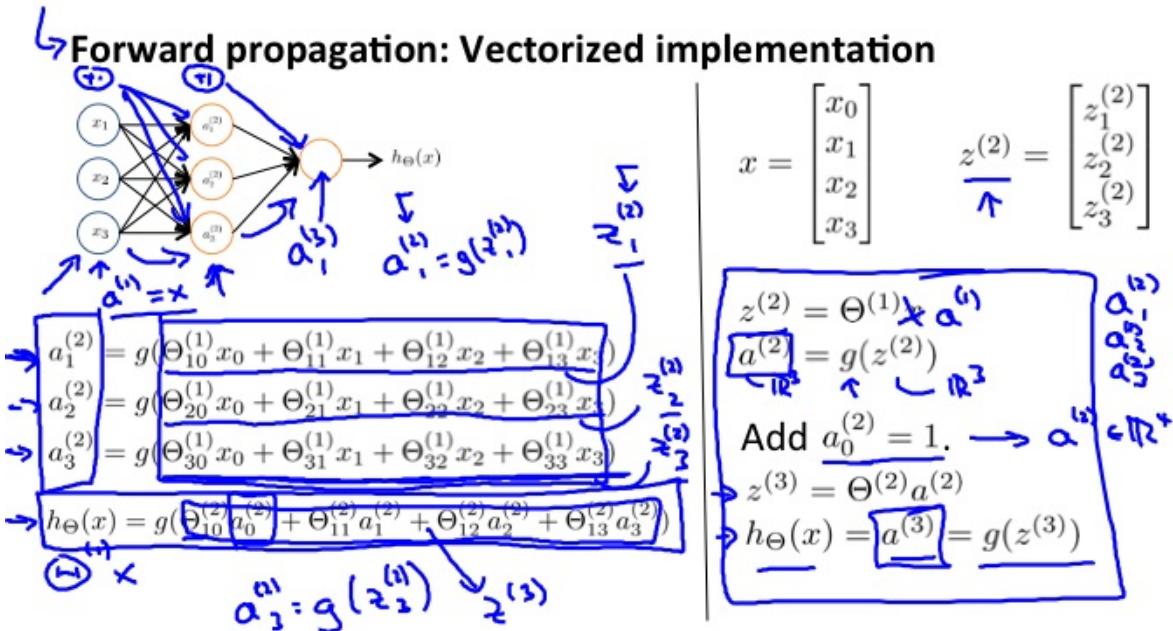
前一层的第几个单元：j

a 的上标是层级 1，下标是第几个单元i。

Θ 上标是层级 1，下标第一个是i 也就是当前a的下标，另一个是j，表示的是前一层的第几个。对于 Θ ，每一行向量是计算的权重向量

8-4 8-5 8-6

神经网络的向前传播算法，向量化表述非常重要，如下图：



Andrew Ng

注意 $h(\theta)$ 的表述，是最后一层，也就是输出层。在每一个增加偏差项会让公式更加完美，不仅仅是 x_0 ，而是所有 $a_0 = 1$ 。

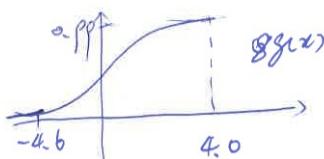
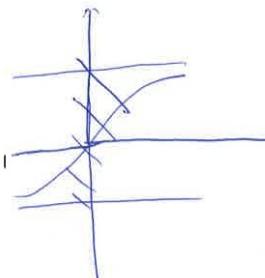
具体的计算实例参考下面的计算过程。

机器学习:

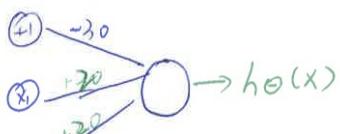
$$f = x_1 \text{ XOR } x_2$$

$$x_1 \text{ XOR } x_2$$

$$\text{NOT}(x_1 \text{ XOR } x_2)$$



AND

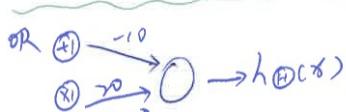


$$h_{\oplus}(x) = g(x_0 \oplus_{10}^{(1)} x_1 \oplus_{11}^{(1)} + x_2 \oplus_{12}^{(1)})$$

$$= g(-3.0 + 2.0x_1 + 2.0x_2)$$

$$= g(-3.0 + 2.0x_1 + 2.0x_2)$$

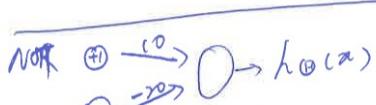
		$h_{\oplus}(x)$	$x_1 \text{ XOR } x_2$
x_1	x_2	$g(-3.0) \approx 0$	0
0	0	$g(-1.0) \approx 0$	0
0	1	$g(-1.0) \approx 0$	0
1	0	$g(-1.0) \approx 0$	0
1	1	$g(1.0) \approx 1$	1



$$h_{\oplus}(x) = g(x_0 \oplus_{10}^{(1)} + x_1 \oplus_{11}^{(1)} + x_2 \oplus_{12}^{(1)})$$

$$= g(-1.0 + 2.0x_1 + 2.0x_2)$$

		$h_{\oplus}(x)$	$x_1 \text{ XOR } x_2$
x_1	x_2	$g(-1.0) \approx 0$	0
0	0	$g(1.0) \approx 1$	1
0	1	$g(1.0) \approx 1$	1
1	0	$g(1.0) \approx 1$	1
1	1	$g(3.0) \approx 1$	1

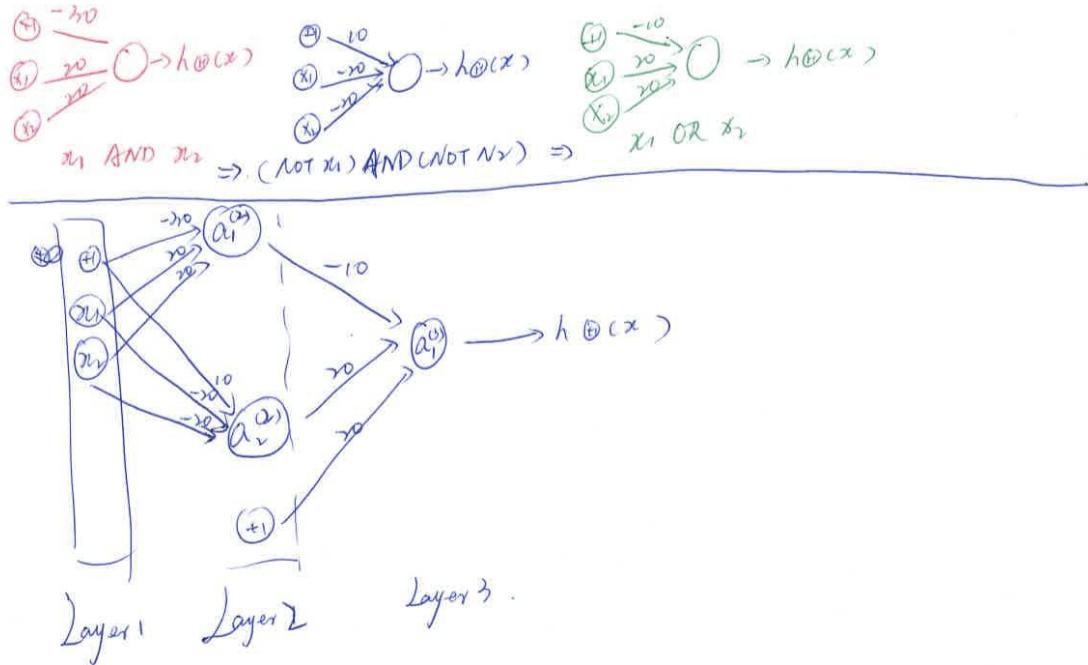


$$h_{\oplus}(x) = g(x_0 \oplus_{10}^{(1)} + x_1 \oplus_{11}^{(1)} + x_2 \oplus_{12}^{(1)})$$

$$= g(1.0 + -20 \cdot x_1)$$

$$= g(1.0 - 20 \cdot x_1)$$

		$h_{\oplus}(x)$	$\text{NOT}(x_1)$
x_1	x_2	$g(1.0) \approx 1$	1
0		$g(1.0) \approx 1$	0
1		$g(1.0) \approx 1$	1



最后需要注意一点，对于神经网络的学习，最后的 y 是一个向量，如果有3个分类结果是 $[1\ 0\ 0]'$ $[0\ 1\ 0]'$ $[0\ 0\ 1]'$ 而不是 $y \in [1, 2, 3]$ 这样的问题。

第九课 神经网络学习

Lesson9 神经网络学习

9-1 代价函数

还记得在逻辑回归中的分类问题，对进行0 1，这样的简单分类是容易解决的。可是如果是0 1 2 3 4 这样的分类讲是非常痛苦的。但是这些在神经网络中的处理确容易了许多。基于神经网络的代价函数如下：

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

K：表示K类的数量。例如 0 1 2 3 那么就是4

逻辑回归函数

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

所以，这个公式与逻辑回归的代价函数进行对比。其实，仅仅是多了K个分类的处理。加入正规化的最终的函数如下：

$$\begin{aligned} J(\theta) &= -\frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(h_\theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\theta(x^{(i)}))_k) \\ &+ \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{j,i}^{(l)})^2 \end{aligned}$$

在逻辑回归中最后一项是所有 θ 的平方和，在这里也是同样的。

L： 表示层数

i, j： 对应于每一层的每个 θ 的下角标

代价函数的向量化

代价函数如下：

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

那么，在进行计算的时候，需要进行向量化，推导结果如下：

$$\begin{aligned}
 J(\theta) &= -\frac{1}{m} \sum_{i=1}^n \sum_{k=1}^K y_k^{(i)} \log(h_\theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\theta(x^{(i)}))_k) \\
 &= -\frac{1}{m} \sum_{i=1}^n \sum_{k=1}^K y_k^{(i)} \log((a_k^{(L)})^{(i)}) + (1 - y_k^{(i)}) \log(1 - (a_k^{(L)})^{(i)}) \\
 &= -\frac{1}{m} \sum_{i=1}^n [\log((a_1^{(L)})^{(i)}) \log(1 - (a_1^{(L)})^{(i)})] * \begin{bmatrix} y_1^{(i)} \\ 1 - y_1^{(i)} \end{bmatrix} + \dots + [\log((a_K^{(L)})^{(i)}) \log(1 - (a_K^{(L)})^{(i)})] * \begin{bmatrix} y_K^{(i)} \\ 1 - y_K^{(i)} \end{bmatrix} \\
 &= -\frac{1}{m} \sum_{i=1}^n [[\log((a_1^{(L)})^{(i)}) \log(1 - (a_1^{(L)})^{(i)})] \dots [\log((a_K^{(L)})^{(i)}) \log(1 - (a_K^{(L)})^{(i)})]] * \begin{bmatrix} y_1^{(i)} \\ 1 - y_1^{(i)} \\ \dots \\ y_K^{(i)} \\ 1 - y_K^{(i)} \end{bmatrix} \\
 &= -\frac{1}{m} \begin{bmatrix} \log((a_1^{(L)})^{(1)}) \\ \log(1 - (a_1^{(L)})^{(1)}) \\ \dots \\ \log((a_K^{(L)})^{(1)}) \\ \log(1 - (a_K^{(L)})^{(1)}) \end{bmatrix}^T \begin{bmatrix} \log((a_1^{(L)})^{(2)}) \\ \log(1 - (a_1^{(L)})^{(2)}) \\ \dots \\ \log((a_K^{(L)})^{(2)}) \\ \log(1 - (a_K^{(L)})^{(2)}) \end{bmatrix}^T \dots \begin{bmatrix} \log((a_1^{(L)})^{(n)}) \\ \log(1 - (a_1^{(L)})^{(n)}) \\ \dots \\ \log((a_K^{(L)})^{(n)}) \\ \log(1 - (a_K^{(L)})^{(n)}) \end{bmatrix}^T * \begin{bmatrix} y_1^{(1)} \\ 1 - y_1^{(1)} \\ \dots \\ y_K^{(1)} \\ 1 - y_K^{(1)} \\ y_1^{(2)} \\ 1 - y_1^{(2)} \\ \dots \\ y_K^{(2)} \\ 1 - y_K^{(2)} \\ \dots \\ y_1^{(n)} \\ 1 - y_1^{(n)} \\ y_K^{(n)} \\ 1 - y_K^{(n)} \end{bmatrix} \\
 &= -\frac{1}{m} \begin{bmatrix} \log((a_1^{(L)})^{(1)}) \\ \log((a_K^{(L)})^{(1)}) \\ \dots \\ \log(1 - (a_1^{(L)})^{(1)}) \\ \log(1 - (a_K^{(L)})^{(1)}) \end{bmatrix}^T \begin{bmatrix} \log((a_1^{(L)})^{(2)}) \\ \log((a_K^{(L)})^{(2)}) \\ \dots \\ \log(1 - (a_1^{(L)})^{(2)}) \\ \log(1 - (a_K^{(L)})^{(2)}) \end{bmatrix}^T \dots \begin{bmatrix} \log((a_1^{(L)})^{(n)}) \\ \log((a_K^{(L)})^{(n)}) \\ \dots \\ \log(1 - (a_1^{(L)})^{(n)}) \\ \log(1 - (a_K^{(L)})^{(n)}) \end{bmatrix}^T * \begin{bmatrix} y_1^{(1)} \\ 1 - y_1^{(1)} \\ y_K^{(1)} \\ \dots \\ y_1^{(n)} \\ 1 - y_1^{(n)} \\ y_K^{(n)} \\ \dots \\ 1 - y_1^{(n)} \\ \dots \\ 1 - y_K^{(n)} \end{bmatrix} \\
 &= -\frac{1}{m} [(\log a^{(1)})^T (\log(1 - a^{(1)}))^T \dots (\log a^{(n)})^T (\log(1 - a^{(n)}))^T] * \begin{bmatrix} y^{(1)} \\ 1 - y^{(1)} \\ \dots \\ y^{(n)} \\ 1 - y^{(n)} \end{bmatrix} \\
 &= -\frac{1}{m} [(\log a^{(1)})^T \dots (\log a^{(n)})^T (\log(1 - a^{(1)}))^T \dots (\log(1 - a^{(n)}))^T] * \begin{bmatrix} y^{(1)} \\ 1 - y^{(1)} \\ \dots \\ y^{(n)} \\ 1 - y^{(n)} \end{bmatrix} \\
 &= -\frac{1}{m} [\log a \log(1 - a)] \begin{bmatrix} y \\ 1 - y \end{bmatrix} \\
 \text{Let: } a^{(i)} &= \begin{bmatrix} a_1^{(i)} \\ \dots \\ a_K^{(i)} \end{bmatrix}, a = [(\log a^{(1)})^T \dots (\log a^{(n)})^T] = \begin{bmatrix} a^{(1)} \\ \dots \\ a^{(n)} \end{bmatrix}^T
 \end{aligned}$$

引入正规化 λ 的计算项是，所有的 θ 的平方，那么就是将 Θ 按照行展开，然后进行点乘计算即可。

$$\frac{\lambda}{2m} \sum_{l=1}^L \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{j,i}^{(l)})^2 = \frac{\lambda}{2m} \Theta(:) * (\Theta(:))^T$$

代价函数的计算

将每一层的 $a^{(l)}$ 计算出来。带入 $J(\Theta)$ 即可。

For 1 sample:

$$\begin{cases} a^{(t)} = \Theta^{(t-1)} * a^{(t-1)} \\ a^{(1)} = [1 \ x] \\ \text{Let: } \text{size}(a^{(t)}) = (sl, 1) \end{cases}$$

For i th sample:

$$\begin{cases} (a^{(t)})^{(i)} = \Theta^{(t-1)} * (a^{(t-1)})^{(i)} \\ (a^{(1)})^{(i)} = [1 \ x^{(i)}] \\ \text{Let: } \text{size}(a^{(t)}) = (sl, 1) \end{cases}$$

For m samples:

$$\begin{cases} a^{(t)} = \Theta^{(t-1)} * a^{(t-1)} \\ a^{(1)} = [1 \ X] \\ \text{Let: } \text{size}(a^{(t)}) = (sl, 1), X \text{ is } m \text{ sample matrix.} \end{cases}$$

正规化 λ 项的计算，只需要将 Θ 展开成行向量，使用点乘即可。

$$\frac{\lambda}{2m} \sum_{l=1}^L \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{j,i}^{(l)})^2 = \frac{\lambda}{2m} \Theta(:) * (\Theta(:))^T$$

y的向量化

在给出的分类数据中 $y \in [1, 2, 3, \dots, 10]$ 这样给出的每一条数据，而在代价函数中我们需要的是 $y_k = [0 \ 0 \ 0 \ 1 \ \dots \ 0]$ 这种向量表述，也就是对于每一个 y 都要进行这种向量化转换。

$y=1, [1 \ 0 \ 0 \ 0 \ 0 \ \dots \ 0]$

$y=2, [0 \ 1 \ \dots \ 0]$

...

那么这种是如何计算呢？对于Octave来说计算步骤如下：

- 1 先生成一个标准的循环向量。例如： $Y1 = [1 \ 2 \ 3 \ \dots \ 10 \ 1 \ 2 \ \dots \ 10 \ \dots]$ 从1 到
- 2 将 y 进行扩展和上面的 $Y1$ 一样的维度，就是 $Y2 = y \ .^* [1 \ 1 \ \dots \ 1]$
- 3 $Y = (Y1 == Y2)$ 就是所要的矩阵

$y \in \{1, 2, 3, 4, 5\}$, 计算分类 y 的Octave代码如下:

```

y = [ 1, 2, 3, 3, 5]

Y1 = [];
Y2 = [];

for i = 1: length(y)
    yy = y(i) * [1, 1, 1, 1, 1];
    Y1 = [Y1, yy];

    Y2 = [Y2, [1 2 3 4 5]];
end

Y1
Y2

Y = (Y1 == Y2)

```

输出结果:

```

y = 1   2   3   3   5

Y1 = 1   1   1   1   1   2   2   2   2   2   3   3   3   3   3   3

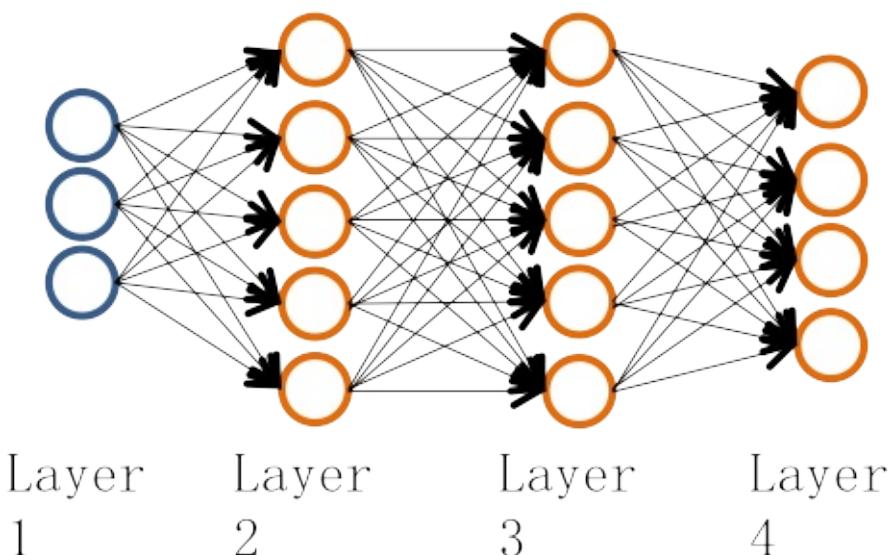
Y2 = 1   2   3   4   5   1   2   3   4   5   1   2   3   4   5   1

Y = 1   0   0   0   0   0   1   0   0   0   0   0   0   1   0   0   0   0   0   0

```

9-2 反向传播算法

要搞清楚反向传播算法，先看看向前传播算法的计算过程。如下图:



推导过程如下:

$$\begin{aligned}
 a^{(1)} &= x \Rightarrow z^{(2)} = \Theta^{(1)} a^{(1)} \\
 a^{(2)} &= g(z^{(2)}) \Rightarrow z^{(3)} = \Theta^{(2)} a^{(2)} \\
 a^{(3)} &= g(z^{(3)}) \Rightarrow z^{(4)} = \Theta^{(3)} a^{(3)} \\
 h_{\Theta}(x) &= a^{(4)} = g(z^{(4)})
 \end{aligned}$$

现在的问题是有了代价函数我们需要计算 $J(\theta)$ 的偏导数。正向计算太难了，于是使用了反向传播算法。

反向传播算法：

$$\begin{aligned}
 \delta^{(L)} &= a_j^{(L)} - y_j \\
 \delta^{(L-1)} &= (\Theta^{(L-1)})^T \delta^{(L)} * g'(z^{(L-1)}) \\
 &\dots \\
 \delta^{(2)} &= (\Theta^{(2)})^T \delta^{(3)} * g'(z^{(2)})
 \end{aligned}$$

具体的计算梯度下降的偏导如下：

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

Set $\Delta_{ij}^{(l)} = 0$ (for all l, i, j)

for $i = 1$ to m

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

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

Compute $a^{(L)}$

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

Back propagation compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$

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

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

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

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

9-3 反向传播算法的解释

对 δ 来说，其实就是表示的偏差，那么所以就是使用偏微分即可表示。

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

另外，如果从正向的传播去思考反向传播也是可以的。因为 i 层的delta偏差，传递给 $i+1$ 层是分散的。

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

这样理解和推导也是可以的。

反向传播算法完全解析

符号定义：

l : 层序号，从1开始

s_l : 第 l 层的激励项数目

s_{l+1} : 第 $l+1$ 层的激励项数目，而并非是 $(s_l) + 1$ 的意思，而仅仅是下标的表示

s_{l+2} : 第 $l+2$ 层的激励项数目

δ : 表示的误差

$$\frac{\partial (J(\Theta))}{\partial (\Theta_{i,j}^{(l)})}$$

本质上就是计算 $\frac{\partial (J(\Theta))}{\partial (\Theta_{i,j}^{(l)})}$ ，那么就直接从这个入手记性推导。

$$\frac{\partial (J(\Theta))}{\partial (\Theta_{i,j}^{(l)})} = \frac{\partial (J(\Theta))}{\partial (a_i^{(l+1)})} \frac{\partial (a_i^{(l+1)})}{\partial (z_i^{(l+1)})} \frac{\partial (z_i^{(l+1)})}{\partial (\Theta_{i,j}^{(l)})}$$

J, a, z, Θ 之间是存在函数关系的，如下描述：

$$J(\Theta) = f(a_i^{(l+1)}); a_i^{(l+1)} = g(z_i^{(l+1)}); z_i^{(l+1)} = f(\Theta_{i,j}^{(l)})$$

所以，能够有上面的 J 和 Θ 的复合函数的连续求偏导。下面就是逐个求解，从易到难：

第一步：

$$\begin{aligned} a_i^{(l+1)} &= g(z_i^{(l+1)}) = \frac{1}{1 + e^{-z_i^{(l+1)}}} \\ \frac{\partial (a_i^{(l+1)})}{\partial (z_i^{(l+1)})} &= \frac{\partial}{\partial (z_i^{(l+1)})} \left(\frac{1}{1 + e^{-z_i^{(l+1)}}} \right) \\ &= (-1) * \frac{1}{(1 + e^{-z_i^{(l+1)}})^2} * e^{-z_i^{(l+1)}} * (-1) \\ &= \left(\frac{1}{1 + e^{-z_i^{(l+1)}}} \right) * \left(1 - \frac{1}{1 + e^{-z_i^{(l+1)}}} \right) \\ &= g(z_i^{(l+1)}) * (1 - g(z_i^{(l+1)})) \\ &= a_i^{(l+1)} * (1 - a_i^{(l+1)}) \end{aligned}$$

第二步：

$$\begin{aligned}
z_i^{(l+1)} &= f(\Theta_{i,j}^{(l)}) = \Theta_{i,0}^{(l)} a_0^l + \Theta_{i,1}^{(l)} a_1^l + \Theta_{i,2}^{(l)} a_2^l + \dots + \Theta_{i,s}^{(l)} a_s^l \\
\frac{\partial(z_i^{(l+1)})}{\partial(\Theta_{i,j}^{(l)})} &= \frac{\partial}{\partial(\Theta_{i,j}^{(l)})}(f(\Theta_{i,j}^{(l)})) \\
&= \frac{\partial}{\partial(\Theta_{i,j}^{(l)})}(\Theta_{i,0}^{(l)} a_0^l + \Theta_{i,1}^{(l)} a_1^l + \Theta_{i,2}^{(l)} a_2^l + \dots + \Theta_{i,s}^{(l)} a_s^l) \\
&= a_j^{(l)}
\end{aligned}$$

第三步也是最难的：

$$\begin{aligned}
J(\Theta) &= f(a_i^{(l+1)}) \\
J(\Theta) &= F(z_1^{(l+2)}) + F(z_2^{(l+2)}) + \dots + F(z_{s+2}^{(l+2)})
\end{aligned}$$

$$\begin{aligned}
z_1^{(l+2)} &= \Theta_{1,0}^{(l+1)} a_0^{(l+1)} + \Theta_{1,1}^{(l+1)} a_1^{(l+1)} + \dots + \Theta_{1,i}^{(l+1)} a_i^{(l+1)} + \dots + \Theta_{1,s+1}^{(l+1)} a_{s+1}^{(l+1)} \\
z_2^{(l+2)} &= \Theta_{2,0}^{(l+1)} a_0^{(l+1)} + \Theta_{2,1}^{(l+1)} a_1^{(l+1)} + \dots + \Theta_{2,i}^{(l+1)} a_i^{(l+1)} + \dots + \Theta_{2,s+1}^{(l+1)} a_{s+1}^{(l+1)} \\
z_{s+2}^{(l+2)} &= \Theta_{s+2,0}^{(l+1)} a_0^{(l+1)} + \Theta_{s+2,1}^{(l+1)} a_1^{(l+1)} + \dots + \Theta_{s+2,i}^{(l+1)} a_i^{(l+1)} + \dots + \Theta_{s+2,s+1}^{(l+1)} a_{s+1}^{(l+1)}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial(J(\Theta))}{\partial(a_i^{(l+1)})} &= \frac{\partial(J(\Theta))}{\partial(z_1^{(l+2)})} \frac{\partial(z_1^{(l+2)})}{\partial(a_i^{(l+1)})} + \frac{\partial(J(\Theta))}{\partial(z_2^{(l+2)})} \frac{\partial(z_2^{(l+2)})}{\partial(a_i^{(l+1)})} + \dots + \frac{\partial(J(\Theta))}{\partial(z_{s+2}^{(l+2)})} \frac{\partial(z_{s+2}^{(l+2)})}{\partial(a_i^{(l+1)})} \\
&= \sum_{k=1}^{s+2} \frac{\partial(J(\Theta))}{\partial(z_k^{(l+2)})} \frac{\partial(z_k^{(l+2)})}{\partial(a_i^{(l+1)})} \\
&= \sum_{k=1}^{s+2} \delta_k^{(l+2)} \Theta_{k,i}^{(l+1)}
\end{aligned}$$

最后的计算结果如下：

$$\begin{aligned}
\frac{\partial(J(\Theta))}{\partial(\Theta_{i,j}^{(l)})} &= \frac{\partial(J(\Theta))}{\partial(a_i^{(l+1)})} \frac{\partial(a_i^{(l+1)})}{\partial(z_i^{(l+1)})} \frac{\partial(z_i^{(l+1)})}{\partial(\Theta_{i,j}^{(l)})} \\
&= \left(\sum_{k=1}^{s+2} \delta_k^{(l+2)} \Theta_{k,i}^{(l+1)} \right) (a_i^{(l+1)} * (1 - a_i^{(l+1)})) (a_j^{(l)}) \\
&= \sum_{k=1}^{s+2} \delta_k^{(l+2)} \Theta_{k,i}^{(l+1)} * a_i^{(l+1)} * (1 - a_i^{(l+1)}) * a_j^{(l)}
\end{aligned}$$

$$\begin{aligned}
\delta_i^{(l+1)} &= \frac{\partial(J(\Theta))}{\partial(z_i^{(l+1)})} = \frac{\partial(J(\Theta))}{\partial(a_i^{(l+1)})} \frac{\partial(a_i^{(l+1)})}{\partial(z_i^{(l+1)})} \\
&= \sum_{k=1}^{s+2} \delta_k^{(l+2)} \Theta_{k,i}^{(l+1)} * a_i^{(l+1)} * (1 - a_i^{(l+1)})
\end{aligned}$$

上面计算的是 l+1 层中第 i 个 δ ，对于第 l+1 层的所有 $\delta^{(l)}$ 的向量化表示：

$$\begin{aligned}\delta^{(l+1)} &= (\Theta^{(l+1)})^T \delta^{(l+2)} * a^{(l+1)} * (1 - a^{(l+1)}) \\ &= (\Theta^{(l+1)})^T \delta^{(l+2)} * g(z^{(l+1)})'\end{aligned}$$

注意：这个表达式，就是NG在课程中直接给出的结论。这里给出了推导。

最终的表述，J对l层所有 Θ 的求导，结论为：

$$\frac{\partial (J(\Theta))}{\partial (\Theta^{(l)})} = \delta^{(l+1)} * (a^{(l)})^T$$

到这里，已经给出了J对 Θ 的偏导的计算方法。如果计算发现还有问题，那就是当 $l=L$ ，第 $L+1$ 层是不存在的，所以 $\delta^{(L)}$ 是需要计算出来的，而不是没有办法推导计算的。对于 $\delta^{(L)}$ 的计算就相对来说，简单很多，因为最后一层可以直接和 $J(\Theta)$ 建立起函数关系，直接求导即可。

$$\begin{aligned}J(\Theta) &= -y \log(a^{(L)}) - (1-y) \log(1-a^{(L)}) \\ \delta_i^{(L)} &= \frac{\partial}{\partial (z_i^{(L)})} (J(\Theta)) = \frac{\partial (J(\Theta))}{\partial (a_i^{(L)})} \frac{\partial (a_i^{(L)})}{\partial (z_i^{(L)})} \\ &= \frac{\partial}{\partial (a_i^{(L)})} (-y_i \log(a_i^{(L)}) - (1-y_i) \log(1-a_i^{(L)})) \frac{\partial}{\partial (z_i^{(L)})} (g(z_i^{(L)})) \\ &= \left(\frac{a_i^{(L)} - y_i}{a_i^{(L)}(1-a_i^{(L)})} \right) * (a_i^{(L)}(1-a_i^{(L)})) \\ &= a_i^{(L)} - y_i\end{aligned}$$

Vectorization \Rightarrow

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

所以，综合以上所有推导，针对一个样本的反向传播算法如下：

$$\begin{aligned}(1) \delta^{(L)} &= a^{(L)} - y \\ (2) \delta^{(l+1)} &= (\Theta^{(l+1)})^T \delta^{(l+2)} * a^{(l+1)} * (1 - a^{(l+1)}) \\ &= (\Theta^{(l+1)})^T \delta^{(l+2)} * g(z^{(l+1)})' \\ (3) \frac{\partial (J(\Theta))}{\partial (\Theta^{(l)})} &= \delta^{(l+1)} * (a^{(l)})^T\end{aligned}$$

9-4 辗转参数

对于每一层的 Θ 来说都是一个矩阵，例如 $s1=10, s2=10, s3=1$ 那么对应的 $\Theta1$ 是 1011 矩阵， $\Theta2$ 是 1011 ， $\Theta3$ 是 $1*11$ 。

对于 δ 来说 $\delta1$ 是 1011 $\delta2$ 是 1011 $\delta3$ 是 $1*11$

对于计算函数来说

```
fminunc(@costFunction, initialTheta, options)
```

现在的问题就是 Θ 变成初始化向量，现在的 Θ 是矩阵。

```
thetaVec = [ Theta1(:); Theta2(:); Theta3(:)];  
DVec = [D1(:); D2(:); D3(:)];  
Theta1 = reshape(thetaVec(1:110), 10, 11);  
Theta2 = reshape(thetaVec(111:220), 10, 11);  
Theta3 = reshape(thetaVec(221:231), 1, 11)
```

9-5 梯度检验

在实现反向传播算法的时候，因为算法过于复杂所以很容易出现细节的错误，而更要命的是细节的错误你可能无法发现。于是，这时候我们需要梯度检验来帮助鉴别梯度下降。

先看看导数的计算方法，对于给定函数 $J(\Theta)$ ，那么

$$\frac{d(J(\theta))}{d(\theta)} = \frac{(J(\theta+\varepsilon) - J(\theta-\varepsilon))}{2\varepsilon}$$

那么，上面的方法应用到 $J(\theta)$ ，就可以用来计算偏导数了。具体的公式如下：

$$\begin{cases} \frac{\partial}{\partial \theta_1} J(\theta_1) = \frac{J(\theta_1 + \varepsilon, \theta_2, \theta_3, \dots, \theta_n) - J(\theta_1 - \varepsilon, \theta_2, \theta_3, \dots, \theta_n)}{2\varepsilon} \\ \frac{\partial}{\partial \theta_2} J(\theta_2) = \frac{J(\theta_1, \theta_2 + \varepsilon, \theta_3, \dots, \theta_n) - J(\theta_1, \theta_2 - \varepsilon, \theta_3, \dots, \theta_n)}{2\varepsilon} \\ \vdots \\ \frac{\partial}{\partial \theta_n} J(\theta_n) = \frac{J(\theta_1, \theta_2, \theta_3, \dots, \theta_n + \varepsilon) - J(\theta_1, \theta_2, \theta_3, \dots, \theta_n - \varepsilon)}{2\varepsilon} \end{cases}$$

通过这个方法能计算出近似的偏导。

对于DVec来说每一个值的计算都与 $d(J(\theta))/d(\theta)$ 比较接近或者相等才是正确的，那么用上面的简化近似算法就可以知道现在计算的Dvec是否正确的。

9-6 随机初始化

这里我们使用的 Θ 是随机化的初始值，不能够全部设置成0，而是通过随机化，将 Θ 的值设置在 $[-\varepsilon, +\varepsilon]$ 之间。

```
Theta1 = rand(10, 11) * (2*INIT_EPSILON) - INIT_EPSILON;  
Theta2 = rand(1, 11) * (2*INIT_EPSILON) - INIT_EPSILON;
```

9-7 放在一起

神经网络要解决几个问题，分别是输入的特征值数量和输出的分类数量，以及中间的隐藏单元。一般来说，对于中间隐藏层的数量越多会效果越好，但是计算速度会越慢。一般都是从一个隐藏层开始处理的。

训练一个神经网络的步骤如下：

1 随机初始化权重

- 2 实现向前传播，得到 $h\theta(x)$ 对于任何一个 x
- 3 实现代码，计算代价函数 $J(\theta)$
- 4 实现向后传播算法计算偏导 $d(J(\theta)) / d(\theta)$

```
for i = 1:m
    执行向前和向后传播算法 得到 a 以及 δ
5 使用梯度检查，数学方法计算  $d(J(\theta)) / d(\theta)$  与 向后传播算法计算的进行比对
6 使用梯度下降或者高级的向后传播来最小化  $J(\theta)$ 
```

9-8 无人车的应用

机器学习的应用。

总结

最后，总结神经网络的算法，需要从后向前推。为了求出每一层的 θ ，所以需要使用梯度下降来进行逐渐的逼近，求出 θ ；为了梯度下降算法来计算 θ ，那么需要计算 $J(\theta)$ 的偏导；为了计算对每一层，注意是每一层的 θ 的偏导，需要使用反向传播来计算 $J(\theta)$ 的偏导；

那么，在从正向的捋顺一下，就是：

先把每一层的所有 θ 随机初始化（注意是每一层），然后计算 $J(\theta)$ 的偏导数，最后使用梯度下降来逼近 θ ，求出最小的 $J(\theta)$ 。记着梯度的方法，就是最终的事：

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

第十课 机器学习的优化方法

Lesson10 机器学习的优化方法

10-1 接下来尝试什么来改善机器学习的效果

当机器学习的效果不是那么理想的时候，有哪些方法可以尝试呢？

- 获取更多的训练集
- 尝试更小的特征集
- 增加特征集
- 尝试使用多项式，例如 (x^2 , x_1x_2 , etc)
- 增加 λ
- 减小 λ

问题是，你可能花费了很长的时间在一个工作上，但是，效果依然不理想。那么，有可能你选择的方向就是错误的。那么，接下来通过机器学习诊断来让你更清楚接下来要做什么。

10-2 评估你的假设函数

当我们计算出来了 $J(\theta)$ 的正确表达式之后，通过评估的方式来知道当前算法时候好坏。评估的方法，步骤如下：

- 1 将测试集随机化
- 2 其中70%用作训练 $J(\theta)$, 30%用作评估

对于线性回归来说，评估方法如下：

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

$test$ 下标表示测试集的意思

$h(\theta)$ 是通过线性回归得到的函数

对于逻辑回来说，评价方法如下：

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

$$err(h_{\theta}(x), y) = \begin{cases} 1 & // J(\theta) \text{ is error. } h_{\theta}(x) \geq 0.5, Y = 0; \text{otherwise} \\ 0 & // J(\theta) \text{ is right. } h_{\theta}(x) \leq 0.5, Y = 1; \text{otherwise} \end{cases}$$

$$TestErr = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} err(h_{\theta}(x_{test}^{(i)}), y_{test}^{(i)})$$

10-3 模型选择：训练集，交叉验证集，测试集

对于模型的选择，例如下面的线性回归方程，d(degree)表示x的幂，d=1,2,3...10

$$1. h_{\theta}(x) = \theta_0 + \theta_1 x \Rightarrow \Theta^{(1)} \Rightarrow J_{test}(\Theta^{(1)})$$

$$2. h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \Rightarrow \Theta^{(2)} \Rightarrow J_{test}(\Theta^{(2)})$$

...

$$10. h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_{10} x^{10} \Rightarrow \Theta^{(10)} \Rightarrow J_{test}(\Theta^{(10)})$$

那么，现在通过对Err(J_test($\Theta^{(5)}$))效果最好，那么，我们选择了 $\Theta^{(5)}$ 但这并非是正确的，因为现在变成了对测试集的最好的拟合。所以，需要重新进行数据集的划分。

分别是：训练集占数据总量60%，交叉验证集占20%，测试集占20%。评估的方法，依然是寻找最小偏差。

$$\begin{aligned} Training Set &\Leftrightarrow \left\{ \begin{array}{l} (x^{(1)}, y^{(1)}) \\ (x^{(2)}, y^{(2)}) \\ \dots \\ (x^{(m)}, y^{(m)}) \end{array} \right. \\ Cross Validation Set &\Leftrightarrow \left\{ \begin{array}{l} (x_{cv}^{(1)}, y_{cv}^{(1)}) \\ (x_{cv}^{(2)}, y_{cv}^{(2)}) \\ \dots \\ (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})}) \end{array} \right. \\ Test Set &\Leftrightarrow \left\{ \begin{array}{l} (x_{test}^{(1)}, y_{test}^{(1)}) \\ (x_{test}^{(2)}, y_{test}^{(2)}) \\ \dots \\ (x_{test}^{(m_{test})}, y_{test}^{(m_{test})}) \end{array} \right. \end{aligned}$$

偏差公式：

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:

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

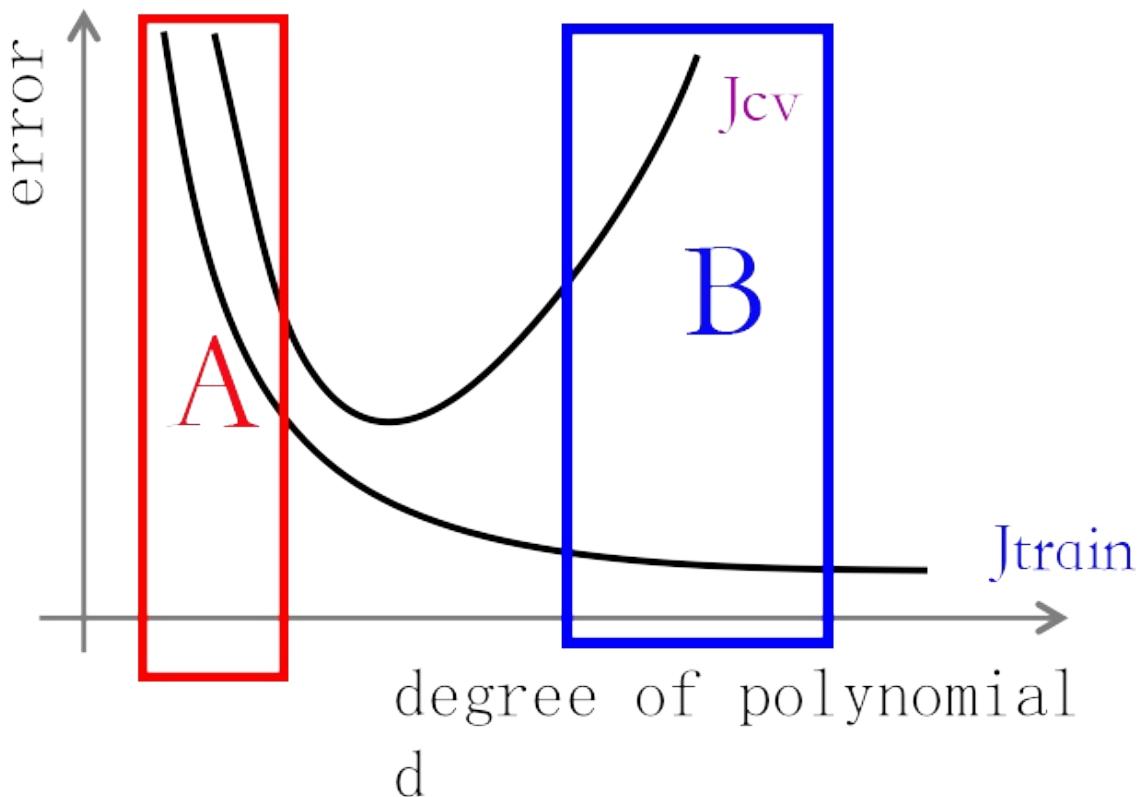
到现在，对机器学习其实有了一个总体框架的认识。就是寻找拟合，通过训练集，写出

拟合标准就是 $J(\theta)$ ，剩下的工作是寻找 $J(\theta)$ 的最小化的 θ ；因为在选择拟合函数 $h(\theta)$ 的时候，可能有多种选择，这没关系，把每一种情况计算出来，使用 $J_{cv}(\theta)$ 来寻找最好的 $h(\theta)$ 模型。最后通过，测试集来进行测试。

对于模型的选择，就是根据某一个变量划分的模型，选择最好的那一个模型。事实上就是，绘制变量和 $J_{test}(\theta)$ 的曲线，来找到 $J_{test}(\theta)$ 最小的那个。对于线性回归来说，是确定维度，那么维度 d 和 $J_{test}(\theta)$ 就构成了函数曲线。

10-4 偏差与方差

求解的假设函数，要么是偏差要么是方差大。这是针对训练集和测试集来说的。如果是偏差，说明是欠拟合，如果是误差大说明是过拟合。所以搞清楚这个问题，非常重要，给了我们一个指示器，进行优化。



横坐标： d 表示多项式的最高次幂
纵坐标：表示误差

我们看到，当 d 的取值很小的时候，对于训练集来说产生的误差比较高，同样对于验证集来说也很高，这是一种欠拟合的形态。对应 A 的区域。再看 B 的区域，因为 d 的取值很高，所以对于训练集来说能够很好的拟合，但是，对于验证集来说，却误差很大。这说明是过拟合。

欠拟合，偏差的定义如下：

1. $J_{train}(\theta)$ 很高
2. $J_{cv}(\theta)$ 约等于 $J_{train}(\theta)$

过拟合，误差的定义如下：

1. $J_{train}(\theta)$ 很小
2. $J_{cv}(\theta)$ 远远大于 $J_{train}(\theta)$

上面的图告诉我们如何来确定是高偏差还是高误差。然后，后面的优化再对症下药。因为每一个参数的调整，要么产生偏差要么产生误差。

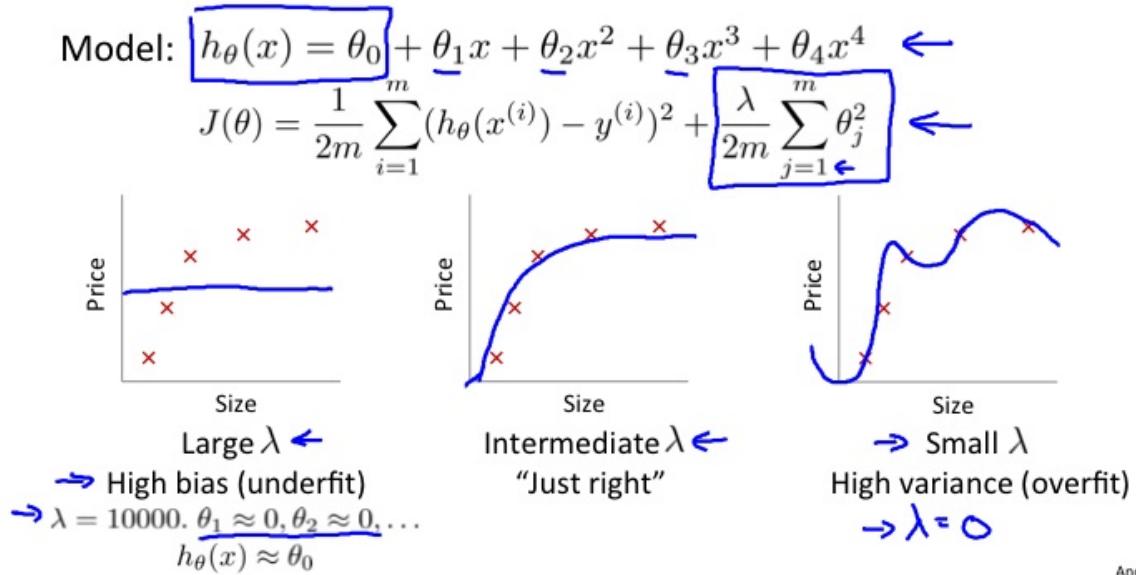
10-5 正规化

正规化可以有效的解决过拟合的问题。同样，正规化可以有效的处理偏差和方差的问题。看下，正规化的表达式。

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

通过这个表达式，我们可以理解到，正则化是为了在选定的d的情况下进行再次优化。是针对特定的d的细致优化。但是即使对于选定的d，依然可以进行再优化，那就是通过上面的正规化方法。通过对 θ 的惩罚来达到更加好的拟合。

Linear regression with regularization



上面的图像展示了对于同一个 $d=4$ 来说，选取不同的 λ ，可以产生不同的拟合。选择恰当的 λ 也可以实现正常的拟合。

那么对于误差公式来说：

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:

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

特别注意：训练集误差是没有正规化表达式的。正规化表达式是中间的过程优化的计算思想，而对于误差的最终描述是没有这一项的，这一点要搞清楚。

之间的图像和变化关系又是什么样的呢？

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

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

计算的过程如下表：

λ 取值	计算步骤1	计算步骤2	计算步骤3
0	$\min(J(\theta))$	得出 θ^0	得出 $J_{cv}(\theta)$
0.01	$\min(J(\theta))$	得出 θ^1	得出 $J_{cv}(\theta)$
0.02	$\min(J(\theta))$	得出 θ^2	得出 $J_{cv}(\theta)$
0.04	$\min(J(\theta))$	得出 θ^3	得出 $J_{cv}(\theta)$
...	$\min(J(\theta))$...	得出 $J_{cv}(\theta)$
10	$\min(J(\theta))$	得出 θ^9	得出 $J_{cv}(\theta)$

那么，从中选择最小的 $J_{cv}(\theta)$ 的值，就是我们需要的 θ 取值。

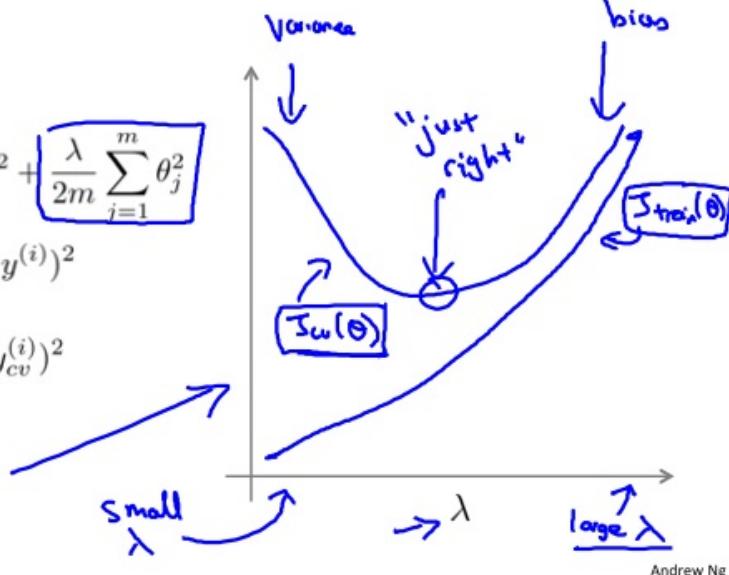
通过上面的表格，通过图像来看就会更加直观。将 $J_{train}(\theta)$ 和 $J_{cv}(\theta)$ ，两个误差函数进行绘制。

Bias/variance as a function of the regularization parameter λ

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

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

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

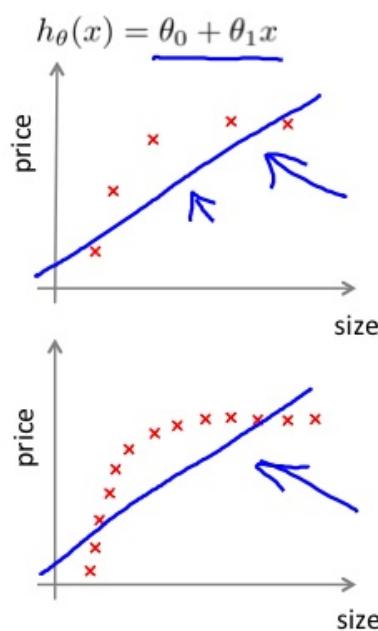
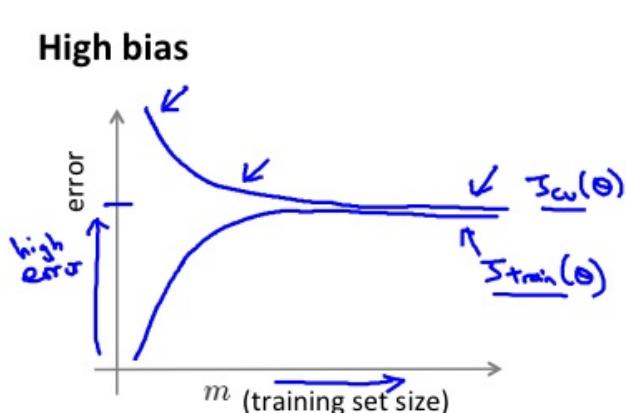


可以看到对于 λ 很小的时候，产生的开口很大，因为对于 θ 来说，没有惩罚，所以还是会高误差；而对于 λ 的增加，会对 θ 惩罚很大，最后会出现很接近。对于我们来说，选择中间的点是最合理的。

10-6 学习曲线

绘制学习曲线是非常有用的方法。能够检测你的学习算法，以及所处的问题是偏差还是方差问题。

高偏差（欠拟合）解释



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

这张图是对高偏差的学习曲线的一个描述。坐标系描述，如下：

error：纵轴表示误差数值

m : 表示数据集个数。这一点需要注意，如果是训练集表示的训练集个数；而交叉验证集是使用

学习曲线的解释：对于每一个训练集，训练出来的参数，应用到整个交叉验证集，计算出训练集误差；再应用到训练集，计算出训练误差。

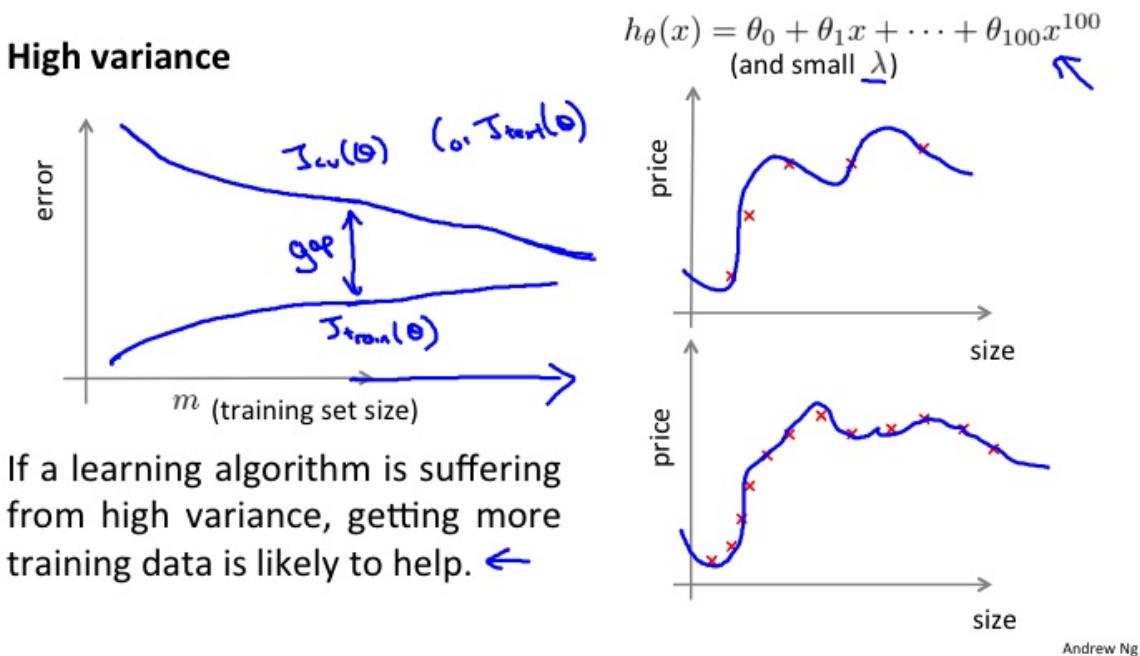
对于图上的高偏差（欠拟合）的理解。对于 $J_{\text{train}}(\theta)$ 在数据量很小的时候，也就是 $m=1, 2, \dots$ 很小的时候，因为 $h(\theta)$ 能够对训练集数据很好的拟合，所以 $J_{\text{train}}(\theta)$ 就会很小；随着 m 的增加，会使得 $J_{\text{train}}(\theta)$ 的偏差会逐渐增大；但是，当 m 增加到一定的数量，因为欠拟合，所以 $J_{\text{train}}(\theta)$ 维持在一个较高的位置，但是不会再急速上升，而是平稳下来。

而对于 $J_{\text{cv}}(\theta)$ 的理解是这样的。当数据量很小的时候， $m=1, 2, \dots$ 的时候，因为验证集的数据很少，所以不会很好的与 $h(\theta)$ 拟合，就会产生较大偏差，也就是 $\text{error}(J_{\text{train}}(\theta))$ 会比较大；随着数据量的增加，逐渐的拟合度增加，误差值就会逐渐的变小；当数据量大到一定的程度的时候。 $J_{\text{cv}}(\theta)$ 和 $J_{\text{train}}(\theta)$ 就会很接近处于一致的位置。

这就是对于高偏差的解释。所以对于学习曲线来说，高偏差的特征是：

- 1 $J_{\text{train}}(\theta)$ 和 $J_{\text{cv}}(\theta)$ 比较接近，而且处于较高位置
- 2 对于这样的情况来说，增加数据集是没有任何意义的

高误差（过拟合）解释



对于高误差（过拟合）的理解。对于 $J_{\text{train}}(\theta)$ 在数据量很小的时候，也就是 $m=1, 2, \dots$ 很小的时候，因为 $h(\theta)$ 能够对训练集数据很好的拟合，所以 $J_{\text{train}}(\theta)$ 就会很小；随着 m 的增加，会使得 $J_{\text{train}}(\theta)$ 的偏差会逐渐增大；对于这一点上与高偏差是基本一致的。

而对于 $J_{\text{cv}}(\theta)$ 的理解是这样的。当数据量很小的时候， $m=1, 2, \dots$ 的时候，因为验证集的数据很少，所以不会很好的与 $h(\theta)$ 拟合，就会产生较大偏差，也就是 $\text{error}(J_{\text{train}}(\theta))$ 会比较大；随着数据量的增加，逐渐的拟合度增加，误差值就会逐渐的变小；而当训练集和验证集的数据使用完的时候，对于 $J_{\text{train}}(\theta)$ 、 $J_{\text{cv}}(\theta)$ 和 $J_{\text{train}}(\theta)$ 就会在一个位置保持很大的差距，也就是 gap 。

但是通过图像，可以看出，如果继续增加训练集和验证集的数据，二者会继续下降，所以增加数据集是有用的一个方法。

这就是对于高误差的解释。所以对于学习曲线来说，高误差的特征是：

- 1 $J_{\text{train}}(\theta)$ 和 $J_{\text{cv}}(\theta)$ 之间的差距较大。
- 2 对于这样的情况来说，可以增加数据集来使得结果拟合的更好。

10-7 回顾和总结

最初提出的问题的解决方案如下：

序号	解决方案	适用的问题
1	获取更多的训练集	高误差（过拟合）
2	尝试更小的特征集	高误差（过拟合）
3	增加特征集	高偏差（欠拟合）
4	尝试使用多项式，例如 $(x^2, x_1x_2, \text{etc})$	高偏差（欠拟合）
5	增加 λ	高误差（过拟合）
6	减小 λ	高偏差（欠拟合）

对于当前是高偏差还是高误差的判断，使用学习曲线来判断。

神经网络的优化

前面所描述的都是基于线性回归的描述，那么，基于神经网络的优化又是什么呢？面临着两个问题。

一、隐藏层数的选择。究竟是几层。这个问题的处理与多项式选择中的最高次幂是一样的。可以通过 $J_{\text{cv}}(\theta)$ 的比较来得出最好的 θ 。

二、使用正规化来进行优化。一个大型的神经网络配置正规化修订，要比一个小型的神经网络更加准确。

可以通过正规化来进行过拟合的优化。

第十一课 系统设计

Lesson11 系统设计

11-1 垃圾邮件分类

这一章主要介绍在我们设计机器学习系统时候，我们着重考虑的。

现在要设计一个区分垃圾邮件的系统。那么，第一种考虑方法是，我们选择所有可能成为垃圾邮件的单词，作为特征值。例如：

特征值 垃圾邮件单词 非垃圾邮件单词

x1	buy	NG
x2	sell	you
...

然后通过这些特征量根据不同的邮件进行训练。

如何做能够降低你的误差？

- 收集更多的数据
- 从邮件的地址出发
- 从邮件的正文内容入手

你可能使用头脑风暴，这样做其实已经很不错了。也有可能你会突发奇想，想出一个特征来。

11-2 错误分析

当你开始一个机器学习的项目的时候，不是先想出很复杂的算法。而是以最快的速度构建出一个机器学习的方法。所以，正确的设计机器学习系统的姿势如下：

- 1 快速的实现一个系统。并测试你的交叉验证集。
- 2 绘制学习曲线。搞清楚是高偏差还是高误差问题
- 3 错误分析。例如，分析那些被系统过滤出来的垃圾邮件和非垃圾邮件，找出其中可能存在的系统性规律。通过人工检查这些错误，可能知道是什么类型的错误，例如可能是药物邮件等。另外，可以看看哪些其他的特征量可以帮助我们，例如有很多感叹号。

错误分析的本质就是从哪些被错误分类的数据中再次寻找规律，并将这些规律做成特征。

除了这些，还需要一种能够评估你的机器学习好坏的数值方法，例如误差率，也就是增加了一种改善的方式，误差率如何。

11-3 误差度量（偏斜类）

例如，在计算癌症的逻辑回归中，你计算的癌症患者的误差是1%，然而，实际上在所有的病人中得癌症的只有0.05%。那么，你的这个误差就显得不那么和谐，甚至是错误的。出现这种问题的根源在于其中一个样本的数量非常少，这种样本被叫做偏斜类。因此当你的数据中

有偏斜类，那么产生的问题就是很小的误差，同时就算算法优化了，得到了误差的提升也是无法确定算法是否提升了。对于偏斜类问题不能使用上节中的误差率进行分析。

因此当我们遇到具有偏斜类的数据集的时候，采用的是：查准率和召回率进行比较分析。下面的例子，介绍查准率和召回率。

对于癌症分类， $y=1$ 表示有； $y=0$ 表示没有

实际 $y=1$ 实际 $y=0$

预测 $y=1$ 真阳性 假阳性

预测 $y=0$ 假阴性 真阴性

查准率 = 真阳性 / 预测阳性 = 真阳性 / (真阳性 + 假阳性)

召回率 = 真阳性 / 实际阳性 = 真阳性 / (真阳性 + 假阴性)

通过查准率和召回率来看具有偏斜类算法就更好。

11-4 平衡量查准率和召回率

例如：

预测 1 if $h(x) \geq 0.5$

预测 0 if $h(x) < 0.5$

高查准率，令 $h(x) = 0.7$ 或者 0.9 可以更加的容易查出。

高召回率，令 $h(x) = 0.3$

当我们进行股票的涨跌预测的时候，使用高查准率更好。

那么，如何选择算法呢？需要对比不同算法下的查准率和召回率。

N/A 查准率(P) 召回率(R)

算法1 0.5 0.4

算法2 0.7 0.1

算法3 0.02 1.0

那么，究竟如何选择哪个算法呢？使用如下公式：

$$F = 2 \cdot PR / (P+R)$$

11-5 数据

劣质算法，在数据量很大的情况和优质算法可能差不多。这种情况往往是因为忽略数据的上下文中的一些特点。

那么，我们要做的是在预测一件事情的时候，将现有的特征量假设告诉一个这方面的专家，那么，他能否预测出结果呢？例如，仅仅知道房子的大小，专家能否预测价格呢？显然不能。所以，如果你的机器学习特征量使用的是仅仅房子大小，显然算法和算法之间是无法区分好坏的。

第十二课 支持向量机(SVM)

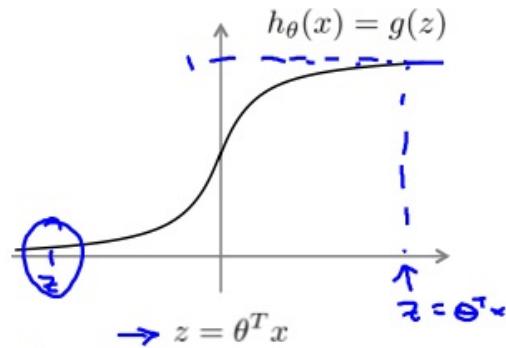
Lesson12 支持向量机(SVM)

12-1 优化目标

支持向量机(SVM)在进行非线性学习的时候，显得更加强大。在很多工业和学术上使用的监督学习就是SVM。

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

Andrew Ng

最后计算的SVM优化函数如下：

$$\min_{\theta} C \sum_{i=1}^m [y^{(i)} \cos t_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \cos t_0(\theta^T x^{(i)})] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

$$\cos t_1(\theta^T x^{(i)}) = -\log h_{\theta}(x^{(i)})$$

$$\cos t_0(\theta^T x^{(i)}) = -\log(1 - h_{\theta}(x^{(i)}))$$

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

Hypothesis:

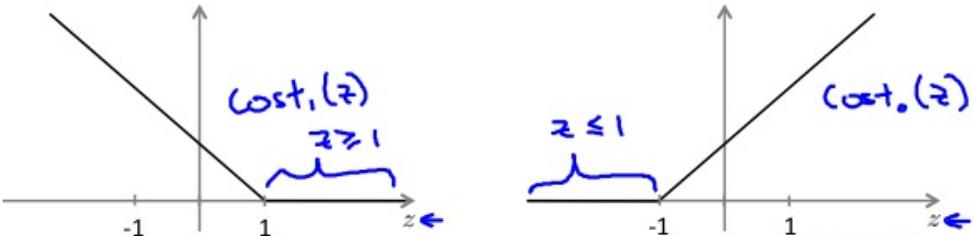
$$h_{\theta}(x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

注意C是一个常量，而最后的正规化的 λ 没有了，但这其实是一回事，相当于乘以 $1/\lambda$.

12-2 大间距分类器

Support Vector Machine

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



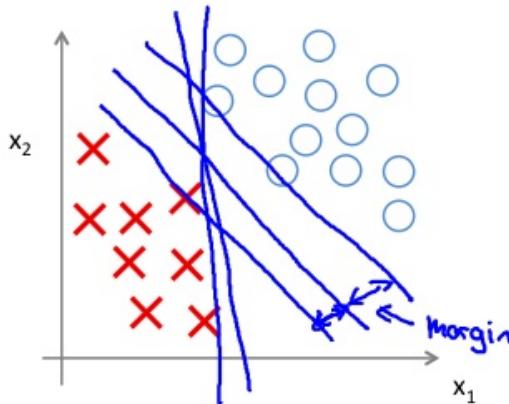
\rightarrow If $y = 1$, we want $\theta^T x \geq 1$ (not just ≥ 0)
 \rightarrow If $y = 0$, we want $\theta^T x \leq -1$ (not just < 0)

$$C = 100,000$$

Andrew Ng

上图，我们看到在最小化代价函数的时候，对于 $y=1$ 时， $\theta^T x \geq 1$ 则第一项为0；当 $y=0$ 时， $\theta^T x \leq -1$ ，则第一项为0. 所以当 C 足够大的时候，例如 $C=100000$, 那么 \min 就是希望第一项为0才能达到最小化。这样就形成了一个决策边界。

SVM Decision Boundary: Linearly separable case



Large margin classifier

Andrew Ng

从图上可以看到，其实本质上求解得到两种样本对于中间的决策边界的距离最大。注意，上面的所有分析是建立在 C 是很大的数值的假设基础上的。

那么，对于其他的异常情况，例如在正样本中有负样本；负样本中有正样本。这种Case是否能够处理呢？这是可以的，当选取的 C 不是很大的时候，是可以很好的拟合的。

12-3 大距离分类器背后的数学原理

证明前提是 C 足够大，这样就变成了：

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

$$\begin{cases} \theta^T x^{(i)} \geq 1 & \text{if } y^{(i)} = 1 \\ \theta^T x^{(i)} \leq -1 & \text{if } y^{(i)} = 0 \end{cases}$$

所以最小化了 θ 向量的模即可。

而对于 $\theta^T x^{(i)}$ 本质上是两个向量的内积 $\theta^T x^{(i)} = p^{(i)} \|\theta\|$ ，其中 p 是 $x^{(i)}$ 在 θ 上的投影 $p = \|x^{(i)}\| * \cos\alpha$ 。

变换后的表达式如下：

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

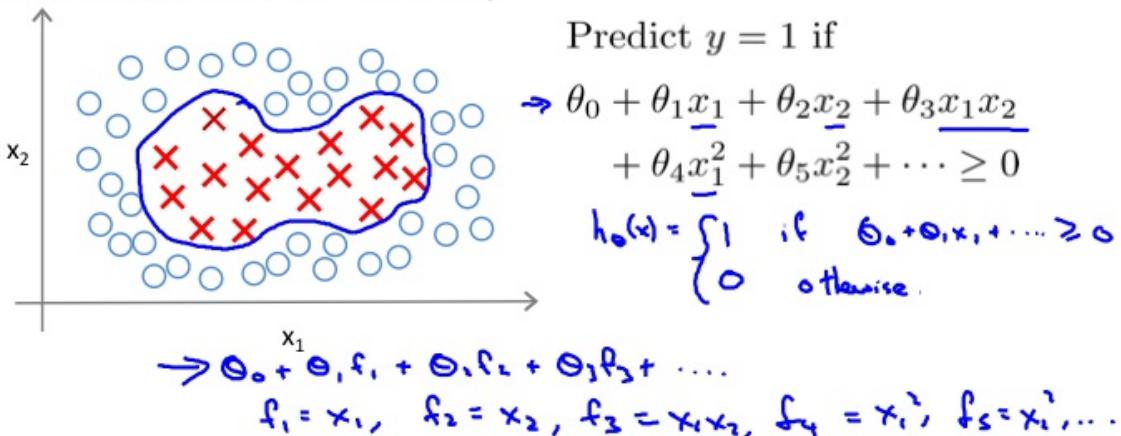
$$\begin{cases} p^{(i)} \|\theta\| \geq 1 & \text{if } y^{(i)} = 1 \\ p^{(i)} \|\theta\| \leq -1 & \text{if } y^{(i)} = 0 \end{cases}$$

使用图形分析方式，我们知道边界线与 θ 向量是垂直的，所以要想 $\|\theta\|$ 小， $p^{(i)} \|\theta\| \geq 1$ ，所以只有 $p^{(i)}$ 较大的时候，才能够使得 θ 较小。而 p 是 x 到 θ 的投影，因为 θ 垂直于边界，所以 p 就是样本 x 到决策边界的距离，而 p 越大 θ 越小，所以最小化代价函数。经过这些分析，就清楚大距离原理。

关键点是将 x 到边界的距离进行可以运算化的过程。

12-4 核函数 I

Non-linear Decision Boundary



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

Andrew Ng

现在我们要解决的是上图的非线性的分类问题。假设在 $y = 1$ 的情况下 $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \dots \geq 0$ ，使用这个多项式来代替 $\theta^T x$ 在计算量上是很大的，因为是多项式。所以进行变通：

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

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \theta_4 f_4$$

$$f_1 = x_1, f_2 = x_2, f_3 = x_1 x_2, \dots$$

那么f是什么？f表示的是x的近似值函数，也就是核函数，选择高斯核函数如下：

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

$$\begin{cases} \text{if } x \approx l^{(i)}, f_i \approx 1 \\ \text{if } x \gg l^{(i)}, f_i \approx 0 \end{cases}$$

我们看到，当x与l接近的时候 $f=1$ 表示很“相似”；当x与l差得很远的时候， $f=0$ 表示相差很大。

12-5 核函数 II

标记点l如何选择？通过将所有的样本点当做l，作为初始的选择，所以l有m个，与样本点是一样的。

SVM的核：

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

$$\text{Choose } l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)},$$

Given example x:

$$\begin{cases} f_1 = \text{similarity}(x, l^{(1)}) \\ f_2 = \text{similarity}(x, l^{(2)}) \\ \dots \\ f_m = \text{similarity}(x, l^{(m)}) \end{cases}$$

这里需要解释一下 x，是指多个特征量组成的向量。如果有3个特征量，那就是 x_1, x_2, x_3 。所以对于 f来说，是x向量与l向量之间的距离关系。

对于 $x^{(1)}$ 来说，产生的 $f^{(1)}$ 如下：

For $x^{(1)}$

$$\begin{cases} f_1^{(1)} = \text{similarity}(x^{(1)}, l^{(1)}) \\ f_2^{(1)} = \text{similarity}(x^{(1)}, l^{(2)}) \\ \dots \\ f_m^{(1)} = \text{similarity}(x^{(1)}, l^{(m)}) \end{cases} \Rightarrow f^{(1)} = \begin{bmatrix} f_1^{(1)} \\ f_2^{(1)} \\ \dots \\ f_m^{(1)} \end{bmatrix}$$

最终的支持向量机如下：

Hypothesis: Give x , compute $f \in R^{m+1}$

Predict " $y = 1$ " if $\theta^T f \geq 0, \theta_0 f_0 + \dots + \theta_m f_m, f_0 = 1$

Training:

$$\min_{\theta} C \sum y^{(i)} \cos t_1(\theta^T f^{(i)}) + (1 - y^{(i)}) \cos t_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

引入支持向量机的关键点，在于SVM在最小化代价函数的过程中的优化计算。这才是引入支持向量机的关键。只有这种方式，才能够快速的计算。所以SVM解决的是逻辑回归的非线性分类器的多项式快速计算问题。

对于在12-2的距离解释，实际上是一种解释。

关于， C ($C=1/\lambda$)，

很大的 C ， λ 很小，意味着，低偏差，高误差
较小的 C ， λ 很大，意味着，高偏差，低误差

关于， δ^2

很大的 δ ，意味着核函数变化度平滑，所以高偏差，低误差
较小的 δ ，意味着核函数变化度剧烈，所以低偏差，高误差

12-6 使用SVM

使用SVM软件包,例如 liblinear, libsvm等

- 1 选择 C
- 2 选择核函数
- 3 不选择核函数直接使用 $\theta^T x \geq 0$

选择的方法，

- 1 如果 n (特征值) 很多, m (数据量) 很小。可以考虑使用线性kernel，也就是不用kernel
- 2 如果 n 很多， m 很多，并且样本可能是非线性的分类，考虑使用高斯核函数。

对于其他的核函数的使用要小心，必须符合 "Mercer's Theorem"，因为这些是被SVM优化的。

逻辑回归和SVM之间的选择。如果 $n \geq m$ ，使用线性核函数的SVM; 如果 n 很小 (1-1000), m 中等程度 (10-10,000) 使用SVM并且高斯核函数. 如果 n 很小($n=1-1000$), m 很大 (50,000+)，考虑使用逻辑回归或者不带核函数的SVM，因为带有核函数，会导致计算量非常大。

SVM的另外好处是比神经网络运行的速度要快。而且SVM是凸的，所以能够得到全局最优化。这一点很多时候比神经网络更好。

算法虽然很重要，但是更多的数据更重要。你需要的是学会处理误差等技能。

第十三课 聚类

Lesson13 聚类

13-1 非监督学习

监督学习和非监督学习的区别在于，监督学习的每个数据是有标签，例如是0还是1，是有效还是无效；而非监督学习的数据是没有标签的，由程序自动产生分类。因此对于，非监督学习的训练集来说，就不会写y，因为y就是标签。

训练集: $\{x^1, x^2, \dots, x^m\}$

第一个学习的是聚类算法，进行探查数据的内部结构。

13-2 K均值算法(K-means)

K均值算法:

Input Arg 1: K, (分成的聚类的数量) Input Arg 2: 训练集 $\{x^1, x^2, \dots, x^m\}$ $x^i \in R^n$
($x_0 = 1$)

执行:

随机初始化K簇中心 $u_1, u_2, \dots, u_K \in R^n$

循环 { for $j = 1$ to K for $i = 1$ to m $c = \|x_i - u_j\|$ 最小的 c , 是当 $j=n$, 将 x_n 归为 j 的簇

```
// 重新分配中心
for k = 1 to K
    uk = 所有属于K簇的计算平均值
}
```

// 特殊情况，如果某个簇没有点，那么将会将这个簇删除掉，就变成了K-1个簇。

思考，监督学习与非监督学习的其他区别，是用途区别，非监督学习能够对所有数据进行分类，然后再打上标签。而监督学习是打上标签之后，再研究其中的曲线拟合。

13-3 优化目标

$c^{(i)}$ = index of cluster ($1, 2, \dots, K$) to which example $x^{(i)}$ is currently assigned.

μ_k = cluster centroid k ($\mu_k \in R^n$)

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

example: $x^{(i)} \rightarrow 5$ (index) $\Rightarrow c^{(i)} = 5$ $\mu_{c^{(i)}} = \mu_5$

Optimization objective:

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

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

13-4 随机初始化

现在的问题是避免局部最优化。基本思想是，进行多次的随机初始化K，而选择min(J)的那个K值。

```
for i = 1 to 100 {
    随机初始化 K均值 中心点
    运行K均值算法，得到 c, u
    计算代价函数： J(c, u)
}
```

选择minJ(c, u)，所计算出的值。

如果K属于2-10这个规模，会得到较好的效果；而如果K很大，那么，其实后面的优化并不能得到更多好的优化结果。

13-5 选择簇的数量

对于K的数量的选择，一是肘部分析法；二是根据商业规则来设定。

肘部分析，是将K=1, 2, 3, ... 和J，找到拐点，但是这不总是有效的。所以，从商业角度分析可能更好。

第十四课 降维

Lesson14 降维

14-1 数据压缩

降维的过程可以将多个特征量合并成一个，这样就产生了数据压缩特性。

14-2 数据可视化

降维的好处是多个维度的数据能够降低3维或者2维，在这个维度上可以对数据进行可视化。

14-3 降维算法-主成分分析法(PCA)

OCTAVE算法:

```
 $\Sigma = (x(1) * x(1)' + x(2)*x(2)' + \dots + x(m)*x(m)')/m;$ 
//  $\Sigma = (X'*X)/m$  如果X是所有行向量组成的数据 上面是基于x都是列向量的算法
[U, S, V] = SVD( $\Sigma$ );
Ureduce = U(:, 1:K);
Z = U'*X;
```

最终: $Z \in R^{k1}$ 而 X 属于 R^{n1} ，所以实现了降维。

注意: $x(i)$ 是列向量

14-5 选择K的方法

上面给出了降维的方法，那么究竟从n维降到多少维才是合适的呢？遵从下面的方法：

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

这表示99%的误差被保留，这是很高的标准。可以降低到0.05，那就是95%的误差被保留。

那么在OCTAVE中如何计算呢？

```
[U, S, V] = SVD( $\Sigma$ );
```

计算的结果S是一个对角矩阵，所以上面的公式就变成了：

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

前k个对角矩阵的相加和除以所有对角元素的相加和。

而这个算法事实就是：

循环k=1:n
计算每一个 除数因子是否 >= 0.99
直到找到最小的k，满足这个条件

14-6 原始数据重建

我们如何通过Z重建X呢？通过

$$Z = U' * X \text{ 得出 } X = U^* Z.$$

注意最后计算的X和原先是有差别的称之为X_approx，是一种近似，因为投影产生的信息丢失是无法完全映射回去的。就像上节14-5中所描述的X_approx那样。

这里还有个疑问就是：

$$X = \text{pinv}(U')^* Z,$$

难道 $\text{pinv}(U') = U$ ？在NG的课上面并没有这个解释。

14-7 PCA在监督学习中的应用

PCA在监督学习中，可以将数据降维从而加速运算。具体的方法是：

1 先对X进行降维到Z, 映射记做 map 2 进行监督学习的就是(Z, Y)

但这里需要注意的是，这里仅仅是用作训练的，用作训练参数.将在训练集上得到这种映射关系，应用到交叉验证集和测试集，将Z映射到Z，再带入到假设函数中进行预测。

PCA有一些错误的用法？要特别注意不要使用

1、用来处理过拟合。因为过拟合是因为特征量过多，所以通过PCA来降低特征量。因为处理过拟合的最好的方法是正规化而不是PCA,虽然有些时候PCA能够解决过拟合的问题。对于我们使用PCA来说的，目的就是加快算法的速度和在存储空间上要求过于巨大的时候才考虑PCA，正常的情况下我们最好选择原始数据，因为PCA必定是会损失一些信息的。例如在图像方面，可以考虑PCA，因为图像大小越大产生的特征越多越复杂。

总结

这里主要讲述了PCA的方法，进行降维处理。所谓的降维就是将多个特征值合并成一个。这么做的好处，是减少的维度，加速的计算，而主要使用的算法就是PCA。

这里唯一要说明的就是，线性回归与降维很像，但是，绝对不是一个东西。

参考资料

[主成分分析](#)

[PCA&SVD](#)

第十五课 异常检测

Lesson15 异常检测

15-1 问题的动机

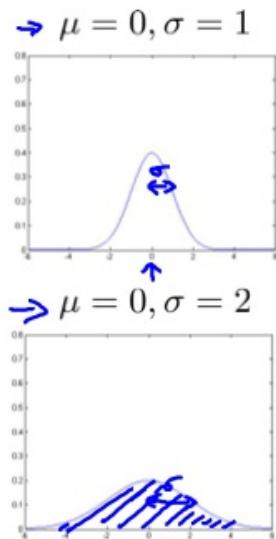
在一堆数据中，根据一定的阈值寻找出来那个异常的数据。许多购物网站就是这样来识别异常用户的，收集所有用户的行为特征，包括打字速度，页面停留时间等等，来找出哪些用户是异常的。

或者计算机集群，监控内存使用率，CPU使用率等，来找出异常的机器。

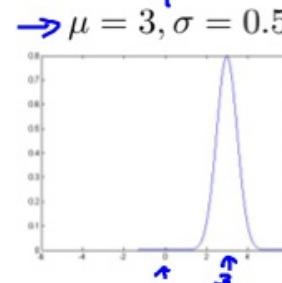
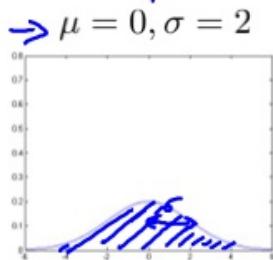
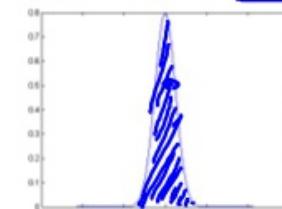
15-2 高斯分布

标准差 δ 决定了高斯分布的宽度。

Gaussian distribution example



$\rightarrow \mu = 0, \sigma = 0.5$ $\sigma^2 = 0.25$



Andrew Ng

15-3 异常检测算法

异常检测的算法是利用高斯概率分布来计算每一个特征量的概率，再相乘求出整体特征量的方法。但，这里要求特征量必须是独立的。但是，在机器学习中，即使不独立，效果依然不错。

算法如下：

1 选择 x_i

2 计算 μ 和 δ

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

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

3 给定样本 x ，计算 $p(x)$:

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

15-4 开发一个异常检测系统

具体做法，是引入标签的，交叉验证集和测试集。因为属于偏斜率数据，所以，计算查准率和召回率进行比较算法的优劣。

本来是非监督学习，现在引入了监督学习的评价方法。

15-5 异常检测 VS. 监督学习

通过上节，我们可以知道数据究竟是 $y=0$ 还是 $y=1$ ，也就是带有标签的数据。那么，既然数据可以是带有标签的，为什么不使用逻辑回归或者神经网络这些监督学习来预测是否是OK的？

区别	异常检测	监督学习
1	非常小的正样本和非常大的负样本；反之，亦然	正负样本数量差不错
2	异常的种类很多 不确定，所以无法进行预测	异常或者正常的情况是重复的，通过学习可以学到

那么，如果异常的数据很多，可以考虑尝试从异常检测转换到监督学习。

15-6 如何选择特征量

有些特征量可能不符合高斯分布，那么可以通过一些函数变换来产生高斯分布。例如:

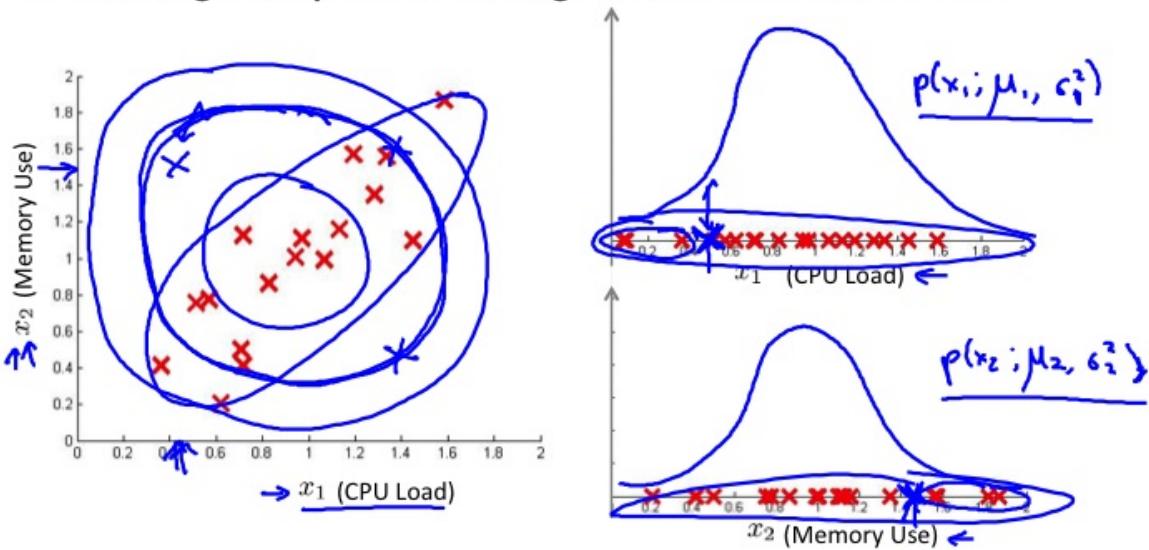
$$y = \log x; \quad y = x^*x$$

或者

$$y = x_1/x_2$$

15-7 多元高斯分布

Motivating example: Monitoring machines in a data center



Andrew Ng

多元高斯分布是为了解决当有两个特征值相关联的好方法。看上图，对于上面蓝色的异常点，对于 x_1, x_2 来说是正常的，所以无法被检测到。但是，放到所有的数据中观察，就会发现是异常的点。

15-8 多元高斯分布应用

方程定义如下：

Anomaly detection with the multivariate Gaussian

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

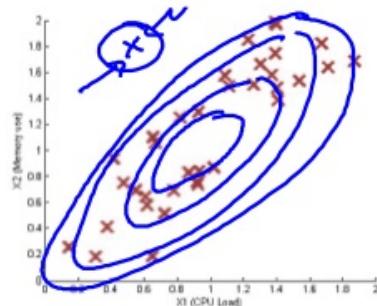
$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

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

2. Given a new example x , compute

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

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



Andrew Ng

第十六课 推荐系统

Lesson16 推荐系统

16-1 形式化问题

学术界关注的很少关于推荐系统。

电影推荐问题，是系统对某个用户根据用户对现有电影的评分来对用户进行推荐其他电影的问题。那么，这个问题如何形式化表述呢？

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

n_u = 用户数量

n_m = 电影的数量

$r(i, j) = 1$ 如果用户 j 给电影 i 打分了是 1

$y(i, j) = 0, 1, \dots, 5$ 用户 j 给电影 i 打的分 是 $0, 1, 2, 3, 4, 5$

通过这些定义，就可以对推荐系统进行定义了。推荐系统要做的就是对上面"?"的地方给出推荐的分数。

16-2 基于内容的推荐

基于内容的推荐的原理是通过内容来寻找特征值，例如将电影的内容分解成两个特征值，分别是 x_1 :爱情程度 x_2 :动作程度，每一个特征值的程度是属于 $[0,1]$ ，而后使用线性回归进行预测的方法。

电影	Alice(1)	Bob(2)	Carol(3)	Dave(4)	x_1	x_2
Love at last	5	5	0	0	1.0	0.0
Romance forever	5	?	?	0	0.9	0.1
Cute puppies of love	5	4	0	?	0.99	0.01
Nonstop car chases	0	0	5	4	0	1
Sword vs. karate	0	0	5	?	0.2	0.8

具体的算法如下：

Optimization objective:

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

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

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

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

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

Andrew Ng

其中 $\theta^{(j)}$ 表示的是对 j 个用户的线性回归。

那么，将所有的用户整合到一个线性回归的方程，如下：

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:

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

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

$\frac{\partial}{\partial \theta_k^{(j)}} J(\Theta^{(1)}, \dots, \Theta^{(n_u)})$

Andrew Ng

但是，这种方法是依赖于内容的。而有些时候，是没有内容特征的，那么该如何处理呢？

16-3 协同过滤

这个算法最厉害的地方是能够自己学习特征，这样就解决上面所提到的特征问题。那么，能否系统自己学习特征值？

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

假设能够得到每一个用户的喜好向量，例如 $\theta^{(1)}$ 表示的是第一个用户的喜好，爱情片5分，动作片0. 接下来，要做的事，就是通过这些用户的喜好来学习出来特征量。这里的思想是什么？在前面的线性回归中，是有 Alice: $(\theta^{(1)})^T x^{(1)} = 5$; Bob: $(\theta^{(2)})^T x^{(1)} = 5$; Carol: $(\theta^{(3)})^T x^{(1)} = 0$; Dave: $(\theta^{(4)})^T x^{(1)} = 0$;

对于，第2节阐述的线性回归，是我们知道了 x 特征量，通过线性回归计算出 θ 。而现在，我们要做的事，通过参考的已知的 θ 来计算 x ，方法还是一样的。

接下来的算法事实上和上面的就是一样的。只不过现在是知道 θ 求解 x ，本质上没有任何区别。

现在的问题来了，初始的 θ 不一定准确，而 θ 和 x 似乎是鸡和蛋的关系，事实也是如此，二者互相推导，最后都收敛。

$\theta \Rightarrow x \Rightarrow \theta \Rightarrow x \dots$

所以 θ 可以使用随机初始化一个值计算出来 x ，用 x 在求 θ ，如此往复。随着数据逐渐增多，就会渐渐的收敛。

算法如下：

Optimization algorithm

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

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

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

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

Andrew Ng

16-4 协同过滤算法

在上一节中阐述的算法，知道是 x 和 θ 的不停迭代最终导致的收敛。那么，其实可以将两个代价函数合并，统一计算，直接随机初始化 x 和 θ ，然后同时计算出二者。这里需要注意的一点是， $x_0=1$ 这一项是不存在的，也就是偏置项是不存在的，因为没有必要将一个特征固定为1，如果是1，那么系统会自己学习这个特征。如果 x_0 不存在，那么 θ_0 也就不存在了。所以 $x \in \mathbb{R}^n, \theta \in \mathbb{R}^n$.

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:

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

$$\min_{x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$$

Andrew Ng

将代价函数合并，然后使用梯度下降进行统一计算。

Collaborative filtering algorithm

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

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

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

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

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

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

Andrew Ng

关于协同过滤算法理解。看算法，对于X和θ的初始化是随机初始化的，这点要特别的注意。进行和线性回归的对比如下表：

比对特性	线性回归	协同过滤算法
特征量 X	已知数据	待求解数据
权重 θ	待求解数据	待求解数据
y值	已知数据	已知数据

通过这个表格就清楚的知道，对于协同过滤算法来说，是仅仅知道y(在电影评价中就是评分)，而去寻找合适的θ和x来对应y。

16-5 向量化

所谓的推荐系统，就是能够对一个用户，将这个用户没有评价的商品打上一个分值。如果这个分值超过一个阈值，就可以将这个商品推荐给这个用户。这就是推荐系统的核心理念。

当推荐系统学习到了一些特征值，这些往往非常有意义。

当我们全部计算好了 x 和 θ ，那么通过下面的向量化的公式就非常预测和看到某个电影的评分了。

Collaborative filtering

The diagram illustrates the Collaborative Filtering matrix factorization process. It starts with a sparse rating matrix Y (User x Movie matrix) where some values are circled in blue. This matrix is multiplied by X and Θ^T to produce predicted ratings. The predicted ratings are shown as a matrix where each row corresponds to a user and each column to a movie. The matrix is labeled "Predicted ratings: $(\Theta^{(i)})^T(x^{(n)})$ ". Below this, the matrices X and Θ are shown as low-rank matrices, with X having dimensions $(n_u \times n_m)$ and Θ having dimensions $(n_u \times n_f)$. The formula for the predicted rating is $(\Theta^{(i)})^T(x^{(n)})$.

Andrew Ng

疑问：特征量的维度是多少呢？也就是说 θ 和 x 的维度是多少？貌似目前并没有给出合理的定义和求解方式？难道这个维度需要自己来选择1, 2, ... 某个数值吗？显然太多不行，太少也不行。

如果，需要看两个商品的关联度如何，通过下面的公式：

$$d = \|x^{(i)} - x^{(j)}\|$$

这个公式计算了特征量的距离，如果 d 越小，说明越相似。前面说，通过对没有评价的商品进行评分可以推荐，其实本质上是寻找关联度最接近的商品，也就是 d 最小的几个商品。例如电影，当用户看过了一个电影，就可以通过关联度最大的其他几部电影推荐给用户，这就是推荐算法的理念。这和给一个电影打分后，选择最高的给用户是不冲突，因为打分是 θx ，而对于同一个用来说 θ 是一样的，所以就由 x 来决定。而在用户看过的电影中，一定是 θx 最大，所以也就是计算的推荐得分最高。

这种方法，可以用来进行对股票的相关性的测试上。

向量化的矩阵对应关系

$$\begin{aligned}
 Y &= \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix} \Leftrightarrow \begin{bmatrix} (\theta^{(1)})^T(x^{(1)}) & (\theta^{(2)})^T(x^{(1)}) & \dots & (\theta^{(4)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & \dots & (\theta^{(4)})^T(x^{(2)}) \\ (\theta^{(1)})^T(x^{(3)}) & (\theta^{(2)})^T(x^{(3)}) & \dots & (\theta^{(4)})^T(x^{(3)}) \\ (\theta^{(1)})^T(x^{(4)}) & (\theta^{(2)})^T(x^{(4)}) & \dots & (\theta^{(4)})^T(x^{(4)}) \\ (\theta^{(1)})^T(x^{(5)}) & (\theta^{(2)})^T(x^{(5)}) & \dots & (\theta^{(4)})^T(x^{(5)}) \end{bmatrix} \\
 &\Leftrightarrow \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \\ x^{(4)} \\ x^{(5)} \end{bmatrix} \left[\begin{array}{cccc} (\theta^{(1)})^T & (\theta^{(2)})^T & (\theta^{(3)})^T & (\theta^{(4)})^T \end{array} \right] \\
 &\Leftrightarrow X\Theta^T
 \end{aligned}$$

$$n_m = 5; n_u = 4$$

对于这个矩阵来说，其中电影的总量是5(n_m)，用户的数量是4(n_u)。上面就是矩阵对应的向量化关系。

16-6 均值归一化

现在的问题是，如果一个用户从来没有对任何电影评价过，那么，进行推荐的时候将会是对每一个电影都计算出来同样的结果，那么，这样就没有办法进行推荐。进行修订的方法是使用均值归一化，这样通过每一个电影的平均评分，来对该用户的评分进行修正处理。因为每一个电影的评分均值不同，所以最后就不会出现每一个电影都是同样的评分了。

对于矩阵行进行均值归一化，算法是：

$$\begin{aligned}
 M &= \text{行所有元素相加/总数} \\
 \text{每一行数据} - M
 \end{aligned}$$

$$\text{最后计算出来的结果 } (\theta^{(j)})^T(x^{(i)}) + M$$

这样做好处是可以避免全是0的case.

第十七课 大规模机器学习

Lesson17 大规模机器学习

17-1 大数据集学习

大量的数据产生的问题就是计算性能问题。在前面，我们知道对于高方差的（过拟合）的时候，使用更多的数据才更有效果；而对于高偏差，使用更多的数据集是没有意义的。所以在进行模型的训练的时候，先选取小量数据集，看看模型算法是处于高方差还是高偏差，进而决定是否增加数据的量。例如: $m=100,000,000$ ，先选取 $m=1,000$ 来进行训练，做出训练模型图，知道是高方差还是高偏差，进而决定是否使用 $m=100,000,000$ 来进行处理。因为小的数据集会让你的算法很快的运行。

17-2 随机梯度下降

提出的目的是因为，当我们进行正常的梯度下降的时候，我们每次使用的是所有数据，如果数据量很大，那么就需要批次的将数据导入到内存，执行梯度下降，而这样的效率就很低。

而随机梯度下降算法，是逐个样本进行梯度下降，最后，直到部分的样本就能够完成这个过程，也不需要批量的导入。

所以算法如下：

```
Repeat {
    for i=1 to m {
        θ_j := θ_j - α(h_θ(x^(i)) - y^(i))(x_j)^(i)
    }
}
```

所以数据要随机化。这里要特别注意 "Repeat" 这一层，这里通常取1-10或者更多，因为这一层会使得结果逐渐收敛。

我觉得更好的随机化先进行聚类分析，然后再进行随机化，会使得数据更加随机。

实际上就是将梯度下降算法中的 $m=1$ 来进行计算的。

这里对于整个算法纯数学证明，目前没有，只是一种解释。

17-3 最小批量梯度下降

最小批量梯度下降是使用一个比较小的样本来进行梯度下降。随机梯度下降中 $m=1$ ，而标准的是 m ，那么小批量就是 $b \in [2, 100]$ 使用这样一个小的样本数据集进行每次的梯度下降。

为什么小批量的梯度下降会更快一点的呢？是因为通过向量化，可以进行适当的并行计算，从而加速数据的计算。

17-4 随机梯度下降的收敛

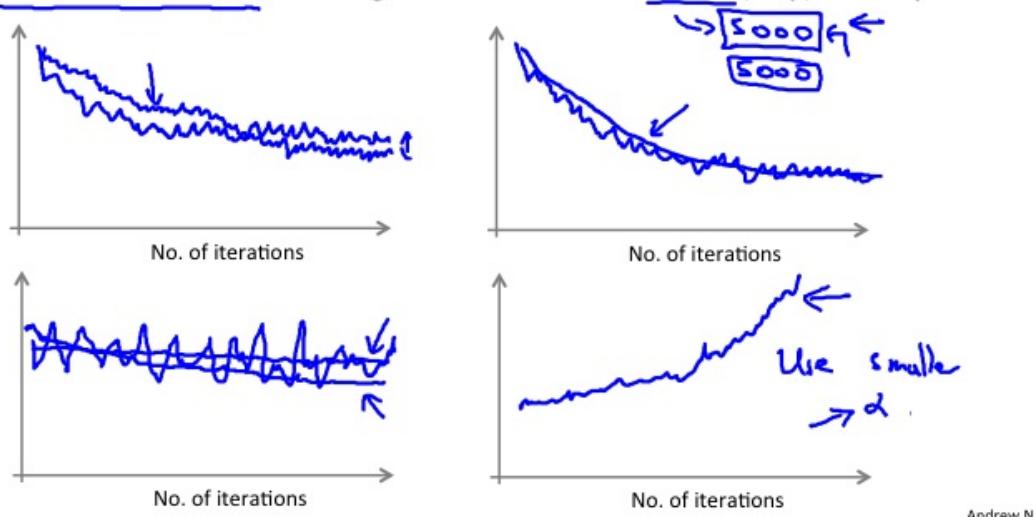
这其实在前面已经阐述过，就是通过图像进行观察是否收敛。只是现在是通过每1000次迭代，来计算 $\text{cost}(\theta, (x^{(i)}, y^{(i)}))$ 的算术平均，来观察是否向下收敛。观察是否收敛是非常重要的步骤和环节。

另外关于学习速率 α 来说，是可以通过随着时间的变化来使得 α 变小来得到更加精准的拟合。

绘制代价函数的曲线，有助于我们分析和判断。当我们对不同的批量数据，例如1000或5000进行曲线的绘制，会得到不同的平滑曲线。但最终都是要向下收敛的。如果曲线过于平着震荡，那么很有可能是因为选择绘制曲线的样本数量过少导致的，尝试使用更大的熟练进行绘制。也可能是 α 值过大。

Checking for convergence

Plot $\text{cost}(\theta, (x^{(i)}, y^{(i)}))$, averaged over the last 1000 (say) examples



关于 α 的变化问题，可以尝试变化，使用这个算法：

$$\alpha = \text{const1} / (\text{迭代次数} + \text{const2})$$

这样就会随着迭代次数的增加而逐渐减小。但是一些人不愿意使用这样的方法，因为需要调整const1和const2也就意味着需要调整更多的参数。

17-5 在线学习

在线学习系统，最大的有点在于不需要存储数据，随着每天的新的数据自动学习。这本质上就是随机梯度下降算法，每次来一个数据，就学习一次，来一个数据就学习一次。这不需要保存之前的数据，而是随着网站的运行自动的不断优化自身。这个算法很牛呀！！！

点击率: (click-through rate)

一个在线学习的系统将是非常厉害。能够根据每一个数据的输入自动训练 θ 。当 θ 训练之后，可以使用 θ 对任意一个数据和现有商品之间的关系进行预测。使用逻辑回归预测的就是概率 p ，那么，就可以根据 p 的大小来进行商品的推荐。

17-6 Map-reduce和数据并行

Map-reduce 能够处理非常大的数据。可以，通过分布式开源系统，例如Hadoop，来进行

Map-reduce的实现。也可以，针对单一计算机的多核进行map reduce.其实, map reduce并非是关键的，关键的是对算法的分解。

第十八课 应用举例：图片 OCR

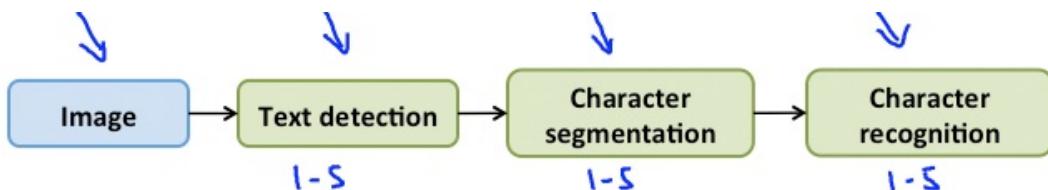
Lesson18 应用举例：图片 OCR

18-1 问题描述和流水线

1. 文字检测。将文字的区域检测出来
2. 字母分段。将是字母的区域分割出来。
3. 字母识别。对每一个字母进行识别。

作为机器学习系统设计，就是分解成流水线，然后安排不同的人去工作。我想TensorFlow可能使用的就是这样的方式建立起来的机器学习系统。

管道系统，是设计复杂机器学习的核心：



18-2 滑动窗口

所谓的滑动窗口，是对这样一个窗口所表达的区域进行训练，结果这个矩形区域能够正确认别处在窗口内部的事物。然后，再移动这个窗口，来不停的与图片中的区域进行比对，进行分类预测。所以步骤如下：

- 1 训练窗口区域
- 2 使用较小窗口区域进行循环移动，覆盖整个图片
- 3 增大窗口区域，但要保证依然是82*36， 所以需要缩小图片。重复执行第二步

在上面的算法中，注意一点：增大识别窗口事实上是缩小图片来处理的。因为在进行训练的时候，使用的就是82*36这个数据进行识别的。在人像识别中，因为人的比例是基本固定的，所以选择82*36这样的图像，进行分类学习。

文字检测与人像检测类似。

- 1 训练窗口
- 2 识别完文字，使用白色和灰色标记。
- 3 膨胀识别块
- 4 丢弃明显比率错误的块
- 5 获得文字检测的区域

文字拆分，问题是哪个位置进行拆分？学习。

- 1 找到在矩形框中间能够正确拆分的样例和反向样例
- 2 训练窗口
- 3 拆分字符

字符识别，使用其他方法进行识别，例如神经网络。

18-3 人工合成数据

通过人工合成的方法可以得到更多的数据。例如，字符识别，语音识别，可以通过变形改造等形成新的训练集数据。

在你准备进行人工合成数据的时候，必须问清楚自己你的机器学习是否是高偏差的。以及你要多少工作来得到10倍的数据。

18-4 上限分析

当我们建立了机器学习流水线的各个组成部分之后，我们需要分析在哪个部分应该花费最多的时间。事实上，这个问题就是在进行性能分析，找出性能的瓶颈。显然，这种上限分析是基于管道模型的，分析管道中每一个节点。

Component	Accuracy	性能增量
Overall System	72%	0
Text Detection	89%	17
Character Segmentation	90%	1
Character Recognition	100%	10

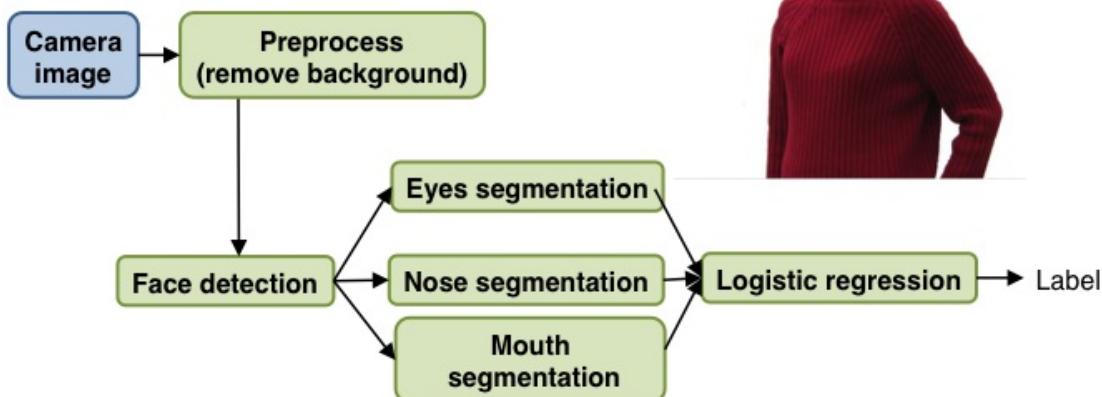
通过这样的分析，我们知道性能增量最大的地方是17，所以应该优化这部分组件。

那么这里的 "Accuracy" 是如何测量的呢？Accuracy的含义是系统这时候的正确率，最初的72%是正常的系统准确率；89%是当我们将 "Text Detection" 使用完全准确的结果之后，进行的继续测试得到89%，那么我们就知道当 "Text Detection" 是完全准确的时候，能够提升系统准确率17. 接下来，在Text Detection是完全正确的情况下，提升" Character Segmentation"，那么系统从89% 提升到90%，说明优化空间不大。最后一定是100%。通过这种方式来知道究竟提升哪个部分来增强系统的准确性。

基于管道模型的上限分析.

Another ceiling analysis example

Face recognition from images
(Artificial example)



Andrew Ng

第十九课 总结

Lesson19 总结

机器学习虽然完成。但其实你才刚刚上路。

机器学习作业

机器学习作业

git hub: <https://github.com/cjopengler/MachineLearning.git>

NG机器学习 **Python**实践

NG机器学习**Python**实践

使用Python完成NG机器学习中的所有代码。在NG的网课中，所有代码使用Octave来完成的。这里使用Python来完成所有代码。

Scikit-learn实践

scikit-learn实践

学习 scikit-learn 主流算法并应用到实践中.

数据集

数据集

数字

每一个是8*8的矩阵，用灰度表示，0表示黑色，16表示白色，中间值表示不同的灰度。

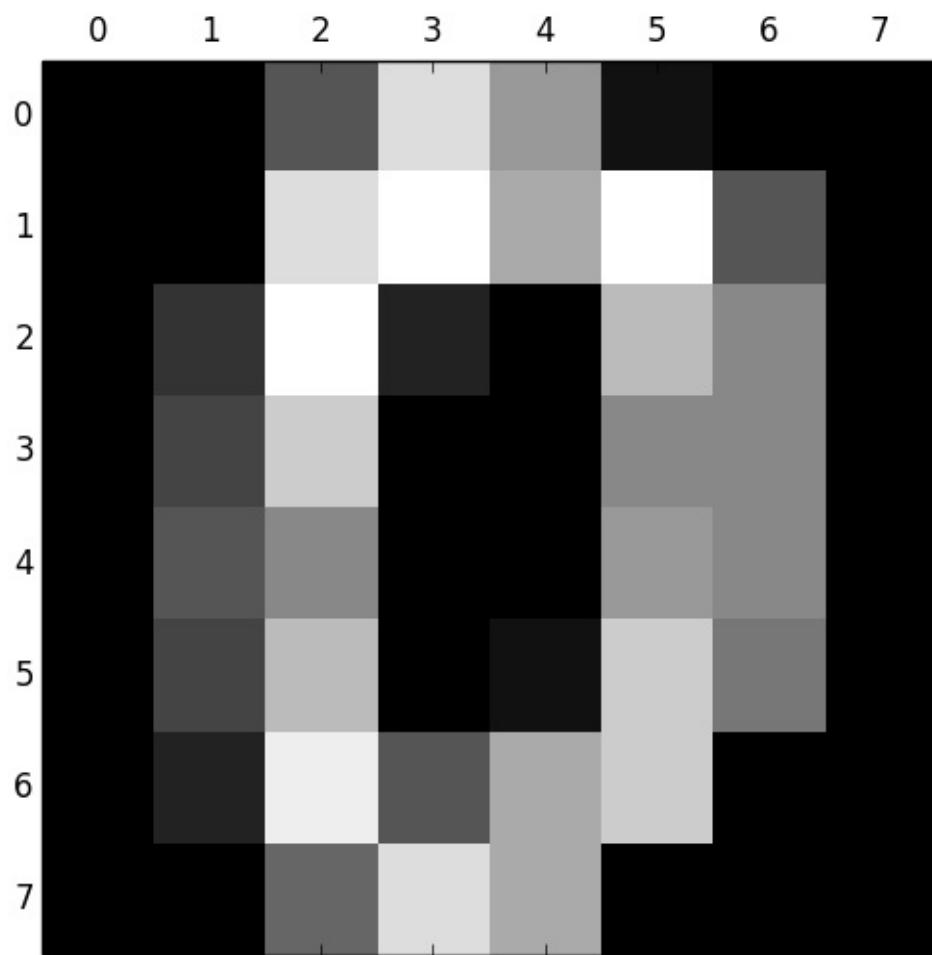
```
from sklearn.datasets import load_digits
digits = load_digits()
print(digits.data.shape)

image = digits.images[0]
print type(image), digits.images[0].shape
print digits.images[0]

image[1][0] = 16
image[1][1] = 16
image[1][2] = 16

print type(image), image.shape
print image

import pylab as pl
pl.gray()
pl.matshow(image)
pl.show()
```



波士顿房价

/System/Library/Frameworks/Python.framework/Versions/2.7/bin/python2
(506, 13)

Boston House Prices dataset

Notes

Data Set Characteristics:

:Number of Instances: 506

:Number of Attributes: 13 numeric/categorical predictive

:Median Value (attribute 14) is usually the target

:Attribute Information (in order):

- CRIM per capita crime rate by town
- ZN proportion of residential land zoned for lots over 25,000
- INDUS proportion of non-retail business acres per town

- CHAS Charles River dummy variable (= 1 if tract bounds
- NOX nitric oxides concentration (parts per 10 million)
- RM average number of rooms per dwelling
- AGE proportion of owner-occupied units built prior to
- DIS weighted distances to five Boston employment cent
- RAD index of accessibility to radial highways
- TAX full-value property-tax rate per \$10,000
- PTRATIO pupil-teacher ratio by town
- B $1000(Bk - 0.63)^2$ where Bk is the proportion of b.
- LSTAT % lower status of the population
- MEDV Median value of owner-occupied homes in \$1000's

:Missing Attribute Values: None

:Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.

<http://archive.ics.uci.edu/ml/datasets/Housing>

This dataset was taken from the StatLib library which is maintained at

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic prices and the demand for clean air', J. Environ. Economics & Management vol.5, 81-102, 1978. Used in Belsley, Kuh & Welsch, 'Regression diagnostics...', Wiley, 1980. N.B. Various transformations are used in the text pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning problems.

References

- Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Points and Sources of Collinearity'
- Quinlan, R. (1993). Combining Instance-Based and Model-Based Learning
- many more! (see <http://archive.ics.uci.edu/ml/datasets/Housing>)

Process finished with exit code 0

Irish- 鸢尾花

iris: 鸢尾花

鸢尾花数据集是非常容易的多分类的数据集

3个分类，每个分类50个样本，4个维度

该数据集包含了5个属性：

- & Sepal.Length (花萼长度)，单位是cm;
- & Sepal.Width (花萼宽度)，单位是cm;
- & Petal.Length (花瓣长度)，单位是cm;
- & Petal.Width (花瓣宽度)，单位是cm;
- & 种类：Iris Setosa (山鸢尾)、

Iris Versicolour (杂色鳶尾) ,
以及Iris Virginica (维吉尼亚鳶尾) 。

Bunch是字典的子类，事实上就是一个字典

代码

```
print type(data)
print data.keys()
print type(data['target_names'])
print data['target_names']
print type(data['data'])
print data['data'], data['data'].shape
print data['target']
print data['DESCR']
print data['feature_names']
print type(data['feature_names'])
```

输出

```
/System/Library/Frameworks/Python.framework/Versions/2.7/bin/python2
<class 'sklearn.datasets.base.Bunch'>
['target_names', 'data', 'target', 'DESCR', 'feature_names']
<type 'numpy.ndarray'>
['setosa' 'versicolor' 'virginica']
<type 'numpy.ndarray'>
[[ 5.1  3.5  1.4  0.2]
 [ 4.9  3.   1.4  0.2]
 [ 4.7  3.2  1.3  0.2]
 [ 4.6  3.1  1.5  0.2]
 [ 5.   3.6  1.4  0.2]
 [ 5.4  3.9  1.7  0.4]
 [ 4.6  3.4  1.4  0.3]
 [ 5.   3.4  1.5  0.2]
 [ 4.4  2.9  1.4  0.2]
 [ 4.9  3.1  1.5  0.1]
 [ 5.4  3.7  1.5  0.2]
 [ 4.8  3.4  1.6  0.2]
 [ 4.8  3.   1.4  0.1]
 [ 4.3  3.   1.1  0.1]
 [ 5.8  4.   1.2  0.2]
 [ 5.7  4.4  1.5  0.4]
 [ 5.4  3.9  1.3  0.4]
 [ 5.1  3.5  1.4  0.3]
 [ 5.7  3.8  1.7  0.3]
 [ 5.1  3.8  1.5  0.3]
 [ 5.4  3.4  1.7  0.2]
 [ 5.1  3.7  1.5  0.4]
 [ 4.6  3.6  1.   0.2]
 [ 5.1  3.3  1.7  0.5]
 [ 4.8  3.4  1.9  0.2]
 [ 5.   3.   1.6  0.2]
 [ 5.   3.4  1.6  0.4]
 [ 5.2  3.5  1.5  0.2]]
```

```
[ 5.2  3.4  1.4  0.2]
[ 4.7  3.2  1.6  0.2]
[ 4.8  3.1  1.6  0.2]
[ 5.4  3.4  1.5  0.4]
[ 5.2  4.1  1.5  0.1]
[ 5.5  4.2  1.4  0.2]
[ 4.9  3.1  1.5  0.1]
[ 5.  3.2  1.2  0.2]
[ 5.5  3.5  1.3  0.2]
[ 4.9  3.1  1.5  0.1]
[ 4.4  3.  1.3  0.2]
[ 5.1  3.4  1.5  0.2]
[ 5.  3.5  1.3  0.3]
[ 4.5  2.3  1.3  0.3]
[ 4.4  3.2  1.3  0.2]
[ 5.  3.5  1.6  0.6]
[ 5.1  3.8  1.9  0.4]
[ 4.8  3.  1.4  0.3]
[ 5.1  3.8  1.6  0.2]
[ 4.6  3.2  1.4  0.2]
[ 5.3  3.7  1.5  0.2]
[ 5.  3.3  1.4  0.2]
[ 7.  3.2  4.7  1.4]
[ 6.4  3.2  4.5  1.5]
[ 6.9  3.1  4.9  1.5]
[ 5.5  2.3  4.  1.3]
[ 6.5  2.8  4.6  1.5]
[ 5.7  2.8  4.5  1.3]
[ 6.3  3.3  4.7  1.6]
[ 4.9  2.4  3.3  1. ]
[ 6.6  2.9  4.6  1.3]
[ 5.2  2.7  3.9  1.4]
[ 5.  2.  3.5  1. ]
[ 5.9  3.  4.2  1.5]
[ 6.  2.2  4.  1. ]
[ 6.1  2.9  4.7  1.4]
[ 5.6  2.9  3.6  1.3]
[ 6.7  3.1  4.4  1.4]
[ 5.6  3.  4.5  1.5]
[ 5.8  2.7  4.1  1. ]
[ 6.2  2.2  4.5  1.5]
[ 5.6  2.5  3.9  1.1]
[ 5.9  3.2  4.8  1.8]
[ 6.1  2.8  4.  1.3]
[ 6.3  2.5  4.9  1.5]
[ 6.1  2.8  4.7  1.2]
[ 6.4  2.9  4.3  1.3]
[ 6.6  3.  4.4  1.4]
[ 6.8  2.8  4.8  1.4]
[ 6.7  3.  5.  1.7]
[ 6.  2.9  4.5  1.5]
[ 5.7  2.6  3.5  1. ]
[ 5.5  2.4  3.8  1.1]
```

```
[ 5.5  2.4  3.7  1. ]
[ 5.8  2.7  3.9  1.2]
[ 6.   2.7  5.1  1.6]
[ 5.4  3.   4.5  1.5]
[ 6.   3.4  4.5  1.6]
[ 6.7  3.1  4.7  1.5]
[ 6.3  2.3  4.4  1.3]
[ 5.6  3.   4.1  1.3]
[ 5.5  2.5  4.   1.3]
[ 5.5  2.6  4.4  1.2]
[ 6.1  3.   4.6  1.4]
[ 5.8  2.6  4.   1.2]
[ 5.   2.3  3.3  1. ]
[ 5.6  2.7  4.2  1.3]
[ 5.7  3.   4.2  1.2]
[ 5.7  2.9  4.2  1.3]
[ 6.2  2.9  4.3  1.3]
[ 5.1  2.5  3.   1.1]
[ 5.7  2.8  4.1  1.3]
[ 6.3  3.3  6.   2.5]
[ 5.8  2.7  5.1  1.9]
[ 7.1  3.   5.9  2.1]
[ 6.3  2.9  5.6  1.8]
[ 6.5  3.   5.8  2.2]
[ 7.6  3.   6.6  2.1]
[ 4.9  2.5  4.5  1.7]
[ 7.3  2.9  6.3  1.8]
[ 6.7  2.5  5.8  1.8]
[ 7.2  3.6  6.1  2.5]
[ 6.5  3.2  5.1  2. ]
[ 6.4  2.7  5.3  1.9]
[ 6.8  3.   5.5  2.1]
[ 5.7  2.5  5.   2. ]
[ 5.8  2.8  5.1  2.4]
[ 6.4  3.2  5.3  2.3]
[ 6.5  3.   5.5  1.8]
[ 7.7  3.8  6.7  2.2]
[ 7.7  2.6  6.9  2.3]
[ 6.   2.2  5.   1.5]
[ 6.9  3.2  5.7  2.3]
[ 5.6  2.8  4.9  2. ]
[ 7.7  2.8  6.7  2. ]
[ 6.3  2.7  4.9  1.8]
[ 6.7  3.3  5.7  2.1]
[ 7.2  3.2  6.   1.8]
[ 6.2  2.8  4.8  1.8]
[ 6.1  3.   4.9  1.8]
[ 6.4  2.8  5.6  2.1]
[ 7.2  3.   5.8  1.6]
[ 7.4  2.8  6.1  1.9]
[ 7.9  3.8  6.4  2. ]
[ 6.4  2.8  5.6  2.2]
[ 6.3  2.8  5.1  1.5]
```

Notes

Data Set Characteristics:

:Number of Instances: 150 (50 in each of three classes)

:Number of Attributes: 4 numeric, predictive attributes and the target attribute.

:Attribute Information:

- sepal length in cm
 - sepal width in cm
 - petal length in cm
 - petal width in cm
 - class:
 - Iris-Setosa
 - Iris-Versicolour
 - Iris-Virginica

:Summary Statistics:

	Min	Max	Mean	SD	Class Correlation
sepal length:	4.3	7.9	5.84	0.83	0.7826
sepal width:	2.0	4.4	3.05	0.43	-0.4194
petal length:	1.0	6.9	3.76	1.76	0.9490 (high!)
petal width:	0.1	2.5	1.20	0.76	0.9565 (high!)

:Missing Attribute Values: None

:Class Distribution: 33.3% for each of 3 classes.

:Creator: R A Fisher

Donor: Michael Marshall (MARSHAL1%PLUTO.arc.nasa.gov)

• Donor: Michael H.
• Date: July 1988

This is a copy of UCI ML iris datasets.
<http://archive.ics.uci.edu/ml/datasets/Iris>

The famous Iris database, first used by Sir R.A Fisher

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example) The data set contains 3 classes of 50 instances each, where each class represents a type of iris plant. One class is linearly separable from the other two; the latter are NOT linearly separable from each other.

References

- Fisher, R.A. "The use of multiple measurements in taxonomic problems". *Annual Eugenics*, 7, Part II, 179-188 (1936); also in "Contributions to Mathematical Statistics" (John Wiley, NY, 1950).
- Duda, R.O., & Hart, P.E. (1973) *Pattern Classification and Scene Analysis* (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.
- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New Structure and Classification Rule for Recognition in Partially Incomplete Environments". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 1, 67-71.
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". *IEEE Transactions on Information Theory*, May 1972, 431-433.
- See also: 1988 MLC Proceedings, 54-64. Cheeseman et al's AUTOCLUSTER conceptual clustering system finds 3 classes in the data.
- Many, many more ...

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
<type 'list'>
```

Process finished with exit code 0

决策树

决策树

[决策树中文](#)

对于决策树来说，使用起来还是很方便的，最大的优点在于不需要进行数据的统一，其中可以有分类数据也可以有数值数据，这样就容易得多了。

参数调试

逻辑回归

逻辑回归

参数

对于逻辑回归来说最重要的参数是C，也就是 $1/\lambda$. C是惩罚因子，用来调节过拟合问题。

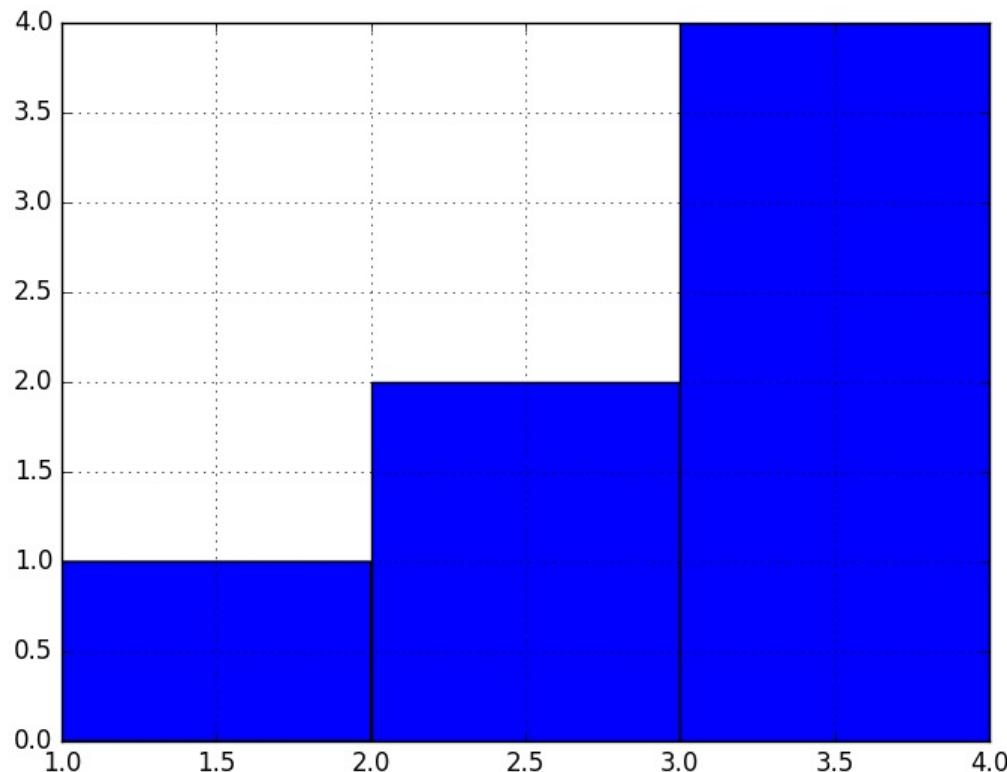
Pandas实践

Pandas实践

数据可视化

直方图绘制 hist

```
data = pd.Series([1, 2, 3, 4, 3, 2, 3])
data.hist(bins=3)
plt.show()
```



直方图图中 bins 参数表示有几个竖条，会将整个区域进行分隔，现在的最小值是1，最大值是4，所以横坐标从1开始到4，被分成了bins个竖条，所以就是[1,2) [2,3) [3,4] 纵坐标就是每个区间中的数量，例如[1,2) 之间有1个1，所以纵坐标是1；[3,4]有3个3+1个4=4个。既然知道直方图是这样来进行分隔和绘制的，那么可以用来看概率分布的。

另外，如果将1,2数值换成字母'a','b','c'是否可以呢？答案是不可以因为，'a","b","c'不是数值无法进行除以bins的分隔计算，所以不能是字母。

分类数据的可视化

Tesorflow实践 0.8

Tensorflow

记录在实践tensorflow时的点滴经验。

环境搭建

环境搭建

[官方网址](#)

mac安装使用的是pip，具体命令参考 [pip安装](#)

安装问题列表

1 C compiler cannot create executables

当安装程序的时候，有的时候会出现这个错误。

解决方案

是因为“Command Line Tools”没有安装。安装的方法如下：

[Command Line Tools 安装](#)

2 Operation not permitted

当mac升级到最新的时候，通过脚本安装程序的时候因为可能会修改系统文件夹。那么就会出现这个错误，即使管理员权限也不行。这是苹果的保护机制，只有苹果的签名程序可以操作。

[知乎解决方案1](#)

[国外解决方案2](#)

3 ImportError: numpy.core.multiarray failed to import

这是因为在安装Tensorflow的时候会下载，numpy到对应的版本。而MAC本身也带了numpy，而mac自身的numpy版本较低导致的。办法就是，删除mac自带的numpy即可。使用tensorflow 下载的。

第一步确定系统的numpy安装路径：

```
import numpy as np  
np.path
```

输出结果：

```
['/System/Library/Frameworks/Python.framework/Versions/2.7/Extras/lil
```

删除该系统目录下的numpy：

```
sudo rm -rf /System/Library/Frameworks/Python.framework/Versions/2.7.
```

4 ImportError: No module named protobuf

这是因为 protobuf的版本比较旧了。最简单的办法是删除 protobuf，在重新安装 tensorflow 即可。

```
sudo pip uninstall protobuf  
sudo pip uninstall tensorflow
```

卷积神经网络 **CNN**

卷积神经网络

参考资料

[tensorflow 做手写识别 详解CNN](#)

机器学习实践

机器学习实践

这里介绍机器学习实践中的笔记和思考。参考《机器学习实践》这本书。

第二章 k邻近算法

第二章 k邻近算法

k邻近算法与聚类算法很相似，通过对比当前预测数据与打好标签的数据之间的距离来判定属于那种标签。

优缺点分析

k邻近算法优点在于，简单实现速度快。缺点是因为每次都要计算大量的距离会导致性能问题。而其他分类算法，只是在训练的时候耗费资源，在真正预测的时候，速度非常的快。因为 Θ 都已经训练好。所以，这种算法适合快速的给出小量数据的原型，大量数据则不能使用这种方法。

第三章 决策树

第三章 决策树

决策树的数据必须是枚举类型的。或者说是整数。

信息增益和熵

符号 x_i 的信息定义为：

$$I(x_i) = -\log p(x_i)$$

熵的定义：

$$H = -\sum(p(x_i)\log p(x_i))$$

熵的这种表示形式和逻辑回归很像。

对于数据集中标签的熵的计算方法，是：

1 计算概率 p

1.1 $p = \text{label}$ 的数量 / 总的数据集的数量

2 计算熵

2.1 将所有的 $-p * \log(p)$ 相加求和

熵越高，表示混合的数据越多，可以在数据集中增加更多的分类，以便观察熵的变化。

分类

有了熵的定义，我们就知道所谓的分类就是使得熵不断的下降收敛，就是最好的分类。
(因为熵越高，表示混合的数据越多；熵越低，表示逐渐在进行分类)

如何进行分类？所谓的分类就是将其中一个指定的特征量抽取出来，余下的组成数据集。例如：[[1, 0, 0], [1, 0, 2] [0, 1, 1]] 按照第一列抽取，就是[1, 1, 0]和剩下的[[0,0], [0, 2], [1, 1]]，指定特征量为0，那么结果就是[[0, 0], [0, 2]].按照第二列抽取，就是 [[1,0], [1, 2], [0, 1]] 指定特征量如果为1，就是[[1, 0], [1, 2]]

例如：

序号 不浮出水面 是否可以生存 是否有脚蹼 是否属于鱼类

1	是	是	是
2	是	是	是
3	是	否	否
4	否	是	否
5	否	是	否

6 是	否	是
-----	---	---

上面的表格，我们按照第二个特征量进行分类，按照value='是'进行分类的结果是：

Value='是'

序号 不浮出水面 是否可以生存 是否属于鱼类

1 是	是	是
2 是	是	是
4 否	否	否
5 否	否	否

Value='否'

序号 不浮出水面 是否可以生存 是否属于鱼类

3 是	否	否
6 是	是	是

注意：上面的结构表示，最后一列“是否属于鱼类”这个分类标签的位置永远没有改变，永远是最后一列。

熵的计算

对‘是’和‘否’的熵进行计算，找出熵最小的作

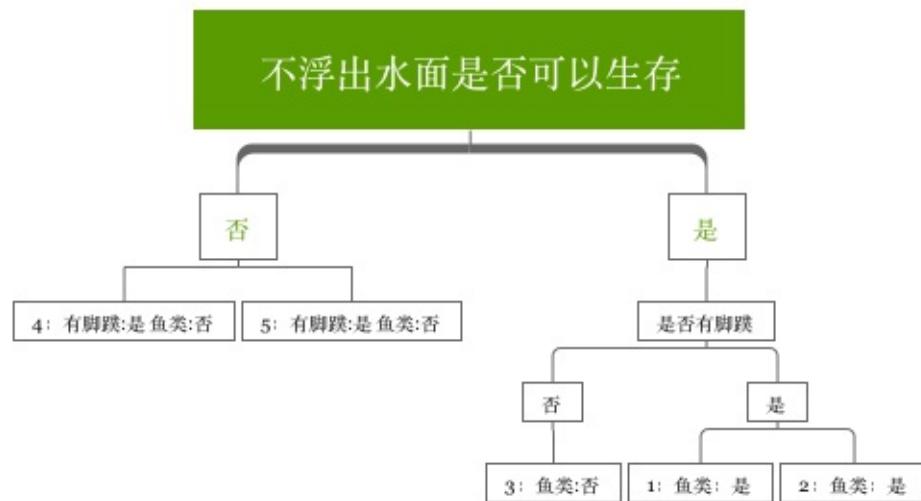
决策树的构建

在构建决策树中，看每层是否是相同的分类，如果是相同的分类，则不能继续划分，也就是叶子节点；如果不是相同的分类，可以继续划分。看下面的例子。

序号 不浮出水面 是否可以生存 是否有脚蹼 是否属于鱼类

1 是	是	是
2 是	是	是
3 是	否	否
4 否	是	否
5 否	是	否

决策树的结果：



4，5组成的分类就是一个叶子节点，因为4和5的 label 都是一样的。都是 fish:否。

第四章 朴素贝叶斯分类

第四章 朴素贝叶斯分类

贝叶斯分类的标准解答

$w_i = \{ \text{第 } i \text{ 个特征值出现的事件} \}$, 对于给定的一个样本 $W = \{ w_k \text{ 个特征出现} \}$, 其中 $w_k = \{ \text{从 } w_1, w_2, \dots, w_n \text{ 中取出 } k \text{ 个特征} \}$ 。这里又告诉我们一点, 关于事件的定义, 就是元素出现的集合。所以对于 W 的分类预测是: $\max(P(c_i|W))$, 其中 $i \in [1, \text{size}(c)]$. 也就是求 W 发生的情况的, 每个分类出现的概率的最大值。那么这个分类就是 W 所对应的分类。现在计算这个条件概率, 使用 bayes 公式:

$$P(c|W) = P(c) * P(W|c) / P(W) = P(c) * P(w_1 w_2 \dots w_n | c) / P(w_1 w_2 \dots w_n)$$

假设 w_1, w_2, \dots, w_n 是相互独立的, 那么有:

$$P(c|W) = P(c) P(w_1|c) P(w_2|c) \dots P(w_n|c) / P(w_1) P(w_2) \dots P(w_n)$$

这个相互独立的假设非常关键, 不然这个问题没办法计算下去了。

1 计算 $P(c)$:

$$P(c) = \text{分类的数量} / \text{数据集的数量}$$

2 计算 $P(w_1)$

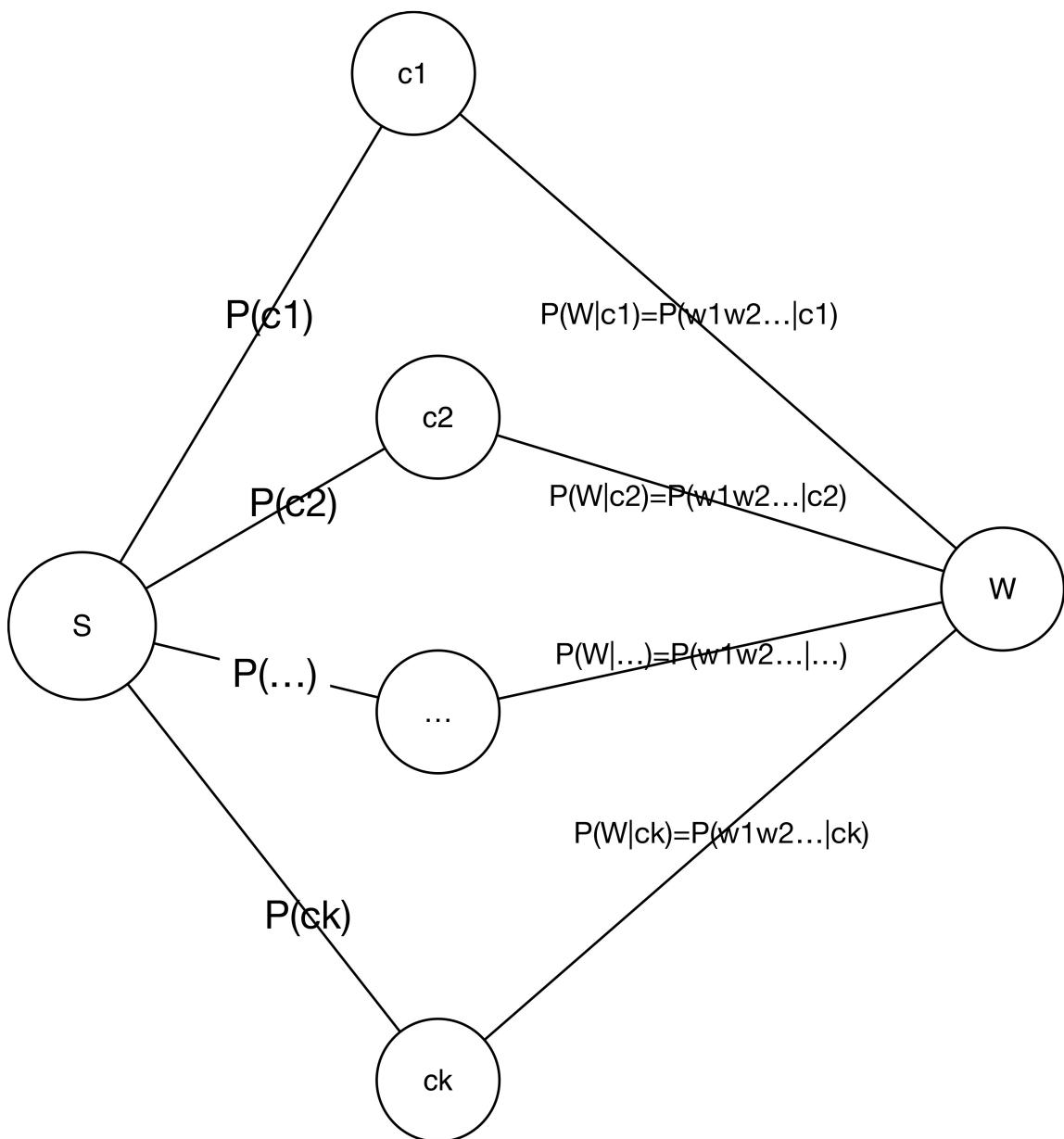
$$P(w_1) = w_1 \text{ 的总数量} / \text{数据集所有特征的数量}.$$

注意这里所有单词的数量, 因为是计算单词出现的概率。这一项不用计算, 因为是比较

3 计算 $P(w_1|c)$

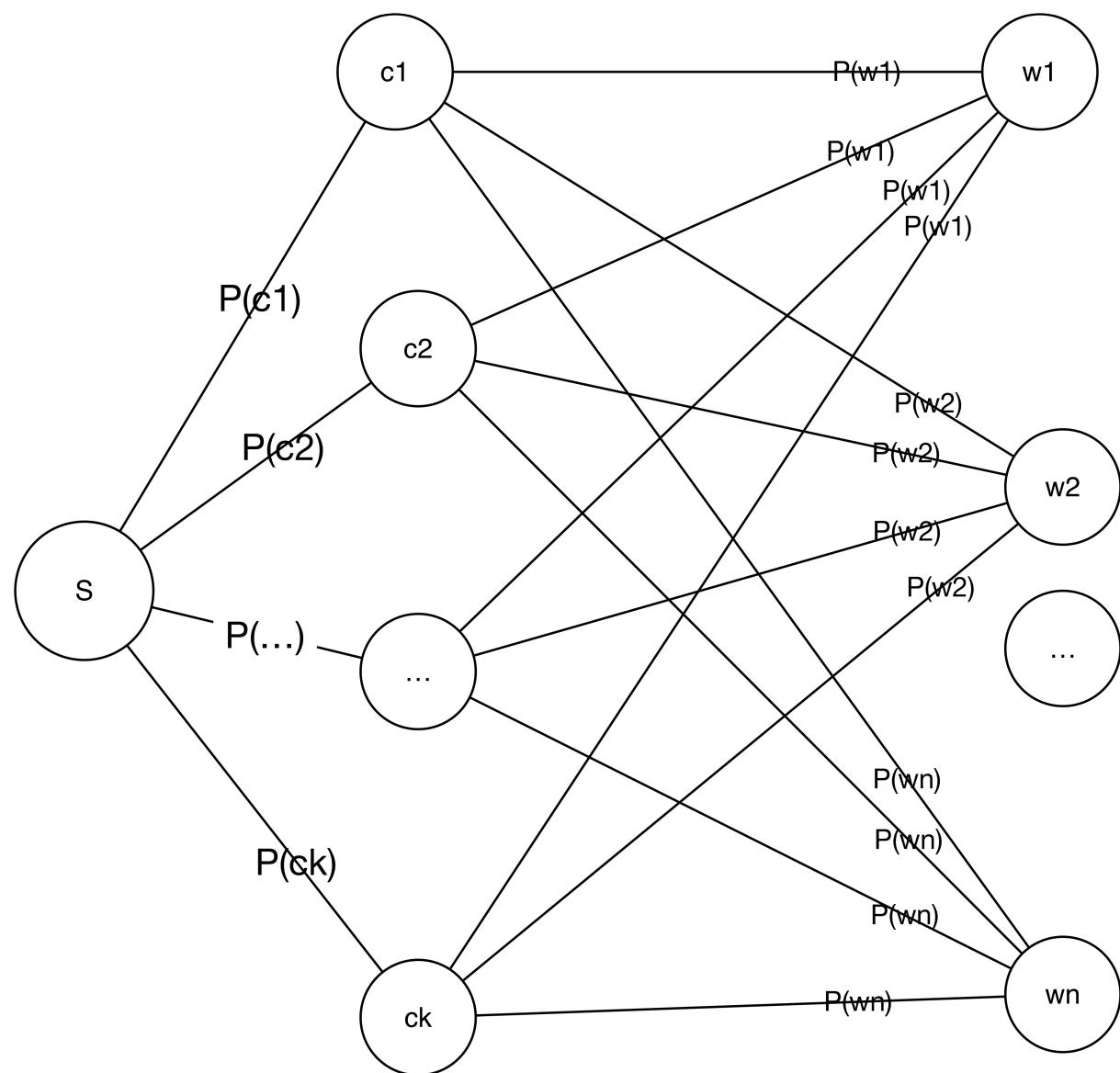
表示在 c 的分类中, w_1 的概率, 所以

$$P(w_1|c) = w_1 \text{ 在 } c \text{ 的集合中出现的数量} / c \text{ 中所有的特征出现的数量}$$



对于W的条件概率问题，看上面的图，解释的很清楚。在这张图中描述了以W作为事件的条件概率模型图。在这个图里，我们依然遇到的问题是，如何求 $P(W|c)$ 的问题。使用相互独立，进行简化。就是上面算法中所给，求 $P(w_i|c)$ 的问题。那么如何理解单一特征量的条件概率呢？

我们再从图的角度来看。下图描述一个条件概率的模型，表示了每一个特征出现的条件概率模型。通过这张图，可以更加直观的理解贝叶斯。



第五章 逻辑回归

第五章 逻辑回归

在处理数据的时候，会出现特征值不全的情况。有如下方法来进行填充：

1. 使用均值来填充
2. 使用特殊值填充，例如0或者-1
3. 忽略
4. 使用相似的样本填充
5. 使用另外的机器学习来预测

在逻辑回归中使用0来填充，因为0不会影响梯度下降的处理，另外 $\text{sigmoid}(0)=0.5$ 不带有倾向性。但是如果标签缺失，则只能舍弃。

Python

Python笔记

语法

语法

- 每一行后面没有 ";"
- for elment in Collection : 执行语句
- while 条件 : 执行语句。注意是没有大括号作用于的 通过:来处理
- isinstance 来判断是否是一个类型 例如: list
- 函数定义 def 函数名称(变量名称) 没有类型
- Python的作用域居然是通过缩进对齐来确定的。好大一个惊雷！
- 注释是使用 "''' 三个引号
- import 是导入其他文件书写的方法。而在使用的时候，必须将函数所在的文件名一起加入
- 参数是可以缺省的 例如 print_lol(list, level=0)
- open 打开一个文件之后 别忘记调用 close
- try: except: 异常的处理
- pickle 用来载入数据更方便
- data.sort 进行排序
- 字典的生成 a = {} a[no]=1. 很简洁
- 注释只能单行#
- // 表示的是真正的除法，不同于普通的 / 除法运算。例如 $1//3 = 0.333$ 进行四舍五入，而 $1/3 = 0.$
- ** 表示乘方运算
- $3<4<5$ 等于 $3 < 4 \text{ and } 4 < 5$ 是这样一种逻辑运算，结果是 true
- python 中不支持 -- 和 ++ 这种操作。
- pythoh 中的 long 表示长整形，这一点要与 java 和 C 中区分。这种长整形类似 java BigInteger，是不存在整形溢出问题的。也就是说可以表述非常非常大的整形。
- 字符串，可以使用双引号也可以使用单引号，字符的索引从 0 开始，最后一个 -1， 好奇葩的定义。

问题

- pycharm 的 python 不支持中文。解决方法是:1 在每一个文件的头部增加 # coding:utf-8 这个注释，然后在 setting 中将文件编码方式设置为 utf-8 即可。

代码样例

文件读取

```
the_file = open("SH600000.txt")

for each_line in the_file:
    print (each_line)
the_file.close()
```

类的定义

```
class A:  
def __init__(self, value):  
    print ("a is init")  
    self.mValue = value
```

使用的时候

```
a = myclass.A(2)  
print (a.mValue)
```

看起来很神奇，init函数中的self相当于this，所以对于类来说，都是静态函数。直接调用A(2)事实上就是调用init产生了一个A的对象。注意，不需要声明成员变量，这很方便。

类的继承

```
class A(list):  
def __init__(self, value):  
    list.__init__()  
    print ("a is init")  
    self.mValue = value
```

相关环境搭建

Python相关环境搭建

MySQL的python库安装

在python的标准库中是不包含MySQL的。具体的安装方法请参考: [MySQL库安装](#)

MySQL的图形界面

参考[MySQL Workbench](#)

Python核心编程

Python核心编程笔记

记录学习Python核心编程的笔记.

第二章 快速入门

Chapter2

字符串操作

字符串的索引从0开始，最后一个-1。[begin:end] 这种表述的区间是 [begin, end)。

```
myString = 'HelloWorld'
print myString

print "string test: %c %c %s %s %s" % (myString[0], myString[-1], myString[1:-1])
```

输出结果：

```
string test: H d el HelloWorld HelloWorld
```

```
powerString = 'abc'*3
```

输出：

```
abcabcabc
```

这种表示可以用来划出分割线。

下面是测试代码

```
import sys

# print >> sys.stderr, 'Fatal error: invalidate pinut'

logfile = open('./log.txt', 'a')
print >> logfile, 'Fatal Error: invalid input'

logfile.close()

# num = raw_input('Now enter a number: ')
# print 'Double your number: %d' % (int(num) * 2)

# help(raw_input)
n = 3
print --n

alist = [1, 2, 3, 4]
print alist[2:]
print alist[3]

aTuple = ('name', 1, 2, 'year')
print aTuple
print aTuple[:3]

aDict = {'host': 'earth', 'port':80}
```

```
print aDict

print aDict['host']

for key in aDict:
    print key, aDict[key]

a = 2

if a == 2:
    print 'a is 2'

elif a < 2:
    print 'a less 2'

else:
    print 'a > 2'

while a > 0:
    print 'a is %d' % a
    a -= 1

for num in [1, 2 ,3]:
    print num,

print '\n'

print range(1, 9, 2)
print range(4)

foo = 'abc'

for i, ch in enumerate(foo):
    print ch, '%d' % i,

print '\n'

squared = [x**2 for x in range(4)]

print squared

sqdEvens = [x**2 for x in range(4) if not x%2]
print sqdEvens

# fobj = open('log.txt')
#
# for eachline in fobj:
#     print eachline,
#
# fobj.close()

def addMe2Me(x=2):
    'apply + operation to argument'
```

```

    return (x+x)

x = addMe2Me()

print x

class FooClass(object):
    'my very first class: foo class'
    version = 0.1
    def __init__(self, nm = 'John'):
        'constructor'
        self.name = nm
        print 'Create a class instatns for', nm

    def showname(self):
        'display instance attribute and class name'
        print 'Your name is', self.name
        print 'm=My name is', self.__class__.__name__

    def showver(self):
        'display class attribute'
        print self.version

foo1 = FooClass()

foo1.showname()

print dir(sys)

```

输出：

```

Python is number 1
abcaabcabc
3
[3, 4]
4
('name', 1, 2, 'year')
('name', 1, 2)
{'host': 'earth', 'port': 80}
earth
host earth
port 80
a is 2
a is 2
a is 1
1 2 3

```

```

[1, 3, 5, 7]
[0, 1, 2, 3]
a 0 b 1 c 2

```

```

[0, 1, 4, 9]
[0, 4]
4

```

```
Create a class instatns for John
Your name is John
m=My name is FooClass
['__displayhook__', '__doc__', '__egginsert', '__excepthook__', '__n:
```

第三章 Python基础

Chapter3

特殊符号

"\" 连接上下两行。

```
a = 3 \
    + 2
print a
```

赋值

多元赋值,可以处理C语言中 x, y使用临时变量互相赋值的case.

```
(x, y, z) = (1, 2, 'abc')
print "x=%d, y=%d, z=%s" % (x, y, z)
(x, y) = (y, x)
print "x=%d, y=%d, z=%s" % (x, y, z)
```

输出：

```
x=1, y=2, z=abc
x=2, y=1, z=abc
```

name属性

对于一个模块来说，如果想要被执行，将 `__ name__ = main`，而如果只是被调用，那么 `__name__ = 模块名`.

第四章 Python对象

Chapter4

type函数

类型也是对象，所以type()返回的是一个类型对象，而不是一个字符串。

列表逆向输出

```
foostr = 'abcdef'  
print foostr[::-1]
```

输出：

```
fedcba
```

is进行对象指针的比较

```
x = [1, 2, 3, 4]  
y = x  
z = [1, 2, 3, 4]  
  
print x is y  
print x is z  
print x is not y
```

输出：

```
True  
False  
False
```

布尔运算符

and or not

几个有用的内建函数

cmp(obj1, obj2) 比较。

str() # 致力于获取到一个对象的好的字符串表述。所以无法使用 eval() 函数来重新建立该对象。

repr() # 该函数返回对象的字符串形式。可以使用eval()来重建对象。

操作符与repr()函数功能一致。

python是不支持函数重载的。

isInstance和type的用法

```
def displayNumType(num):
    print num, 'is',
    if isinstance(num, (int, long, float, complex)):
        print 'a number of type:', type(num).__name__
    else:
        print 'not a number at all'

displayNumType(-69)
```

输出：

-69 is a number of type: int

isinstance与java中是一样的。type则负责输出该类型的名字。

类型工厂

int() 事实上int是类，所以int()是个方法。

Python的内建类型

数据类型	存储模型	更新模型	访问模型
数字	Scalar	不可更改	直接访问
字符串	Scalar	不可更改	顺序访问
列表	Container	可更改	顺序访问
元组	Container	不可更改	顺序访问
字典	Container	可更改	映射访问

不支持的类型

char, byte. Python没有float和double两种类型，只有double. 另外，还有一种Decimals用的任意精度的数值，用来处理金钱这类数据很有用。

第五章 数字

Chapter5

除法

"/" 和 "//" 的区别。"/" 现在的性质与 Java 是一样的，"//" 的结果是去掉小数部分，返回一个最接近的整数。

```
# from __future__ import division
a = 1/2
b = 1 // 2
c = 1.0 / 2.0
d = 1.0 // 2.0
e = -1 // 2
f = -1 / 2
print a, b, c, d, e, f
```

输出：

0 0 0.5 0.0 -1 -1

看到最上面注释掉的，如果打开：

```
from __future__ import division
a = 1/2
b = 1 // 2
c = 1.0 / 2.0
d = 1.0 // 2.0
e = -1 // 2
f = -1 / 2
print a, b, c, d, e, f
```

输出：

0.5 0 0.5 0.0 -1 -0.5

如果加上上面的引用，那么 "/" 则执行真正的除法。

位运算

位运算与 C 语言是一样的。

一些对数值进行处理的内建函数

```
m = coerce(1.1, 11L)
```

```
print abs(-1), coerce(1.3, 134L), m[1], divmod(10, 3), pow(2, 5), rou
```

输出：

```
1 (1.3, 134.0) 11.0 (3, 1) 32 3.45
```

round 计算四舍五入，以及小数点后面几位。

函数	功能
abs(num)	返回 num 的绝对值
coerce(num1, num2)	将num1和num2转换为同一类型,然后以一个元组的形式返回
divmod(num1, num2)	除法-取余运算的结合。返回一个元组(num1/num2,num1 % num2)。对浮点数和复数的商进行下舍入(复数仅取实数部分的商)
pow(num1, num2, mod=1)	取 num1 的 num2次方,如果提供 mod参数,则计算结果再对mod进行取余运算
round(flt, ndig=0)	接受一个浮点数 flt 并对其四舍五入,保存 ndig位小数。若不提供ndig 参数,则默认小数点后0位。
hex(num)	将数字转换成十六进制数并以字符串形式返回 oct(num) 将数字转换成八进制数并以字符串形式返回
chr(num)	将ASCII值的数字转换成ASCII字符,范围只能是0 <= num <= 255。
ord(chr)	接受一个 ASCII 或 Unicode 字符(长度为1的字符串),返回相应的ASCII 或 Unicode 值。
unichr(num)	接受Unicode码值,返回 其对应的Unicode字符。所接受的码值范围依赖于你的Python是构建于UCS-2还是UCS-4。
randrange()	它接受和 range()函数一样的参数, 随机返回range([start,]stop[,step])结果的一项几乎和 randint()一样,不过它返回的是二者之间的一个浮点数(不包括范围上限)。
uniform()	类似 uniform() 只不过下限恒等于 0.0,上限恒等于 1.0 choice()随机返回给定序列(关于序列,见第六章)的一个元素
random()	

十进制数据

```
from decimal import Decimal
dec = Decimal('.1')

print dec, dec + Decimal('1.0')
```

注意 `dec = Decimal(.1)` 是不对的，必须使用字符串。这很好理解，因为Decimal是语言本身定义出来的，所以使用double进行转换，也是无法转换的。

随机

第六章 序列:字符串、列表和元组

Chapter6

数组的索引

```
s = 'abcde'

negtiveRange = range(-1, -len(s) - 1, -1)
positiveRange = range(0, len(s))

print s
for i in negtiveRange:
    print 's[%d]:%s' % (i, s[i]),

print '\n'
for i in positiveRange:
    print 's[%d]:%s' % (i, s[i]),
```

输出：

```
abcde
s[-1]:e s[-2]:d s[-3]:c s[-4]:b s[-5]:a

s[0]:a s[1]:b s[2]:c s[3]:d s[4]:e
```

关键理解最后一个字符的索引，可以是-1，也可以是4。每一个字符的索引都可以是负数。这一点方便反向遍历。

对于切片，

```
negtiveRange = [None] + negtiveRange

print negtiveRange
for i in negtiveRange:
    print s[:i]
```

输出：

```
[None, -1, -2, -3, -4, -5]
abcde
abcd
abc
ab
a
```

这里要特别注意None的用法，因为数组的最后一个元素的索引是-1，那么-1之后是什么，可以用None来代表，也可以使用len(s)+1。但是使用None显得更加优雅。下面是：

```
print s[0:len(s)+1]
print s[0:None]
print s[-1:-len(s)-1:-1]
```

```
print s[-1:None:-1]
```

输出：

```
abcde  
abcde  
edcba  
edcba
```

字符串

在python中，没有字符这个概念，所以都是字符串。内建函数 str 将一个对象转换成字符串。注意这个函数是将一个对象转换成最好的字符串表示方式。

```
print str(range(4))
```

输出：

```
[0, 1, 2, 3]
```

字符串可以直接按照Ascii进行比较。

```
x = 'abc'  
y = 'aBc'
```

```
print x < y
```

输出：

```
False
```

```
# 判断字符串是否包含另外的字符串  
print 'ab' in 'cabde', 'ab' not in '123'
```

输出：

```
True True
```

字符串连接，虽然+可以实现字符串连接，但是没有下面的方式性能好。

```
strJ = '%s%s' % ('ab', 'cd')  
print strJ
```

```
strJ2 = ''.join(('abddd', 'cd'))  
print strJ2
```

输出：

```
abcd  
abddcd
```

另外的长的字符串分行写：

```
print 'kkk''jjj'
```

字符串模板

一些常用的字符串格式，可以使用字符串模板来搞定。

原始字符串的保留

```
print r'\n'
```

输出:

```
\n
```

在字符串前面加'r'，因为有很多时候需要保留原始字符串，例如文件路径中的\n等等。这种情况使用在字符串前面增加r就可以了。

zip 函数

```
print zip('abc', '123')
```

输出:

```
[('a', '1'), ('b', '2'), ('c', '3')]
```

''' 三个引号的字符串

三个引号的字符串和r起到的作用是一样的。可以防止特殊字符。例如html或者sql.

关于unicode 编码

如果文件中使用了 # coding:utf-8 那么，则文件中的就是unicode,如果没有这个标志，那么需要 u'你的字符串' 并进行 .encode('utf-8')这种方式来处理。

unicode 规则:

- 程序中出现字符串时一定要加个前缀u.
- 不要用str()函数,用unicode()代替.
- 不要用过时的 string 模块 -- 如果传给它的是非 ASCII 字符,它会把一切搞砸。
- 不到必须时不要在你的程序里面编解码 Unicode 字符.只在你要写入文件或数据库或者网络时,才调用 encode() 函数;相应地,只在你需要把数据读回来的时候才调用 decode() 函数.

失误 #1: 你必须在一个极有限的时间内写出一个大型的应用,而且需要其他语言的支持,但是产品经理并没有明确定义这一点。你并没有考虑 Unicode 的兼容,直到项目快要结束...,这时候再添加 Unicode 的支持几乎不太可能,不是吗?

结果 #1: 没能预测到最终用户对其他语言界面的需求,在集成他们用的面向其他语种的应用时又没有使用 Unicode 支持.更新整个系统既让让人觉得枯燥和更是浪费时间。

失误 #2: 在源码中到处使用 string 模块或者 str() 和 chr() 函数.

结果 #2: 通过全局的查找替换把 str() 和 chr() 替换成 unicode() 和 unichr(),但是这样一来很可能就不能再用 pickle 模块,要用只能把所有要 pickle 处理的数据存成二进制形式,这样以来就必须修改数据库的结构,而修改数据库结构就意味着全部推倒重来.

失误 #3: 不能确定所有的辅助系统都完全地支持 Unicode.

结果 #3: 不得不去为那些系统打补丁,而其中有些系统可能你根本就没有源码.修复对 Unicode 支持的 bug 可能会降低代码的可靠性,而且非常有可能引入新的 bug.

总结: 使应用程序完全支持 Unicode,兼容其他的语言本身就是一个工程.

格式化时候，要先变成unicode:

```
u"%s %s" % (u"abc", "abc")    u"abc abc"
```

元组

元组是有隐性元组和显性元组，正常来讲我们都应该使用显性元组。

```
# 显性和隐性元组
```

```
x, y = 1, 2
```

```
(a, b) = (3, 4)
```

```
print x, y, a, b
```

对于x,y来说，其实就是隐性元组。

只有一个元素的元组呢 ('xyz') 这不是一个元组而是一个字符串，要表示一个元素的元组是 ('xyz',) 这才是表示一个元素的元组。列表和元组之间的转换 list() 和 tuple().

浅拷贝和深拷贝

对于像数字，字符串等都是所谓的“深拷贝”是没有浅拷贝的说法。元组的内容都是数字或者字符串也是同样的。但是对于列表来说，就要注意深拷贝和浅拷贝的根本区别。

第七章 映像和集合类型

Chapter7 映射和集合类型

字典也就是Java的hashtable，使用{}来表示。列表是[]元组是().

字典的创建

```
# 声明一个空的字典
dict1 = {}

# 声明有数据的字典
dict2 = {'name':'earth', 'port':80}

# 使用dict()工厂方法来创建字典，参数是元组
fdict = dict([('x',1), ('y', 2)])

# 使用fromkeys来创建默认字典
ddict = {}.fromkeys('x', 'y'), -1)

edict = {}.fromkeys('foo', 'bar')

print dict1
print dict2
print fdict
print ddict
print edict
```

输出：

```
{}
{'name': 'earth', 'port': 80}
{'y': 2, 'x': 1}
{'y': -1, 'x': -1}
{'foo': None, 'bar': None}
```

集合

集合操作：

```
# coding:utf-8

# 创建集合，可变集合不要求一定可hash
aset = set('abcde')
print aset

# 集合的操作
aset.add('q')
aset.add(123)
```

```
print aset

# 创建不可变集合. 不可变集合必须是可哈希的
bfrozen = frozenset('abcdef')

print bfrozen

# 集合支持 交 并 补 和 差分
bset = set('uabc')

print aset | bset
print aset & bset
print aset - bset
print aset ^ bset
```

输出：

```
set(['a', 'c', 'b', 'e', 'd'])
set(['a', 'c', 'b', 'e', 'd', 'q', 123])
frozenset(['a', 'c', 'b', 'e', 'd', 'f'])
set(['a', 'c', 'b', 'e', 'd', 'q', 'u', 123])
set(['a', 'c', 'b'])
set(['q', 123, 'e', 'd'])
set(['q', 'e', 'd', 'u', 123])
```

第八章 条件和循环

Chapter8 条件和循环

在这一章，最应该搞清楚的就是列表解析和生成器。

```
# coding:utf-8

# 三元表达式

x = 1
y = 2

smaller = x if x < y else y

print smaller

# 索引和数据同时获取 enumerate()

nameList = ['Jhone', 'Shirley', 'Ben']

for (i, eachLee) in enumerate(nameList):
    print '%d %s Lee' % (i+1, eachLee)

# 空语句 pass的代替.类似下面的语法,如果去掉pass,会显示错误.

if True :
    pass
else:
    pass

# for ... else; while ... else 会在循环break跳出的时候是不会执行的,只有非break跳出是因为发生了问题,那么问题的处理就应该在break的时候处理
# 没有任何问题才会进入到else处理.
# 换句话说 break处理错误, else处理正确.

i = 3

while i > 0:
    if (i > 4):
        print 'in break %d' % i
        break;
    i = i -1
    print 'while', i

else:
    print 'break for else'

# 迭代器使用 next来访问
nameIt = iter(nameList)
```

```

while True:
    try:
        name = nameIt.next()
        print name

    except StopIteration:
        break

# 文件迭代器

myFile = open('log.txt')

for eachLine in myFile:
    print eachLine

myFile.close()

# 列表解析 重中之重 [expr for iter_var in iterable]
# 列表解析的表达式取代内建的map以及lambda 效率更高
print [x**2 for x in range(6) if x % 2]

print [(x+1, y+1) for x in range(3) for y in range(5)]

x = [1, 2, 3, 4]

# 计算文件中的所有单词数目，下面使用的是生成器表达式，比列表解析效率更好

myFile = open('log.txt')

print sum(len(word) for line in myFile for word in line.split())

myFile.close()

# 列表解析和生成器 是 python有别于其他语言的根本区别

# 输出

/System/Library/Frameworks/Python.framework/Versions/2.7/bin/python2
1
1 Jhone Lee
2 Shirley Lee
3 Ben Lee
while 2
while 1
while 0
break for else
Jhone
Shirley
Ben
Fatal Error: invalid input

Fatal Error: invalid input

```

```
Fatal Error: invalid input  
Fatal Error: invalid input
```

```
Fatal Error: invalid input
[1, 9, 25]
[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 1), (2, 2), (2, 3), (2,
1081

Process finished with exit code 0
```

第九章 文件输入和输出

Chapter9 文件输入和输出

这里的文件是一种广义的概念，而并非是传统意义的文件。所谓的文件就是一个字节快。

在文件处理中，file()和open()函数是可以互相替换的。

UNS

因为不同的系统的换行符是不同的，所以在打开文件的时候使用 rU或者Ua就可以统一换行符为 \n. 这个特性默认是打开的，在config中可以关闭。

遍历文件

```
for eachLin in file:  
    do smoethin
```

这种方式使用了文件迭代器，效率更高。因为 file.readlines()会将所有的行都缓存，所以，当大文件的时候，用这种方法就会有问题。

命令行参数

sys.argv

对于文件的路径和目录的操作

都在os模块中。

第十章 错误和异常

Chapter10 错误和异常

```
# coding: utf-8

## 异常处理

def safe_float(obj):
    'safe version of float()'
    try:
        retval = float(obj)
    except (ValueError, TypeError), diag:
        retval = str(diag)
    else:
        print '没有任何错误'
    finally:
        print '最后还是要执行'

    return retval

print safe_float(11)

# raise 似乎与throw是一样的? sys.exc_info()可以获得异常信息
try:
    assert 1==2
except AssertionError:
    import sys
    exc_tuple = sys.exc_info()
    print exc_tuple

输出:
/System/Library/Frameworks/Python.framework/Versions/2.7/bin/python2
没有任何错误
最后还是要执行
11.0
(<type 'exceptions.AssertionError'>, AssertionError(), <traceback ob:

Process finished with exit code 0
```

第十一章 函数和函数式编程

Chapter11 函数和函数式编程

python支持内嵌函数，也就是一个函数内部可以再声明一个函数。因为函数本身来说，就是类。

函数的装饰者

这就是装饰者模式。装饰者，在数学的表达很优雅 $g(f(x))$ 就是这种表达式。

关于装饰者的详解。在Java的装饰者模式，我们清楚是对一个接口的装饰。这和函数的装饰本质上是一样的。那么，在Java的装饰者模式中，是如何进行处理的呢？

1 需要传递一个接口对象，作为装饰类的参数 2 调用接口的时候，会先执行装饰的接口调用，再次执行真正本身的调用。

那么，上面的步骤在函数的装饰者中是如何执行？

无参函数装饰

```
print '... dec'
def dec():
    def wraper(f):
        print 'dec is called'
        return f

    return wraper

@dec()
def f():
    print 'f is called'

f()
```

逐条进行分析。

1 需要传递一个接口对象，作为装饰类的参数。这里的解释是，`dec()`中没有参数，那么 `f` 会作为参数传递给 `dec()` 中的第一个内部函数，也就是 `wraper` 的参数。

2 执行。执行的时候，先执行 `dec() -> wraper -> f` 注意，最后返回的是 `f` 函数，也就是说经过装饰之后，最后返回的是 `f`

有参数的装饰者

在这里必须搞清楚一点，就是在装饰者模式中，因为不论是装饰者还是被装饰者都是继承自同一个接口，所以，二者的接口是一样的，也就是参数是一样的。而在python的函数装饰中，可能就会产生不一样的参数。

```
def decarg(a, b):
```

```

print 'decarg is called', a, b

def wraper(func):
    def test(a, b):
        print 'test is called', a, b
        return func()

    return test

return wraper

@decarg('a', 'b')
def farg():
    print 'farg is called'
    return 3

print farg('a', 'b')

```

输出：

```

... decarg
decarg is called a b
test is called a b
farg is called
3

```

继续逐个分析：

1 参数传递。这里显然 func 就是 frag 函数。2 执行。注意这里的，decarg 和 farg 的参数是不一样的。那么如何来处理这个问题呢？增加了一个新的内部函数 test。所以这里的调用顺序是这样的：decarg->wraper -> test ->func

这里看起来还有一点疑惑的地方，是什么呢？是不是感觉和无参的对比多了一层 test？如果去掉 test，直接返回 func 呢？会得到一个错误。所以关于有参数的装饰器来说，farg('a', 'b') 的参数 ('a', 'b') 究竟传给了谁？是 wraper 函数的返回值。所以对于装饰器来说的真的执行顺序是这样的：

decarg => Run => return wraper => frag 作为参数传递给 wraper => wraper run => return test => ('a', 'b') 作为参数传递给 test => test run => frag Run => return frag 的执行结果

如果和类对比，到很有意思。decarg 类似于最外层的类，wraper 是接口的实现，执行完 wraper，执行 func。而 test 是一个适配器，为了解决参数不一致的适配器，最终执行 func。

注意：@decarg('a', 'b') 这里的 'a', 'b' 表示传递给 decarg 函数的，不是形参。

生成器

生成器函数和函数长成一个样子，难道是靠内部有 yield 来标识这是一个生成器吗？

```

def simpleGen():
    yield 1
    yield '2-->punch'

myG = simpleGen()
print myG.next()

```

```
print myG.next()  
  
for eachItem in simpleGen():  
    print eachItem
```

输出：

```
1  
2-->punch  
1  
2-->punch
```

生成器的最大的威力在于进行“协同程序”作战。yield 将当前程序挂起，然后去执行其他的程序，执行完其他程序再继续执行当前程序。可以进行多线程和多进程的程序的协同工作。

通过对生成器传递参数也可以。

```
count = counter(5)  
print count.next(), count.next()  
  
count.send(9)  
  
print count.next()  
count.close()
```

输出：

```
5 6  
10
```

第十二章 模块

Chapter12 模块

模块的路径搜索

PYTHONPATH 这个环境变量中定义了python的模块路径。也可以在程序中导入，使用 `sys.path.append('/home/wesc/py/lib')` 来进行导入。

缩短导入

```
import Tkinter as tk  
from cgi import FieldStorage as form
```

使用 `as` 关键字。

第十三章 类

Chapter13 类

接口是如何体现的？

多态

```
# coding:utf-8

class P(object):
    def foo(self):
        print 'I am p foo'

class C(P):
    def foo(self):
        P.foo(self)
        print 'I am c foo'

c = C()
c.foo()
```

看看 `c=C()` 按照Java应该是 `P c = new C()`，但是因为python不需要写类型。所以 `c = C()` 这就是多态的表示了。

这里要注意一点，当覆盖`__init__`的时候不要忘记调用父类的`__init__`。

动态添加属性

这个有点夸张呀。

关于private,public,protected

Python中是没有这些关键字的。通过 Attr 这种方式来混淆Attr属性，实际上是被混淆成了额 ``ClassName_Attr``，所以这种混淆本质上意义不大，但是可以有效的区分，父类和子类的同一个属性。

另外在模块中的单下划线的好处是，当模块被加载的时候 属性不会被执行。

描述符

描述符是针对属性的，是属性的一个代理。“get”获取，“set”修改，“delete”删除。就是实现类中的`__get__`, `__set__` 和 `__delete__`。而我们真正需要的是类似java的getter和setter，那么这是通过 property 来实现的。避免使用描述符。

```
# coding:utf-8
```

```

class DevNull2(object):
    def __get__(self, instance, owner):
        print 'Accessing attribute ... ignoring'

    def __set__(self, instance, value):
        print 'attempt to assign %r ... ignoring' % (value)

class C2(object):
    foo = DevNull2()

    def __init__(self, x):
        assert isinstance(x, int), \
            '''x must be an integer'''

        self.__x = ~x

    def get_x(self):
        print 'get_x is called'
        return ~self.__x

    def set_x(self, val):
        print 'set_x is called'
        self.__x = ~val

    x = property(get_x, set_x)

c2 = C2(5)

# 下面是对foo进行赋值,那么就会默认执行 __set__
c2.foo = 'bar'

# 下面是获取foo,也就是会调用 __get__
x = c2.foo

# 这里的描述符和Java是有些区别的,与getter和setter
# Java中是写在C的类中,对foo进行封装,而这里是对foo所属的类进行封装

# 类似于java中的封装是 property这个函数的处理
a = c2.x
c2.x = 9

```

输出:

```

attempt to assign 'bar' ... ignoring
Accessing attribute ... ignoring
get_x is called
set_x is called

```

使用下面的技巧更加容易实现getter和setter

```

# 先前的property使得getX和setX能够被外部调用,下面的技巧将避免这个问题

class HideX(object):

```

```
def __init__(self, x):
    self.__x = ~x

@property
def x(self):
    return ~self.__x

@x.setter
def x(self, val):
    self.__x = ~val

h = HideX(2)
print h.x
```

关于抽象类和接口

参考: [抽象类和接口](#)

第十四章 执行环境

Chapter14 执行环境

这里介绍的是调用其他或者操作系统相关的，到实际应用到的时候再查询。

第十五章 正则表达式

Chapter15 正则表达式

这一章是正则表达式，需要在实践中逐渐学习。

第十六章 网络编程

Chapter16 网络编程

使用 `SocketServerTCP` 这个标准库会让你省事许多。然而，使用 Twisted 框架会让你的程序更加简单。

[Twisted框架](#)

第十七章 网络客户端编程

Chapter17 网络客户端编程

邮件挺有意思的。

第十八章 多线程编程

Chapter18 多线程编程

Python中线程三个部分，`thread`, `threading`, `queue`，其中 `thread`是低级的线程库，我们不应该使用。而是使用高级的`threading`.

守护线程

使用`Thread`模块，当主线程退出时，所有的子线程都会退出，不管子线程是否在工作。有时候，我们是不期望这种行为，这时候就引入了守护线程。

`Threading`模块支持守护线程，守护线程表示这个线程不重要的，当进程退出的时候不用等这个线程是否退出。如果你的主线程退出的时候不用等待那些子线程完成，那么设置成守护线程即可。而如果你想等待子线程完成后再退出，那么就用普通的线程即可。

总结：非守护线程的退出，会被主线程退出等候，直到所有的非守护线程都退出了，主线程才会退出；而守护线程，可以不用管是否退出，主线程都可以退出。

join

`join`是挂起当前主进程，当当前线程执行完毕后，主进程继续执行。用这种方法，可以保证所有线程都执行完毕，主线程再继续执行。

另外，`join`也可以不用调用，因为当线程函数执行完毕，线程结束。主线程可以不用调用`join`，而去做其他的事情。

类

当以类作为线程的执行对象的时候，是调用的类中的`__call__`函数，这和java是一样的。

也可以集成`threading.Thread`类，执行其中的`run`函数。

Queue

看来`Queue`本身就是线程同步的。

实战技巧

实战技巧

Unicode的处理

在使用python处理文本的时候，存在各种不同的编码方式。可能是gbk,utf-8以及acii.对于python的字符串来说，本质是存储的是字节码，注意是字节码。那么utf-8只是将多个字节码组织在一起的一种新的表现形式。明白这个道理，就很容易明白了。

```
a = '你好'

print a
print repr(a)

print 'test b', '-'*20
b = u'你好'
print b
print repr(b)
print b.encode('utf-8')
print repr(b.encode('utf-8'))
```

输出：

```
你好 # 以utf-8的表达形式输出
'\xe4\xbd\xba\xe5\xbd' # 是 a='你好'的字节码
test b -----
你好 # 以utf-8的表达形式输出
u'\u4f60\u597d' # utf-8的编码表现形式
你好 # 变成utf-8的表现形式
'\xe4\xbd\xba\xe5\xbd' # 在encode之后，在print输出的是字节码
```

那么如果是以字符串形式展示的utf-8，比如: d = "\u4f60\u597d". 该如何转化成字节码呢? dee = d.decode('raw_unicode_escape'); 使用这种进行decode.

注意的要点

格式的统一

对于字符串进行组合的时候要统一格式. 如下例子:

```
a = '你好'
b = u'我很好'
```

虽然在字节码这个层面上是一样的，但是对于字符串来说是不同。a表示的是字节码，'\xe4\xbd\xba\xe5\xbd'.而对于b来说是 u'\u4f60\u597d'，所以a,b不能放在一起进行操作。如果要放在一起进行需要将a转成utf-8的形式或者将b转成字节码的形式。

```
a = '你好'
```

```
b = u'我很好'

c = a + b.encode('utf-8')
print c, repr(c)

c = unicode(a, 'utf-8') + b
print c, repr(c)

c = a + b
print c
```

输出：

```
你好我很好 '\xe4\xbd\xae\x88\x91\xbe\x88\xae\xae\x
你好我很好 u'\u4f60\u597d\u6211\u5f88\u597d'
Traceback (most recent call last):
  File "/Users/panxu/Documents/MyProjects/kmeanGame/testUnicode.py",
    c = a + b
UnicodeDecodeError: 'ascii' codec can't decode byte 0xe4 in position
```

如果没有进行转换，就会出现错误。

Private和Protected的表现方式

abc : 用两个下划线作为前缀是private
abc: 用一个下划线作为前缀是protected

注意：这是语法层面的私有化，而不是真正意义的私有化，外界还是能破解调用的。
不过java也是一样反射依然可以调用。

用**Python**做科学计算

用**Python**做科学计算

主要是一本网络图书: [用Python做科学计算](#)

其中，介绍了 scipy和numpy以及matplotlib等工具和库的使用.

关于维度

a.shape (6,)表示1行6列; (6,1)才表示6行1列。

数值计算要点 1

数值计算要点 1

getA() in np

对于矩阵来说，通过getA来将数组类型变成ndarray，因为在matplotlib中需要的是ndarray类型。

绘制散点

```
ax.scatter(xcord1, ycord1, s=30, c='red', marker='p')
```

其中的 marker 表示点的形状。xcord1是array，所以一维矩阵不行，必须转换成array.

random.uniform

返回指定范围的随机浮点值[small, big]。和random.random()是返回[0.0, 1.0)的浮点值。

数值计算的集中类型

1. list 普通的列表，这是python自身的数据类型，不能进行矩阵运算
2. numpy.array 这是可以进行对其进行统一运算的。例如 $A = \text{array}(\text{ListA})$. $A * 3$.但是，他不是矩阵，他的性质只是能够对每一个元素进行ufunc操作而已。
3. numpy.matrix 是像矩阵一样处理。所以需要matrix(ListA)或者matrix(ArrayA) 他们又是可以互相转化的使用 mat.A或者mat.getA转换成array.

错误: s32 类型错误

这是说明在处理数据的时候，没有转换成float，而数据依然是字符串类型。

pandas 要点

pandas要点

读取csv: titanic_df = pd.read_csv(TRAIN_PATH)

显示头几行: titanic_df.head()

显示每一列的数量以及类型: titanic_df.info() 这可以用来看是否有缺失值

量化

量化宣言

人生的梦想只有全力去追寻，哪怕是捧得头破血流。你能做的只有一件事，那就是最快速的前进。不要畏惧失败，因为成功就在失败之后。

量化交易计划

我的量化交易目标。

成立私募

在量化领域做出成绩

进入量化领域

自动化的交易程序

机器学习与数据分析

量化交易框架

量化交易理论

目前，我在进行的是机器学习和量化交易框架，但是，并没有引入量化交易理论。

机器学习

每日必须完成2课，这样才能在1个月之内完成。也就是春节假期结束，机器学习必须完成第一遍。然后，再用一个月的时间进入到第二轮编程实战阶段。

scikit-learn 学习

必须引入这个工具的学习，因为这个工具是进行量化分析的基础，有助于我们进行数据的分析与理解。NG的课程只是最基本的课程。

量化知识

1个月的时间完成《统计套利》。每3天必须完成一章。这一章应该有详细的笔记。春节前完成这本书。然后开始下一本

零碎时间的学习与积累

概率论。每天的课程其实不多也就20分钟。这一部分通过早晨完成，去了先学习概率论。然后，对于课后习题，是通过将课件打印出来与习题打印出来。下班后，回家之后搞定。scikit-learn. 每天在工作中抽出时间就可以搞定一部分的学习。不看其他新闻，专心学习scikit-learn. 先把概率论和scikit-learn学完。在学完逻辑回归就要进入到优矿的练习了。先将自己的最简单的择时模型搞定。然后，在优矿上进行回测。考察一下tushare的数据可靠性，如果能够拿到嘉幸的数据，那就最好了。kaggle的学习，这一点必须进行。因为通过scikit-learn和概率论的学习，可以进行kaggle的训练。这样可以培养自己对数据的敏感性。下一步，阅读最新的关于量化的paper.

统计套利

统计套利

这里主要介绍统计套利的读书笔记

第一章 蒙特卡罗的谬误

第一章 蒙特卡罗的谬误

最初的摩根士丹利的方法是找到相关性很高的两只股票，然后，当两只股票在出现大的价格偏差的时候买入价格低的股票，而在两只股票价格进入到相似的时候，则清空头寸。

第二章 统计套利

第二章 统计套利

许多实际发生的状况，都可以被轻易地视为随机变化，但是，有时隐藏在变化中的重要信号可能是要警告我们，除了什么问题，或是提示我们出现了新的机会。

当大陆航空和美国航空的价差波动是在[-2,4]之间波动，那么就在0的时候买入，在4的时候卖出。在这样的规律得出之后，继续得出另外的规则。

规则2 反向的情况进行交易

规则3 在更强的进场点进入，重复交易

所谓的反向交易就是在0的时候进行反向的多头或者空头。

重复交易设计了多个进场点。

使用多长时间的历史价格对交易规则进行校验？这是一个很关键的问题。因为你进行交易的参数，因为选取不同的时间段，会有不同的有效性。

不好的交易规则是使用了极大值和极小值。极值建模是一个复杂的研究领域，要小心。采用均值会修正得好些。其中典型的就是布林线。但是，布林线所产生的极值占用的权重，会导致极值更加不稳定。所以会采取去掉样本中的极值。

金融数据是非正态分布的，会常常出现非对称的情况，在距离均值几个标准差的位置会出现较大的观察值。这就是所谓的“后尾”现象。后尾是与正态分布相比的，并不是这些金融数据有什么特殊情况。所以，采用正态分布，计算尾巴代表的概率会错估风险，通常意味着低估。但是，使用禁烟分布，也就是数据分布，而不是假设的数据公式，大多数这种错误是能避免的。

使用样本均值和标准差，产生很多错误。尤其是反转现象非常明显，幅度也很大。粗糙的标准差标准不再是好的交易方式，因为期望的交易回报出现萎缩，并低于交易成本。可以通过设置最小的回报率下限方法来解决这个问题。

目前为止，所涉及的交易规则都强烈地认为，价差会在均值上下系统性波动。这种随时间波动的模式，其原型是正弦波。但是，很多时候会出现爆米花的形态。随着一个突然出现的干扰，价差会突然偏离均值，然后慢慢向均值回归。在这个模型中，取消了价差模型过度波动的限制。

正弦函数的方程是：

$$y(t) = \sin(t)$$

爆米花函数是：

$$y(t) = I(t)\sin(t)$$

$I(t)$ 是一个指示函数，不是1就是-1，用于表示封顶或者峰谷。

爆米花过程提出了一个新的交易规则：当价差回归均值，就产生离场信号。而不是价差会从建立头寸的部位，超过均值向相反方向移动。

规则4 当价差比均值偏差增加（减少）足够大（如K个标准差），卖出（买进）这个价

差；当价差回到均值的时候，清空头寸。

识别匹配交易

早起的股票按照行业进行分类，每个分类中的承兑股票都是候选的交易机会。风险管理使用Barra模型，构建投资组合，然后从差异性很大的匹配交易投资组合中识别出因素暴露的偏差（例如使用股票或者标准普尔期货来抵消 β 风险）。

关于匹配交易的另外一种理解。那就是寻找相关性很低或者负相关的股票，这样做不是收益更好吗？事实不是这样的：短期看，高相关的交易会错过一些获利机会，但是长期看，极大地改善了风险状况。与具有相同运动趋势的股票相比，哪些出现负相关的股票，将来更有可能与市场整体的发展方向背道而驰。所以某个时候，不相关的股票非常有可能产生代价高昂的匹配交易。

风险最小化为目标的匹配交易

选择相似历史轨迹的股票，也就是出现波峰和波谷的时间很接近，并且出现这些事件的时候，移动的大小程度也相似。因为出现一些大的事件的时候，这些股票不可能出现完全迥异的反应。

利润最大化的目标的匹配交易

选择负相关性较强的股票。

事件分析

- 1 如果价格序列下降一个量，出现一个负回报，这个负回报的绝对值超过了年度回报波动率的
- 2 上升产生一个正回报，也可以当做一个转折点。

对于成交量的事件分析法，有时候也能识别价格序列中无法识别的信息。成交量不能影响价格，但是成交量急剧放大或者缩小，都是一种警告，这种警告是受到了不寻常的交易活动的影响。识别成交量是重要的风险管理公主。在成交量连续增加前被识别，因为事件发生之后才识别就晚了。

所有上面的理论是基于，股票的波动是因为有个事件，如果没有事件那么股价会保持一个平稳的状态。

到目前为止还没有出现容易使用的商业工具去识别事件或者转折点。

投资组合结构和风险控制

投资组合风险管理越来越受重视。均值方差方法在很长一段时间受到人们的喜爱，被当成将风险“控制在一定程度内”的方法。但是，在1998年夏天背证明这种方法是愚蠢的。

为什么？是因为所有的都是基于预测的。

对于风险控制的方法，可以采用如下。首先，必须先理解风险暴露程度。对于一只股票对每个指数的风险暴露，例如上证300，上证50，创业板，外贸系数，等等。把其称为 I 。我们要做的是控制风险，那就是所有的风险暴露程度为0.也就是。

$$I_1 + I_2 + \dots + I_n = 0;$$

接下来，量化风险，风险定义为：投资组合方差的期望 Σ 。而股票的回报的期望值是f, 投资为p. 所以有：

$$F = p^*f - k^*\Sigma - (\lambda_1^*l_1 + \lambda_2^*l_2 + \dots + \lambda_n^*l_n)$$

使用拉格朗日乘数法，求出最小值。

市场冲击力也会产生影响。

第三章 结构模型

第三章 结构性模型

一项交易逐日发生损失暗示着可能存在的潜在问题；预测情况与实际结果之间的差异模式提供的有关信息，有助于说明问题的可能性质。这种信息认为同时关注多项指标对损失情况的反应情况，回避一个迟钝的止损规则更有效果。

指数加权移动平均模型

移动平均的计算方法：

$$MA(n) = MA(n-1) + (Y(n) - Y(1)) / n$$

加权移动平均

$$EWMA(n) = (1-\lambda)Y(n) + \lambda * EWMA(n-1)$$

动态线性模型

波动率的模型

广义自回归条件异方差模型 **GARCH**

随机波动率模型

小波分析

神经网络

神经网络是挖掘数据模式的一种优秀工具。只要同样的模式再次出现，网络的预测能力就会得到非常好的表现。不过，它有一个严重的缺点，就是缺乏可解释性。

神经网络最大的优势，就是他们非常具有弹性，而这也是他们从一开始就能够成为优秀的模式识别方法的原因。当结构发生变化时，神经网络可以非常快速地识别出正在进行的变化，并随后对新的稳定模式进行特征上的描述。但是，这样的弹性也总会伴随着一定的风险，那就是，被识别出来的模式可能只是存在很短暂的时间，从而导致无法使用。欧威尔：“描述现状没问题，但预测未来，行不通。”

分型分析

机器学习课题

机器学习课题

对一些自己的工作和生活的问题使用机器学习进行解决的课题。

课题一 基于神经网络的股票研究

基于神经网络的股票研究

基于逻辑回归，计算明天涨跌的概率。如果上涨的概率较大，则在当天收盘买入；如果上涨概率较小，在持有头寸的情况下卖出，没有持有头寸则不会买入。

特征量的选取

基于价格，成交量的特征

特征量	描述
x1	前1天的收盘价, 开盘价, 最高点, 最低点, 成交量
x2	前2天的收盘价, 开盘价, 最高点, 最低点, 成交量
...	...
xn	前n天的收盘价, 开盘价, 最高点, 最低点, 成交量

基于网络言论人们情绪特征

这种特征通过分析和抓取在主流社区的言论，给出具体数字化的情绪特征。也可以通过搜索引擎对特定股票的搜索进行特征分析。

头寸规模

在得到 $h(x)$ 的基础上，使用凯利公式进行头寸确定

课题二 对自己的邮件进行自动分类整理

分类邮件

描述

自己的邮件很多，那么可以设计一个机器学习的系统来自动对邮件进行重要性分类，以便使得自己能够第一时间查看最重要的邮件。

课题三 最个性网络阅读器

课题三 最个性网络阅读器

我们每天都会被各种新闻，知识，文章等内容环绕，但是，我们真正感兴趣的，其实不多。于是，我们花费了很多时间会逐个的浏览，然后再精读自己喜欢的文章。那么为什么不做一个最个性网络阅读器呢？

内容的来源

1. csdn文章和论坛。因为CSDN有开放平台接口
2. 微博
3. 网页。基于爬虫的网页抓取，然后，根据机器学习对网页内容按照自己的兴趣进行分类。然后，自己能够浏览感兴趣的文章。
4. RSS源

兴趣分类

! 分类 | 描述 || ---- | -----|| 1 | 机器学习 || 2 | Python || 3 | 大数据 |

等等随着不断的深入不断的增加。

课题四 知识分享型机器人

课题四 知识分享型机器人

突破技术

有一种机器人，可以学习任务，同时将知识传送到云端，供其他机器人学习

重要意义

如果不许要对所有类型的机器进行单独编程，那么可以极大地加快机器人的发展进程

主要研究者

Brain of Things, 布朗大学, 加利福尼亚伯克利分校, 德国达姆施塔特工业大学

描述

如果机器人能够独立解决更多的问题，并且互相分享这些内容，那么会怎么样？布朗大学的教授斯蒂芬妮 泰勒正在进行的一项研究，目的使世界各地的研究性机器人学习如何发现和处理简单的物品，并将数据传送到云端，供其他机器人分析和使用。她们已经收集了大约200个物品的数据，并且共享了这些数据，她希望能够建立一个信息库，让机器人能够很容易的获取他们的信息。

概率论

概率论

这是机器学习的数学基础之一，所以需要再学习一遍，打下坚实的数学基础.

[浙江大学网课](#)

第一周

第一周

第1节 样本空间

所谓的样本空间就是所有事件构成的全集，理解这一点很关键，因为只有有了全集，才可以去考虑其他的概率问题。

第2节 事件的相互关系及运算

样本空间是个集合，某个事件也是个集合，所以事件的相互关系就是集合的关系。

$$A-B = A \cap \sim B$$

$A \cap B = \emptyset$ 称作AB不相容或者互斥

习题

3 若A与B不相容，则对于任意事件C与D，AC与BD也不相容。

$$A \cap B = \emptyset, \quad AC \cap BD = C \cap AB = C \cap AD = C \cap DB = C \cap D = \emptyset$$

$$4 \quad A-B-C = A \cap \sim B \cap \sim C$$

$$A-B-C = A \cap \sim B \cap \sim C = A \cap \sim B \cap \sim C$$

第3节 频率

频率的定义: $f_n(A) = n_A/n$ 其中: n_A 是A发生的次数, n 是实验总次数

频率的性质:

1. $0 \leq f_n(A) \leq 1$

2. $f_n(S) = 1$

3. 弱 A_1, A_2, \dots, A_k 两两不相容, 则
$$f_n\left(\bigcup_{i=1}^k A_i\right) = \sum_{i=1}^k f_n(A_i)$$

第4节 概率

随着逐渐增多次数，频率会稳定在一个固定的值，这个值就称作概率。事件A的概率是 $P(A)$. A的概率必须满足三条:

1. 非负性. $P(A) \geq 0$

2. 规范性: $P(S) = 1$

3. 可列可加性。 $P(\bigcup A) = \sum P(A)$

满足这三条称 $P(A)$ 为事件A的概率。

基本等式: $P(A-B) = P(A) - P(AB)$

第二周

Week2

第五讲 等可能概型

古典概率论，要求如下两点：

1. 样本空间S中的样本点，必须是有限的
2. 每个元素出现的概率，必须是等可能的

所以对于事件A来说，A中所包含的样本点除以S的样本点数，就是概率。

$$P(A) = \text{中所包含的样本点} / S \text{的样本点数}$$

问题：如果有N个球，其中a个白球，b=N-a个黄球，采用不放回抽样取n个球($n \leq N$)，记 $A_k=\{\text{恰好取到}k\text{个白球}\}$ ($k \leq a$)，求 $P(A_k)$ ？

- 1 计算S。取出n个球，那么S的样本点数是 $C(N, n)$
- 2 计算 A_k 包含的样本点数。 A_k 的含义是n个球，其中k个白球， $n-k$ 个黄球。那么： $C(a, k) * C(b, n-k) / C(N, n)$
- 3 $P(A_k) = C(a, k) * C(b, n-k) / C(N, n)$

第六讲 条件概率

$P(B|A)$ 表示A发生的条件下B发生的概率。这里可以理解为，在A是一个全新的样本空间，只是A的概率不是1.由此可以看出对于任意概率，事实都是 $P(A|S)$ 在全样本空间下的条件概率。条件概率公式：

$$P(B|A) = P(AB)/P(A);$$

当 $A=S$ 的时候， $P(B|S)=P(BS)/P(S)=P(B)$

注意区别 $P(B|A)$ 和 $P(AB)$ 的含义。 $P(AB)$ 的描述是AB同时发生；而 $P(B|A)$ 是A发生的情况下B的概率。通过公式知道 $P(B|A) > P(AB)$. 都是AB的概率，但是样本空间不同。

条件概率满足概率的所有性质。

计算用到的推导式：

$$P(AB) = P(A) * P(B|A)$$

心得体会：条件概率，是改变了样本空间，这一点理解才是条件概率的根本，不是文字的理解。所以针对取两次球的问题， $A_1, A_2, P(A_1)*P(A_2|A_1)$ 这种条件概率的表述，是因为取走了 A_1 之后，样本空间改变了。

第七讲 全概率公式和贝叶斯

一袋中有a个白球，b个黄球，记 $a+b=n$. 设每次摸到各球的概率相等，每次从袋中摸一球，不放

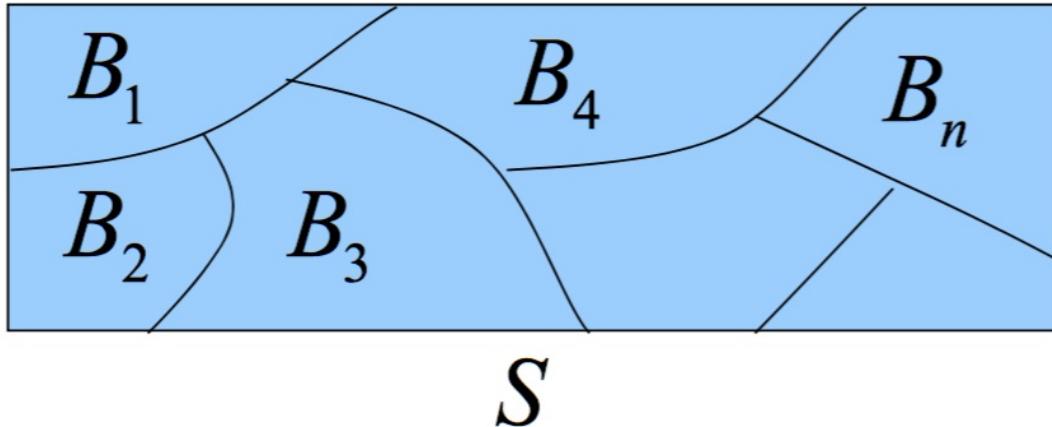
解：

设 A_i 表示第*i*次取到白球， $i=1, 2$

$$\begin{aligned}
 P(A_2) &= P(A_1A_2 \cup \sim A_1A_2) = P(A_1A_2) + P(\sim A_1A_2) \\
 P(A_1A_2) &= P(A_1) * P(A_2 | A_1) = a/n * (a-1)/(n-1) = a(a-1)/n*(n-1) \\
 P(\sim A_1A_2) &= P(\sim A_1) * P(A_2 | \sim A_1) = b/n * a/(n-1) = ab/n*(n-1) \\
 P(A_2) &= (a(a-1)+ab) / n*(n-1) = a(a+b-1)/n*(n-1) = a(n-1)/n*(n-1) = a/n
 \end{aligned}$$

全概率定义：称 B_1, B_2, \dots, B_n 为 S 的一个划分，若

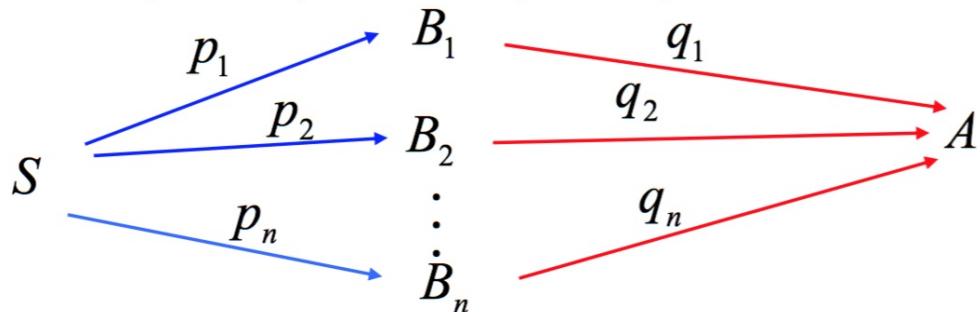
1. 不漏。即 $B_1 \cup B_2 \cup \dots \cup B_n = S$
2. 不重。即 $B_i \cap B_j = \emptyset, i \neq j$, 任意两个不相容.



全概率公式：

$P(A) = \sum P(B_j) * P(A | B_j)$, 其中 B_j 是第 j 个划分， B_j 满足不漏和不重两个重要性质。

设 $P(B_j) = p_j, P(A | B_j) = q_j, j = 1, 2, \dots, n$.



$$\text{则 } P(A) = \sum_{j=1}^n p_j q_j.$$

这样理解全概率， S 被划分了 n 个部分，用 B 来表示，每一个部分有自己的概率，用 p 来表示， A 相对于每一个区域的概率是 q . 所以 $P(A) = pq$. 注意， $P(A)$ 的概率的样本空间是 S . 所以在使用全概率的时候，重要的就是 B 的划分。

已知区域 B 的概率 p , 求 A 的概率 q ，被称作先验概率。那么能否，知道 A 的概率 q ，求 B 的概率呢？这就是后验概率，也就是贝叶斯公式。

$$P(B_i | A) = P(AB_i) / P(A) = P(B_i) * P(A | B_i) / P(A) = p_i * q_i / \sum p_i q_i$$

贝叶斯的语言描述是，A发生时，每个划分的概率。

在机器学习中的应用，是这样的。每一个划分， B_i 其实就是一个分类，所以对于机器学习来说，就是通过给定的样本，对于不同特征的发生情况下，求 B_i 的过程。形式化描述，如下：

样本每条记录有n个特征值，分别记做 $X=\{x_1, x_2, \dots, x_n\}$ ，其中每一个特征值，有 k_i 种取值，那么， A_j 属于哪一个分类呢？

训练算法：

计算 $P(B_i|A_j)$ 的概率， i 从1到 m ，取最大值，就是 A_j 的分类。解释， $P(B_i|A)$ 的含义就是 A

预测算法：

给定一条记录，将该条记录中的所有特征值比对 A_j 中的那种组合，就是相应的分类。

具体举例来说。

样本有3个特征量， $X=\{x_1, x_2, x_3\}$ ， x_1 的取值有2种(w,t)， x_2 有3种(0,1,2)， x_3 有2种(p,q)， $k_1=2$ ， $k_2=3$ ， $k_3=2$ 。 A 的集合就是所有组合，一共有 $2 \times 3 \times 2 = 12$ 个元素。 $A=\{(w,0,p), (w,1,p), (w,0,q), \dots\}$ 。假设一共有3个分类。 $B=\{B_1, B_2, B_3\}$ 。

训练的样本数据如下：

x1 x2 x3 B

w 0 p b

w 1 q c

...

t 1 p a

算法伪代码：

训练算法：

1. 提取 x_1, x_2, x_3 所有特征值的值
2. 将 x_1, x_2, x_3 中分别取出一个元素，组成三元组，作为一个事件，事件的集合记做 A
3. 计算 A 中，每一个元素的概率 $P(A_j)$
 - 3.1 $P(A_j)$ 计算：找到所有 A_j 的行/总的样本数
4. 取出 B 中的标签种类，记做集合 B
5. 计算 B 中每一个标签种类的概率
 - 5.1 计算 B_i 的行数/总的样本数
6. 计算所有的条件概率概率 $P(A_j|B_i)$ ， i 取 $[1, \text{size}(B_i)]$ ， j 取 $[1, \text{size}(A_j)]$
7. 计算分类的条件概率 $\max(P(B_i|A_1))$ ， i 取 $[1, \text{size}(B_i)]$ ，当 $i=I_1$ 时， $\max(P(B_i|A_1))$ 就是分类
8. I_1, I_2, \dots, I_k 就是分类

预测算法：

1. 给定一个预测样本 x ，匹配在 A 中的位置，是 x_k ，那么分类就是 I_k 。

心得体会：全概率，条件概率问题绘制区域分布图，是一种最好的解决方案。

tips

$$P(A-AB) = P(A) - P(AB). P(A) = P(A \sim B \cup AB) = P(A \sim B) + P(AB) - 0 = P(A-AB) + P(AB)$$

第八讲 事件的独立性

$P(AB) = P(A)P(B)$ 称作 A 和 B 是相互独立的事件。

- (1) $P(AB) = P(A)P(B)$
- (2) $P(AC) = P(A)P(C)$
- (3) $P(BC) = P(B)P(C)$
- (4) $P(ABC) = P(A)P(B)P(C)$

其中：(1) (2) (3) 成立，称作 A B C 两两独立；而 (1) (2) (3) (4) 成立，都成立才称

考虑 A B 独立和 A B 相容。如果 A B 不相容，说明 $A \cap B = \emptyset$ ，显然 $P(AB) = 0$ ，而 $P(A), P(B) > 0$ ，所以不成立。也就是说如果 A B 不相容，那么 A B 一定不独立。为什么会有这样的结论？这样理解，假设 $S = A \cup B$ ，并且 $A \cap B = \emptyset$ ，则当 A 发生的时候，B 的概率就是 $\sim A$ 。所以二者是绝对关联的，所以不可能是独立的事件。所以只有 $A \cap B \neq \emptyset$ ，才能去讨论独立的问题。

在实际问题中，往往不是通过公式来计算是否独立的，而是根据实际情况的特点来确定是否独立的。然后使用概率的乘积来简化计算。所以对于现在来说，计算 $P(AB)$ 优先考虑是否是独立的再计算。这样的方式对并集的概率也提供了支撑。

$$P(A \cup B) = P(A) + P(B) - P(AB) = P(A) + P(B) - P(A)P(B)$$

小结

小结一

概率的定义，基本运算，条件概率，全概率，独立事件和不相容事件。重点是条件概率和全概率，而这两者的计算，依赖于画图，解决的问题是那些可以通过分步来分析解决的概率问题，因为每一次分步，都是在重新划分区域。而独立事件和不相容事件，是用来进行计算的简化。

第九讲 随机变量

第九讲 随机变量

很多时候得到的事件不是数量化的而是“枚举”类型的，例如：阴天，晴天，红球白球，红球红球……这些组合使得无法使用数的形式来研究。那么随机变量就是通过一个映射 X ，将事件 e 映射到一个实数区间内。

随机变量的定义：

随机变量实验的样本空间为 S ，若 $X=X(e)$ 为定义在 S 上的单值函数，则称 $X(e)$ 为随机变量，简所以 $X(e):S \rightarrow R$

$$A = \{e : X(e) \in I\} = \{X \in I\}$$

因为是单值函数，所以 $\{X=i\} \cap \{X=j\} = \emptyset$

离散型随机变量是 X 的取值是有限个或者可数个。例如正奇数集，从中任取一个数字，总是能

离散随机变量是以概率分布律形式，也就是表格展现。因为 X 是可数的，所以可以做成表格。

$X x_1 x_2 \dots x_k \dots$

$P p_1 p_2 \dots p_k \dots$

因为 p 是所有取值的概率所以： $p_k \geq 0$ ， $\sum p_k = 1$ 或者通项表达： $P(X=x_k) = p_k$

注意：因为 $\{X=x_i\} \cap \{X=x_j\} = \emptyset$ ，而 $\{X=x_1\} \cup \{X=x_2\} \cup \dots = I$ ，所以 x_1, x_2, \dots, x_n 是 n 个划分，可以使用条件概率和全概率来求解。

第十讲 离散型随机变量

第十讲 离散型随机变量

0-1分布

若X的概率分布律为：

$$X | 0 | 1 | \dots | \dots | \dots | P | 1-p | p |$$

其中 $0 < p < 1$ 则称X服从参数为p的0-1分布。记做 $X \sim B(1,p)$

这里，回想全概率区间划分，也就是这里有两个区间。同时，x的取值要么0要么1。

0-1分布，是指一个随机试验，A是一个随机事件， $P(A)=p$,若仅仅考虑A发生与否，就可以定义一个服从参数为p的0-1随机变量分布. $X=1$, 若A发生; $X=0$, 若A不发生。对于只有两个可能结果的试验，称作伯努利试验，也称作伯努利分布。

二项分布

$$P(X=k) = C(n, k)p^k(1-p)^{n-k}, \quad k=0, 1, \dots, n. \quad \text{其中 } n \geq 1, \quad 0 < p < 1, \text{ 就称X服}$$

解释:1 每次试验的结果只有两个，要么发生要么不发生 2 每次的试验结果是相互独立的 3 n是表示总共试验的次数 4 k是表示发生的次数 5 注意 $k=0, 1, \dots, n$ 是 n 个划分

总结：试验n次，其中每次发生的概率是p,那么发生k次的概率就是 $P(X=k)$.

泊松分布

$$P(X=k) = (\lambda^k e^{-\lambda}) / k!, \quad k = 0, 1, 2, \dots, \quad \lambda > 0$$

$$X \sim \pi(\lambda)$$

泊松分布目前的作用是计算二项分布的近似，当 $n > 10, p < 0.1$, 二者近似相等 $\lambda = np$.

几何分布

$$P(X=k) = p(1-p)^{k-1}$$

文字解释：在重复多次的伯努利实验中，实验进行到某种结果出现为止。也就是 第k次出现事

习题解析

一盒中有4个大小形状一致的球，其中3个为红球，1个为白球，采用放回抽样，第5次取到第2

解：注意，题的要求是第五次取到第二个白球，也就是说第5次取到的一定是白球，前面4次，前4次有一个白球的概率是二项分布， $P(A) = P(X=1) = C(4, 1) * (1/4) * (1-3/4)^3$. 第5次是

这个问题要特别注意和二项分布的区别。

第十一讲 分布函数

第十一讲 分布函数

$$F(x) = P(X \leq x)$$

$$P(X < x) = P(X \leq x) - P(X = x) = F(x) - F(x - 0)$$

对于分布函数的求解来说，要覆盖整个区间。

第十三讲 连续分布和指数分布

第13讲 连续分布和指数分布

着重理解指数分布，

$$f(x) = \lambda e^{-\lambda x}, \quad x > 0 \\ f(x) = 0, \quad x \leq 0$$

其中 $\lambda > 0$, 叫做指数分布, $X \sim E(\lambda)$

分布函数

$$F(x) = 1 - e^{-\lambda x}, \quad x > 0 \\ F(x) = 0, \quad x \leq 0$$

重要性质 无记忆性, $P(X > t_0 + t | X > t_0) = P(X > t)$

对于无记忆性的理解:

当一个时间发生了, 那么, 在此基础上在发生一个delta事件的概率是独立的。

例如:

交通事故发生的时间间隔服从指数分布, 那么, 过去的13个小时没有发生事故, 再经过两个小时

无论前面是否发生事故, 经过两个小时发生的事故的概率都是一样的。

$$P(X > 13+2 | X > 13) = P(X > 2) = 1 - F(2)$$

这里为什么不是 $P(X > 2 | X > 13)$?

从数学的角度看 $P(X > 2 | X > 13) = P(X > 2, X > 13) / P(X > 13) = P(X > 13) / P(X > 13) = 1$
显然是不对的, 因为X是随机变量, 也就是说在条件概率的情况下, X的起始点是一样的。

还有要注意是 $X > 13$ 而不是 $X = 13$, 这种语文意义的理解非常关键。

第十四讲 正态分布

第十四讲 正态分布

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, -\infty < x < +\infty$$

正态分布的概率密度函数是:
 $\sigma > 0, \mu \in \mathbb{R}$, 记做 $X \sim N(\mu, \sigma^2)$. 其中,

$$P(X \leq x) = F(x) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^x \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt$$

正态分布的概率计算,
 很遗憾上面的积分从数学的角度是算不出来的。计算的方法是使用计算机编程做近似计算。另外，常用的是通过标准正态分布来计算。

标准正态分布

$$\varphi(z) = \frac{1}{\sqrt{2}} e^{-\frac{z^2}{2}}$$

若 $Z \sim N(0, 1)$, 称 Z 服从标准正态分布。 Z 的概率密度函数:

标准正太分布可以通过查表获取。

有如下性质:

当 $X \sim N(\mu, \sigma^2)$ 时, $X - \mu / \sigma \sim N(0, 1)$

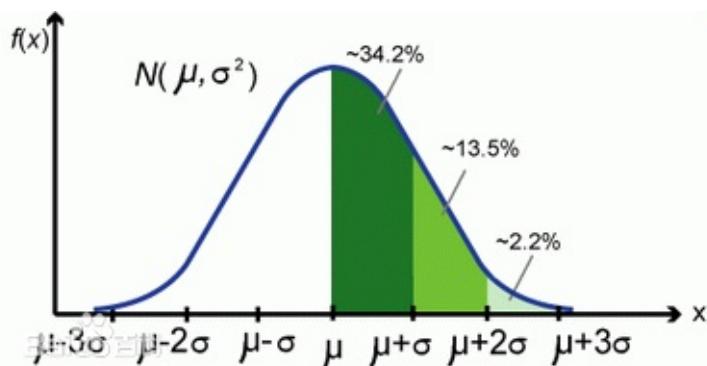
标准正态分布使用 这里就是NG的课程中, 特征量正规化的函数实现。通过上面的方法来做的normalize标准化。所以在计算概率的时候转化成标准正态分布。

概率分布小结

概率分布小结

在概率分布中，涉及到几个概念，随机变量 x ，概率密度 $f(x)$ ，分布函数 $F(x)$ 以及概率 $P(X < x)$ 。对于计算来说，一般知道 $f(x)$ 来计算 P 。所以就是 $f(x) \rightarrow F(x) \rightarrow P$ 这样的一个计算步骤。

现在考虑， $f(x)$ 的实际意义是什么？是否存在实际意义？我们说 x 服从标准正态分布，那么，这个概率密度函数 N 究竟代表什么意义呢？



看这个图这条概率密度的曲线究竟是什么实际意义？让我们来看实际例子：

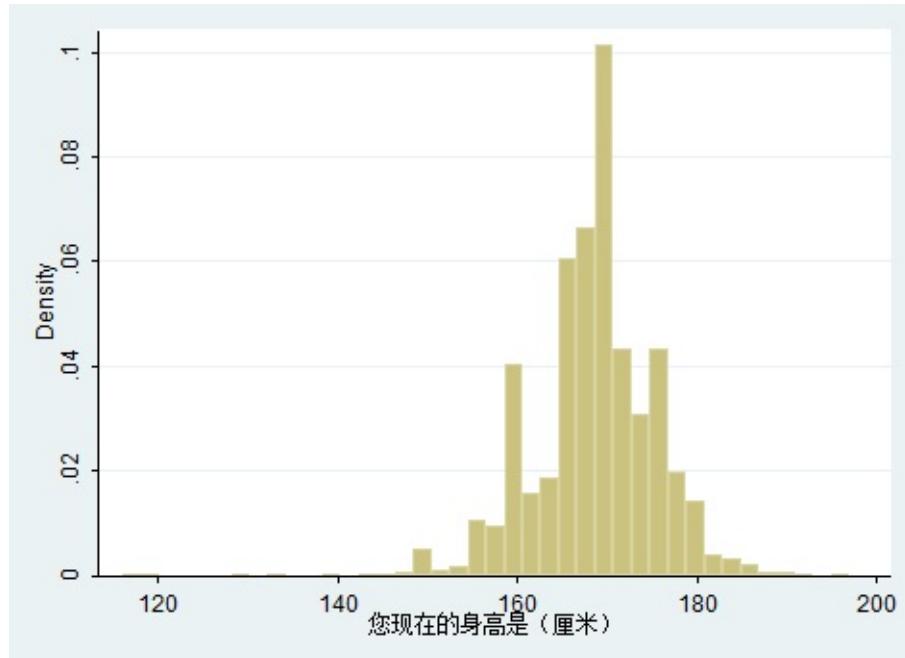
例子：身高分布

人的身高是服从正态分布的。那么，这样一个概率分布是如何得到的呢？统计10W人，将身高绘制成一张表，其中包括不同身高的人数。

身高 人数

1.1	1000
1.65	50000
1.8	3000

那么，y轴表示数量，x轴表示身高绘制函数图像。那么， $y=f(x)$ 就是概率密度函数。如下图：



下图中 $y \in [0,1]$ ，说明对 y 做了处理，这样影响的是 σ 参数。

这个理解要特别小心，特别小心的误区，就是通过离散的身高数据，计算出概率，能够拟合 $f(x)$ ，成为一个连续的函数分布。但是，不能通过 $f(x)$ 来反推 x 的人数或者概率。因为假设 $f(1) = 500$ ，那么 $f(1+\delta)$ 约等于 500，当 δ 足够小的时候，这样总人数就是 1000 人了，会超过总人数的。所以不能通过 $f(x)$ 来推导人数。原因就在于 $f(x)$ 本身是连续的。所以下面的结论很关键：

在现实生活中，通过离散的 x 取值，也就是 x_1, x_2, \dots, x_n ，并求得 $P(X=x_1), P(X=x_2) \dots F$ 如果使用 $f(x)$ 反推 P ，那么仅有 $P(X=x_1)=f(x_1), P(X=x_2), \dots P(X=x_n)$ ，将随机变量 X 看那么，如何认为 X 是连续型随机变量，通过 $f(x)$ 如何求任意概率呢？通过 $F(x)=P(X \leq x)=\int f$ 所以现实中对连续型随机变量的研究是从离散入手，绘制概率密度 $f(x)$ 曲线，再反求概率。而 使用这种思想来思考逻辑回归：

对于一条样本记录 $X=(x_1, x_2, \dots, x_n) \ Y=(0 \text{ 或 } 1)$ 来说，事实就是将 X 向量变成一个数字，现在假设 θ 已经求出了，因为符合高斯分布， $F(x)$ 概率函数可知 $(F(x)=\int f(x)dx)$ 。现在是将现在剩下的问题就是如何训练 θ ？同样使用 $F(x)$ 进行拟合。也就是有 $F(x)=\int f(x)dx$ 。那么那么对于 $F(x)=P(X \leq x)$ 和分类之间的实际意义是什么呢？思考分类的本质，就是给定一个

第十六讲 二元随机变量，离散型随机变量分布规律

第十六讲 二元随机变量，离散型随机变量分布规律

定义

假设 E 是一个随机试验，样本空间 $S=\{e\}$;假设 $X=X(e)$ 和 $Y=Y(e)$ 是定义在 S 上的随机变量，由他们构成的向量 (X, Y) 成为二元随机变量。

对于 $P(X=x_i, Y=y_j)$ 就是 $P(AB)$ 的问题。看到这里，看到这里是否明白一点，所谓的机器学习，其实就是n元随机变量的概率分布规律的研究。

性质：

1. $p_{ij} \geq 0$
2. $\sum p_{ij} = 1$
3. $P((X, Y) \in D) = \sum p_{ij}$

其中 $p_{ij} = P(X=x_i, Y=y_j)$

根据下面的例子，其实也明白事件A是 $X=x_i$ ，事件B是 $Y=y_j$.那么就有 $P(AB)=P(A)P(B|A)$ 。所以本质上是条件概率。当然，如果A，B是独立的，那么就有 $P(AB)=P(A)P(B)$.

第十七讲 二元离散型随机变量边际分布率和条件分布律

第十七讲 二元离散型随机变量边际分布率和条件分布律

X,Y	y1	y2	...	Yj	...	P(X=xi)
x1	p11	p12	...	p1j	...	p1*
x2	p21	p22	...	p2j	...	p2*
...	p3*
xi	pi1	pi2	...	pij	...	pi*
...
P(Y=yj)	p*1	p*2	...	p*j	...	1

边际分布率的含义是 $P(X=xi)$ 或者 $P(Y=yj)$ 的概率。这样的概率往往作为条件概率中的一部分来出现的。并用来计算条件分布率。 $P(X=x1 | Y=y1) = P(X=x1, Y=y1) / P(Y=y1)$

第十八讲 二元随机变量分布函数，条件分布函数

第十八讲 二元随机变量分布函数，条件分布函数

具体的求解步骤:1 联合分布率 2 条件分布率 3 条件分布函数。

第二十三讲 随机变量的独立性

第二十三讲 随机变量的独立性

随机变量的独立性可以用来进行分类计算？

二元概率分布小结

小结3

二元随机变量比对表

下面的对照表，理解了离散型概率和连续随机变量的对应关系。

二元离散型随机变量	二元连续型随机变量
(X, Y) 联合分布律: $P(X=x_i, Y=y_j)=p_{ij}$, $i, j=1, 2, \dots$	(X, Y) 联合概率密度 $f(x, y), (x, y) \in D$
X 的边际分布律 $P(X=x_i)=\sum p_{ij}=p_i^*$ $, i=1, 2, \dots$	X 的边缘概率密度 $f_X(x)=\int f(x, y) dy$
$X=x_i$ 时 Y 的条件分布律 $P(Y=y_j X=x_i) = p_{ij} / p_i^*$	$X=x$ 时， Y 的条件概率密度 $f_{Y X}(y x) = f(x, y) / f(x)$ $y \in D_x$
$F(x, y)=P(X \leq x, Y \leq y)=\sum p_{i...j}$ 表示的是在 x, y 范围内所有的概率相加	$F(x, y)=\iint f(x, y) dx dy$, 表示的是 (x, y) 在区域 D 的积分

一元随机变量比对表

根据二元随机变量比对表，也给出一个一元随机变量比对表。

序号	比对名称	一元离散型随机变量	一元连续型随机变量
1	分布律	$P(X=x_i)=p_i, i=1, 2, \dots$	x 概率密度 $f(x)$
2	$F(x)$	$F(x) = P(X < x) = \sum p_{i...j}$	$F(x) = P(X \leq x) = \int f(x) dx$

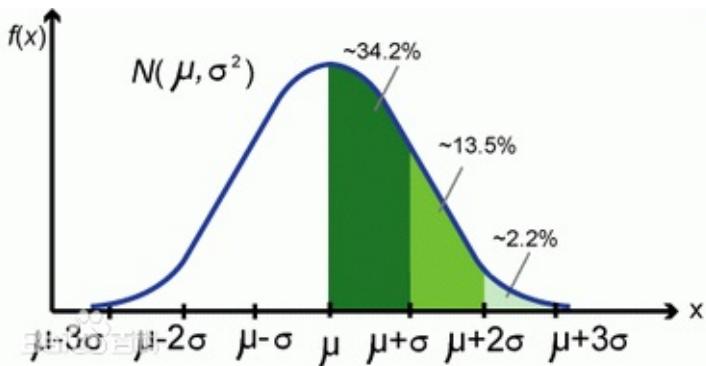
再次分析分类原理

分类过程包括:1 训练 2 预测。给定一个样本集，其中一条样本是 $x=(x_1, x_2, \dots, x_n)$ ，分类标签是 $y=0$ 或 1 。

原理:分类的就是假定 $\mathbf{x} = \Theta \mathbf{x}'$,符合某种概率分布，然后，当新出现一个 \mathbf{X}_{New} 的出现概率是多少进行分类。所以问题1：假定那种概率分布呢？答案是高斯分布。只有高斯分布才最能体现自然界的规律。也可以说，是当所有特征值以某种形态的均值是普遍符合高斯分布的。注意 $\Theta \mathbf{x}'$ 是所有特征值的均值。在高斯分布的假定下，我们先看看预测过程。

预测过程

观察正态分布的图像， \mathbf{x} 服从正态分布，也就是 $\mathbf{x} \sim N(\mu, \sigma^2)$



如果是分成两类，那么当XX出现在 $(-\infty, \mu]$ 的区间的时候，认为分类是0；当XX出现在 $[\mu, +\infty)$ ，认为分类是1。这很好理解，对于计算出来的一个xx，落在不同的位置取不同的分类。所以只要我们成功计算出 Θ ，那么，对于预测的 $xx_Predict = \Theta x'_{Predict}$ ，如果 $xx_Predict <= \mu$, 分类为0； $xx_Predict >= \mu$ 分类为1.

训练过程

训练的过程就是求解 Θ 的过程，使得 $XX = \Theta X'$ 符合正态分布。这里有两种做法，方法一 拟合概率密度函数；方法二 拟合概率分布函数。

拟合概率密度函数

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, -\infty < x < +\infty$$

对于一条样本X来说，当y=0的时候， $f(xx)$ 可以计算出来，可问题是样本中无法得出 $f(xx)$ 是多少，也就无法进行拟合。也就是说y=0对于这个标签来说表达的意思究竟是什么？是概率密度的函数值吗？假设是概率密度的函数值，那么假设 $f(xx)$ 的实际值是0.4，在y=0.4的情况下，因为是对称的，所以存在两个XX，所以无法使用这一点来进行拟合。那么我们只能走另外一条路，概率分布函数。

拟合概率密度分布函数

现在看看y=0或者1代表究竟是否是概率函数。对于分类来说，就是将x的整个取值区间划分成分类个区域，现在因为是分成两个类别，所以是 $(-\infty, \mu]$ 和 $[\mu, +\infty)$ 。对于任意一个样本y=0来说，有 $xx \leq \mu$ ，那么，来寻求xx的关系。对于随机变量X来说， $P(X \leq xx)$ 有 $P(X \leq xx \leq \mu)$ 表示的就是分类y=0，所以 $F(xx) = P(X \leq xx) \leq P(X \leq \mu) = 0.5$ 是与y=0对应的。也就是说 $F(xx) < 0.5$ 类别是0，那么，y=0，可以用作样本的F取值，进行拟合。也就是求 $\sum(F(xx)-y)$ 的最小值问题。

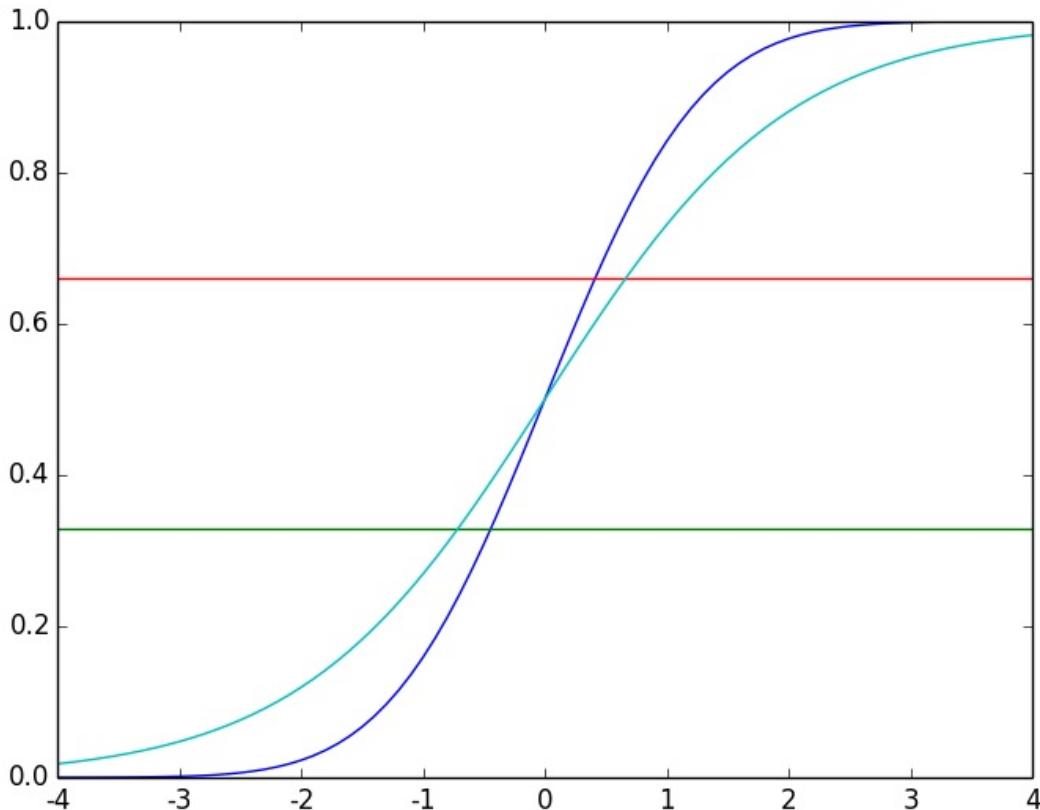
实际问题举例

身高问题，分类是高和矮，那么当 $x < \mu$ 的时候认为是矮；当 $x > \mu$ 的时候被认为是高。所以拟合X符合高斯分布即可。对于给定的身高x，拟合的是概率分布函数，原因是概率密度无法拟合。根据前面的比对表格，也明白概率就是概率密度函数。

对于多个分类问题

在NG的课程上解释的是0, 1分类问题，那么对于0, 1, 2, 更多分类问题如何处理呢？其实本质上是一样的，假设分成3个类别，那么就是将x的取值范围进行3等分。那么，如何3等分？直接将x的区间平均划分成3等分吗？显然不行。因为这个所谓的3等分是对样本数量的3等分，假设有3000条样本，合理的划分是每个分类中有1000条。现在因为是高斯分布所以会

在靠近 μ 的附近 x 会集中的分布，要比两边的数量多。所以均分的概念，如果想象成离散型的是： x_1 的数量+...+ x_k 的数量 = x_{k+1} 的数量+...+ x_l 的数量 = x_l 的数量+...+ x_n 的数量。也就是说分成了三个区间 $[x_1, x_k], [x_k, x_l], [x_l, x_n]$ ，要求有 $\text{count}(x_1)+\dots+\text{count}(x_k) = \text{count}(x_{k+1})+\dots+\text{count}(x_l) = \text{count}(x_l) + \dots + \text{count}(x_n)$ 。全部除以总数量S，就变成了额 $P(x_1)+\dots+P(x_k) = P(x_{k+1}) + \dots + P(x_l) = P(x_l) + \dots + P(x_n)$ 也就是 $P(x_1 \leq X < x_k) = P(x_k \leq X < x_l) = P(x_l \leq X \leq x_n) = 1/3$ 。对于连续型随机变量就是 $F(x_k) = F(x_l) - F(x_k) = F(x_n) - F(x_l)$ ，也就是 $\int f(x)dx|(-\infty, x_k) = \int f(x)dx|(x_l, x_k) = \int f(x)dx|(x_k, +\infty)$ 。所以就是在 $F(x)=1/3, 2/3$ ，这两个位置绘制两条直线将 $F(x)$ 进行了分割。这种方式，如果考察 $F(x)$ 的图像曲线就更加容易看清楚。



对于不同的标签来说， $y=0$ ，就是让所有的 $F(x)$ 为 0 进行逼近； $y=1$ ， $F(x)=0.5$ 进行拟合； $y=2$ ， $F(x)=1$ 进行拟合。这也就是逻辑回归的拟合方式。

分布表

第二十九讲 数学期望的性质

这里给出了计算期望的公式，尤其是: $E(ax_1+bx_2+c)=aE(x_1)+bE(x_2)+cE(c)$ 和 $E(\prod X_i) = \prod E(X_i)$, 其中 X_i 相互独立. 令 $Z=ax_1+bx_2+c$ 或者 $Z=x_1 \cdot x_2 \cdot x_3 \dots x_n$ 都是可以通过对一个随机事件Z的求期望问题，变成将随机变量分解来做。典型有:求和的期望，令 $x_i=0$ 或 1 , 求和的期望就是 $Z=x_1+\dots+x_n$.

概率全息知识小结

小结4-概率全息表

概率分布

对照表,各种分布,概率密度,分布函数,期望,以及相关的所有公式的全方位的总结。

序号	离散型随机变量分布	记号	公式
1	0-1分布	--	$X = \begin{cases} 1, & \text{if } A \text{ happen} \\ 0, & \text{if } A \text{ not happen, } (\bar{A} \text{ happen}) \end{cases}$
2	二项分布	$X \sim B(n, p)$	$P(X = k) = C_n^k p^k (1 - p)^{n-k}, k = 0, 1, \dots, n$
3	泊松分布	$x \sim \pi(\lambda)$	$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}, k = 0, 1, 2, \dots$
4	几何分布	$X \sim Geom(p)$	$P(X = k) = p(1 - p)^{k-1}, k = 1, 2, 3, \dots$

分布函数

离散型分布函数:

- 1 $F(x) = P(X \leq x)$
- 2 $P(a < X \leq b) = P(X \leq b) - P(X \leq a) = F(b) - F(a)$
- 3 $P(a < X < b) = P(a < X \leq b - \theta) = F(b) - F(a)$
- 4 $P(X=b) = P(X \leq b) - P(X < b) = F(b) - F(b-\theta)$
- 5 $P(a < X < b) = P(a < X \leq b) - P(X=b)$

一般地, 离散型随机变量的分布函数为阶梯函数, 假设X的分布律为 $P(X=x_k)=p_k, k=1,$

2,..., X的分布函数为 $F(x) = \sum_{x_k \leq x} p_k$, $F(x)$ 在 $x=x_k(k=1,2,\dots)$ 处有跳跃, 跳跃值为 $p_k=P(X=x_k)$.

$$F(x) = \int_{-\infty}^x f(t) dt$$

连续型随机变量X的分布函数 $F(x)$: 其中 $f(x)$ 称为X的概率密度函数。

概率密度的性质:

1. $f(x) \geq 0$
2. $\int_{-\infty}^{+\infty} f(x) dx = 1; \because F(+\infty) = 1$

$$P(X \in D) = \int_D f(x) dx \rightarrow$$

3. $P(x_1 < X \leq x_2) = \int_{x_1}^{x_2} f(t) dt = F(x_2) - F(x_1)$

4. $F'(x) = f(x)$

概率密度典型分布函数:

序号	随机变量	记号	连续型 量概率 密度	公式
1	均匀分布	$X \sim U(a, b)$	\square	
2	指数分布	$X \sim E(\lambda)$		$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x > 0 \\ 0, & x \leq 0 \end{cases}$ $F(x) = \begin{cases} 1 - e^{-\lambda x}, & x > 0 \\ 0, & x \leq 0 \end{cases}$
3	正态分布	$X \sim N(\mu, \sigma^2)$		$X \sim N(\mu, \sigma^2) \Rightarrow f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, -\infty < x < +\infty$ $Z \sim N(0, 1) \Rightarrow \varphi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$ $\Phi(z) = \int_{-\infty}^z \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt$ $\Phi(-z_0) = 1 - \Phi(z_0)$

Y=g(x)的概率密度求解定理:

设随机变量 $X \sim f_X(x), -\infty < x < +\infty$, $Y = g(X)$, $g'(x) > 0$ 或者 $g'(x) < 0$, 则 Y 具有概率密度为:

$$f_Y(y) = \begin{cases} f_X(h(y)) * |h'(y)|, & \alpha < y < \beta \\ 0, & \text{otherwise} \end{cases}$$

$$h(y) = x \Leftrightarrow y = g(x), h = g^{-1}$$

$$\text{if } X \sim N(\mu, \sigma^2), Y = aX + b \Rightarrow Y \sim N(a\mu + b, a^2\sigma^2)$$

二元随机变量

离散型二元随机变量:

$$1 p_{ij} \geq 0$$

$$2 \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} p_{ij} = 1$$

$$3 P((X, Y) \in D) = \sum_{(x_i, y_j) \in D} p_{ij}$$

$$4 p_{ij} = P(X = x_i, Y = y_j)$$

离散型二元随机变量边际分布率:

$$P(X = x_i) = P(X = x_i, \bigcup_{j=1}^{\infty} (Y = y_j)) = \sum_{j=1}^{\infty} p_{ij} = p_{i \cdot}$$

$$P(Y = y_j) = P(Y = y_j, \bigcup_{i=1}^{\infty} (X = x_i)) = \sum_{i=1}^{\infty} p_{ij} = p_{\cdot j}$$

下面的表格很清楚的表达了二元随机变量的概率分布。

$X \setminus Y$	y_1	y_2	\dots	y_j	\dots	$P(X = x_i)$
x_1	p_{11}	p_{12}	\dots	p_{1j}	\dots	$p_{1 \cdot}$
x_2	p_{21}	p_{22}	\dots	p_{2j}	\dots	$p_{2 \cdot}$
x_i	p_{i1}	p_{i2}	\dots	p_{ij}	\dots	$p_{i \cdot}$
$P(Y = y_j)$	$p_{\cdot 1}$	$p_{\cdot 2}$	\dots	$p_{\cdot j}$	\dots	1

离散型二元随机变量条件分布律:

$$P(X = x_i | Y = y_j) = \frac{P(X = x_i, Y = y_j)}{P(Y = y_j)} = \frac{p_{ij}}{p_{\cdot j}}$$

连续型二元随机变量:

$$\text{联合分布函数: } F(x, y) = P\{(X \leq x) \cap (Y \leq y)\} = P(X \leq x, Y \leq y)$$

将 $X \leq x, Y \leq y$ 看成是 (x, y) 的区域，就是这个区域内的所有概率相加.

$$F_x(x) = F(x, +\infty) = \lim_{y \rightarrow \infty} F(x, y)$$

$$\text{边际分布函数. } F_y(y) = F(+\infty, y) = \lim_{x \rightarrow \infty} F(x, y)$$

离散型条件分布函数:

$$F_{x|y}(x | y) = P(X \leq x | Y = y) = \frac{P(X \leq x, Y = y)}{P(Y = y)}$$

$$\begin{aligned}
 F_{X|Y}(x | y) &= \lim_{\varepsilon \rightarrow 0^+} P(X \leq x | y < Y \leq y + \varepsilon) \\
 &= \lim_{\varepsilon \rightarrow 0^+} \frac{P(X \leq x, y < Y \leq y + \varepsilon)}{P(y < Y \leq y + \varepsilon)} \\
 \text{连续型条件分布函数: } &= P(X \leq x | Y = y)
 \end{aligned}$$

二元连续型随机变量

$$F(x,y) = P(X \leq x, Y \leq y) = \int_{-\infty}^y \int_{-\infty}^x f(u,v) du dv$$

性质如下：

1. $f(x,y) \geq 0$
2. $\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x,y) dx dy = 1$
3. 设D是xoy平面的区域，点(X,Y)落在D内的概率为：

$$P((X,Y) \in D) = \iint_D f(x,y) dx dy$$

$$\frac{\partial^2 F(x,y)}{\partial x \partial y} = f(x,y)$$
4. 在f(x,y)的连续点(x,y)，有

边际分布函数：

$$\begin{aligned}
 F_X(x) &= F(x, +\infty) = \lim_{y \rightarrow \infty} F(x,y) \\
 F_Y(y) &= F(+\infty, y) = \lim_{x \rightarrow \infty} F(x,y)
 \end{aligned}$$

边际概率密度：

$$\begin{aligned}
 f_X(x) &= \int_{-\infty}^{+\infty} f(x,y) dy \\
 f_Y(y) &= \int_{-\infty}^{+\infty} f(x,y) dx
 \end{aligned}$$

条件概率密度：

对于固定的y(注意是固定的y，也就是y等于某个常数), $f_Y(y)>0$,并且 $f_Y(y)$ 连续，则在 $Y=y$ 的条件下,X的条件概率是：

$$f_{X|Y}(x | y) = \frac{f(x,y)}{f_Y(y)}$$

二元均匀分布

二元随机变量(X,Y)的概率密度在一个游街的区域D内是常数，其他地方为0，则称(X,Y)在D上服从均匀分布。设 $f(x,y)=1/A$ ($x,y \in D$) 或者 $f(x,y)=0$, 其他.

二元正态分布

设二元随机变量(X,Y)的概率密度为:

设二元随机变量(X, Y)的概率密度为:

$$f(x, y) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \times \\ \exp \left\{ \frac{-1}{2(1-\rho^2)} \left[\frac{(x-\mu_1)^2}{\sigma_1^2} - 2\rho \frac{(x-\mu_1)(y-\mu_2)}{\sigma_1\sigma_2} + \frac{(y-\mu_2)^2}{\sigma_2^2} \right] \right\} \\ (-\infty < x < +\infty, -\infty < y < +\infty)$$

其中 $\mu_1, \mu_2, \sigma_1 > 0, \sigma_2 > 0, -1 < \rho < 1$ 都是常数, 称(X, Y)

为服从参数为 $\mu_1, \mu_2, \sigma_1, \sigma_2, \rho$ 的二元正态分布。

记为: $(X, Y) \sim N(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \rho)$

5

边际概率密度:

$$\therefore f_X(x) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} \quad -\infty < x < +\infty$$

$$\text{同理 } f_Y(y) = \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(y-\mu_2)^2}{2\sigma_2^2}}, \quad -\infty < y < +\infty$$

即二元正态分布的两个边际分布都是
一元正态分布, 并且都不依赖于参数 ρ .

条件概率密度:

设二元随机变量 $(X, Y) \sim N(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \rho)$ ；
求条件概率密度 $f_{Y|X}(y|x)$.

$$f_{Y|X}(y|x) = \frac{f(x, y)}{f_X(x)}$$

$$= \frac{1}{\sqrt{2\pi}\sigma_2\sqrt{1-\rho^2}} \exp\left\{ \frac{-1}{2(1-\rho^2)\sigma_2^2} \left[y - (\mu_2 + \rho \frac{\sigma_2}{\sigma_1} (x - \mu_1)) \right]^2 \right\}$$

即在 $X=x$ 条件下， Y 的条件分布仍是正态分布，

$$Y|X=x \sim N(\mu_2 + \rho \frac{\sigma_2}{\sigma_1} (x - \mu_1), (1 - \rho^2)\sigma_2^2).$$

11

二元随机变量的独立性

若 $P(X \leq x, Y \leq y) = P(X \leq x)P(Y \leq y)$ 即 $F(x, y) = F_X(x)F_Y(y)$ ，则称 X, Y 相互独立。

离散型独立判断：对一切 i, j 都有： $P(X=x_i, Y=y_j) = P(X=x_i)P(Y=y_j)$

连续型独立判断：对于平面的点 (x, y) ，处处有 $f(x, y) = f_X(x)f_Y(y)$

二元正态随机变量 (X, Y) ，独立的充要条件是 $\rho=0$.

多元随机变量独立定理：

设 (X_1, X_2, \dots, X_m) 与 (Y_1, Y_2, \dots, Y_n) 相互独立，则有

1. X_i 与 Y_j 相互独立
2. $h(x_1, x_2, \dots, x_m)$ 和 $g(y_1, y_2, \dots, y_n)$ 是连续函数，则 h 与 g 相互独立。也就是说线性相关的也相互独立。

二元随机变量的函数分布

➤ 设二元连续型随机变量 (X, Y) 具有概率密度 $f(x, y)$,
 Z是 X, Y 的函数, $Z = g(X, Y)$.

问题 Z的概率分布或密度函数是什么?

方法 先求Z的分布函数再求导得到密度函数.

$$F_Z(z) = P(Z \leq z) = P(g(X, Y) \leq z) = \iint_{g(x,y) \leq z} f(x, y) dx dy$$

$$f_Z(z) = F'_Z(z)$$

$Z=X+Y$ 的概率密度为:

★ 卷积公式:

将 X 和 Y 相互独立时, $Z = X + Y$ 的密度函数公式称为**卷积公式**

$$\text{即 } f_Z(z) = \int_{-\infty}^{+\infty} f_X(z-y) f_Y(y) dy = \int_{-\infty}^{+\infty} f_X(x) f_Y(z-x) dx$$

正态分布的分布:

推广结论: 设 X, Y 相互独立, $X \sim N(\mu_1, \sigma_1^2)$, $Y \sim N(\mu_2, \sigma_2^2)$, 则

$$Z = X + Y \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

更一般的结论:

n 个独立的正态变量的线性组合仍服从正态分布, 即:

若 $X_i \sim N(\mu_i, \sigma_i^2)$ $i=1, 2, \dots, n$, 且它们相互独立, 则其线性组合:

$$c_0 + c_1 X_1 + c_2 X_2 + \dots + c_n X_n \sim N(\mu, \sigma^2)$$

其中 c_1, c_2, \dots, c_n 是不全为0的常数, 两个参数(可由期望及方差得到)为:

$$\mu = c_0 + c_1 \mu_1 + \dots + c_n \mu_n, \quad \sigma^2 = c_1^2 \sigma_1^2 + c_2^2 \sigma_2^2 + \dots + c_n^2 \sigma_n^2$$

离散型随机变量的独立和分布

➤ 离散变量的独立和分布

1. X_1, X_2, \dots, X_n 独立且均服从 $B(1, p)$, 则 $X_1 + X_2 + \dots + X_n \sim B(n, p)$
2. $X \sim B(n_1, p), Y \sim B(n_2, p)$, 两者独立, 则 $X + Y \sim B(n_1 + n_2, p)$
3. $X \sim \pi(\lambda_1), Y \sim \pi(\lambda_2)$, 两者独立, 则 $X + Y \sim \pi(\lambda_1 + \lambda_2)$

$M = \max(X, Y)$ 和 $N = \min(X, Y)$ 的分布

➤ $M = \max(X, Y)$ 和 $N = \min(X, Y)$ 的分布

设 X, Y 是两个相互独立的随机变量, 它们的分布函数分别为

$F_X(x)$ 和 $F_Y(y)$, 求 M, N 的分布函数 $F_{\max}(z)$ 和 $F_{\min}(z)$ 。

$M = \max(X, Y)$ 的分布函数为:

$$\begin{aligned} F_{\max}(z) &= P(M \leq z) \\ &= P(X \leq z, Y \leq z) \\ &\stackrel{\text{独立}}{=} P(X \leq z)P(Y \leq z) \end{aligned}$$

$$F_{\max}(z) = F_X(z)F_Y(z)$$

$N = \min(X, Y)$ 的分布函数为:

$$\begin{aligned} F_{\min}(z) &= P(N \leq z) = 1 - P(N > z) \\ &= 1 - P(X > z, Y > z) \\ &\stackrel{\text{独立}}{=} 1 - P(X > z)P(Y > z) \end{aligned}$$

$$F_{\min}(z) = 1 - (1 - F_X(z))(1 - F_Y(z))$$

n 个相互独立的随机变量的情况

设 X_1, X_2, \dots, X_n 是 n 个相互独立的随机变量, 它们的分布

函数分别为: $F_{X_i}(x_i)$, $i = 1, 2, \dots, n$

$M = \max(X_1, \dots, X_n)$ 及 $N = \min(X_1, \dots, X_n)$ 的分布函数为:

$$F_{\max}(z) = F_{X_1}(z)F_{X_2}(z) \cdots F_{X_n}(z)$$

$$F_{\min}(z) = 1 - [1 - F_{X_1}(z)][1 - F_{X_2}(z)] \cdots [1 - F_{X_n}(z)]$$

$$F_{\max}(z) = (F(z))^n$$

$$F_{\min}(z) = 1 - [1 - F(z)]^n$$

共同的分
布函数 $F(z)$

$$f_{\max}(z) = n[F(z)]^{n-1} f(z)$$

$$f_{\min}(z) = n[1 - F(z)]^{n-1} f(z)$$

X_1, X_2, \dots, X_n 是连续的独立同分布变量

共同的密度
函数 $f(z)$

3

数学期望

$$E(X) = \sum_{k=1}^{+\infty} x_k p_k$$

$$E(X) = \int_{-\infty}^{+\infty} xf(x) dx$$

分布名称	期望
泊松分布 $X \sim \pi(\lambda)$	$E(X) = \lambda$
正态分布 $X \sim N(\mu, \sigma^2)$	$E(X) = \mu$
指数分布 $X \sim E(\lambda)$	$E(X) = 1/\lambda$
二项分布 $X \sim B(n, p)$	$E(X) = np$
几何分布 $X \sim G(p)$	$E(X) = 1/p$
均匀分布 $X \sim U(a, b)$	$E(X) = (a+b)/2$

随机变量函数的期望：

定理1：设 Y 是随机变量 X 的函数： $Y = g(X)$ ，
 X 是离散型随机变量，它的分布律为：

$$P(X = x_k) = p_k, \quad k = 1, 2, \dots,$$

若 $\sum_{k=1}^{\infty} g(x_k) p_k$ 绝对收敛，则

$$E(Y) = E[g(X)] = \sum_{k=1}^{\infty} g(x_k) p_k;$$

定理1（续）：

设 Y 是随机变量 X 的函数： $Y = g(X)$ ，
 X 是连续型随机变量，它的概率密度函数为 $f(x)$ ，
 若 $\int_{-\infty}^{+\infty} g(x) f(x) dx$ 绝对收敛，则

$$E(Y) = E(g(X)) = \int_{-\infty}^{+\infty} g(x) f(x) dx$$

定理2：

设 Z 是随机变量 X, Y 的函数： $Z = h(X, Y)$ ，
若二元离散型随机变量 (X, Y) 的分布律为：
 $P(X = x_i, Y = y_j) = p_{ij}$, $i, j = 1, 2, \dots$, 则有

$$E(Z) = E[h(X, Y)] = \sum_{i=1}^{+\infty} \sum_{j=1}^{+\infty} h(x_i, y_j) p_{ij};$$

定理2（续）：

设 Z 是随机变量 X, Y 的函数： $Z = h(X, Y)$ ，
若二元连续型随机变量 (X, Y) 的概率密度
为 $f(x, y)$, 则有

$$E(Z) = E(h(X, Y)) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h(x, y) f(x, y) dx dy.$$

特别地, $E(X) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x f(x, y) dx dy$

$$E(Y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} y f(x, y) dx dy.$$

9

数学期望的性质

1. $E(c_0 + \sum_{i=1}^n c_i X_i) = c_0 + \sum_{i=1}^n c_i E(X_i)$

2. $E(\prod_{i=1}^n X_i) = \prod_{i=1}^n E(X_i)$

本题是将 X 分解成数个随机变量之和, 然后利用随机变量和的数学期望等于随机变量数学期望之和来求数学期望, 这种处理方法具有一定的普遍意义。

方差

$$D(X) = \text{Var}(X) = E\{[X - E(X)]^2\}.$$

将 $\sqrt{D(X)}$ 记为 $\sigma(X)$, 称为 X 的标准差或均方差.

$$D(X) = \sum_{i=1}^{\infty} [x_i - E(X)]^2 p_i$$

$$D(X) = \int_{-\infty}^{+\infty} [x - E(X)]^2 f(x) dx$$

$$D(X) = E(X^2) - [E(X)]^2$$

分布名称	期望	方差
泊松分布 $X \sim \pi(\lambda)$	$E(X)=\lambda$	$D(X)=\lambda$
正态分布 $X \sim N(\mu, \sigma^2)$	$E(X) = \mu$	$D(X)=\sigma^2$
指数分布 $X \sim E(\lambda)$	$E(X) = 1/\lambda$	$D(X)=1/\lambda^2$
二项分布 $X \sim B(n,p)$	$E(X) = np$	$D(x)=np(1-p)$
几何分布 $X \sim G(p)$	$E(X) = 1/p$	
均匀分布 $X \sim U(a,b)$	$E(X) = (a+b)/2$	$D(X)=(b-a)^2/12$

性质:

- 设 c 是常数, 则 $D(c) = 0$.
- 设 X 是随机变量, c 是常数, 则有 $D(cX) = c^2 D(X)$.

(特例: $D(-X) = D(X)$.)

- 设 X, Y 是两个随机变量, 则有

$$D(X + Y) = D(X) + D(Y) + 2 \cdot \text{tail},$$

其中, $\text{tail} = E\{[X - E(X)][Y - E(Y)]\}$.

特别, 若 X, Y 相互独立, 则有 $D(X + Y) = D(X) + D(Y)$.

综合上述三项，设 X, Y 相互独立， a, b, c 是常数，则 $D(aX + bY + c) = a^2 D(X) + b^2 D(Y)$.

(特例： $D(X + c) = D(X)$.)

推广到任意有限个独立随机变量线性组合的情况

$$D(c_0 + \sum_{i=1}^n c_i X_i) = \sum_{i=1}^n c_i^2 D(X_i),$$

其中 $X_i, i = 1, 2, \dots, n$, 相互独立.

4. $D(X) = 0 \Leftrightarrow P(X = c) = 1$ 且 $c = E(X)$.

正态分布的期望和方差：

n 个独立的正态随机变量的线性组合仍服从正态分布.

若 $X_i \sim N(\mu_i, \sigma_i^2)$, $i = 1, 2, \dots, n$, 且相互独立, 则它们的线性组合:

$$\begin{aligned} & c_0 + c_1 X_1 + c_2 X_2 + \dots + c_n X_n \\ & \sim N(c_0 + c_1 \mu_1 + \dots + c_n \mu_n, c_1^2 \sigma_1^2 + c_2^2 \sigma_2^2 + \dots + c_n^2 \sigma_n^2), \end{aligned}$$

其中 c_1, c_2, \dots, c_n 是不全为0的常数.

$E(2X - 3Y)$

随机变量标准化的过程:

例4: 设随机变量 X 具有数学期望 $E(X) = \mu$, 方差 $D(X) = \sigma^2 \neq 0$,

记 $X^* = \frac{X - \mu}{\sigma}$, 称 X^* 为 X 的标准化变量.

证明: $E(X^*) = 0$, $D(X^*) = 1$.

特别注意：标准化的过程是将随机变量的量纲消除了。并且将均值变为0，方差变为1.

协方差

定义：数值 $E\{[X - E(X)][Y - E(Y)]\}$
为随机变量 X 与 Y 的协方差，记为 $Cov(X, Y)$ ，即

$$Cov(X, Y) = E\{[X - E(X)][Y - E(Y)]\}.$$

此时 $D(X+Y) = D(X) + D(Y) + 2Cov(X, Y)$

协方差 $Cov(X, Y)$ 反映了随机变量 X 与 Y 的线性相关性：

- 当 $Cov(X, Y) > 0$ 时，称 X 与 Y 正相关；
- 当 $Cov(X, Y) < 0$ 时，称 X 与 Y 负相关；
- 当 $Cov(X, Y) = 0$ 时，称 X 与 Y 不相关。

计算公式： $Cov(X, Y) = E(XY) - E(X)E(Y)$

性质：

■ 协方差的性质：

1. $Cov(X, Y) = Cov(Y, X);$
2. $Cov(X, X) = D(X);$
3. $Cov(aX, bY) = ab \cdot Cov(X, Y)$, 其中 a, b 为两个实数；
4. $Cov(X_1 + X_2, Y) = Cov(X_1, Y) + Cov(X_2, Y).$

在计算方差的时候优先考虑性质，而不是公式。

相关系数：

定义：数值

$$\rho_{XY} = \frac{Cov(X, Y)}{\sqrt{D(X)D(Y)}}$$

称为随机变量 X 与 Y 的相关系数，是没有量纲的。

若记标准化变量 $X^* = \frac{X - E(X)}{\sqrt{D(X)}}, Y^* = \frac{Y - E(Y)}{\sqrt{D(Y)}}$,

则 $\rho_{XY} = Cov(X^*, Y^*)$.

性质：

1. $|\rho_{XY}| \leq 1$;
2. $|\rho_{XY}| = 1 \Leftrightarrow$ 存在常数 a, b , 使 $P(Y = a + bX) = 1$

特别的, $\rho_{XY} = 1$ 时, $b > 0$; $\rho_{XY} = -1$ 时, $b < 0$.

相关系数是一个用来表征两个随机变量之间线性关系密切程度的特征数, 有时也称为“线性相关系数”.

当 $|\rho_{XY}|$ 较大时, 表明 X 与 Y 的线性关系程度较好;

当 $|\rho_{XY}|$ 较小时, 表明 X 与 Y 的线性关系程度较差.

特别地,

当 $|\rho_{XY}| = 1$ 时, 表明 X 与 Y 之间以概率 1 存在线性关系;

当 $\rho_{XY} = 0$ 时, 表明 X 与 Y 之间没有线性关系, 称两个变量不相关.

ρ 的理解很关键, 表示两个随机变量的相关性。如果二者线性相关, 那么使用 1 个随机变量就可以了, 没有必要使用两个随机变量。这一点是否可以用来进行对机器学习中特征量的维度进行化简呢?

第三十五讲 依概率收敛和切比雪夫不等式

第三十五讲 依概率收敛和切比雪夫不等式

依概率收敛

依概率收敛和切比雪夫不等式是为大数定律提供了理论基础。而大数定律告诉我们，通过大量的实验，可以认为符合概率。

这个问题告诉我们一个事实，我们常识认为的，例如100个中30个是好的，那么合格概率是30%，会因为抽样太少而用处不大的问题。

辛钦大数定律告诉我们，不用关注方差，只需要独立同分布，并且数学期望存在即可。

中心极限定理

设随机变量X的概率密度为 $f(x)=2x, 0 \leq x \leq 1/3$ 的近似值为：

计算Y的概率 $P(Y) = \int f(x)dx (0 < x < 1/3) = 1/9$

那么在162次，Y是符合二项分布的，就像扔硬币一样。这个地方的理解不容易
 $Y \sim B(162, 1/9)$

$P(Y > 22)$ Y近似等于 $N(np, np(1-p)) = N(18, 16)$

$P(Y > 22) = 1 - P(Y \leq 22) = 1 - N(Y-18/4 < (22-18)/4) = 1 - N(-1) = N(-1)$