第一章-机器学习概论

引言

机器学习($Machine\ Learning$)这一学科诞生于人工智能的中，它是计算机的一种新的能力，不同于通常的任务，例如编写程序实现一个文件管理系统，或许你需要分层的分模块的去实现它，但它总还是可以使用确定的程序逻辑去表达的，机器学习面对的任务通常是需要收集分析大量来自生活（自然、社会）中的数据，换句话说，机器学习就是用数据编程。例如给一张图片，人依据经验能够很快判断出图中是否有一只猫，那么我们能否按照通常的业务逻辑处理思想解决它呢？例如我们把图像的每一个像素点的信息（X、Y坐标、颜色的RGB表示）作为程序输入，这个算法要怎么设计以得到Y或者N这样肯定的判断呢？例如分块看像素平均值？最大值？等等等等，这些方法似乎都不太可行。于是不妨以结果为导向，给出许多许多的照片，这些照片中有的有猫，有的则没有，给出这些已知的结果，让机器去学习它们，从而在面对新的照片时能够一定程度上给出它的答案

机器学习在生活中的应用

- 数据挖掘($Database\ mining$)
  例如: 网络点击数据（推荐算法、用户个性化推荐），医疗记录（汇编成医疗大数据，辅助诊断治疗），生物学（基因编码、相似基因片段寻找等），工程应用
- 一些无法编程的应用
  例如，自动驾驶（小汽车、无人机）、手写体识别（自动信箱投递）、自然语言处理(NLP)、计算机视觉(CV)
- ... ...

什么是机器学习

- 它是一门致力于研究如何通过计算的手段，利用经验（数据）来提升自身的性能的学科, 研究的主要内容是从数据中产生 `模型(model)` 的算法，也即 `学习算法(Learning Algorithm)`
- 机器学习并没有一个严谨的定义，一个可能地说法是
  - A computer program is said to *learn from* experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.
  - 例如假设你的邮件系统有过滤垃圾邮件的功能，那么识别一封邮件是否为垃圾邮件就是任务T，经验E就是哪些被你归为垃圾邮件，度量P可以是正确分类的邮件数量

术语表述

- 给一个例子，判断 breast cancer 是 malignant 或者 benign, 假定影响肿瘤是否为良性的依据是肿瘤的大小(Tumor size), 并且有一系列已知的数据例
  如 `[{size:3, condition: benign}, {size:7, condition: malignant}, ... ...]` 那么据此进行学习，其目标是区分肿瘤是良性(0)或者恶性(1)，这样的任务称为 `分类学习(Classification)`，也即预测值为离散值，特别的，这里只预测是或者不是，即为二分类学习。
- 与此同时，若预测值为连续值，称之为 `回归学习(Regression)` 例如，基于房屋面积(HouseArea)来预测房屋售价，同样的，这里也会有一些已知的销售数据。
- 观察以上两个例子, 对于其输入数据例
  如 `[{size:3, condition: benign}, {size:7, condition: malignant}, ... ...]`, 可以看到其结果都是已知的，于是 `Classification` 以及 `Regression` 合称为 `监督学习(Supervised Leaning)`
- 对于以上几段话, 有一些概念需要明确, 以肿瘤良性或恶性的预测判断为例, 假定还有一个影响因素是年龄(Age), 将已有的数据用表格展示如下:

size	age	benign
3	43	N
5	28	Y
4	32	N
8	30	Y
6	45	N

  - 整个表格的内容称为 数据集(dataset), 其中用于训练的子集称为 训练集(trainning set), 数据集由一个个条目组成, 每一个条目称为 样本(sample) 或 实例(instance/example), 对于每一个 实例, 有多个 属性(attribute)/特征(feature) 进行描述, 例如这里是两(d)个, 分别是 size age, 称这个样本是二(d)维的, 这样, 这一组属性就张成了一个 属性空间, 这里属性空间是二维的, 同时对于每一组属性都有一个对应值Y or N, 称为属性值 或者 标记, 因此 实例 对应于属性空间的一个点或者向量, 又被称为 特征向量. 另外, 再看 数据集大小 即为数据集包含的实例数目, 这里是5
- 与$supervised\ learning$相对应的是$unsupervised\ learning$, 即无监督学习
- 任务会包含一系列的输入数据, 它们并没有事先标记, 任务目标是找到数据的内在联系或者结构并进行划分, 据此可能得到一些新的特征, 这样的任务称为 聚类分析(clustering analysis).典型的聚类问题例如
  1. 主题搜索, 将网页上的新闻关于同一个主题的归到一个组
  2. 基因组中特定基因的表达程度
  3. 社交网络分析, 比如判断哪些人是在一个圈子里
  4. 客户市场细分(market sementation), 将客户划分进更细微的市场, 以便针对性的实施对策
  5. ... ...
     有一个有趣的例子, 称为鸡尾酒算法, 有两个人同时讲话, 与此同时有两个麦克风进行录音, 那么

是否能依据两个输入把两个人声分离
又或者, 有一段bgm和一个人的诗朗诵, 能否进行分离
这是无监督学习的另一个例子

下一节我们将以一个简单的例子 单变量线性回归($univariate\ linear\ regression$) 来讨论一个机器学习问题的实际处理过程, 与此同时讨论这一过程中的细节问题

单变量线性回归(Linear Regression with One Variable)

- 机器学习的核心是学习算法$learning\ algorithm$。在监督学习任务中, 学习算法的工作是确定一个从数据集(dataset)到目标值的映射, 使得当得到一个新的样本时, 依据这样一个映射(模型)能够得到一个估计值, 并且希望这个估计值尽可能与真实值符合。
- 这里谈到了许多的问题, 学习算法的目标是依据数据生成模型, 或者一个映射, 这样的映射可能有多种形式, 可能很简单例如线性函数(称为假设函数$hypothesis$), 或者很复杂例如神经网络. 另外, 注意到评价一个算法或者模型的好坏总是离不开真实数据集的, 脱离数据集去谈论学习算法谁更“聪明”是没有意义的, 因此, 学会依据实际数据集去选择合适的算法或者改进乃至于创造一个全新的算法是一个机器学习工作者最宝贵的特质, 所谓经典套路要学懂, 但也必须随机应变.
- 另外, 在确定一个模型, 或者说确定各个参数时, 它们总是在训练集上进行的, 对于模型的评判不能靠训练集上的准确度来衡量, 而是应该在新的数据集上进行测试, 也即应当在测试集上也工作得很好. 这样一种能力称之为 泛化能力(generalization)另外一个问题是假设没有更多的数据集的话, 可以对原始数据集进行划分, 有多种划分方法.
- 最后针对估计值与真实值符合程度的问题, 也就是模型性能的度量, 对于不同问题有不同的考虑, 例如准确率、查准率、查全率等等

房价预测问题

问题总结起来是利用特征的线性组合(h)去尽可能的逼近标记值(y), 使用均方误差刻画逼近的程度, 称为代价函数(J), 目标是使J最小, 方法之一是利用梯度下降算法(Gradient Descent), 它的核心是使参数(自变量)沿着导数(绝对值)减小的方向不断移动, 最终收敛于某个最小值(极小值)

- 标记(mark) or y 是 房价 price
- 样本包含一个 特征(feature) 也即 面积x(area)
- 假设函数(hypothesis)是一个一次函数
  - $h_\theta(x) = \theta_0 + \theta_1 * x$
- 使用 均方差(mean square) 作为 代价函数(cost function), 对应于上述 "估计值与真实值符合程度" 的问
- 目标是不断的改变θ_0和θ_1的值, 使得cost function 最小

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

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

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

$x^{(i)}$表示数据集中第i个实例的属性部分, $y^{(i)}$为值

一种可行的方法是 梯度下降算法(gradient descent), 对于损失函数, 它的自变量是θ_0和θ_1, 设定一个θ_0和θ_1的初值, 并使它们向着导数减小的方向同步改变

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

- 其中是α是 学习率 $learning\ rate$, 它控制着更新幅度大小, α不能设置的过大, 这样可能导致算法发散, 也不能太小以至于算法收敛过慢.
- *Gradient descent can converge to a local minimum, even with the α fixed.* 也即学习速率没必要手动减小, 因为随着收敛过程的进行, 微分项会自动的减小

至此, θ的更新过程变为:

Repeat {

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

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

}

事实上, 这样一个过程又称为 批量梯度下降(batch gd)，指的是在梯度下降的每一步中，我们都用到了所有的训练样本, 也即求和

具体实现(jupyter notebook)

- 代价函数$J_{(\theta_0,\theta_1)}$的计算
  我们应该使用 向量(vector) 或者 矩阵(matrix) 来统一的对数据进行操作, 而不是使用 `for-while-loop` 自己去遍历它们。设 θ = $[\theta_0, \theta_1]_{(1\times2)}$, X = $[1_{m\times1}, \vec{x}_{(m\times1)}]_{(m\times2)}$, y = $[y_1, y_2, \ldots, y_m]_{(m\times1)}$
  - 因此, $h(X) = X * \theta^T$, $J_{(\theta_0,\theta_1)} = \frac{innnerSum([h(X)-y]^2)}{2*m}$
  - 其中, $h(X)$以及y均为m行的列向量, 则他们的差也是m维的列向量, 这里的平方是指对向量的每一个元素进行平方, innerSum是指将向量中的每一个元素进行求和。代码实现如下

```
def cost(X, y, theta):
  """
  X, y, theta均为np.matrix类型
  """
  m = len(y)
  err = X*theta.T-y
  return np.sum(np.power(err, 2))/(2*m)
```

- 梯度下降算法(gradient descent)
  基本的公式形式为: $\theta_j \leftarrow \theta_j - \alpha * \Delta_j$
  其中Δ_j是$J_{(\theta)}$对θ_j的偏导数, 代码的实现关键也就在这里, 如何以一个向量的形式去实现它呢? 针对θ_0, 有:

```
theta_0 = theta_0 - alpha * (np.sum(err)/m)
# 其中的err, m等变量与前述一致
```

  针对θ_1, 多了一个x_i项(这里的i是从行看的, i的范围从1到m, 例如在房价预测中, x_i指第i个房子的面积), 考虑抽出$X = [1_{m\times1}, \vec{x}_{(m\times1)}]_{(m\times2)}$包含特征的第一列$\vec{x}$并记为$x_1$(换名为$x_1$是方便后续多变量回归分析的讨论)。
  因此有:
  $\Delta_1 = \frac{1}{m} \cdot (x_1^T \times err)$
  x_1^T是一个m维的行向量, 而err为m维列向量, 这个乘法是数学意义上的向量乘法, 结果是一个实数, 恰好表达了θ_1的更新过程

```
delta = np.multiply(err, X[:,1])
theta_1 = theta_1 - alpha*(1/m)*delta
```

完整的代码实现

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# 1. 查看原始数据的情况
data = pd.read_csv('./ex1data1.txt', header=None, names=['population', 'profit'])
data.head() # 前几行
data.info() # 97x2(float)
data.plot(kind='scatter', c='red', marker='x', x='population', y='profit')
plt.show()

# 2. 构造矩阵X, y, theta
data.insert(0, 'ones', 1)
n = data.shape[1]
X = np.matrix(data.iloc[:,0:n-1]) # 所有的行，取前n-1列
y = np.matrix(data.iloc[:,n-1:n]) # 所有的行，取最后一列
theta = np.matrix(np.zeros(n-1)) # 初值为0, 0
# X.head(), y.head() # 查看划分是否正确
# X.shape, y.shape, theta.shape

# 3. 构造costFunc(X, y, theta)
def costFunc(X, y, theta):
    m = len(y)
    err = X*theta.T-y
    return np.sum(np.power(err, 2))/(2*m)
# 4. 初始化 迭代次数，学习速率alpha，进行梯度下降
def gradientDecent(X, y, theta, alpha, iter_n):
    m = len(y)
    cost = np.zeros(iter_n)
    for i in range(iter_n):
        err = X*theta.T-y
        delta = (X.T*err).T/m
        theta = theta - alpha*delta
        cost[i] = costFunc(X, y, theta)
    return theta, cost
iter_n = 300
alpha = 0.018
theta, cost = gradientDecent(X, y, theta, alpha, iter_n)

# 5. 作出拟合曲线，作出 代价-迭代次数曲线
fig, ax = plt.subplots() # 绘制多种曲线
ax.scatter(data.population, data.profit, c='red', marker='x') # 注意操作 .
x = np.linspace(data.population.min(), data.population.max(), 100) # 注意操作 linspace
h = theta[0,0]+theta[0,1]*x
ax.plot(x, h)
plt.show()
plt.plot(range(iter_n), cost)
# 可以截取cost的不同部分查看，例如前10次，实际上在3到4次的时候就变得很小了（相对）
# 此后减小的速度大大放缓
plt.show()
```

多变量回归分析

- 多变量线性回归, 此时特征不止一个了, 例如对于房价预测, 除了面积因素, 还可以考虑 房屋使用年数、卧室数量等等, 设共有n个特征, 仍然假设数据集包含m个样本。仍然记$x^{(i)}$为数据集中第i个训练实例, $x^{(i)}$是一个n(在下面公式中, 添加了$x_0 = 1$, 成为$n+1$维)维行向量, 对应第j个特征的值为$x_j^{(i)}$, x_j是一个m维列向量, 由m个样本中第j个特征的值组成.
- h和J以及Δ_j的讨论
  $h(x^{(i)}) = \theta_0 + \theta_1 * x_1^{(i)} + ... + \theta_n * x_n^{(i)}$
  $X = [x^{(1)}, x^{(2)}, \ldots, x^{(m)}]^T = [1, x_1, x_2, \ldots, x_n]_{m\times(n+1)}$
  $\theta = [\theta_0, \theta_1, \ldots, \theta_n]_{1\times(n+1)}$
  $h(X) = X\theta^T_{m\times1}$
  $J(\theta) = \frac{1}{2m} \cdot \sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2$
  $\Delta_j = \frac{1}{m} \cdot (x_j^T \times (X\theta^T - y))$
- 经常见到这样的情况, 也即各个特征的取值范围相差很大, 因此需要$feature\ scaling$也即特征缩放, 一般的取$\frac{x-\mu}{\delta}$
- α学习率的选取, 通常可以这样尝试
  $\alpha = 0.01, 0.03, 0.1, 0.3, 1, 3, 10, ...$
- 特征和多项式回归
  - 特征的转换
    有时, 对于给出的一些特征, 需要对其作出一定的变换来得到更优的模型, 例如预测房屋价格的问题中, 假设给出的特征是 $frontage$ = 房屋临街宽度, $depth$ = 房屋侧边长度.可以直接构造假设函数为:
    $h(x) = \theta_0 + \theta_1 * frontage + \theta_2 * depth$
    但另一方面看, 房屋价格更可能与 特征$area = frontage \times depth$ 也即房屋面积相关. $area$或许是一个更好的特征.因此有时需要依据实际数据选择合适的特征
  - 据此, 提出多项式回归, 例如假设函数的形式是:
    $h(x;\theta) = \theta_0 + \theta_1 * x + \theta_2 * x^2$是一个二次曲线, 那么作变换 $x_1 = x, x_2 = x^2$就有
    $f(x;\theta) = \theta_0 + \theta_1 * x_1 + \theta_2 * x_2$
  - 此外, 多项式可以拟合任意曲线, 除了常数和整数次幂还包括1/2次幂

正规方程

正规方程是通过求解下面的方程来找出使得代价函数最小的参数的：$\frac{\partial}{\partial\theta_j}J\left(\theta_j\right)=0$ 。
假设我们的训练集特征矩阵为 X（包含了$x_0 = 1$）并且我们的训练集结果为向量 y，则利用正规方程解出向量 $\theta = \left(X^TX\right)^{-1}X^Ty$ 。
上标T代表矩阵转置，上标-1 代表矩阵的逆。

实验过程中遇到的问题及解决方案

1. jupyter notebook 更换主题

```
pip install jupyterthemes grade3 jt -t grade3 -f fira -fs 17 -cellw 90% -ofs 15 -dfs 15 -T -T
```

2. 关于dataframe删除某一行或某一列的方法

```
df.drop(columns=['b'])       # drop a column
```

3. Markdown基本教程
    1. Markdown基本教程
    2. Markdown数学公式

$linear\ Algebra\ review$

$scalar\ multiply$ 标量乘法
$commutative$ 可交换的
$associative\ property$ 可结合性
$Identity\ Matrix\ I \times A = A \times I = A$
$martix\ inverse$ **方阵的逆**
$square\ matrix$
$singular\ or\ degenerate$ **没有逆的矩阵**
$transpose\ A^T$ 转置

$Linear\ Regression\ with\ mutiple\ variables\ (features)$

Notation
n = number of features
m = size of training set
$x^{(i)}$ = the i^{th} sample(instance/example) of training set, which is a n-dimmension vector
$x_j^{(i)}$ = value of feature j in i^{th} sample

$how\ to\ express\ hypothesis\ for\ LRM$

$Gradient\ descent\ in\ practice:feature\ scaling$

$Gradient\ descent\ in\ practice:learning\ rate\ \&\ debugging$

$debugging$: how to make sure that GD works correctly

$learning\ rate$: how to choose α

$Features\ and\ polynomial\ regression$

- how we choose appropriate features

正规方程组($Solution$)

求偏导, 置为0, 得方程组, 求解之

结果是

$Vectorize$

对率回归模型(Logistic Rregression Model)

内容回顾与模型引出

对率回归模型(Logistic Regression Model)或者称逻辑回归模型是一类针对分类问题而产生的模型，它是从线性回归模型中衍生而来的。

在线性回归模型中，我们使用特征的线性组合来对连续的标记值进行拟合，并确保总体拟合值与标记值尽可能接近，为了刻画这种接近程度，定义了一个损失函数或者代价函数，也即均方差，表现为两点间的距离。同时为了方便计算机对参数进行迭代，从微分角度给出了参数变化方向也即梯度函数，并使用学习率控制参数变化的快慢以确保能够收敛。

注意到，线性回归模型也不是仅仅只能够拟合线性的标记值分布。如果对其中的特征进行变换例如它的幂并将其作为新的特征，就可以以多项式曲线的方式对标记值进行拟合，理论上可以逼近任意曲线，然而，当幂次过高时可能出现过拟合现象，这可以通过正则化的手段进行处理，其思想核心在于尽可能减小参数值以使得曲线不会过于扭曲。

另一方面，还可以考虑对y值进行变换，例如，假设通过对数据的观察我们猜测y与x呈指数函数关系，例如$y = ke^{mx}, (k > 0, m \neq 0)$，那么不妨在等式两边取对数，此时就变成最简单的线性回归模型了。基于这一思想，设想一个函数将任意R内的实数都映射到(0, 1)内, 也即从$h(x) = \theta^T X$变为$h(x) = g^{(-1)}(\theta^T X)$, 同时还要定义一个代价函数cost，使得在给定一组参数值θ时，若$y = 0$(以二分类为例)但$h(x) \to 1$时cost会很大(此时表示预测结果反了，就应该给予较大的代价), 同时cost也必须是凸函数并存在关于θ的导数，这样才可以求梯度并可以迭代收敛。

逻辑回归模型的建立

貌似找到这样一个函数很困难，但有人找到了并告诉了我，因此这里我也将告诉你。首先是这个$R \to (0, 1)$的函数，可以取$sigmoid$函数，意为S形的曲线函数，函数定义是$sigmoid(z) = \frac{1}{1+e^{-z}}$。因此有假设函数$h(x) = sigmoid(\theta^T X)$。另外，代价函数可以取为$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}log(h(\boldsymbol{x}^{(i)})) - (1 - y^{(i)})log(1 - h(\boldsymbol{x}^{(i)}))]$。注意这里$\boldsymbol{x}^{(i)}$是一个向量, 表示第i个实例（特征向量），假设有n个特征，则$\boldsymbol{x}^{(i)}$就是n维向量。最后再求它对θ_j的偏导，值得注意的是，对于$sigmoid\ function\ g$，有$g' = g(1 - g)$。这里推导过程不展开讲，但令人惊讶的是，它的数学形式竟和线性回归模型中的偏导数相同！也即$\frac{\partial}{\theta_j}J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[h(\boldsymbol{x}^{(i)})) - y^{(i)}]x_j^{(i)}$

那么陈述至此或许你已经明白，对率回归只不过是在线性回归的基础上做了一点改动而已，尽管这样的改动最初或许是十分难以想到的，但没关系我们现在已经掌握了它。补充几点说明，已经训练好了参数，当给出一个新的示例，例如$x^{(m+1)}$，那么如何知道它的类别呢？（以二分类示例）只需要计算$h(x^{(m+1)})$然后与0.5比较，大于0.5即为正类，否则为负类。同时结合sigmoid函数图像知当$\theta^T X > 0$是就有$h(x) = g(\theta^T X) > 0.5$，因而$\theta^T X = 0$就成为了决策边界，如果是仅有两个特征且为二分类的问题，那么可以进行可视化，会看到$\theta^T X = 0$形成了类边界。那么对于多分类问题，可以采用oneVSall或者oneVSrest的方法，不再展开陈述。同样的，逻辑回归模型也可以采取正则化的方法缓解过拟合。

关键代码实现

- 将课件中的公式以向量化的形式完整呈现出来，这样的代码是原汁原味的，但是参数（初值）调整会十分麻烦，不能随意的赋初值，否则绝大多数情况不会收敛到合理值。

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

# 1 读取数据
data = pd.read_csv("ex2data1.txt", header=None, names=["ScA", "ScB", "Accepted"])
# 2 画散点 正类负类分开
pos_d = data[data['Accepted'] == 1]
neg_d = data[data['Accepted'] == 0]
ax, fig = plt.subplots()
fig.scatter(pos_d.ScA, pos_d.ScB)
fig.scatter(neg_d.ScA, neg_d.ScB, c='r', marker='x)
plt.show()
# 3 划分X, y, 初始化theta
data.insert(0, 'Ones', 1)
n = data.shape[1]
X = np.matrix(data.iloc[:, 0:n-1])
y = np.matrix(data.iloc[:, n-1:n])
theta = np.matrix(np.zeros(n-1))
# X.shape, y.shape, theta.shape
# ((100, 3), (100, 1), (1, 3))
# 4 写sigmoid函数
def sigmoid(z):
    return 1/(1+np.exp(-z))
# 5 写costFunc
def costFunc(theta, X, y):
    m = len(y)
    h_x = sigmoid(X*theta.T)
    return ((-y.T)*np.log(h_x)-(1-y.T)*np.log(1-h_x))/m
# costFunc(theta, X, y) # 0.693147
# 6 写出梯度下降算法
def gradientDescent(theta, X, y, alpha, iters):
    cost = np.zeros(iters)
    for i in range(iters):
        cost[i] = costFunc(theta, X, y)
        h_x = sigmoid(X*theta.T)
        err = h_x-y
        delta = (X.T*err)/m
        theta = theta - alpha*delta.T
    return theta, cost
# 7 初始化参数，进行训练，作出cost图、决策边界曲线
alpha = 0.0003
iter_n = 2000
theta = np.matrix([-30, 2, 1])
theta, cost = gradientDescent(X, y, theta, iter_n, alpha)
plt.plot(range(iter_n), cost)
plt.show()
# cost[-1], theta
# (0.20651168067356068, matrix([[-30.02499304,   0.24517859,   0.24078545]]))
# 决策边界图代码略
```

- 使用库函数，但需要实现代价函数cost和梯度（一步）gradient，注意参数类型合法以及参数顺序

```
"""
import scipy.optimize as opt
result = opt.fmin_tnc(func=costFunc, x0=theta, fprime=gradient, args=(X, y))
"""
def costFunc(theta, X, y):
  pass
def gradient(theta, X, y):
  return delta.T
# 定义好后，传入fmin_tnc方法中，args是costFunc除第一个参数theta以外的其它参数，gradient返回导数向
# fmin_tnc会在内部自动设置学习率并不断调整，theta会不断迭代，且为array类型，期间多次报错矩阵乘法维
```

非线性的二分类与正则化参数的使用

- 代价函数

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

- 梯度
  如果我们要使用梯度下降法令这个代价函数最小化，因为我们未对θ_0 进行正则化，所以梯度下降算法将分两种情形：

$$Repeat\ until\ convergence\{$$

$$\theta_0:=\theta_0-a\frac{1}{m}\sum_{i=1}^{m}[h_\theta\left(x^{(i)}\right)-y^{(i)}]x_0^{(i)}$$

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

$$\}$$

$$Repeat$$

  对上面的算法中 j=1,2,...,n 时的更新式子进行调整可得：

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

Neural Network(神经网络)

题外话(verbose)：

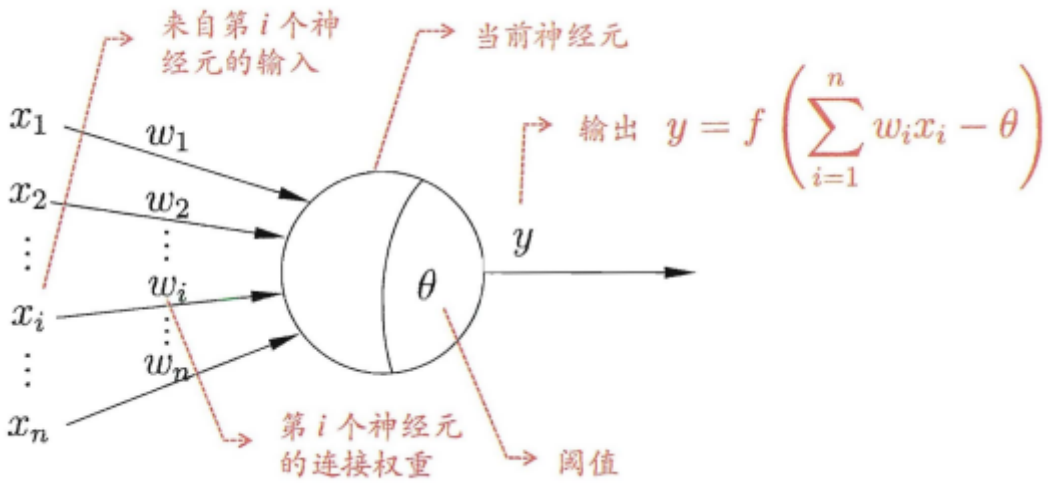
- 关于vscode与github的互动, 我起初的想法是:
  1. 我在本地写代码，写着写着忽然觉得有必要为他创建一个github仓库来管理代码。此时我已经安装了git命令工具，vscode的源代码管理也有对应的图形化界面，先认证，然后提交和推送到远程仓库，这里我想让他自己把文件夹作为仓库直接推上去（新建），不可行
  2. 因此，你需要一开始就在github上新建一个仓库，再克隆下来，接着使用vscode编辑你的代码、笔记等等文件，之后再进行更多的操作。具体做法参考：Windows+VScode配置与使用git，超详细教程，赶紧收藏吧
- github时不时就打不开 DNS污染
  告别无法访问的github（附解决方案）

缘起

在前面关于线性回归模型和对率回归模型的讨论中，我们为了使模型能够适应更多的情景，一方面对于特征进行转化，例如使用多项式，另一方面对于标记值进行转换等等。在多项式模型中，当幂次过高时会出现过拟合现象，一种可行的办法是使用正则化来缓解过拟合现象。而神经网络则是通过将对率回归模型进行嵌套以提高表达能力，它从生物学神经的角度出发，使用一个对率模型作为单个的神经元，神经元之间使用权值矩阵进行连接从而形成神经网络

神经网络

- **神经元-（MP神经元模型）**



如图所示为MP神经元模型，它包括一个输入$x_i(i=1\rightarrow n)$，一系列的权值$\omega_i(i=1\rightarrow n)$(预置参数、待学习)，接着把它们对应相乘($\omega^T x$)得到输入，神经元的处理是将其与一个阈值比较接着套上一个转化函数(例如sigmoid函数或者阶跃函数等)，函数值即为输出值y.
因此:

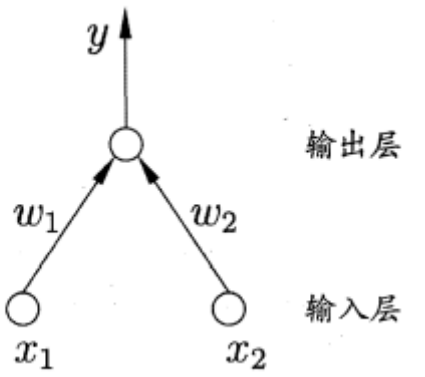
$$y=sigmoid(w_1x_1+w_2x_2+\cdots+w_nx_n-\theta)$$

如果使用熟悉的记号

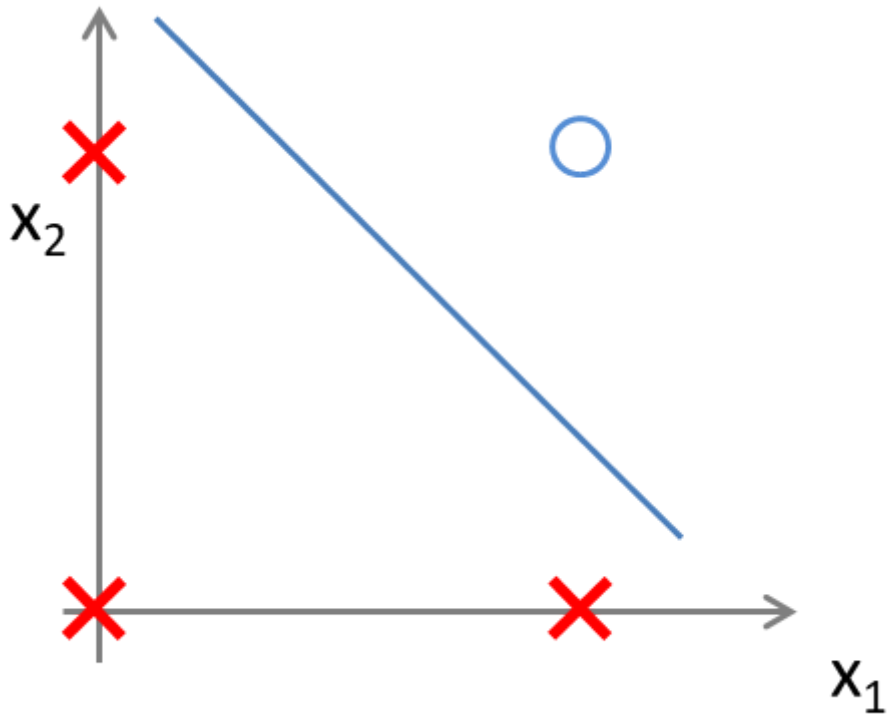
$$\theta = [\theta_0, \theta_1, \ldots, \theta_n]_{(n+1)\times 1}$$
$$x_0 = 1$$
$$x = [x_0, x_1, \ldots, x_n]_{(n+1)\times 1}$$
$$so\ \ that,$$
$$y = sigmoid(\theta^T x)$$

那么这将与对率函数的假设函数相同

- **感知机(Perceptron)**
  感知 (Perceptron 由两层神经元组成:

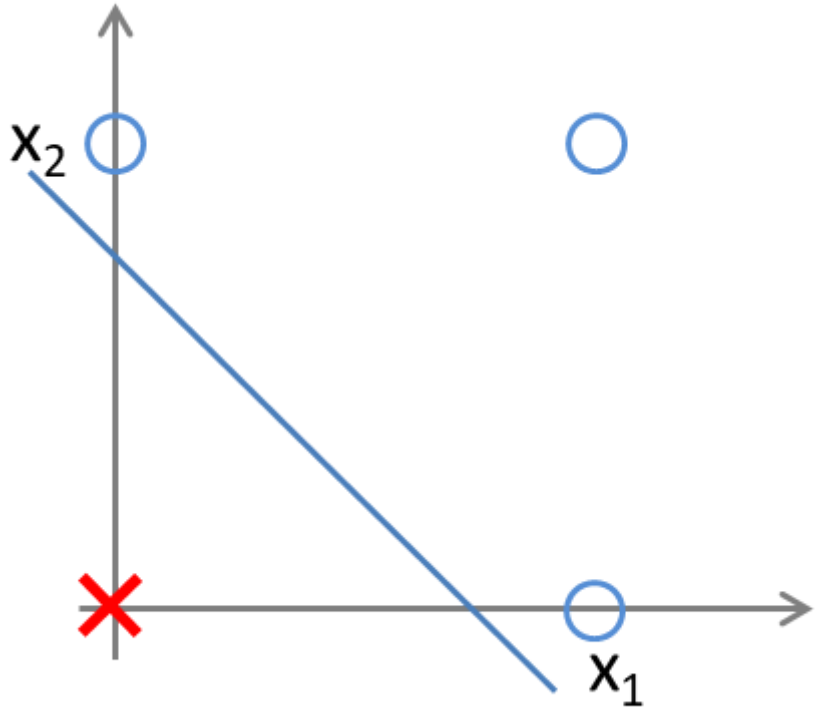


  $y = w_1 x_1 + w_2 x_2 - \theta$, 能够进行线性的划分，例如与或非运算，
  - AND, $only\ \ when\ \ x_1 = x_2 = 1 \rightarrow y = 1$



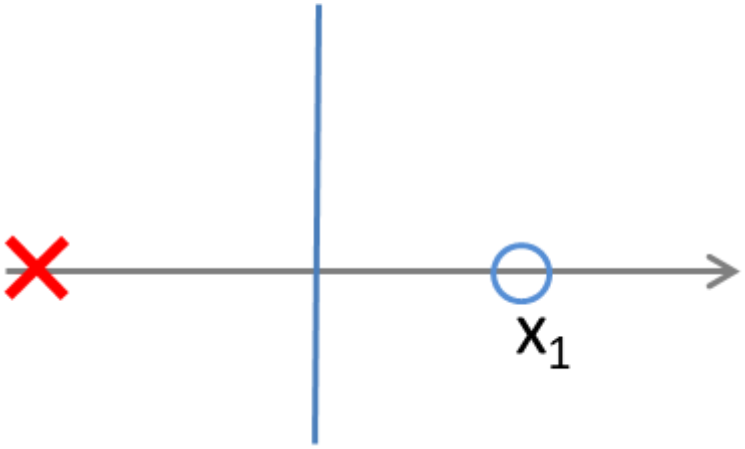
$w_1 = 1, w_2 = 1, \theta = -1.5$

- OR, $only \ \ when \ \ x_1 = x_2 = 0 \rightarrow y = 0$



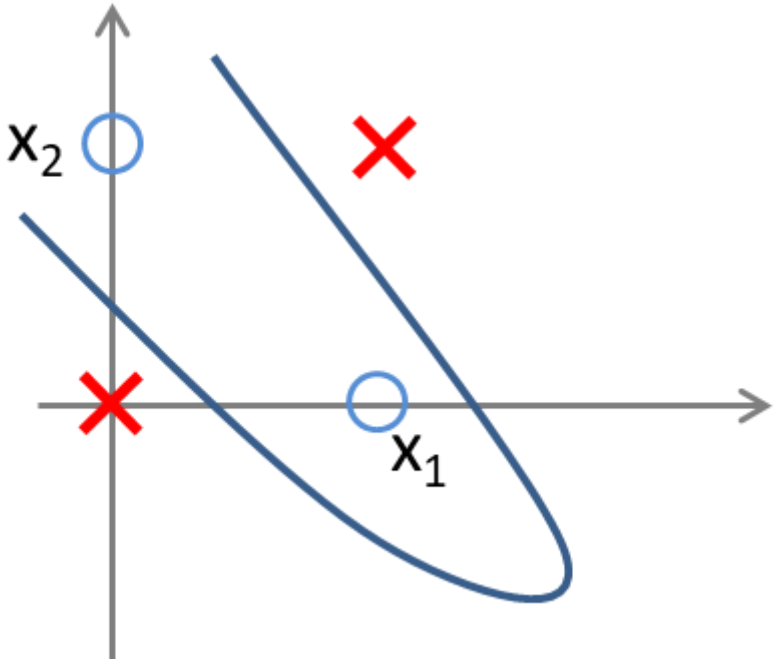
$w_1 = 1, w_2 = 1, \theta = -0.5$

- NOT

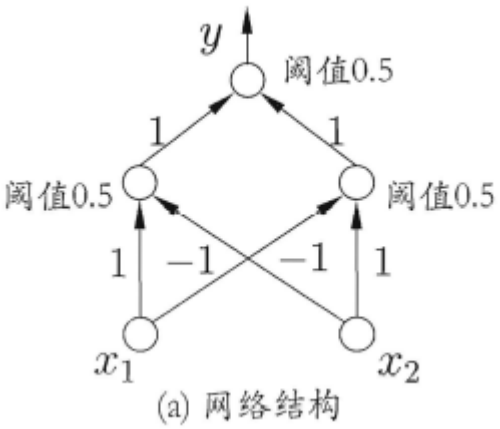


$w_1 = -1, w_2 = 0, \theta = 0.5$

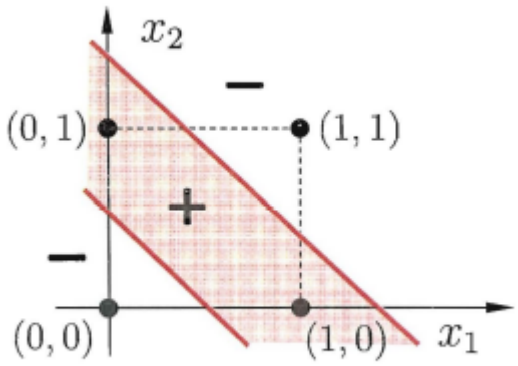
那么XOR异或问题呢？先看图示



可见不是一个线性分类问题，那么仅使用一层是不行的，考虑$x_1 \otimes x_2 = (\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)$，可以将其分成两层，先是输入层$x_1, x_2$，接下来一层是两个神经元，分别计算两个括号值，再有一层含一个神经元求并值。

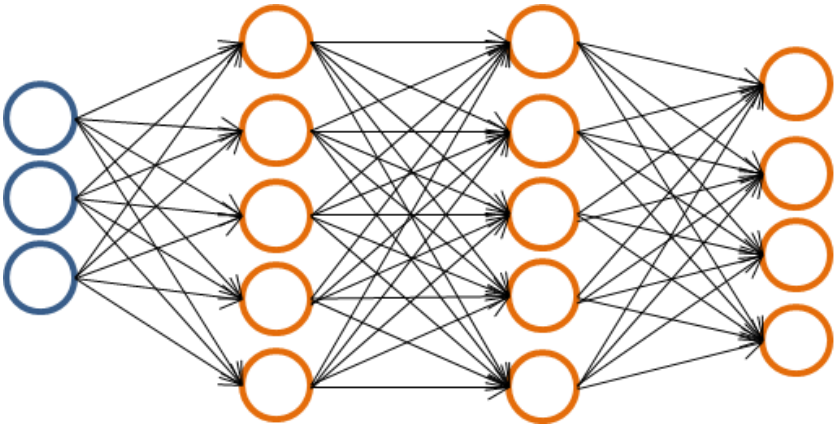


(a) 网络结构



(b) 分类区域

- **神经网络**
  - 实际上，上面的实现异或这一非线性划分的两层感知机可以视为一个具备完整结构的神经网络，它包含一个输入层(input layer)，计算中间值的隐含层(hidden layer)以及一个输出层(output layer)，同时输出值只有一个，因而是个二分类问题。
  - 一个典型的神经网络结构如下图所示：



$$h_\Theta(x) \in \mathbb{R}^4$$

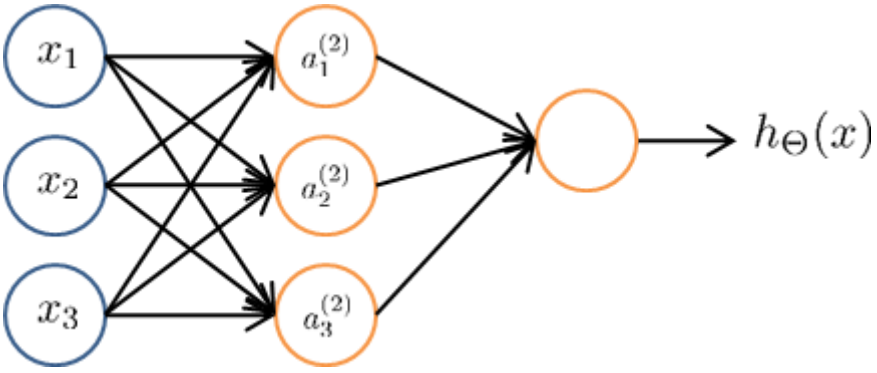
由图可知，蓝色部分为输入层，含有3个特征，有两个隐含层，一个输出层，输出是一个四维向量，因此是一个多分类问题，且包含四个类别。其中$h_\Theta(x)$是形如$[0,1,0,0]$的向量（该实例表示分类结果是第二类）。

接下来将对神经网络进行更为详细的探讨，首先是一些记号表示：

数据集: $\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\cdots,(x^{(m)},y^{(m)})\}$

1. L: 神经网络的层数，例如上图中$L=4$。
2. S_l: 第l层的神经单元数, 例如上图中$S_L=4$, 也即输出层包含4个结点(注意不考虑偏置节点)。
3. $\Theta^{(l)}$: 从第l层连接到$l+1$层的权值矩阵。
4. $a_j^{(l)}$: 表示第l层第j个节点的输出值，而$a^{(l)}$则是一个S_l+1维的向量, 作为$l+1$层的输入, 其中$a_0^l=1$是偏置节点(起到阈值的效果).又例如$a_1^{(1)}=x_1$表示第一层第一个结点值(下标从0开始)。
5. $z_j^{(l)}$: 表示$a_j^{(l)}$的输入，此处$l>=2, j>0$. 因此$z^{(l)}$是一个S_l维的向量。

  - 接下来是实例探讨，如图：



例如，计算$a_2^{(2)}$有:

$$z_2^{(2)} = \Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3$$
$$a_2^{(2)} = sigmoid(z_z^{(2)})$$

则权值矩阵$\Theta^{(1)}$的形状是3x4, 据此$\Theta^{(l)}$的形状是$S_{l+1} \times (S_l + 1)$
将其向量化, $z^{(2)} = [z_1^{(2)}, z_2^{(2)}, z_3^{(2)}]$以及$x = [x_0, x_1, x_2, x_3]$并假设$z^{(2)}$和$x$均为列向量:

$$z^{(2)} = \Theta^{(1)}x$$
$$a^{(2)} = [a_0^{(2)}, sigmoid(z^{(2)})], a_0^{(2)} = 1$$

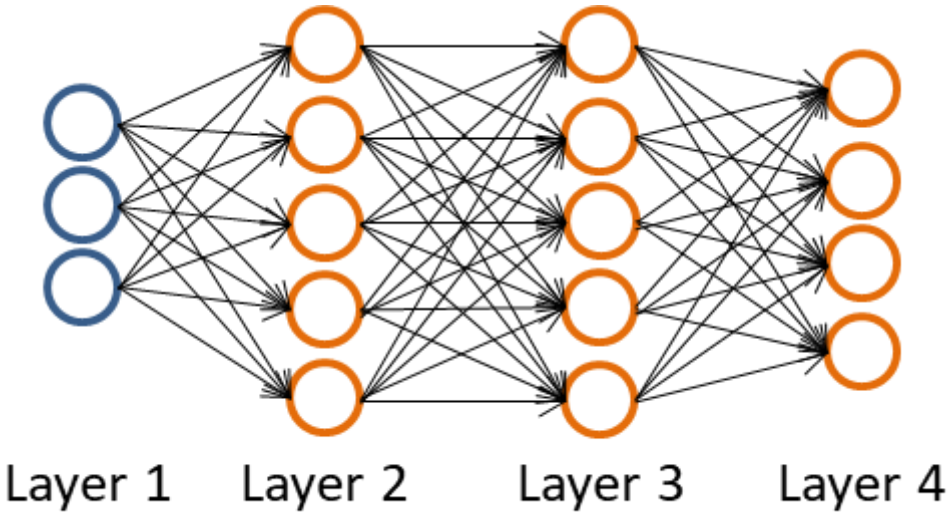
据此计算出$a^{(3)} = sigmoid(\Theta^{(2)}a^{(2)}) = h_\Theta(x)$

- 代价函数
  以上得到了假设函数$h_\Theta(x)$, 接下来是代价函数的书写，仍假设有m条样本数据，输出类别数为K。则依据直观思想，代价函数为这K个类别的代价之和，其形式类似于对率回归的代价函数，如下:

$$J_\Theta = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\ln(h_\Theta(x^{(i)})_k) + (1-y_k^{(i)})\ln(1-h_\Theta(x^{(i)})_k)\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{S_{l+1}}\sum_{j=1}^{S_l}\Theta_{ij}^{(l)}$$

  其中$h_\Theta(x^{(i)})_k$表示K维向量假设函数输出的第k个值

- 梯度
  梯度的求解似乎很麻烦呢！不过也会一步一步的推导呢！这里的方法主要是BP算法(Backward Propagation)，接下来将一步步的说明
  误差$\delta_j^{(l)}$表示第l层第j个节点的误差值, $\delta^{(l)}$则是第l层的误差向量



从第4层算起:

$$\begin{aligned}
\delta^{(4)} &= h_\Theta(x) - y \\
\delta^{(3)} &= (\Theta^{(3)})^T \delta^{(4)} \cdot g'(z^{(3)}) \\
&= (\Theta^{(3)})^T \delta^{(4)} \cdot g(z^{(3)})(1 - g(z^{(3)})) \\
&= (\Theta^{(3)})^T \delta^{(4)} \cdot a^{(3)} \cdot (1 - a^{(3)}) \\
\delta^{(2)} &= (\Theta^{(2)})^T \delta^{(3)} \cdot a^{(2)} \cdot (1 - a^{(2)})
\end{aligned}$$

- 有了误差计算后，就可以完整的实现一个BP算法了，给定训练数据集
  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$
  令$\Delta_{ij}^{(l)} = 0$

  For $i = 1$ to m

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

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

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

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

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

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

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

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

- 完整的神经网络训练过程
  1. 选择合适的网络结构。Pick a network architecture (connectivity pattern between neurons)一般来说，只包含一个隐含层的神经网络是最常见的，如果选择包含多个隐含层的网络，各隐含层的节点数应当相当相等。
  2. 对Θ随机初始化，不可以初始化为0。Randomly initialize weights.
  3. Implement forward propagation。实现前向传播计算$h_\Theta(x)$
  4. Implement code to compute cost function $J(\Theta)$实现代价函数
  5. Implement backprop to compute partial derivatives。通过BP算法计算代价函数对各个θ的偏导数.
  6. 梯度检测。Use gradient checking to compare computed using backpropagation VS using numerical estimation. Then disable gradient checking code. ($estimation = \frac{J_{\theta+\epsilon} - J_{\theta-\epsilon}}{2\epsilon}$)
  7. 对$J(\Theta)$进行优化。Use gradient descent or advanced optimization method with backpropagation to try to minimize $J(\Theta)$

实验三、XXXX

问题记录

- 如何设置markdown的字体颜色

  ```
  <font color=#ABC>something</font>
  ```

  - 示例

    CornflowerBlue

    Chartreuse

    DarkRed

    GoldenRod

    A0A

    【Markdown笔记】设置字体颜色

- LaTex数学公式中的空格与换行
  Latex 中的空格汇总
  如下分别是换行和 空格（最常用的一种）

```
\\
\enspace
```

应用机器学习的建议(Advice for Applying Machine Learning)

本部分主要讨论如何评判模型的性能以及在新样本上模型预测值与实际值相差较大时应当如何去改进等等问题。

下一步做什么?

在你得到你的学习参数以后，如果你要将你的假设函数放到一组新的房屋样本上进行测试，假如说你发现在预测房价时产生了巨大的误差，现在你的问题是要想改进这个算法，接下来应该怎么办?

1. 获取更多的训练样本；这通常是困难的，很多时候并非增加数据量就可以解决问题
2. 尝试去掉一些特征；仔细挑选需要去掉的特征以防止过拟合，由于没有合适的准则，因此不一定凑效
3. 尝试发现更多的特征；从更多的角度看待这一问题
4. 尝试多项式特征；对于已有的特征，增加其多项式表达
5. 增加正则化系数λ的大小
6. 减小正则化系数λ的大小

Evaluate your hypothesis

在房价预测的示例中，我们通过作图来直观的感受了过拟合与欠拟合的问题与原因。然而，大多数情况下，由于特征的复杂性，我们无法作出图像，因此需要一种数据化的形式去评估我们的假设模型.

将数据集划分为两个部分，即训练集和测试集。例如，可以将数据划70%为Trainning set and 30% for test.注意数据划分应当均匀，一个做法是对原数据进行shuffle.另外，针对划分方法，可以是直接一刀划为两部分，依据不同划法，进行多次训练，称之为**留出法(hold-out)** 得到的结果可以使用算术平均；也可以将数据集划分为k个大小近似的互斥子集，每次使用k-1个子集进行训练，剩下一个用于验

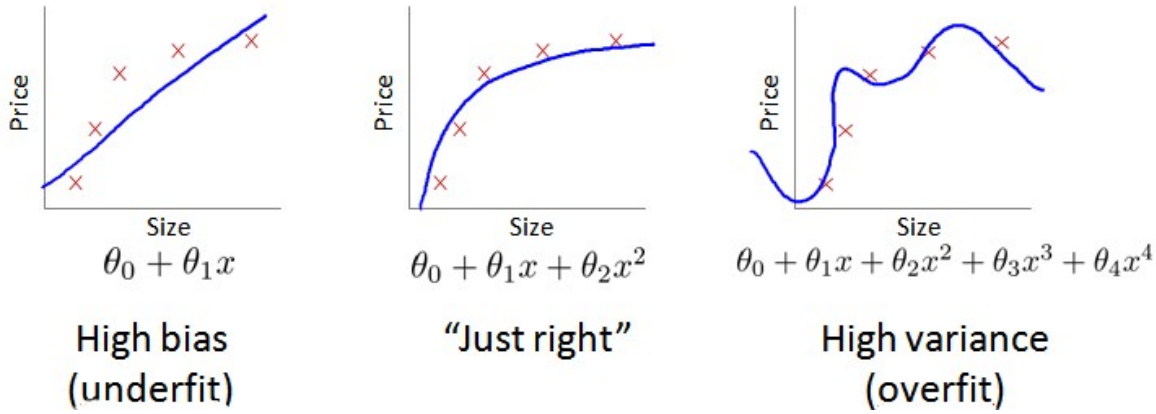
证，因而可以进行k次训练，由于子集划分也具有任意性，因而可以进行m组，称之为**k折交叉验证(k-fold cross validation)**.

将数据集划分为三个部分，即训练集、交叉验证集和测试集。具体做法是：

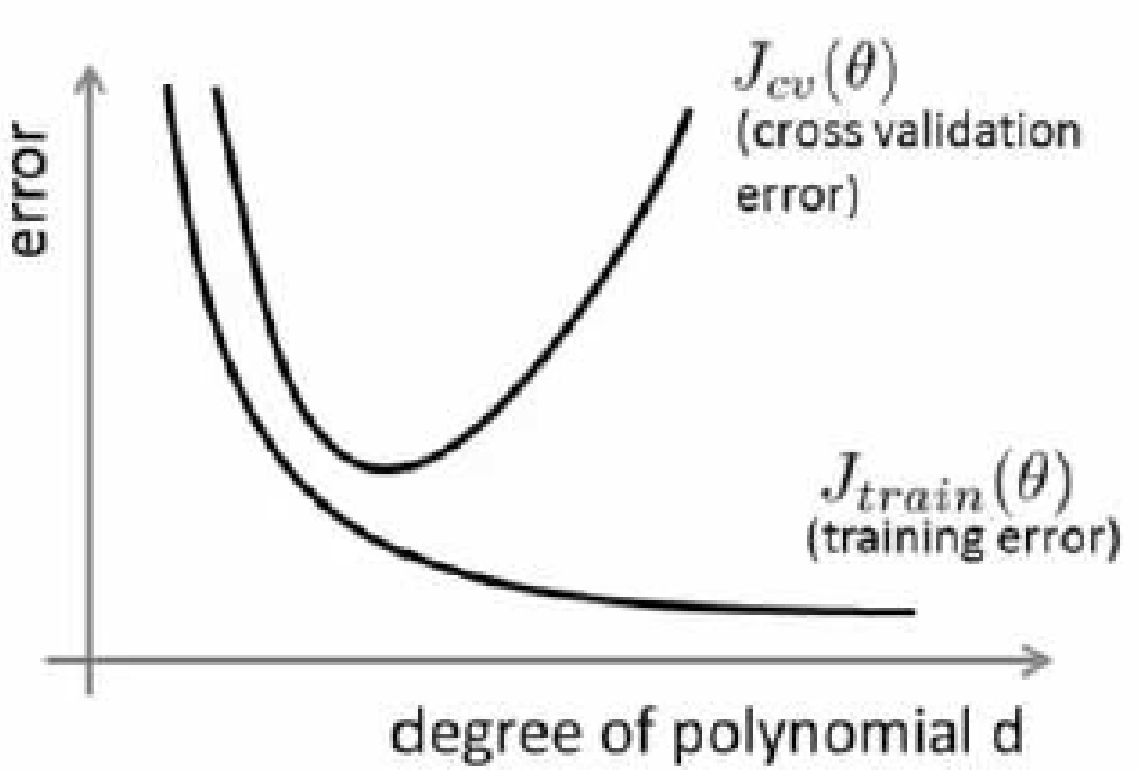
1. 使用训练集训练出10个模型
2. 用10个模型分别对交叉验证集计算得出**交叉验证误差**（代价函数的值）
3. 选取代价函数值最小的模型
4. 用步骤3中选出的模型对测试集计算得出**推广误差**（代价函数的值）

Bias and Variance(偏差与方差)

偏差和方差是描述模型拟合程度的度量，先给出结论，设数据集为$\mathbf{x}$，当$bias^2(\mathbf{x})$也即偏差过高时表示欠拟合(underfit)，当$var(\mathbf{x})$也即方差过高时表示过拟合(overfit).如图所示：



我们通常会通过将训练集和交叉验证集的代价函数误差与多项式的次数绘制在同一张图表上来帮助分析:



如图所示，横轴表示多项式特征的最高次数，随着幂次的增大，模型在训练集上的拟合程度越来越高，表现为训练误差不断减小，而交叉验证集误差则呈对勾状，表现为一开始误差较大随后不断下降至一个最低点随后又开始不断上升。可以得出，当训练集误差和交叉验证集误差均比较大时，模型欠拟合，偏差高了，需要更多的特征输入，单纯增加训练集大小效果很可能效果不大。而当训练集误差比较小但交叉验证集误差较大时，此时模型过拟合，方差高了，需要提高正则化系数或者减小多项式特征的幂次。

关于偏差与方差的含义或者定义，从西瓜书上了解到：**方差**是指在大小相同的不同训练集上预测值与预测期望值差的平方的期望：

$$var(\boldsymbol{x}) = \mathbb{E}_D \left[\left(f\left(\boldsymbol{x}; D\right) - \bar{f}\left(\boldsymbol{x}\right) \right)^2 \right]$$

期望输出与真实标记的差别称为偏差：

$$bias^2(\boldsymbol{x}) = \left(\bar{f}\left(\boldsymbol{x}\right) - y \right)^2 \ .$$

偏差度量了期望输出与真实结果的偏离程度，即刻画了学习算法本身的拟合能力，方差度量了同样大小的训练集的变动所导致的学习性能的变化，即刻画了数据扰动所造成的影响（此外还有噪声，她=它表达了在当前任务上所有算法能达到的期望泛化误差下界，即刻画了学习问题本身的难度）

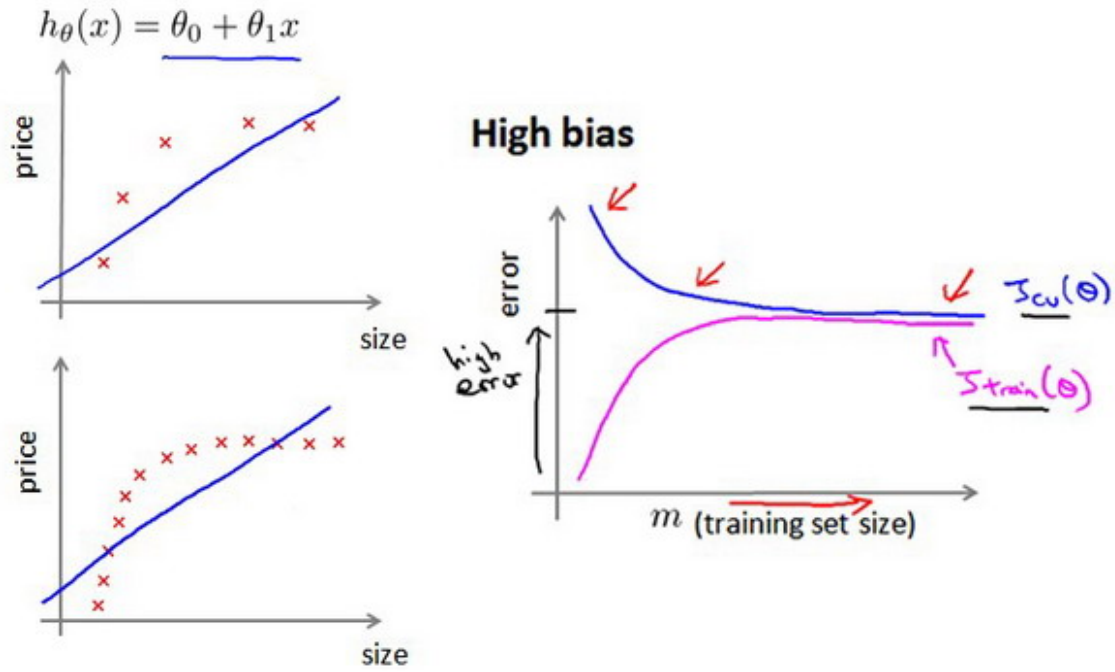
正则化和偏差/方差

我们选择一系列的想要测试的λ值，通常是0-10之间的呈现2倍关系的值（如 $0, 0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64, 1.28, 2.56, 5.12, 10$：共12个）。 我们同样把数据分为训练集、交叉验证集和测试集。
选择的方法为：

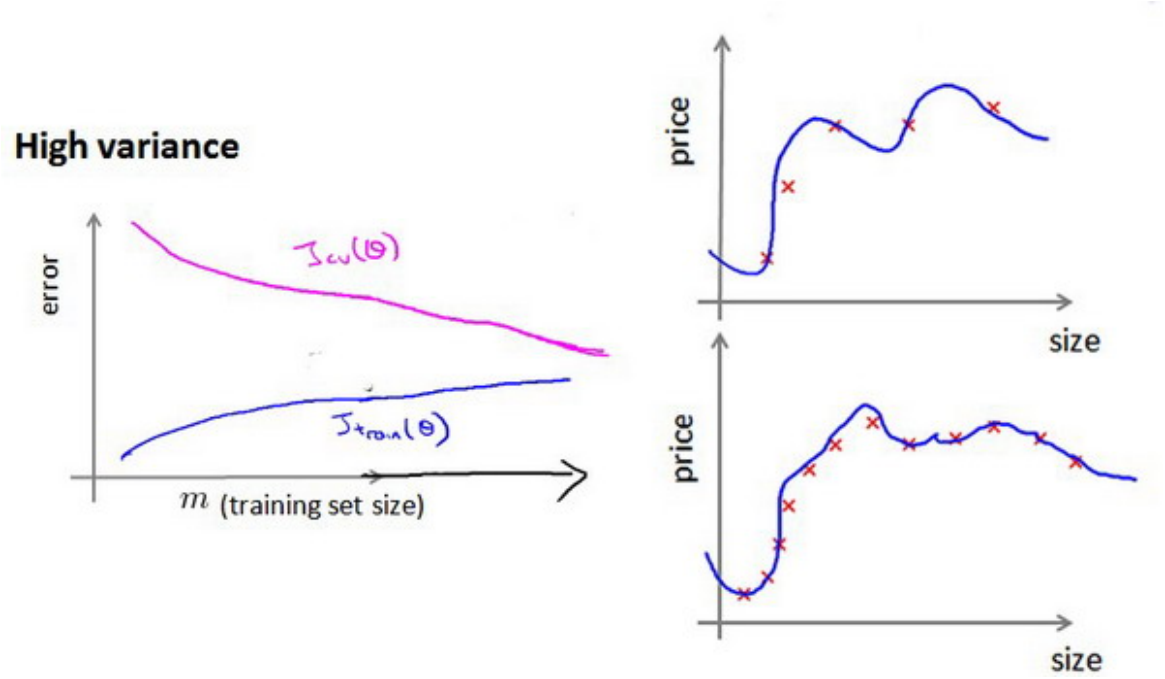
1. 使用训练集训练出12个不同程度正则化的模型
2. 用12个模型分别对交叉验证集计算的出交叉验证误差
3. 选择得出交叉验证误差最小的模型
4. 运用步骤3中选出模型对测试集计算得出推广误差，我们也可以同时将训练集和交叉验证集模型的代价函数误差与λ的值绘制在一张图表上

学习曲线

将数据一行一行的加入到训练集，绘制训练误差和交叉验证集误差随训练集大小m的曲线，此即为学习曲线。一开始，由于数据量较少，模型曲线可以完全通过训练样本点，训练误差较小，但验证集误差较大，随着样本数据的增大，训练误差增大而验证集误差减小，最后他们应当稳定在一个平台。
如何利用学习曲线识别高偏差/欠拟合：作为例子，我们尝试用一条直线来适应下面的数据，可以看出，无论训练集有多么大误差都不会有太大改观：



也就是说在高偏差/欠拟合的情况下，增加数据到训练集不一定能有帮助。 如何利用学习曲线识别高方差/过拟合：假设我们使用一个非常高次的多项式模型，并且正则化非常小，可以看出，当交叉验证集误差远大于训练集误差时，往训练集增加更多数据可以提高模型的效果。



也就是说在高方差/过拟合的情况下，增加更多数据到训练集可能可以提高算法效果。

决定下一步做什么

1. 获取更多的训练样本←高方差/过拟合/训练集误差较小但验证集误差较大
2. 尝试去掉一些特征←高方差/过拟合/训练集误差较小但验证集误差较大
3. 尝试发现更多的特征←高偏差/欠拟合/训练集误差和验证集误差都较大
4. 尝试多项式特征←高偏差/欠拟合/训练集误差和验证集误差都较大
5. 增加正则化系数λ的大小←高方差/过拟合/训练集误差较小但验证集误差较大
6. 减小正则化系数λ的大小←高偏差/欠拟合/训练集误差和验证集误差都较大

- 使用较小的神经网络，类似于参数较少的情况，容易导致高偏差和欠拟合，但计算代价较小使用较大的神经网络，类似于参数较多的情况，容易导致高方差和过拟合，虽然计算代价比较大，但是可以通过正则化手段来调整而更加适应数据。 通常选择较大的神经网络并采用正则化处理会比采用较小的神经网络效果要好。 对于神经网络中的隐藏层的层数的选择，通常从一层开始逐渐增加层数，为了更好地作选择，可以把数据分为训练集、交叉验证集和测试集，针对不同隐藏层层数的神经网络训练神经网络， 然后选择交叉验证集代价最小的神经网络。

机器学习系统的设计(Machine Learning System Design)

构建一个学习算法的推荐方法为：

1. 从一个简单的能快速实现的算法开始，实现该算法并用交叉验证集数据测试这个算法
2. 绘制学习曲线，决定是增加更多数据，或者添加更多特征，还是其他选择

3. 进行误差分析：人工检查交叉验证集中我们算法中产生预测误差的样本，看看这些样本是否有某种系统化的趋势

> 以我们的垃圾邮件过滤器为例，误差分析要做的既是检验交叉验证集中我们的算法产生错误预测的所有邮件，看：是否能将这些邮件按照类分组。例如医药品垃圾邮件，仿冒品垃圾邮件或者密码窃取邮件等。然后看分类器对哪一组邮件的预测误差最大，并着手优化。 思考怎样能改进分类器。例如，发现是否缺少某些特征，记下这些特征出现的次数。 例如记录下错误拼写出现了多少次，异常的邮件路由情况出现了多少次等等，然后从出现次数最多的情况开始着手优化。 误差分析并不总能帮助我们判断应该采取怎样的行动。有时我们需要尝试不同的模型，然后进行比较，在模型比较时，用数值来判断哪一个模型更好更有效，通常我们是看交叉验证集的误差。 在我们的垃圾邮件分类器例子中，对于“我们是否应该将discount/discounts/discounted/discounting处理成同一个词？”如果这样做可以改善我们算法，我们会采用一些截词软件

类偏斜的误差度量(Error Metrics for Skewed Classes)

例如癌症的检测，假定是一个二分类问题，即患癌症或者不患癌症，此时一般而言，是健康人的可能性远大于患病的可能性，例如健康者占比99.5%，两个类别占比极度不平衡，称为类偏斜(skewed class)。那么有这样一个分类器，他总是预测为健康，那么他的错误率也只有0.5%，但显然这个预测器不能称之为一个好的预测器，因为它无法检测出癌症患者。因此错误率并不总是很好的刻画预测器的好坏。

这里对其进行细化，例如，在预测出的癌症患者(预测为阳性)中有多大比例确实患上了癌症(真阳性，True Positive)，这一比例称为准确率(***precision***)或者**查准率**，要尽可能的准确判断出哪些人确实患病，也即预测器的策略应尽可能的严格，尽可能地保证被判断阳性的人确实患病，尽可能避把正常人划分为患者。

另一方面，我们还关心在所有确实患病的人中有多大比例的人是被预测器预测出来了的，我们不希望有患者没有被检测出来，这一比例称之为**查全率**或者**召回率**(***recall***)

$recall$和$precision$的矛盾关系。$precision$描述了预测器是否能够准确判断出阳性案例，为了提高$precision$，则仅在具有较大把握时才会将某一样本判断为阳性。$recall$描述了预测器是否能够把所有的阳性病例找出来，因此策略是当样本表现出一定的可能性就判定为阳性，假想一个极端例子，预测器把所有样本均预测为阳性，此时必然有$recall = 1$但$precision$就会很低。

因此，我们需要依据问题实际来决定是更看重P或者更看重R。另一方面，在一个实际问题中，或许可以使用1个数字或图像来帮助选择哪一个模型。

混淆矩阵(confusion matrix)

对于二分类问题，依据预测结果与真实结果组合有四种情况，即真阳性(True Positive), 假阳性(False Positive), 真阴性(False Negative), 假阴性(False Negative).通常，我们把占比较低的称为阳性或正例标记为1，而占比较高的称为阴性或反例标记为0.

表 2.1 分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

$$precision = \frac{TP}{TP + FP} = P$$

$$recall = \frac{TP}{TP + FN} = R$$

为此，定义一个F_1值，它是P和R的调和平均。

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

可以使用F_1值作为评价指标，当然也可以求加权调和平均F_β，是F_1的推广。

PR曲线

很多情形下我们可根据学习器的预测结果对样例进行排序，排在前面的是学习器认为"最可能"是正例的样本?排在最后的则是学习器认为"最不可能"是正例的样本.按此顺序逐个把样本作为正例进行预测，则每次可以计算出当前的查全率、 查准率以查准率为纵轴、查全率为横轴作图 ， 就得到了查准率 查全率曲线，简称 P-R曲线。

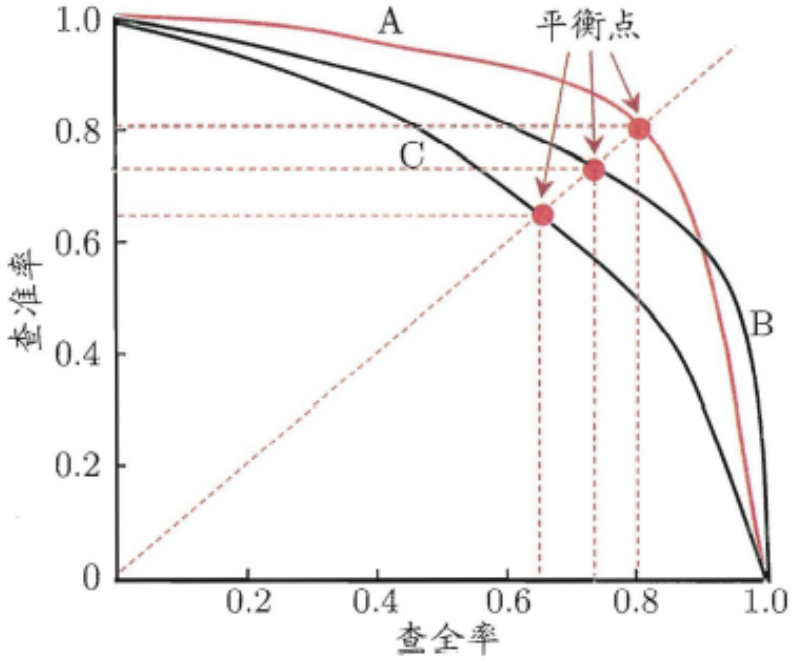


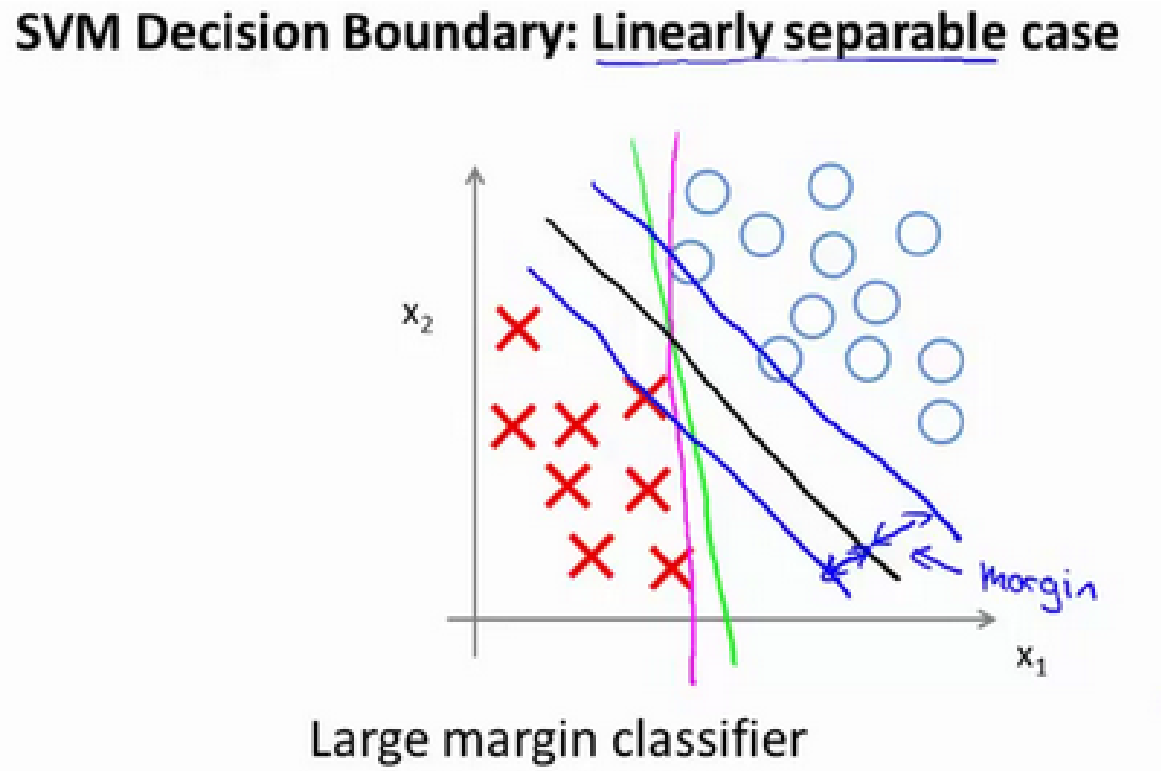
图 2.3 P-R曲线与平衡点示意图

Support Vector Machines(支持向量机)

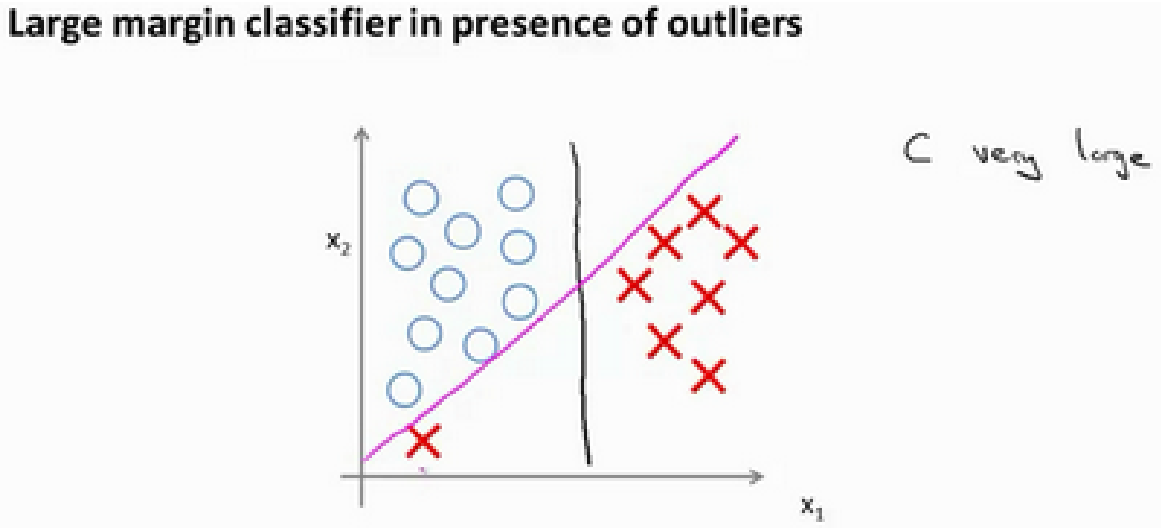
支持向量机的直观理解--他要做什么?

支持向量机，这个名字并不能望文生义，也就是很难从字面上理解这个概念到底是在描述的是什么。

首先，他是用于监督学习的分类任务中的算法。其次，以二分类任务为例，假设两个类是可以线性划分的，也即，假设样例只有两个特征，将每个样本点绘制在二维平面上，支持向量机讨论的是找到一个“最合适的”直线，这一直线能够很好的划分两类数据，并且“最合适”的意思是这一直线（超平面）离两类样本的间隔最大，而一类样本相对于一个超平面的距离自然是该类中距离该超平面最近的点对应的间隔，也即要求最短间隔最大，直观上看，直线应恰好位于样本点的中央。



如上图，支持向量机要做的就是找到这个黑色的决策边界，使得当有新的示例加入时，这一个决策能够尽可能正确的工作。但是，**当面对异常值时，支持向量机应当具备这样的策略，不至于太差，主要是正则化的使用**。如下图所示，支持向量机算法应该能够仍然选择黑色的决策边界，而不是洋红色那一个。



从线性回归模型开始讨论支持向量机的构造--进一步理解支持向量机

线性回归模型(Linear Regression Model)是使用特征的线性组合来对连续型的标记值进行拟合，这个线性组合被称为**假设函数**$h_\theta(x) = \theta^T x = \omega^T x + b$，LRM的目标是尽可能的使$h_\theta(x)$与$y$接近，为此定义了**代价函数**，它表征了$h$与$y$的差距，其数学表现形式是均方差，也即$J = \frac{1}{m}\sum(y-h)^2$，这

里m表示样本数量。为了使J最小，一种方法是**梯度下降算法**，这是一种迭代算法，核心在于确定向量θ变化的方向（也即导数变化的负方向，梯度）以及每一次迭代的步长（**学习率**）。另一方面，为了拟合非线性变化的标记值y，引入了**多项式特征**，意思是将原有特征的多项式形式作为新的特征，例如x_1x_2或者x_1^2等。这样做就可以拟合非线性的标记值，但另一方面当幂次过高时，可能出现的情况是假设函数可以很好的拟合已知数据集（**训练集**），但对于新的数据点（**验证集**）误差就很大，出现了**过拟合**现象，假设函数学习到了训练集的局部数据特征，这也被称之为**高方差问题(High Variance)**，方差描述的是假设函数对抗数据扰动的能力，这里出现的新数据使得代价函数值显著增大，采取的对策是在代价函数中**加入正则化项**$\lambda\sum\theta_j^2$，基本思想是由于要使J尽可能小，加入这一惩罚项后，θ也会对应的减小，而假如高幂次项的系数更小了（极端假设为0，相当于没有高幂次项），这也使得曲线更平滑，不至于过拟合。观察到正则化项前的系数λ，λ太大则会引发另一个问题，也即**欠拟合**，或者**高偏差（bias）问题**，这表明假设函数本身的学习能力较差，此时可以减小正则化系数，增加更多的有效特征等。

那么接下来考虑分类问题，例如二分类问题，可以先这样想，此时仅仅是y的值不再是连续变化在实数域取值了，变成了两个值，例如1（正类）或者0（负类）（当然也可以选择任何其他的值，例如1和-1）。此时假设我们仍然LRM来对样本进行拟合，或许我们也可以得到一条直线，似乎也能把样本分开？（或许？）。但是θ^Tx可能很大或者很小，因此引入sigmoid函数，它可以进行$\mathbb{R}\to(0,1)$的映射，函数形式为$sigmoid(z)=\frac{1}{1+e^{-z}}$，此时就得到**对率回归模型**(Logistic Regression Model)的假设函数$h=sigmoid(\theta^Tx)$。此时h可以看作样本为正例的概率，h不小于0.5时预测样本x为正例否则负例。另一方面假设样本为正例此时$y=1$如果h接近0则赋予较大的代价值，接近1则赋予较小的代价值，因此代价函数变为$\sum y(-\ln h)+(1-y)(\ln(1-h))$。巧妙地是，sigmoid函数的数学性质很好，这一函数是可以优化求解的，形式上与线性回归相同。

在对率回归模型中，对x分类的依据是h是否大于0.5，依据sigmoid函数的性质，这等价于判断$\theta^Tx=\omega^Tx+b$是否大于0。对于训练好的参数ω和b，$\omega^Tx+b=0$就形成了决策边界。回到svm的指导思想，他是要最近距离最大。对于任意一个样本空间的点x，该点与决策平面的距离是

$$d=\frac{|\omega^Tx+b|}{|\omega|}$$

。为方便讨论，设负类标记值为-1，正类为1，则当决策边界正确分类时，若$\omega^Tx+b<0$则$y=-1$，也即他们同号。因此

$$d=\frac{y(\omega^Tx+b)}{|\omega|}$$

，遍历所有样本点得到$dmin$，因此问题转化为

$$max\to dmin$$

s.t.

–

$$d = \frac{|\omega^T x + b|}{|\omega|} >= dmin$$

同时，由于ω和b总是可以同步缩放的，此时不影响超平面的方程。不妨设当样本的某些点满足s.t.的等号条件时有$1 = \omega^T x + b$或$-1 = \omega^T x + b$，这些点即为**支持向量**。问题转化为:

$$max \frac{1}{||\omega||}$$

s.t.

$$y_i(\omega^T x_i + b) >= 1(i = 1, ..., m)$$

从代价函数的变化看支持向量机

对率回归模型的代价函数基本形式是:

$$J = -\frac{1}{m}\sum_{i=1}^{m}(y_i * ln(h) + (1 - y_i) * ln(1 - h)) + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

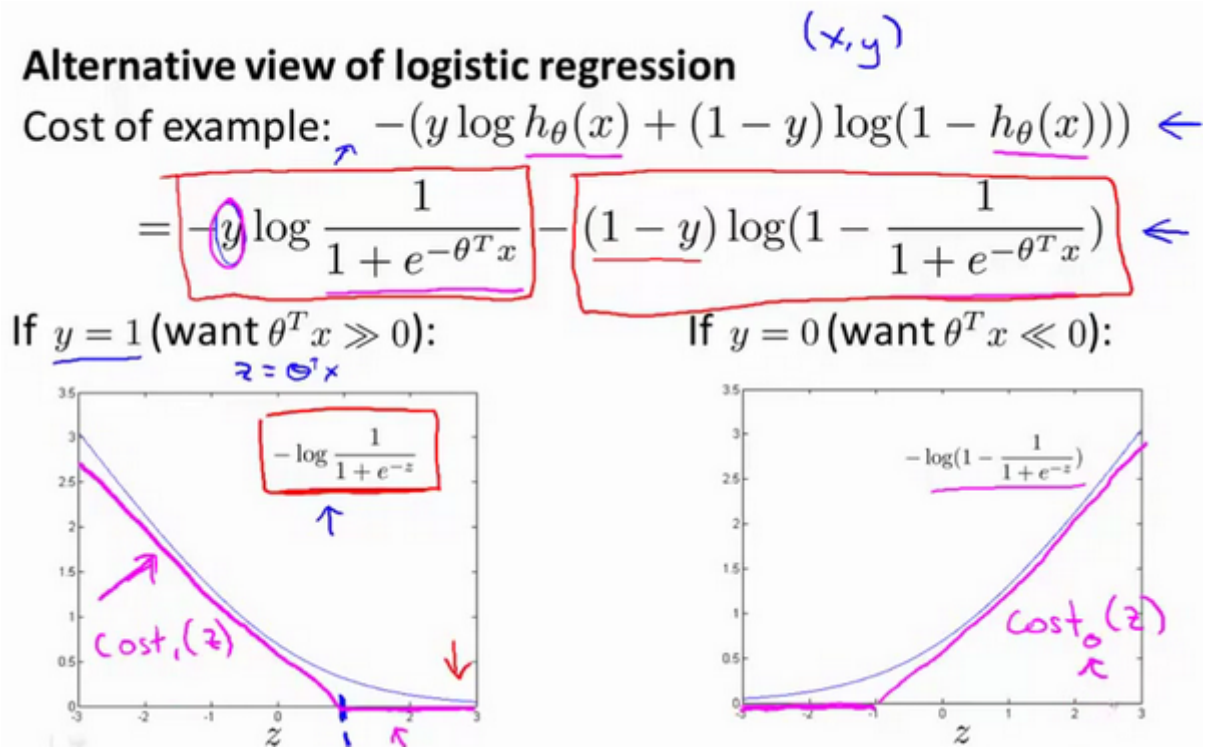
不考虑正则化项, 对于单个样本

$$J = y * (-ln\frac{1}{e^{-\theta^T x} + 1}) + (1 - y) * (1 - ln\frac{1}{e^{-\theta^T x} + 1})$$

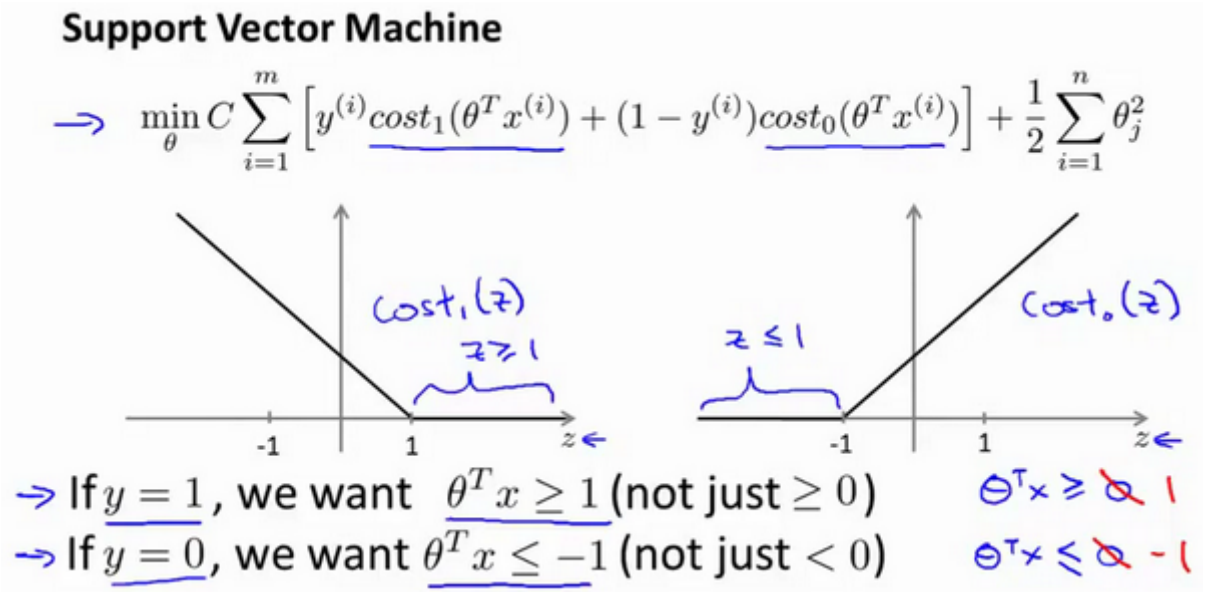
也即

$$J = y * cost_1(\theta^T x) + (1 - y) * cost_0(\theta^T x)$$

$y = 1$时$J = cost_1$而$y = 0$时$J = cost_0$，分别作出$cost_0$和$cost_1$的函数图像如下:



洋红色的曲线即为SVM对应的两个$cost$函数，可见SVM将其进行了简化，但同时又与对率函数的很接近。并且我们对代价函数进行了一定的抽象，同时在SVM中，不再使用$A+\lambda B$的形式，而是使用$C*A+B$的形式，同时不考虑m这个值，就有了



观察到，当样本分类为正类时，必须满足条件$\theta^T x >= 1$否则必须满足$\theta^T x <= -1$，因而支持向量机有时被称为**大间距分类器**，此时将得到一个“有趣的”决策边界。

如何理解SVM的决策边界具有这样的大间隔

考虑SVM的总体代价函数：

$$C\sum_{i=1}^{m}[ycost_1 + (1-y)cost_0] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

为了便于理解，尽量考虑理想化情况，假设C无限大，为了优化J，就必须满足有$y=1 \rightarrow cost_1=0$ 而 $y=0 \rightarrow cost_0=0$使得前面的项无限接近于0，因此优化目标就变为了$minimize \rightarrow \sum_{j=1}^{n}\theta_j^2$，从向量角度的观点，也即是要使得$||\theta||$最小化，此即为$\theta$的模长。优化问题表征如下：

SVM Decision Boundary

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

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

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

因此支持向量机做的全部事情，就是极小化参数向量范数的平方，或者说长度的平方

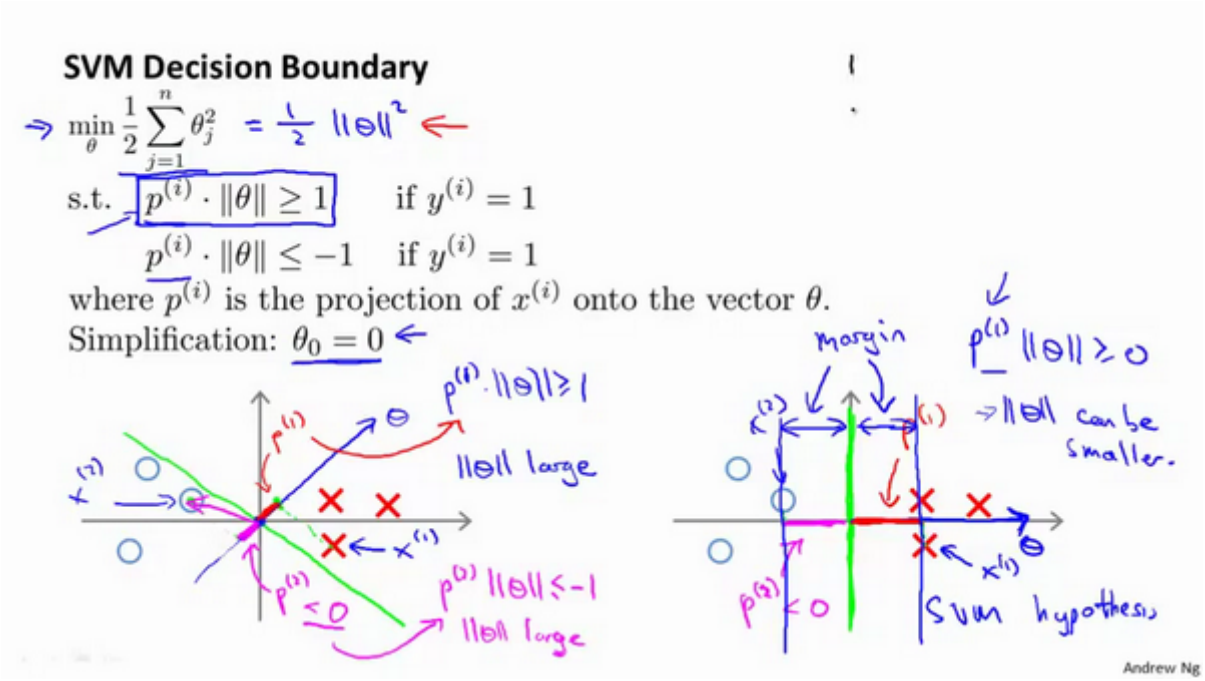
向量内积的讨论：$(u,v)=u^Tv=v^Tu$。因此θ^Tx也即θ向量与样本x向量的内积，不考虑截距θ_0，假定只有两个特征，这就变为了两个平面向量的点乘。

$$\theta^Tx=\theta.x=||\theta|| * ||x|| * cost < \theta, x >$$

接着引入**投影**的概念，他表示的是一个向量在另一个向量上的投影，是一个标量，具备正负号，取决于两个向量夹角，就有了：

$$\theta^Tx=\theta.x=Proj_\theta^x * ||\theta||$$

于是一切变得明晰起来，如下图：



图中，由于截距项为0，因此决策边界总是过原点，同时，决策边界上的点总满足$\theta^Tx=0$，因此决策边界与θ向量垂直。以正例中位于第四象限的样本x为例，在满足$Proj_\theta^x * ||\theta|| >= 1$的条件下

（也即可以正确划分样本的条件下），为了减小θ的模长，则x在θ方向上的投影长度越大越好，图中左侧给出了一个坏的边界绿色线条，作出θ方向，再给出投影，也即红色短线的长度，比较小，因而是个坏边界。而如果像右侧取y轴作为决策界限，此时θ方向为x轴正方向，样本在θ方向上的投影都比较大。再次见证**SVM是一个大间隔分类器**。

总结起来，SVM向量机是一个大间隔分类器，简化了逻辑回归中的代价函数，所做的事情就是在保证大间距也就是正确分类的条件下最小化参数向量的模长，换句话说，就是总体来看，样本在参数向量方向上的投影最大。

无监督学习算法(Unsupervised Learning Algorithm)

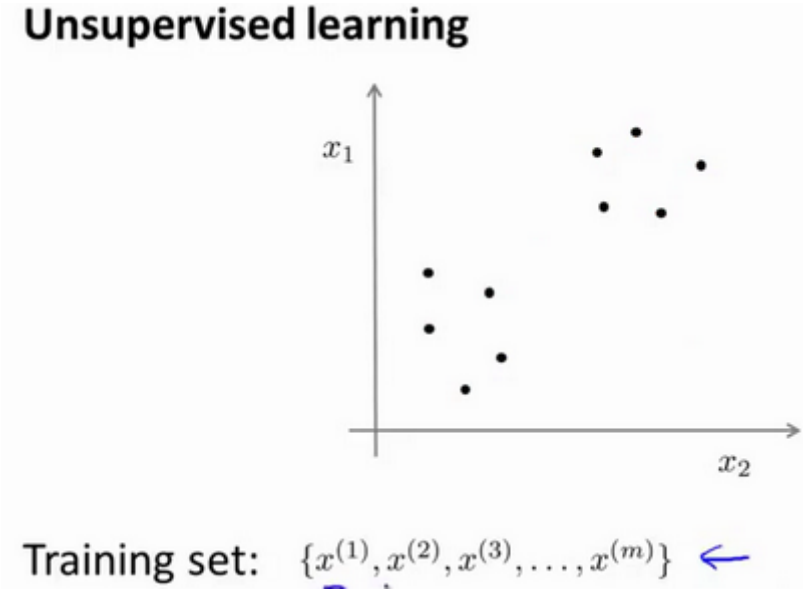
这是一个激动人心的时刻，此前，我们学习的都是监督学习的算法，他们处理的数据都是带标签的数据，例如处理带连续值标签的线性回归算法以及用于典型的离散值标签分类问题的对率回归、神经网络以及支持向量机算法等等。而在无监督学习的任务中，我们拿到的将是一组没有任何标签的数据。

$$Trainingset: x^{(1)}, x^{(2)}, ..., x^{(m)}$$

2-1、聚类分析(Clustering)

2-1（1）引言

聚类分析算法是一类典型的无监督学习算法，他试图从一系列的无标签训练数据中发现数据的内在结构（聚簇划分）。如下图，我们可以让聚类算法帮助我们画出两个圈，将数据集分为两个簇(cluster)



2-1（2）应用场景举例

市场划分(Market Segementation)：也许你在数据库中存储了许多客户的信息，而你希望将他们分成不同的客户群，这样你可以对不同类型的客户分别销售产品或者分别提供更适合的服务。
社交网络分析(Social network analysis)：简单来说就是依据人的社交活动划出一个个的社交圈，比如说：你经常跟哪些人联系，而这些人又经常给哪些人发邮件，由此找到关系密切的人群。
计算机集群管理(Organize computing clusters)：希望使用聚类算法来更好的组织计算机集群，或者更好的管理数据中心，有哪些计算机经常协作工作。
星系形成(Astronomical data analysis)：最后，我实际上还在研究如何利用聚类算法了解星系的形成。然后用这个知识，了解一些天文学上的细节问题。

2-1（3）K-均值算法的步骤

K-均值是最普及的聚类算法，算法接受一个未标记的数据集，然后将数据聚类成不同的组。
K-均值算法是一个迭代算法
第一步，在样本空间中随机选取K个点，称之为 ***聚类中心(cluster centroids)*** ；

第二步，对于数据集$\{x^{(i)}_{i=1\ldots m}\}$中的每一个样本点，计算样本点与K个样本中心的距离$\{d_1^{(i)}, d_2^{(i)}, ..., d_K^{(i)}\}$，选取其中最小的距离$d_k^{(i)}$，将点$x^{(i)}$与聚类中心$k$关联起来，也即属于第k类，这一步为聚类。

第三步，对于第二步形成的K个聚类，得到新的聚类中心为该类所有样本点求均值得到的点

重复第二步和第三步直到新的聚类中心与原聚类中心基本重合。

2-1（4）k-均值算法的优化目标

由第3小节关于K-均值算法步骤的分析可知，算法的优化目标是样本点与所属聚类的聚类中心的距离的和的均值。代价函数（又称 ***畸变函数distortion function*** ）。

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

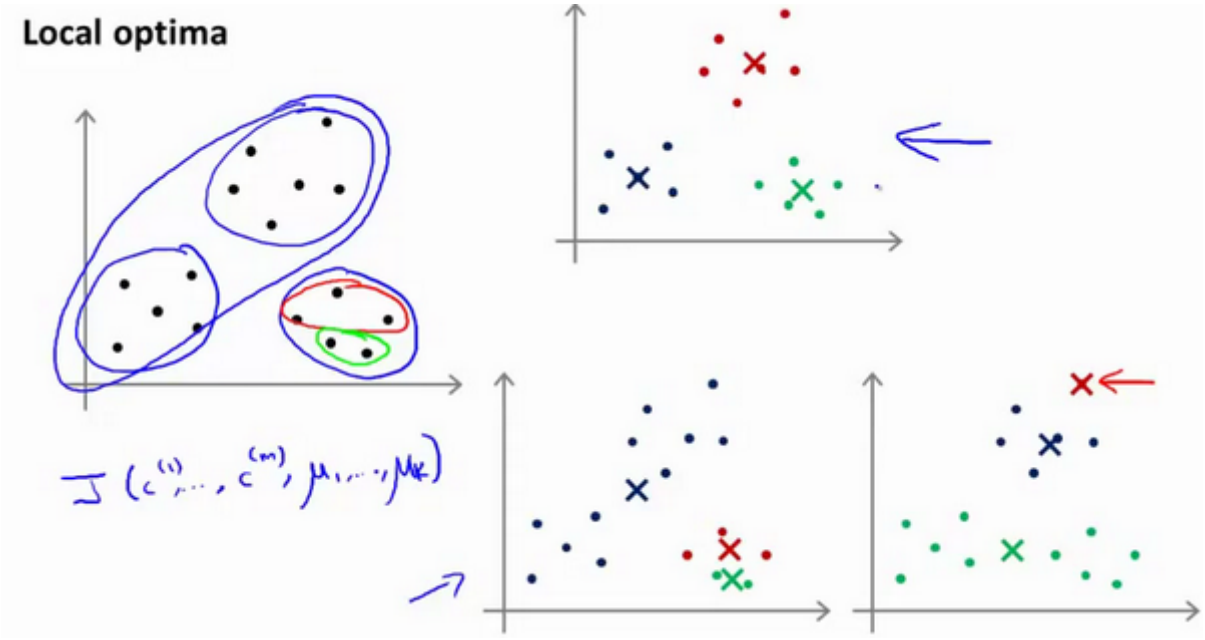
其中$\mu_{c^{(i)}}$代表与$x^{(i)}$最近的聚类中心点。 我们的的优化目标便是找出使得代价函数最小的 $c^{(1)},c^{(2)},...,c^{(m)}$和$\mu^1,\mu^2,...,\mu^k$：

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

2-1（5）k-均值算法的细节问题

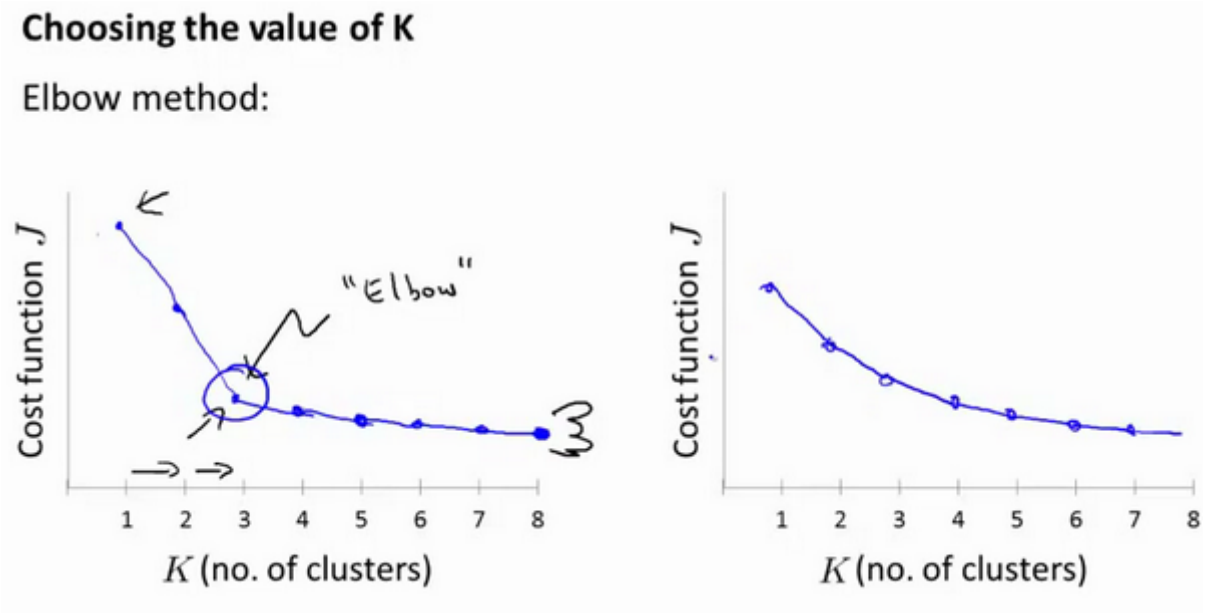
如何得到随机K个聚类中心

方法是随机选取K个训练示例作为聚类中心，其次K-均值的一个问题在于，它有可能会停留在一个局部最小值处，而这取决于初始化的情况。方法是多次运行K-均值算法，每一次都重新进行随机初始化，最后再比较多次运行K-均值的结果，选择代价函数最小的结果。这种方法在较小的时候（2--10）还是可行的，但是如果K较大，这么做也可能不会有明显地改善。



如何选取K的值

一种考量是“肘部法则elbow rule”:

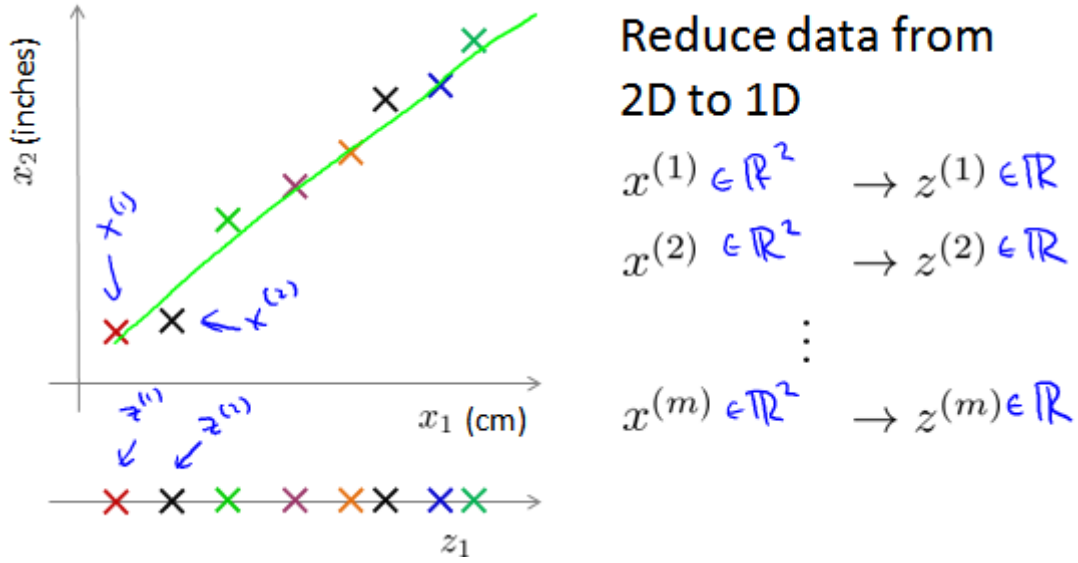


但通常更常见的是基于实际需要的考量
例如，我们的 T-恤制造例子中，我们要将用户按照身材聚类，我们可以分成3个尺寸S M L:，也可以分成5个尺寸XS S M L XL，这样的选择是建立在回答“聚类后我们制造的T-恤是否能较好地适合我们的客户”这个问题的基础上作出的。

2-2、降维(Dimensionality Reduction)

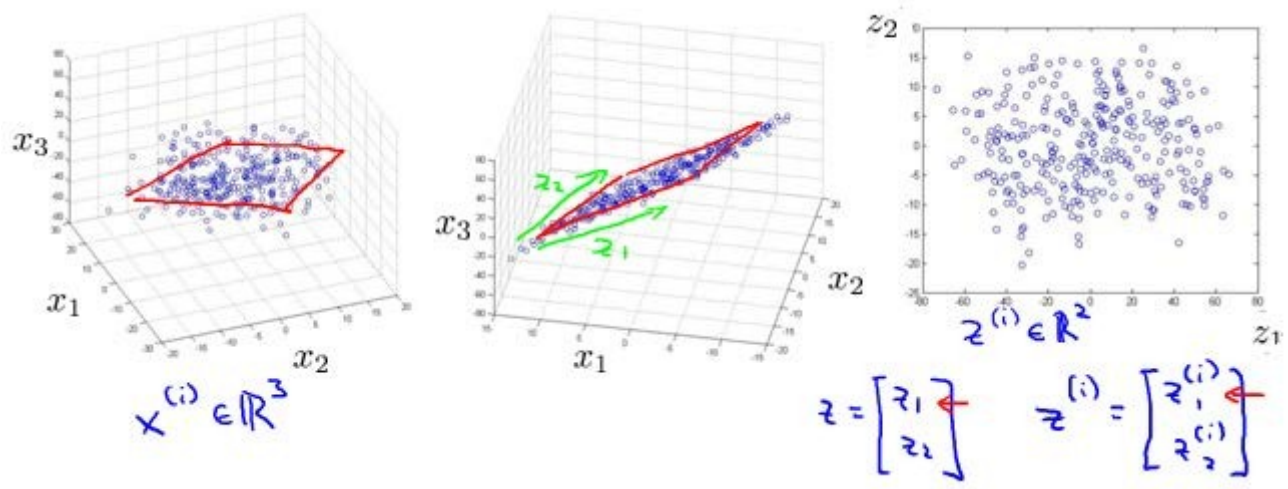
2-2（1）降维的动机

首先，让我们谈论降维是什么。作为一种生动的例子，我们收集的数据集，有许多特征，我绘制两个在这里。假设我们未知两个的特征: x_1是用厘米表示的高度；x_2是用英寸表示同一物体的高度。所以，这给了我们高度冗余表示：



因而，我们希望将这个二维的数据降至一维。这样的处理过程可以被用于把任何维度的数据降到任何想

要的维度，例如将1000维的特征降至100维。



一方面，对数据进行降维可以压缩数据量，这被称为数据压缩(data compressing)，它使得可以占用较少的计算机内存或磁盘空间，但它也让我们加快我们的学习算法。另一方面，由于维度的减小，数据可视化也成为了可能。

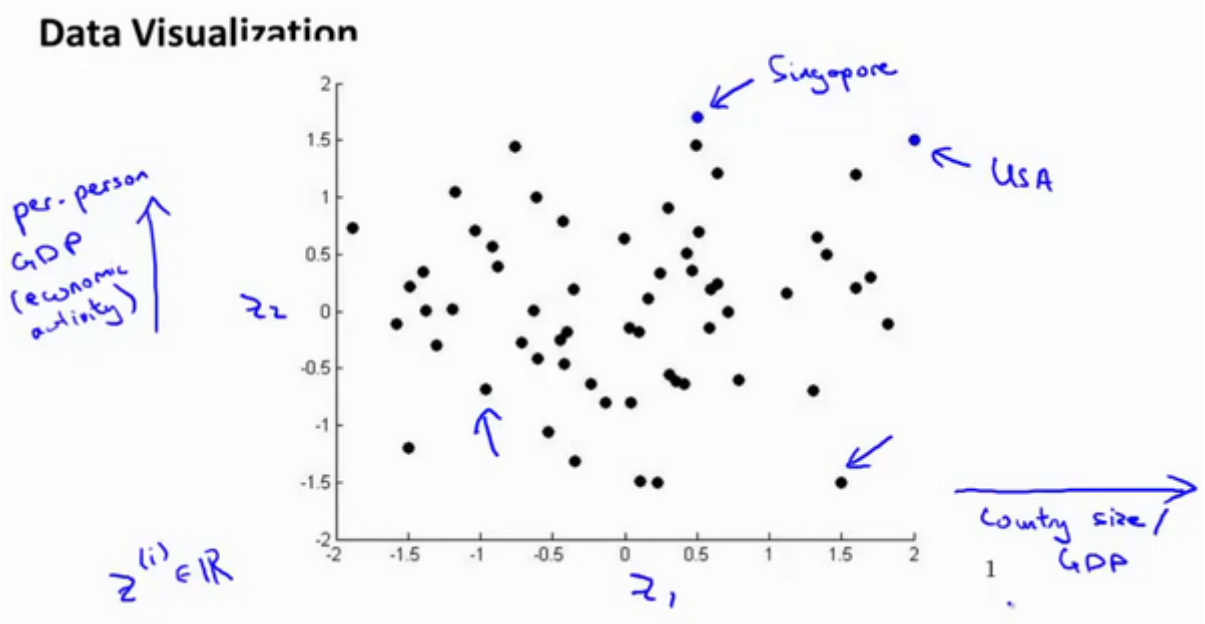
Data Visualization

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

[resources from en.wikipedia.org]

Andrew Ng

假使我们有有关于许多不同国家的数据，每一个特征向量都有50个特征（如GDP，人均GDP，平均寿命等）。如果要将这个50维的数据可视化是不可能的。使用降维的方法将其降至2维，我们便可以将其可视化了。



2-2（2）降维算法--PCA(Principal Component Analysis)

主成分分析(PCA)是最常见的降维算法
在PCA中，我们要做的是***找到一个方向向量（Vector direction），当我们把所有的数据都投射到该向量上时，我们希望投射平均均方误差能尽可能地小***。方向向量是一个经过原点的向量，而投射误差是从特征向量向该方向向量作垂线的长度。

PCA问题的描述：将n维数据点降到k维，目标是找到k维空间的一组基$u^{(1)}, u^{(2)}, ..., u^{(k)}$使得总体投影最小。

PCA算法步骤：
第一步是**均值归一化**。我们需要计算出所有特征的均值μ_i，然后作差$x_i - \mu_i$。如果特征是在不同的数量级上，我们还需要将其除以标准差δ

第二步计算**协方差矩阵(covariance matrix)Σ:

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

每一个样本点$x^{(i)}$都是一个n维的列向量

第三步是计算协方差矩阵Σ的**特征向量(eigenvectors)**:
编程时可使用奇异值分解(Singular Value Decomposition)求解 `[U, S, V] = svd(Sigma)`

Principal Component Analysis (PCA) algorithm

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

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

最后取矩阵U的前k个列向量得到U_{reduce}，他是$n \times k$维的矩阵，于是对于每一个样本点$x^{(i)}$，计算$z^{(i)} = U_{reduce}^T x^{(i)}$，它是$k$维的向量，此即为样本点投影到新的$k$维空间的坐标。

2-2（3）PCA算法的细节问题

如何选择主成分数量k

训练集方差: $Var_{train} = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)}||^2$
投影误差: $Var_{proj} = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - U_{reduce} z^{(i)}||^2$
则相对误差定义为: $\alpha = \frac{Var_{proj}}{Var_{train}}$

回忆SVD分解中有: `[U, S, V] = svd(sigma)`。其中S是一个对角矩阵，也即S除主对角线外其他元素均为0。因而一个更简单的解法是:
$\alpha = 1 - S_k / S_n$。其中S_i表示S主对角线前i个元素之和。

主成分分析法的应用建议

以下场合不建议使用PCA算法 **非必要不使用**

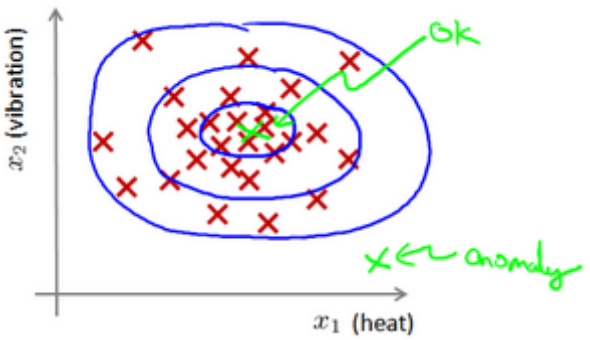
一个常见错误使用主要成分分析的情况是，将其用于减少过拟合（减少了特征的数量）。这样做非常不好，不如尝试正则化处理。原因在于主要成分分析只是近似地丢弃掉一些特征，它并不考虑任何与结果变量有关的信息，因此可能会丢失非常重要的特征。

另一个常见的错误是，默认地将主要成分分析作为学习过程中的一部分，这虽然很多时候有效果，最好还是从所有原始特征开始，只在有必要的时候（算法运行太慢或者占用太多内存）才考虑采用主要成分分析。

异常检测(Anomaly Detection)

这一概念或者关于异常检测这一系列问题的方法系列来源于工业品制造中，想想你生产一批次的产品m个，对于每个产品x_i，有d个指标可以量化产品的质量，假设现在生产了一个新的该产品，测得这d个量化指标，那么如何判断这个产品是不是合格呢或者是不是不合格呢?

假设d=2, 将已知的m个数据绘制在坐标轴上, 同时也把新的数据点绘制在图中



那么，一个数据点落在蓝色圈内的概率很大，是合格品，落在圈外的概率很低，一般情况下不会发生，因此以很大概率判断这是个异常品。

异常检测主要用来识别欺骗。例如在线采集而来的有关用户的数据，一个特征向量中可能会包含如：用户多久登录一次，访问过的页面，在论坛发布的帖子数量，甚至是打字速度等。尝试根据这些特征构建一个模型，可以用这个模型来识别那些不符合该模式的用户。

再一个例子是检测一个数据中心，特征可能包含：内存使用情况，被访问的磁盘数量，CPU的负载，网络的通信量等。根据这些特征可以构建一个模型，用来判断某些计算机是不是有可能出错了。

高斯分布

高斯分布或者正态分布，上述问题描述了一个具有两个量化指标的产品检测问题，这两个指标相互独立，各自都满足正态分布，这样的检测类似于检测指标是否落在置信区间内。为了形式化的定义这样的异常检测算法，我们有：

通常如果我们认为变量 x 符合高斯分布 $x \sim N(\mu, \sigma^2)$则其概率密度函数为： $p(x, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ 我们可以利用已有的数据来预测总体中的μ和σ^2的计算方法如下：

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

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

算法流程

异常检测算法：

对于给定的数据集 $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$，我们要针对每一个特征计算 μ 和 σ^2 的估计值。

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

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

一旦我们获得了平均值和方差的估计值，给定新的一个训练实例，根据模型计算 $p(x)$：

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

当$p(x) < \varepsilon$时，为异常。

算法为每个特征建立了高斯分布，并将评估函数记为他们的积$p(x)$，用$p(x) < \epsilon$时就表明x属于异常值。当特征数量为2，特征间没有相关性时，它是一个钟形图像。

异常检测和监督学习的区别

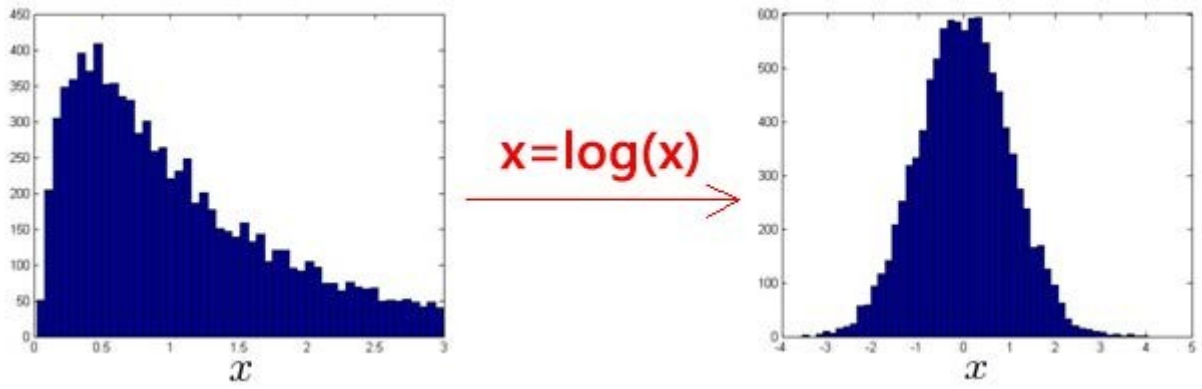
异常检测某种角度上似乎与监督学习十分类似，也即已知的样品假定都是正品，没有残次品，需要将新的样本分类，也即是不是正品。但他们使用的方法和场景大有不同，监督学习适用于能够获取充足量的不同类的样本，例如垃圾邮件识别，我们可以得到大量的垃圾邮件用于学习，通常方法是使用对率回归、神经网络、支持向量机等等算法，他们都是迭代算法，试图得到一个决策边界。

异常检测则面对的是这样的场景，在通常情况下，样本总是正品（阴性），出现残次品的情况很少，因而无法学习到残次品的特征，转而对正品的特征进行建模，主要依据（多元）正态分布以及置信区间检验等等的数学基础

异常检测	监督学习
非常少量的正向类（异常数据），大量的负向类	同时有大量的正向类和负向类
许多不同种类的异常，非常难根据非常 少量的正向类数据来训练算法。	有足够多的正向类实例，足够用于训练算法， 未来遇到的正向类实例可能与训练集中的非常近似。
未来遇到的异常可能与已掌握的异常、非常的不同。 例如： 欺诈行为检测 生产（例如飞机引擎） 检测数据中心的计算机运行状况	例如：邮件过滤器 天气预报 肿瘤分类

特征的设计与选择

异常检测假设特征符合高斯分布，如果数据的分布不是高斯分布，异常检测算法也能够工作，但是最好还是将数据转换成高斯分布，例如



使用多元高斯分布

协方差矩阵表示特征间的相关性

我们首先计算所有特征的平均值，然后再计算协方差矩阵：

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

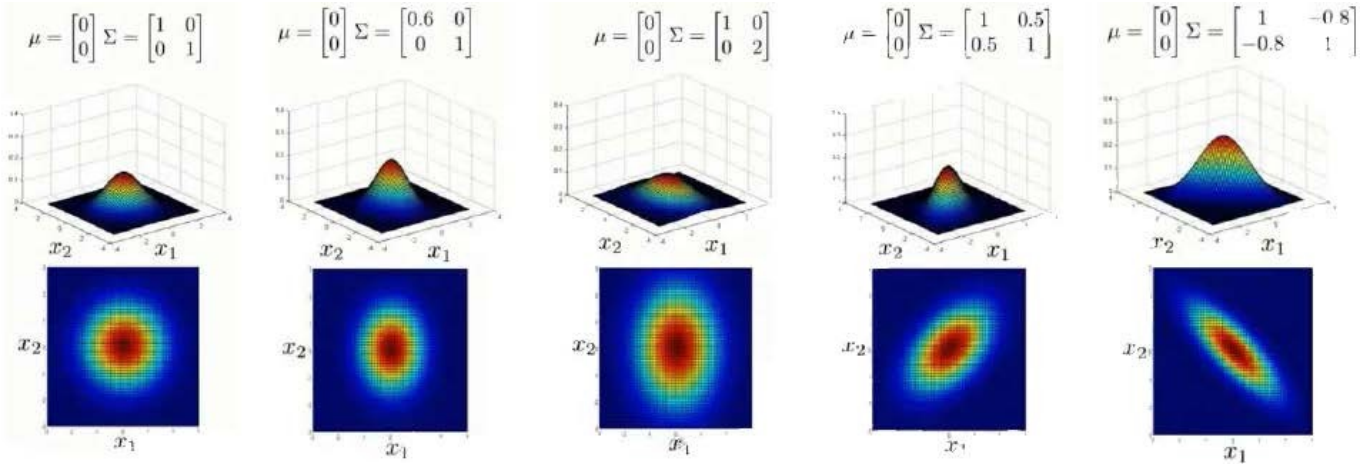
$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$

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

注:其中μ 是一个向量，其每一个单元都是原特征矩阵中一行数据的均值。最后我们计算多元高斯分布的$p\left(x\right)$:

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

1. 是一个一般的高斯分布模型
2. 通过协方差矩阵，令特征1拥有较小的偏差，同时保持特征2的偏差
3. 通过协方差矩阵，令特征2拥有较大的偏差，同时保持特征1的偏差
4. 通过协方差矩阵，在不改变两个特征的原有偏差的基础上，增加两者之间的正相关性
5. 通过协方差矩阵，在不改变两个特征的原有偏差的基础上，增加两者之间的负相关性



Recommender System 推荐系统

A big idea

自动的学习一系列的特征features

事例：电影评分预测

假定评分用星级0-5表示

研究的问题是：给出n_u n_m $r(i,j)$ $y^{(i,j)}$，我们能否给出那些未被评分的值

content-based recommendation

特征描述，情爱内容与动作内容

这类似于线性回归

collaborate filter 协同过滤

如何得到电影的这些特征呢？或者简单一点，我想到了这些特征，怎么得到各个电影的这些特征的值？更进一步，假设我们知道了用户评价，如何推测特征的值？

例如Alice喜欢romance电影，她给1号电影评分高，推测1号电影更可能是romance.

协同过滤算法的问题与改进

同时minimize x和 θ

相关推荐

也即预测用户可能喜欢（评分高）的电影
Low rank matrix factorization(低秩矩阵分解)

未知用户

均值归一化