机器学习

斯坦福《机器学习》公开课笔记

Probability theory provides a framework for expressing such uncertainty in a precise and quantitative manner, and decision theory allows us to exploit this probabilistic representation in order to make predictions that are optimal according to appropriate criteria.

概率理论提供了用精度和量化的方法来描述随机现象，而决策理论则使我们能够根据概率做出适当预测。

关丹辉 chrispher2012@gmail.com

数据，为梦想而生 | FOR THE DREAM

仅以此笔记，献给我们奋斗过的青春

目录

1. 机器学习的动机与应用 4
   1.1. 机器学习定义 4
   1.2. 基础知识需求与常见应用 4
   1.3. 机器学习算法分类 4
   1.4. 监督式学习 5
2. 监督学习的应用与梯度下降 6
   2.1. 监督学习 6
   2.2. 梯度下降 6
      2.2.1. 最小均方算法 6
      2.2.2. 标准方程组推导 7
      2.2.3. 概率解释 8
3. 欠拟合与过拟合概念 9
   3.1. 欠拟合与过拟合概念 9
   3.2. 局部加权线性回归 9
   3.3. 回归模型的概率解释 11
   3.4. Logistic 回归模型 12
   3.5. 感知器学习模型 14
4. 牛顿法与广义线性模型 15
   4.1. 牛顿法求解最优值 15
   4.2. 广义线性模型 16
      4.2.1. 指数族分布 16
      4.2.2. 广义线性模型结构 17
      4.2.3. Softmax 回归 18
5. 生成学习算法 22
   5.1. 生成学习算法引入 22
   5.2. 高斯判别模型 22
   5.3. GDA VS logistic 回归 24
   5.4. 朴素贝叶斯 24
      5.4.1. 朴素贝叶斯模型 25
      5.4.2. 拉普拉斯平滑 27
6. 朴素贝叶斯算法 29
   6.1. 文本分类模型 29
   6.2. 神经网络模型 30

6.3. 支持向量机 32
6.3.1. 直觉上的间隔 32
6.3.2. 符号表示 33
6.3.3. 几何间隔 33
7. 最优间隔分类器 36
7.1. 最优间隔分类器 36
7.2. 拉格朗日对偶 37
7.3. 最优间隔分类器 39
8. 顺序最小化算法 43
8.1. 核方法 43
8.2. 正则化和线性不可分 46
8.3. 坐标上升法 48
8.4. 序列最小算法(SMO) 49
9. 经验风险最小化 52
9.1. 偏差和方差的权衡 52
9.2. 预备知识 53
9.3. 有限情况下的$\mathcal{H}$ 54
10. 特征选择 58
10.1. 无限情况下的$\mathcal{H}$ 58
10.2. 交叉验证(Cross validation) 61
10.3. 特征选择 62
11. 贝叶斯统计正则化 65
11.1. 贝叶斯统计和正则化 65
11.2. 附录：过拟合思考 66
11.3. 感知器和最大间隔分类 67
11.4. 使用机器学习的一些建议 67
11.4.1. 调试学习算法 67
11.4.2. 误差分析 70
12. K-means 算法 71
12.1. k-means 聚类算法 71
12.2. 高斯混合模型和最大期望算法 72
12.3. 最大期望算法一般化 74
12.3.1. Jensen 不等式 74
12.3.2. EM 算法 75

13. 高斯混合模型 78
13.1. 高斯混合模型回顾 78
13.2. 因子分析 79
13.2.1. 限制 Σ 80
13.2.2. 边际和条件高斯分布 80
13.2.3. 因子分析模型 81
14. 主成分分析法 83
14.1. 因子分析 83
14.1.1. EM 算法求解因子分析 83
14.2. 主成分分析 85
15. 奇异值分解 90
15.1. 潜在语义索引 90
15.2. 奇异值分解 90
15.3. 独立成分分析 90
15.3.1. 独立成分分析引入 90
15.3.2. ICA 的不确定性(ICA ambiguities) 91
15.3.3. 密度函数和线性变换 92
15.3.4. ICA 算法 93
16. 马尔可夫决策过程 95
16.1. 概述 95
16.2. 马尔可夫决策过程 95
16.3. 值迭代和策略迭代 97
16.4. MDP 的一个学习模型 98
17. 离散与维灾难 100
17.1. 连续状态下的 MDP 100
18. 线性二次型调节控制 103
18.1. 使用一个模型或 103
18.2. 拟合值迭代 104

1. 机器学习的动机与应用

1.1. 机器学习定义

广义的来说，机器学习(wiki)是在不明确编程的情况下赋予机器学习的能力的一门科学。而机器学习经典的定义是 Mitchell 的《机器学习》中的一段：A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E。(对于某类任务 T 和性能度量 P，如果一个计算机程序在 T 上以 P 衡量的性能随着经验 E 而自我完善，那么我们称这个计算机程序在从经验 E 学习。)

举例子：垃圾邮件分类。任务 T 是判断一封邮件是否是垃圾邮件，性能度量 P 是识别的准确率，经验 E 是我们收集到的垃圾邮件和非垃圾邮件。我们的目标是设计一个算法，该算法能够通过学习已有的数据，来准确的判断一封新的邮件是否是垃圾邮件。

一个完整定义的学习问题需要一个明确界定的任务、性能度量标准以及训练经验的来源。

1.2. 基础知识需求与常见应用

机器学习是近 20 多年兴起的一门多领域交叉学科，涉及概率论、统计学、逼近论、凸分析、算法复杂度理论等多门学科。

基础知识主要包括：①计算机科学的基本知识，包括基本的编程能力，知道算法复杂度等概念以及一些数据结构如队列、栈、二叉树等。②概率论与统计知识(达到本科要求)，如随机变量，期望等等。③基本的线性代数课程(本科阶段)，知道什么是矩阵以及矩阵特征向量等等。

机器学习已经有了十分广泛的应用，例如：数据挖掘、计算机视觉、自然语言处理、生物特征识别、搜索引擎、医学诊断、检测信用卡欺诈、证券市场分析、DNA 序列测序、语音和手写识别、战略游戏和机器人运用。

1.3. 机器学习算法分类

- 监督学习：从给定的训练数据集中学习出一个函数，当新的数据到来时，可以根据这个函数预测结果。监督学习的训练集要求是包括输入和输出，也可以说是特征和目标。训练集中的目标是由人标注的。常见的监督学习算法包括回归分析和统计分类。
- 无监督学习与监督学习相比，训练集没有人为标注的结果。常见的无监督学习算法有聚类。
- 半监督学习介于监督学习与无监督学习之间。

- 增强学习通过观察来学习做成如何的动作。每个动作都会对环境有所影响，学习对象根据观察到的周围环境的反馈来做出判断。

具体的机器学习算法有：

- 构造条件概率：回归分析和统计分类

  人工神经网络 决策树(Decision tree) 高斯过程回归

  线性判别分析 最近邻居法 感知器 径向基函数核

  支持向量机
- 通过再生模型构造概率密度函数(Probability density function)：

  最大期望算法(Expectation-maximization algorithm)

  graphical model：包括贝叶斯网和 Markov 随机场

  Generative Topographic Mapping
- 近似推断技术：

  马尔可夫链(Markov chain) 蒙特卡罗方法 变分法

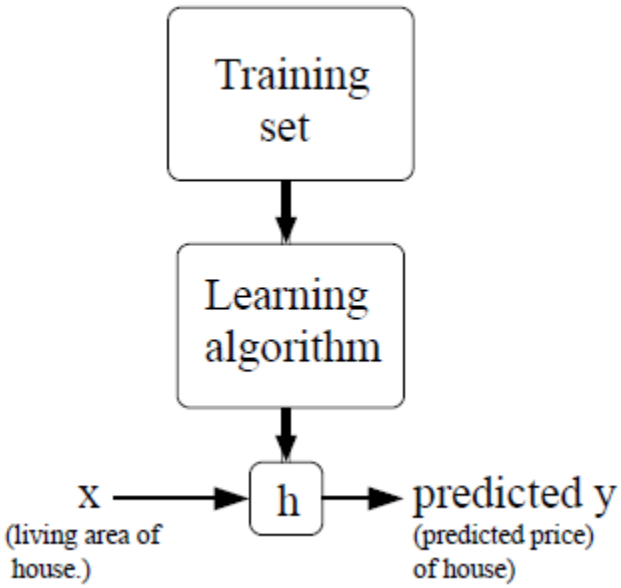
最优化(Optimization)：大多数以上方法，直接或者间接使用最优化算法。

1.4. 监督式学习

在监督式学习中，如果预测的结果是连续的变量，我们称之为回归问题(regression)，比如房价预测问题，房子的价格是连续变量，可以使用回归算法建模预测。如果预测的结果是离散变量，则称之为分类(classification)，比如判断病人的肿瘤是否是良性，结果有是或否，是离散变量，可以使用分类算法建模预测。

2. 监督学习的应用与梯度下降

2.1. 监督学习



如上图所示，监督学习:对于给定的训练集合，按照某一学习算法学习之后，得到一种好的假设(Hypotheses)用于预测新的数据。

2.2. 梯度下降

已知 m 组数据$(x_1,y_1)...(x_m, y_m)$，其中 x_i 是具有 n 维特征的向量，此外，我们设定 $x_i(0)=1$(即截距项)。我们做如下假设：

$$h(x) = \sum_{i=0}^{n} \theta_i * x_i = \Theta^T * x \text{ (此为回归模型的假设模型)}$$

对于给定的训练集合，如何选择最优的 θ 值(权重或参数)呢(这里的 x 是 n+1*m 维矩阵)？一个合理的方法是：至少在训练集合上，θ 会使预测值 h(x)越接近实际值 y 越好。因此，我们定义一个成本函数(cost function) J(θ)：

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

该成本函数使用误差的平方和，类似于普通最小二乘法(没有平均化误差平方和)。

2.2.1. 最小均方算法

给定的训练集合，如何选择最优的 θ 值使 J(θ)？这里我们采用梯度下降法。梯度下降法的基本思想是在起始随机得到 θ 值后，之后每次更新 θ 值的方式如下：

$$\theta_j := \theta_j - \alpha * \frac{\partial}{\partial(\theta)} J(\theta)$$(其中 α 称之为学习速率)

即 θ 每次以一定的步伐按照 J(θ)最快速下降的方向更新值。我们进一步分解参数更新公式的右边。

$$\frac{\partial}{\partial(\theta j)} J(\theta) = \frac{\partial}{\partial(\theta j)} \left(\frac{1}{2}(h(x) - y)^2\right) = 2 * \frac{1}{2}(h(x) - y) * \frac{\partial}{\partial(\theta j)}((h(x) - y))$$

$$= (h(x)-y) * \frac{\partial}{\partial(\theta j)} \left(\sum_{i=0}^{n} \theta_i * x_i\right) = (h(x)-y) * x_i$$

因此，参数更新的方式如下：

$$\theta_j := \theta_j + \alpha(y-h(x^i)) * x_j^i$$

该更新方法称之为 LMS 算法(least mean squares，最小均方法)，也被称为 Widrow-Hoff 学习算法。该方法显得很自然和直观化。比如，当误差(y-h(x))越大的时候，那么参数更新的步伐就越大。检测是否收敛的方法：1) 检测两次迭代 θ_j 的改变量，若不再变化，则判定收敛；2) 更常用的方法：检验 J(θ)，若不再变化，判定收敛。需要注意的是，学习系数一般为 0.01 或者 0.005 等，我们可以发现样本量越大，$(y-h(x^i)) * x_j^i$ 会相对较大，所以在学习系数中用一个常数除以样本来定相对合理，即 0.005/m。

对于 LMS 算法，刚才推导的公式是针对于一个样本，如果样本多于一个，那么我们可以有两种方式更新参数。一种是每一次更新都使用全部的训练集合，该方法称之为批量梯度下降(batch gradient descent)。需要注意的是，梯度下降法很可能得到局部最优解，而我们这里的回归分析模型仅有一个最优解，因此局部最优解就是最终的最优解。(成本函数为凸函数)。另外一种是针对训练集合中每一个样本，每次都更新所有的参数值，该方法称之为随机梯度下降(stochastic gradient descent)。当数据量很大的时候，批量梯度下降法计算量较大，而随机梯度下降方法往往相对较优。通常情况下，随机梯度下降比批量梯度下降能更快的接近最优值(但也许永远也得不到最优值)，数据量大的情况下，通常选择使用随机梯度下降法。

梯度下降法的缺点是:靠近极小值时速度减慢(极小值处梯度为 0)，直线搜索可能会产生一些问题(得到局部最优等)，可能会'之字型'地下降(学习速率太大导致)。

2.2.2. 标准方程组推导

最小化成本函数的方式不只是有梯度下降法，这里我们采用标准方程组的方式求解得到精确的解。对于成本函数 J(θ)，我们定义输入变量 X 是 m*(n+1)维矩阵(包含了截距

项)，其中 m 表示样本数，n 表示特征数。输出$\vec{y}$是 m*1 维矩阵，参数 θ 是 n+1*1 维矩阵，则 J(θ)可以如下表示：

$$J(\theta) = \frac{1}{2}(X\theta - \vec{y})^T * (X\theta - \vec{y})$$

要求解 θ 使得 J(θ)最小，那么只需要求解 J(θ)对 θ 的偏微分方程即可。

$$\nabla_\theta J(\theta) = \frac{\partial}{\partial\theta}\left[\frac{1}{2}(X\theta - \vec{y})^T * (X\theta - \vec{y})\right]$$

$$= \frac{1}{2}\frac{\partial}{\partial\theta}[(X\theta - \vec{y})^T]*(X\theta - \vec{y}) + \frac{1}{2}\frac{\partial}{\partial\theta}[(X\theta - \vec{y})^T]*(X\theta - \vec{y})$$

$$= \frac{1}{2}X^T*(X\theta - \vec{y}) + \frac{1}{2}X^T*(X\theta - \vec{y})$$

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

$$= X^T*X*\theta - X^T*\vec{y}$$

因此，令$\nabla_\theta J(\theta) = 0$，即可求得使 J(θ)最小的 θ 的值，因此：$X^T*X*\theta - X^T*\vec{y} = 0$，得到 $\theta = (X^T*X)^{-1} * (X^T*\vec{y})$。在求解回归问题时候，可以直接使用该结果赋值于 θ，不过这里存在的问题是对矩阵的求逆，该过程计算量较大，因此在训练集合样本较大的情况并不适合。

注：$\nabla_\theta J(\theta)$表示对 J(θ)中的每一个 θ 参数求偏微分。这里化简的方式是矩阵偏微分注意：$\frac{\partial}{\partial\theta}(A^T*B) = \frac{\partial}{\partial\theta}A * B + \frac{\partial}{\partial\theta}(B^T)*A$, $\frac{\partial}{\partial\theta}A^T = (\frac{\partial}{\partial\theta}A)^T$, $A^T*B = (B^T*A)^T$。

2.2.3. 概率解释

对于回归问题，我们不禁要问为什么线性回归或者说为什么最小均方法是一个合理的选择呢？这里我们通过一系列的概率假设给出一个解释。(下一讲)

3. 欠拟合与过拟合概念

3.1. 欠拟合与过拟合概念

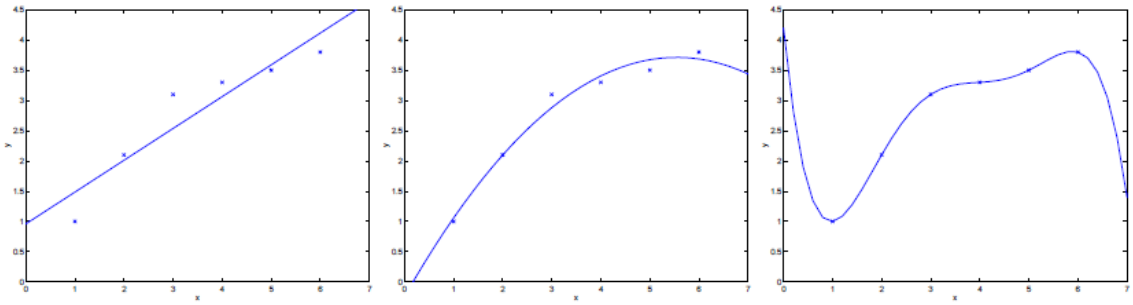


图 3-1 欠拟合与过拟合概念演示

通常，你选择让交给学习算法处理的特征的方式对算法的工作过程有很大影响。如图 3-1 中左图所示，采用了 $y = \theta_0 + \theta_1 x$ 的假设来建立模型，我们发现较少的特征并不能很好的拟合数据，这种情况称之为欠拟合(underfitting)。而如果我们采用了 $y = \theta_0 + \theta_1 x + \theta_2 x_2$ 的假设来建立模型，发现能够非常好的拟合数据(如中图所示)；此外，如果我们采用了 $y = \theta_0 + \theta_1 x + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5$，发现较多的特征导致了所有的训练数据都被完美的拟合上了，这种情况称之为过拟合(overfitting)。

这里，我们稍微谈一下过拟合问题，过拟合的标准定义(来自 Mitchell 的机器学习)标准定义：给定一个假设空间 H，一个假设 h 属于 H，如果存在其他的假设 h'属于 H,使得在训练样例上 h 的错误率比 h'小，但在整个实例分布上 h'比 h 的错误率小，那么就说假设 h 过度拟合训练数据。过拟合问题往往是由于训练数据少(无法覆盖所有的特征学习，换句话也可以认为是特征太多)等原因造成的。在以后的课程会具体讲解。

对于此类学习问题，一般使用特征选择算法(有一讲专门讲)或非参数学习算法，下面将要降到的局部加权线性回归就是属于该方法，以此缓解对于特征选取的需求。

3.2. 局部加权线性回归

局部加权线性回归(locally weighted linear regression)属于非参数学习算法的一种，也称作 Loess。

对于原始的回归分析，我们基本的算法思想是：

1) 寻找合适的 θ 使得$\frac{1}{2}\sum_{i=1}^{m}(y(i) - \Theta^T * x)^2$ 最小；2) 预测输出$\Theta^T * x$。

而对于局部加权线性回归算法的基本思想是：

1) 寻找合适的 θ 使得$\frac{1}{2}\sum_{i=1}^{m} w_i(y(i)-\theta^T * x)^2$ 最小；2) 预测输出$\theta^T * x$。

这里，局部加权线性回归与原始回归分析不同在于，多了权重 w_i，该值是正的。对于特点的点，如果权重 w 较大，那么我们选择合适的 θ 使得$(y(i)-\theta^T * x)^2$最小；如果权重 w 较小，那么误差的平方方$(y(i)-\theta^T * x)^2$在拟合过程中将会被忽略掉。换言之，对于局部加权回归，当要处理 x 时，会检查数据集合，并且只考虑位于 x 周围的固定区域内的数据点(较远点不影响因权重较低而被忽略)，对这个区域内的点做线性回归，拟合出一条直线，根据这条拟合直线对 x 的输出，作为算法返回的结果。

一个标准的且常用的权重选择如下：

$$w_i = \exp(-\frac{(x_i-x)^2}{2*\tau^2})$$

需要注意，这里的 x 是我们要预测的输入，而 x_i 是训练样本数据。从公式看，离 x 越近的点，权重越大，而这里的权重公式虽然与高斯分布很像，但是没有任何关系，当然用户可以选择不同的函数作为权重函数。而 τ 决定了各个点权重随距离下降的速度，称之为波长。τ 越大，即波长越大，权重下降速度越慢。如何选择合适的 τ 值，将会在模型选择一讲讲述。另外需要注意的是，如果 x 是多维特征数据的时候，那么权重是多维特征参与计算后的结果(结果为一维)，即 $w(i) = \exp(-(x(i)-x)^T (x(i)-x)/(2*\tau^2))$。(i 表示样本下标，j 表示特征下标)

参数学习算法(parametric learning algorithm)定义：参数学习算法是一类有固定数目参数，以用来进行数据拟合的算法。设该固定的参数集合为。线性回归即使参数学习算法的一个例子。非参数学习算法(Non-parametric learning algorithm)定义：一个参数数量会随 m(训练集大小)增长的算法。通常定义为参数数量虽 m 线性增长。换句话说，就是算法所需要的东西会随着训练集合线性增长，算法的维持是基于整个训练集合的，即使是在学习以后。

由于每次进行预测都要根据训练集拟合曲线，如果训练样本非常大，那么该方法可能是代价较大，可以参考 Andrew Moore 的 KD-tree 方法来思考解决。此外，局部加权线性回归依旧无法避免欠拟合和过拟合的问题。

3.3. 回归模型的概率解释

对于回归问题，我们不禁要问为什么线性回归或者说为什么最小均方法是一个合理的选择呢？这里我们通过一系列的概率假设给出一个解释。

首先，我们假设预测值 y 与输入变量满足如下方程：

$$y(i) = \theta^{T} * x(i) + \varepsilon(i)$$

其中 $\varepsilon(i)$表示由于各种未考虑的因素造成的残差或者噪音等，我们进一步假设 $\varepsilon(i)$独立同分布(误差项彼此之间是独立的，并且他们服从均值和方差相同的高斯分布)，服从均值为 0，方差为 δ^2 的正态分布，即 $\varepsilon(i) \sim \aleph(0, \delta^2)$，(为什么如此假设呢？一个是便于数学计算，另一个是可以通过中心极限定理等证明该残差服从正态分布)具体表示为：

$$p(\varepsilon(i)) = \frac{1}{\sqrt{2\pi}\delta} \exp\left(-\frac{(\varepsilon(i))^2}{2\delta^2}\right)$$

代入之后如下：

$$p(y(i)|x(i); \theta) = \frac{1}{\sqrt{2\pi}\delta} \exp\left(-\frac{(y(i)-\theta^T x(i))^2}{2\delta^2}\right)$$

这里的 θ 不是随机变量，而是具体的值。$p(y(i)|x(i); \theta)$，表示对于给定的 x(i)和 θ，y(i)出现的概率。那么对于一组 y 的话，可以表示为 $p(\vec{y}|X; \theta)$，这就相当于 θ 的似然性。(注意似然性与概率性不同，概率用于在已知一些参数的情况下，预测接下来的观测所得到的结果，而似然性则是用于在已知某些观测所得到的结果时，对有关事物的性质的参数进行估计。那么参数 θ 的似然函数如下：

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y}|X; \theta)$$

根据以上的假设和分析，我们得到：

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{m} p(y(i)|x(i);\ \theta) \\ &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\delta} \exp\left(-\frac{(y(i)-\theta^T x(i))^2}{2\delta^2}\right) \end{aligned}$$

对于该似然函数，我们已知了 x,y，需要找一个合适且合理的方法求取 θ。最大似然法则告诉我们选择的 θ 应该是 L(θ)最大。实际意义，选择参数使数据出现的可能性尽可能的大(让发生的事情概率最大化)。为了方便计算，我们通常对似然函数求取对数，将乘机项转化成求和。化简如下(注意本文中 log 的含义表示取自然对数，而不是取 10 为底对数)：

$$\ell(\theta) = \log(L(\theta))$$

$$= \log(\prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\delta} \exp(-\frac{(y(i)-\theta^T x(i))^2}{2\delta^2}))$$

$$= \sum_{i=1}^{m} \{\log\left(\frac{1}{\sqrt{2\pi}\delta}\right) + \log(\exp(-\frac{(y(i)-\theta^T x(i))^2}{2\delta^2})\}$$

$$=\text{m*log}(\frac{1}{\sqrt{2\pi}\delta}) - \frac{1}{\delta^2} * \frac{1}{2}\sum_{i=1}^{m}(y(i) - \theta^T x(i))^2$$

对于上式，最大化似然函数 L(θ)就相当于最小化$\frac{1}{2}\sum_{i=1}^{m}(y(i) - \theta^T x(i))^2$，即 J(θ)。

总结而言，对于数据作出概率假设之后，最小二乘回归方法相当于最大化 θ 的似然函数。通过概率假设，我们验证该方法的合理性，然而这些概率假设对于最小二乘法的合理性却并不是必要的，当然确实有其他的更自然的假设能够证明最小二乘法的合理性。

需要注意的是，对于上面的讨论，最终的 θ 选择并不依赖于δ^2(我们假设的误差俯冲的分布方差)尽管δ^2值并不知道。当讨论到指数族和广义线性回归的时候，我们还会提到这些。

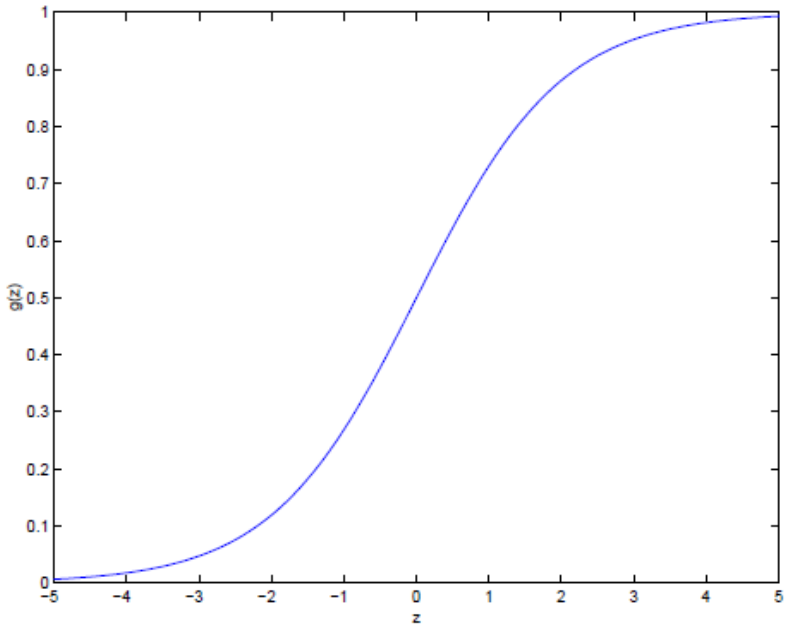
注意：我们用到的假设，第一残差项独立同分布且分布均值为 0，样本之间相互独立，此外，要想有结果需要输入特征不存在多重共线性，即输入特征是满秩矩阵。

3.4. Logistic 回归模型

在以上的讨论中，主要是关于回归问题。借来下，我们讨论分类问题，与回归问题很类似，不过待预测的 y 不是连续变量，而是有限的离散变量。这里，我们主要讨论二元分类，即预测的结果只有是否或者对错之类，习惯用 01 表示，其中 0 表示否定(negative)的结果，1 表示肯定(positive)的结果。对于给定的输入 X，y 也被称之为标签(label)。

针对分类问题，我们可以暂时忽略 y 的离散性，而采用回归分析的方法进行分析。然而，我们很容易发现回归分析效果很差，尤其是不好解释预测值不等于{0,1}的情况。为此，我们需要更改我们假设：

$h_\theta(x) = g(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)}$ 这里的 $g(z) = \frac{1}{1+\exp(z)}$，被称之为 logistic 函数或 sigmoid 函数。函数图形如下：



当 z 趋近于无穷大的时候，g(z)趋近于 1，当 z 趋近于无穷小的时候，g(z)趋近于 0。这里需要注意的是：我们仍然设定截距 x0=1,即$\theta^T x = \theta_0 + \sum_{j=1}^{n} \theta_j * x_j$。此外，对于 g(z)的导数 g' (z) : g'(z) = g(z)*(1-g(z))。因此，对于使用 logistic 函数作为假设 H 的模型，我们同样采用假设和使用最小二乘法来最大化参数的最大似然函数。

我们假设：$P(y=1 \mid x;\theta) = h_\theta(x)$ $P(y=0 \mid x;\theta) = 1- h_\theta(x)$ 该假设可以归结为：

$$p(y \mid x;\theta) = (h_\theta(x))^y * (1- h_\theta(x))^{(1-y)}$$

那么最大似然函数如下：

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X;\theta) \\ &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)};\theta) \\ &= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

取对数优化后：

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)})) \end{aligned}$$

同样，我们采用梯度下降法来迭代求取参数值，即按照 $\theta_j := \theta_j - \alpha * \frac{\partial}{\partial(\theta)} J(\theta)$方式跟新参数值。

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}\ell(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial\theta_j}g(\theta^T x) \\
&= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)g(\theta^T x)(1-g(\theta^T x))\frac{\partial}{\partial\theta_j}\theta^T x \\
&= \left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right)x_j \\
&= \left(y - h_\theta(x)\right)x_j
\end{aligned}$$

在推导过程中，我们使用 g'(z)=g(z)*(1-g(z))该特性。最终的更新结果如下：

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)}$$

对比 LMS 算法中更新参数的方法，与 logistic 回归法更新权重的一样。唯一不同的在于假设 $h_\theta(x)$，这个时候的假设是非线性的。当我们将到广义线性回归的时候，我们再来看是不是所有的算法够可以归结到这样的更新方式呢？

3.5. 感知器学习模型

在 logistic 方法中，g(z)会生成[0,1]之间的小数，但如何是 g(z)只生成 0 或 1？这里我们重新定义 g(z)函数如下：

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

之后，我们同样假设 $h_\theta(x) = g(\theta^T x)$，按照之前的方法(梯度下降法)我们将会同样得到参数更新的方式为：$\theta_j := \theta_j + \alpha(y - h(x^i))*x_j^i$，该模型称之为感知器学习算法(perceptron learning algorithm)。

20 世纪 60 年代，认为在神经网络模型采用“感知器学习”算法的神经元是较为粗糙的，但是他仍然能很好的表达学习算法理论。虽然从公式外形上来看，感知器学习与 logistic 回归和线性回归差不多，但是我们很难赋予感知器预测值的概率解释，或者使用最大似然估计推导出感知器模型。

4. 牛顿法与广义线性模型

4.1. 牛顿法求解最优值

在上面优化 cost function 的时候，我们采用了梯度下降法。在这里，我们采用另外一种方法求解 J(θ)的最小值(极小值)——牛顿法(Newton's method)。

首先，选择一个接近函数 f(x)零点的 x_0，计算相应的 $f(x_0)$和切线斜率 $f'(x_0)$(这里 f'表示函数 f 的导数)。然后我们计算穿过点$(x_0, f(x_0))$并且斜率为 $f'(x_0)$的直线和 x 轴的交点的 x 坐标，也就是求如下方程的解：

$$f(x_0) = (x_0\text{-}x) * f'(x_0)$$

我们将新求得的点的 x 坐标命名为 x_1，通常 x_1 会比 x0 更接近方程 f(x)=0 的解。因此我们现在可以利用 x_1 开始下一轮迭代。迭代公式可化简为如下所示：

$$x_{n+1} = x_n - f(x_n) / f'(x_n)$$

已经证明，如果 f'是连续的，并且待求的零点 x 是孤立的，那么在零点 x 周围存在一个区域，只要初始值 x_0 位于这个邻近区域内，那么牛顿法必定收敛。 并且，如果 f'(x)不为 0, 那么牛顿法将具有平方收敛的性能. 粗略的说，这意味着每迭代一次，牛顿法结果的有效数字将增加一倍。

回归到最大似然函数优化 $\ell(\theta)$(或 L(θ))，求解 $\ell(\theta)$的最大值也就相当于令 l(θ)对 θ 的导数为 0，即求解 θ 使得 $\ell'(\theta) = 0$，那么牛顿法迭代的过程就会变为：

$$\theta := \theta - \ell'(\theta) / \ell''(\theta)$$ (一阶导数除以二阶导数)

其中 := 表示更新时的赋值。需要注意的是，无论是求解最大值还是最小值，该迭代公式都是减号。(求取的极值，既可以是极大值也可以是极小值)

以上讨论的 θ 都是一个值，而有时候 θ 也可能是一个矩阵，比如在选择多个特征的线性回归模型中。当 θ 是一个矩阵的时候，牛顿法需要变换如下：

$$\theta := \theta - H^{-1} \nabla_\theta \ell(\theta)$$

其中，$\nabla_\theta \ell(\theta)$表示 $\ell(\theta)$对 θ 的偏微分矩阵，而 H 是一个 n*n 的矩阵(n 表示特征数目，实际上因加上截距项，常为(n+1)*(n+1))，称之为海森矩阵(hessian 矩阵)，这里表示 $\ell(\theta)$对 θ 的偏微分后再对 θ 的偏微分，定义如下：

$$H_{ij} = \frac{\partial^2 \ell(\theta)}{\partial \theta_i \partial \theta_j}$$

通常而言，牛顿法比批量梯度下降法收敛速度要快(牛顿法没有学习系数，且采用了二阶导数)，但是由于需要求解 hessian 矩阵逆矩阵，因此计算量较大，故牛顿法不适合 n 较大(特征较多)的优化求解。当使用牛顿法来优化 logistic 模型中的 $\ell(\theta)$时，该方法又称为费歇尔得分(fisher scoring)。

4.2. 广义线性模型

目前为止，我们讲解了一个回归模型和一个分类模型，在线性回归模型中，我们假设 $y|x;\theta \sim N(\mu, \sigma_2)$，在分类模型中，我们假设 $y|x;\theta \sim Bernoulli(\varphi)$，其中 μ,φ 均是 x 和 θ 的函数。然而，这两个模型只是一个广义线性模型(Generalized Linear Models)下的两种情况而已。

4.2.1. 指数族分布

我们指定一类分布：

$$p(y;\eta) = b(y)\exp(\eta^T T(y) - a(\eta))$$

其中，η 称为该分布的自然参数(natural parameter)或标准参数(canonical parameter)，通常是一个实数(也可能是实数矩阵，注意转置符号)；而 T(y)称之为充分统计量(sufficient statistic)，统计量，依赖且只依赖于样本 $y_1,y_2...y_n$，它不含总体分布的任何未知参数，通常情况下 T(y) = y；a(η)是累计函数(cumulant function，log partition function 或 normalization factor)，$\exp(-a(\eta))$主要是为了归一化，保证 p(y; η)的值在 0-1 之间。指数族分布主要是 a,b,T 三个函数，而参数是 η，不同的 η 值将会得到不同的概率分布，接下来分别以伯努力分布和高斯分布为例。

以伯努力(Bernoulli)分布为例：伯努力随机变量只有两个值 $y \in \{0, 1\}$，假设伯努力分布服从均值为 φ 的 B(φ)，那么 p(y = 1; φ) = φ，p(y = 0; φ) =1- φ，综合起来就是如下：

$$\begin{aligned} p(y;\varphi) &= \varphi^y(1-\varphi)^{1-y} \\ &= \exp(\, y\log(\varphi) + (1\text{-}y)\log(1-\varphi)\,) \\ &= \exp(\, y\log(\varphi) + \log(1-\varphi) + y\log(1/(1\text{-}\varphi))\,) \\ &= \exp(\, y\log(\varphi/(1\text{-}\varphi)) + \log(1-\varphi)) \end{aligned}$$

对比指数族分布，我们可以得到：$\eta = \log(\varphi/(1\text{-}\varphi))$，反过来由 η 可以求得 $\varphi = 1/(1+\exp(-\eta))$(恰好是 sigmoid 函数)。继续把 b,a,T 求解完整，那么 T(y) = y，$a(\eta) = -\log(1-\varphi) = \log(1+\exp(\eta))$，b(y) = 1。即伯努力分布可以是指数族分布的一种。

同样，我们考虑高斯分布。在推导回归问题的时候，我们提到最终的结果与高斯分布中的 σ^2 没有关系，因此可以随意选择方差值，这里为了方便计算设定 $\sigma^2=1$，那么：

$$
\begin{aligned}
p(y;\mu) &= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(y-\mu)^2}{2}\right) \\
&= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(y^2-2*y*\mu+\mu^2}{2}\right) \\
&= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{y^2}{2}\right)\exp\left(\frac{2*y*\mu-\mu^2}{2}\right)
\end{aligned}
$$

因此，可以得到：$\eta=\mu$，$T(y)=y$，$a(\eta)=\mu^2/2=\eta^2/2$，$b(y)=\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{y^2}{2}\right)$

当然还有很多其他的分布，比如泊松分布(Poisson，适合于描述单位时间内随机事件发生的次数的概率分布，如某一服务设施在一定时间内受到的服务请求的次数，电话交换机接到呼叫的次数、汽车站台的候客人数、机器出现的故障数、自然灾害发生的次数、DNA 序列的变异数、放射性原子核的衰变数等等)，伽马分布(gamma，伽玛函数可将整数拓展到了实数与复数域上)，指数分布(exponential，指数分布可以用来表示独立随机事件发生的时间间隔，比如旅客进机场的时间间隔、时间序列问题等等)，贝塔分布(beta)，狄利克雷分布(Dirichlet)等等。

4.2.2. 广义线性模型结构

不管是回归问题还是分类问题，我们可以推广到：预测一个随机变量 y,且 y 是 x 的函数。为了推倒广义线性回归模型(GLM),我们需要以下三个假设：

1) $y \mid x;\theta \sim \text{ExponentialFamily}(\eta)$，假设试图预测的变量 y 在给定 x，以 θ 作为参数的条件概率，属于以 η 作为自然参数的指数分布族。
2) 对于给定的 x，我们的目标是预测 T(y)的期望值。在我们的很多例子中，$T(y)=y$，这就意味着我们通过学习到的假设(hypothesis)的预测结果 h(x)满足 $h(x)=E[y|x]$。
3) 自然参数 η 与 x 是线性关系，即 $\eta=\theta^T x$。

为了证明普通最小二乘法和 logistic 回归是 GLM 的一种特殊情况，我们使用 GLM 的基本假设重新推倒一下最小二乘法和 logistic 回归。

对于普通最小二乘法，根据 GLM 的第一个假设，我们有对于给定的 x,y 服从高斯分布 $N(\mu, \sigma^2)$，根据假设二和假设三，我们知道 $h_\theta(x) = E[y|x; \theta] = \mu = \eta = \theta^T x$；(其中 $\mu = \eta$ 在上一小节已经求得)。

同样对于 logistic 回归，我们假设对于给定的 x，y 服从伯努力分布，即 $y|x; \theta \sim Bernoulli(\varphi)$。对于 φ，我们已经求得 $\varphi = 1/(1 + \exp(-\eta))$ 。根据假设二和假设三，$h_\theta(x) = E[y|x; \theta] = \varphi = 1/(1 + \exp(-\eta)) = 1/(1 + \exp(-\theta^T x))$。

更技术上的来说，对于 $g(\eta) = E[T(y); \eta]$而言，g 是自然参数 η 的函数，称之为正则响应函数(canonical response function)，而对于 g^{-1} 则称之为正则关联函数(canonical link function)。这样看，对于指数分布组而言正则响应函数只是辨别函数，比如对于伯努力分布而言就是 logis+tic 函数。

4.2.3. Softmax 回归

这里我们考虑一个更复杂的 GLM 模型。对于分类问题，我们这里分类结果不是二元分类，而是有 k 个分类结果，即 $y \in \{1, 2, \ldots, k\}$，比如对于邮件分类，我们不想分成垃圾邮件和非垃圾邮件，而是分成私人邮件，工作邮件和垃圾邮件。对于我们的响应变量(目标变量)y 仍然是离散的，不过数量超过两个，这里我们采用多项分布(multinomial distribution)来分析。

为了推导出符合多分类的广义线性模型，我们需要让参数多项话。参数多项化的一种方式是设定 k 个参数分别为 $\varphi_1, \ldots, \varphi_k$，来应对每一个分类结果的概率。但是这样的结果可能是冗余的，更准确的说是不能保证各个结果之间的相互独立性(因为我们知道概率之和等于一，即所有 φ_i 之和等于 1，那么最后一个类的参数值将会由之前的类的参数值决定)。因此，我们只需要设定 k-1 个参数 $\varphi_1, \ldots, \varphi_{k-1}$，而 $\varphi_k = 1 - sum(\varphi_i)$，但是我们需要注意的是：我们知道 φ_k 的值，但是 φ_k 并不是我们设定的参数。

为了更好的表示多项式指数族分布，我们定义 $T(y) \in R^{k-1}$，如下：

$$T(1) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, T(2) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, T(3) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots, T(k-1) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, T(k) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

与之前不同，不再有 $T(y)=y$，而且 $T(y)$是一个 k-1 维度的向量，$(T(y))_i$ 表示向量 $T(y)$的第 i 个元素。这里我们引入一种表达表达方式，$1\{\cdot\}$表示如果大括号里为真，则结果为 1，比如$(1\{True\}=1, 1\{False\}=0$，$1\{2=3\}=0$。由此，我们找到了 $T(y)$与 y 之间的关系，$T(y)_i=1\{y=i\}$。因此，$E[(T(y))_i]=P(y=i)=\varphi_i$。

因此，在多元广义回归模型下，我们根据独立事件概率乘法原则有：

$$
\begin{aligned}
p(y;\varphi) &= \varphi_1^{1\{y=1\}}\ \varphi_2^{1\{y=2\}}\ \dots\ \varphi_k^{1\{y=k\}} \\
&= \varphi_1^{1\{y=1\}}\ \varphi_2^{1\{y=2\}}\ \dots\ \varphi_k^{1-\sum_{i=1}^{k-1}1\{y=i\}} \\
&= \varphi_1^{T(y)_1}\ \varphi_2^{T(y)_2}\ \dots\ \varphi_k^{1-\sum_{i=1}^{k-1}T(y)_i} \\
&= \exp((T(y))_1\log(\varphi_1)+(T(y))_2\log(\varphi_2)+\cdots+(1-\textstyle\sum_{i=1}^{k-1}T(y)_i)\log(\varphi_k)) \\
&= \exp((T(y))_1\log(\varphi_1/\varphi_k)+(T(y))_2\log(\varphi_2/\varphi_k)+\cdots \\
&\qquad +(T(y))_{k-1}\log(\varphi_k-1/\varphi_k)+\log(\varphi_k)) \\
&= b(y)\exp(\eta^T T(y)-a(\eta))
\end{aligned}
$$

因此，针对参数有如下结果：

$$
\begin{aligned}
\eta &= \begin{bmatrix} \log(\phi_1/\phi_k) \\ \log(\phi_2/\phi_k) \\ \vdots \\ \log(\phi_{k-1}/\phi_k) \end{bmatrix}, \\
a(\eta) &= -\log(\phi_k) \\
b(y) &= 1.
\end{aligned}
$$

对于 1 至 k-1，$\eta_i=\log(\varphi_i/\varphi_k)$，转换一下结果为：$\exp(\eta_i)=\varphi_i/\varphi_k$ ，继续转换得到，$\varphi_k\ \exp(\eta_i)=\varphi_i$，根据 $sum(\varphi_i)=1$，因此可以得到：

$$\varphi_k\sum_{i=1}^{k}\exp(\eta_i)=sum(\varphi_i)=1$$

因此，$\varphi i=\exp(\eta_i)\ /\sum_{i=1}^{k}\exp(\eta_i)$，该方程将 φ_i 和 η_i 联系起来了，称之为 softmax 函数。为了完成该模型，我们使用假设三，即对于 i 从 1 到 k-1，$\eta_i=\theta_i{}^T x$，其中 $\theta\in R^{n+1}$。此外，我们可以假设$\theta_k=0$，那么就像之前的矩阵，$\eta_k=\theta_k{}^T x=0$。因此，针对于给定 xθ 的 y 的条件概率：

$$p(y=i|x;\theta)=\varphi_i=\exp(\eta_i)\ /\sum_{i=1}^{k}\exp(\eta_i)=\exp(\theta_i{}^T x)\ /\sum_{i=1}^{k}\exp(\theta_i T\ x)$$

这个针对于 $y\in\{1,\dots,k\}$的多元分类模型，称之为 softmax 回归(softmax regression)。

我们的假设输出结果是 $h_\theta(x) = E[T(y)|x;\theta]$，即

$$
\begin{aligned}
h_\theta(x) &= E[T(y)|x;\theta] \\
&= E\left[\begin{array}{c} 1\{y=1\} \\ 1\{y=2\} \\ \vdots \\ 1\{y=k-1\} \end{array}\middle| x;\theta\right] \\
&= \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{k-1} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\exp(\theta_1^T x)}{\sum_{j=1}^k \exp(\theta_j^T x)} \\ \frac{\exp(\theta_2^T x)}{\sum_{j=1}^k \exp(\theta_j^T x)} \\ \vdots \\ \frac{\exp(\theta_{k-1}^T x)}{\sum_{j=1}^k \exp(\theta_j^T x)} \end{bmatrix}.
\end{aligned}
$$

从结果看，我们的输出结果将包含 1 到 k 的各个结果的概率值，且概率值之和等于 1。

最后，我们讨论一下参数估计的问题。类似于最小二乘法和 logistic 回归，假设我们有 m 组训练数据集，那么利用最大似然函数如下：

$$
\begin{aligned}
\ell(\theta) &= \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta) \\
&= \sum_{i=1}^{m} \log \prod_{l=1}^{k} \left(\frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \right)^{1\{y^{(i)}=l\}}
\end{aligned}
$$

接下来，我们可以使用梯度下降法或牛顿法求解其极大值。这里我们分别对 k 个 θ 分别求解偏微分，那么得到θ_l的迭代公式，最终的结果与之前结果类似，如下：

$$
J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)}=j\right\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right]
$$

$$
\nabla_{\theta_j} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[x^{(i)}\left(1\{y^{(i)}=j\} - p(y^{(i)}=j|x^{(i)};\theta)\right)\right]
$$

$$
\theta_j := \theta_j - \alpha \nabla_{\theta_j} J(\theta)
$$

关于更详细的介绍请参见深度学习——softmax 回归。

注：如果需要对 k 个类别的数据分类，那么选择使用 softmax 分类器呢，还是使用 logistic 回归算法建立 k 个独立的二元分类器呢？这一选择取决于你的类别之间是否互斥，例如，如果互斥的话，可以选择 softmax 分类器。如果不互斥的话，选择 K 个独立的二分类的 logistic 回归分类器更为合适。

5. 生成学习算法

5.1. 生成学习算法引入

目前为止，我们主要讲解了条件概率模型 p(y|x,θ)的学习算法。接下来，我们将讨论其他的学习算法。接下来举个例子，比如现在遇到一个分类问题，基于一些特征来判断一个动物是大象 (y = 1) 还是小狗 (y = 0)。基于给定的数据集，我们可以采用 logistic 回归或者感知器学习的方法来寻找一条直线(称之为决策边界)把大象和小狗分割开割开来。之后预测新的动物时，只要判断它在决策边界的哪一边就可以预测其所属分类。

现在有另外一种方法。首先是查看大象的数据，然后建立一个什么是大象的模型，之后查看小狗的数据，建立一个什么是小狗的模型。之后，遇到新的动物，只要分别代入两个模型中，看更像小狗还是大象就可以分类了。

通过训练集合来学习 p(y | x)的学习算法称之为判别学习算法(discriminative learning algorithms)，而通过训练集合来学习得到 p(x | y)的学习算法则称之为生成学习算法(generative learning algorithms)。对于上大象和小狗的例子，p(x | y=0)表征了小狗的特征分布。

在知道了类的先验概率(class priors)p(y)之后，根据贝叶斯规则(Bayes rule)，可以得到：p(y | x) = p(x | y) p(y) / p(x)，其中分母 p(x) = p(x | y=1)p(y = 1) + p(x | y =0) p(y = 0) (全概率公式)。如果只是为了预测(判断哪一个 p(y | x)更大)，我们就不需要计算分母，只需要比较分子的大小就可以。

5.2. 高斯判别模型

第一个生成学习算法，我们通过高斯判别分析(Gaussian discriminant analysis (GDA))讲述。在这个模型中，将假设 p(x | y)服从多元正态分布(multivariate normal distribution)。这里简单的介绍下多元正态分布。

多元正态分布，又称为多元高斯分布，它的均值 $\mu \in R^n$，它的方差 Σ 称为方差矩阵(covariance matrix)， $\Sigma \in R^{n*n}$，是一个对称矩阵，表示为 N (μ, Σ)，表示如下：

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

更多介绍可以参见(多元正态分布)

当我们遇到一个分类问题，且特征是连续变量时，我们可以采用高斯判别模型，通过多元正态分布来表达 p(x|y)，具体为 y ~ Bernoulli(φ)，x|y = 0 ~ N (μ_0, Σ)，x|y = 1 ~ N (μ_1, Σ)，表达式如下：

$$\begin{aligned} p(y) &= \phi^y(1-\phi)^{1-y} \\ p(x|y=0) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right) \\ p(x|y=1) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right) \end{aligned}$$

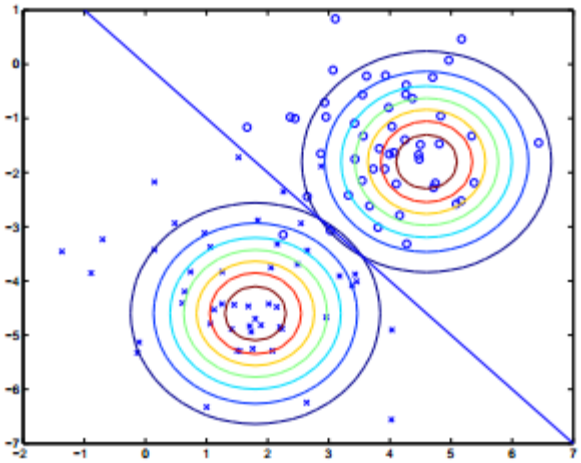
这里，需要注意我们的假设条件中有两个不同的 μ(分别为 μ_0 μ_1), 但是 Σ 却只有一个(特征之间的方差)。之后，采用最大似然估计，可以得到：

$$\begin{aligned} \ell(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \\ &= \log \prod_{i=1}^{m} p(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi). \end{aligned}$$

通过求解最大似然估计，我们得到各个参数的估计值(实际推导过程中，可以把样本分成 1-k,k-m 分别表示结果为 0 和 1 的结果，这样能够将复杂的表达式简单化，其中 k=$\sum_{i=1}^{n} 1\{y^{(i)} = 1\}$)：

$$\begin{aligned} \phi &= \frac{1}{m}\sum_{i=1}^{m} 1\{y^{(i)} = 1\} \\ \mu_0 &= \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}} \\ \mu_1 &= \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}} \\ \Sigma &= \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T. \end{aligned}$$

形象的来说，该算法可以表述如下：对于用于训练集合得到的两个高斯分布，他们具有相同的轮廓和等高线(因为方差矩阵相同)，而均值不同。我们也可以看到决策边界 p(y = 1|x) = 0.5，将两个类分开。



5.3. GDA VS logistic 回归

GDA 模型和 logistic 回归模型具有相似的关系，即如果我们把概率质量 $p(y = 1|x; \varphi, \mu_0, \mu_1, \Sigma)$看成是 x 的一个函数，那么我们可以用如下公式来表述：

$$p(y = 1|x; \varphi, \mu_0, \mu_1, \Sigma) = 1 / (1 + \exp(-\theta^T x))$$

其中 θ 是 $\varphi, \mu_0, \mu_1, \Sigma$ 的函数，这恰好是 logistic 回归——是对于 $p(y = 1|x)$的高斯判别分析。然而，通常来说，GDA 和 logistic 回归会给出不同的决策边界，那么哪一种模型更好呢，或者说我们该如何选择 GDA 还是 logistic 回归呢?

如果 $p(x|y)$服从多元正态分布(且 Σ 相同)，那么 $p(y|x)$可以使用 logistic 函数。相反，如果 $p(y|x)$是 logistic 函数，那么 $p(x|y)$未必服从多元正态分布。这说明 GDA 比 logistic 对数据集的假设要求更强。特别的，如果 $p(x|y)$服从多元正态分布(且 Σ 相同)，那么采用 GDA 将是渐进有效(asymptotically efficient)的。通俗的讲，就是随着样本量的增加，分类精度将会增加。换句话说，如果样本量足够大的话，那么没有任何方法能够比 GDA 更准确的描述 $p(y|x)$。此外，即使数据集较小，我们也认为 GDA 模型更优于 logistic 回归。

如果对数据集有较弱的假设，那么 logistic 回归鲁棒性(robust)更强，而且对不正确的数据分布假设也不会过于敏感。总结的说，GDA 有更强的模型假设，对于服从模型假设或近似于假设的数据而言，GDA 更依赖于数据的有效性。而 logistic 回归具有较弱的模型假设，而且对于模型假设偏差鲁棒性更好。此外，如果数据集分布不服从正态分布，那么 logistic 回归通常优于 GDA。

5.4. 朴素贝叶斯

5.4.1.朴素贝叶斯模型

在 GDA 模型中，我们的特征数据是连续型变量，现在我们讨论一种特征数据是离散变量的模型。在这里，我们通过机器学习在垃圾邮件中的分类案例，来讲述该模型。即我们通过判断一封邮件是垃圾邮件还是非垃圾邮件来进行邮件的分类，在学习得到该模型之后，我们可以自动的判断一封邮件是否是垃圾邮件。此外，垃圾邮件分类是属于文本分类(text classification)的一种。

假设我们现在有一堆训练集合(即已经标注为垃圾和非垃圾的一堆邮件)，我们首先需要做的就是寻找一组合适的特征 x_i 来表示每一封邮件。这里我们采用字典里的所有单词作为一封邮件的特征，即如果该邮件中出现字典中第 i 个单词，那么 $x_i = 1$，如果不包含，那么 $x_i = 0$。整体来说，我们使用如下向量：

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \text{a} \\ \text{aardvark} \\ \text{aardwolf} \\ \vdots \\ \text{buy} \\ \vdots \\ \text{zygmurgy} \end{matrix}$$

来表示一封邮件的特征，即邮件中是否包含字典中的单词。然而，通常情况下，我们不使用字典(所有单词都用的话，特征太多)，而是使用一定的方法从训练集合中找出合适的单词(有的时候需要使用停用词(stop words)或者 tf-idf 等方法来选择合适的特征)。对于 x 中的所有单词，我们称之为词汇表(vocabulary)，特征属性的大小就是词汇表中词汇的个数。在得到我们的特征向量(feature vector)之后，我们开始构建生成学习算法。

假设我们有 50000 个单词，那么我们首先要构建 p(x|y)，其中 $x \in \{0, 1\}^{50000}$(x is a 50000 维向量，值仅含 0 和 1)。如果我们通过明确的分布来描述 x(直接构建 p(x|y))，那么我们需要 2^{50000} 个结果，即 2^{50000}-1 维向量的参数，将会有特别多的参数。为了方便 p(x|y)建模，我们需要做出一个很强的假设：对于给定的 y，x_i 条件独立(conditionally independent 即特征之间相互独立)，该假设称之为朴素贝叶斯假设(Naive Bayes (NB) assumption)，因此该模型又称之为朴素贝叶斯分类(Naive Bayes classifier)。举例子而言，对于给定的邮件比如垃圾邮件，那么特征单词 buy 是否出现并不影响特征单词 price 的出现。因此，有：

$$p(x_1, \ldots, x_{50000}|y) = p(x_1|y)p(x_2|y, x_1)p(x_3|y, x_1, x_2) \cdots p(x_{50000}|y, x_1, \ldots, x_{49999})$$
$$= p(x_1|y)p(x_2|y)p(x_3|y) \cdots p(x_{50000}|y) \quad = \sum_{i=1}^{n} p(x_i|y)$$

在第二步转换过程中，我们使用了 NB 假设。需要注意的是，即使贝叶斯有很强的假设要求，但是该方法在很多问题上(特征并非完全条件独立)仍然后较好的结果。

对于训练集合$\{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$，我们的模型参数主要有

$$\varphi_{i|y=1} = p(x_i=1 \mid y=1),$$
$$\varphi_{i|y=0} = p(x_i=1 \mid y=0)$$
$$\varphi_y = p(y=1)$$

那么我们的最大似然函数如下：

$$\mathcal{L}(\phi_y, \phi_{j|y=0}, \phi_{j|y=1}) = \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}).$$

最大化似然函数之后，我们得到：

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}$$
$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}}$$
$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}$$

该公式比较容易理解，$\varphi_{j|y=1}$ 分子是出现单词 j 的垃圾邮件个数，分母是所有垃圾邮件个数；而 φ_y 分子表示垃圾邮件个数，而分母表示总样本数。对于新出现的一封邮件，我们在得到特征向量之后，计算 p(y=1|x)的值，就可以判断其所属类别。

$$p(y = 1|x) = \frac{p(x|y = 1)p(y = 1)}{p(x)}$$
$$= \frac{\left(\prod_{i=1}^{n} p(x_i|y = 1)\right) p(y = 1)}{\left(\prod_{i=1}^{n} p(x_i|y = 1)\right) p(y = 1) + \left(\prod_{i=1}^{n} p(x_i|y = 0)\right) p(y = 0)}$$

此外，我们也可以扩展 NB 模型。比如 x=1,2,3...k，那么我们可以使用多项分布来替代伯努力分布。如果 x 是连续变量，可以将 x 离散化。当原始数据是连续型的，且不能很好的满足多元正态分布，此时采用离散化后，使用贝叶斯分类(相比于 GDA)，通常会有较好的结果。

5.4.2.拉普拉斯平滑

对于很多问题，采用朴素贝叶斯方法都能够取得很好的结果，但是有些情况则需要对朴素贝叶斯分类进行一定的优化，尤其是在文本分类领域。接下来，我们讨论朴素贝叶斯方法在使用过程存在的缺陷，并探讨如何去优化。

仍然考虑垃圾邮件分类问题。比如在预测一封信邮件是否是垃圾邮件过程中，特征单词中的第 3500 个单词 nips 恰好在训练集合中从来没有出现，那么通过最大似然估计得到的 $\varphi_{35000|y}$ 如下：

$$\begin{aligned}\phi_{35000|y=1} &= \frac{\sum_{i=1}^{m} 1\{x_{35000}^{(i)} = 1 \wedge y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}} = 0 \\ \phi_{35000|y=0} &= \frac{\sum_{i=1}^{m} 1\{x_{35000}^{(i)} = 1 \wedge y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}} = 0\end{aligned}$$

$$\begin{aligned}p(y=1|x) &= \frac{\prod_{i=1}^{n} p(x_i|y=1)p(y=1)}{\prod_{i=1}^{n} p(x_i|y=1)p(y=1) + \prod_{i=1}^{n} p(x_i|y=0)p(y=0)} \\ &= \frac{0}{0}.\end{aligned}$$

因为单词 nips 从来没有出现，因此垃圾邮件和非垃圾邮件的条件概率均为 0，因此在计算是否是垃圾邮件的后验概率中，因为分子中存在一个 0 项，导致最终结果无法判断。

推而广之，即因为训练集合的有限性而使某些特征值没有被包含，导致统计得到的条件概率为 0。为了方便说明，我们取某一个特征 z 来说明，z 的取值在$\{1, \ldots, k\}$，在给定的 m 个训练集里 m，$\{z^{(1)}, \ldots, z^{(m)}\}$，根据传统的最大似然估计，我们得到条件概率：

$$\phi_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}{m}.$$

我们知道该方法存在一定缺陷，因此采用拉普拉斯平滑(Laplace smoothing)进行优化，即在分子中加 1(即默认每一个值都已经出现过一次)，考虑左右的频率和为 1，因此分母加上 k(k 个值)，如下：

$$\phi_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} + 1}{m + k}.$$

这样如果一个特征没有出现，那么条件概率将不会是 0，解决了刚才提到的问题。对于垃圾邮件分类，那么我们知道一个单词只有两个值(0，1)，因此分子需要加 1，而分母

需要加 2。实际应用中对于先验概率 φ_y(不是条件概率 $\varphi_{x|y}$)是否采用拉普拉斯平滑，并没有很大的影响，因为收集的数据集相对公平，不会导致 0 值的出现。

此外，需要注意的是，在计算新邮件的后验概率的分子的时候，$\varphi_{i|y=1} = p(x_i=1 \mid y=1)$，这里不管是 y=1 还是 y=0，都只计算 x=1 的情况，即只计算存在的单词(即出现的单词，不出现的单词不用计算)。此外，因为条件概率值介于 0 到 1 之间，50000 特征的条件概率相乘(分子的计算)将会出现极小值，在某些情况下可能无法进行比较，因此，可以针对条件概率取对数，这样不仅可以放大概率值，还可以将乘法转化成加法，并得到可比较的值。

6. 朴素贝叶斯算法

6.1. 文本分类模型

在结束生成算法模型之前，我们将一种专门用于文本分类的算法。对于分类问题，朴素贝叶斯算法通常效果很好，而对于文本分类而言，则有更好的模型。

对于文本分类，之前提到的朴素贝叶斯算法又称之为多元伯努力事件模型(multi-variate Bernoulli event model)。模型分析，在之前已经讨论过了。这里，我们解释一下如何理解数据的生成呢？在该模型中，我们假设邮件是按照先验概率(p(y))随机发送邮件或垃圾邮件到用户手里的，之后遍历词汇表，以伯努力分布生成该邮件单词(特征向量，同时假设了各个单词在邮件中出现的概率是条件独立)，之后根据 $p(x_i= 1|y) =\varphi_{i|y}$ 进行后验概率的计算，得到 $p(y) \prod_{i=1}^{n} p(x_i|y)$(只要计算后验概率的分子即可，各个类分母均相等，用于归一化)。该模型中，x 取值仅有{0，1}，且生成特征的方式是以遍历词汇表的方式。

这里我们介绍另外一种方法，称之为多项事件模型(multinomial event model)。为了更好的表示，这里采用另一种表示的方法。其中，x_i 表示单词 i 在词汇表中的地址，那么 x_i 取值范围是{1,2,3...|V|}，其中 |V| 是词汇表的词汇数量。举个例子而言，一封邮件有 n 个单词，表示为向量$(x_1, x_2, \ldots, x_n)$，注意这里的 n 对于不同的文档来说是不一样的。举例而言，一封邮件开头是“a nips...”那么 $x_1=1,x_2=3500$(词汇表中，a 是第一个单词，nips 是第 3500 个)。

在多项事件模型中，同样看看数据是如何生成呢？我们假设以随机的方式产生(当然还是存在先验概率 p(y))垃圾邮件或非垃圾邮件，之后按照多项分布生成邮件中的第一个单词，在同样的分布下，生成其他的单词直到 x_n(每次的生成都相互独立)，因此产生该邮件的全部概率值就是 $p(y) \prod_{i=1}^{n} p(x_i|y)$，虽然该生成概率值与之前的多项伯努力分布很像，但是代表含义不同，尤其是 $x_i|y$ 现在是多项分布而不是伯努力分布。

对于新模型中的参数如下，注意这里假设了对于所有的 j(j 是样本的下标), $p(x_j|y)$的值均相等(生成哪个单词的概率分布并不依赖于该样本在整体中的位置)，其中 k 表示具体的单词。

$$\varphi_y = p(y=1)$$
$$\varphi_{k|y=1} = p(x_j=k \mid y=1)\text{(对于每一个 j)}$$
$$\varphi_{k|y=0} = p(x_j=k \mid y=0)\text{)(对于每一个 j)}$$

对于给定的数据集$\{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$，其中$x^{(i)}=(x^{(i)}_1, x^{(i)}_2, \ldots, x^{(i)}_{ni})$(这里的 ni 表示第 i 的个样本中的单词数)，对于数据的最大似然函数如下：

$$\begin{aligned}\mathcal{L}(\phi, \phi_{k|y=0}, \phi_{k|y=1}) &= \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}) \\ &= \prod_{i=1}^{m} \left(\prod_{j=1}^{n_i} p(x_j^{(i)}|y; \phi_{k|y=0}, \phi_{k|y=1}) \right) p(y^{(i)}; \phi_y).\end{aligned}$$

求解最大似然函数，结果如下：

$$\begin{aligned}\phi_{k|y=1} &= \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} n_i} \\ \phi_{k|y=0} &= \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} n_i} \\ \phi_y &= \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}.\end{aligned}$$

该结果显示，$\varphi_{k|y=1}$ 的分子是单词 k 在每一封垃圾邮件中的出现次数的总和，分母是每一封垃圾邮件长度的总和；φ_y 分子表示垃圾邮件个数，分母表示样本总数。

如果采用拉普拉斯平滑，那么结果如下，即在分子中加 1，在分母中加上词汇表长度：

$$\begin{aligned}\phi_{k|y=1} &= \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} n_i + |V|} \\ \phi_{k|y=0} &= \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} n_i + |V|}.\end{aligned}$$

在许多分类问题中，尽管可能不一定是最优模型，贝叶斯模型也是最值得第一个尝试的模型，因为他相对简单且容易解释。

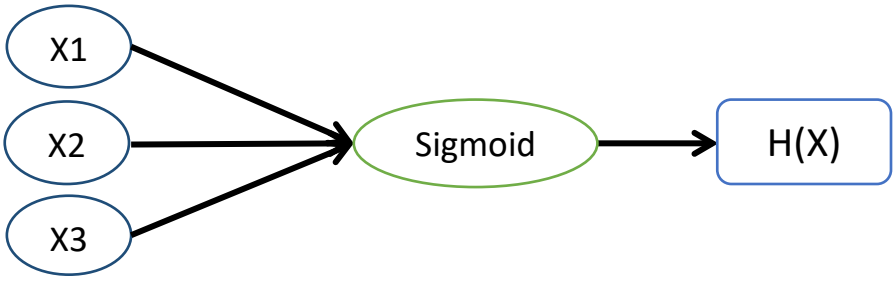
在文本分类的过程中，通常情况下多项事件模型比朴素贝叶斯模型要好，因为他考虑了词语出现的次数。当然，研究者仍存在争论。在文本分类中，仍然是非常优秀的。当然，考虑了词语的顺序等等，也会提高分类效果，如马尔可夫链模型，但只是很细微提高。

6.2. 神经网络模型

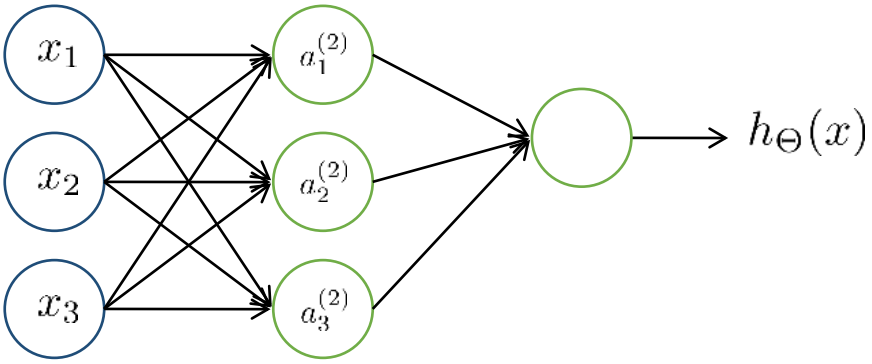
在之前的模型中，无论是 logistic 回归还是贝叶斯分类(朴素贝叶斯模型是前置假设是多项分布的多项事件模型，所以也可以划到 logistic 回归)等模型，其最终结果反映在数据

上都是一条直线或是一个超平面，即这些模型均是线性模型。如果数据并不线性可分的话，这些模型的性能就会变差。针对该问题，出现了很多针对非线性可分数据进行分类的算法。其中之一就是神经网络模型，这也是出现最早的一种。

对于 logistic 回归，我们可以用如下的方法表示：



其中，Xi 是输入的特征变量，中间的 sigmoid 函数用计算单元的形式表示，最终输出结果。而神经网络就是将这些简单的单元组合起来，如下图所示：



其中，a1,a2,a3 是中间单元的计算输出。通常，我们也会在输入特征中则加 x0=1，这样更吻合实际的 logistic 回归。这里，我们取函数 g 来表示 sigmoid 函数，则有下式：

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

这里就不再细数神经网络模型，关于神经网络模型是一个非常复杂的模型，包含的种类也很多。采用 sigmoid 函数作为计算单元(在神经网络中称之为神经元)，且仍然采用梯度下降法进行参数的优化，这种神经网络通常称之为反向传播(back propagation)神经网络。

上面的例子中，与输入直接相连的层称之为隐藏层(hidden layer)，与输出直接相连的称为输出层(output layer)。在神经网络中，可以自定义的设置隐藏层中神经元个数，然而我们并不知道隐藏层计算的东西的意义何在，属于黑盒子模型。此外，对于神经网络而言，获得全局最优将会更难，因为它不是凸函数优化，存在很多局部最优解。相比于神经网络，我们会更倾向于 svm，因为 svm 高效且无需定制，而神经网络用的会比较少，因为其存在严重的优化问题，且相对复杂。

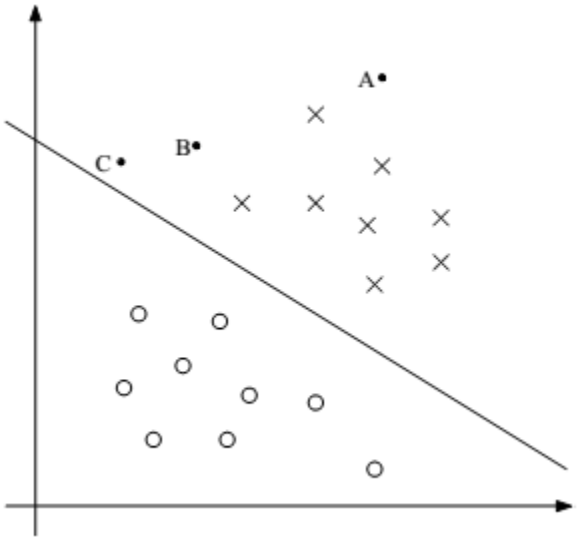
6.3. 支持向量机

在上一讲我们提到了一点 SVM，这里我们将详细的讲述 SVM 算法，当然有一部分认为 SVM 是最好的分类算法。在监督式学习算法中，SVM 无需做出太多的改动，就能得到很好的分类效果。为了更好的讲述 SVM，我们需要从划分数据的空间间隔(margins)说起，然后讨论最优间隔分类器(optimal margin classifier)，其中会涉及拉格朗日对偶(Lagrange duality)。此外，还会涉及核方法(kernels)以此解决高维特征数据的分类问题，最后我们使用 SMO(Sequential minimal optimization，序列最小优化)。

6.3.1.直觉上的间隔

首先我们从间隔开始说起。这里先从直观的角度去看间隔和分类的效果，之后再正式的推倒。考虑一下 logistic 回归，概率 $p(y=1 \mid x;\theta)$ 是等于 $h_\theta(x) = g(\theta^T X)$ 的，如果 $h_\theta(x)$ 大于 0.5(或者等价于 $\theta^T X$ 大于 0)，那么我们就预测分类结果是 1，反之则预测为 0。现在我们考虑一个训练样本 y=1，如果 $\theta^T X$ 越大，那么 $h_\theta(x) = p(y = 1|x; w, b)$ 越大，因此我们越有信心(confidence)判断结果为 1。通俗的说，如果 $\theta^T x \gg 0$，那么我们就有十足的信心认为 y=1，同样如果 $\theta^T x \ll 0$，那么我们就有十足的信心认为 y=0。因此，对于所有的训练集合，如果我们找到了合适的参数，使得在 $y^{(i)}=1$ 的样本中 $\theta^T x^{(i)} \gg 0$，而在 $y^{(i)}=0$ 的样本中 $\theta^T x^{(i)} \ll 0$，这就暗示了我们有很大的信心说这是一个好的分类器。这个想法看起来很通俗直观，之后我们会用正式的表达式表示函数间隔。

此外，我们用另一种直观的表示，如下图所示，x 标记的点表示 y=1 的样本点，o 标记的点表示 y=0 的样本点，中间的决策边界(decision boundary)如图所示，该边界直线方程是 $\theta^T x=0$。其中 A 点离决策边界非常远，那么 $\theta^T x^{(i)} \gg 0$，也就是说我们有很大的信心认为 A 点属于 y=1 类别。而 C 点离决策边界较近，那么我们就没 A 点和 B 点那么大的信心认为 C 点属于 y=1。通俗的说，就是在决策边界的两边的点，距离决策边界越远的点，我们越有信心确定该点的所属分类。即找到一个分割两类样本的决策边界，且使得样本点离决策边界足够远，这样我就更有信心确定该点的所属分类。



6.3.2.符号表示

为了更好的讨论 SVM，这里使用一些新的符号表示。这里仍然是从二元分类说起，对于类别为 y，特征为 x 的问题，我们使用 $y \in \{-1, 1\}$ (与之前的 {0, 1}不同)表示不同的类别。与之前线性分类器不同的向量参数 θ 不同，这里的参数使用 w 和 b，分类器假设为：

$$h_{w,b}(x) = g(w^T x + b)$$

其中，如果 $z \geq 0$，则 $g(z) = 1$，反之，$z<0$，则 $g(z) = -1$。这里设置两个不同参数，故可以把截距项 b 单独的对待(与之前不同，之前会在特征数据中增加 $x_0=1$)。简单的看，b 类似于之前的 θ_0，而 w 则表示了$[\theta_1,\theta_2,...\theta_n]^T$。需要注意的是，与之前定义的 g 不同，这里的 g 仅有-1 和 1 两个值，中间并没有估计 y=1 的概率。

6.3.3.几何间隔

这里我们正式的表示函数间隔(functional margin)和几何间隔(类似于点到直线距离公式，但有区别)。对于一个训练样本$(x^{(i)}, y^{(i)})$，函数间隔为

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b)$$

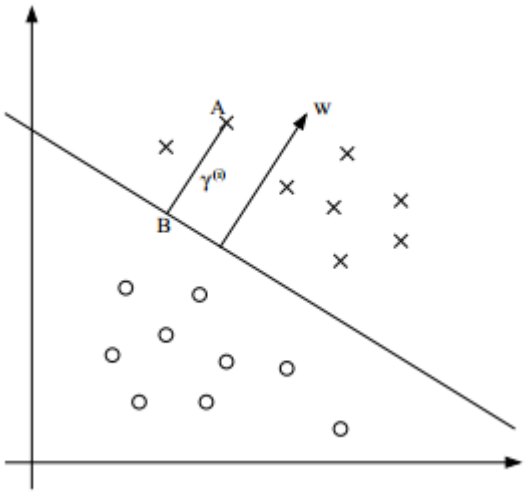
如果 $y^{(i)} = 1$，为了让函数间隔$\hat{\gamma}^{(i)}$越大(即预测为 y=1 类的信心愈大)，那么我们希望 $w^T x + b$ 是一个非常大的正数。相反，如果 $y^{(i)} = -1$，为了让函数间隔$\hat{\gamma}^{(i)}$越大，那么我们希望 $w^T x + b$ 是一个非常小的负数。此外，如果 $y^{(i)}(w^T x + b) > 0$，则说明该样本的分类预测正确。因此，函数间隔$\hat{\gamma}^{(i)}$越大，则说明我们越有信心得到一个正确的分类结果。

对于一个选择了函数 g(仅有结果{-1,1})的线性分类器，有一个特征使得他并不能很好的度量信心。对于 g，如果我们把 w 和 b 分别用 2w 和 2b 来替换，那么 g 就可以表示为 $g(2w^Tx + 2b)$，但是这对 $h_{w,b}(x)$没有任何影响，因为 $h_{w,b}(x)$仅依赖于正负号。但是对于间隔来说却是翻倍了，换句话说我们不需要考虑实际的样本，而仅通过改变 w 和 b 的值就可以无限的扩大间隔。因此，直觉上我们希望通过归一化如$||w||_2= 1$ 来解决这个问题。我们之后在进行讨论归一化的问题。

上面的函数间隔定义是针对于一个样本，而对于给定的数据集 $S = \{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$，这里我们定义的函数间隔为有所间隔中最小的间隔：

$$\hat{\gamma} = \min_{i=1-m}(\gamma^{(i)})$$

现在我们讨论一下几何间隔(geometric margins)，考虑下图：



对应(w, b)的决策边界如图所示，其中 w 正交于超平面。(对于直线 $w^T x+b=0$,方向为)。对于表示训练集合中输入特征 $x^{(i)}$且分类为 $y^{(i)} =1$ 的 A 点而言，它到决策边界的距离为 $\gamma^{(i)}$，即线段 AB。那么我们如何求得 $\gamma^{(i)}$呢？我们知道 w/||w|| 表示 B 到 A 的单位向量，即方向向量 w 的单位向量。对于已知 A 点是 $x^{(i)}$，那么 B 点的输入为：$x^{(i)} - \gamma^{(i)} *(w/||w||)$ (即由 A 点按照向量 w 的反方向移动 $\gamma^{(i)}$个单位之后得到 B 点)。同时，我们知道 B 点在决策边界上，满足决策边界方程。因此有：

$$w^T (x^{(i)} - \gamma^{(i)} *(w/||w||)) + b = 0$$

因为 $w^Tw/||w|| = ||w||$，因此求得 $\gamma^{(i)} = (w^T/||w||) x^{(i)} + b / ||w||$ 。这里求得的距离是针对 y=1 的 A 点，距离 γ 是正数。如果是在决策边界的另一侧呢？

更普遍的说，我们定义几何距离(geometric margin)，对于给定的训练样本($x^{(i)}$，$y^{(i)}$)，几何距离如下：

$$\gamma^{(i)} = y^{(i)} \left((w^T / ||w||) x^{(i)} + b / ||w|| \right)$$

注意如果 $||w|| = 1$，那么几何间隔恰好等于函数间隔，这也就是两种不同表示之间的关系。同样，几何间隔不随着参数 wb 尺度的改变而改变。 举例而言，如果 w 和 b 分别扩大一倍为 2w 和 2b，那么几何间隔不会变化(因为分母||w||也会随之变为 2||w||)。在后面。这会非常方便做一些变化处理等，比如我们可以利用这该不变性灵活的处理 w 和 b，比如通过缩放 wb 使得$|w1| = 5$ 或者 $|w1 + b| + |w2| = 2$ 等等。

最后，对于给定的训练集合 $S = \{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$，这里我们定义的几何间隔为有所几何间隔中最小的间隔：

$$\gamma = \min_{i=1\text{-}m}(\gamma^{(i)})$$

7. 最优间隔分类器

7.1. 最优间隔分类器

对于一个给定的数据集，目前有一个很现实的需求就是要找到一个合适的决策边界，使得样本中的最小间隔(几何间隔)最大，而且这样的分类器能够使得分割的训练样本集之间的间隔(gap)最大。现在，我们假设训练集合线性可分，即可以找一条超平面把正样本和负样本分割开来。那么我们如何找到一个超平面来最大化几何间隔呢？我们得到了如下的优化问题：

$$\max_{\gamma,w,b}\ \gamma$$
$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq \gamma,\ i = 1, \ldots, m$$
$$||w|| = 1$$

也就是说，我们希望最大化 γ，其中对于每一个样本而言，每一个函数间隔值都不小于 γ。其中，设定$||w||$ 固定为 1，能够保证函数间隔等于几何间隔，也能保证几何间隔最小可达到 γ。因此，解决这个优化问题后就可以得到该训练集合的最大可能几何间隔。

但是很难解决上面的优化问题，因为$||w|| = 1$ 是非凸优化，这就无法使用标准优化软件去解决这个优化问题。因此，我们把这个优化问题转化成相对简单的问题，如下：

$$\max_{\gamma,w,b}\ \hat{\gamma}/||w||$$
$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq \hat{\gamma},\ i = 1, \ldots, m$$

这里我们将要最大化$\hat{\gamma}/||w||$，约束条件是函数间隔最小达到$\hat{\gamma}$。因为几何间隔和函数间隔存在关系 $\gamma = \hat{\gamma}/||w||$，这将给我们想要的答案。此外，我们也摆脱了$||w|| = 1$ 的限制。但是依然存在的问题是我们要最优化的$\hat{\gamma}/||w||$仍然是一个非凸函数，也就是说我们仍然不能通过常规的优化软件进行优化。

回想一下之前的讨论，我们通过缩放 w 和 b 的值并不影响最终的结果，这一点很关键。我们将通过介绍如何去缩放 wb，使得训练集合的函数间隔一定是 1，即$\hat{\gamma} = 1$。因此 w 和 b 乘以一个常数之后，得到的函数间隔也会相应的乘以该常数。这是一个比例约束，可以通过缩放 wb 来满足。把这一点加入我们上面的优化问题，可以得到最大化$\hat{\gamma}/||w|| = 1/||w||$ 与最小化$||w||^2$ 是一样的效果(这里已经没有了 $||w|| = 1$ 的限制)。因此，我们得到了如下的优化问题：

$$\min_{\gamma,w,b}\ \tfrac{1}{2} * ||w||^2$$
$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1,\ i = 1, \ldots, m$$

我们现在已经把之前的问题转化成一个相对容易解决的问题了。上面的优化是一个凸二次优化问题，而且是线性约束。该结果给出的是最优化间隔分类器。此外，该优化问题可以用商业二次规划(quadratic programming，程序)程序来求解。

在解决这个优化问题之前，我们插入一段关于拉格朗日对偶(Lagrange duality)的介绍。这将使我们进入优化问题的对偶形式，这在后面提到的用核方法来优化最大间隔分类器中起到很重要的作用，能够使该分类器在高维数据中更有效率的工作。此外，对偶形式能够使我们比普通的 QP 代码更有效率的解决上面的优化问题。

7.2. 拉格朗日对偶

我们先把 SVM 和最大间隔分类放在一边，这里主要讨论一下如何求解约束优化问题。考虑如下形式的问题：

$$\min_w \ f(w)$$
$$\text{s.t.} \ \ h_i(w) = 0, i = 1, \ldots, l.$$

在这里我们将介绍拉格朗日乘子法(Lagrange multipliers)。在该方法中，我们定义拉格朗日算符(Lagrangian)如下：

$$L(w, \beta) = f(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

这里的 β_i 称之为拉格朗日乘子(Lagrange multipliers)。求解该式的最优解，只需要使得 L 对 β_i 和 w_i 的偏微分为 0 即可，如下：

$$\frac{\partial L}{\partial w_i} = 0; \quad \frac{\partial L}{\partial \beta_i} = 0$$

之后只要求解出 w 和 β 即可。

在这部分，我们将把这个问题扩展到约束优化问题中，即找到约束优化中不等式来代替拉格朗日乘子法中的等式约束。这里不再讲解拉格朗日对偶理论和详细的证明，但是仍将给出主要的方法和结果，以便我们能够把该方法应用到最优间隔分类器的优化问题上。

考虑如下问题，我们称之为原始优化问题(primal optimization problem)：

$$\min_w f(w)$$
$$\text{s.t.} \ \ g_i(w) \le 0, i = 1, \ldots, k$$
$$h_i(w) = 0, i = 1, \ldots, l.$$

为了解决该问题，我们开始定义广义拉格朗日(generalized Lagrangian)：

$$L(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

这里，α_i 和 β_i 都是拉格朗日乘子。考虑下式：

$$\theta_P(w) = \max_{\alpha,\beta\,:\,\alpha i \geq 0} L(w, \alpha, \beta)$$

这里下标 P 表示原始的(primal)。现在我们赋一些值给 w。如果 w 违反了原始限制(如对于某些 i，$g_i(w) > 0$ 或者 $h_i(w) ! = 0$)，那么我们将得到：

$$\theta_P(w) = \max f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w) = \infty$$

相反的，如果对于某个 w，约束条件能够很好的满足，那么 $\theta_P(w) = f(w)$，因此有：

$$\theta_P(w) = \begin{matrix} f(w) \text{ 如果满足约束条件} \\ \infty \text{ 如果不满足约束条件} \end{matrix}$$

因此，对于所有满足原始约束条件的 w，θ_P 取得相同的值作为我们问题的目标。如果不满足约束条件，则 θ_P 结果为正无穷大。因此，考虑最小优化问题：

$$\min_w \theta_P(w) = \min_w \max_{\alpha,\beta\,:\,\alpha i \geq 0} L(w, \alpha, \beta)$$

我们可以看到这和原始优化问题是相同(具有相同的解)。为了后面的使用，我们定义最优目标值为 $p^* = \min_w \theta_P(w)$,称之为原始问题解。

现在我们考虑一个稍微不同的问题，定义 $\theta_D(\alpha, \beta) = \min_w L(w, \alpha, \beta)$，这里的下标 D 表示对偶(dual)。注意这里与 θ_P 不同，θ_P 是针对 α, β 的最优化函数，而 θ_D 是针对 w 的最优化函数。现在，我们可以得到对偶优化问题：

$$\max_{\alpha,\beta:\alpha i \geq 0} \theta_D(\alpha, \beta) = \max_{\alpha,\beta:\alpha i \geq 0} \min_w L(w, \alpha, \beta)$$

这恰好和原始问题一样，只是我们改变了最大化和最小化的顺序。我们定义该对偶问题的最优化值为 $d^* = \max_{\alpha,\beta\,:\,\alpha i \geq 0} \theta_D(w)$。那么原始问题和该对偶问题有什么联系呢？很容易看出：$d^* = \max_{\alpha,\beta:\alpha i \geq 0} \min_w L(w, \alpha, \beta) \leq \min_w \max_{\alpha,\beta:\alpha i \geq 0} L(w, \alpha, \beta) = p^*$

当然，在某些情况下，存在 $d^* = p^*$ 。因此，我们可以通过解决对偶问题替换原始问题。我们来看看能够替代的情况有哪些。

假设 f 和 g 是凸函数，而且 h_i 是仿射(Affine，类似于线性，但是允许截距项的存在)函数。此外，我们认为 g_i 是可行的，即对于所有的 i，存在 w 使得 $gi(w)<0$ 。针对该假设，一定存在 $w*, \alpha*, \beta*$使得 $w*$为原始问题的解， $\alpha*, \beta*$是对偶问题的解，而且 $p^* = d^* = L(w^*, \alpha^*, \beta^*)$。此外， w^*, α^*, β^*还满足 KKT 条件(Karush-Kuhn-Tucker)，如下所示。

$$
\begin{aligned}
\frac{\partial}{\partial w_i}\mathcal{L}(w^*, \alpha^*, \beta^*) &= 0, \quad i = 1, \ldots, n \\
\frac{\partial}{\partial \beta_i}\mathcal{L}(w^*, \alpha^*, \beta^*) &= 0, \quad i = 1, \ldots, l \\
\alpha_i^* g_i(w^*) &= 0, \quad i = 1, \ldots, k \\
g_i(w^*) &\leq 0, \quad i = 1, \ldots, k \\
\alpha^* &\geq 0, \quad i = 1, \ldots, k
\end{aligned}
$$

此外，如果存在 w^*, α^*, β^*满足 KKT 条件，那么这也是原始问题和对偶问题的解。对于条件中 $\alpha^*_i g_i(w^*) = 0, i=1, \ldots, k$，称之为 KKT 对偶互补条件(dual complementarity condition)。特别的，这里暗含着如果 $\alpha^*_i > 0$，那么 $g_i(w^*) = 0$。之后，这在 SVM 中的支持向量(support vectors)起到关键作用。此外，KKT 对偶互补条件也将在我们 SMO 的收敛测试中用到。

7.3. 最优间隔分类器

之前我们讨论了如下优化问题(即原始优化问题)来求得最优间隔分类器。

$$\min_{\gamma,w,b} \ \frac{1}{2} * ||w||^2$$

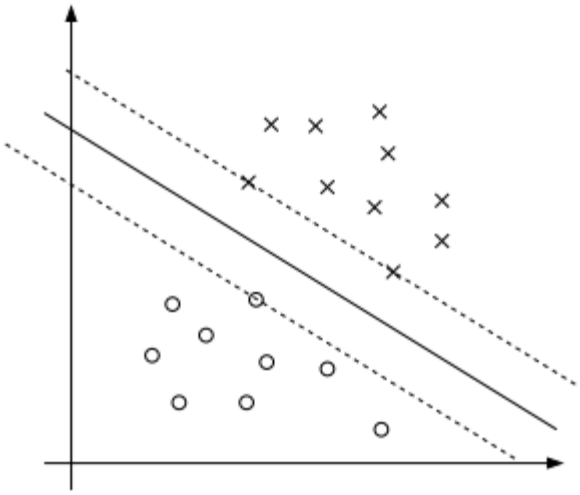
$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, \ldots, m$$

我们可以把约束条件表示成如下形式：

$$g_i(w) = -y^{(i)}(w^T x^{(i)} + b) + 1 \leq 0$$

针对每一个样本，我们都存在该约束条件。根据 KKT 对偶互补条件 $\alpha_i g_i(w) = 0$，我们知道仅对于训练集合中函数间隔恰好等于 1 的样本，才存在线性约束前面的系数 $\alpha_i > 0$，也就是说这些约束式 $g_i(w) = 0$。对于其他不在线上的点($g_i(w)<0$),极值不会在他们所在的范围内取得，$\alpha_i = 0$。

看下面的图：实线是最大间隔超平面，假设×号的是正例，圆圈的是负例。在虚线上的点就是函数间隔是 1 的点，那么他们前面的系数 $\alpha_i > 0$，其他点都是 $\alpha_i = 0$。这三个点称作支持向量。一般而言，支持向量数量相对于整个训练集合而言是非常少的，之后我们会发现这是非常有用的。



之后，得到了我们优化问题的拉格朗日形式，如下：

$$\mathcal{L}(w,b,\alpha)=\frac{1}{2}||w||^2-\sum_{i=1}^{m}\alpha_i\left[y^{(i)}(w^Tx^{(i)}+b)-1\right].$$

注意到这里只有 α_i 和 β_i，因为原问题中没有等式约束，只有不等式约束。接下来我们需要寻找该问题的对偶式。首先，对于给定的 α_i，$L(w,b,\alpha)$仅受 w 和 b 影响，我们需要最小化 $L(w,b,\alpha)$，只需要让 L 对 w 和 b 的偏微分为 0。对 w 求偏微分如下：

$$\nabla_w\mathcal{L}(w,b,\alpha)=w-\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}=0$$

由此得到 $w=\sum_{i=1}^{m}a_i y^{(i)}x^{(i)}$，之后对 b 求偏微分得到：

$$\frac{\partial}{\partial b}\mathcal{L}(w,b,\alpha)=\sum_{i=1}^{m}\alpha_i y^{(i)}=0.$$

我们把得到的 w 值带回到构造的拉格朗日函数中，化简如下：

$$\mathcal{L}(\mathrm{w},\mathrm{b},\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}\alpha_i\left[y^{(i)}\left(w^Tx^{(i)}+b\right)-1\right]$$

$$=\frac{1}{2}\mathrm{w}^T\mathrm{w}-\sum_{i=1}^{m}\alpha_iy^{(i)}w^Tx^{(i)}-\sum_{i=1}^{m}\alpha_iy^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$$=\frac{1}{2}\mathrm{w}^T\sum_{i=1}^{m}\alpha_iy^{(i)}x^{(i)}-\sum_{i=1}^{m}\alpha_iy^{(i)}w^Tx^{(i)}-\sum_{i=1}^{m}\alpha_iy^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$$=\frac{1}{2}\mathrm{w}^T\sum_{i=1}^{m}\alpha_iy^{(i)}x^{(i)}-w^T\sum_{i=1}^{m}\alpha_iy^{(i)}x^{(i)}-\sum_{i=1}^{m}\alpha_iy^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\mathrm{w}^T\sum_{i=1}^{m}\alpha_iy^{(i)}x^{(i)}-\sum_{i=1}^{m}\alpha_iy^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\mathrm{w}^T\sum_{i=1}^{m}\alpha_iy^{(i)}x^{(i)}-b\sum_{i=1}^{m}\alpha_iy^{(i)}+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\left(\sum_{i=1}^{m}\alpha_iy^{(i)}x^{(i)}\right)^T\sum_{i=1}^{m}\alpha_iy^{(i)}x^{(i)}-b\sum_{i=1}^{m}\alpha_iy^{(i)}+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\sum_{i=1}^{m}\alpha_iy^{(i)}\left(x^{(i)}\right)^T\sum_{i=1}^{m}\alpha_iy^{(i)}x^{(i)}-b\sum_{i=1}^{m}\alpha_iy^{(i)}+\sum_{i=1}^{m}\alpha_i$$

$$=-\frac{1}{2}\sum_{i=1,j=1}^{m}\alpha_iy^{(i)}\left(x^{(i)}\right)^T\alpha_jy^{(j)}x^{(j)}-b\sum_{i=1}^{m}\alpha_iy^{(i)}+\sum_{i=1}^{m}\alpha_i$$

$$=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1,j=1}^{m}y^{(i)}y^{(j)}\alpha_i\alpha_j(x^{(i)})^Tx^{(j)}-b\sum_{i=1}^{m}\alpha_iy^{(i)}$$

由于$\sum_{i=1}^{m}a_iy^{(i)}=0$，因此最终得到：

$$\mathcal{L}(w,b,\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{m}y^{(i)}y^{(j)}\alpha_i\alpha_j(x^{(i)})^Tx^{(j)}.$$

这里我们把向量内积 $(x^{(i)})^Tx^{(j)}$ 表示为$<(x^{(i)}),x^{(j)}>$。此时的拉格朗日函数只包涵了变量 α_i，求解出 α_i 之后即可以得到 w 和 b。根据对偶问题的求解，我们得到了如下的最大化问题：

$$\begin{aligned}\max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle. \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0,\end{aligned}$$

这里我们需要考虑 $p^* = d^*$ 的条件，即 KKT 条件。因为目标函数和线性约束都是凸函数，而这里不存在等式约束 h，存在 w 使得对于所有的 i，使得 $g_i(w) <= 0$，因此存在 w^* 和 a^*使得 w^*是原问题的解， a^*是对偶问题的解。这里求解的 a_i 就 a^*如果求解出了 a_i，根据 $w = \sum_{i=1}^{m} a_i y^{(i)} x^{(i)}$ 即可以求解出 w(也就是 w^*，原问题的解)。同样根据下式得到 b：

$$b^* = -\frac{\max_{i:y^{(i)}=-1} w^{*T} x^{(i)} + \min_{i:y^{(i)}=1} w^{*T} x^{(i)}}{2}.$$

求解得到 b，即距离超平面最近的正的函数间隔要等于距离超平面最近的负的函数距离。关于上面的对偶问题如何求解，在下一讲中我们将采用 SMO 算法来阐明。

此外，我们额外考虑一下 w 的求解公式。假设我们已经通过训练集合得到了模型的各个参数，现在要预测一个新的输入 x，我们需要计算 w^T+b ，如果得到的值大于 0，那么 y=1。这里我们把 $w = \sum_{i=1}^{m} a_i y^{(i)} x^{(i)}$ 考虑进去，得到如下结果，

$$\begin{aligned} w^T x + b &= \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \right)^T x + b \\ &= \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b. \end{aligned}$$

由此可以看出，与之前计算 w^T+b 不同，这里只要计算新的样本和训练样本的内积即可。当然，从前面的 KKT 条件中知道，只有支持向量的 $a_i > 0$，其他情况 $a_i = 0$，其他情况下 $a_i=0$。因此，我们只需要进算新的样本和支持向量的内积就可以。

通过优化问题的对偶形式，我们更深入的看到了需要优化问题的结构，并且找到了能够适合于特征向量内积的算法。在下一讲，我们会讨论核方法在高维数据中使用以及求解支持向量机的算法。

8. 顺序最小化算法

8.1. 核方法

考虑这样一个问题，如果输入 X 是房子的面积，我们要使用回归方法预测房子的价格。从样本点的分布中，我们看到三次方程(即使用 x，x^2，x^3 进行回归曲线拟合)能够更好的拟合数据。为了区分这两组不同的变量(前者为 x，后者为(x，x^2，x^3))，我们称问题的特征 x 为原始特征(attribute)。当我们把原始特征扩展到一些新的变量的时候，我们称这些新生成的变量为输入特征(features，当然，不同的人对这两种变量也存在不同的命名。但英文中主要采用 attribute 和 feature 进行区分)。我们使用 φ 表示特征映射(feature mapping)，即把原始特征映射到特征。对于刚才的例子，$\varphi = [x, x^2, x^3]^T$.

与其直接使用原始特征来构建 SVM，我们可能更希望使用一些特征映射后的特征 $\varphi(x)$(一方面是上面提到的为了更好的拟合，另一个重要原因是样本存在线性不可分的情况，而映射到高维空间后，往往就可分了)。因为算法中原始特征是内积形式$<x, z>$，采用特征映射之后，即可以表示为$<\varphi(x), \varphi(z)>$。因此，对于给定的特征映射 φ，我们定义核函数(Kernel)为：$K(x,z) = <\varphi(x), \varphi(z)> = \varphi(x)^T\varphi(z)$。因此，不管我们的原始特征是什么，我们都可以简单的使用核函数 $K(x,z)$进行替换，之后我们的算法就相当于使用了特征 φ。

对于给定的特征 φ，我们很容易通过计算 $\varphi(x)$和 $\varphi(z)$的内积来得到 $K(x, z)$。但是更多的情况是，$K(x,z)$的计算量很大，而 $\varphi(x)$ 本身就很复杂(毕竟从低维映射到了高纬，得到的是高维向量)。在算法中使用核方法的情况下，SVM 是在给定特征的高维度特征空间下学习得到，并没有直接或明确的得到特征向量 $\varphi(x)$。

考虑下面一个例子，$x,z \in R^n$，$K(x, z) = (x^Tz)^2$，对于该式，我们可以转换如下：

$$\begin{aligned} K(x,z) &= \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_i z_i\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) \end{aligned}$$

因此，我们看出 $K(x, z) = \varphi(x)^T\varphi(z)$ 中，特征映射 φ 是(这里仅展示对于 $n=3$ 的情况)，$\varphi = [\, x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3 \,]^T$。

这里注意的是，如果计算高维度 φ(x)情况下的时间复杂度为 $O(n^2)$，而实际 K(x, z)只需要要计算原始特征的内积方法即可(上面推导的结果，时间复杂度为 O(n))。

此外，考虑下面情况

$$\begin{aligned} K(x,z) &= (x^T z + c)^2 \\ &= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) + \sum_{i=1}^{n} (\sqrt{2c}x_i)(\sqrt{2c}z_i) + c^2. \end{aligned}$$

这里的特征映射 φ(仍然考虑 n=3 的情况)为：$\varphi(x) = [x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3, \text{sqrt}(2c)x_1, \text{sqrt}(2c)x_2, \text{sqrt}(2c)x_3, c]^T$ 。这里的 c 控制着 x_i(一阶)和 x_ix_j(二阶)的相对权重。

对于更一般的情况，核函数 $K(x, z) = (x^Tz + c)^d$ 对应的映射空间维度是 $C(n + d, d)$。然而，对于 $O(n^d)$维空间，计算 K(x, z) 仍然只需要 O(n) 时间。因此，我们无需直接计算高维度的映射特征向量。

现在，我们采用一些不同的视角来看核函数。直观上来看，如果 φ(x) φ(z)越接近，那么 $K(x, z) = \varphi(x)^T\varphi(z)$ 也会越大。相反，如果 φ(x) φ(z)越分开，那么 K(x, z)也会越小。因此，我们可以认为从某种程度上而言，表示了 φ(x) φ(z)的相似性，或者说 x 和 z 的相似性。

鉴于该直观，假设现在正在解决一个机器学习问题，我们想到了一个核方法 K(x,z)，而该核方法能够合理的测量 x 和 z 的相似性。

举例子而言，$K(x, z)= \exp(-||x-z||^2 / (2\sigma^2))$ 。这是一个合理的测量 xz 间的相似度，当 x 和 z 越相近的时候值越接近 1，当 x 和 z 越分开的时候值越接近于 0 。那么我们可以定义这样的 K 作为 SVM 中的一个核方法吗？当然，在这里例子中答案是肯定的(该核称之为高斯核(Gaussian kernel)，又称为径向基函数(Radial Basis Function 简称 RBF，它能够把原始特征映射到无穷维)。更广的说，给定一个函数 K，我们如何判定是否是有效的核函数？即对于特征映射 φ，是否使得所有的 x, z 均有 $K(x, z) = \varphi(x)^T\varphi(z)$?

假设 K 是一个对应于特征映射 φ 的有效的核函数。现在考虑对于有限数据集，含有 m 个点(不一定是训练集合)$\{x^{(1)}, \ldots, x^{(m)}\}$，K 是一个 m*m 的方阵且 $K_{ij} = K(X^{(i)},x^{(j)})$，该矩阵称之为核矩阵(Kernel matrix)。注意：为了方便，我们重用了 K，既是核函数 K(x,z)，也是核矩阵 K。

如果 K 是一个有效核，那么 $K_{ij} = K(X^{(i)},x^{(j)}) =\varphi(X^{(i)})^T\varphi(x^{(j)}) =\varphi(,x^{(j)})^T\varphi(,x^{(i)}) = K(X^{(j)},x^{(i)}) = K_{ji}$ 因此核矩阵 K 一定是对称的。此外，令 $\varphi_k(x)$ 表示向量 $\varphi(x)$的第 k 个值，对于任意向量 z，有如下：

$$
\begin{aligned}
z^T K z &= \sum_i \sum_j z_i K_{ij} z_j \\
&= \sum_i \sum_j z_i \phi(x^{(i)})^T \phi(x^{(j)}) z_j \\
&= \sum_i \sum_j z_i \sum_k \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\
&= \sum_k \sum_i \sum_j z_i \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\
&= \sum_k \left(\sum_i z_i \phi_k(x^{(i)}) \right)^2 \\
&\geq 0.
\end{aligned}
$$

倒数第二步和前面证明 $K(x, z) = (x^Tz)^2$ 用到的方法是一样的。而 z 是任意的，这就表明了 K 是半正定矩阵($K\geq 0$)。因此，得到了 K 是有效核(对于一些特征映射 φ)的必要条件是 $K \in R^{m\times m}$ 是半正定矩阵。此外，这不仅仅是必要条件，同时也是 K 是一个有效核(也被称为 Mercer kernel)的充分条件。

Mercer 定理：如果函数 K 是 $R^n\times R^n \rightarrow R$ 上的映射(也就是从两个 n 维向量映射到实数域)。那么如果 K 是一个有效核函数，那么当且仅当对于$\{x^{(1)}, \ldots, x^{(m)}\}$ $(m < \infty)$，其相应的核函数矩阵是对称半正定的。

给定一个函数 K，除了试图找到一个与之对应的特征映射 φ 外，该定理提供了另一种测试 K 是否是有效核的方法。

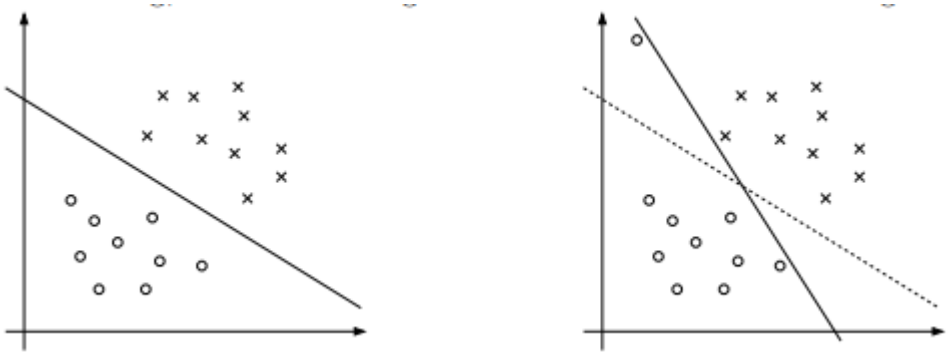
这里举一些核方法的例子。比如对于手写数字识别，输入是 16*16 的像素的手写数字 0-9 的图片，我们需要一个模型来判断一个图片上的数据是多少。采用简单的线性核函数 $K(x, z) = (x^Tz)^d$ 或者高斯核，SVM 在该问题上都能得到非常好的结果。这是一个非擦汗那个令人吃惊的结果，因为输入特征 x 是一个 256 维的向量(各个像素点的值)，而系统并没有任何先验知识，甚至不知道哪个像素点和哪个点连在一起。另外一个例子是，我们试图分类的对象是字符串(比如说，x 是一串氨基酸列表，可以组成蛋白质)，对于很多学习算法而言，很难组成一个合理的特征集，尽管不同的字符串有不同的长度。然而，我们使用特征映射 $\varphi(x)$为特征向量，表示 x 中长度为 k 的子字符的出现次数。如果我们英文字母

组成的字符串，那么就会有 26^K 个这样的子字符串。因此，φ(x)就是一个 26^K 维度的向量，即使对于中等的 k，这也是非常大的($26^4 \approx 460000$)。可是，使用字符匹配算法，有效的计算 $K(x,z) = \varphi(x)^T \varphi(z)$，使得我们能够在这么高维度特征空间下实现而不用直接的在高维度下计算。

此外，需要注意的事情是核方法不仅仅是在 SVM 中有着重要的地位。对于任何机器学习算法，你都可以把输入特征的内积<x,z>使用核方法改写成 K(x,z)，使得算法在高维度下也能很高效的工作。比如把核方法应用在感知器上得到一个核感知器算法。在以后降到的很多算法都会接受核方法。

8.2. 正则化和线性不可分

目前为止的 SVM 都是假设数据是线性可分的。尽管采用特征映射 φ 到高维空间能够增加数据可分的似然性，但是仍然不能保证一定是线性可分的。而且有些情况下我们异常值的影响而不能保证分割的超平面恰好是需要的。举个例子说，如下图：左边的图展示了一个最优的分类器，而当出现一个异常值的时候(如右图所示)，引起了决策边界的转动，使得间隔变小。



为了使算法能够更好的适应非线性数据集和减少对异常值的敏感性，我们前的最优化式子做出一些改变(使用正则化 ℓ (regularization))：

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m \\ & \xi_i \geq 0, \quad i = 1, \ldots, m. \end{aligned}$$

因此，样本现在允许存在函数间隔小于 1 的情况了。如果一个样本的函数间隔 $1 - \xi_i$ (其中 $\xi > 0$，称之为松弛变量)，那么我们会使得目标函数成本增加 $C*\xi_i$。C 是离群点的权

重，C 越大表明离群点对目标函数影响越大，也就是越不希望看到离群点。参数 C 是控制着双重目标的相对权重，其一还是使得||w||^2 较小(也就是之前说的使得间隔最大)，其次是使得更多的样本的函数间隔至少是 1 。那么就像之前那样，我们也可以得到如下的拉格朗日公式：

$$\mathcal{L}(w,b,\xi,\alpha,r) = \frac{1}{2}w^T w + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i\left[y^{(i)}(x^T w + b) - 1 + \xi_i\right] - \sum_{i=1}^{m} r_i\xi_i.$$

这里的 α_i 和 r_i 是拉格朗日乘子。回想我们在拉格朗日对偶中提到的求法，先写出拉格朗日公式(如上式)，然后将其看作是变量 w 和 b 的函数，分别对其求偏导，得到 w 和 b 的表达式，然后代入公式中，求代入后公式的极大值，得到最终的结果如下：

$$\begin{aligned}
\max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)}, x^{(j)}\rangle \\
\text{s.t.} \quad & 0 \le \alpha_i \le C, \quad i = 1,\ldots,m \\
& \sum_{i=1}^{m}\alpha_i y^{(i)} = 0,
\end{aligned}$$

对比之前的公式，我们仍然有 $w = \sum_{i=1}^{m} a_i y^{(i)} x^{(i)}$ ，之后解决了对偶问题之后，我们可以同样通过下式得到我们的预测结果。

$$\begin{aligned}
w^T x + b &= \left(\sum_{i=1}^{m}\alpha_i y^{(i)} x^{(i)}\right)^T x + b \\
&= \sum_{i=1}^{m}\alpha_i y^{(i)}\langle x^{(i)}, x\rangle + b.
\end{aligned}$$

而需要注意的是，引入正则化 ℓ_1 后对偶问题的唯一变化是原始约束 $0 \le \alpha_i$ 变成了 $0 \le \alpha_i \le C$ 。而 b^*的计算也需要修改，改变的结果在 SMO 算法中介绍。先看看 KKT 条件(在 SMO 算法中用于测试算法是否收敛)的变化：

$$\begin{aligned}
\alpha_i = 0 \quad &\Rightarrow \quad y^{(i)}(w^T x^{(i)} + b) \ge 1 \\
\alpha_i = C \quad &\Rightarrow \quad y^{(i)}(w^T x^{(i)} + b) \le 1 \\
0 < \alpha_i < C \quad &\Rightarrow \quad y^{(i)}(w^T x^{(i)} + b) = 1.
\end{aligned}$$

第一个式子表明在两条间隔线外的样本点前面的系数为 0，离群样本点前面的系数为 C，而支持向量(也就是在超平面两边的最大间隔线上)的样本点前面系数在(0,C)上。通过

KKT 条件可知，某些在最大间隔线上的样本点也不是支持向量，相反也可能是离群点。现在剩下的问题，就是如何用算法实现该对偶问题的求解了。

8.3. 坐标上升法

SMO(sequential minimal optimization) 算法是由 John Platt 给出的一个求解用于由 SVM 衍生出的对偶问题的有效解法。在讲解 SMO 算法之前，我们先引入一个有趣的算法——坐标上升法(coordinate ascent algorithm) 。

假如我们解决如下无约束的优化问题：

$$\text{Max } W(\alpha_1, \alpha_2, \ldots, \alpha_m)$$

这里我们认为 W 只是 α_i 的函数，这里先不考虑这个问题和 SVM 有什么关系。我们目前已经掌握了两种优化算法：梯度下降法和牛顿法。这里我们使用新的算法，称之为坐标上升法。

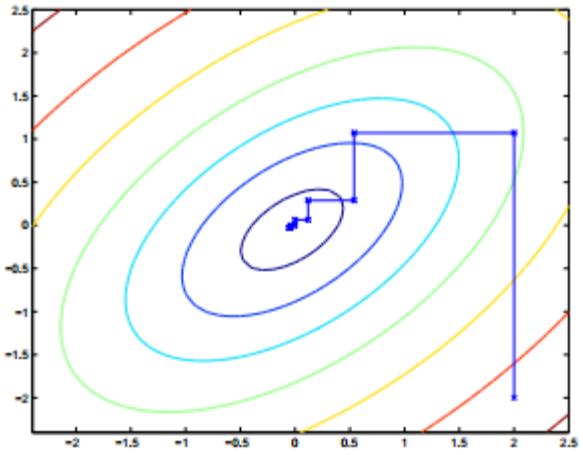
循环直到收敛：{

对于每一个 i = 1,..., m {

$$\alpha_i = \arg\max_{\alpha i} \text{Max } W(\alpha_1, \alpha_2, \ldots, \alpha_m) \quad \}\}$$

在该算法最里层的循环里，我们保持其他的变量(α_j (j !=i))不变，仅更新 α_i，即认为 W 仅受 α_i 影响，而不受 α_j (j !=i) 的影响。该循环按照 $\alpha_1, \alpha_2, \ldots, \alpha_m, \alpha_1, \alpha_2, \ldots$ 的顺序不断的循环优化。当然，我们也可以按照自定义顺序去循环，比如按照下一个是能够使得 W(α) 增长最大的变量。

如果函数 W 能够在该循环中有效的得到最优解，那么坐标上升法将是一个相当有效的算法。下图是坐标上升的一个过程：途中椭圆是指我们要优化的二次函数的等高线。坐标上升法起始点在 (2，-2)，图中的直线式迭代优化的路径，可以看到每一步都会向最优值前进一步，而且前进路线是平行于坐标轴的，因为每一步只优化一个变量。



8.4. 序列最小算法(SMO)

SMO 算法由 Microsoft Research 的 John C. Platt 在 1998 年提出，并成为最快的二次规划优化算法，特别针对线性 SVM 和数据稀疏时性能更优。关于 SMO 最好的资料就是他本人写的《Sequential Minimal Optimization A Fast Algorithm for Training Support Vector Machines》了。这里我们简单的讨论 SMO 算法，给出 SMO 算法的基本流程和推导过程。

首先，我们的优化问题如下：

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle. \\ \text{s.t.} \quad & 0 \le \alpha_i \le C, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0. \end{aligned}$$

即需要解决的是在参数 $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ 上求的最大值 W 的问题，其中 x，y 均是已知数，而 C 是由我们预先设定的，也是已知。

我们先假设 α_i 满足了上式优化问题的约束，根据坐标上升法，我们首先固定 $\alpha_2, \ldots,$ α_m 不变，然后在 α_1 上面求得 W 的极值，并更新 α_1。但是，这个思路有问题，因为固定了 α_1 以外的所有参数，那么根据 $\sum_{i=1}^{m} a_i y^{(i)} = 0$ 得到 $\alpha_1 y^{(1)} = -\sum_{i=2}^{m} a_i y^{(i)}$ (这里的 $y^{(1)} \in \{-1, 1\}$，$(y^{(1)})^2 = 1$，因此可以直接移动到等式右边)，即得到是一个 α_1 固定值。

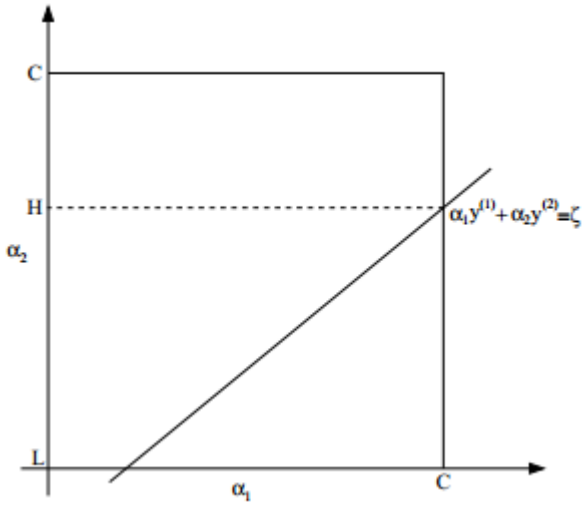
因此，如果我们想要更新 α_i 的话，必须至少选择两个参数进行更新以满足约束条件。这就促生了 SMO(Sequential Minimal Optimization)算法，主要步骤如下：

循环直到收敛{

1、选择一对(或一些对) α_i 和 α_j 准备下一步更新(采用启发式算法(后面会讲到，即选择能够使我们向全局最优解取得最大进步的两个参数))
2、在使得其他 α 不变的下，求得 α_i 和 α_j 使得 W(α)达到最优解，注意求得 α_i 即可， α_j 可以用 α_i 来表示}

为了测试该算法是否收敛，我们可以采用是否在一定的公差 τ 内满足了 KKT 条件。τ 是指收敛公差参数，一般都设定在 0.01 或 0.001 。SMO 之所以高效就是因为在固定其他参数后，对一个参数优化过程很高效。接下来，我们讨论一下 SMO 是如何高效的更新参数。

我们先认为 α_i 满足约束条件满足为 $0 \le \alpha_i \le C$ i=1, ..., m 和$\sum_{i=1}^{m} a_i y^{(i)} = 0$，这里先限制 α_2，..., α_m 固定，现在需要有重新优化 $W(\alpha_1, \alpha_2, ...\alpha_m)$，其中变量仅有 α_1 和 α_2，根据约束条件，我们有 $\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = -\sum_{i=3}^{m} a_i y^{(i)}$ 。该等式的右边是固定值，我们可以用 ζ 来表示，那么等式可以表示为 $\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = \zeta$ 。当 $y^{(1)}$和 $y^{(2)}$异号时，也就是一个为 1，一个为-1 时，他们可以表示成一条直线，斜率为 1。之后，我们可以用下图来表示在 α_1 和 α_2 上的约束关系。



从约束条件看，α_1 和 α_2 一定在正方形[0, C]×[0, C]里面，且一定在图中直线 $\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = \zeta$ 上。从中我们可以得到：$L \le \alpha_2 \le H$；否则(α_1, α_2) 就不能同时满足正方形和直线约束。这个例子中 L=0，不过这取决于直线的位置，不同的位置 L 和 H 值不一样，即会存在一个较大的值 H 和较小的值 L。我们将 α_1 用 α_2 表示：$\alpha_1 = (\zeta - \alpha_2 y^{(2)}) y^{(1)}$。因此，$W(\alpha)$ 可以表示如下：$W(\alpha_1, \alpha_2, \ldots, \alpha_m) = W((\zeta - \alpha_2 y^{(2)}) y^{(1)}, \alpha_2, \ldots, \alpha_m)$。

固定 $\alpha_3, \ldots, \alpha_m$ 不变，之后 w 就只是关于 α_2 的函数，即可以表示为 $a\alpha_2^2+b\alpha_2+c$。如果我们忽略正方形的边界约束(即 $L \le \alpha_2 \le H$)，可以很容易通过导数为 0 求得最大化 W 的 α_2 值。我们使用 $\alpha_2^{new,unclipped}$ 表示最终的 α_2 结果。如果考虑了边界的话，那么结果如下：

$$\alpha_2^{new} = \begin{cases} H & \text{if } \alpha_2^{new,unclipped} > H \\ \alpha_2^{new,unclipped} & \text{if } L \le \alpha_2^{new,unclipped} \le H \\ L & \text{if } \alpha_2^{new,unclipped} < L \end{cases}$$

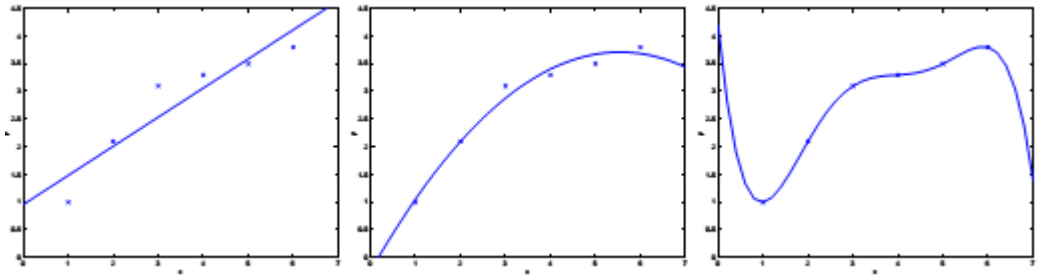
在得到 α_2^{new} 之后，可以很容易得到 α_1^{new}。以下是关于 Platt 的论文 http://research.microsoft.com/en-us/um/people/jplatt/smo-book.pdf。也可以参考斯坦福的笔记 http://cs229.stanford.edu/materials/smo.pdf。

9. 经验风险最小化

之前几章，我们主要讨论不同的思考角度下产生的机器学习算法。如果只是掌握几种算法的话，那么还是停留在纸上谈兵的阶段。比如在生活中，我们使用了贝叶斯算法，发现效果并不是非常好，那么我们应该怎么做呢？是修改算法还是怎么做？这是区分一个人是懂得机器学习与否的一个关键点。现在，我们从“学习理论”(Learning Theory)来更本质的思考机器学习的一些问题。

9.1. 偏差和方差的权衡

当我们讨论到线性回归的时候，我们可以采用简单的模型如 $y = \theta_0 + \theta_1 x$，也可以采用相对复杂的模型，如 $y = \theta_0 + \theta_1 x + \cdots \theta_5 x^5$，结果如下图所示。



采用 5 阶多项式拟合数据(如上图右侧图所示)并不是一个好的模型，因为尽管该 5 阶多项式非常好的拟合了训练样本中的 x 和 y 之间的关系，但是在非训练数据集上却并不能得到非常好的结果。换句话说，从训练样本中得到的经验知识并不能很好的泛化(generalize)到其他的情况(即不能够很好的适用于非训练样本的情况)。一个假设模型的泛化误差(generalization error)未必就是训练集合误差的期望。

不管是左边的还是右边的模型，都具有较大的泛化误差。然而，这两个模型却有明显的不同之处。如果 y 和 x 之间的关系并不是线性的，那么即使我们使用特别大量的数据进行来拟合线性模型，得到的结果仍然不能够刻画数据中真正的结构。因此，我们定义：一个模型，即使是在非常大的训练集(可以认为是无限大)拟合得到的，那么其偏差(bias)仍为其泛化误差的期望。因此对于提到上面的问题，线性模型具有非常大的偏差，是欠拟合(即不能很好的刻画数据的结构)。

除了偏差，泛化误差还存在另一种情况，即方差(variance)。如上图右侧图，采用 5 阶多项式拟合，会存在很大的风险即这些参数恰好非常适合于我们有限的较小的训练集合，但是却不能反映出 x 和 y 之间的更加广泛的模式。因为在训练集合中得到了一些较为虚假的模式，因此我们可能得到一个具有非常大泛化误差的模型。这样的情况下，我们称之为模型具有较大的方差。

通常情况下，我们需要权衡偏差和方差。如果我们的模型太简单，模型的参数特别少，那么我们可能会有较大的偏差(但是有较小的方差)。如果我们的模型特别复杂，具有较多的参数，那么模型很有可能具有较大的方差(但是有较小的偏差)。在上面的例子中，采用 3 阶多项式拟合是相对于左边和右边均较优的模型。

9.2. 预备知识

从现在开始，我们正式进入”学习理论“(learning theory)。这部分讨论不仅仅是有趣和具有启发性的，同时能够帮助我们更好的理解一些关于如何在不同的设置下使用最好的学习算法的规则。我们也将寻找一些疑问的答案：第一，我们能否在偏差和方差之间做出一个合理的权衡？这将直接引导我们关于模型选择的问题，举例子说，能够自动的选择合适阶次的多项式来拟合训练集。第二，在机器学习中，我们真正关心的是泛化误差，而绝大多数学习算法都是使用训练集合得到的。为什么在训练集合有较优的表现会告诉我们一些关于泛化误差的信息呢？特别的，我们能否得到训练误差和泛化误差之间的一些关系呢？最后，是否存在一些条件，让我们能够证明这些学习算法能够很好的工作呢？

我们先从两个简单的定律说起。

定律 1(联合边界(union bound))：对于 k 个不同的事件 $A_1, A_2, \ldots, A_k$(不一定相互独立的)，存在 $P(A_1 \cup \cdots \cup A_k) \leq P(A_1) + \ldots + P(A_k)$

在概率论理论中，联合边界定律通常作为公理来说，同时也能够直觉到：包含 k 个事件的事件发生的可能性最大不超过 k 个事件独立发生的概率和。

定律 2(Hoeffding 不等式(Hoeffding inequality))：假设 $Z_1, \ldots, Z_m$ 独立同分布(iid)变量，服从伯努力分布，如 $P(Z_i = 1) = \varphi, P(Z_i = 0) = 1 - \varphi$。这些随机变量的均值，可以表示为 $\hat{\varphi} = (1/m)\sum_{i=1}^{m} Z_i$ ，对于任意 $\gamma > 0$，存在：

$$P(|\varphi - \hat{\varphi}| > \gamma) \leq 2\exp(-2\gamma^2 m)$$

该定律在机器学习理论中也被称之为 Chernoff 边界(Chernoff bound)，该定律表明：在 m 非常大的情况下，对于 m 个服从伯努力分布的随机变量的均值 $\hat{\varphi}$，会非常接近参数 φ 的值。举个例子，对于一枚有偏的硬币(一面是头，另一面没有头)，出现头的概率是 φ，如果你投掷 m 次，然后计算一下头出现的次数，那么我们估计 φ 的结果就会有很高的置信度。

使用这两个定律，我们能够证明学习理论中最重要和最深的一些结论。为了方便我们的表达，我们的讨论目前限定在二元分类，即类别 $y \in \{0, 1\}$，这里提到的表达都可以泛化到其他情况，包括回归和多元分类等等问题。

假设给定的训练集为 $S = \{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$，其中每个样本$(x^{(i)}, y^{(i)})$独立同分布，服从于概率分布$\mathcal{D}$，对于假设 h，我们定义训练误差(training error，也可以称之为经验风险(empirical risk)、经验误差(empirical error))为：

$$\hat{\varepsilon}(h) = \frac{1}{m}\sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y^{(i)}\}$$

这只是训练集合中假设 h 错误分类的样本比例。当我们想知道对于特定集合 S 的训练误差，可以表示为 $\hat{\varepsilon}_S(h)$ 我们也定义泛化误差为：$\varepsilon(h) = P(x,y) \sim \mathcal{D}(h(x^{(i)}) \neq y^{(i)})$，即对于从分布$\mathcal{D}$产生的一个新样本，h 错误分类的概率。

我们已经假设了训练集合是从同一个分布$\mathcal{D}$产生的，而该分布是未知，也是我们评估假设的分布(在泛化误差中定义的 h)。有时，这也被称之为 PAC 假设。(PAC，probably approximately correct，概率渐进正确，是一个框架和一系列假设，这些结论是学习理论中已经证明的一些结果。当然，这些关于训练集合和测试集合的独立同分布假设，在训练集合上是最重要的)。

考虑线性分类器(如 PLA，感知器学习)，假设 $h_\theta(x) = 1\{\theta^T x \geq 0\}$，一个合理的拟合参数 θ 的方法是什么？一个尝试是尝试最小化训练误差，然后选择：$\hat{\theta} = \arg\min_\theta \hat{\varepsilon}(h_\theta)$

我们称这个过程为经验风险最小(ERM，empirical risk minimization)，根据该学习算法得到的最终假设是 $\hat{h} = h_{\hat{\theta}}$。我们认为 ERM 是最基本的学习算法。在本节的讨论，主要是针对该学习算法进行讨论。(比如 logistic 回归也可以近似的看成是经验风险最小化算法)。

在学习理论的学习过程中，我们为了从区分模型是否是线性分类器或者是辨别具有特定参数的假设等问题中抽象出一些假设。我们定义假设集合(hypothesis class)$\mathcal{H}$为一个学习算法下的所有假设。如对于线性分类器，那么$\mathcal{H} = \{h_\theta : h_\theta(x) = 1\{\theta^T x \geq 0\}, \theta \in R^{n+1}\}$，$\mathcal{H}$是一个对于输入域 X 而言，决策边界都是直线的模型的集合，即线性分类器的集合。更宽泛的说，比如神经网络，我们可以认为$\mathcal{H}$是具有一些神经网络结构下的所有分类器。经验风险最小化可以认为是从假设集合$\mathcal{H}$中找到一个最优的假设 h，即

$$\hat{h} = \arg\min_{h \in H} \hat{\varepsilon}(h)$$

9.3. 有限情况下的$\mathcal{H}$

首先我们从有限集的假设集合$\mathcal{H} = \{h_1, \ldots, h_k\}$问题开始。因此$\mathcal{H}$就是 k 个函数，这些函数能够把输入 X 映射到 {0,1}，而经验风险最小化原理就是从这 k 中选择一个能够使得训练集合误差最小的$\hat{h}$。

首先我们想要保证$\hat{h}$ 的泛化误差。我们的策略主要分成两步：第一、展示$\hat{\varepsilon}(h)$ 是对于所有 h 的 $\varepsilon(h)$ 合理的估计。其次，我们要找到 $\hat{h}$ 泛化误差的上限。对于第一个，对于

$h_i \in \mathcal{H}$，考虑一个伯努力变量 Z，对于样本$(x, y) \sim D$，$Z = 1\{h_i(x) \neq y\}$。同样的，我们定义 $Z_j = 1\{h_i(x^{(j)}) \neq y^{(j)}\}$，因样本均是来自 D，所以 Z 是独立同分布。因此，对于一个随机样本，错误分类的概率 $\varepsilon(h)$ 可以通过 Z 的期望值来表示。更一般的，训练误差可以表示为：$\hat{\varepsilon}(h_i) = \frac{1}{m}\sum_{j=1}^{m} Z_j$，这里的$\hat{\varepsilon}(h_i)$就是 m 个独立同分布(服从伯努力分布)的随机变量 Z_j 的均值。因此，使用 Hoeffding 不等式，得到：

$$P\left(|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma\right) \leq 2\exp(-2\gamma^2 m)$$

这表明了对于特定的 h_i，如果 m 足够大，那么训练误差将会以很高的概率接近泛化误差。但是，我们并不想只是保证特定的 h_i 下 $\varepsilon(h_i)$会接近$\hat{\varepsilon}(h_i)$(伴随着很高的概率)。我们希望这对于所有 $h \in \mathcal{H}$都成立。为了证明，我们用 A_i 表示事件 $|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma$，对于任意 A_i，我们已经证明了 $P(A_i) \leq 2\exp(-2\gamma^2 m)$。之后采用联合边界(union bound)，得到：

$$\begin{aligned} P(\exists h \in \mathcal{H}.|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(A_1 \cup \cdots \cup A_k) \\ &\leq \sum_{i=1}^{k} P(A_i) \\ &\leq \sum_{i=1}^{k} 2\exp(-2\gamma^2 m) \\ &= 2k\exp(-2\gamma^2 m) \end{aligned}$$

如果两遍都用 1 减去的话，那么有如下结果：

$$\begin{aligned} P(\neg\exists h \in \mathcal{H}.|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(\forall h \in \mathcal{H}.|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| \leq \gamma) \\ &\geq 1 - 2k\exp(-2\gamma^2 m) \end{aligned}$$

其中'¬'的意思是取反。因此，在概率至少是 $1- 2k\exp(-2\gamma^2 m)$的情况下，我们有对于所有的 $h \in \mathcal{H}$，存在 $\varepsilon(h)$在 $\hat{\varepsilon}(h)$的 γ 范围内。这称之为一致收敛(uniform convergence)结果，因为这是同时对于所有的 $h \in \mathcal{H}$都有的边界。

对于上面我们所做的，就是对于特定的 m 和 γ，给出了 $h \in \mathcal{H}$的概率边界，即 $|\varepsilon(h) - \hat{\varepsilon}(h)| > \gamma$。这里有三个我们感兴趣的变量 m, γ 和错误率，我们可以通过任意两个求得第三个。举个例子而言，对于给定的 γ 和 $\delta > 0$，那么 m 至少要多大才能保证训练误差将在泛化误差的 γ 范围内的概率至少是 $1 - \delta$。由此可得 $\delta = 2k\exp(-2\gamma^2 m)$，得到 $m \geq 1/(2\gamma^2) * \log(2k/\delta)$。因此，对于 $h \in \mathcal{H}$，我们至少有 $1 - \delta$ 的概率认为$|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$。该边界

告诉了我们训练样本至少要达到多少。对于某些算法或方法，为了达到一定的效果而要求一定的样本数量，称之为算法的样本复杂度(sample complexity)。

边界的核心关键点是，为了保证边界，我们的训练样本的个数只需要是 k 的对数就可以，即是假设集合个数的对数个，这在后面将会很重要。

类似的，我们可以固定 m 和 δ，来求解 γ。我们可以得到在概率 1-δ 下，对于所有的 $h \in \mathcal{H}$，存在

$$|\hat{\varepsilon}(h) - \varepsilon(h)| \leq \sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}.$$

现在我们假设一致收敛的情况，故对于所有的 $h \in \mathcal{H}$，$|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$。那么关于我们选择的学习算法$\hat{h} = \arg\min_{h \in H} \hat{\varepsilon}(h)$(最小化训练误差)，能够证明什么呢？首先定义是对于假设集合$\mathcal{H}$ 中最优假设为 $h^* = \arg\min_{h \in H} \varepsilon(h)$(最小化泛化误差)。注意，$h^*$是最优的，因此我们有：

$$\begin{aligned} \varepsilon(\hat{h}) &\leq \hat{\varepsilon}(\hat{h}) + \gamma \\ &\leq \hat{\varepsilon}(h^*) + \gamma \\ &\leq \varepsilon(h^*) + 2\gamma \end{aligned}$$

第一行使用了一致收敛假设下的$|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$，第二行是因为我们选择了$\hat{h}$ 来最小化 $\hat{\varepsilon}(h)$，因此 $\hat{\varepsilon}(\hat{h}) \leq \hat{\varepsilon}(h)$，因此 $\hat{\varepsilon}(\hat{h}) \leq \hat{\varepsilon}(h*)$ 。第三行再次使用了一致收敛假设下的 $\hat{\varepsilon}(h^*) \leq \varepsilon(h^*) + \gamma$。因此，结果展示了：在一致收敛情况下，假设 $\hat{h}$ 的泛化误差最多比最优假设的泛化误差情况下差 2γ。(注意 ε 是泛化误差，h^*是训练误差最小的最优假设)。

我们现在把这些东西和在一起，有了如下定理：假设集合中假设个数为$|H| = K$，对于固定的 m, δ，在至少 1 – δ 的概率下：

$$\varepsilon(\hat{h}) \leq \left(\min_{h \in \mathcal{H}} \varepsilon(h) \right) + 2\sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}.$$

这也就量化了之前提到了模型选择时的偏差和方差的权衡。比如假设我们有一个假设集合$\mathcal{H}$，考虑一个较大的假设集合$\mathcal{H}' \supseteq \mathcal{H}$ 。如果我们使用了类$\mathcal{H}'$，那么定理中不等式的右侧的第一项 $\min h_\varepsilon(h)$ 只能够下降(因为我们考虑了更大的类的最小情况)。因此，使用更大一点的假设集合，我们的偏差(bias)只能下降。可是如果 k 增加，那么第二项将会增加，这个增长对应的我们的方差(variance)。

在固定 γ 和 δ，就像之前那么我们可以求得 m：

推论(Corollary.)：假设集合中假设的个数为|H| = K，对于固定的 γ, δ，在 ε(ĥ) ≤minh∈H_ε(h) + 2γ 有 1 – δ 的概率下，有：

$$\begin{aligned} m &\geq \frac{1}{2\gamma^2}\log\frac{2k}{\delta} \\ &= O\left(\frac{1}{\gamma^2}\log\frac{k}{\delta}\right) \end{aligned}$$

10. 特征选择

本节将接着上一讲，并结束掉经验风险最小化一章，之后讨论特征选择等相关问题。

10.1. 无限情况下的$\mathcal{H}$

我们在假设集合内假设有限的情况下已经证明了一些有用的理论。但是，很多假设集合$\mathcal{H}$，内部的参数都是实数(比如线性回归)即假设集合中包含了无数多个假设。那么，我们能否得到类似的结论呢?

首先，我们讨论一下不是很正确的一些论点，而这些论点是更好的且更加广泛，而且有利于我们的分析。假设$\mathcal{H}$的参数是实数 d，因为计算机存储实数的时候使用双精度即 64b 来存储一个浮点型数据。这就意味着我们的学习算法会假如我们使用了双精度类型存储数据，即参数是 64b 大小，那么对于我们的假设集合实际上包含 $k = 2^{64d}$ 个不同的假设。根据上一节的结论，我们可以保证 $\varepsilon(\hat{h}) \leq \varepsilon(h^*) + 2\gamma$，在概率至少是 $1-\delta$ 的情况下，使得 $m \geq O(1/\gamma^2 * \log(2^{64d}/\delta)) = O(d/\gamma^2 * \log(1/\delta) = O_{\gamma,\delta}(d)$ (这里的下标 γ, δ 表示大 O 是依赖于该两参数)。因此，需要的训练样本数最多只是模型的参数的线性个数。

问题是，我们依据 64b 的浮点型数据存储并不能完全满足该论点，但是结论却是差不多对的。如果我们继续尝试最小化训练误差，也就是为了更好的学习到假设集合中的参数 d，通常的我们需要训练样本达到 d 的线性个数。从这个意义上来说，这些结论对于一个使用了经验风险最小化的算法而言，根本不需要证明。因此对于多数判别学习算法，他们的样本复杂度是 d 的线性复杂度，然而这些结论并不总是那么容易实现。为了保证许多非经验风险最小化的学习算法的理论可用性，这方面仍在研究中。

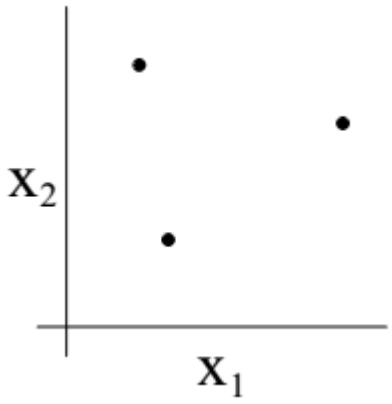
之前的讨论是依赖于假设集合$\mathcal{H}$的参数，在某些情况下仍然是无法满足的。直观的看，这并不像是什么问题。假设我们的线性分类器假设集合为，$h_\theta(x) = 1\{\theta_0 + \theta_1 x_1 + \cdots \theta_n x_n \geq 0\}$，含有 n+1 个参数 $\theta_0 \ldots \Theta_n$，但是他同时也可以写成 $h_{u,v}(x) = 1\{(u^2_0 - v^2_0) + (u^2_1 - v^2_1)x_1 + \cdots (u^2_n - v^2_n)x_n \geq 0\}$，包含 $2n+2$ 个参数 u_i, v_i。然而这些都是定义了同一个假设集合$\mathcal{H}$： n 维度下的线性分类器。

为了推导出更令人满意的结论，我们定义一些东西如下：

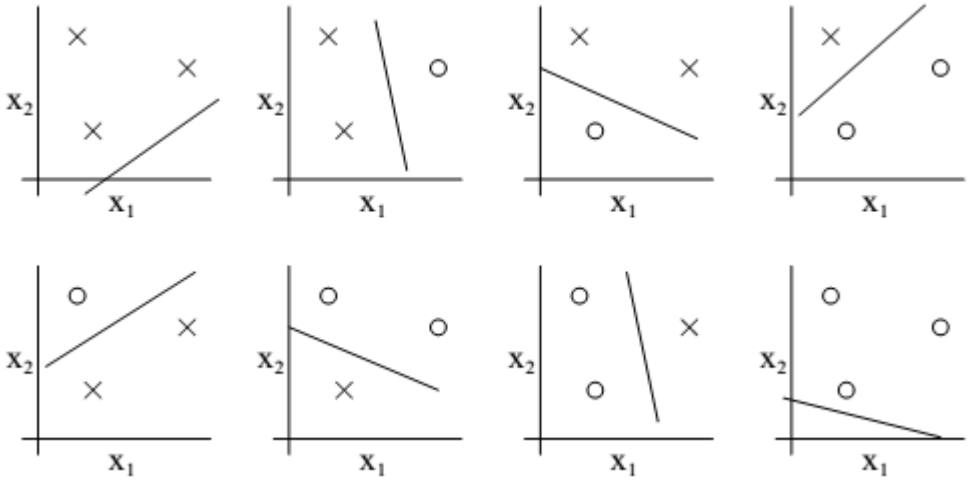
对于给定的数据集 $S = \{x^{(i)}, \ldots, x^{(d)}\}$(和训练数据集无关)，其中 $x^{(i)} \in X$，我们说$\mathcal{H}$粉碎(shatter)了 S，如果$\mathcal{H}$能够实现 S 上的任意类别(label)。举例子而言，如果对于任意类别集合$\{y^{(1)}, \ldots, y^{(d)}\}$，$\mathcal{H}$中总是存在一些 h 使得对于所有 $i = 1, \ldots d$ 存在 $h(x^{(i)}) = y^{(i)}$。

对于一个假设集合$\mathcal{H}$，我们定义它的 VC 维(Vapnik-Chervonenkis dimension)，记作 $\mathrm{VC}(\mathcal{H})$，指被假设集合粉碎(shatter)掉的最大的数据集的大小。(如果$\mathcal{H}$能够粉碎任意大小的数据集，那么 $\mathrm{VC}(\mathcal{H}) = \infty$)。

举例子而言，如下图的三个点：对于二维的线性分类器假设集合($h(x) = 1\{\theta_0 + \theta_1 x_1 + \theta_2 x_2 \geq 0\}$)能否粉碎(shatter)掉途中的数据集呢？答案是可以。

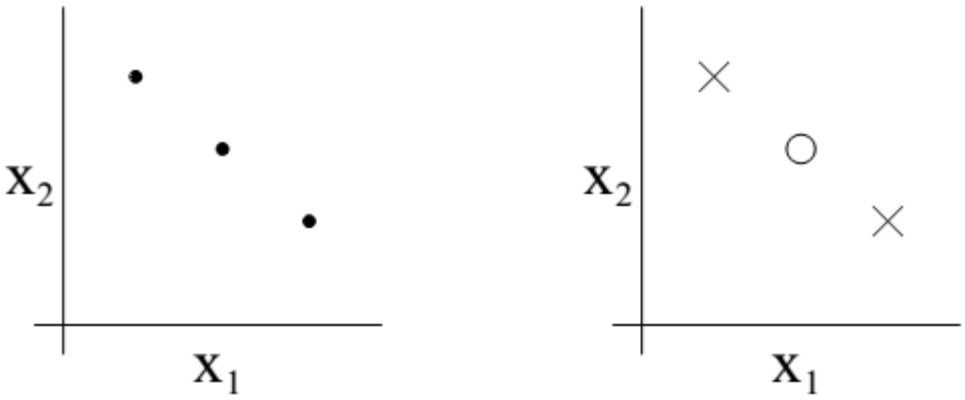


具体的粉碎(shatter)方案如下：



对于上图点中八个可能分类情况的任何一个，我们都能找到一个线性分类器使得训练误差为 0。此外，对于 4 个点的数据集，这个假设集合就无法全部粉碎(shatter)掉所有可能的分类。因此，对于该假设集合$\mathcal{H}$能够最大粉碎(shatter)掉的数据集大小是 3，即 $\mathrm{VC}(\mathcal{H}) = 3$。

注意：虽然假设集合$\mathcal{H}$的 VC 维是 3，但是也可能存在大小是 3 的数据集，该该假设集合无法粉碎(shatter)的情况。比如下图，如果三个点坐落在一条直线上(下图左)，那么我们无法找到一条直线把右图的分类情况分割开来。



换句话说，在 VC 维的假设下，为了证明 VC($\mathcal{H}$)最小是 d，那么我们仅需要展示假设集合$\mathcal{H}$至少能够粉碎(shatter)一个大小为 d 的数据集。下面是基于 Vapnik 的理论(这是机器学习理论中最重要的理论。)

定律：对于给定的假设集合$\mathcal{H}$，令 VC($\mathcal{H}$) = d，那么至少有 1 – δ 的概率下，对于所有的 h∈ $\mathcal{H}$，存在 $|\varepsilon(h) - \hat{\varepsilon}(h)| \leq O(\sqrt{\frac{d}{m} log \frac{m}{d} + \frac{1}{m} log \frac{1}{\delta}})$。因此，在至少 1 – δ 的概率下，也存在 $\hat{\varepsilon}(h) \leq \varepsilon(h^*)$ $O\left(\sqrt{\frac{d}{m} log \frac{m}{d} + \frac{1}{m} log \frac{1}{\delta}}\right)$。

换句话说，如果假设集合$\mathcal{H}$存在有限的 VC 维，那么随着 m 足够大，将会发生一致收敛(uniform convergence)。正如之前那样，这允许我们给出泛化误差 ε(h)和 $\varepsilon(h^*)$(最小训练误差的假设的泛化误差)的关系，关系为：

推论：对于$|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$，如果对所有 h∈ $\mathcal{H}$，在至少 1 – δ 的概率下均成立(因此 $\varepsilon(\hat{h}) \leq \varepsilon(h^*) + 2\gamma$)，那么满足 $m = O_{\gamma,\delta}(d)$。

换句话说，为了使得假设集合$\mathcal{H}$能够学习的足够好，训练样本个数跟假设集合的 VC 维是线性关系的。实际表明，对于绝大多数假设集合，VC 维(选用一个合理的参数)也大致是和参数个数成线性关系。把这些结论放在一起，我们总结下：对于一个试图最小化训练误差的学习算法，训练样本个数需要大致和假设集合$\mathcal{H}$中参数的个数呈现线性关系(即样本复杂度和参数个数是线性关系的)。

现在假设在一个学习问题中，我们尝试在不同的模型中选择一个模型。举例子说，我们可能使用多项式回归模型 $h\theta(x) = g(\theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_k x^k)$，我们要决定 k 究竟是 0, 1 … 还是 10。我们的模型能否自动的选择一个模型，并较好的权衡了模型的偏差(bias)和方差(variance)。其他的，如假设我们想自动的选择一个参数 τ 的范围，对于局部权重回归或者选择一个合适的 C 对于我们的正则化的 SVM 模型。我们应该如何做?

为了方便表达，我们定义一个有限的模型集合 $M=\{M_1, \dots M_d\}$，我们想从中选择一个最好的模型。比如说，在刚才例子中，模型 M_i 是 i 阶的多项式回归模型。当然，M 也可以是包含 SVM、NN(neural network，神经网络)或者 logistic 回归的模型。

10.2. 交叉验证(Cross validation)

假设我们给定的训练集合为 S，如果我们基于经验风险最小化原则选择模型，下面是基本的思路：

1) 在训练集合 S 上训练模型集合 M 内所有的模型 M_i，得到一些假设 h_i；
2) 选择训练误差最小的假设 h。

然而该方法并不可行。考虑选择一个多项式回归的阶次，我们知道阶次越高，模型对训练集合拟合的效果越好，也就会得到更低的训练误差。因此，这个方法通常会选择出一个高方差、高阶次的多项式模型，正如刚才说的，这通常都是一个很差的选择。接下来，我们对该方法进行改进。改进后的方法称之为 hold-out 交叉验证(hold-out cross validation)，也称之为简单交叉验证(simple cross validation)，过程如下：

1) 把训练集合 S 随机分割成 S_{train} 和 S_{cv}(比如随机选择 70%的样本为 S_{train}，余下的为 S_{cv})，数据集 S_{cv} 称之为 hold-out 交叉验证数据集；
2) 仅在数据集 S_{train} 上训练模型 M_i，得到一些假设 h_i；
3) 选择在 S_{cv} 数据集上误差 $\hat{\varepsilon}_{S_{cv}}(h_i)$最小的假设最为最终的假设。

通过在为训练的数据集 S_{cv} 上进行测试，我们能够更好的估计各个假设 h_i 的真实泛化误差，然后选择一个泛化误差估计值最小的假设作为最后假设。通常情况下，数据的 1/3-1/4 用于 hold-out 交叉验证，30%是一个典型的选择。对于步骤 3，也可以根据 $\arg\min_i \hat{\varepsilon}_{S_{cv}}(h_i)$来选择 h_i，之后再使用全部数据集，即 S 来重新训练模型。(通常情况下这是一个好的方法，但是有一个例外，就是当学习算法对原始状态或数据的波动很敏感的时候。对于这些方法，M_i 在 S_{train} 上表现的很好并不意味着在 S_{CV} 上也会表现很好，这个时候就不需要重新训练模型了)。

使用简单交叉验证的缺点是他浪费了 30%的数据，即得到的最佳模型是在 70%的训练数据上选出来的，不代表在全部训练数据上是最佳的。尽管最后我们可以选择使用全部训练数据来训练模型，但是在选择最优的模型的时候，我们仍然只是用来 70%的数据。当数据流比较大或者比较容易获得的时候，该方法是很好的。但是当数据是比较稀缺的时候(比如样本数 m=20)，我们需要更好的一些方法。

这里有一个方法称之为 k 折交叉验证(k-fold cross validation)，如下：

1) 把 S 随机分割成 k 个互斥的子集，每个子集样本数为 m/k，这些子集称之为 $S_1, \ldots, S_k$；
2) 对于每一个模型 M_i，我们进行如下的测试：
   对于 j = 1，2，...，k
   在数据集 $S_1 \cup \cdots \cup S_{j-1} \cup S_{j+1} \cup \cdots S_k$(即除去 S_j 以外)上训练模型 M_i，得到一些假设 h_{ij}
   在数据集 S_j 上测试该假设 h_{ij}，得到误差 $\hat{\varepsilon} S_j(h_{ij})$
   那么模型 Mi 的泛化误差估计值就等于各个$\hat{\varepsilon} S_j(h_{ij})$的平均值(所有的 j)。
3) 选择一个泛化误差估计值最低的模型 M_i，之后在所有的训练集合 S 上重新训练，最终得到的假设就是我们们最终的结果。

一个比较常用的选择是 K=10，即 10 折交叉验证。因为训练数据集仅占整体的 1/k，所以会比之前 hold-out 的小很多，而且这将导致更多的计算量(因为每个模型都要训练和测试 k 次。)。虽然 k 经常选用 10，但是当数据集比较小的时候，我们还是倾向于让 k=m，即让 k 等于整个数据集的大小，这样让更多的数据参与训练(每次只在一个样本上测试)。在这样的设计下，我们最终的泛化误差估计其实是考虑所有交叉之后的总体均值误差。这种情况也有一个自己的名字——leave-one-out 交叉验证(leave-one-out cross validation)。

最后，虽然我们介绍不同的交叉验证方法用于模型的选择，当然这些方也可以用在一个模型中，用于模型的或算法的评估。举个例子，如果你使用了一些机器学习的算法，然后想要测试他在你的应用上效果如何(比如你有一个新奇的学习算法，想要很正式的评估该方法在测试集合上的效果如何)，交叉验证将会是一个相对合理的方法。

10.3. 特征选择

在模型选择中，有一个很特殊且很重要的一步，就是特征选择(feature selection)。为了引出这个，先想像你有一个监督式学习问题，其中特征数目 n 非常的大(比如 $n \gg m$),但是你猜想可能只有一小部分的特征和学习任务相关。尽管你在 n 个特征上使用了简单的线性分类器(比如感知器学习)，假设集合的 VC 维仍然是 O(n)，因此过拟合将是一个潜在的问题，除非你的训练集足够的大。

在这样的设置下，你可以使用特征选择算法来减少特征数量。对于给定的 n 个特征，有 2^n 个可能的特征子集(因为每个特征都包含两个选择，被引入算法和不被引入算法)，因此特征选择可以看作是在 2^n 个可能模型中的模型选择。对于特别大的 n 来说，这通常是计算量非常大的计算，因为要比较 2^n 个模型。因此，有些启发式的搜索策略被用于好的特征子集的发现。下面我们展示一个搜索策略，称之为前向搜索(forward search)：

1) 初始化 $F = \emptyset$
2) 循环 {
   a) 对于 $i = 1$，…，n，如果 $i \notin F$，那么让 $F_i = F \cup \{i\}$，之后使用交叉验证来评估特征 F_i(即仅使用 F_i 中有的特征来训练我们的学习算法，之后测试得到他的泛化误差)；
   b) 令 F 等于 a 步骤中发现的最好一个特征子集 F_i。}
3) 选择并输出最好的特征子集。

停止该循环过程算法可以通过当 $F=\{1,\ldots,n\}$，即涵盖了所有的特征为止，或者当 F 达到了某些阈值(比如设置学习算法想要使用的最大特征数)。

上述的算法是 wrapper 模型选择(wrapper model feature selection)的一个实例，其中 wrapper 指不断地使用不同的特征集来测试学习算法。与前向搜索相对应的，也有其他的一些搜索策略。比如后向搜索(backward search)，即开始的时候 $F=\{1,\ldots,n\}$，每次更新需要删除一个特征(测试删除一个特征之后的效果，与前向搜索特征一个特征类似)，不断的循环直到 $F = \emptyset$。

Wrapper 特征选择算法通常都能够很有效的工作，但是计算量非常大，需要非常多次数的调用学习算法。而且，完整的前向搜索算法(即直到 $F=\{1,\ldots,n\}$才停止)调用学习算法的次数复杂度是 $O(n^2)$。

过滤特征选择(filter feature selection)是给出了一个启发式但是计算量非常低的方法来进行特征子集的选择。该方法的主要思路是通过计算每个特征 x_i 对于分类标签 y 的信息量得分 S(i)，然后选择得分 S(i)最大的前 k 个特征即可。

那么接下来的问题就是使用什么样的方法作度量得分 S(i)。关于得分 S(i)的可能选择，我们可以定义 S(i)为 x_i 和 y 之间的相关性大小的绝对值，这将会使得我们的结果是选择出来与分类标签相关性最强的一些特征。而在实际过程中，更常用的(对于特征 x_i 是离散值的情况)是选择 x_i 和 y 之间的互信息 MI(mutual information)作为 S(i)。其中 MI 计算如下：

$$\mathrm{MI}(x_i, y) = \sum_{x_i \in \{0,1\}} \sum_{y \in \{0,1\}} p(x_i, y) log \frac{p(x_i, y)}{p(x_i)p(y)}$$

该等式是建立在 x_i 和 y 的值是二元情况，等式很容易推广到 xy 是多个离散值的情况。其中 $p(x_i, y)$, $p(xi)$ 和 $p(y)$能够很容易的从训练集合中的得到。

为了更好的明白互信息是怎么来，我们注意到互信息可以表示为：KL 距离(Kullback-Leibler divergence)，$\mathrm{MI}(x_i, y) = \mathrm{KL}(p(x_i, y) \,||\, p(x_i)p(y))$。这是衡量了概率分布 $p(x_i, y)$ 和 $p(x_i)p(y)$之间有多大程度的不同。如果 x_i 和 y 是相互独立的变量，那么 $p(x_i, y) =$

$p(x_i)p(y)$，而两个分布之间的 KL 距离是 0，这就意味着 x_i 对于 y 没有很明显的信息量，那么 S(i)就会很小；相反，如果 x_i 和 y 是很相关，即 x_i 对于 y 具有很大的信息量，那么 $MI(x_i, y)$将会很大。

最后一点，现在我们根据得分 S(i)对所有特征进行排序，那么我们究竟应该选择多少个特征 k 呢？一个标准的做法是对于所有可能 k 个特征(也可以设定阈值等等)采用交叉验证。比如，在使用朴素贝叶斯方法进行文本分类的时候，一个问题是特征 n 是单词表的大小，通常都非常大，使用这个方法选择出的特征子集通常都能够使得分类的准确率上升。

11. 贝叶斯统计正则化

11.1. 贝叶斯统计和正则化

在这一部分，我们将讨论一个避免过拟合的另外一种方法。

在开始之前，回顾一下，线性回归中使用的估计方法是最小二乘法，logistic 回归是条件概率的最大似然估计，朴素贝叶斯是联合概率的最大似然估计，SVM 是二次规划。现在我们先讨论一下使用最大似然估计(maximum likelihood (ML))进行参数拟合。我们选择参数的时候，是根据如下式子：

$$\theta_{ML} = \arg\max \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta)$$

这里，我们把 θ 看作是未知的参数(这里采用了频率学派的观点)。从频率学派(frequentist statistics)看，θ 是一个固定的但未知的值，即 θ 并不是随机的，只是在这里他恰好是未知的，我们的任务就是通过统计方法(如最大似然)来尝试估计该参数值。

另外一种视角是贝叶斯学派(Bayesian)，认为 θ 是一个未知的随机变量。在该方法中，我们将用先验概率(prior distribution)p(θ)来表达我们对于参数 θ 的先验信念(prior beliefs)。对于给定的训练集合 $S = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$ 当我们在新的 x 上预测时，我们按照如下的方式计算参数的后验概率：

$$P(\theta|S) = \frac{p(S|\theta)p(\theta)}{p(S)} = \frac{\left(\prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)\right) p(\theta)}{\int_{\theta} \left(\prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)p(\theta)\right) d\theta}$$

在上面的等式中，$p(y^{(i)}|x^{(i)},\theta)$ 来自于你使用的学习算法的任何模型。比如说，如果你使用贝叶斯 logistic 回归(BLR，Bayesian logistic regression)，那么你可能选择 $p(y^{(i)}|x^{(i)},\theta) = h_\theta(x^{(i)})^{y^{(i)}} \left(1 - h_\theta(x^{(i)})^{1-y^{(i)}}\right)$，其中 $h_\theta(x^{(i)}) = 1/(1 + \exp(-\theta^T x^{(i)}))$ 。

当我们给出一个新的测试样本 x 的时候，需要在上面做出一个预测，我们可以基于θ的后验分布来计算分类标签上的后验分布：$p(y|x, S) = \int_\theta p(y|x,\theta)p(\theta|S)d\theta$ 。在该等式中，$p(\theta|S)$是θ的后验分布。因此，如果我们的目标是预测对于给定 x 下的 y 值，那么我们将输出 $E[y|x, S] = \int_y yp(y|x, S)dy$ 。这里展示的过程可以被看成是使用了全贝叶斯（fully Bayesian）预测，预测的结果是基于θ的后验概率而计算的平均情况。然而不幸的是，通常情况计算后验概率分布非常的困难，因为这要求对θ进行积分（通常都是高维的,这里对于连续变量 y 是用积分，如果是离散变量 y 则是求和），这通常不能在闭合形式解(closed-form)上的完成。

因此，在实际过程中我们通常用近似的情况来替换θ的后验概率分布。为了替换θ的后验概率分布，一个常用的近似是简单的点估计。对于θ的最大后验（MAP，maximum a posteriori）估计如下：

$$\theta_{MAP} = \arg\max_\theta \prod_{i=1}^{m} p\left(y^{(i)}\middle|x^{(i)},\theta\right)p(\theta)$$

注意到，除了后面的先验概率 p(θ)外，这个式子和θ的最大似然(ML, maximum likelihood)估计的式子是一样的。

在实际应用中，对于先验概率 p(θ)一个常用的选择是θ~ N (0, τ^2 I)。采用这样的先验分布，拟合得到的θ_{MAP} 将会比如采用最大似然得到的结果更小。实际过程中，贝叶斯最大后验概率估计相比于最大似然估计，不容易引起模型的过拟合。比如对于文本分类而言，尽管特征 n ≫ m，而贝叶斯 logistic 回归仍然是一个非常有效的分类算法。

11.2. 附录：过拟合思考

过拟合问题，个人认为是缺乏先验认知而盲目学习的结果，比如为了估计硬币正面朝向的概率，设该二项分布的参数为 θ，0<=θ<=1。如果一个硬币三次投币都是正面向上，采用最大似然估计的结果是：$\max_\theta \theta^3$,则 θ=1，这样的结果不符合常理或是过于偏激。而采用贝叶斯观点，认为先验的 θ 是在 0-1 上的均匀分布，即概率密度为 $f_\theta(\theta) = 1$ 其中 $0 \le \theta \le 1$ 。

接下来，我们将会看到观测值是如何改变我们对 θ 的认知。对于给定的 θ，观测 x 服从 n 次试验成功的概率的二项分布是：

$$f_{x|\theta}(x|\theta) = C^n_x\theta^x(1-\theta)^{n-x}$$，其中 x=0,1,...n。

因此他们具有的联合分布是:

$$f_{x,\theta}(x,\theta) = f_{x|\theta}(x|\theta)\, f_\theta(\theta) = C^n_x\theta^x(1-\theta)^{n-x}$$ 其中 x=0,1,...n。

我们的目标 $f_{\theta|x}(\theta|x) = f_{x,\theta}(x,\theta) / f_x(x)$ ，而 $f_x(x)$ 等于联合分布密度 $f_{x,\theta}(x,\theta)$对 θ 的积分（这里就不再求解积分了，一般求解积分的方法可以用计算机求解，也可以使用贝塔密度来化简求解）得到 $f_x(x) = 1/(n+1)$ ，因此得到在观测结果上的 θ 的条件密度:

$$f_{\theta|x}(\theta|x) = (n+1)\, C^n_x\theta^x(1-\theta)^{n-x} = \Gamma(n+2)\,/[\Gamma(x+1)\,\Gamma(n-x+1)]\;\theta^x(1-\theta)^{n-x}$$

即后验密度是参数为 $a = x+1$， $b = n-x+1$ 的贝塔密度。根据观测，我们知道 n=x=3，即 a=4,b=1，那么贝塔分布的期望是 $E(\theta) = a / (a+b) = 4/5 = 0.8$ 这里得到了一个结果为 0.8 而不是之前最大似然的结果 1.0 。

注意：最终θ的期望 E(θ) = a/(a+b) = (x+1) / (n+2)，对于二元分类，这不恰好是拉普拉斯平滑吗？当然，改变θ的先验认识会改变分子和分母的,比如 $f_\theta(\theta)$服从 Beta(i,j)，那么最终结果是 E(θ) = x+i / (n+j+i)，可以推而广之就类似于 m 估计了。

11.3. 感知器和最大间隔分类

这部分是关于学习理论的最后一部分了，我们将介绍机器学习中不同的模型。特别的，到目前为止，我们在训练数据中都是考虑批量学习(batch learning)，而假设 h 的测试是在分开的测试集合上。在这部分的内容中，我们将考虑在线学习（online learning），即在学习过程中能持续的进行预测。

在这样的设置下，学习算法是一系列有序的样本$(x^{(1)}; y^{(1)}); (x^{(2)}; y^{(2)}); : : : (x^{(m)}; y^{(m)})$。特别的，算法首先是看到 $x^{(1)}$，然后预测 $y^{(1)}$的值。在预测之后，算法才能知晓 $y^{(1)}$ 的真实值（算法可能利用这个信息进行学习）。之后，算法会接收 $x^{(2)}$，然后预测 $y^{(2)}$的值，在真实值 $y^{(2)}$被算法知晓之后，算法会进行学习，之后不断的循环，直到最后的$(x^{(m)}; y^{(m)})$。在在线学习的设置中，我们感兴趣的是算法在整个过程引起的误差。因此尽管它仍然在学习之中，使用该算法的模型一开始就不得不作出预测。

我们将会给出在线学习的误差边界，这里以感知器学习为例 (perceptron algorithm)。为了让后续的推导更加容易点，我们对类别标签做如下的表示 $y = \in \{-1,1\}$。回忆一下感知器学习算法，他的参数 $\theta \in R^{n+1}$，而所做的预测假设是 $h_\theta(x) = g(\theta^T x)$，其中 g(z) 是符号函数，即当 $z \geq 0$ 时 g(z) = 1 ，当 z<0 时，g(z) =-1。此外，对于给定的训练样本(x，y)，感知器学习算法更新参数的方法如下：如果 $h_\theta(x) = y$，那么参数不做变化，否则 $\theta := \theta + yx$。

接下来的理论将会给出在线学习的感知器模型的误差边界。需要注意的是接下来的误差边界是受样本序列的个数 m 的影响，同时也会受到输入维度 n 的影响。

理论 (Block, 1962, and Noviko, 1962)：一有序学习样本为$(x^{(1)}; y^{(1)}); (x^{(2)}; y^{(2)}); : : : (x^{(m)}; y^{(m)})$，假设对于所有的 i，均存在$||x^{(i)}|| \leq D$，进一步的，如果存在 u（$||u||_2 = 1$），那么对于序列中的所有样本有：$y(i) (u^T x(i)) \geq \gamma$ （比如，如果 $y(i) = 1$，那么 $u^T x(i) \geq \gamma$，如果如果 $y(i) = -1$，那么 $u^T x(i) \leq -\gamma$），那么感知器学习算法在该序列上的误差最大是$(D=\gamma)^2$。

在此就不进行证明了。

11.4. 使用机器学习的一些建议

接下来将会讲一些如何在不同的应用中使用机器学习算法。这些内容并不是很数学化，但是仍然比较难理解。

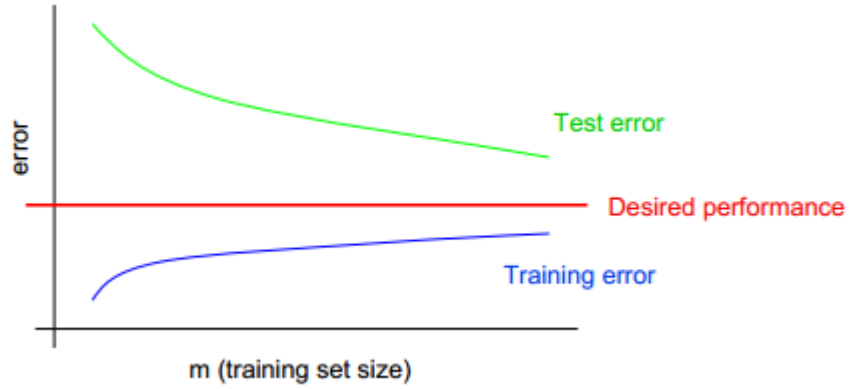
11.4.1. 调试学习算法

举一个例子，在反垃圾邮件的应用中，你很仔细的选择了 100 个单词作为特征（而不是选择英语中所有 50000 多个单词）。在采用了贝叶斯 logistic 回归（BLR， Bayesian logistic regression）模型，并且使用了梯度下降法（优化目标为 $\max_\theta \sum_{i=1}^{m} \log(p(y^{(i)}|x^{(i)},\theta) - \lambda||\theta||^2)$），最终得到了难以接受的测试误差为 20%。接下来，我们该怎么做呢？

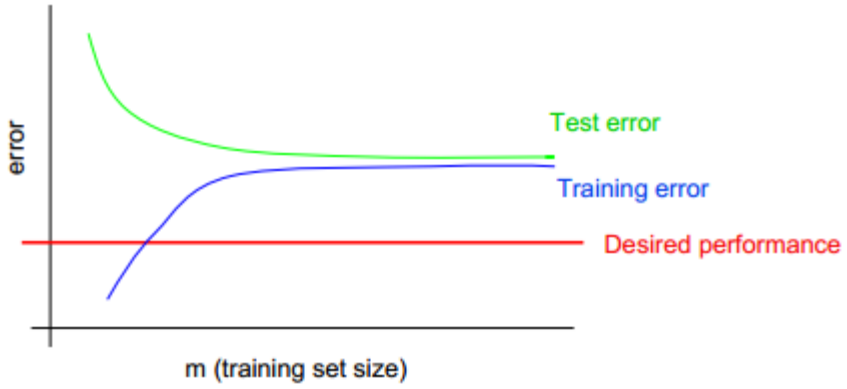
通常情况下，我们有以下不同的方法可以尝试：

1) 尝试获得 更多的训练样本；
2) 尝试选用更小的特征集，即减少使用的特征数量；
3) 尝试选用更大的特征集；
4) 尝试改变特征，如选用邮件的标头还是邮件本身的内容；
5) 尝试在梯度下降法过程中进行更多次数的迭代；
6) 尝试使用牛顿法；
7) 尝试使用不同的λ；
8) 尝试使用 SVM；

这些方法可能有用，但是去尝试的话可能太耗费时间，而且不能确定最终是否能够解决这个问题。那么我们应该怎么做呢？怎么调试来发现真正的问题以及如何去解决呢？通常情况下，我们的问题是过拟合（高方差）和欠拟合（高偏差）。其中，对于方差而言，训练误差将会比测试误差小很多，而偏差的话，训练误差也会很高。



上图是一个典型的高方差学习曲线。测试误差随着样本量的增加而不断的降低，且训练误差和测试误差之间存在较大差距，此时增加样本将会有助于降低误差。



上图是一个典型的高偏差学习曲线。每一个训练误差都高的难以接受，且训练误差和测试误差之间差距很小。误差和偏差的权衡是常见的一种调试，对于我们提到的八种调试方法中，增加样本、减少特征数量能够解决高方差问题，而增加特征数量和尝试改变特征，通常能够解决高偏差问题。然而，对于其他的一些问题，也许就需要自己去调试，找出模型的错误和优待提高的地方了。

比如举个例子，采用 BLR 能够得到：在垃圾邮件上误差为 2%，而不可接受的是在非垃圾邮件上的误差也是 2%。尝试使用线性核函数的 SVM 之后，得到在垃圾邮件上误差为 10%，而在非垃圾邮件上误差是 0.01%（这是可以接受的）。但是，考虑到计算量的问题，你仍然要使用 logistic 回归，那么我们应该怎么做?

这个时候可能伴随的一个问题是，是不是算法并没有收敛呢？通常情况下，很难判断是否真的收敛了。一般而言，都是根据迭代次数增加，判断是否会有优化等方法来判断。

其他的一些常见问题，比如我们的学习算法是否是凸函数？是否正确的优化了？我们是否关心了代价呢，比如准确率 $a(\theta)=\sum_{i=1}^{m} w^{(i)} 1\{h_\theta(x^{(i)})=y^{(i)}$，其中非垃圾邮件的权重 $w^{(i)}$会比垃圾邮件的更大。是否引入了正确的正则化系数λ？对于 SVM 是否选择了合适的 C?

对比 SVM 和 BLR，如果引入了带有权重（或代价的）准确率，如果 $a(\theta_{SVM}) > a(\theta_{BLR})$，且 $J(\theta_{SVM}) > J(\theta_{BLR})$，那么问题可能是优化算法存在问题；如果如果 $a(\theta_{SVM}) > a(\theta_{BLR})$，且 $J(\theta_{SVM}) \le J(\theta_{BLR})$，那么问题可能是目标函数最优化的问题。

总结一下如下：

1) 尝试获得 更多的训练样本 ————————————————解决高方差问题
2) 尝试选用更小的特征集，即减少使用的特征数量； ————解决高方差问题
3) 尝试选用更大的特征集； ————————————————解决高偏差问题
4) 尝试改变特征，如选用邮件的标头还是邮件本身的内容；——解决高偏差问题

5) 尝试在梯度下降法过程中进行更多次数的迭代；————————解决优化算法问题
6) 尝试使用牛顿法；————————————————————解决优化算法问题
7) 尝试使用不同的λ；————————————————————解决优化目标问题
8) 尝试使用 SVM；————————————————————解决优化目标问题

通常情况下，我们需要调试来判断学习算法究竟发生了什么。尽管有时候算法效果很好，但是为了更好的理解算法在做些什么，还是需要调试的。调试能够帮助我们更好的理解应用的问题（做多了，经验就丰富了）。也能够帮助我们写好论文，让我们更深入的理解问题，比如说这个算法有效，相比起来下面的更好：这个算法有效是因为其中的 X。

11.4.2. 误差分析

很多项目都是由许多学习算法组成的。比如人脸识别，包括相机照片、图片预处理、脸部检测、眼睛分割、鼻子分割、嘴分割、到使用 Logistic 回归得到分类结果。对于整个过程而言，准确率（分别是指在当前部分分类为 100%准确的情况下，整体的准确率为多少）分别为：整体 85%，图片预处理 85.1%、脸部检测 91%、眼睛分割 95%、鼻子分割 96%、嘴分割 97%、Logistic 回归 100%。最有待提升的部分是脸部识别和眼的划分（分别较上一步增长较大，即如果改善了这部分，整体的准确率改进最大）。

12. K-means 算法

12.1. k-means 聚类算法

在聚类问题中，给定数据集$\{x^{(1)}, \ldots, x^{(m)}\}$，想要把这些数据划分成几个紧密联系的簇（clusters）。通常情况下，这里的$x^{(i)} \in R^n$，而标签$y^{(i)}$是未知的。因此这是一个非监督式学习（unsupervised learning）问题。

最简单的聚类算法是 k-means，k-means 聚类算法如下：

1) 初始化质心（即簇中心，cluster centroids），随机生成 k 个质心$\mu_1, \mu_2, \ldots, \mu_k \in R^n$；

2) 循环下面的步骤直到收敛 {

对于每一个 i，设定：

$$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2$$

对于每一个 j，设定

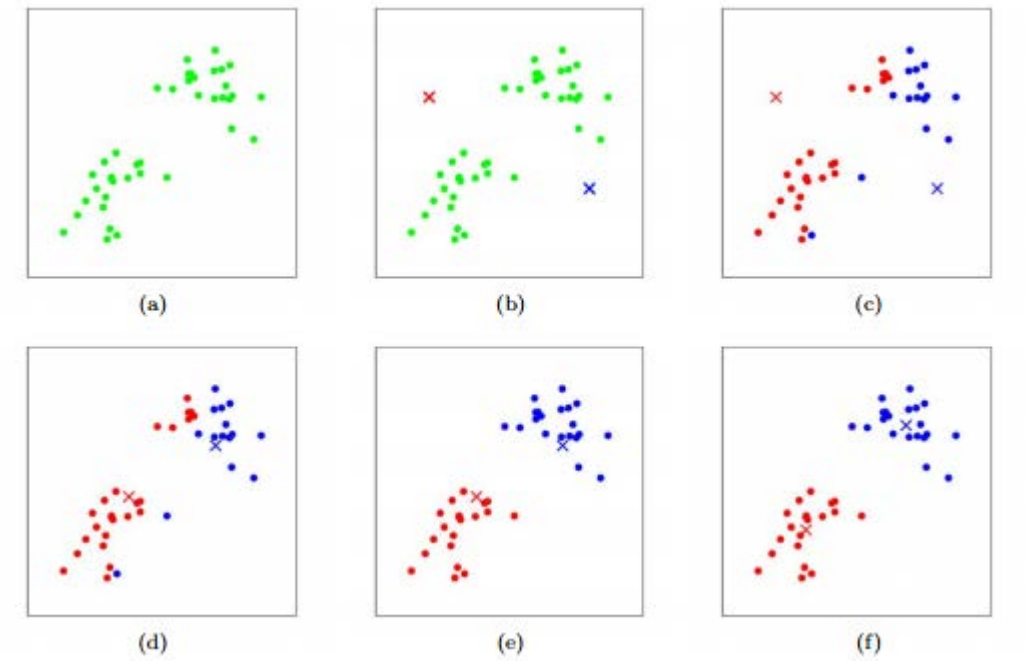
$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)}=j\}x^i}{\sum_{i=1}^{m} 1\{c^{(i)}=j\}}$$

}

上述算法中，k(算法的参数)就是我们要找的簇的个数，也是事先给定的聚类数；$c^{(i)}$代表样本$x^{(i)}$与 k 个类中距离最近的那个类，$c^{(i)}$的值是 1 到 k 中的一个。而质心μ_j代表了我们对当前属于同一个类的样本中心的猜测。在初始化质心时，我们可以随机选择 k 个训练样本，然后让初始质心为这 k 个样本的值（当然，也有其他的随机初始化方法。）

在算法的内循环中主要包括两个步骤：第一步是把每一个样本$x^{(i)}$分配距离他最近的质心μ_j，第二步是把质心μ_j重新设置为分配后的各个样本的质心。下图展示了 k-mean 是如何学习的。

对于图中，各个点表示样本，而质心用×表示，图(a) 是原始数据集，图(b)是随机生成了两个质心（这里并没有随机使用两个训练样本）。图(c-f) 展示了 k-means 是如何运行的。通过不断的迭代，重新分配质心，使得结果越来越好。



那么 k-means 是否一定会收敛呢？是的，一定会收敛。首先我们定义一个畸变函数(distortion function) $J(c,\mu)=\sum_{i=1}^{m}||x^{(i)}-\mu_c^{(i)}||^2$，那么 J 描述了各个点 $x^{(i)}$ 到对应质心 $\mu_c^{(i)}$ 的距离的平方和。可以看得出 k-means 在 J 上恰好使用了坐标下降法(coordinate descent，参照 SVM 章节坐标下降法)。具体的，在 k-means 算法的内循环中，通过固定 μ 不变，求得 c 使得 J 最小，之后在固定 c，更新 μ 使得 J 最小。这样不断的循环着，J 就一定会单调下降，因此 J 的值会收敛(这也暗示了 c 和 μ 是收敛的)。从理论上来说，k-means 可能会在几个不同的聚类中摆动，比如存在不同的 c 和 μ，使得 J 都是一样，不过这种情况很少见。

畸变函数 J 是非凸函数，因此坐标下降法并不能保证最终会收敛到全局最小值。换句话说，k-means 可能会得到局部最优解。然而，通常情况下 k-means 能够得到很好的聚类结果。但如果你怕陷入局部最优，那么可以选取不同的初始值跑多遍 k-means，得到不同的聚类结果，然后取其中最小的 J (c, μ)。

12.2. 高斯混合模型和最大期望算法

在这一部分，我们讨论一下 EM(Expectation Maximization，期望最大化算法，又称为最大化期望算法)用于核密度估计(density estimation)。给定数据集$\{x^{(1)},\ldots,x^{(m)}\}$，因为是非监督式学习的设定，因此这些数据点没有标签。

我们希望把数据用一个特定的联合分布 $p(x^{(i)}, z^{(i)}) = p(x^{(i)}|z^{(i)})p(z^{(i)})$来建模。这里，我们认为 $z^{(i)}$服从多项式分布，$z^{(i)} \sim \text{Multinomial}(\Phi)$(这里 $\Phi_j \geq 0$，$\sum_{i=1}^{k} \emptyset_j = 1$，参数 $\Phi_j = p(z^{(i)} = j)$，此外，对于给定的 $z^{(i)}$，存在 $x^{(i)}|z^{(i)} = j \sim N(\mu_j, \sum_j)$。$z^{(i)}$有 k 个值可以取{1, … ,k}，我们的模型认为每一个样本 $x^{(i)}$都是在随机从{1, … ,k}取得的 $z^{(i)}$而生成的，而 $x^{(i)}$是可以用依赖于 $z^{(i)}$的 k 维高斯分布表示，这称之为混合高斯(mixture of Gaussians)模型。注意这里的 $z^{(i)}$是潜在(latent)随机变量，以为这他们是隐藏的或者不可见的。

那么我们的模型的参数是 Φ, μ 和 Σ，为了估计出这些参数值，我们采用最大似然法则求解，这里的最大似然函数如下：

$$\ell(\Phi, \mu, \Sigma) = \sum_{i=1}^{m} \log p(x^{(i)}; \phi, \mu, \Sigma) = \sum_{i=1}^{m} \log \sum_{z^{(i)}=1}^{k} p(x^{(i)}; \phi, \mu, \Sigma)\, p(z^{(i)}; \phi)$$

然而，如果我们令该式子的偏导数为 0，希望能够求解参数值，可是结果发现这是不可能的（因为求的结果不是封闭式 (close form)）。随机变量 $z^{(i)}$指明了 k 维高斯分布 $x^{(i)}$是怎么来的。注意如果我们知道了 $z^{(i)}$，那么最大似然问题就很容易求解了，因此我们可以这样写我们的最大似然函数：

$$\ell(\Phi, \mu, \Sigma) = \sum_{z^{(i)}=1}^{k} p(x^{(i)}; \phi, \mu, \Sigma) + \log p(z^{(i)}; \phi)$$

最大化似然函数，得到参数 Φ, μ,Σ 值分别如下：

$$\begin{aligned}
\phi_j &= \frac{1}{m} \sum_{i=1}^{m} 1\{z^{(i)} = j\}, \\
\mu_j &= \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}, \\
\Sigma_j &= \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}.
\end{aligned}$$

Φ 就是样本类别中 $z^{(i)}$=j 的比率。μ 是类别为 j 的样本特征均值，Σ 是类别为 j 的样例的特征的协方差矩阵。最终，我们看到了如果 $z^{(i)}$已知，并且把 $z^{(i)}$看作是分类的标签的话，那么这里的最大似然估计就相当于我们在估计高斯判别分析模型中的参数的时候一样。而我们的核心问题是，$z^{(i)}$是未知的，那么我们该怎么做呢?

EM 算法是一个迭代算法，主要包括两个步骤。在我们的问题中，可以用 E-步，即猜一个潜在变量 $z^{(i)}$的值，在 M-步中，我们根据上一步的猜测的模型来更新参数值，以获得最大似然估计。如下是 EM 算法：

循环直到收敛 {

(E-步) 对于每一个 i，j：设定

$$w^{(i)}_j := p(z^{(i)} = j|x^{(i)}; \Phi, \mu, \Sigma)$$

(M-步) 按如下方式更新参数：

$$\begin{aligned}
\phi_j &:= \frac{1}{m}\sum_{i=1}^{m} w_j^{(i)}, \\
\mu_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}, \\
\Sigma_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}
\end{aligned}$$

在 E-步，我们计算了给定 $x^{(i)}$和设定的参数下，$z^{(i)}$的后验概率（即 $w^{(i)}_j$）可以通过贝叶斯法则来求解：

$$p(z^{(i)} = j|x^{(i)}; \phi, \mu, \Sigma) = \frac{p(x^{(i)}|z^{(i)} = j; \mu, \Sigma)p(z^{(i)} = j; \phi)}{\sum_{l=1}^{k} p(x^{(i)}|z^{(i)} = l; \mu, \Sigma)p(z^{(i)} = l; \phi)}$$

这里 $p(x^{(i)}|z^{(i)}=j; \Phi, \mu, \Sigma)$是通过估计均值为 μ，方差为 Σ 的高斯分布在 x 时的概率密度的，而 $p(z^{(i)}= j ; \Phi)$是给定了 Φ_j，其他的依次类推。此外，E-步中的 $w^{(i)}_j$ 代表了我们对 $z^{(i)}$的软猜测(软(soft)的意思是我们的猜测是一个在[0,1]之间的概率值，而硬(hard)猜测是一个最佳的具体值，比如从{1, … , k}中选择一个)。此外，我们需要对比一下 M-步中，当 $z^{(i)}$给定时的式子的更新，与之前的高斯判别时给出的不一样，这里用 $w^{(i)}_j$ 代替了 $z^{(i)}$ 。

其实 k-means 也是一种 EM 算法，只是相比上面的软猜测，k-means 里采用了硬猜测选择簇的分配 c(i)。与 k-means 相似，EM 也会容易获得局部最优解，因此多次尝试是一个不错的方法。通过不断循环的猜测 $z^{(i)}$，EM 给出了一个直观的解释，但是并没有给出他一定收敛的定量解释。下一讲，我们会深入研究 EM 算法。

12.3. 最大期望算法一般化

之前我们主要讲了用于高斯混合拟合的 EM 算法。这里，我们将 EM 拓展到含有潜在变量的一类估计问题。在开始之前，先讨论一下 Jensen 不等式。

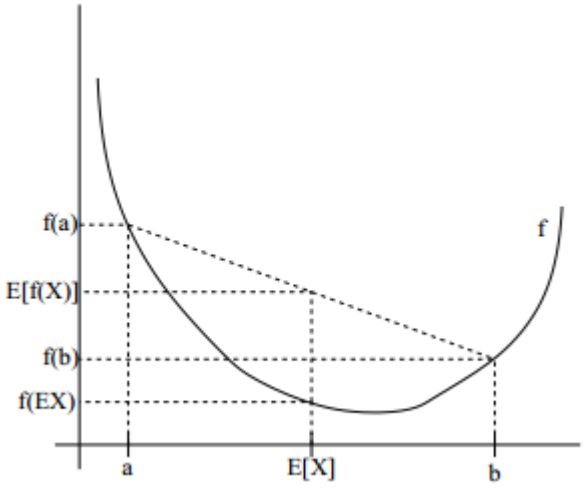
12.3.1. Jensen 不等式

假设函数 f 是在实数域上，如果 $f''(x) > 0$，那么 f 就是一个凸函数（convex function）。考虑 f 的输入是向量形式的，那么它的 hession 矩阵一定是半正定的（semi-definite），

即 $H \geq 0$。如果对于所有的 x，均存在 $f''(x) > 0$，那么可以认为 f 是严格（strictly）凸函数（对应的在输入为向量的情况下，则是 $H > 0$）。Jensen 不等式，可以表示如下：

定理：如果 f 是凸函数，那么对于任意一随机变量 X，存在 $E[f(X)] \geq f(EX)$；更进一步，如果 f 是严格凸函数（非正式的说，就是曲线连续，不存在直线部分），那么当且仅当 $X = E[X]$时，存在 $E[f(X)] = f(EX)$。注意：在写期望的时候，我们通常都去掉了括号，即 $f(EX) = f(E[X])$。

为了解释这个定理，我们从下图中看：



图中实线部分是凸函数 f，X 是一个随机变量，有 0.5 的机会选择值 a，有 0.5 的机会选择 b（如坐标轴上所示），那么 X 的期望 E[X]就在 ab 中间的位置。同时，我们在 y 轴上标识出了 f(a)，f(b)和 f(E[X])。在这个例子中，能够看出一定存在 $E[f(X)] \geq f(EX)$。顺带一提，很多人都记不清楚不等式的符号，这里可以通过这个图来帮助记忆。注意：当且仅当-f 也是凸的情况下（$f''(x) \leq 0$ or $H \leq 0$），f 是严格([strictly])凹的。Jensen 不等式也存在：对于凹函数 f，$E[f(X)] \leq f(EX)$。

12.3.2. EM 算法

假设我们有一个估计问题，对于 m 个训练数据$\{x^{(1)}, \ldots, x^{(m)}\}$，我们希望能够通过数据拟合得到 p(x,z)。通过最大似然估计，我们可以得到：

$$\ell(\theta) = \sum_{i=1}^{m} \log p(x;\theta) = \sum_{i=1}^{m} \log \sum_{z} p(x,z;\theta)$$

这里如果直接去求解参数 θ 的值，可能比较困难。这里的$z^{(i)}$是潜在变量，通常情况下如果$z^{(i)}$是观察得到的，那么该最大似然函数估计求解是非常容易的。在这样的设定下，EM 算法给出了一个有效的方法来求解最大似然估计。直接最大似然函数$\ell(\theta)$是困难的，我们的策略是通过不断的建立$\ell(\theta)$的下界（E-步），然后优化下界（M-步）。

对于每一个 i，用 Q_i 表示潜在变量 z 的分布（故 $Q_i \geq 0$，$\sum_{j=1}^{k} Q_j = 1$，注意如果 z 是连续变量，那么 Q_i 就是概率密度，求和就需要换成积分），那么：

$$\begin{aligned}\sum_i \log p(x^{(i)};\theta) &= \sum_i \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)};\theta) \\ &= \sum_i \log \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})} \\ &\geq \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}\end{aligned}$$

最后一步推导使用了 Jensen 不等式。具体的 $f(x) = \log x$ 是一个凹函数，因为 $f''(x) = -1/x^2 < 0$，其中实数域 $x \in R^+$，项 $\sum_{z^{(i)}} Q_i(z^{(i)})[\frac{p(x^{(i)},z^{(i)};\theta)}{Q_i(z^{(i)})}]$ 是 $[p(x^{(i)}, z^{(i)}; \theta)/Q_i(z^{(i)})]$的期望。根据 Jensen 不等式，我们有：

$$f\left(\mathrm{E}_{z^{(i)}\sim Q_i}\left[\frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}\right]\right) \geq \mathrm{E}_{z^{(i)}\sim Q_i}\left[f\left(\frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}\right)\right]$$

这里的下标 $z^{(i)} \sim Q_i$ 是指 $z^{(i)}$从 Q_i 产生。这样对于任意分布 Q_i，上式推导给出了$\ell(\theta)$的一个下界。这里 Q_i 有很多选择，那么我们应该选择什么呢？如果我们现在猜一个参数 θ，那么很自然的我们能够得到该参数下的$\ell(\theta)$的一个下界值。在后面我们将会证明 EM 的迭代是能够保证$\ell(\theta)$单调上升的。

在上述推导中，我们使用 Jensen 不等式，根据 Jensen 不等式如果要取等号的话，需要随机变量变成常数值，即 $p(x^{(i)}, z^{(i)}; \theta)/Q_i(z^{(i)}) = c$。c 是不依赖于的 $z^{(i)}$常数，因此 $Q_i(z^{(i)})$ 正比于 $p(x^{(i)}, z^{(i)}; \theta)$，又因为$\sum_{j=1}^{k} Q_j = 1$（因为这是一个概率分布），因此可以推导出：

$$\begin{aligned}Q_i(z^{(i)}) &= \frac{p(x^{(i)}, z^{(i)};\theta)}{\sum_z p(x^{(i)}, z;\theta)} \\ &= \frac{p(x^{(i)}, z^{(i)};\theta)}{p(x^{(i)};\theta)} \\ &= p(z^{(i)}|x^{(i)};\theta)\end{aligned}$$

这里，我们得到了 Q_i 是在给定 $x^{(i)}$下 $z^{(i)}$的后验概率。这就解决了如何选择 Q_i 的问题。这就是在 E-步中所要的建立$\ell(\theta)$的下界。在 M-步，即通过最大化似然函数来更新参数 θ。这样循环这两步就是所有的 EM 算法，即如下：

循环直到收敛 {

E-步：对于每一个 i，设定

$$Q_i(z^{(i)}) := p(z^{(i)}| x^{(i)}; \theta)$$

M-步 设定

$$\theta := \arg\max_\theta \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) log[\frac{p(x^{(i)},z^{(i)};\theta)}{Q_i(z^{(i)})}]$$

}

然而，我们是怎样保证算法的收敛的呢？好吧，假设 EM 算法中成功更新的两次参数分别为 $\theta^{(t)}$和 $\theta^{(t+1)}$。我们知道 EM 算法每次更新都会导致似然函数增大，即$\ell(\theta^{(t)}) \leq \ell(\theta^{(t+1)})$。这里的核心是在于我们对 Q_i 的选择。即在 EM 算法迭代过程中，我们以 $\theta^{(t)}$开始选择 $Q_i^{(t)}(z^{(i)}) := p(z^{(i)}| x^{(i)}; \theta^{(t)})$。我们知道根据 Jensen 不等式，为了取得等号，因此有：

$$\ell(\theta^{(t)}) = \sum_i \sum_{z^{(i)}} Q_i^{(t)}(z^{(i)}) log[\frac{p(x^{(i)},z^{(i)};\theta^{(t)})}{Q_i(z^{(i)})}]$$

参数 $\theta^{(t+1)}$就是在最大化上式右边部分得到的，因此：

$$\begin{aligned}
\ell(\theta^{(t+1)}) &\geq \sum_i \sum_{z^{(i)}} Q_i^{(t)}(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta^{(t+1)})}{Q_i^{(t)}(z^{(i)})} \\
&\geq \sum_i \sum_{z^{(i)}} Q_i^{(t)}(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta^{(t)})}{Q_i^{(t)}(z^{(i)})} \\
&= \ell(\theta^{(t)})
\end{aligned}$$

对于上述推导，第一个式子来自 $\ell(\theta) \geq \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) log[\frac{p(x^{(i)},z^{(i)};\theta)}{Q_i(z^{(i)})}]$，这里对于任意的 θ 和 Q_i 均成立，即对于 $Q_i = Q_i^{(t)}$和 $\theta = \theta^{(t+1)}$ 也成立。之后使用了 $\theta^{(t+1)}$是最大化参数为 $\theta^{(t)}$ 的似然函数，即 $\arg\max_\theta \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) log[\frac{p(x^{(i)},z^{(i)};\theta)}{Q_i(z^{(i)})}]$。而最后一步，就像之前那样，在选择 Q_i 的时候就需要保证等号的成立。

因此，EM 算法保证了似然函数的单调收敛。在 EM 算法的描述中，我们说要一直迭代直到收敛。根据我们推导的结果，我们可以通过测试$\ell(\theta)$在两次成功的迭代中增长的幅度是不是小于某个阈值，即判断 EM 算法中$\ell(\theta)$是否收敛太慢。

注：如果我们定义 $J(Q, \theta) = \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) log[\frac{p(x^{(i)},z^{(i)};\theta)}{Q_i(z^{(i)})}]$，那么我们知道，对于之前的推导而言，$\ell(\theta) \geq J(Q, \theta)$。EM 算法也可以被看成是在 J 上的坐标下降法，其中 E-步在以 Q 为参数的最大化 J，而 M-步则是以 θ 为参数的最大化 J。

13. 高斯混合模型

13.1. 高斯混合模型回顾

根据 EM 的定义，我们重新回顾一下高斯混合中的 ϕ，μ 和 Σ 参数拟合。为了简单起见，这里我们在 M-步中仅更新φ，μ_j，而把 Σ_j 的更新留给大家自己推导。

E-步是很容易的，根据上面的推导，我们计算：

$$w^{(i)}_j = Q_i(z^{(i)} = j) = P(z^{(i)} = j \mid x^{(i)}; \phi, \mu, \Sigma)$$

这里，$Q_i(z^{(i)} = j)$ 表示在概率分布 Q_i 下，$z^{(i)}$取得 j 的概率。在 M-步，我们需要最大化以 ϕ，μ 和 Σ 为参数的似然函数。似然概率为：

$$\begin{aligned}
&\sum_{i=1}^{m}\sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \phi, \mu, \Sigma)}{Q_i(z^{(i)})} \\
&= \sum_{i=1}^{m}\sum_{j=1}^{k} Q_i(z^{(i)} = j) \log \frac{p(x^{(i)} | z^{(i)} = j; \mu, \Sigma) p(z^{(i)} = j; \phi)}{Q_i(z^{(i)} = j)} \\
&= \sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} \log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_j)^T \Sigma_j^{-1}(x^{(i)} - \mu_j)\right) \cdot \phi_j}{w_j^{(i)}}
\end{aligned}$$

接下来，我们以 μ_l 为参数，对其取偏导数，结果如下：

$$\begin{aligned}
&\nabla_{\mu_l} \sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} \log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_j)^T \Sigma_j^{-1}(x^{(i)} - \mu_j)\right) \cdot \phi_j}{w_j^{(i)}} \\
&= -\nabla_{\mu_l} \sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} \frac{1}{2}(x^{(i)} - \mu_j)^T \Sigma_j^{-1}(x^{(i)} - \mu_j) \\
&= \frac{1}{2}\sum_{i=1}^{m} w_l^{(i)} \nabla_{\mu_l} 2\mu_l^T \Sigma_l^{-1} x^{(i)} - \mu_l^T \Sigma_l^{-1} \mu_l \\
&= \sum_{i=1}^{m} w_l^{(i)} \left(\Sigma_l^{-1} x^{(i)} - \Sigma_l^{-1} \mu_l\right)
\end{aligned}$$

让偏导数结果为 0，得到 μ_l 的更新公式如下，即与我们之前笔记提到的一样。

$$\mu_l := \frac{\sum_{i=1}^{m} w_l^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_l^{(i)}},$$

接下来，再举个 M-步中的例子，即更新 ϕ_j，这样我们需要最大化的就是 $\sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} \log \emptyset_j$ 。这里有一个限制条件就是所有 ϕ_j 的和是 1，因为 $\phi_j = p(z^{(i)}=j; \phi)$。为了处理好和为 1 的限制条件，我们采用拉格朗日法，即

$$\mathcal{L}(\phi) = \sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} \log \emptyset_j + \beta(\sum_{j=1}^{k} \phi_j - 1)$$

这里的 β 是拉格朗日乘子。取偏导数得到：$\frac{\partial}{\partial \phi_j}\mathcal{L}(\phi) = \sum_{i=1}^{m} \frac{w_j^{(i)}}{\phi_j} + 1$，让该式子等于 1，则得到：$\phi_j = \frac{\sum_{i=1}^{m} w_j^{(i)}}{-\beta}$。这里$\phi_j$正比于$\sum_{i=1}^{m} w_j^{(i)}$，采用限制条件$\sum_{j=1}^{k} \phi_j = 1$，我们很容易得到 $\beta = \sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} = \sum_{i=1}^{m} 1 = m$ (这里用到了 $w^{(i)}_j = Q_i(z^{(i)}= j)$和概率和为 1，即$\sum_{i=1}^{m} w_j^{(i)} = 1$)。因此，我们得到了 M-步中更新参数 ϕ_j 的公式：

$$\phi_j := \frac{\sum_{i=1}^{m} w_j^{(i)}}{m}$$

此外，M-步中更新 Σ_j 的推导也是类似的，不过稍微复杂点，毕竟是矩阵，这里就不再赘述，而之前已经把混合高斯模型的结果给出了。

注：如果将样本看作观察值，潜在类别看作是隐藏变量，那么聚类问题也就是参数估计问题，只不过聚类问题中参数分为隐含类别变量和其他参数。大家可以去联想监督式学习的等等问题了。

13.2. 因子分析

当我们的数据 $x^{(i)} \in R^n$ 是根据多个混合高斯分布产生的，那么我们可以根据 EM 算法拟合该模型。在这样的情况下，我们通常认为我们有足够的数据来区分数据中的多个高斯分布结构。也就是说，我们的训练样本大小 m 远大于数据的维数 n。现在，我们考虑这样的一种情况，即 $n \gg m$。在这样的问题里，可能很难拟合出一个高斯模型，更别说混合高斯模型了。具体的，对于小于 n 的 m 个样本数据，如果我们使用高斯模型，那么采用最大似然估计得到的均值和协方差如下：

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

我们会发现协方差 Σ 是奇异矩阵，也就是说是 Σ^{-1} 不存在的，而且 $1/|\Sigma|^{1/2} = 1/0$。而这两项在通常情况下的多元高斯分布(Multivariate Gaussian distribution)中都是需要计算的。解决这个问题的另一个困难是高斯模型中对参数的最大似然估计会导致将所有的概率仿射(Affine，由一个线性变换接上一个平移组成，在拉格朗日对偶中也提到过)到一个数据(这里的数据 x 满足 $x=\sum_{i=1}^{m} \alpha_i x^{(i)}$，对于一些 α_i，满足$\sum_{i=1}^{m} \alpha_i = 1$)跨越的空间去。

更一般的说，除非 m 比 n 大到一定数量，否则均值和协方差的最大似然估计结果将是很差的。尽管如此，我们仍然希望通过数据来拟合高斯模型，也许能够捕获数据中方差的一些有趣结构，那么我们该怎么做这些呢？在下一部分，我们开始回顾在 Σ 上的两种可能限制，而之前的 Σ 都不能在较少的数据上给出一个满意的解决方案。我们接下来要讨论高斯分布的特性，尤其是如何发现边际高斯分布和条件高斯分布（marginal and conditonal distributions of Gaussians）。最后我们展示因子分析模型(factor analysis model)，以及在因子分析模型上应用 EM 算法。

13.2.1. 限制 Σ

如果没有足够的数据来拟合全部的协方差矩阵(covariance matrix)，我们考虑是否可以对的 Σ 的矩阵大小做一些限制。比如我们会选择拟合一个是对角矩阵(diagonal)的协方差矩阵。在这样的设置下，可以很容易得到对角矩阵Σ是满足：$\Sigma_{jj} = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2$，这里 Σ_{jj} 仅仅是协方差的对角位置的值。

回忆一下高斯分布的概率密度的等高线都是椭圆的(椭圆的中心由μ决定，而形状由 Σ 决定)，而一个具有对角矩阵 Σ 的高斯分布，这些椭圆的两个轴就和坐标轴平行了。有些时候，我们会对 Σ 做进一步的限制，即 Σ 不仅仅是对角矩阵，而对角线上的值必须都相等，即满足 $\Sigma = \sigma^2 I$，其中 I 为单位矩阵，σ^2 是我们的控制参数。这样，σ^2 可以通过最大似然估计得到：$\sigma^2 = \frac{1}{mn}\sum_{j=1}^{n}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2$，这个模型对应的高斯分布的概率密度等高线图是圆形的(在二维的情况下，而在高维情况就是球体或超球体)。

如果我们使用数据拟合全部的非限制的协方差矩阵 Σ，要求 $m \geq n+1$，这样才能使得协方差矩阵 Σ 是非奇异矩阵。而在上面的两个限制下，我们可以得到当 $m \geq 2$ 时，就可以得到 Σ 是非奇异矩阵。

然而，限制 Σ 是对角矩阵也意味着模型过程中不同下标的 x_i，x_j 是相互独立和不相关的。多数情况下，能够捕获到数据中存在的一些有趣的相关关系是非常好的。如果我们使用了上述两个限制中的任何一个，我们都会因为不独立的问题而失败。在后面，我们会讲到因子分析模型，在该模型中使用了比对角矩阵 Σ 更多的参数且捕获了数据中的一些相关关系，但是又不需要拟合全部的协方差矩阵

13.2.2. 边际和条件高斯分布

在描述因子分析模型之前，我们讨论一下，对一个服从联合多元高斯分布的随机变量，如何寻找它的边际和条件概率。

假设我们有一个向量的随机变量 x，$x=[x_1; x_2]$，这里的 $x_1 \in R^r$， $x_2 \in R^s$, and $x \in R^{r+s}$，假设 $x \sim N(\mu, \Sigma)$，其中 $\mu = [\mu_1; \mu_2]$，$\Sigma = [\Sigma_{11}, \Sigma_{12}; \Sigma_{21}, \Sigma_{22}]$，这里的 $\mu_1 \in R^r$，

$\mu_2 \in R^s$， $\Sigma_{11} \in R^{r\times r}$， $\Sigma_{12} \in R^{r\times s}$，其他依次类推。注意，协方差矩阵是对称的(symmetric)。在我们的假设下，x1 和 x2 是联合的多元高斯分布，那么 x1 的边际分布是什么呢？我们很容易得出 $E(x_1) = \mu_1$，而 $Cov(x_1) = E[(x_1-\mu_1)(x_1-\mu_1)] = \Sigma_{11}$，而根据 x_1 和 x_2 的联合协方差定义，我们有：

$$\begin{aligned}
\mathrm{Cov}(x) &= \Sigma \\
&= \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \\
&= \mathrm{E}[(x-\mu)(x-\mu)^T] \\
&= \mathrm{E}\left[\begin{pmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{pmatrix} \begin{pmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{pmatrix}^T \right] \\
&= \mathrm{E}\begin{bmatrix} (x_1-\mu_1)(x_1-\mu_1)^T & (x_1-\mu_1)(x_2-\mu_2)^T \\ (x_2-\mu_2)(x_1-\mu_1)^T & (x_2-\mu_2)(x_2-\mu_2)^T \end{bmatrix}
\end{aligned}$$

因为多元高斯分布的边际分布仍然是多元高斯分布，因此对于 x_1 的边际分布为 $x_1 \sim N(\mu_1, \Sigma_{11})$。那么在给定条件 x_2 的情况下，x_1 的概率分布是什么呢？根据多元高斯分布的定义，可以得到 $x_1|x_2 \sim N(\mu_{1|2}, \Sigma_{1|2})$，其中：

$$\begin{aligned}
\mu_{1|2} &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2-\mu_2) \\
\Sigma_{1|2} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}.
\end{aligned}$$

在下一部分，我们讲述因子分析模型的时候，这些式子都会有帮助的。

13.2.3. 因子分析模型

在因子分析模型中，我们假设(x,z)的联合分布如下，其中 $z \in R^k$ 是一个潜在变量：

$$z \sim N(0, I) \qquad x|z \sim N(\mu + \Lambda z, \Psi)$$

这里我们模型的参数是 $\mu \in R^n$，矩阵 $\Lambda \in R^{n\times k}$，对角矩阵 $\Psi \in R^{n\times n}$，其中 k 的值通常会选择一个小于 n 的值。这里，我们想象各个数据点 $x^{(i)}$是由 k 维的多元高斯分布 $z^{(i)}$抽样产生的；之后通过 k 维仿射变换 $\mu + \Lambda z^{(i)}$，将 $z^{(i)}$仿射到 R^n 空间；最后，$x^{(i)}$就是由仿射之后的结果加上一个均值为 0，协方差为Ψ的噪声 ε，即 $\varepsilon+\mu + \Lambda z^{(i)}$。

相等的，我们定义因子分析模型如下：$z \sim N(0, I)$； $\varepsilon \sim N(0, \Psi)$；$x = \mu + \Lambda z + \varepsilon$；这里的 ε 和 z 是相互独立的。现在，我们将要给出我们的模型究竟是什么分布。我们的随机变量 z 和 x 有一个联合高斯分布：

$$\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}(\mu_{zx}, \Sigma).$$

现在我们要找出 μ_{zx} 和 Σ。我们知道 $E[z]=0$，因为 $z\sim N(0, I)$。此外，我们还可以知道：$E[x] = E[\mu + \Lambda z + \varepsilon] = \mu + E[\Lambda z] + E[\varepsilon] = \mu$。把这些放到一起，我们得到：

$$\mu_{zx} = \begin{bmatrix} \vec{0} \\ \mu \end{bmatrix}$$

接下来，为了找到 Σ，我们需要计算 $\Sigma_{zz} = E[(z - E[z])(z - E[z])^T]$（$\Sigma$ 的左上角的部分），$\Sigma_{zx} = E[(z - E[z])(x - E[x])^T]$（$\Sigma$ 的右上角的部分），$\Sigma_{xx} = E[(x - E[x])(x - E[x])^T]$（$\Sigma$ 的右下方部分）。

现在，因为 $z\sim N(0, I)$，我们可以很容易得到 $\Sigma_{zz} = Cov(z) = I$，而且：

$$E[(z - E[z])(x - E[x])^T] = E[z(\mu + \Lambda z + \varepsilon - \mu)^T] = E[zz^T]\Lambda^T + E[z\varepsilon^T] = \Lambda^T$$

在这儿的最后一步推导中，用到了 $E[zz^T] = Cov(z)$ 和 $E[z\varepsilon^T] = E[z]E[\varepsilon^T] = 0$（因为 z 和 ε 是相互独立的）。相似的，我们可以得到Σ_{xx} 如下：

$$\begin{aligned} E[(x - E[x])(x - E[x])T] &= E[(\mu + \Lambda z + \varepsilon - \mu)(\mu + \Lambda z + \varepsilon - \mu)^T] \\ &= E[\Lambda zz^T\Lambda^T + \varepsilon z^T\Lambda^T + \Lambda z\varepsilon^T + \varepsilon\varepsilon^T] \\ &= \Lambda E[zz^T]\Lambda^T + E[\varepsilon\varepsilon^T] \quad = \Lambda\Lambda^T + \Psi \end{aligned}$$

把这些都放在一起，我们得到了：

$$\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \vec{0} \\ \mu \end{bmatrix}, \begin{bmatrix} I & \Lambda^T \\ \Lambda & \Lambda\Lambda^T + \Psi \end{bmatrix} \right)$$

因此，我们得到了 x 的边际分布，$x\sim N(\mu, \Lambda\Lambda^T + \Psi)$。因此，在给定训练集合$\{x^{(i)}; i = 1, \ldots, m\}$，我们可以写下参数的最大似然函数如下：

$$\ell(\mu, \Lambda, \Psi) = \log \prod_{i=1}^{m} \frac{1}{(2\pi)^{n/2}|\Lambda\Lambda^T + \Psi|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu)^T(\Lambda\Lambda^T + \Psi)^{-1}(x^{(i)} - \mu) \right)$$

为了进行最大似然估计，我们希望最大化该似然函数。然而，直接进行最大化是很困难的，我们没有学过一个算法能够解决这个闭型解(closed-form)。因此，我们将采用 EM 算法，在下一部分我们会用 EM 推导因子分析

14. 主成分分析法

14.1. 因子分析

14.1.1. EM 算法求解因子分析

对于 EM 算法而言，E-步是非常简单的，我们只需要计算 $Q_i(z^{(i)}) = p(z^{(i)}|x^{(i)}; \mu, \Lambda, \Psi)$。然而在这里的条件分布为，$z^{(i)}|x^{(i)}; \mu, \Lambda, \Psi \sim N(\mu_{z(i)|x(i)}, \Sigma z^{(i)}|x^{(i)})$，这里满足：

$$\begin{aligned}\mu_{z^{(i)}|x^{(i)}} &= \Lambda^T(\Lambda\Lambda^T + \Psi)^{-1}(x^{(i)} - \mu), \\ \Sigma_{z^{(i)}|x^{(i)}} &= I - \Lambda^T(\Lambda\Lambda^T + \Psi)^{-1}\Lambda.\end{aligned}$$

由此，我们得到了 E-步的更新公式：

$$Q_i(z^{(i)}) = \frac{1}{(2\pi)^{k/2}|\Sigma_{z^{(i)}|x^{(i)}}|^{1/2}} \exp\left(-\frac{1}{2}(z^{(i)} - \mu_{z^{(i)}|x^{(i)}})^T\Sigma^{-1}_{z^{(i)}|x^{(i)}}(z^{(i)} - \mu_{z^{(i)}|x^{(i)}})\right)$$

现在，我们想办法解决 M-步。我们需要最大化的下式：

$$\sum_{i=1}^{m}\int_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \mu, \Lambda, \Psi)}{Q_i(z^{(i)})} dz^{(i)}$$

这里的参数是 μ, Λ, Ψ。这里仅给出 Λ 的推导过程。根据我们需要最大化的式子，我们可以化简如下：

$$\begin{aligned}&\sum_{i=1}^{m}\int_{z^{(i)}} Q_i(z^{(i)}) \left[\log p(x^{(i)}|z^{(i)}; \mu, \Lambda, \Psi) + \log p(z^{(i)}) - \log Q_i(z^{(i)})\right] dz^{(i)} \\ &= \sum_{i=1}^{m} \mathrm{E}_{z^{(i)}\sim Q_i}\left[\log p(x^{(i)}|z^{(i)}; \mu, \Lambda, \Psi) + \log p(z^{(i)}) - \log Q_i(z^{(i)})\right]\end{aligned}$$

这里的下标 $z^{(i)}\sim Q_i$ 表示 $z^{(i)}$服从 Q_i 分布。然后去掉与 Λ 无关的项。得到我们最终想要最大化的结果：

$$\begin{aligned}&\sum_{i=1}^{m} \mathrm{E}\left[\log p(x^{(i)}|z^{(i)}; \mu, \Lambda, \Psi)\right] \\ &= \sum_{i=1}^{m} \mathrm{E}\left[\log \frac{1}{(2\pi)^{n/2}|\Psi|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu - \Lambda z^{(i)})^T\Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)})\right)\right] \\ &= \sum_{i=1}^{m} \mathrm{E}\left[-\frac{1}{2}\log|\Psi| - \frac{n}{2}\log(2\pi) - \frac{1}{2}(x^{(i)} - \mu - \Lambda z^{(i)})^T\Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)})\right]\end{aligned}$$

这里，只有最后一项是依赖于 Λ 的。对其求偏导数，如下：

$$
\begin{aligned}
&\nabla_{\Lambda} \sum_{i=1}^{m} -\mathrm{E}\left[\frac{1}{2}(x^{(i)}-\mu-\Lambda z^{(i)})^{T}\Psi^{-1}(x^{(i)}-\mu-\Lambda z^{(i)})\right] \\
&= \sum_{i=1}^{m} \nabla_{\Lambda} \mathrm{E}\left[-\mathrm{tr}\frac{1}{2}z^{(i)^{T}}\Lambda^{T}\Psi^{-1}\Lambda z^{(i)}+\mathrm{tr}z^{(i)^{T}}\Lambda^{T}\Psi^{-1}(x^{(i)}-\mu)\right] \\
&= \sum_{i=1}^{m} \nabla_{\Lambda} \mathrm{E}\left[-\mathrm{tr}\frac{1}{2}\Lambda^{T}\Psi^{-1}\Lambda z^{(i)}z^{(i)^{T}}+\mathrm{tr}\Lambda^{T}\Psi^{-1}(x^{(i)}-\mu)z^{(i)^{T}}\right] \\
&= \sum_{i=1}^{m} \mathrm{E}\left[-\Psi^{-1}\Lambda z^{(i)}z^{(i)^{T}}+\Psi^{-1}(x^{(i)}-\mu)z^{(i)^{T}}\right]
\end{aligned}
$$

上述的推导，使用了矩阵的一些公式，对于 $a \in R$，$\mathrm{tr}\, a = a$(tr 表示取矩阵的迹)，$\mathrm{tr}AB = \mathrm{tr}BA$，$\nabla A \mathrm{tr}ABA^{T}C = CAB + C^{T}AB$。之后，让其等于 0，化简结果如下：

$$
\sum_{i=1}^{m} \Lambda \mathrm{E}_{z^{(i)}\sim Q_i}\left[z^{(i)}z^{(i)^{T}}\right] = \sum_{i=1}^{m}(x^{(i)}-\mu)\mathrm{E}_{z^{(i)}\sim Q_i}\left[z^{(i)^{T}}\right]
$$

因此，我们得到 Λ 的值如下：

$$
\Lambda = \left(\sum_{i=1}^{m}(x^{(i)}-\mu)\mathrm{E}_{z^{(i)}\sim Q_i}\left[z^{(i)^{T}}\right]\right)\left(\sum_{i=1}^{m}\mathrm{E}_{z^{(i)}\sim Q_i}\left[z^{(i)}z^{(i)^{T}}\right]\right)^{-1}
$$

这里我们发现一个有趣的现象。这里得到的 Λ 结果和回归模型中最小二乘法的方程结果很类似($\theta^{T} = (y^{T}X)(X^{T}X)^{-1}$)。这里类比一下，x 是 z 的线性函数(包含了一定的噪声)，在 E-步中给出了 z 的估计之后，我们需找的 Λ 实际上是 x 和 z 的线性关系。而最小二乘法也是去寻找特征和结果的直接的线性关系。而这两者很重要的不同点在于最小二乘用的 z 是最好的猜测(也就是所谓的观测到的分类)。之后，我们会看到这些不同。

为了完成 M-步的更新，我们需要求解出 Λ 结果中的各个期望值。从开始的 Q_i 定义，我们很容易得到：

$$
\begin{aligned}
\mathrm{E}_{z^{(i)}\sim Q_i}\left[z^{(i)^{T}}\right] &= \mu^{T}_{z^{(i)}|x^{(i)}} \\
\mathrm{E}_{z^{(i)}\sim Q_i}\left[z^{(i)}z^{(i)^{T}}\right] &= \mu_{z^{(i)}|x^{(i)}}\mu^{T}_{z^{(i)}|x^{(i)}} + \Sigma_{z^{(i)}|x^{(i)}}.
\end{aligned}
$$

第一步是根据 z 的条件分布得到的，第二步是根据 $\mathrm{E}[Y\,Y^{T}] = \mathrm{E}[Y]\mathrm{E}[Y]^{T} + \mathrm{Cov}(Y)$得到的。再带回到 Λ 的结果中，得到了 M-步中的更新如下：

$$
\Lambda = \left(\sum_{i=1}^{m}(x^{(i)}-\mu)\mu^{T}_{z^{(i)}|x^{(i)}}\right)\left(\sum_{i=1}^{m}\mu_{z^{(i)}|x^{(i)}}\mu^{T}_{z^{(i)}|x^{(i)}} + \Sigma_{z^{(i)}|x^{(i)}}\right)^{-1}.
$$

这里很重要的一点是右侧等式中的$\Sigma_{z^{(i)}|x^{(i)}}$，这是后验分布 $p(z^{(i)}|x^{(i)})$的协方差，M-步中一定要考虑后验分布的 $z^{(i)}$不确定性。在使用 EM 中的一个常见错误是在 E-步中，假设我们需要计算潜在变量 z 的期望 E[z]，然后加入到 M-步中任何出现 z 的优化中。这个在简单的问题，比如混合高斯模型中，在因子分析的推导中，我们需要 $E[zz^T]$和 E[z]。如我们所见，E[zzT]和 E[z]E[z]T 需要减去$\Sigma_{z|x}$。因此，在 M-步的更新中考虑后验分布 $p(z^{(i)}|x^{(i)})$的协方差。

最后，给出在 M-步中更新参数 μ 和 Ψ 的公式。其中，μ 在迭代过程中保持不变。

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

$$\Phi = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}x^{(i)^T} - x^{(i)}\mu_{z^{(i)}|x^{(i)}}^T\Lambda^T - \Lambda\mu_{z^{(i)}|x^{(i)}}x^{(i)^T} + \Lambda(\mu_{z^{(i)}|x^{(i)}}\mu_{z^{(i)}|x^{(i)}}^T + \Sigma_{z^{(i)}|x^{(i)}})\Lambda^T$$

根据上面的 EM 的过程，要对样本 X 进行因子分析，只需知道要分解的因子数（z 的维度）即可。通过 EM，我们能够得到转换矩阵 Λ 和误差协方差 Ψ。因子分析实际上降维，在得到各个参数后，可以求得 z，而 z 的各个参数的含义需要自己去琢磨了。

附录：上面的推导比较理性，这里提供其他的一些说法。因子分析(factor analysis)是一种数据简化的技术。它通过研究众多变量之间的内部依赖关系，探求观测数据中的基本结构，并用少数几个假想变量来表示其基本的数据结构。这几个假想变量能够反映原来众多变量的主要信息。原始的变量是可观测的显在变量，而假想变量是不可观测的潜在变量，称为因子。

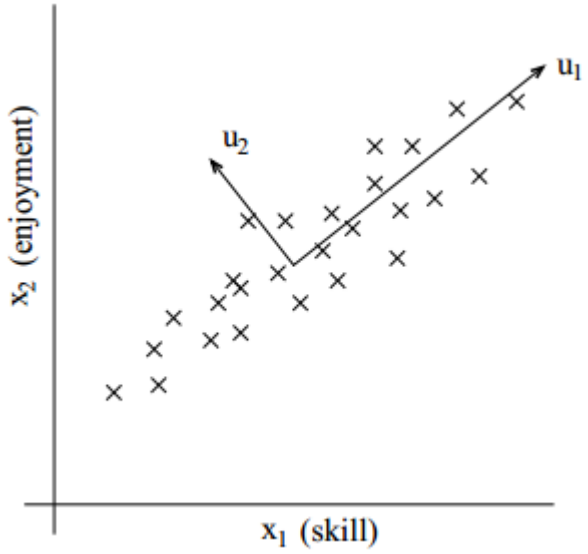
14.2. 主成分分析

在我们讨论因子分析的时候，我们给出了一个模型，即将 $x \in R^n$ 近似的落到 k 维度子空间，这里的 $k \ll n$。特别的，我们想象得到：每个数据点 x(i)都是由 k 维度下的 $z^{(i)}$通过仿射$\{\Lambda z + \mu; z \in R^k\}$变化之后，增加一个均值为 0，协方差为Ψ的噪声得到的。因子分析是基于概率模型的，而参数的估计则是使用了 EM 算法。在这部分，我们将讲述一个模型主成分分析(PCA， Principal Components Analysis)，也是尝试定义一个数据可以映射的子空间。然而，PCA 将会采用更直接的方法，但仅要求一些特征向量的计算(在 matlab 里面，仅仅是命令 eig 就可以实现)，且不需要使用 EM 算法。

对于给定的数据集$\{x^{(i)}; i = 1,\ldots, m\}$，包含了汽车的 m 个不同特征(attribute，表示原始特征或属性)，比如最大速度、转速等等。对于每一个 i，$x^{(i)} \in R^n (n \ll m)$，但是，我们不知道其中有两个特征 x_i 和 x_j 都是汽车最大速度，但是前者以“米/小时”为单位，后者以

“千米/小时”为单位。因此，这两个特征基本上是线性相关的(因为单位换算的一些四射五入的问题，可能不完全不相同)。因此，这些数据就可以近似的落在 n-1 维子空间。

我们举另一个例子。考虑一个无线遥控直升飞机(RC 直升飞机，radio-controlled helicopters)飞行员的调查数据，这里的 $x^{(i)}_1$ 衡量了飞行员 i 的飞行技巧。$x^{(i)}_2$ 刻画了他是否喜欢飞行。因为 RC 直升飞机很难操作，只有那些最投入的学生，真正喜欢飞行的学生才能成为好的飞行员。因此 x_1 和 x_2 是强相关的。事实上，我们可以通过画图来刻画这种现象，比如飞行员 karma，如下图所示，数据基本上都坐落在方向μ_1 上，也有一些噪声在该方向上(这里的μ就是我们想要获得的)。



之后我们会具体的看到 PCA 算法。但是在做 PCA 之前，通常我们需要先把数据进行标准化(normalize)，即按如下方式标准化：

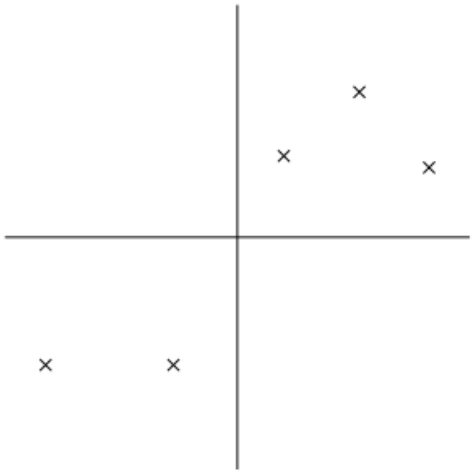
1) 取$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$
2) 之后，用 $x^{(i)} - \mu$ 来替换 $x^{(i)}$
3) 取 $\sigma^2_j = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu)^2$
4) 之后，用 $x_j^{(i)}/\sigma_j$ 来替换$x_j^{(i)}$

这里的第 1-2 步是减去均值，这样最后的数据的均值就是 0，而 3-4 步则是处以标准差(这里计算中没有使用 m-1，通常建议使用 m-1)，这样得到的数据最终的方差就是 1，即保证了各个特征是在同一个范围内。如果各个数据已经在同一个范围或者范围差距不大的话，那么 3-4 是可以省略的。

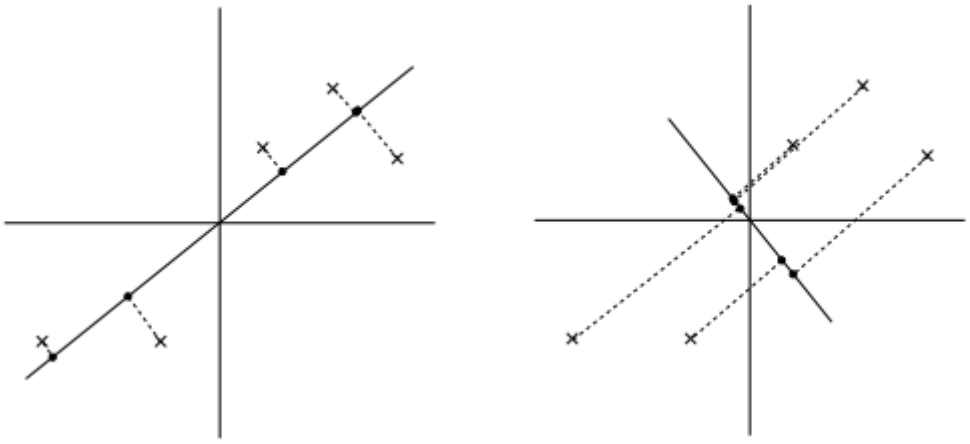
现在，我们已经完成了标准化，那么我们该如何刻画“变量的主要轴” μ(即数据近似落在的方向)呢？一个方法是找到一个单位向量μ，使得当数据投射(projected)到向量μ所

在的方向时，投射后数据的方差最大。直观的看，数据开始的时候具有一定量的方差(或信息)，我们希望选择一个方向μ，使得数据落在该方向上后，能够尽可能保持这样的方差(或信息)。（注：在信号处理中认为信号具有较大的方差，噪声有较小的方差，信噪比就是信号与噪声的方差比，越大越好。如前面的图，样本在μ_1的投影方差较大，在μ_2上的投影方差较小，那么认为μ_2上的投影是由噪声引起的。这在最后会有进一步的说明。）

考虑下面一系列数据，这些数据已经经过标准化处理了。



现在考虑我们选择了一个μ，圆点表示原始数据点投射到该向量方向上的点。我们看到：如下图左侧所示，投射后数据具有相当大的方差，且数据点都远离原点。想反，如下图右侧所示，投射后的点具有很小的方差，而且距离原点很近。



我们希望能够自动选择一个类似于上图左侧的单位向量。为了正式的给出，我们如下表示：对于给定的单位向量μ和一个点 x，x 投射到向量μ上后的投影长度为 $x^T\mu$。比如，如果 $x^{(i)}$是我们数据集中一个点（在二维坐标图上），那么他投射到μ(类似于图上的圆点)。

那么投射后点到原点的距离是 $x^T\mu$。故问题转化为最大化投影的方差，因为均值为 0，且μ是单位长度的向量，因此我们得到需要最大化的公式如下：

$$\frac{1}{m}\sum_1^m (x^{(i)^T}\mu)^2 = \frac{1}{m}\sum_{i=1}^m \mu^T x^{(i)} x^{(i)^T}\mu = u^T\left(\frac{1}{m}\sum_{i=1}^m x^{(i)}x^{(i)^T}\right)\mu$$

我们可以很容易的看出该最大化受限于$||u||^2 = 1$，因此该最大化给出了主特征向量 $\Sigma = \frac{1}{m}\sum_{i=1}^m x^{(i)}x^{(i)^T}$，而这恰好是训练集合数据的协方差矩阵（我们假设了数据的均值为 0）

总结一下，如果我们希望找到一个一维子空间来近似原始数据，那么我们应该选择的向量是Σ的主特征向量。更普遍的说，如果我们希望把我们的数据投射到 k 维子空间，我们应该选择的向量 $u_1, \ldots, u_k$ 应该是Σ的前 k 个特征向量。这里 u_1 对于数据而言，是正交的。

在这部分，为了表示 $x^{(i)}$，我们只需要计算相应的矩阵：

$$y^{(i)} = \begin{bmatrix} u_1^T x^{(i)} \\ u_2^T x^{(i)} \\ \vdots \\ u_k^T x^{(i)} \end{bmatrix} \in \mathbb{R}^k.$$

这里 $x^{(i)} \in R^n$，而向量 $y^{(i)}$ 现在是一个低纬度的、k 维度的，可以近似或替代 $x^{(i)}$。PCA 因此也可以被称为降维(dimensionality reduction)算法，向量 $u_1, \ldots, u_k$，称之为数据的 k 主成分(principal components)。

注意：尽管上面正式给出的结果是在 k=1 的情况，而实际上根据特征向量的性质，很容易得到所有的可能正交基(orthogonal bases)$u_1, \ldots, u_k$，用于最大化$\sum_i ||y^{(i)}||^2$，因此，我们选择一个正交基来尽可能的保留原始数据的方差。

PCA 有很多应用。接下来，我们说一些应用。首先是数据压缩(compression)是一个明显的例子，即用低维度的 $y^{(i)}$来表示 $x^{(i)}$。如果把高维度数据降到 k=2 或 3，那么我们就可以很容易的可视化 $y^{(i)}$，比如如果我们把直升机数据降到 2 维，那么我们就可以画出来(比如一个轴可能是汽车的类型)去观察各个车可能的聚类结果。

另外一个标准应用是，在监督式学习之前对输入数据 $x^{(i)}$进行 PCA 处理。这不仅仅可以降低计算量，同时降维能够降低假设类(hypothsis class)的复杂度来避免过拟合(比如线性分类器在低维度上的输入空间会有较低的 VC 维度)。

最后，在我们这个 RC 直升飞机的例子中，我们也可以认为 PCA 是一种降低噪音的算法。在我们的例子中，可以看成是从有噪声的飞行员的飞行技术和兴趣度的数据中，估计其内在的飞行能力(即不受噪声干扰的飞行能力)。而课程中，我们也能看到另一个案例，

即人脸图片中的特征脸(eigenface)方法。即每张图片 $x^{(i)} \in R^{100\times100}$，是 10000 维的向量，向量中的每个值表示了人脸图片中 100*100 的各个像素点。使用 PCA，我们可以用更低维度的 $y^{(i)}$来表示每张图片 $x^{(i)}$，在这样做的过程中，我们希望找到的主成分能够保留捕获不同人脸中一个人是什么样子的主要的变化，但是要去掉因为光线变化、不同的成像条件等造成的噪声。之后我们使用“降维”之后的数据，测试两个脸 i，j 之间的距离即$||y^{(i)}-y^{(j)}||^2$，最后这让人脸匹配和检索算法的结果令人吃惊的好。

注意：关于 PCA 的解释有 9-10 多种解释。这里只是给出了最大方差解释，也有最小平方误差解释(把原始数据映射到子空间，衡量子空间好坏的方式不再是方差，而是使得映射后的点到原始点的距离的平方最小)等等。

聚类(k-means)、因子分析、主成分分析，这三种是相互联系的。K-means 可以看成是对样本点的划分(这里 $m > n$)，而后两者($m < n$)是对变量的划分，而这里如果我们把特征数 n 和样本数 m 互换一下，发现这三者是同一性质的。这些都是非监督式学习，可以把非监督式学习看成是含有潜在类别的监督式学习(当然潜在变量是未知的，所以称不上是监督，这里仅是为了对比而已)，这三个方法的主要不同在于：对潜在类别的解释不同。k-means 的潜在分类是针对于样本的硬猜测，使得簇内间距很小而簇间间距较大；而因子分析的潜在类别是针对变量的软猜测，潜在变量是服从多元高斯分布；主成分分析的潜在类别则是最优子空间(不同的 PCA 解释主要区别在于如何定义最优子空间)。

15. 奇异值分解

15.1. 潜在语义索引

作为 PCA 的经典应用之一，是在文本分类中，这样的方法有一个专有的名字，叫潜在语义索引(LSI , laten semantic indexing)。这部分需要注意的是，在文本分类中，不需要先进行归一化处理（PCA 要求归一化处理），因为这里考虑了词语出现的次数。鉴于课件空缺，这里从网上补充资料。

LSI 的基本思想是文本中的词与词之间不是孤立的，存在着某种潜在的语义关系，通过对样本数据的统计分析，让机器自动挖掘出这些潜在的语义关系，并把这些关系表示成计算机可以"理解"的模型。它可以消除词匹配过程中的同义和多义现象。它可以将传统的 VSM 降秩到一个低维的语义空间中，在该语义空间中计算文档的相似度等。总的说来，LSI 就是利用词的语义关系对 VSM 模型进行降维，并提高分类的效果。

LSI 的降维过程

1.将文档库表示成 VSM 模型的词-文档矩阵 Am×n(词-文档矩阵那就是词作为行，文档作为列，这是矩阵先行后列的表示决定的，当然如果表示成文档-词矩阵的话，后面的计算就要用该矩阵的转置了),其中 m 表示文档库中包含的所有不同的词的个(行数是不同词的个数)，即行向量表示一个词在不同文档出现的次数，n 表示文档库中的文档数(列数是不同文档的个数)，即列向量表示的是不同的文档.A 表示为 $A = [\alpha_{ij}]$,在此矩阵中 ,α_{ij} 为非负值 , 表示第 i 个词在第 j 个文档中出现的频度。显然，A 是稀疏矩阵(这是 VSM 和文档决定的)。

2.利用奇异值分解 SVD(Singular Value Decomposition)求 A 的只有 K 个正交因子的降秩矩阵，该过程就是降维的过程。SVD 的重要作用是把词和文档映射到同一个语义空间中，将词和文档表示为 K 个因子的形式。显然，这会丢失信息，但主要的信息却被保留了。为什么该过程可以降维呢？因为该过程解决了同义和多义现象。可以看出，K 的取值对整个分类结果的影响很大。因为，K 过小，则丢失信息就越多；K 过大，信息虽然多，但可能有冗余且计算消耗大。K 的选择也是值得研究的，不过一般取值为 100-300，不绝对。

15.2. 奇异值分解

注：关于 SVD 的，在以后会继续补充，可以参考：
http://www.cnblogs.com/LeftNotEasy/archive/2011/01/19/svd-and-applications.html。

15.3. 独立成分分析

15.3.1. 独立成分分析引入

我们接下来的话题是独立成分分析(ICA，Independent Component Analysis)。类似于 PCA，我们将会找到新的基础来表示我们的数据。然而，两者的目标是不同的。

考虑一下经典的鸡尾酒宴会问题(cocktail party problem)， 假设在宴会中有 n 个人在同时说话，房间中一些角落里共放置了 n 个声音接收器（Microphone）用来记录声音（这里记录的声音是多个人的混合结果）。这里，我们假设 n 个声音接收器放在不同的地方，这样说话的人距离不同的声音接收器的距离均不相同。使用这个声音接收器，我们是否能够区分出不同的人发出的原始声音呢？为了正式引入问题，我们想象一组数据 $s \in R^n$，由 n 个独立的信号源产生。我们观测到的是：$x = As$，这里的 A 是未知的方阵，称之为混合矩阵(mixing matrix)。不断的观测，我们得到数据集$\{x^{(i)}; i = 1, \ldots, m\}$，我们的目标是重新获得产生我们数据($x^{(i)}=A\ s^{(i)}$)信号源 $s^{(i)}$。在鸡尾酒宴会问题中，$s^{(i)}$是一个 n 维的向量，$s^{(i)}_j$是说话人 j 在时间 i 时发出的声音信号。同时，$x^{(i)}$是一个 n 维的向量，$x^{(i)}_j$是声音接收器 j 在时间 i 时接受到的声音信号。

引入 $W = A^{-1}$，称之为分离矩阵(unmixing matrix)。我们的目标是找到 W，这样对于声音接收器记录的信号 $x^{(i)}$，我们可以通过计算 $s^{(i)} = W\ x^{(i)}$求得信号源信号。为了方便表达，我们取 W 的第 i 个行为 w^T_i为，因此有如下表示：

$$W = \begin{bmatrix} — & w_1^T & — \\ & \vdots & \\ — & w_n^T & — \end{bmatrix}$$

因此，$w_i \in R^n$，第 j 个信号源可以这样恢复：$s^{(i)}_j = w^T_j x^{(i)}$

15.3.2. ICA 的不确定性(ICA ambiguities)

我们通过 $W=A^{-1}$ 能够恢复多少呢？这里，我们并没有任何关于信号源和混合矩阵的先验知识，而 A 存在很大的不确定性，因此仅靠 x(i)是无法完整的恢复 s（见下例子）。

比如，对于 n*n 维度的置换矩阵(permutation matrix) P，这意味着 P 的每一行每一列恰好仅有一个 1，举例如下：

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \quad P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \quad P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

如果 z 是一个向量，Pz 的结果是置换了 z 的坐标后的向量，如 z=[1,2]，Pz = [2,1]。对于仅给定 $x^{(i)}$，那么我们没办法区分 W 和 PW（即等式均成立，只是最终的 s 排列不同）。也就是说，信号源的排列是有不确定性的。不过，这些都不影响我们绝大多数的应用。

另外，如果我们对 W 进行缩放变换，我们也没有办法正确的恢复 s。即如果 A 变成 2A，对于每一个 $s^{(i)}$，只要乘以 0.5 就可以满足等式，即 $x^{(i)} = 2A*0.5*s^{(i)}$。因此，我们不能恢复正确比例的信号源。然而，对于应用而言，比如鸡尾酒宴会问题，这种不确定性问题也无关紧要。通过把说话人的声音信号 $s^{(i)}_j$ 乘以正数α的结果，只是放大了这个人的声音音量而已。而且正负号变化也不影响，$s^{(i)}_j$ 和-$s^{(i)}_j$ 对于每一个说话的人都是唯一的。因此，如果 w_i 在算法中被乘以一个非 0 的实数，那么对于恢复的信号源 $s_i = w^T_i x$，只要乘以相同的系数就可以。但是，通常情况下，影响不大。（也可以把 ICA 用于脑电图数据分析等等）

那么这些就是 ICA 的不确定性的所有来源吗？事实证明，这些只在 s_i 是非高斯分布的情况才成立。为了了解与高斯分布数据的不同之处，我们考虑一个例子：对于 n=2，s~ N (0, I)，I 是一个 2*2 的单位矩阵。回想一下标准正态分布 N(0,I)的概率密度的等高线图，是中心在原点的一个个圆，概率密度是对称的。现在，假设我们观测到一些数据 x = As，这里的 A 是我们的混合矩阵。x 服从高斯分布，其均值为 0，方差为 $E[xx^T] = = E[Ass^TA^T] = AA^T$。现在，取 R 为任意一正交矩阵（或者旋转矩阵），那么 $RR^T = R^TR = I$。令 A′ = AR。那么如果数据根据 A′而不是 A 混合在一起，那么我们将得到 x′ = A′s，那么 x′ 也是服从高斯分布，均值为 0，方差为 $E[x'(x')^T] = E[A'ssT(A')^T] = E[ARss^T(AR)^T] = ARR^TA^T = AA^T$，因此，对于混合矩阵 A，A′，我们观测到的数据均产生自 $N(0,AA^T)$，那么我们就无法区分信号源是根据 A 还是 A′混合得到的。因此，在混合矩阵中存在观测数据无法决定的一个旋转矩阵，那么我们无法恢复原始信号。

我们上面的讨论是基于一个事实，即多元标准正态分布是旋转对称的。对于在高斯分布的数据集上使用 ICA 是很难有效果的。而对于非高斯分布数据，只要有足够的数据，那么恢复 n 个独立信号源是可能的。

15.3.3. 密度函数和线性变换

在正式的导出 ICA 算法之前，我们先简单的讨论一下线性变换对密度函数的影响。

假设我们有一个随机变量，概率密度是 $p_s(s)$，为了简便，我们认为 $s\in R$，是一个实数，现在去一个随机变量 $x=As$(这里 $x\in R$，$A\in R$)，p_x 是 x 的概率密度，那么 p_x 是多少呢？这里取 $W=A^{-1}$，为了计算对于给定 x 值的概率，可以先计算出 $s=Wx$，然后在那个点上估计出 ps，然后得到 $p_x(x) = p_s(Wx)$。然而，这是不正确的。举例而言，s~ U[0, 1](Uniform,均匀分布)，那么 s 的概率密度 $p_s(s)=1\ \{0\le s\le 1\}$，这里我们使 A=2，那么 $x=2s$，x 是服从 U[0,2]。那么 x 的概率密度 $p_x(x)=0.5\{0\le x\le 2\}$，但是这并不等于 $p_s(Wx)$，其中 W=0.5，而是满足 $p_x(x) = p_s(Wx)|W|$。

更普遍而言，如果 s 是一个向量，值服从概率密度为 p_s 的分布，$x=As$，且 A 是一个可逆方阵，那么 x 的概率密度满足：$p_x(x) = p_s(Wx)|W|$，这里 $W=A^{-1}$。

推导如下：

$$F_X(x) = P(X \le x) = P(AS \le x) = P(S \le Wx) = F_S(Wx)$$

$$p_x(x) = F_X'(x) = F_S'(Wx) = p_s(Wx)|W|$$

15.3.4. ICA 算法

现在开始正式的引入 ICA 算法。这里描述的 ICA 是源自 Bell 和 Sejnowski 的论文，这里的算法解释使用了最大似然估计（而原文中使用了更加复杂的方法，称之为 infomax principal）。

假设每个信号源 s_i 的分布的概率密度是 p_s，对于多个信号源的联合分布是：$p(s) = \prod_{i=1}^{n} p_s(s_i)$，注意这里指所有能用乘法，是因为我们假设了各个信号源是相互独立的。根据之前提到的公式，我们知道：对于 $x=As=W^{-1}s$，$p(x)= \prod_{i=1}^{n} p_s(W_i^T x)\ |W|$

回忆一下，对于给定的一个实数随机变量 z，它的累计分布函数(cdf, cumulative distribution function) F 是 $F(z_0) = P(z\le z_0) = \int_{-\infty}^{z_0} p_z(z)dz$，同时 z 的概率密度可以通过对累计分布函数求导得到，即 $p_z(z) = F'(z)$。

因此，为了求得 s 的概率密度，我们需要知道 s 的累计分布函数。cdf 一定是一个单调递增的函数，根据我们之前的讨论，因为 ICA 不能用于高斯分布的数据上，故我们选择的 cdf 不能是高斯 cdf。我们将要选择一个合理的默认函数来替代 cdf，该函数结果在 0-1 之间，且单调递增，很自然的我们想到了 sigmoid 函数，即 $g(s) = 1/(1+\exp(-s))$，因此 $p_s(s) = g'(s) = \exp(s) / (1+\exp(s))^2$。方阵 W 是我们模型的参数，对于给定的训练集合$\{x^{(i)}; i = 1, \ldots, m\}$，对数似然函数如下：

$$\ell(W) = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} \log g'(w_j^T x^{(i)}) + \log |W| \right)$$

我们希望最大化似然函数以求得到 W，$\nabla_W|W| = |W|(W^{-1})^T$ (在梯度下降法中提到)，得到 W 的更新规则：

$$W := W + \alpha \left(\begin{bmatrix} 1 - 2g(w_1^T x^{(i)}) \\ 1 - 2g(w_2^T x^{(i)}) \\ \vdots \\ 1 - 2g(w_n^T x^{(i)}) \end{bmatrix} x^{(i)T} + (W^T)^{-1} \right)$$

这里的α是学习速率。在算法收敛之后，我们只需要计算 $s^{(i)} = Wx^{(i)}$，即可恢复得到原始的信号。

注意：在我们使用最大似然估计时，就假设了 x 之间是相互独立的，然而对于语音信号或者其他具有时间连续依赖特性(比如温度)上，这个假设是不成立的。但是，在数据足够多时，独立假设对最终效果影响不大，同时如果事先打乱样本，并运行随机梯度上升算法，那么能够加快收敛速度。

16. 马尔可夫决策过程

16.1. 概述

现在我们开始讨论增强学习(RL，reinforcement learning)和自适应控制(adaptive control)。在监督式学习中，我们的算法总是尝试着在训练集合中使预测输出尽可能的模仿（mimic）实际标签 y（或者潜在标签）。在这样的设置下，标签明确的给出了每个输入 x 的正确答案。然而，对于许多序列决策和控制问题（sequential decision making and control problems），很难提供这样的明确的监督式学习。比如我们现在正在做一个四条腿的机器人，目前我们正在尝试编程让他能够行走，在最开始的时候，我们就无法定义什么是正确的行走，那么我们就无法提供一个明确的监督式学习算法来尝试模仿。

在增强学习框架下，我们的算法将仅包含回报函数(reward function)，该函数能够表明学习主体(learning agent)什么时候做的好，什么时候做的不好。在四足机器人行走例子中，回报函数可以是在机器人向前移动时，给予机器人积极的回报，在机器人后退后者跌倒时给予消极的回报。那么我们的学习算法任务是如何随着时间的变化而选择一个能够让回报最大的行动。

目前增强学习已经能够成功的应用在不同的自主直升飞机驾驶、机器人步态运动、手机网络路由、市场策略选择、工厂控制和不同的网页检索等等。在这里，我们的增强学习将从马尔可夫决策过程(MDP， Markov decision processes)开始。

16.2. 马尔可夫决策过程

一个马尔可夫决策过程是一个元组 $(S, A, \{P_{sa}\}, \gamma, R)$，其中（以自主直升飞机驾驶为例）：

- S 是状态(states)集合，例：直升飞机的所有可能的位置和方向的集合。
- A 是动作(actions)集合，例：可以控制直升飞机方向的方向集合
- P_{sa} 是状态转移概率，例：对于每个状态 $s \in S$，动作 $a \in A$，P_{sa} 是在状态空间的一个分布。之后我们会详细介绍，简而言之 P_{sa} 给出了在状态 s 下采取动作 a，我们会转移到其他状态的概率分布情况。
- $\gamma \in [0, 1)$，称之为折现因子(discount factor)
- $R: S \times A \rightarrow R$ 是回报函数。有些时候回报函数也可以仅仅是 S 的函数。

MDP 动态过程如下：我们的学习体(agent)以某状态 s_0 开始，之后选择了一些动作 $a_0 \in A$ 并执行，之后按照 Psa 概率随机转移到下一个状态 s_1，其中 $s_1 \sim P_{s_0 a_0}$。之后再

选择另一个动作 $a_1 \in A$ 并执行，状态转移后得到 $s_2 \sim P_{s_1 a_1}$，之后不断的继续下去。即如下图所示：

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} \dots$$

基于上面一系列的在状态 s_0，s_1，…，伴随着动作 a_0，a_1，…，我们最终的回报为：$R(s_0,a_0)+\gamma R(s_1,a_1)+\gamma^2 R(s_2,a_2)+\dots$当我们仅以状态来定义回报，总回报为 $R(s_0)+\gamma R(s_1)+\gamma^2 R(s_2)+\dots$，在多数情况下，尽管基于状态-动作的回报函数 R(s,a)并不是很难给出，但是我们仍将会选择较为简单的基于状态的回报函数 R(s)。

在增强学习中，我们的目标就是在时间过程中，选择合适的动作，使得最终的回报期望最大，其中回报期望为 $E[R(s_0)+\gamma R(s_1)+\gamma^2 R(s_2)+\dots]$。注意在时间步 t 下的回报函数的折现因子为$\gamma^t$。为了使得期望最大，我们希望过程中尽可能的取得积极的回报（也尽可能的减少消极回报）。在经济学应用中，R(·)是赚取的钱的数量，而γ则可以解释为利率（即今天的一美元比明天的一美元更值钱）。

一个策略(policy)是对于任意函数$\pi : S \to A$，即从状态到动作函数。无论什么时候，如果我们在状态 s，那么我们就会执行一些策略π，使得我们的动作 $a=\pi(s)$。对于策略π，我们定义值函数(value function)为：$V^\pi(s)=E[R(s_0)+\gamma R(s_1)+\gamma^2 R(s_2)+\cdots|s_0=s,\pi]$，其中$V^\pi(s)$是在状态为 s 下，采取基于策略π而选择行动 a 的各个折现回报总和的期望（the expected sum of discounted rewards）。

对于一个策略π，它的值函数 V^π满足贝尔曼等式(Bellman equations)：

$$V^\pi(s)= R(s) +\gamma\sum_{s'\in S} P_{s\pi(s)}(s')V^\pi(s')$$

这说明折现回报总和的期望 $V^\pi(s)$包含两项：其一是即时汇报(immediate reward) R(s)，即在开始状态 s 就可以直接得到的汇报。而第二项是未来折现回报总和的期望。进一步看第二项，求和项可以写成 $E_{s'\sim P_{s\pi}(s)}[V^\pi(s')]$，这是在状态 s’ 下的折现回报总和的期望，这里的 s’ 满足 $P_{s\pi}(s)$分布，而 $P_{s\pi}(s)$是我们在 MDP 中状态 s 下采取动作 a 的转移概率。这样，第二项就可以看成是 MDP 第一步之后，观察到的折现回报总和的期望。

我们可以用贝尔曼等式求解 V^π，即在有限状态的 MDP($|S|<\infty$)，对于每一个状态 s 都可以得到 $V^\pi(s)$，那么我们就得到了一系列关于 |S| 的线性等式(对于每个状态 s，$V^\pi(s)$未知)。此外，我们定义最优值方程为 $V^*(s) = \max_\pi V^\pi(s)$，即在所有的策略π中，寻找能够使得折现回报总和期望最大的最优策略。把该表达式转换成贝尔曼等式形式，即

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^{\pi}(s')$$

这里的第一项就是之前说的即时回报，而第二项就是在行动 a 之后各个状态的折现回报总和的期望。

我们也定义最优策略π^*：S → A，如下：

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s') V^*(s')$$

注意是$\pi^*(s)$给出了动作 a，而动作 a 使得该式子最大化。

事实上，对于每一个状态 s 和策略π，我们都有 $V^*(s) = V^{\pi^*}(s) \geq V^{\pi}(s)$，这里第一个等式说明对于最优策略$\pi^*$的值等于最优值。而不等式说明，$\pi^*$的值在所有的策略中是最优的。这里需要注意$\pi^*$对于所有的 s 都是最优策略。也就是说，$\pi^*$并不是对于某些状态的最优策略，而是对于所有状态而言的，换句话说无论我们 MDP 的初始状态是什么，都是最优的策略。

16.3. 值迭代和策略迭代

我们讨论两个有效的算法用来求解有限状态下 MDP。接下来，我们讨论的 MDP 是在状态空间和动作空间均是有限空间的情况下，即$|S| < \infty, |A| < \infty$。

第一个方法是值迭代，如下：

1）对于每一个状态 s，初始化 V(s) := 0
2）循环直到收敛 {
对于每一个状态，更新 $V(s) := R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^{\pi}(s')$ }

该算法是根据贝尔曼等式不断的尝试更新值函数，以得到最优值函数。

在内循环中，有两种方法。第一种是，我们可以计算每一个状态 s 的新值 V(s)，然后用这些值更新旧的值(统一更新)。这称之为同步更新(synchronous update)。在这个例子中，算法可以被认为是 Bellman backup operator 的一个应用，即估计当前的值作为新值。此外，我们也可以采用异步更新(asynchronous updates)，也就是在按照某个顺序循环所有状态，然后更新一次值。在同步更新中，第一次更新之后，V(s)=R(s)，而对于异步更新，第一次更新之后大部分 V(s)>R(s)。

无论是同步更新还是异步更新，值迭代都可以由 V 收敛到 V^*，在得到 V^* 之后，我们就可以根据$\pi^*(s)$得到最优的策略。

另外一种求解算法是策略迭代(policy iteration)，算法如下：

1）随机初始化策略π

2）循环直到收敛 {

a) 令 V := V^{π}

b) 对于每一个状态 s，取π(s) := $\max_{a\in A} \gamma\sum_{s'\in S} P_{sa}(s')V^{\pi}(s')$}

这里的内循环不断重复的计算当前策略的值函数，然后使用当前的值函数更新策略(b)中更新的策略π也被称为对于 V 的贪婪策略)。注意步骤 a)可以用求解贝尔曼方程来得到 V^{π}。在足够多的迭代下，V 会收敛到 V^{*}，π会收敛到π^{*}。

以上两种算法是求解 MDP 的标准算法，但并不是说没有更好的方法。对于小的 MDP，策略迭代通常能够更快的收敛。而对于具有较大状态空间的 MDP 而言，求解 V^{*} 是非常困难的(需要求解大量的线性等式)，这时候一般选择值迭代。因此，在实际问题中值迭代通常多余策略迭代。

16.4. MDP 的一个学习模型

到目前为止，我们已经讨论了 MDP 以及在状态转移概率和回报已知的情况下求解 MDP 的算法。在许多现实的例子里，我们并不知道明确的状态转移概率和回报，因此必须从数据中提取(通常 S，A，γ是已知的)。

举个例子，对于倒摆问题(inverted pendulum problem，倒摆是一个棒通过一个连接轴，其底端连接在一辆可推动的小车上，控制的目的是当倒摆出现偏角时，在水平方向上给小车以作用力，通过小车的水平运动，使倒摆保持在垂直的位置。)，我们有多个路径(这里成为转移链)的状态转移路线如下：

$$
\begin{aligned}
&s_0^{(1)} \xrightarrow{a_0^{(1)}} s_1^{(1)} \xrightarrow{a_1^{(1)}} s_2^{(1)} \xrightarrow{a_2^{(1)}} s_3^{(1)} \xrightarrow{a_3^{(1)}} \dots \\
&s_0^{(2)} \xrightarrow{a_0^{(2)}} s_1^{(2)} \xrightarrow{a_1^{(2)}} s_2^{(2)} \xrightarrow{a_2^{(2)}} s_3^{(2)} \xrightarrow{a_3^{(2)}} \dots \\
&\dots
\end{aligned}
$$

这里的 $s^{(j)}{}_i$ 是在时间 i 转移链 j 下的状态，$a^{(j)}{}_i$ 是在那个状态之后采取的动作。实际上，每个转移路径中状态数是有限的，每个转移链要么进入终结状态，要么达到规定的步数就会终结。如果我们获得了很多上面类似的转移链（相当于有了样本），那么我们就可以使用最大似然估计来估计状态转移概率，$P_{sa}(s')$ = A/B，其中 A 是样本中在状态 s 采取动作 a 而到达 s’ 的总次数，B 是在状态 s 下采取行动 a 的总次数。此外，如果 A=B=0，

就表明了在状态 s 下从来没有采取过动作 a，这个时候可以令 $P_{sa}(s') = 1/|S|$ (即转移到各个状态的概率相等)。

注意，如果我们在 MDP 中得到越多的经验(越多的转移链)，那么我们有很有效的方法来更新我们的转移概率。也就是说我们只需要根据更多的观测样本，就可以很容易的更新分子分母，进而更新得到 P_{sa}。相似的，如果 R 是未知的，那么我们可以选择在状态 s 下的即时回报 R(s)的期望为在所有样本观测到的状态 s 下回报的均值。

在此基础之上(使用转移链中估计到的转移概率和回报)，我们可以选择使用值迭代或策略迭代求解 MDP。以值迭代为例，把所有的放在一起，得到一个 MDP 学习算法，整体如下：

1）随机初始化策略π

2）循环{

a) 统计执行了策略π的样本；

b) 根据样本估计 P_{sa} (如果可以，则得到 R)；

c) 根据估计的状态转移概率和回报，采用值迭代得到一个 V 新的估计值；

d) 根据 V 的贪婪策略更新策略π }

我们注意到，对于该算法，有一个优化的方法使得他可以运行的更快。即在算法的内循环中，我们使用了值迭代，如果初始化 V 的时候不为 0，而使用在上一次大循环中得到的 V，那么在一个更好的初始化 V 值的情况下，该算法迭代的速度将会更快。

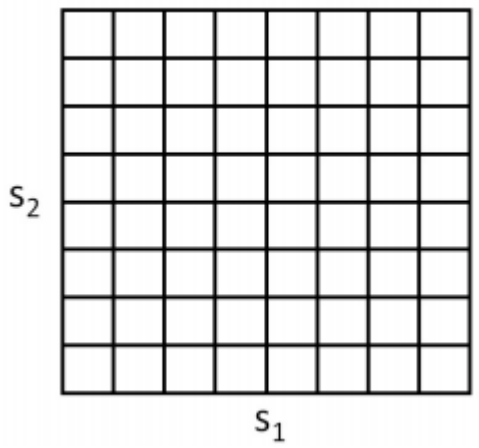
17. 离散与维灾难

17.1. 连续状态下的 MDP

到目前为止，我们主要讲述了在有限状态空间下的 MDP，现在我们讨论一下在可能是无限状态空间下的 MDP。举个例子说，一辆车，我们可以用$(x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})$来表示状态，其中(x,y)表示位置，θ 是方向，在 x、y 方向速度分别为$\dot{x}, \dot{y}$，角速度为$\dot{\theta}$。因此，$S = R^6$，是在无限的状态集合里，因为对于一辆车可能的位置和方向是无限的。相似的，在自主驾驶中，直升飞机在 3 维空间里，状态空间是 9 个维度的，且也是状态集合也是无限的。在这部分，我们讨论在状态空间为 R^n 下的一些求解 MDP 的方法。

求解连续状态 MDP 的最简单的方法也许就是离散化状态空间了（有限元），之后采用之前提到的值迭代或策略迭代的方法求解。

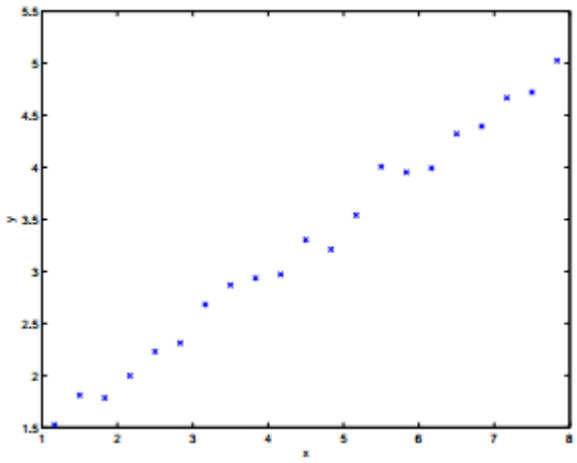
举个例子而言，假如我们有一个二维的状态(s_1,s_2)，我们可以用如下方格离散化状态空间。



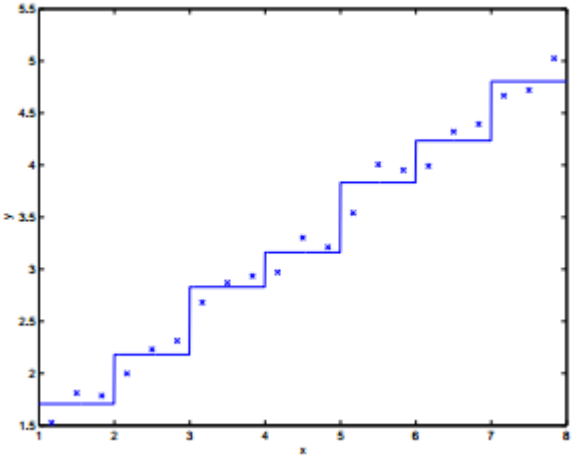
这里每一个网格表示一个离散状态$\bar{s}$，我们可以采用离散状态空间$(\bar{S}, A, \{P_{sa}\}, \gamma, R)$来近似连续的状态空间，这里$\bar{S}$是离散状态空间，$\{P_{sa}\}$是我们的状态转移概率矩阵。我们可以使用值迭代或者策略迭代求解 $V^*(\bar{s})$和 $\pi^*(\bar{s})$。当我们的实际系统状态为某个连续状态 $s \in S$，我们需要选择一个合适的动作，这个时候我们只要计算 s 对应的离散状态$\bar{s}$，就可以得到需要采取的最优动作 $\pi^*(\bar{s})$。

离散化方法可以应用在很多问题中。然而，该方法有两个缺点：第一，他使用了相当朴素代表来表示 V^*(或 π^*)。被逼入说，他假设了对于各个离散化的间隔的值函数是一个常

数(比如，在各个网格中，值函数是分段常数)。为了更好的理解这种情况，考虑在监督式学习下的一个问题，拟合一个如下数据集合：



很明显，线性回归能够很好的解决这个问题。然而，如果我们对于 x 轴采用离散化，之后选用带标点即在每个离散间隔中的分段常数，然后拟合结果如下图：



这里的分段常数代表并不能很好的表示许多平滑的常数。结果并不能很好的平滑这些输入，且不能泛化到其他的不同点。使用这样的代表点法，我们需要一个非常好的有限离散(即采用更小的网格)来取得更好的结果。

另一个问题是这种代表法也被称之为维灾难(curse of dimensionality)。比如 $S = R^n$，我们离散化每一个 n 维的状态到 k 个值。那么整个离散空间将会达到 k^n，状态空间维度指数增长太大而无法扩展到很多大型问题上。比如，我们有一个 10 维的状态空间，吴国

我们离散化每个状态到 100 个值，那么我们将会得到 $100^{10} = 10^{20}$ 维度离散状态，这么大的空间已经无法在一般的电脑上运行了。

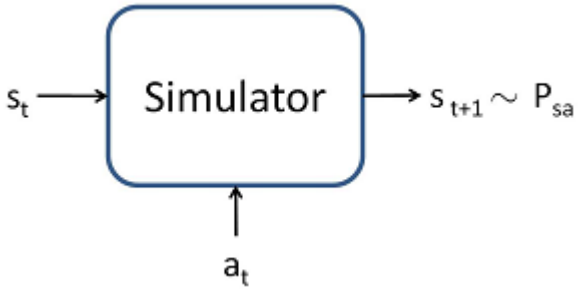
总的来说，在一维或者二维的问题上，离散化方法还是非常有效的方法。有些时候到达 4 维度状态空间时，选择一个合适的离散化方法也能达到较好的结果。如果你足够幸运和聪明的话，即使在 6 维问题上，也可能得到到一个不错的结果。但是，如果再高维度，那么基本是没什么希望了。

18. 线性二次型调节控制

现在我们讨论一个用于连续状态 MDP 的一个寻找最优策略的一个方法。该方法中我们直接近似 V^*，而不采用离散化。该方法称之为值函数近似，在很多实际 RL 问题都有很好的应用。

18.1. 使用一个模型或

为了发展一个值函数近似算法，我们假设对于 MDP，我们有一个模型(model)或仿真器(simulator)。通俗的说，一个仿真器是一个黑盒子，输入是任何一个连续值的状态 s_t，动作 a_t，根据状态转移概率矩阵 P_{stat} 输出到下一个状态 s_{t+1}，如下图：



有很多方法来得到该模型。其一是，使用物理仿真。举个例子来说，对于之前提到的倒摆的一个仿真器，该倒摆是通过一些列物理定律来精确计算小车和木杆的位置和方向，即在时间 t 时，给定一个动作 a，如果我们知道了各个参数(比如木板的长度、质量等等)，我们就能精确的计算到时间 t+1 时候的小车和木杆的位置和方向。或者我们可以使用已有的一些物理仿真包来求解，即输入完整的物理参数，以及当前的状态 s_t 和要采取的动作 a_t，就可以计算系统在下一秒的状态 s_{t+1}。

一个获得该模型的方法是从 MDP 收集的数据中得到。比如说，我们重复试验 MDP 后，得到了 m 条链，每条链有 T 个时间步。这个过程中，我们可以随机的选择动作，或者执行特定的策略或者根据某一方法选择动作。我们得到了 m 个状态序列，如下：

$$
\begin{aligned}
&s_0^{(1)} \xrightarrow{a_0^{(1)}} s_1^{(1)} \xrightarrow{a_1^{(1)}} s_2^{(1)} \xrightarrow{a_2^{(1)}} \cdots \xrightarrow{a_{T-1}^{(1)}} s_T^{(1)} \\
&s_0^{(2)} \xrightarrow{a_0^{(2)}} s_1^{(2)} \xrightarrow{a_1^{(2)}} s_2^{(2)} \xrightarrow{a_2^{(2)}} \cdots \xrightarrow{a_{T-1}^{(2)}} s_T^{(2)} \\
&\cdots \\
&s_0^{(m)} \xrightarrow{a_0^{(m)}} s_1^{(m)} \xrightarrow{a_1^{(m)}} s_2^{(m)} \xrightarrow{a_2^{(m)}} \cdots \xrightarrow{a_{T-1}^{(m)}} s_T^{(m)}
\end{aligned}
$$

之后，我们可以应用一些算法来预测状态 s_{t+1}，其参数为 s_t 和 a_t。比如，我们可以选择一个线性模型：$s_{t+1} = As_t + Ba_t$，使用的该仿真算法类似于线性回归。这里的参数是矩阵 A 和矩阵 B，我们可以使用观测到的 m 条马尔可夫链进行估计：

$$\arg\min_{A,B} \sum_{i=1}^{m} \sum_{t=0}^{T-1} \left\| s_{t+1}^{(i)} - \left(As_t^{(i)} + Ba_t^{(i)} \right) \right\|^2$$

这与最大似然估计参数是一致的。在学得 AB 之后，一个方法是构建一个确定性(deterministic)的模型，即在给定输入 s_t 和 a_t 时，输出的 s_{t+1} 是确定的。即我们总是根据得到的方程来计算下一个状态 s_{t+1}；或者我们可以建立一个随机(deterministic)模型，即 st+1 是不确定的，根据 $s_{t+1} = As_t + Ba_t + \varepsilon_t$ 得到，这里的是 ε_t 噪音项，服从 $N(0,\Sigma)$分布(这里的矩阵 Σ 也可以通过观测到的数据估计得到)。

因此，我们得到了在当前状态和动作下的下一个状态的线性模型，但是非线性模型该怎么办呢？一个方法是通过变换，即我们可以学习一个模型 $s_{t+1} = A\varphi_s(s_t) + B\varphi_a(a_t)$，这里的 φ_s 和 φ_a 都是状态和动作的一些非线性特征映射函数。另一个方法是使用非线性学习算法，比如局部加权线性回归来学习估计状态 s_{t+1}。这些模型都可以用来构建确定性的或不确定性的 MDP 仿真器。

18.2. 拟合值迭代

我们现在描述一个拟合值迭代(fitted value iteration)算法，用于估计连续状态 MDP 的值函数。这里我们将假设一个问题，含有连续状态空间 $S=R^n$，但是动作空间 A 是很小的而且是离散的。回想一下值迭代，我们希望的更新如下：

$$V(s) := R(s) + \gamma \max_a \int_{s'} P_{sa}(s')V(s')ds'$$

$$= R(s) + \gamma \max_a E_{s' \sim Psa}[V(s')]$$

在之前的部分，对于离散状态的 MDP 是求和，而这里是连续状态因此是积分。

拟合值迭代的主要是想是：我们将使用有限样本的状态空间 $s^{(1)}, \ldots, s^{(m)}$，特别的，我们将使用监督式学习算法——如下描述的线性回归，用一个线性或非线性的状态函数来近似值函数。$V(s) = \theta^T\varphi(s)$，这里的φ是状态空间的一些适当的特征映射。

对于我们有限样本的 m 个状态，拟合值迭代将会首先计算一个量 $y^{(i)}$，可以用 $R(s) + \gamma \max_a E_{s' \sim Psa}[V(s')]$来近似，之后采用监督式学习，努力使得 V(s)尽可能的接近 $R(s) + \gamma \max_a E_{s' \sim Psa}[V(s')]$，即接近 $y^{(i)}$。

具体算法如下：

1) 随机样本的 m 个状态 $s^{(1)}, \ldots, s^{(m)} \in S$

2) 初始化 $\theta := 0$

3) 循环 {

对于 $i = 1, \ldots, m$\{

对于每个动作 $a \in A$ {

样本 $s'_1, \ldots, s'_k \sim P_{s(i)a}$(使用一个 MDP 模型)

让 $q(a) = \frac{1}{k}\sum_{j=1}^{k} R(s^{(i)}) + \gamma V(s'_j)$

//这里 q(a)就是 $R(s) + \gamma \max_a E_{s' \sim Psa}[V(s')]$的估计

}

令 $y^{(i)} = \max_a q(a)$ // $y^{(i)}$是 $R(s)+\gamma\max_a E_{s' \sim Psa}[V(s')]$的估计

}

// 在最开始的值迭代算法中，我们根据 $V(s^{(i)}) := y^{(i)}$更新值函数

// 在该算法中，我们希望采用监督式学习，使得 $V(s^{(i)}) \approx y^{(i)}$

令$\theta := \arg\min_\theta \frac{1}{2}\sum_1^m (\theta^T \varphi(s^i) - y^{(i)})^2$

}

在上面，我们使用了线性回归算法的拟合值迭代，来努力使得 $V(s^{(i)})$ 接近 $y^{(i)}$，算法的步骤和标准的监督式学习都是类似的。尽管算法描述使用了线性模型，但是其他的回归算法(如局部加权线性回归)也是可以使用的。

与离散状态空间的值迭代不同，拟合值迭代不能被证明是一定收敛的。然而在实际应用中，他通常都是收敛的或近似收敛，在许多问题中都能有很好的结果。需要注意，如果我们使用 MDP 的确定性的仿真器或模型，那么拟合值迭代算法中可以通过在最开始设置

k=1 来简化算法。因为 $E_{s'\sim Psa}[V(s')]$变成了对确定性分布的期望了，所以一个样本就能够求得期望。此外，上述算法，我们不得不抽取 k 个样本，然后用均值来近似期望值。

最后，拟合值迭代输出 V，是近似于 V^*，这就含蓄的定义了我们的策略。特别的，当我们的系统在状态 s 时，我们需要选择一个动作，我们希望选择一个动作满足：

$$\arg\max_a E_{s'\sim Psa}[V(s')]$$

计算该式子的方法类似于拟合值迭代的内循环，即对于每一个动作，用样本 $s'_1, \ldots, s'_k \sim P_{s(i)a}$ 来近似期望。同样，对于确定性的仿真器，我们可以设置 k=1。

在实际应用中，也会有一些其他的方法来近似这个步骤。比如，常用的一个方法是如果仿真器是 $s_{t+1} = f(s_t, a_t) + \varepsilon_t$ 的形式，这里的 f 是一些确定性的状态函数，比如 $f(s_t, a_t) = As_t + Ba_t$，而 ε 均值为 0 的高斯噪声。在这个例子中，我们可以通过 $\arg\max_a V(f(s, a))$ 来选择动作。换句话说，这里默认了 ε_t=0(即在仿真器中忽略了噪声)，设定 k=1。等同于：

$$E_{s'}[V(s')] \approx V(Es'[s']) = V(f(s, a))$$

这里的期望是基于随机变量 $s' \sim P_{sa}$。只要噪声项 ε_t 是很小的，这通常都是很合理的近似。然而，对于不适用于该近似方法的，即样本空间为 K|A|，如果采用上式进行期望的近似，那么计算量是很大的。