深度思考之机器学习系列

xjtu-zhongyingLi
2018-07-22

目录

> *数学是人类智慧的结晶，统计学习是数学领域最璀璨的明珠，算法可以让这个明珠照亮整个世界!*
>
> *— 李中英*
>
> *世界知名互联网公司高级算法研究猿👍👍👍*

1. 机器学习基本概念

1.1. 统计学习

统计学习是关于计算机基于数据构建概率统计模型并运用模型对数据进行预测与分析的一门学科，也成为统计机器学习。 统计学习的对象是数据，它从数据出发，提取数据的特征，抽象出数据的模型，发现数据中的知识，又回到对数据的分析与预测中去，统计学习关于数据的基本假设是：同类数据具有一定的统计规律性.

统计学习的目的是对数据进行预测与分析，是通过构建概率统计模型实现的，统计学习总的目的就是考虑学习什么样的模型和如何学习模型， 以使模型能对数据进行准确的预测和分析，同时考虑尽可能的提高学习效率.

统计学习的方法包括：监督学习、非监督学习、半监督学习和强化学习，我们重点讨论监督学习.

1.2. 统计学习三要素

1.2.1. 模型

在监督学习中，模型就是指要学习的条件概率分布或决策函数。假设空间中的模型一般有无穷多个，假设空间可定义为：

$$F = \{f|Y = f(X)\}$$

假设空间通常是由参数向量决定的函数簇，其中参数向量取值于n维欧氏空间，称为参数空间。

$$F = \{f|Y = f_\theta(X), \theta \in R^n\}$$

假设空间也可以定义为条件概率的集合:

$$F = \{P|P_\theta(Y|X), \theta \in R^n\}$$

1.2.2. 策略

有了模型的假设空间，统计学习接着要考虑的是按照什么样的准则学习或选择最优的模型，这就是策略。

1.2.2.1. 损失函数

策略用来解决最优模型的选择问题，那么如何评价模型优劣，这就是损失函数或代价函数，下面是一些常见的损失函数：
（1）0-1损失

$$L(Y|f(X)) = \begin{cases} 1, Y \neq f(X) \\ 0, Y = f(X) \end{cases}$$

（2）平方损失

$$L(Y|f(X)) = (Y - f(X))^2$$

（3）绝对损失

$$L(Y|f(X)) = |Y - f(X)|$$

（4）对数损失

$$L(Y|f(X)) = -log(P(Y|X)$$

损失函数越小，模型就越好，理论上的最优模型应该是损失函数的期望值最小，而理论模型是关于联合分布下的平均损失，称为期望损失或风险损失，然而现实的问题是联合分布是未知的，如果已知也就不需要学习了。

所以机器学习采用的方法时是通过用**训练数据集上的平均损失近似期望损失**，训练集上的风险我们称为经验风险，根据大数定理，当训练样本数量趋近于无穷时，经验风险趋近于期望风险。

但是现实中的训练样本数量是有限的，甚至很小，所以直接使用经验风险估计期望风险常常不理想，需要对经验风险进行一定的矫正，这就关系到监督学习的两个基本策略：**经验风险最小化和结构风险最小化**。

1.2.2.2. 经验风险最小化

经验风险最小化的策略认为：经验风险最小化的模型是最优模型，根据这一策略， 按照经验风险最小化策略求最优模型就是求解最优化问题：

$$min\frac{1}{m}\sum_{i=1}^{m}L(y_i,f(x_i))$$

极大似然估计就是经验风险最小化的典型例子，当模型是条件概率，损失函数是对数损失函数时，经验风险最小化就等价于极大似然估计。

但是，当样本量较少时，经验风险最小化学习的效果未必很好，会产生过拟合的问题，结构风险最小化是为了防止过拟合而提出的策略。

1.2.2.3. 结构风险最小化

结构风险最小化等价于正则化，结构风险在经验风险上加上表示模型复杂度的正则化项或惩罚项，结构风险求解的最优化问题是：

$$min\frac{1}{m}\sum_{i=1}^{m}L(y_i,f(x_i))+\lambda J(f)$$

其中$J(f)$为模型复杂度，$\lambda \geqslant 0$是系数，用以权衡经验风险和模型复杂度。结构风险小需要经验风险和模型复杂度同时小，结构风险小的模型往往对未知的测试数据和已知的训练数据都有较好的预测.

比如，贝叶斯估计中的最大后验概率估计就是结构风险最小化的一个例子。**当模型是条件概率分布、损失函数是对数损失函数、模型复杂度由模型的先验概率表示时，结构风险最小化就等价于最大后验概率估计**。

> *经验风险最小化深度理解*
>
> 模型是条件概率分布，优化目标可以表示为概率连乘的形势，如果损失函数是对数损失，即可以写成连加的形势，由于最大后验概率可以写成似然函数和先验概率分布的乘积，对数损失后，先验概率就变成似然函数连加后的一项，对比上面公式，先验概率就刚好等价于模型复杂度，因此这种情况下，结构风险最小化就等价于最大后验概率估计。
>
> - 我们将会在一个独立的小节阐述如何用贝叶斯理论理解本章的概念

1.2.3. 算法

从假设空间选择最优模型后，需要考虑用什么计算方法求解最优模型，如果最优化问题有显式的解析解，这个最优化问题就比较简单，但通常解析解不存在，这就需要用数值计算的方法求解，如何保证找到全局最优解，并使求解的过程高效，是算法最核心的问题。

1.3. 模型评估

统计学习的目的是使学到的模型不仅对已知数据且对未知数据都有很好的预测能力，一般采用训练误差和测试误差作为评价模型的标准，而最终决定模型是否真的不错，是由模型在未知数据上的预测能力决定的。

通常将学习方法对未知数据的预测能力称为泛化能力。因此选择模型时一定是选择泛化能力强的模型，而这个指标很难量化，通常在模型选择时参考奥卡姆剃刀原理，即当两个模型在测试集上具有相近的误差时，倾向于选择更简单的模型。

换句话说，越复杂的模型，其过拟合的风险也就越高，一味追求对训练数据的预测能力，会导致模型的复杂度高于真是模型的复杂度，这种现象就是**过拟合**。

避免模型过拟合的方法有很多，我们重点介绍常用的两种方法：**正则化和交叉验证**。

1.3.1. 正则化

模型选择的典型方法是正则化，正则化是结构最小化策略的实现，是在经验风险上加上一个正则化项或惩罚项。正则化项一般是模型复杂度的单调递增函数，模型越复杂，正则化值越大，正则化一般的形势：

$$min\frac{1}{m}\sum_{i=1}^{m}L(y_i,f(x_i))+\lambda J(f)$$

其中，第一项是经验风险，第二项是正则化项，正则化项可以取不同的形势，常见的有L1和L2正则化.

正则化的作用是选择经验风险和模型复杂度同时较小的模型，正则化符合奥卡姆剃刀原理：在所有可能选择的模型中，能够很好的解释已知数据并且十分简单才是最好的模型。**从贝叶斯的角度来看，正则化对应于模型的先验概率，可以假设复杂的模型具有较小的先验概率，简单的模型有较大的先验概率。**

> *正则化项深入理解*
>
> 如果正则化项等价于模型的复杂度，那么复杂模型的正则化项应该较大，如果正则化想对应于模型的先验概率，那么复杂模型应该具有较大的先验概率才对？ 其实，最大后验概率是一个max问题，而最优化问题是一个min问题，在最大后验概率转换为最优化问题时，需要给优化项取负值。

1.3.2. 交叉验证

经验告诉我们，交叉验证非常非常重要，任何模型都会多少有一些超参数需要调（即调参），不同的超参数对应了不同的模型，如果选择超参数和对应的模型呢？让不同超参数对应的模型都在同一份验证集上评估，选择性能最优的模型。

理想条件下，当样本数据充足时，一般将数据分成：训练集、验证集和测试集，即模型的学习有完全独立的验证数据，训练集负责模型的训练，验证集负责模型选择，而测试集负责模型最终的评估。

实际情况下，样本的数据量往往较少，此时常用的交叉验证有：简单交叉验证-将数据分成7:3的训练集和测试集，测试集负责验证和模型选择；K折交叉验证-将数据随机分成K份，选取其中一份作为测试集，其余K-1份训练，将这一过程进行K次选择重复进行，最后选出K次评估中平均测试误差最小的模型；留一法-当数据严重缺乏时使用，实际应用很少。

1.3.3. 泛化能力

泛化能力是模型最本质的要求，也机器学习中最核心的概念。经验风险最小化的角度考虑，训练误差小的模型，其泛化误差也会小，应用Hoeffding不等式证明泛化误差的上界。

1.3.4. 生成模型和判别模型

1.3.4.1. 判别模型

以二分类问题为例，在解空间中寻找一条直线把两种类别的样本分开，对于新的样本只需判断在直线的哪一侧即可，这种直接对问题求解的的方法称为判别模型。

1.3.4.2. 生成模型

生成模型会首先对两类样本分别进行建模，用新的样本去分别匹配两个模型，匹配度高的作为新样本的类别。

形式化地说，判别模型是直接对进行建模或者直接学习输入空间到输出空间的映射关系，而生成模型是对条件概率和先验概率进行建模，然后按照贝叶斯公式求出后验概率。使得后验概率最大的类别就是新样本的预测值。

$$p(y|x) = \frac{p(y) \cdot p(x|y)}{p(x)}$$

$$\underset{y}{argmax}\, p(y|x) = \underset{y}{argmax}\, p(x|y) \cdot p(y)$$

示例1. 贝叶斯学派下的生成模型是如何对未知样本进行预测

假设仍然是个二分类问题，问题是预测一个人是男人还是女人，为了简单起见，假设特征只有一个：是否有胡子。

思路肯定是分别求解新样本是男人和女人的概率，取概率最大的类别作为预测结果，假定新样本为"有胡子"：

`p(y=male|x=beard) = p(x=beard|y=male)✖p(y=male)` 1

`p(y=female|x=beard) = p(x=beard|y=female)✖p(y=female)` 2

关键在于如何求解上面两个概率值，其实上面公式的概率值全部都是统计值，也就是训练样本中，根据条件统计出来的概率。比如第一个公式，是男人的概率就是样本中男人的占比，而条件概率（似然函数）就是男人中有胡子的概率。

你可能会问特征参数去哪了？求特征参数的方法是判别模型，而生成模型不需要!

1.4. 深度思考

1.4.1. 贝叶斯理论

第一章的难点主要集中在如何理解下面几句话：

当模型是条件概率分布、损失函数是对数损失函数、模型复杂度由模型的先验概率表示时，结构风险最小化就等价于最大后验概率估计 1

从贝叶斯的角度来看，正则化对应于模型的先验概率，可以假设复杂的模型具有较小的先验概率，简单的模型有较大的先验概率 2

1 当模型是条件概率分布时，例如逻辑回归，此时的参数估计常用最大似然估计，而很容易想到，损失函数应该取似然函数的负数。

$$\theta_{ML} = \underset{\theta}{argmax} \prod_{i=1}^{m} p(y^i \mid x^i; \theta) \quad (1)$$

$$L(\theta) = \underset{\theta}{argmax} \sum_{i=1}^{m} log\, p(y^i \mid x^i; \theta) \quad (2)$$

$$loss(\theta) = \underset{\theta}{argmin}(-\sum_{i=1}^{m} log\, p(y^i \mid x^i; \theta)) + L_n(\theta) \quad (3)$$

上面公式(1)为最大似然估计，公式(2)为对数似然函数，公式(3)是损失函数。

再来看下贝叶斯参数估计策略，贝叶斯方法的参数估计，就是通过最大化后验概率来估计模型的参数的。假定模型参数为w,数据集为D，则：

$$w = \underset{w}{argmax} p(w \mid D) = \underset{w}{argmax} \frac{p(D \mid w) \cdot p(w)}{p(D)} \propto p(D \mid w) \cdot p(w)$$

假定，样本独立不相关且模型参数独立不相关， 则

$$\begin{aligned} p(w)p(D \mid w) &= \prod_{i=1}^{m} p(D_i \mid w) \prod_{j=1}^{n} p(w_j) \\ &\propto \sum_{i=1}^{m} p(D_i \mid w) + \sum_{j=1}^{n} p(w_j) \qquad (4) \end{aligned}$$

对比公式(4)和公式(3)，公式(3)为损失函数，因此越小越好；公式(4)为后验概率，因此越大越好，将公式(4)取负号，其实就可以对应到损失函数，再看看描述， 模型是条件概率满足，损失函数为对数损失满足，如果模型复杂度由模型的先验概率表示，这句话可以对照公式(4)，其中先验概率就是$p(w)$累加项。那么这种情况下， 结构风险最小化(就是加了正则化项的最大似然估计)就等价于最大后验概率(就是似然函数加先验概率)，没毛病，刚好对上。

第二个问题：可以直接使用上一个题的结论，既然要比较，可以将最大后验概率取负，让两种思考方式都变成最小化问题，那么先验概率取负号对应模型的复杂度， 也就意味着复杂模型(值大)具有较小的先验概率(取负后值大)，反之亦然，证毕。

2. 线性回归与逻辑回归

2.1. 线性回归

2.1.1. 概念

线形回归主要研究一个变量(y)关于另一些变量(X)的具体依赖关系的。例如，房价问题，假设我们只考虑房屋的面积和卧室的个数，问题就可以具体解释为：我们希望得到这样一个线形模型，使得给它一个房子的面积和卧室个数，它就可以准确的评估出房价。

假设房价为y，房屋面积为x_1，卧室个数为x_2，那么线形模型就可以表示为：

$$y = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 = \sum_{i=0}^{i=2} \theta_i x_i \qquad (2.1)$$

其中θ为要学习的参数，这里$x_0 = 1$，对于一般问题，公式通常写成：

$$h_\theta(x) = \sum_{i=0}^{m} \theta^i x^i = \Theta^T X \qquad (2.2)$$

我们用x^i表示第i个特征，y^i表示x^i对应的房价，从而定义损失函数$J(\theta)$，一个很自然的想法就是用所有样本预测值和真实值的平方差和作为损失函数：

$$J(\theta) = \frac{1}{2} \sum_{i=0}^{m} (h_\theta(x^i) - y^i)^2 \qquad (2.3)$$

我们的目标就是找出使损失函数最小的参数值，就得到了拟合训练集的最佳参数，至于为什么使用该函数会取得很好的效果，后面会有解释，这种通过最小化误差的平方和来求解最佳参数的方法又称为 最小二乘法或最小平方误差 。

求一个函数的最小值，最简单的方法是对函数求导，令导数为0，直接解出参数，这种方法叫 解正规方程组 ，方法简单直接，却有很多限制，比如未知数（即参数）个数大于方程组个数（即训练样本）时，无法求解，该方法我们将在后面单独讲述。

2.1.2. 梯度下降

这里我们重点介绍 梯度下降（Gradient Descent） 来求参数，即通过不断调整参数的值，使得损失函数值不断减小，迭代收敛至满足损失误差允许的范围，一个直观的更新规则为：

$$\theta_j = \theta_j + \delta \qquad (2.4)$$

关键就在于每次迭代如何求解δ，确定δ是什么的过程不是一个数学上严格的推断（即没有标准答案），而是一种猜想（也正是因为没有标准答案所以更新参数的方法除了梯度下降，还有牛顿法、拟牛顿法等），我们要做的就是如何让猜想尽可能的合理。

首先介绍梯度的概念，我们中学学过的导数即梯度，反映的是 参数往正向的变化量趋向于0时的其函数值变化量的极限 ，比如梯度为5，表示参数往梯度方向增加一点点时，其函数值也增加一点点，但不一定是5；再比如梯度为-5，参数往梯度方向增加一点点的时候，其函数值会将少一点点。而我们的目的是让函数值不断减小，因此不管梯度是正还是负，我们只要将参数往梯度相反的方向增加一点点，函数值就会减小，则更新规则就可以变成:

$$\theta_j = \theta_j + \delta = \theta_j - \frac{\partial}{\partial \theta_j} J(\theta) \qquad (2.5)$$

此时的δ不再是未知数，而是损失函数的负梯度。至此，我们仅确定了没步迭代更新的方向，而每次更新多少合适呢？这个值确实很难给出，不妨引入步长因子α作为超参数，那么最终的更新规则就变为:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{2.6}$$

示例2. 关于梯度下降

> 关于如何理解更新规则的最终形势，大家经常讲到的一个故事场景：想想一下你站在山腰（任意初始位置）上，目标是走到山谷（最小值），你将如何走才能尽快的到达山谷？沿着最陡峭的方向迈一大步！这个最陡峭的方向就是副梯度方向，而你迈得一大步就是步长，一直这样走下去，你会走到谷底。

言归正传，根据公式(2.6)发现：问题的关键就在于如何求解损失函数即公式(2.3)的梯度，下面给出求解过程（假定只有一个训练样本）：

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\ &= (h_\theta(x) - y) \frac{\partial}{\partial \theta_j} h_\theta(x) \\ &= x_j (h_\theta(x) - y) \end{aligned} \tag{2.7}$$

将公式(2.7)带入公式(2.4)得到：

$$\theta_j = \theta_j - x_j (h_\theta(x) - y) \tag{2.8}$$

公式(2.7)是针对只有一个训练样本的情况，考虑到所有 m 个样本时，更新规则就变为：

$$\theta_j = \theta_j - x_j^i \sum_{i=0}^{m} (h_\theta(x^i) - y^i) \tag{2.9}$$

运用这种规则直到收敛，就是批梯度下降算法 BGD（Batch Gradient Descent），判断收敛的方法主要包括：一是,两次迭代后参数的变化量；二是,两次迭代后损失函数的变化量。
规则中的α为步长因子，又称为学习率，需要在实践中调整，过小会导致算法收敛的很慢，过大会导致算法很容易越过最优点，或在最优点附近震荡。

2.1.3. 梯度下降的局限性

梯度下降算法会导致 局部极小值 的产生，可以通过随机初始化，寻找多个最优点来解决该问题，在所有最优点中选择最小的作为最终的最终结果。对于本例中的线形回归问题，不会存在局部极值的问题，因为本问题的随时函数是凸二次函数。

根据公式(2.9)的更新规则，每次迭代更新都需要遍历所有样本，当样本量很大时算法运行速度就会变成龟速。

一种比较好的解决方案就是每次更新时我们只选用一个样本，由于一个样本的梯度方向是随机的，不是全局梯度方向，因此该方法又称为随机梯度下降 SGD（Stochastic Gradient Descent）； BGD 每次使用的样本过多， SGD 每次使用的样本又过少，每次更新时我们还可以随机选择一小批样本进行更新，这种方法叫做小批量梯度下降 Mini-Batch Gradient Descent(MBGD)。

示例3. 扩展问题

- 线形回归的损失函数为什么要选用最小二乘损失（L2损失），如何概率解释？
- 如何解决模型在训练集上过拟合问题？
- 如何从贝叶斯角度深入理解正则化,为什么说L1正则化等价于参数的先验概率分布满足拉普拉斯分布？L2正则化等价于参数的先验概率分布满足高斯分布？

2.1.4. 深度思考

1、先来看下第一个问题： 最小二乘法的概率解释 ，即为什么选择平方函数作为目标函数会使得效果比较好(并非一定是)!

- 假设1:对每一个样本(x^i, y^i)，模型预测结果和真实结果的关系可以写成：

$$y^i = \theta^T x^i + \epsilon^i \qquad (1)$$

其中，ϵ^i表示模型的误差。

- 假设2:误差ϵ^i服从正态分布，即

$$\epsilon \sim N(0, \sigma^2) \qquad (2)$$

假设1只是一种表示形式，那么假设二为何会成立呢？这是因为影响误差的因素有很多，这些因素都是随机分布，根据中心极限定理，即 许多随机变量的和趋向于正态分布 ，我们可以得到假设二，那么

$$p(\epsilon^i) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(\epsilon^i)^2}{2\sigma^2}) \qquad (3)$$

这也表示，当给定参数θ和x时，目标值y也服从正态分布.

$$p(y^{(i)} \mid x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(\theta^T x^{(i)} - y^{(i)})^2}{2\sigma^2}) \qquad (4)$$

- 假设3:对于各个样本的误差ϵ^i是独立同分布的随机变量，这样，我们可以得到似然函数

$$\begin{aligned} l(\theta) &= P(Y \mid X; \theta) \\ &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(\theta^T x^{(i)} - y^{(i)})^2}{2\sigma^2}) \end{aligned} \qquad (5)$$

似然函数什么意义呢？表示是在参数θ下，数据集出现的概率，似然函数同概率的概念很相似，不同之处在于似然函数把θ作为变量，找到使得数据集出现的概率最大时的参数，就是最大似然估计。

对于公式(5)的右端取对数，我们发现可以将似然函数最大化的问题转化为使得平方和最小的问题：

$$\begin{aligned} L(\theta) &= log\, l(\theta) = log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(\theta^T x^{(i)} - y^{(i)})^2}{2\sigma^2}) \\ &= -m log\sqrt{2\pi}\sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (\theta^T x^i - y^i)^2 \\ \Rightarrow \qquad & max(L(\theta)) \propto min(\frac{1}{2} \sum_{i=1}^{m} (\theta^T x^i - y^i)^2) \end{aligned}$$

2、 再来看下第二个问题，其实可以简单解释为：过拟合是经验风险最小化策略带来泛化能力较差的问题，解决方法就常用的就是加正则化项，即

- 采用结构风险最小化的模型选择策略
- 除此之外，结合具体业务，可能还需要把精力放在训练样本的选择上，比如所选样本在某个特征上与真实数据的分布严重不一致，也可能会造成过拟合的问题
- 交叉验证可以解决过拟合问题，这也是一种非常重要的模型选择策略

- 具体到不同类型模型，还有提前终止、dropout、调整网络结构等.

3、第三个问题可以参考第一章的思考题，这里我们可以简单给出结果形式

- 首先什么是拉普拉斯分布，当先验概率满足拉普拉斯分布就是:

$$p(w_i) = N(w_i \mid \mu, b) = \frac{1}{2b} e^{-\frac{|w_i-\mu|}{b}}$$

损失函数取后验概率的负值，可以得到

$$\begin{aligned} w^* &= \underset{w}{argmin} - log \sum_{i=1}^{m} p(D_i \mid w) - log\, p(w) \\ &= \underset{w}{argmin} - log \sum_{i=1}^{m} p(D_i \mid w) - \sum_{j=1}^{n} log\, p(w_j) \\ &= \underset{w}{argmin} - log \sum_{i=1}^{m} p(D_i \mid w) + \sum_{j=1}^{n} \frac{1}{b} |w_j - \mu| \\ &= \underset{w}{argmin} - log \sum_{i=1}^{m} p(D_i \mid w) + \lambda \sum_{j=1}^{n} |w_j| \qquad (\mu = 0, b = \frac{1}{\lambda}) \end{aligned}$$

我们看到目标函数的最后一样刚好就是 `L1` 正则化项，我们从假设参数先验分布为拉普拉斯分布，直接推出了 `L1` 正则化项，所以 `L1` 正则化等价于参数的先验分布满足拉普拉斯分布。 同理，让我么看下参数先验分布为高斯分布的情况

- 当参数的先验概率满足高斯分布时,相当于:

$$p(w_i) = N(w_i \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(w_i - \mu)^2}{2\sigma^2})$$

所以，最优参数为

$$\begin{aligned} w^* &= \underset{w}{argmin} - log \sum_{i=1}^{m} p(D_i \mid w) - log\, p(w) \\ &= \underset{w}{argmin} - log \sum_{i=1}^{m} p(D_i \mid w) - \sum_{j=1}^{n} log\, p(w_j) \\ &= \underset{w}{argmin} - log \sum_{i=1}^{m} p(D_i \mid w) + \sum_{j=1}^{n} \frac{1}{\sigma^2} (w_j - \mu)^2 \\ &= \underset{w}{argmin} - log \sum_{i=1}^{m} p(D_i \mid w) + \lambda \sum_{j=1}^{n} (w_j)^2 \qquad (\mu = 0, \sigma = \sqrt{\frac{1}{\lambda}}) \end{aligned}$$

对比下式

$$w^* = \underset{w}{argmin} \sum_{i} L(y_i, f(x_i; w)) + \lambda \Omega(w)$$

可以看出，似然函数部分对应损失函数(经验风险),而先验概率对应于正则化项，因此 `L2` 正则化等价于模型参数的先验概率满足正态分布。

2.2. 逻辑回归

2.2.1. 揭开面纱

这一节将介绍在工业界应用最广，又最简单、最容易理解的神器 逻辑回归(Logistic Regression)，问题的背景：假设我们需要解决一个二分类问题，比如给你一辆ofo小黄车，判定该车是好车还是坏车。

首先，我们期望学到一个什么样的模型？最简单的需求就是把一批车辆的数据丢给模型，模型返回 0 或者 1 表示好和坏；更近一步，我们希望模型能返回给我们一个 0~1 之间的概率值，这样我们可以根据模型输出的概率值选择坏的概率很大的丢给师傅去修理，选择好的概率很大的推荐给用户骑行。

那么，首先明确了学习模型的输入和输出，输入是车辆的各种维度的信息（比如，昨天发生了几个正常单、报修单、报修部位、报修后是否又有正常骑行等等），输出是一个 0～1 之间的概率值。问题就变成了如何将输入转为输出。

2.2.2. Sigmoid函数

输入空间显然是实数空间，输出空间为 0～1 ，而 sigmoid 刚好就是可以完成这种归一化的函数：

$$h(z) = \frac{1}{1+e^{-z}} \tag{2.10}$$

而我们的输入可以表示为多个特征加权和的形式：

$$z = \theta^T x \tag{2.11}$$

有了这个映射函数，对于一个样例，我们就可以得到它分类的概率值：

$$\begin{aligned} p(y=1|x;\theta) &= h_\theta(x) \\ p(y=0|x;\theta) &= 1-h_\theta(x) \end{aligned}$$

将上面两个公式联合起来可以写成：

$$p(y|x;\theta) = [h_\theta(x)]^y[1-h_\theta(x)]^{1-y} \tag{2.12}$$

这样我们就可以得到在整个数据集上的似然函数:

$$\begin{aligned} l(\theta) &= p(Y|X;\theta) \\ &= \prod_{i=1}^{m} p(y^i;x^i;\theta) \\ &= \prod_{i=1}^{m} [h_\theta(x^i)]^{y^i}[(1-h_\theta(x^i))]^{1-y^i} \end{aligned} \tag{2.13}$$

对似然函数取对数，可以得到：

$$L(\theta) = log(l(\theta)) = \sum_{i=1}^{m}[y^i log h_\theta(x^i) + (1-y^i)log(1-h_\theta(x^i))] \tag{2.14}$$

为了简化其间，我们先只考虑一个样本的情况，则：

$$L(\theta) = y log h_\theta(x) + (1-y)log(1-h_\theta(x)) \tag{2.15}$$

2.2.3. 参数更新

公式(2.15)是二分类问题的最大似然函数，那么我们应该怎样定义损失函数，然后应用梯度下降更新参数呢？首先，损失函数作为优化目标时，其函数值越小越好；而似然函数则刚好相反，其越大越好。所以一个很自然的想法就是取似然函数的负函数作为损失函数：

$$J(\theta) = -L(\theta) = -ylogh_\theta(x) - (1-y)log(1-h_\theta(x)) \tag{2.16}$$

对公式(2.16)应用梯度下降算法，更新规则为:

$$\theta_j = \theta_j - \frac{\partial}{\partial\theta_j}J(\theta) \tag{2.17}$$

求公式(2.16)的导数，公式中有一个复合函数$h_\theta(x)$，我们可以先求解它的导数，求解过程如下:

$$\begin{aligned}\frac{\partial}{\partial h_\theta(x_j)} &= (\frac{1}{1+e^{-\theta^T x}})' \\ &= [(1+e^{-\theta^T x})^{-1}]' \\ &= (-1)(1+e^{-\theta^T x})^{-2}e^{-\theta^T x}(-x^j) \\ &= x_j \cdot \frac{e^{-\theta^T}x}{(1+e^{-\theta^T x})^2} \\ &= x_j \cdot \frac{1+e^{-\theta^T x}-1}{(1+e^{-\theta^T x})^2} \\ &= x_j \cdot [(h_\theta(x))^2 - h_\theta(x)] \\ &= x_j \cdot h_\theta(x)(1-h_\theta(x)) \end{aligned} \tag{2.18}$$

实际上， `sigmoid` 函数 `y=h(x)` 的导数就等于 `y(1-y)` 。
然后求解损失函数$J(\theta)$的导数，利用公式(2.18)的结论，求解过程如下:

$$\begin{aligned}\frac{\partial}{\partial\theta_j}J(\theta) &= -[ylogh_\theta(x) + (1-y)log(1-h_\theta(x))]' \\ &= -[\frac{y}{h_\theta(x)}h'_\theta(x) + \frac{1-y}{1-h_\theta(x)}(-h'_\theta(x)] \\ &= -x_j[\frac{y}{h_\theta(x)}h_\theta(x)(1-h_\theta(x)) - \frac{1-y}{1-h_\theta(x)}h_\theta(x)(1-h_\theta(x))] \\ &= -x_j[y(1-h_\theta(x)) - (1-y)h_\theta(x)] \\ &= -x_j[y - yh_\theta(x) + yh_\theta(x) - h_\theta(x)] \\ &= x_j(h_\theta(x) - y) \end{aligned} \tag{2.19}$$

导入公式(2.17),得到参数更新的规则为:

$$\theta_j = \theta_j - \alpha \cdot (h_\theta(x) - y) \cdot x_j \tag{2.20}$$

考虑多个样本的时候，规则就应该变成:

$$\theta_j = \theta_j - \alpha \cdot \sum_{i=1}^{m}(h_\theta(x^i) - y^i) \cdot x_j^i \tag{2.21}$$

示例 4. 扩展问题

- 为什么二分类问题的模型会叫 `逻辑` *回归* ？
- 损失函数不选用公式(2.16)的形式会怎样？比如，仍然采用最小二乘法。
- 转换函数还有别的选择吗？为什么要选择 `sigmoid` 函数？
- 回归问题通过一个非线性变换就变成了分类问题，从拟合数据转变为拟合决策边界，这是什么原因导致的？
- 如果损失函数就是 `L2` 损失，有什么办法可以求解？

- 最大似然估计和最小化损失函数、最大后验概率等是什么关系?
- 如何用逻辑回归解决多分类问题?

2.2.4. 深度思考

1、 先来看下第一题，为什么叫 逻辑回归 ？

- 简单点回答将 logistic 直译为中文就是了，其实这个模型的准确名字应该叫 对数几率模型 ，因为这个模型(sigmoid函数)可以由正负样本可能性比值的对数推出。

$$\begin{aligned} f(x;\theta) &= ln\frac{p(y=1 \mid x)}{p(y=0 \mid x)} \\ &= ln\frac{p(y=1 \mid x)}{1-p(y=1 \mid x)} \\ &= \theta^T x \end{aligned}$$

由上面公式可以推出

$$\begin{aligned} \frac{p(y=1 \mid x)}{1-p(y=1 \mid x)} &= e^{\theta^T x} \\ p(y=1 \mid x) &= (1-p(y=1 \mid x))e^{\theta^T x} \\ (1+e^{\theta^T x})p(y=1 \mid x) &= e^{\theta^T x} \\ p(y=1 \mid x) &= \frac{e^{\theta^T x}}{1+e^{\theta^T x}} \\ &= \frac{1}{1+e^{-\theta^T x}} \end{aligned}$$

2、逻辑回归的损失函数如果选择均方差损失会怎样呢？

- 均方差损失的函数形式为

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

在逻辑回归中我们的$h_\theta(x)$形式为:

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

代入损失函数，我们可以得到如下图的损失函数。

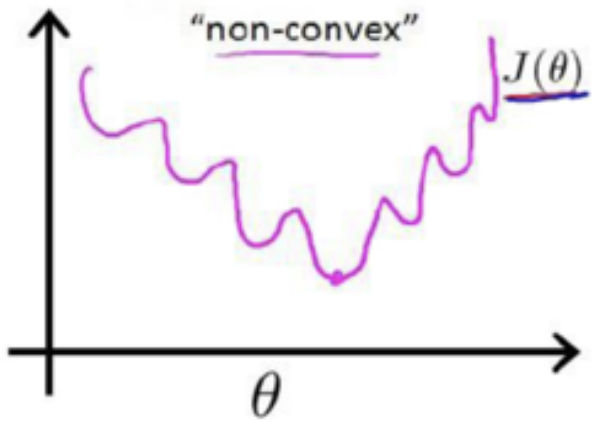


图 1. 具有众多局部极小值的损失函数

这样的损失函数，我们使用梯度下降，会很容易收敛到局部最小值，而不是全局最小值。

3、为什么要选择 sigmoid 函数作为损失函数？ 关于这个问题，你会在网上搜到很多解释和答案，但大部分都在说 sigmoid 函数的数学特性有多好，可以给出预测结果的概率解释等，其实都是在说为什么可以用而没有解释为什么必须用的问题。

- 说到底，源于指数分布簇所特有的最佳性质-最大熵理论，即指数分布簇的最大熵等价于其指数形式的最大似然界
- 二项式的最大熵解等价于二项式指数形式(sigmoid)的最大似然，多项式分布的最大熵等价于多项式分布指数形式(softmax)的最大似然
- 因此为什么要用 sigmoid，那是指数分布簇最大熵特性的必然性，这部分的解释，我们将在第7章 EM 模型给出。

4、这道题很简单，人工智能也好，机器学习也好，原理都是人让它干啥，它就干啥，而决策边界其实就是模型，它是什么完全是由人定义的损失函数决定的。

5、这个问题非常有深度，在这里谈谈我自己的一些想法，欢迎多给宝贵想法和建议。

- 首先，目前讲到的优化算法都是基于梯度下降，朝损失函数变小的方向，不断更新参数，更新策略也是基于步长因子和负梯度综合考虑；
- 算法收敛的依据无非是损失或者参数是否基本不变等，可是现在的问题是有很多极小值，就像你下坡一样，山上有很多山谷，你并不知道哪个是最低的山谷；
- 文章正文在分析梯度下降缺点的时候，给了一些参考的解决方案，如选择多个初始位置，找到多个极小值，取最小的那个，这种方式都是比较传统的方法，也有自身的问题，比如训练代价变得更高，应该初始化几次合适等等；
- 最近在思考有没有一种模型可以直接解决这个问题，想到了 残差，这个概念会在决策树和集成学习那个章节讲到，这里既然提到这个问题，可以大致说下我的想法;
- 在梯度提升算法中，大家最常见的应该就是GBDT，其实这个集成学习的基模型不一定非要是树模型，也可以是线形模型，比如逻辑回归;
- 除了GBDT还有一个类似的模型叫GBR，叫梯度提升回归模型，这里先说一个后面会讲到的结论，当损失函数为平方差损失时，负梯度其实就等于 残差,这个很容易证明，对平方差函数求导就可以得出结论；
- 那一种解决方案是不是可以像 AdaBoost 那样，学习多个逻辑回归模型，后面的模型学习是基于之前模型的 残差 为优化目标(最小化)!

6、这个问题其实没什么技术含量，主要是大家在学习的过程中会经常遇到这些类似的概念，单独拿出来作为思考题，主要是为了加深对这些概念的理解，做到所学知识的融会贯通.

- 首先，我们先说下 最小化损失函数,我们在第一章讲过，模型选择的关键依据就是选择损失函数最小的模型，表现出来的就是经验风险最小化的思想，任何其他最大化**的问题，只要取负值，都可以很容易的将优化目标转化为损失函数;
- 其次，最大似然估计是一种比较传统的参数学习方法,和最小化损失函数一样，都是为了求解最优参数的值，将似然函数取负就可以作为损失函数，然后就变成了最小化损失函数的问题，一般的似然函数都是对数似然，对应的损失函数就是log损失，其实是在做同一件事;
- 最后，最大后验概率估计是贝叶斯思想的一种体现，对应的模型都属于概率模型，因此可以用贝叶斯公式将优化问题转换为似然函数和先验概率的乘积，因此最大后验概率估计其实就是最大似然估计加上正则化项(对应先验概率),当然将后验概率取负又可以转为损失函数的最小化问题;
- 由此可见，上述几个概念除了通用和特例之间的差异外，都是基于最优化问题求解参数的最优解，没有太大本质的差异。

7、逻辑回归是典型的二分类问题，如果将它应用于多分类问题呢？ 这里直接给出吴恩达老师的答案。

- 将多个类型看作两个类型，比如类型A和非A，学习出类型A的逻辑回归模型;
- 然后在对非A的类型执行上述操作即可。

3. 特征工程

3.1. 引子

首先来看一张关于梯度下降的示意图：

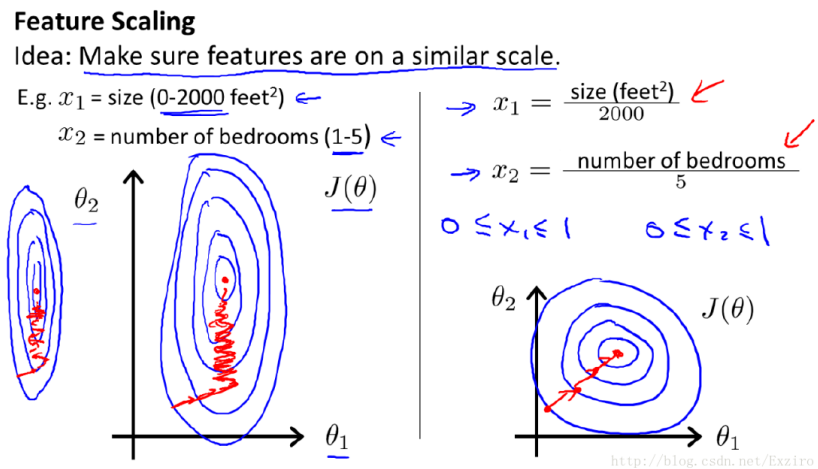


图 2. 梯度下降

上图给出了两个损失函数的等高线，左图是一个很扁的等高线，右图是一个很规整的等高线，哪一个损失函数好？

一个很直观的结论是：当在左图进行梯度下降时，如果我们选择的初始位置为长轴附近，那么需要很多次的迭代才能到达最优点，现象就是算法收敛的很慢，模型学不动。由于模型学习时初始位置都是随机选取的，因此我们更倾向于选择第二种损失函数。

怎样才能保障我们的损失函数像右图那样规整呢？这就是特征工程的一个典型作用，当然特征工程的作用还有很多，以至于它是非常非常重要！

3.2. 特征工程

下图是一个机器学习任务典型的工作流程：

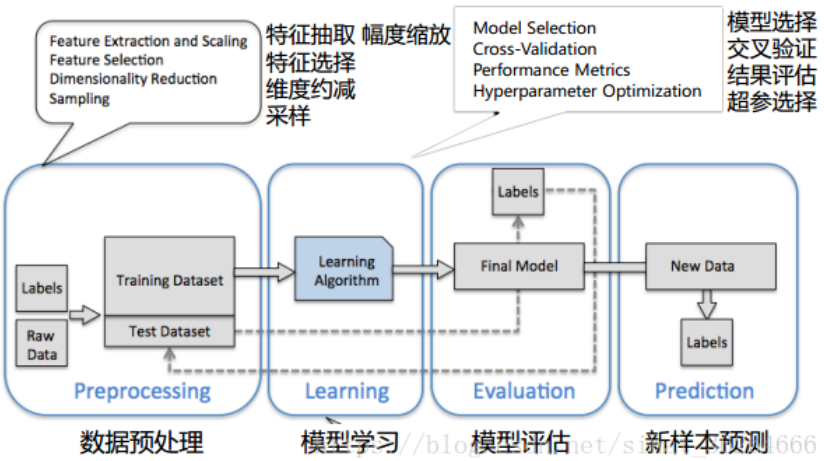


图 3. 工作流程

这个 `pipline` 包括了数据预处理、模型学习、模型评估和新样本预测，每个过程的时间和收益比是按照流程越来越低的，也就是我们把更多的时间花在数据预处理上，获得的收益远大于模型算法的选择；换言之，如果数据处理有问题，后面的环节再完美，也无法得到不错的模型。

工业界，大概有70%的时间都花在了数据预处理上，当然数据预处理中又包含了数据清洗、特征工程等。一般，算法工程师拿到的数据是经过数据挖掘或数据分析师们清洗后的数据，因此更重要的工作是进行特征工程！

不过有的时候，算法工程师们也需要从头做起，数据清洗往往是第一步，我们就从数据清洗开始讲述特征工程。

3.2.1. 数据清洗

`Garbage in , garbage out !` 当你给模型丢进一堆错误数据时，很显然你得到的也肯定是毫无意义的结果;

算法大多数情况下就是个加工厂，至于最后的产品(输出)如何，取决于原材料的好坏:

这个过程会花掉你一般的时间，当然会促进你对业务的理解;

数据清洗要做的事情就是去掉脏数据！

比如，一个人的身高为3米... 缺省值太多的样本丢弃... 数据间存在相互矛盾...

3.2.2. 数据采样

很多情况下数据样本是不均衡的，如小黄车中好车和坏车的比例

往往好车的数量要远远大于坏车的数量，这个时候该如何采样?

如果坏车的数量也很多，那就下采样；如果坏车的数量很有限，多采集、过采样(旋转等)、修改损失函数

3.2.3. 特征处理

3.2.3.1. 数值型特征

- 幅度调整/归一化 如，各种 `scalar`
- `log` 等变换
- 换成统计值，如 `max`、`min`、`mean`、`std`
- 离散化、 `Hash` 分桶
- 尝试转为类别型特征

3.2.3.2. 类别型特征

- `One-hot` 编码, [0 0 0 1 0]
- 哑变量，虚拟变量，如没有填写性别的用户由于数据量较大，给他们一个类别标识
- `Hash` 与聚类处理，如海量数据推荐，不会直接比较每个数据的相似性，往往会先进行聚类
- 尝试转为数值型

3.2.3.3. 时间型特征

- 看作连续值，如转为持续时间或间隔时间
- 看作离散值，如一天中的哪个时段、一周中的星期几、一年中的哪个月、是不是周末、是不是假期等

3.2.3.4. 文本型特征

- 词袋，去掉停用词后，在词库中的映射稀疏向量
- `N-gram`
- 使用 `TF-IDF` 统计
  TF(t) = (词t在当前文中出现次数) / (t在全部文档中出现次数)
  IDF(t) = ln(总文档数/ 含t的文档数)

3.2.3.5. 统计特征

- 多维度统计特征，具备良好特征的潜质

3.2.3.6. 组合特征

- 一般用树模型进行组合特征的筛选，一条组合路径就是一个组合特征

4. 支持向量机

4.1. 开门见山

首先，支持向量机是一个二分类问题，说到二分类大家一定会想到我们之前提过的逻辑回归，其实二分类的方法还有很多，机器学习的目的就是为了找到一个决策边界， 使得样本数据可以很好地按照其类别被决策边界划分到两侧，来看个二维空间的分类图示:

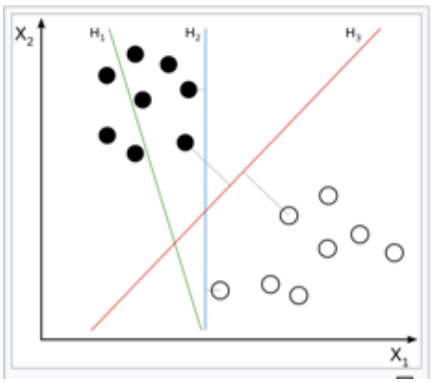


图 4. 二维空间的分类问题

图上的 H1,H2,H3 都是决策边界，大家可以直观的感觉到这三个决策边界中最好的一个是 H3 ，为什么？ 因为 H1 不能将两类样本点很好的分开， H2 可以很好的分开， 但是这个分割线离某些样本点太近了，差点不能分开。

上述的描述，都是很直观的判断，没有任何理论假设。那么我们重点看下 H2 和 H3 这两条决策边界，假设给出的样本就是我们全部的训练集，那么两条决策边界都可以达到0%损失， 也就是全部正确分类，大家想想逻辑回归迭代收敛的条件，是不是就意味着我们找到了最优解，走到了梦寐以求的山谷！没错，这两条决策边界都满足逻辑回归最优解的要求。也就是说， 如果逻辑回归最后收敛在了 H2 这条决策边界上，算法就会停止迭代，我们找到了最优模型。但是很显然，这个仍然不是最佳的，因为直觉告诉我们，它真的是差点就分错，这个差点其实就有很大的概率， 当预测一个新样本时，被错误分类的概率会更大，其实这句话描述的就是模型的 泛化能力 差。

而 H3 我们直观的认为，当它预测一个新样本时，被错误分类的概率会小些，即 泛化能力 好！这就是支持向量机(SVM)和其他分类模型最本质的差异，SVM的目标是找到像 H3 这样， 人为感觉泛化能力最强的分类器，为什么加个 人为 ,因为我们无法获取真实样本的分布，所以一切都是近似！ 好了，下面就让我们来将这个目标进行数学化描述:

很容易发现，这个人为感觉泛化能力最强的分类器，具有这样的特点：这个决策边界和离它最近的样本的距离最大，白话说就是尽可能的将两类样本分的越开越好。因此SVM就是间隔最大的线性分类器！ 所以它牛哄哄了那么多年，因为它是最优分类器。

越火研究的人就越多，越容易发展到新的高度，SVM不仅可以解决线性可分的问题，也可以通过引入惩罚因子解决近似线性可分的问题，当然将核技术应用于SVM，使得SVM可以解决线性不可分的问题， 谈到SVM一个经常被说到的话：高斯核可以将样本点映射到无穷维！这个后面我们会详细说明为什么，这里就感受下它的牛气就行了。

4.2. 高维空间

这个题目有点大，其实这里只是想介绍高维空间的少许基本概念，为后面SVM的推导做基础储备。

4.2.1. 超平面的表示

首先，为什么要研究高维空间，其实这个问题很容易理解。机器学习的核心是数据，而数据都是由特征向量表示，如果我们将数据想象成空间中的点，那么特征的总数就是这个高维空间的总数， 一个特征代表了高维空间的一个维度。

在二维平面上，一个线性分类器的决策边界是一条直线，在高维空间上就是个超平面。那么如何表示这个超平面呢？先来研究下二维平面的情况，我们都知道二维平面上的线性决策面就是直线， 假设二维平面上的两个维度分别为x和y,那么任意一条直线可以表示为如下形势:

$$ax + by + c = 0$$

也许上面的形势你不太习惯，那再变换下形势就可以写成

$$y = -\frac{a}{b}x - \frac{c}{b}$$

这个形式是我们初中所学的基本形式，其中$-\frac{a}{b}$为直线的斜率，而$-\frac{c}{b}$为直线的截距。聪明的你，一定会想办法把这条直线和我们的机器学习模型对比结合起来，比如，你会想到变量x和y就相当于数据中的两个特征，而a和b相当于特征对应的权重，初中学习的数学告诉我们x和y是变量，而a和b一般为常量，而机器学习刚好相反，机器学习的目的是为了学到这条直线，因此未知数就是a和b，我们把未知数用向量表示，令$\vec{w} = (\vec{a}, \vec{b})$，向量$\vec{w}$叫做法向量。来看下法向量和斜率之间的关系，下图假设a=1, b=-1, c=0，于是得到一条斜率为1的直线。

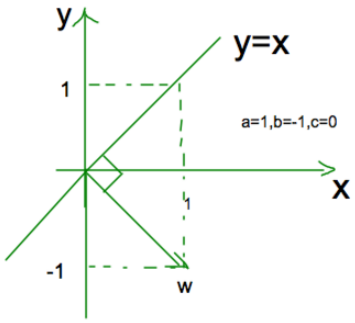


图 5. 二维空间直线和法向量的关系

很容易发现，斜率的方向是和直线平行的，而法向量刚好和直线垂直，这个结论同样适应于高维空间，即高维空间的超平面的法向量和平面也是垂直的，如果我们将x和y也统一成一个变量的不同下标，则直线可以写成$\vec{w} \cdot \vec{x} + c = 0$，为了方便向量的箭头省略掉，将截距$c$换成$b$，则得到超平面的法向量表示。

$$w \cdot x + b = 0$$

4.2.2. 点到平面距离

因为SVM是间隔最大的线形分类器，中间涉及到间隔的概念，因此势必会计算样本点到决策超平面的距离计算，这里我们利用向量点积运算的定义，给出点到平面的距离,向量的点积定义为:

$$\vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}| \cdot cos\theta$$

其中，\theta为两个向量的夹角，为了求解空间中的任意一点x_0到超平面的距离，我们可以在超平面上任取一点x_1，则向量$x_0 \overrightarrow{-} x_1$在法向量上的投影就是点$x_0$到超平面的距离，如下图所示，其实比较直观。

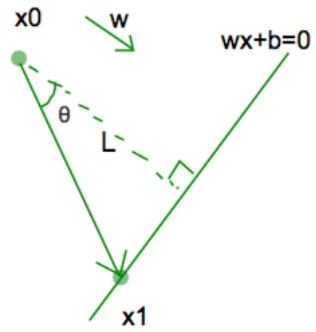


图 6. 点到平面的距离

由上图，我们可以看到点到平面的距离L其实就是线段x_0x_1乘以$cos\theta$，刚好是向量点积的两项。于是得到点到平面的距离公式:

$$\begin{aligned} w\cdot(x_0-x_1) &= |w||x_0-x_1|\cdot cos\theta \\ &= L\cdot|w| \\ \Leftrightarrow L &= \frac{w\cdot(x_0-x_1)}{|w|} \\ &= \frac{w\cdot x_0 - w\cdot x_1}{|w|} \\ &= \frac{wx_0+b}{|w|} \end{aligned}$$

因为x_1是平面$w\cdot x+b=0$上的点，所以$-w\cdot x_1=b$，记录下我们的成果，点到平面的距离为:

$$L=\frac{w\cdot x_0+b}{|w|} \qquad (4.1)$$

根据公式(4.1)，我们发现一个很有趣的地方，SVM的目的就是要求解超平面，也就是参数w和b，方法涉及优化点到平面的距离，但是上面的距离公式却告诉我们，不管你的参数求的结果是多少，比如w=10,b=15，假设此时计算的L=100(为了简化问题,这里假设了维度为1),那么我们都可以等比例放大w和b使得距离同比例变大，放大10倍，则距离变为1000，距离虽然等比里变大了，超平面却没有发生变化，因为放大倍数可以约掉。

上面的问题其实在告诉我们，同一个平面其法向量和截距的结果可以有无数多个，等比例缩放的解都是可行解。换句话说，因为这个特性使得点到平面的距离可以取任意值(除了0),那不妨我们就固定距离，这种约束条件下求得的参数将会是唯一的，想缩放已经不可能了，距离约束了，有点像结果的归一化问题，为了简化计算，后面我们就将这个可以取任意值的距离设为1，即:

$$w\cdot x_0+b=1 \qquad (4.2)$$

$$L=\frac{1}{||w||} \qquad (4.3)$$

刚距离中维度为1，所以绝对值和二范式相等，高维空间一般都用二范式表示向量的模(所有分量的平方和再开根号)，我们的优化目标就是最大化间隔，即

$$\underset{w,b}{max}\frac{1}{||w||}$$

再来看下约束条件有哪些，我们假设二分类的结果y_i取值为1和-1，假设正样本为1，负样本为-1，我们希望所有正样本都满足$w\cdot x_{i+}+b\geqslant 1$,所有负样本都满足$w\cdot x_{i-}+b\leq -1$,这两个约束条件可以写成如下一个约束条件:

$$s.t. \qquad y_i(w\cdot x_i+b)-1\geq 0$$

我们将最大化问题转为最小化问题，就变为凸优化问题，于是我们得到了下面的线形可分支持向量机学习的最优化问题:

$$\underset{w,b}{min} \qquad \frac{1}{2}||w||^2 \qquad (4.4)$$

$$s.t. \qquad y_i(w\cdot x_i+b)-1\geq 0 \qquad (4.5)$$

这是个典型的凸二次优化问题，凸优化问题是指约束最优化问题，这种问题一般会通过引入拉格朗日乘子，转为一个优化函数的形式。

4.3. 问题优化

4.3.1. 理解对偶问题

凸优化问题的约束条件一般限定包括$=0$和≤ 0，因为≥ 0可以很容易转换为≤ 0的问题，所以上节的支持向量机学习的最优化问题应该写成:

$$\min_{w,b} \quad \frac{1}{2}||w||^2 \qquad (4.6)$$

$$s.t. \quad 1 - y_i(w \cdot x_i + b) \le 0 \qquad (4.7)$$

根据上述约束条件，对每一个不等式约束引入拉格朗日乘子$\alpha_i \ge 0$，得到拉格朗日函数:

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 + \sum_{i=1}^{N} \alpha_i[1 - y_i(w \cdot x_i + b)] \qquad (4.8)$$

先不考虑我们原问题是最大问题还是最小化问题，首先这种变换必须是等价的才有意义，也就是公式(4.8)必须满足公式(4.6)和(4.7)的约束条件，先来看下(4.7)的不等式约束

分析:公式(4.7)的约束是小于等于0，我们假设给拉格朗日函数附加最小化约束，即$min\ L(w, b, \alpha)$,显然$1 - y_i(w \cdot x_i + b)$ 为负时会使得无最小值存在， 因此不能满足(4.7)的要求，那么反过来，如果给拉格朗日函数附加最大化约束呢，即$max\ L(w, b, \alpha)$, 我们发现当$1 - y_i(w \cdot x_i + b)$ 想要大于0时(即违背公式(4.7)约束),我们可以取$\alpha_i = 0$, 而当$1 - y_i(w \cdot x_i + b)$ 小于0时，取$\alpha_i > 0$即可，刚好满足公式(4.7)的约束条件。

因此为了满足公式(4.7)的约束条件，拉格朗日函数必须附加最小化约束条件，即:

$$max\ L(w, b, \alpha)$$

再考虑公式(4.6)的约束，这个比较简单，因为是等式约束，所以将最小化约束条件加在朗格朗日函数前就可以了，即得到朗格朗日与原问题完全等价的函数约束形式:

$$\max_{\alpha} \min_{w,b} L(w, b, \alpha) \qquad (4.9)$$

通过朗格朗日变换，我们发现原始问题是一个最小最大值问题，这个问题不好直接求解，是否转换为对偶问题后方便求解呢?

这里先简单说下什么是对偶问题，比如这里原问题是一个最小最大值问题，对偶问题就是最大最小值问题，因此公式(4.9)的对偶问题形式就是:

$$\min_{\alpha} \max_{w,b} L(w, b, \alpha) \qquad (4.10)$$

假设原问题的最优解为p^*，对偶问题的最优解为q^*，那么一个很显然的结论是: $q^* \le p^*$,这个结论不需要什么证明，看下这两个问题的负号关系就知道了， 一个是最大值里取最小的，另一个是在最小值里取最大的，显然这个不等式是成立的。但是，问题来了，什么情况下取等号呢? 取不到等号，显然对偶问题和原始问题不等价！

4.3.2. 等价性证明

由上节可知，对偶问题和原始问题的关系

$$d^* \le q^*$$

我们把上面的不等式约束叫做 弱对偶性质 ,顺其自然，我们可以引入一个重要的概念： 对偶间隔 ，即$q^* - d^*$,因此对偶间隔肯定大于或等于0，那么有没有可能在某种情况下，对偶间隔消失呢? 也就是对偶问题的最优解与原始问题的最优解相等呢?

下面介绍一个让对偶间隔消失的充分条件， Slater条件 :

> 存在x满足:
> $g_i(x) < 0, i = 1, 2, .., m$
> $h_i(x) = 0, i = 1, 2, \dots, p$

$h_i(x)$大家暂时可以不用管，这是一个等式约束问题，SVM问题中，只有不等式约束，所以第二个条件可以认为已满足， Slater条件 即是说存在x，使不等式约束中的"小于等于号"要严格取到"小于号"。

可以证明对于凸优化问题(QP问题)，如果 Slater条件 满足，则

$$d^* = p^*$$

这种情况称为"强对偶性质",如果对偶问题存在最优点α^*，并且对应的最优值$d^* = \theta(\alpha^*)$等于p^*,这时会发生什么？来看下对偶问题的推导过程:

$$\begin{aligned}\theta(\theta(\alpha)) &= \inf_x(L(x,\alpha)) \\ &= \inf_x(f(x) + \sum_{i=1}^{N}\alpha_i g_i(x)) \\ &\le f(x^*) + \sum_{i=1}^{N}\alpha_i g_i(x^*) \\ &\le f(x^*) \\ &= p^*\end{aligned}$$

在对偶间隔消失的情况下，上面的所有不等式都严格取等号，根据上面两个不等式处取等号，我们可以得出下面两个结论

- 第一个不等式处取等，会使得原始问题的最优点x^*是使$L(x,\alpha^*)$取最小值的点。
- 第二个不等式处取等，说明

$$\sum_{i=1}^{N}\alpha_i g_i(x^*) = 0$$

由于我们限制了每个$\alpha_i \ge 0$,所以上式每一项都是非负的，这样我们又可以得到一个结论

$$\alpha_i g_i(x^*) = 0, \quad \forall i$$

因为$g_i(x)$为我们的不等式约束，所以也就是说，要么拉格朗日乘子为0，要么不等式为0；这句话再往深里说就是，不等式约束不为0的样本点对应的朗格朗日乘子都为0， 也就是不等式约束不起作用，对训练模型没有帮助，而不等式为0的点，不等式约束才成立。而使得不等式为0的点都是满足下式的点:

$$y_i(w \cdot x_i + b) = 1 \cdot$$

而满足上面等式的点都是 支持向量 ，因此在模型训练是我们只需要关注那些支持向量就可以了，这也是支持向量机名称的由来。

4.4. 对偶问题求解

根据拉格朗日对偶性，原始问题的对偶问题是极大极小值问题:

$$\max_{\alpha}\min_{w,b}L(w,b,\alpha)$$

所以为了得到对偶问题的解，我们需要先求$L(w,b,\alpha)$对w,b的极小值，再求对α的极大值。

4.4.1. 推导和结论

(1).求$\min_{w,b}L(w,b,\alpha)$

将拉格朗日函数分别对w, b求导，令导数为0.

$$\frac{\partial}{\partial w}L(w,b,\alpha) = w - \sum_{i=1}^{N}\alpha_i y_i x_i = 0$$

$$\frac{\partial}{\partial b}L(w,b,\alpha) = \sum_{i=1}^{N}\alpha_i y_i = 0$$

等价于

$$w = \sum_{i=1}^{N}\alpha_i y_i x_i \qquad (4.11)$$

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \qquad (4.12)$$

将公式(4.11)代入拉格朗日函数，并利用公式(4.12)可以得到:

$$\begin{aligned} L(w,b,\alpha) &= \frac{1}{2}||w||^2 + \sum_{i=1}^{N}\alpha_i[1 - y_i(w\cdot x_i + b)] \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i\cdot x_j) - \sum_{i=1}^{N}[1 - y_i((\sum_{j=1}^{N}\alpha_j y_j x_j)\cdot x_j + b) \\ &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i\cdot x_j) + \sum_{i=1}^{N}\alpha_i \end{aligned}$$

即

$$\min_{w,b}L(w,b,\alpha) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i\cdot x_j) + \sum_{i=1}^{N}\alpha_i$$

(2)求$\min_{w,b}L(w,b,\alpha)$对α的极大值，即是对偶问题

$$\max_{\alpha} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i\cdot x_j) + \sum_{i=1}^{N}\alpha_i \qquad (4.13)$$

$$s.t. \quad \sum_{i=1}^{N}\alpha_i y_i = 0$$

$$\alpha_i \geq 0, \qquad i = 1, 2, \dots, N$$

将公式(4.13)的极大值问题转换为极小值，就得到下面与之等价的对偶最优化问题:

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i\cdot x_j) - \sum_{i=1}^{N}\alpha_i \qquad (4.14)$$

$$s.t. \quad \sum_{i=1}^{N}\alpha_i y_i = 0 \qquad (4.15)$$

$$\alpha_i \geq 0, \qquad i = 1, 2, \dots, N \qquad (4.16)$$

推到这里，接下来怎么求解？ 这么多未知数!

4.4.2. SMO算法

SMO算法和坐标上升类似，坐标上升法是指当有两个变量需要求解时，可以先固定一个变量不变，求另一个变量的最优化值，并更新另一个变量，然后下一轮再用同样的方法更新上一轮作为常量的变量值。 SMO算法和坐标上升法的差异主要体现在两点：

- 由于SVM的限制条件$\sum_i \alpha_i y_i = 0$，所以不能只使用一个坐标，改为同时更新两个坐标
- 采用启发式方法，找到每次更新的坐标，而不是按顺序更新

第一点很好理解，因为有等式的限制，假设共有N个样本，就对应有N个朗格朗日乘子，这N维空间的自由度实际只有$N-1$,因为当$N-1$个变量已知时，最后一个变量的值是固定的； 所以SMO同时更新两个变量相当于同时更新一个;

第二点，有点类似梯度下降，目的都是为了让算法收敛的更快，SMO算法是基于SVM特有KKT条件限制框架下的优化算法，因此可以根据约束条件选择收敛速度最快的样本进行更新，后面具体会介绍。

下面开始正式介绍 `SMO算法` ，又叫序列最小最优化(Sequential Minimal Optimization)算法，该算法是要解决上节提到的如下优化问题:

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N}\alpha_i \qquad (4.17)$$

$$s.t. \quad \sum_{i=1}^{N}\alpha_i y_i = 0 \qquad (4.18)$$

$$0 \le \alpha_i \ge C, \qquad i = 1, 2, \dots, N \qquad (4.19)$$

这里需要说明下公式(4.19)和之前提到的约束条件稍有变化，主要原因是考虑到后面要介绍的软间隔分类器，因为SMO算法可以同时应用于软间隔和硬间隔分类器，为了不失一般性， 我们这里就直接兼顾两种条件下的优化问题，只是稍微改动，其实对介绍算法本身没什么影响。

4.4.2.1. SMO算法思想

SMO算法是一种启发式算法，基本思想是：

- 如果所有变量的解都满足此最优化问题的KKT条件，那么这个最优化问题的解就得到了。因为KKT条件是该最优化问题的充分必要条件。
- 否则选择两个变量，固定其他变量，针对这两个变量构建一个二次规划问题，这个二次规划问题关于这两个变量的解应该更接近原始二次规划问题的解，因为这会使原始二次规划问题的目标函数值变小。
- 更重要的，这时子问题可以通过解析方法求解，说白话就是在固定其他参数以后，这就是一个单变量二次规划问题，显然有闭式解，不必再调用数值优化算法，所以快。
- 每次迭代的子问题有两个变量，一个是违反KKT条件最严重的那一个，另一个由约束条件自动确定，如此SMO算法将原始问题不断分解为子问题并对子问题求解，进而达到求解原始问题的目的。

算法基本思想中提到，每次迭代都是两个变量中有一个是违反KKT条件最严重的那一个，这其实就是启发式方法的核心思想，我们可以这样理解这个观点：根据公式(4.17)原始问题是个最大化问题，而满足KKT条件的解是最优解， 也就是说只要参数α_i中有一个参数不满足KKT条件，那么一轮迭代以后，目标函数的值就会变大(越大越好),KKT条件违反的程度越大，迭代后目标函数值增幅越大，优化效果越显著。

所以SMO算法中提到第一个变量就是违背KKT条件程度最大的变量，那么第二个变量应该选择使目标函数值增大最快的变量，关键在于如何找这个变量，SMO算法使用了启发式方法，即当第一个变量确定后，选择使两个变量对应样本之间差别最大的变量作为第二个变量， 直观上来说，更新两个差别很大的变量，比更新两个相似的变量会带给目标函数值更大的变化，关于怎么度量这个间隔，后面我们会提到可以使用偏差间隔，这里先略过。

为什么说，子问题的两个变量中只有一个自由变量，因为假设α_1, α_2为两个变量，其他参数$\alpha_3, \alpha_4, \dots, \alpha_N$固定,那么由等式(4.18)约束可知

$$\alpha_1 = -y_1 \sum_{i=2}^{N} \alpha_i y_i$$

所以如果α_2确定，那么α_1也随之确定，所以子问题中同时更新两个变量。

整个SMO算法包括两部分内容：求解两个变量二次规划的解析方法和选择变量的启发式方法。

4.4.2.2. 二次规划求解方法

不失一般性的，假设选择的两个变量分别为α_1, α_2，其他变量$\alpha_i (i = 3, 4, \dots, N)$是固定的，于是SMO的最优化问题的子问题可以写成

$$\min_{\alpha_1, \alpha_2} W(\alpha_1, \alpha_2) = \frac{1}{2} K_{11} \alpha_1^2 + \frac{1}{2} K_{22} \alpha_2^2 + y_1 y_2 K_{12} \alpha_1 \alpha_2 - (\alpha_1 + \alpha_2) + y_1 \alpha_1 \sum_{i=3}^{N} y_i \alpha_i K_{i1} + y_2 \alpha_2 \sum_{i=3}^{N} y_i \alpha_i K_{i2} \tag{4.20}$$

$$s.t. \quad \alpha_1 y_1 + \alpha_2 y_2 = -\sum_{i=3}^{N} y_i \alpha_i = \varsigma \tag{4.21}$$

$$0 \le \alpha_i \le C, \quad i = 1, 2 \tag{4.22}$$

其中，$K_{ij} = K(x_i, x_j), i, j = 1, 2, \dots, N, varsigma$是常数，目标函数式(4.20)中省略了不包含$\alpha_1, \alpha_2$的常数项。

为了求解两个变量的二次规划问题(4.20)~(4.22)，首先分析约束条件，然后在此约束条件下求极小。

由于只有两个变量(α_1, α_2),约束可以用二维空间中的图形表示，如下图所示.

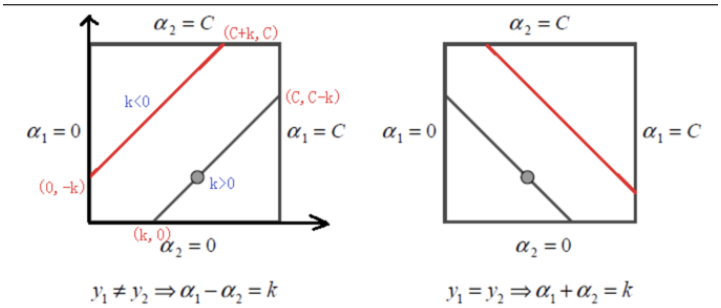


图 7. 二变量优化问题图示

不等式约束(4.22)使得(α_1, α_2)在盒子$[0, C] times [0, C]$内，而由于$y_i \in 1, -1$，因此等式约束(4.21)使得(α_1, α_2)在平行于盒子的对角线的直线上， 因此要求的是目标函数在一条平行于对角线的线段上的最优值。这使得两个变量的最优化问题成为实质上的单变量的最优化问题，这里不妨考虑变量α_2为变量。

先解释下上图中的一些数学关系，α_1你可以理解为水平的横坐标，α_2你可以理解为垂直的纵坐标，如果$y_1 \neq y_2$，由于y_i只能取$(1, -1)$，所以等式约束条件相当于两个变量的差值为常量， 这里定义这个常量为k,如左图，又分为两种情况，当平行线为上面红色线段时，考虑下面的交点，此时的横坐标为0，纵坐标等于$\alpha_2 - \alpha_1 = -k$；再考虑上面的交点，纵坐标一定为C， 那么横坐标刚好等于$\alpha_1 = C + k$。

其他的情况分析完全类似，假定下面的交点的纵坐标为L，上面交点纵坐标为H，问题的初始可行解为$\alpha_1^{old}, \alpha_2^{old}$,最优解为$\alpha_1^{new}, \alpha_2^{new}$,并且假设在沿着约束方向未经剪辑时 的$\alpha_2$的最优解为$\alpha_{new,unc}$，则此时的变量$\alpha_2$满足:

$$L \le \alpha_2^{new} \le H \tag{4.23}$$

并且由上面分析，如果$y_1 \neq y_2$，则

$$L = max(0, \alpha_2^{old} - \alpha_1^{old}), \quad H = min(C, C + \alpha_2^{old} - \alpha_1^{old})$$

而如果$y_1 = y_2$，则

$$L = max(0, \alpha_2^{old} + \alpha_1^{old} - C), \quad H = min(C, \alpha_2^{old} - \alpha_1^{old})$$

由因为α_2^{new}同时满足(4.23)，所以

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

上面公式中的L, H我们已经知道如何求解，那么$\alpha_2^{new,unc}$到底是什么？为了下面的求解方便，我们给出两个负号:

$$g(x) = \sum_{j=1}^{N} \alpha_j y_j K(x_j, x) + b$$

令

$$E_i = g(x_i) - y_i = (\sum_{j=1}^{N} \alpha_j y_j K(x_j, x_i) + b) - y_i, \quad i = 1, 2 \qquad (4.24)$$

当$i = 1, 2$时，E_i为函数$g(x)$对输入x_i的预测值与真实输出y_i之差。

这里我们先给出结论，证明留到下个小节，不关心过程的可以直接跳过下个小节。

$$\alpha_2^{new,unc} = \alpha_2^{old} + \frac{y_2(E_1 - E_2)}{\eta}$$

其中，

$$\eta = K_{11} + K_{12} - 2K_{12} = ||\Phi(x_1) - \Phi(x_2)||^2$$

$\Phi(x)$是输入空间到特征空间的映射，$E_i, i = 1, 2$，由公式(4.24)给出.

计算出了α_2^{new}，顺势可以求得α_1^{new}

$$\alpha_1^{new} = \alpha_1^{old} + y_1 y_2(\alpha_2^{old} - \alpha_2^{new})$$

4.4.2.3. 公式证明

引进记号

$$v_i = \sum_{j=3}^{N} \alpha_j y_j K(x_i, x_j) = g(x_i) - \sum_{j=1}^{2} \alpha_j y_j K(x_i, x_j) - b, \quad i = 1, 2$$

目标函数可写成

$$W(\alpha_1, \alpha_2) = \frac{1}{2} K_{11}\alpha_1^2 + \frac{1}{2} K_{22}\alpha_2^2 + y_1 y_2 K_{12}\alpha_1\alpha_2 - (\alpha_1 + \alpha_2) + y_1 v_1 \alpha_1 + y_2 v_2 \alpha_2 \qquad (4.25)$$

由$\alpha_1 y_1 = \varsigma - \alpha_2 y_2$ 及$y_i^2 = 1$，可将α_1表示为

$$\alpha_1 = (\varsigma - y_2\alpha_2)y_1$$

代入公式(4.25)，得到只有α_2的目标函数:

$$W(\alpha_2) = \frac{1}{2}K_{11}(\varsigma-\alpha_2 y_2)^2+\frac{1}{2}K_{12}\alpha_2^2+y_2K_{12}(\varsigma-\alpha_2y_2)\alpha_2 \\ -(\varsigma-\alpha_2y_2)y_1-\alpha_2+v_1(\varsigma-\alpha_2y_2)+y_2v_2\alpha_2$$

对α_2求导，可得

$$\begin{aligned}(K_{11}+K_{22}-2K_{12})\alpha_2 &= y_2(y_2-y_1+\varsigma K_{11}-\varsigma K_{12}+v_1-v_2) \\ &= y_2[y_2-y_1+\varsigma K_{11}-\varsigma K_{12}+(g(x_1)-\sum_{j=1}^{2}y_j\alpha_jK_{1j}-b) \\ &-(g(x_2)-\sum_{j=1}^{2}y_j\alpha_jK_{2j}-b)]\end{aligned}$$

将$\varsigma=\alpha_1^{old}y_1+\alpha_2^{old}y_2$ 代入，得到

$$\begin{aligned}(K_{11}+K_{22}-2K_{12})\alpha_2^{new,unc} &= y_2((K_{11}+K_{22}-2K_{12})\alpha_2^{old}y_2+y_2-y_1+g(x_1)-g(x_2)) \\ &= (K_{11}+K_{22}-2K_{12})\alpha_2^{old}+y_2(E_1-E_2)\end{aligned}$$

将$\varsigma=K_{11}+K_{22}-2K_{12}$ 代入，于是得到

$$\alpha_2^{new,unc}=\alpha_2^{old}+\frac{y_2(E_1-E_2)}{\eta} \qquad (4.26)$$

证毕.

4.4.2.4. 变量选择方法

根据上节的内容，我们已经知道了如何在每轮迭代中更新参数，现在的问题是我们每轮迭代时应该选择哪两个参数进行更新。如果像坐标上升法那样顺序遍历，显然效率太低。 我们选择方法的依据就是我们的约束条件:

(1)第一个变量的选择

SMO通过启发式方法来选择参数变量，选择第一个变量的过程被称为外层循环，外层循环是在训练样本中选择违反KKT条件最严重的样本点，并将其对应的参数变量作为第一个变量，具体:

$$\alpha_i=0 \Leftrightarrow y_ig(x_i)\geq 1 \qquad (4.27)$$
$$0<\alpha_i<C \Leftrightarrow y_ig(x_i)=1 \qquad (4.28)$$
$$\alpha_i=C \Leftrightarrow y_ig(x_i)\leq 1 \qquad (4.29)$$

其中$g(x_i)=\sum_{j=1}^{N}\alpha_jy_jK(x_i,x_j)+b$.

该检验是在ξ范围内进行的，而且我们优先选择样本系数中满足公式(4.28)的样本点，即支持向量。不考虑其他两个相等界上的样本点，是因为他们不是支持向量，对模型的优化基本不起作用，因此参数基本不会怎么变化。 检查支持向量是否满足KKT条件，如果这些样本点都满足，那么遍历整个训练集，检查他们是否都满足。

(2)第二个变量的选择

SMO将第二个变量的选择过程称为内层循环，假设在外层循环中已经找到了第一个变量α_1，那么第二个变量α_2的选择标准就是希望能使α_2能有足够大的变化。足够大的变化意味着更新后目标值会得到很大程度的优化， 算法收敛速度加快，因为第一个变量已经选择了违反KKT条件的参数，那么如何找到使得第二个变量有足够大变化的参数呢?

由公式(4.27)可知，α_2与$|E_1-E_2|$有关，当α_1确定后，找到违反KKT条件中使得$|E_1-E_2|$最大的样例点的系数作为α_2,这里的选择仍然是启发式的， 即优先选择支持支持向量。

在具体实现时，这里就有一个明显的$trick$,就是将误差值E_i进行缓存。

更新完参数α_1, α_2以后，需要同步更新参数b，由公式(4.28)可知:

$$\sum_{i=1}^{N} \alpha_i y_i K_{i1} + b = 1$$

于是，

$$b_1^{new} = y_1 - \sum_{i=3}^{N} \alpha_i y_i K_{i1} - \alpha_1^{new} y_1 K_{11} - \alpha_2^{new} y_2 K_{21} \qquad (4.30)$$

由E_1(4.24)的定义，可知

$$E_1 = \sum_{i=3}^{N} \alpha_i y_i K_{i1} + \alpha_1^{old} y_1 K_{11} + \alpha_2^{old} y_2 K_{21} + b^{old} - y_i$$

因此公式(4.30)的前两项可以写成

$$y_i - \sum_{i=3}^{N} \alpha_i y_i K_{i1} = -E_1 + \alpha_1^{old} y_1 K_{11} + \alpha_2^{old} y_2 K_{21} + b^{old}$$

再带入公式(4.30)，可得到

$$b_1^{new} = -E_1 - y_1 K_{11}(\alpha_1^{new} - \alpha_2^{new}) - y_2 K_{21}(\alpha_2^{new} - \alpha_2^{old}) + b^{old} \qquad (4.31)$$

同样的，如果$0 < \alpha_2^{new} < C$,那么

$$b_2^{new} = -E_2 - y_1 K_{12}(\alpha_1^{new} - \alpha_1^{old}) - y_2 K_{22}(\alpha_2^{new} - \alpha_2^{old}) + b^{old} \qquad (4.32)$$

由于b只是一个参数变量，那么如何选择应用哪个作为更新标准呢？当然是哪个准选择哪个！那么如何判断准不准呢？就是更新后的哪个α在$(0, C)$之间，也就是属于支持向量了，于是就有了如下的判断:

$$b = \begin{cases} b1, & if \ \ 0 \le \alpha_1^{new} \le C \\ b2, & if \ \ 0 \le \alpha_2^{new} \le C \\ \frac{b1+b2}{2}, & others \end{cases}$$

在每次完成两个变量更新后，还需要同步更新E_i,作为下次计算的缓存，更新方法如下:

$$E_i^{new} = \sum_{S} y_j \alpha_j K(x, x_j) + b^{new} - y_i$$

其中，S是所有支持向量x_j的集合。

4.5. 软间隔分类器

4.5.1. 优化问题

上节我们讲解了线性可分问题的支持向量机学习方法，而在实际中，这种完全线性可分的情况很少，因此不适用于对线性不可分训练的情况， 如何才能将上节结论推广到线性不可分的问题呢?

这就需要修改硬间隔最大化，使其成为软间隔最大化，当然这种情况，我们需要假设训练数据中有少量特异点，将这些特异点去掉后， 剩下的大部分的样本点组成的集合是线性可分的。

线性不可分意味着某些样本点(X_i, y_i)不能满足间隔大于等于1的约束条件。为了解决这个问题，可以给每个样本引入一个松弛因子，$\xi \geq 0$，使得函数间隔加上松弛变量大于等于1，这样约束条件就变味：

$$y_i(w \cdot x_i + b) \geq 1 - \xi$$

同时，对每个松弛变量，支持一个代价，则目标函数变成：

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i$$

其中，$C > 0$称为惩罚参数，一般由应用问题决定，属于超参数，取值大时，意味着对误分类的惩罚增大，目标函数包含两层含义：使得间隔 尽量大，同时使误分类点的个数尽量小。

根据上面的思路就可以解决训练数据集线性不可分(近似可分)的问题，线性不可分的线性支持向量机的学习问题就变成了如下的凸二次规划问题:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & y_i(w \cdot x_i + b) \geq 1 - \xi_i,\ i = 1, 2, \dots, N \\ & \xi_i \geq 0,\ i = 1, 2, \dots, N \end{aligned}$$

上述的优化问题基本等同于上节优化问题，具体的推导这里不再赘述，可以参考上节推导过程，或者直接参考李航博士的 统计学习方法 一书。

上节我们已经了解到了线性可分的问题，支持向量都在间隔边界上，但是线性不可分的情况稍微复杂，这些支持向量可以位于间隔边界上，间隔边界内甚至是误分类的一侧。

4.5.2. 合页损失

其实线性支持向量机还有另一种解释，就是利用前面讲到的损失函数的概念进行描述，相当于最小化以下目标函数:

$$\sum_{i=1}^{N}[1 - y_i(w \cdot x_i + b)]_+ + \lambda||w||^2$$

其中第一项对应经验损失，而函数

$$L(y(w \cdot x + b)) = [1 - y(w \cdot x + b)]_+$$

称为合页损失，下标表示以下取正值得函数:

$$[z]_+ = \begin{cases} z, & z > 0 \\ 0, & z \leq 0 \end{cases}$$

4.6. 核技巧

关于 核技巧 实际同支持向量机模型没有必然的联系，是一个完全独立的方向，只是为了使用支持向量机处理线性不可分的训练样本时， 通过引入核技巧可以完美解决问题。本节我们也不打算详细讲解核技巧，只说其在支持向量机中的应用，但是其实也是比较简单。

首先，用线性分类方法求解非线性分类问题分为两步：首先使用变换将原空间的数据映射到新空间；然后在新空间里使用线性分类学习方法从训练数据中学习分类模型， 而核技巧就是属于这样的方法。

核技巧应用于支持向量机的基本思路就是：通过一个非线性变换将输入空间对应于一个特殊空间(希尔伯特空间)，使得在输入空间中的超曲面模型对应于特征空间中的超平面模型， 说白了就是在输入空间只能通过非线性的曲面分割的问题可以在特征空间线性分割。这样分类模型的学习任务通过在特征空间中求解线性支持向量机就可以完成。

由于支持向量机的特殊性，我们其实可以不需要显示地将输入空间的训练数据分别映射到特征空间，即寻找这个转换函数，我们其实只需要求得输入数据在特征空间中的内积就可以， 这是由于我们的优化目标，见公式(4.17)中仅涉及输入数据的内积运算。这个内积运算$x_i \cdot x_j$，可以直接用核函数来$K(x_i, x_j)$代替，此时对偶问题的目标函数就变为了:

$$W(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{N}\alpha_i$$

这等价于经过映射函数将原来的输入空间变换到一个新的特征空间，将输入空间的内积变换为特征空间的内积，在新的特征空间中学习线性支持向量机，当映射函数是非线性函数时， 学习到的含有核函数的支持向量机就是非线性分类模型。

也就是说，在核函数给定的条件下，可以利用解线性分类问题的方法求解非线性分类问题的支持向量机。而学习是隐式地在特征空间进行的，不需要显示的定义特征空间和映射函数， 这样的技巧就称为 核技巧 .

4.7. 深度思考

本章主要涉及的思考题包括:

- 如何理解KKT条件
- 为什么说高斯核可以将数据映射到无穷维

4.7.1. 深入理解KKT条件

4.7.1.1. 什么是KKT条件

对于具有等式和不等式约束条件的一般优化问题

$$\begin{aligned} &min,\ f(x) \\ &s.t.\ \ g_j(x) \le 0,\ \ (j = 1, 2, \dots, m) \\ &h_k(x) = 0,\ \ (k = 1, 2, \dots, l) \end{aligned}$$

KKT 条件给出了判断x^*是否为最优解得必要条件，即:

$$\begin{cases} \frac{\partial f}{\partial x_i} + \sum_{j=1}^{m}\mu_j\frac{\partial g_j}{\partial x_i} + \sum_{k=1}^{l}\lambda_k\frac{\partial h_k}{\partial x_i} = 0 \\ h_k(x) = 0, (k = 1, 2, \dots, l) \\ \mu_j g_j(x) = 0, (j = 1, 2, \dots, m) \\ \mu_j \le 0 \end{cases}$$

注意不等式的约束条件，必要条件里已经转换为了严格成立，即$\mu_j g_j(x) = 0$，这也就意味者不等式约束函数和拉格朗日乘子至少有一个为0，这也是支持向量的由来。

4.7.1.2. 等式约束优化问题

约束条件共分为两类，首先讨论较为简单的等式约束条件，对于等式约束条件的优化问题，常用的方法就是拉格朗日法，即把等式约束$g_i(x\cdot)$用一个系数与f(x)写为一个式子，称为拉格朗日函数，而系数称为拉格朗日乘子。 通过拉格朗日函数对各个变量求导，令其为零，可以求得候选值集合，然后验证求得最优值。为什么可以这样呢？接下来我们给予证明:

我们考虑下面的问题:

$$\begin{cases} min & f(x, y) \\ s.t. & g(x, y) = 0 \end{cases}$$

我们首先画出$f(x,y)$的等高线，然后画出$g(x,y)=0$这条曲线，如下图所示:

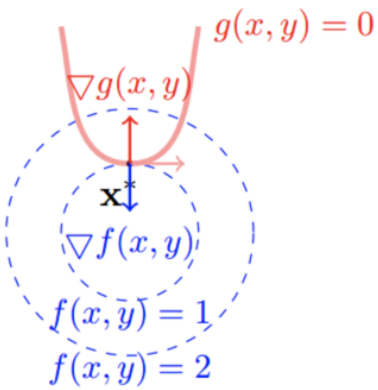


图 8. 等式约束

我们的目标是在曲线$g(x,y)=0$上找到一点使$f(x,y)$最小，我们先看等高线$f(x,y)=2$，发现它与$g(x,y)=0$有两个交点，我们想象当$f(x,y)$的值不断变小，$g(x,y)=0$与与等高线从有两个交点到若即若离的状态（一个交点，相切），最后没有交点。 相切得那个点一定比之前的两个交点值更小，因此只有等高线和曲线相切的时候，才可能取到最优值。

如上图所示，即等高线和曲线在切点的法向量必须有相同方向，这个点就是我们的目标点，我们知道这个目标点的切线与函数和曲线在该点的法线都垂直， 然后我们知道法线可以用该点的梯度表示，因此，我们可以得到目标点处满足:

$$\nabla f(x^*,y^*)=\lambda\nabla g(x^*,y^*)$$
$$\nabla L(x^*,y^*,\lambda)=0$$

于是我们构造函数:

$$L(x,y,\lambda)=f(x,y)-\lambda g(x,y)$$

求解它的最优解就等价于求原带约束的优化问题的最优解。

4.7.1.3. 不等式约束优化问题

KKT条件可以看成是拉格朗日乘子法的进一步推广，即对于不等式约束条件，如何求最优值问题，常用的方法是 KKT条件 .首先看下这类问题:

$$\begin{cases} min & f(x,y) \\ s.t. & g_i(x)\le 0 \end{cases}$$

我们的目标是在下图中阴影去或者边界找到一个点使得目标函数值最小。

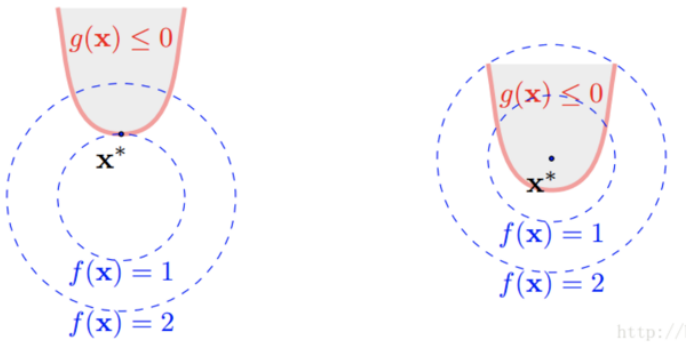


图 9. 不等式约束

要求是$g(x)\le 0$也就是是说图中的阴影区我们是要在阴影区或者边界上找一个点，使得$f(x,y)$最小，我们把他分成两种情况:

- 如果最优解恰好在这条边界上，等价于前面讲到的等式约束，那么这个就可以通过拉格朗日来求解

- 如果最优解在阴影区内部，我们是要找最优解，这个最优解不在边界上，在他内部，如果在内部那这个解一定是单纯考虑$f(x,y)$的最优解，也就是直接求$\nabla f(x,y)=0$的解

关于第二点，其实也很好理解，因为约束条件内包含了原始问题的最优解，也就是约束条件下的最优解等于原来问题的最优解，换句话说约束条件没用，那就可以让其对应的拉格朗日乘子为0就好了。

我们写出拉格朗日的函数如下:

$$L(x,\lambda)=f(x)-\sum_{i=1}^{m}\lambda_i g_i(x)$$

综合上述两种情况，我们可以得到在最优解$x=x^*$处，应满足以下条件:

$$\begin{aligned}
&\nabla_x L(x^*,\mu^*,\lambda^*)=0\\
&\nabla_\mu L(x^*,\mu^*,\lambda^*)=0\\
&\nabla_\lambda L(x^*,\mu^*,\lambda^*)=0\\
&\mu_j^* g_j(x^*)=0, && j=1,2,\dots,m\\
&g_j(x^*)\le 0, && j=1,2,\dots,m\\
&h_i(x^*=0), && i=1,2,\dots,l\\
&\lambda_i^*\ge 0, && i=1,2,\dots,m
\end{aligned}$$

我们直接从约束条件推出了 KKT条件 ,这个例子中的有些函数符号有点乱，不过倒是不影响理解，这里没有修订，是因为图片懒得自己画了，谅解.

4.7.2. 如何理解高斯核映射至无穷维

假设我们的核函数为$k(x,y)=(x^Ty)^2$，输入数据的维度为2，两个数据分别为$X=(x_1,y_1),Y=(x_2,y_2)$，则

$$k(x,y)=(x_1y_1+x_2y_2)^2=x_1^2x_2^2+2x_1x_2y_1y_2+x_2^2y_2^2$$

找到特征映射Φ，因为$k(x,y)=\Phi(x)\Phi(y)$，所以: $\Phi=(x_1^2,\sqrt{2}x_1^2x_2^2,x_2^2)$ 将数据从两维空间 R^2 映射到三维空间R^3.
刚才的核函数为一般的多项式核，再来看下高斯核:

$$k(x,y)=exp(-||x-y||^2)$$

同样每个向量的维度为2,两向量$X=(x_1,y_1),Y=(x_2,y_2)$，则

$$k(x,y)=exp(-||x_1-y_1||^2-||x_2-y_2||^2)=exp(-||x||^2)exp(-||y||^2)exp(2xy)$$

根据泰勒展开式:

$$e^x=\sum_{n=0}^{\infty}\frac{x^n}{n!}$$

带入上式，得到:

$$k(x,y)=exp(-||x||^2)exp(-||y||^2)\sum_{n=0}^{\infty}\frac{(2x^Ty)^n}{n!}$$

可以看出公式中的的泰勒展开式其实是0-n维的多项式核函数的和,我们知道多项式核函数将低维数据映射到高维(维度是有限的)，那么对于无限个不同维的多项式核函数之和的高斯核，其中也包括无穷维度的多项式核函数, 因此高斯核可以将输入维度从低维空间映射到无穷维。

5. 主题模型

如果你已经对 LDA 模型研究了很久，看过Rickjin的 LDA数学八卦 ，但是你仍然对其中的细节不是很理解，无法将所有知识串联起来，你可以直接跳到5.3.6节，那里梳理的思路中将所有涉及的知识进行了串联，希望可以解决你的困惑！

5.1. 解决什么问题

主题模型主要应用于自然语言处理领域，其核心应用正如其名，可以用来进行文章的主题发现，信息的语义提取，生成文章摘要，更进一步可以进行社区的挖掘。为了更好地说明本节主题内容，这里可以假定我们要处理地应用为 文章主题发现 。

文章主题发现的模型有很多，我个人比较喜欢LDA(Latent Dirichlet Allocation，隐狄利克雷分布)模型，它在文本建模中是一个非常优雅的模型， 刚开始学习这个模型的时候，读了Richjin的《LDA数学八卦》，被文章中涉及的数学之美深深吸引，可能是由于篇幅过长的原因，读完感觉很爽， 却无法将所有内容串联起来，每部分的内容都可以看懂，到最后却没能领悟到如此简单的实现过程中，那些数学的美都藏在何处?

个人感觉，学习LDA的门槛并不低，想要让大家都能理解也绝非易事。我反复阅读了许多相关博客和论文，等到拨开云雾见青天的时候，觉得很有必要将自己的领悟和遇到的坑都记录下来， 也可能是自己能力有限，无法用简短的语言将这个模型讲清楚，本节内容会比较多，我将尽我所能，从先验我本人为一个小白开始，不断假设、追问，又不断找到答案， 从一个正常人的思路，循序渐进讲述一个小白都可以理解的LDA，希望对大家有所助益。

5.2. 给问题建模

我们的目的是想训练一个主题模型，它可以自动完成新文章主题的发现，也就是说当我们将一篇文章输入给训练好的主题模型后，它可以返回该文章对应的 主题分布 ， 比如：我们将 天龙八部 这本书(假定为一篇很长的文章)作为模型的输入，则模型返回的结果为：[60%:武侠，30%:爱情，0%:计算机, 5%:教育...], 返回结果显示 天龙八部 属于武侠爱情小说的概率很高，这比较符合我们的预期，所以对应的我们可以说主题模型学的不错，而如果返回的结果显示90%的是计算机， 那将是一个很糟糕的模型。

上面的例子看上去很好理解，但是有个问题大家有没有考虑到：那些主题(武侠、爱情、计算机等)是哪里来的？其实，在模型训练时，我们丢给模型的是一堆文章， 这些文章的主题是什么我们也不知道，因此这个模型是一个无监督的模型，我们希望计算机可以帮我们自动发现主题，但这里的主题是数字，不能包含具体的含义。 打个比方，我们认为所有的文章共有K个主题，那么我们将主题进行编号 Z_i 表示第i个主题，模型可以返回文章对应每个主题编号的概率， 我们可以按照文章概率最大的主题编号对文章进行分类，这样就可以将文章分为k类，简单看下各类文章的内容，可以很容易的将文章分为：新闻、体育、教育等， 这不就是基于内容的新闻网站的典型应用么。

图 10. 跑题

在继续阅读前，大家可以想一个问题，我们的模型是否真的可以学出来？我个人任务这个思考很有意义，而且我也认为一定是可以学的出来，大家可以想下：文章是如何产生的？ 思考下我们上学时写作文的场景，在一篇作文开始书写前，我们首先看到的是作文的要求，比如写一篇 我的爸爸是... ，很显然这是一篇写人的文章， 然后有了这个主题后，我们的作文里的词就会围绕这个主题展开，很难想象在写这篇作文的时候你会用到 百合花 这种写物的词。简单点说，文章是由一个个词构成的， 但是每个文章的词又是由这篇文章的主题构成的，当老师给你作文打了0分，并批注 跑题 了的时候，就说明你文章里那800多个词真的有可能是 百合花 ， 其实我们模型学习的目的就是希望它可以像老师一样发现你作文的主题，很显然这个事情是可以做到的。

5.2.1. 建模

假定：我们提供的文章数目为M，称为 预料 ，预料中所有不重复的词共有V个，假定所有文章共包含的主题有K个，用d表示某个文档，k表示主题， w表示词汇,n表示词，下面的定义完全不用care。

- $z[d][w]$：第d篇文章的第w个词来自哪个主题，M行，X列，X为相应文档的长度，即词(可重复的数目
- $nw[w][t]$：第w个词是第t个主题的次数， `word-topic` 矩阵，列向量$nw[][t]$表示主题t的词频分布，共V行，K列
- $nd[d][t]$：第d篇文章中第t个主题出现的次数， `doc-topic` 矩阵，行向量$nd[d]$表示文档主题频数的分布，共M行，K列

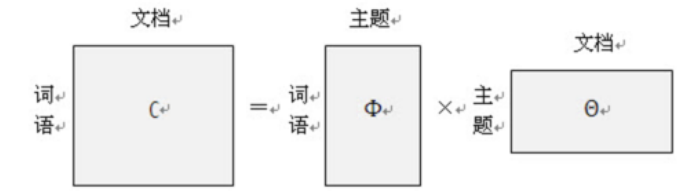


图 11. 跑题

5.2.2. 第一次思考

示例 5. 脑洞大开

- 训练的目标是什么? 如何抽象这个问题?

为了回答这个问题，首先想下我们有什么，我们有一个矩阵，$M*N$的矩阵，N是一个向量，元素N_i表示第i篇文章的长度(词个数)。 我们希望通过训练让计算机帮我们自动获得预料库中每篇文章的主题， 如何表征一篇文章的主题? 文章的主题不是显示存在的，是隐藏在文章的所有词背后的隐变量。
假定主题个数$K=5$，分别为 爱情、武侠、教育、音乐、体育 ，如果可以统计出文章对应这5个主题的概率就可以了，怎么统计？一个很直观的想法就是统计出文章的所有N_i个词对应主题的个数。

例如：我们有两篇文章，第一篇文章为：“小龙女教杨过武功，后来爱上了杨过”；第二篇文章为：“学习英语重在培养语感，多听英文歌曲可以培养语感”。
如何进行统计呢？每篇文章的词是已知，但是主题我们并不知道，只知道个数，假定我们已知每个主题对应的词分布，比如：

- 爱情：爱、爱上
- 武侠：小龙女、杨过、武功、教
- 教育：教、学习、英语、语感、培养、英文
- 音乐：歌曲、听
- 体育：篮球、足球、运动·

那统计后的结果大致为：

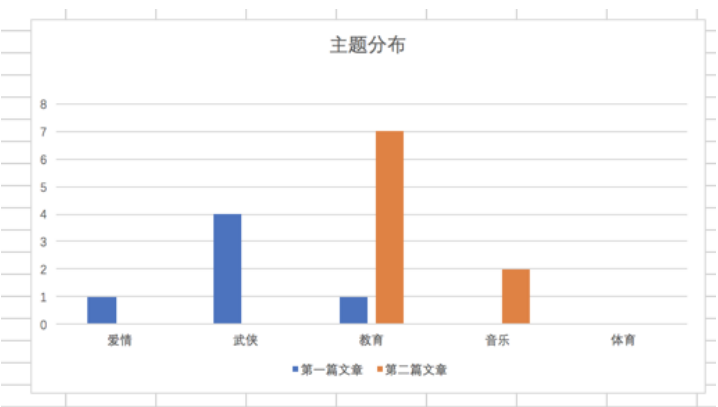


图 12. 文章主题分布

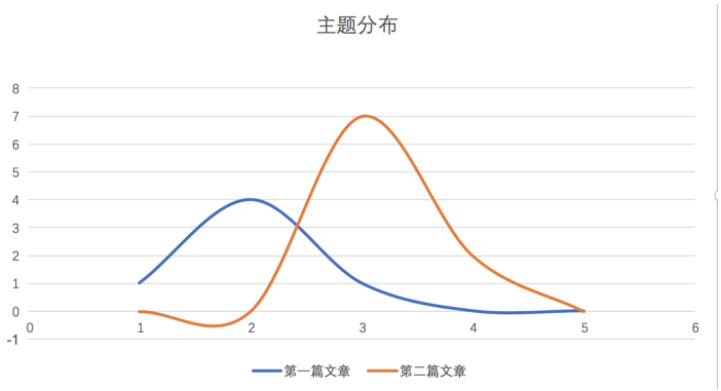


图 13. 文章主题分布-平滑

统计的过程比较简单，对文章进行分词，然后按照主题进行统计，上图可以看出第一篇文章的主题是武侠，第二篇文章的主题是教育，对于平滑后的分布图，可以看出最高点就是文章对应的主题，这里假设每个文章只有一个主题。 而平滑主题分布图中的最高点实际就是对应主题分布的 期望 。对于上面的两张图的横坐标，我采用了两种形式表示，第一幅图是明文的主题，第二幅图是主题的编号，而实际计算机训练时是按照第二副图进行的，它并不知道主题具体是什么。

事情进展的很顺利，看上去主题发现是很简单的问题，只不过在上面的描述中我们有一个假定条件： 假定我们已知每个主题对应的词分布 。如果这个分布已知，事情就会像上面那么简单顺利，因此问题的关键就转换为如何求解这个分布！

[关键点01]

如何求解主题对应的词分布?

5.2.3. 第二次思考

通过第一次的思考，我们终于把要解决的问题想清楚了，没错，就是要求解所有K个主题对应的词分布。
.脑洞大开

- 如何求解所有主题对应的词分布?

一个很大胆的想法就是，如果我们看到的不是每篇文章都写满了词，而是每篇文章中写满了词和对应主题编号，形如：[word:topic]，那么我们就可以统计出每个主题对应的词(统计每个主题下的词频即可)，当然也可以统计出每篇文章的主题分布。
悲剧的是，开始训练时，我们对主题一无所知，只知道主题的个数K，还是拍脑袋定的！！！

那么，再来一个大胆的想法： 能不能我随便给每个词指定一个主题，然后通过不断迭代，最终所有词都可以收敛到它本应该对应的主题上呢?
直觉告诉我们，这是有可能的，就像随机梯度下下降算法，不管我们初始位置选哪里，而且尽管每次我们都随机的挑选一个样本来更新参数，最后仍然可以收敛，关键在于如何定义损失，在这里就是如何找到一个合理的方向让算法迭代到收敛，毕竟每篇文章不是胡乱编写的，它背后是隐藏这一个明确主题的！

[关键点02]
能不能我随便给每个词指定一个主题，然后通过不断迭代，最终所有词都可以收敛到它本应该对应的主题上呢?

如果你之前了解过相关知识，我相信你应该能想到答案了，没错，马尔可夫链的平稳分布就具有这个特点： 不管初始状态是什么，经过有限次的迭代，最终收敛到一个稳定的分布 。我不敢假设所有人都有这个先验的知识，如果你不知道，那就让我来讲个故事，把马氏链引出来吧。

社会学家经常把人按照其经济状态分成3类：下层、中层和上层，我们用1、2、3分别代表这三个阶层。社会学家们发现决定一个人的收入阶层的最重要因素就是其父母的收入阶层。如果一个人的收入属于下层类别，那么他的孩子属于下层输入的概率为0.65， 属于中层收入的概率是0.28，属于上层收入的概率是0.07。事实上，从父代到子代，收入阶层变化的转移概率如下：

		子代		
	State	1	2	3
	1	0.65	0.28	0.07
父代	2	0.15	0.67	0.18
	3	0.12	0.36	0.52

图 14. 收入阶层的转换概率

使用矩阵的表达方式，转换概率矩阵记为：

$$P = \begin{bmatrix} 0.65 & 0.28 & 0.07 \\ 0.15 & 0.67 & 0.18 \\ 0.12 & 0.36 & 0.52 \end{bmatrix}$$

图 15. 转换概率矩阵

假定当前一代人处于下层、中层和上层的人的比例是概率分布向量$\pi_0 = [\pi_0(1), \pi_0(2), \pi_0(3)]$,那么他们子女的分布比例将是$\pi_1 = \pi_0 P$,他们孙子代的分布比例将是$\pi_2 = \pi_1 P = \pi_0 P^2, \ldots,$第$n$代子孙的收入分布比例将是$\pi_n = \pi_{n-1} P = \pi_0 P^n$。

假设初始概率分布为$\pi_0 = [0.21, 0.68, 0.11]$，则我们可以计算前$n$代人的分布状态如下：

第n代人	下层	中层	上层
0	0.210	0.680	0.110
1	0.252	0.554	0.194
2	0.270	0.512	0.218
3	0.278	0.497	0.225
4	0.282	0.490	0.226
5	0.285	0.489	0.225
6	0.286	0.489	0.225
7	0.286	0.489	0.225
8	0.289	0.488	0.225
9	0.286	0.489	0.225
10	0.286	0.489	0.225
...	...	...	...

图 16. 前n代人的分布状态

我们发现从第7代人开始，这个分布就稳定不变了，这个是偶然吗？我们换一个初始概率分布$\pi_0 = [0.75, 0.15, 0.1]$试试看，继续计算前n代人的分布状况如下：

第n代人	下层	中层	上层
0	0.75	0.15	0.1
1	0.522	0.347	0.132
2	0.407	0.426	0.167
3	0.349	0.459	0.192
4	0.318	0.475	0.207
5	0.303	0.482	0.215
6	0.295	0.485	0.220
7	0.291	0.487	0.222
8	0.289	0.488	0.225
9	0.286	0.489	0.225
10	0.286	0.489	0.225
...	...	...	...

图 17. 前n代人的分布状态

我们发现到第9代的时候，分布又收敛了。最奇怪的是，两次给定不同的初始概率分布，最终都收敛到概率分布$\pi = [0.286, 0.489, 0.225]$，也就是说 收敛的行为和初始概率分布无关 。走到这一步，你一定会惊叹：
发现上帝了，我们的问题可以求解了!

5.2.3.1. 一个重大的发现

为了突出惊叹，我们就另起一个小节吧，先把重大的发现记录下来：
概率分布收敛的行为和初始概率分布无关
继续聊上面的例子，如果最终的分布同初始分布无关，那就说明主要是由状态转移矩阵P决定的，让我们计算下P^n

$$P^{20} = P^{21} = \cdots = P^{100} = \cdots = \begin{bmatrix} 0.286 & 0.489 & 0.225 \\ 0.286 & 0.489 & 0.225 \\ 0.286 & 0.489 & 0.225 \end{bmatrix}$$

我们发现当n足够大的时候，这个p^n矩阵的每一行都稳定的收敛到$\pi = [0.286, 0.489, 0.225]$这个概率分布。自然的这个收敛现象并不是马氏链独有的， 而是绝大多数马氏链的共同行为，关于马氏链的收敛性我们有个漂亮的定理。

在继续下去之前，我们需要重新更新下我们的问题：**找到一个转移矩阵，使得我们随机指定每个词的主题，经过n轮迭代，最终所有词的主题分布会收敛到稳定分布，即合理分布**。

定理01: 如果一个非周期马氏链具有转移概率矩阵P,且它的任何两个状态都是联通的，那么$\lim_{n\to\infty} P_{ij}^n$存在且与$i$无关，记$\lim_{n\to\infty} P_{ij}^n = \pi(j)$,我们有：

- $\lim_{n\to\infty} P^n = [\pi_1 \pi_2 \ \dots \ \pi_j \ \dots \ ]$
- $\pi(j) = \sum_{i=0}^{\infty} \pi(i) P_{ij}$
- π是方程$\pi P = \pi$唯一非负解,其中$\pi = [\pi_1, \pi_2, \dots, \pi_j, \dots]$，$\sum_{i=0}^{\infty} \pi(i) = 1$

则π称为马氏链的平稳分布。
这个马氏链的收敛定理非常重要， 所有的MCMC(Markov Chain Monte Carlo)方法都是以这个定理作为理论基础的 。 定理的证明比较复杂，一般的随机过程的课本中也不给出证明， 我们就不纠结于此了，直接用这个定理就好了。下面对于这个定理的内容做一些解释说明：

- 该定理中马氏链的状态不要求有限，可以是无穷多个
- 定理中的 非周期 概念，我们不打算解释，因为我们遇到的绝大多数马氏链都是非周期的

- 两个状态i,j是联通的，并不是指i可以一步就转移到j，而是指有限步联通，马氏链的任意两个状态是联通的含义是指存在一个n，使得矩阵P^n中任何一个元素的数值都大于零。
- 由于马氏链的收敛行为，假定n步后收敛，则$x_n, x_{n+1}, \dots$都是平稳分布π_x的样本。

上面这些其实都不重要，重要的是你要想到： 我们的目标是一个概率分布，如果我们可以构造一个转移矩阵，使得马氏链的平稳分布刚好就是求解的分布，那么我们从任何一个初始状态出发，沿着马氏链转移，如果第n步收敛，则我们就得到了所求分布对应的样本 。

这个绝妙的想法在1953年被 Metropolis 想到了， Metropolis 考虑了物理学中常见的玻尔兹曼分布的采样问题，首次提出了基于马氏链的蒙特卡洛方法，即 Metropolis 算法,Metropolis 算法是一个普适的采样方法，并启发了一系列 MCMC 方法， 所以人们把它视为随机模拟技术腾飞的起点。 Metropolis 算法也被选为二十世纪十大最重要的算法之一。

5.2.3.2. Metropolis Hastings算法

收下我们的小心思，想想我们的问题走到哪里了：
我们最初是要求解每篇文章主题的概率分布，可以通过统计每个词对应主题的个数来近似估计，但是每个词对应主题我们也不知道，于是我们希望随便给每个词指定一个主题，通过迭代收敛到稳定的分布，即每个词应该对应的主题编号上，后来我们神奇地发现马氏链的平稳分布的性质可以应用于我们的问题求解中，关键在于如何获得这个转移矩阵！

好，略轻思路，我们继续，我们目前关注的问题仍然是如何获得这个神奇的 转移矩阵 。

接下来我们要介绍的 MCMC 算法是 Metropolis 算法的一个改进变种，即常用的 Metropolis Hastings 算法。由上节的例子和定理我们看到了，马氏链的收敛性质主要是由转移矩阵 P 决定的， 所以基于马氏链做采样的关键问题是如何构造转移矩阵 P ，使得平稳分布刚好是我们想要的分布$p(x)$。如何做到这一点呢？我们主要用到下面的定理。

定理02(细致平稳条件) 如果非周期马氏链的转移矩阵 P 和分布$\pi(x)$满足：

$$\pi(x)P_{ij} = \pi_j P_j \qquad for\ all\ i,j \qquad (5.1)$$

则π_x是马氏链的平稳分布，上式被称为 细致平稳条件(detail balance condition) 。证明也非常简洁:

$$\sum_{i=0}^{\infty}\pi(i)P_{ij} = \sum_{i=0}^{\infty}\pi(j)P_{ji} = \pi(j)\sum_{i=0}^{\infty}P_{ji} = \pi(j)$$
$$\Rightarrow \pi P = \pi$$

假设我们已经有一个转移矩阵为 Q 马氏链($p(i,j)$表示从状态i转移为状态j的概率)，显然通常情况下

$$p(i)q(i,j) \neq p(j)q(j,i)$$

也就是细致平稳条件不成立，所以$p(x)$不太可能是这个马氏链的平稳分布。我们可否对马氏链做一个改造，使得细致平稳条件成立呢？譬如我们引入$\alpha(i,j)$，我们希望：

$$p(i)q(i,j)\alpha(i,j) = p(j)q(j,i)\alpha(j,i) \qquad (5.3)$$

取什么样的$\alpha(i,j)$以上等式能成立呢？最简单的，按照对称性，我们可以取：

$$\alpha(i,j) = p(j)q(j,i) \qquad \alpha(j,i) = p(i)q(i,j)$$

于是公式(5.3)就成立了，所以有

$$p(i)\underbrace{q(i,j)\alpha(i,j)}_{Q'(i,j)} = p(j)\underbrace{q(j,i)\alpha(j,i)}_{Q'(j,i)} \qquad (5.4)$$

于是我们就把原来具有转移矩阵Q的一个普通的马氏链，改造成了具有转移矩阵Q'的马氏链，而 Q' 恰好满足细致平稳条件，由此马氏链Q'的平稳分布就是$p(x)$！

暂停一下，让我们来思考两个问题

- 我们为何要找满足细致平稳条件的转移矩阵?
- p(x)我们并没有改变，为何改变转移矩阵后就成了平稳分布了?

第二个问题比较容易，因为平稳分布就是相对于转移矩阵的，不管p(x)初始状态是什么，转移矩阵都是使得p(x)最终收敛到平稳分布。 那为何p(x)就是平稳分布呢?

这个问题和第一个问题是等价的，首先我们的目的是为了找到转移矩阵使得不论初始状态为何，都可以最后收敛到稳定分布。首先，这个转移矩阵不好找；其次，这个转移矩阵不止一个。 因此我们只需要找到一个就可以了，我们寻找的思路是从平稳分布和转移矩阵的确定关系出发，发现所有的平稳分布和转移矩阵都满足细致平稳条件，因此我们只要找到满足细致平稳条件的分布和转移矩阵， 那么这个分布就是平稳分布，因为满足平稳分布的定义。

在改造Q的过程中，我们引入了$\alpha(i,j)$称为接受率，物理意义可以理解为在原来的马氏链上，从状态i以$q(i,j)$的概率跳转到状态j的时候，我们以$\alpha(i,j)$的概率接受这个转移， 于是得到新的马氏链Q' 的转移矩阵为$q(i,j)\alpha(i,j)$。

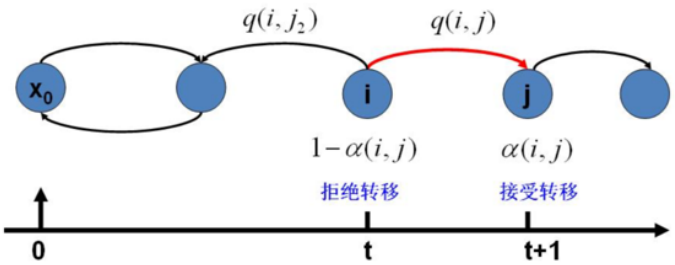


图 18. 马氏链转移和接受概率

把以上的过程整理一下，我们就可以得到如下的用于采样的概率分布$p(x)$的算法:

Algorithm 5 MCMC 采样算法

1: 初始化马氏链初始状态$X_0 = x_0$

2: 对$t = 0, 1, 2, \cdots$, 循环以下过程进行采样

- 第t个时刻马氏链状态为$X_t = x_t$, 采样$y \sim q(x|x_t)$
- 从均匀分布采样$u \sim Uniform[0,1]$
- 如果$u < \alpha(x_t, y) = p(y)q(x_t|y)$ 则接受转移$x_t \to y$，即$X_{t+1} = y$
- 否则不接受转移，即$X_{t+1} = x_t$

图 19. MCMC采样算法

以上过程不仅适应于离散的情形，对于分布是连续的，以上算法仍然有效。以上的 MCMC 算法已经能很漂亮的工作了，不过它有一个小问题：马氏链在状态转移过程中的接受率$\alpha(i,j)$可能偏小， 这样采样过程中马氏链容易原地踏步，拒绝大量的跳转，这使得马氏链 遍历所有的状态空间 要花费太长的时间，收敛到平稳分布$p(x)$的速度太慢，有没有办法提高接受率呢?

可以假设我们的接受率$\alpha(i,j) = 0.1, \alpha(j,i) = 0.2$，此时满足细致平稳条件，于是:

$$p(i)q(i,j) \times 0.1 = p(j)q(j,i) \times 0.2$$

上式两边扩大五倍，可以改写为:

$$p(i)q(i,j) \times 0.5 = p(j)q(j,i) \times 1$$

看，我们提高了接受率，而细致平稳条件并没有打破！这启发我们可以把细致平稳条件中的接受率同比例放大，使得两个数中的最大一个数放大到1，这样我们就提高了采样的跳转接受率，所以我们可以取

$$\alpha(i,j) = min\left\{\frac{p(j)q(j,i)}{p(j)q(i,j)}, 1\right\}$$

经过如上改动，我们就得到了最常见的 Metropolis-Hastings 算法，算法伪代码如下：

Algorithm 6 Metropolis-Hastings 采样算法

1: 初始化马氏链初始状态$X_0 = x_0$

2: 对$t = 0, 1, 2, \cdots$, 循环以下过程进行采样

- 第t个时刻马氏链状态为$X_t = x_t$, 采样$y \sim q(x|x_t)$
- 从均匀分布采样$u \sim Uniform[0, 1]$
- 如果$u < \alpha(x_t, y) = \min\left\{\frac{p(y)q(x_t|y)}{p(x_t)p(y|x_t)}, 1\right\}$则接受转移$x_t \rightarrow y$，即$X_{t+1} = y$
- 否则不接受转移，即$X_{t+1} = x_t$

图 20. Metropolis-Hastings算法

至此，我们已经得到了一个解决方案，即不论我们给文档中的每个词初始化哪个主题编号，只要找到转移矩阵，我们都可以在迭代有限步后收敛到主题和词的稳定分布，伟大的 Metropolis-Hastings 算法就是我们的救世主， 而且它还告诉我们如何选取这样的转移矩阵，只是有一点瑕疵，这个转移矩阵虽然进行了优化，接受率仍然是个概率值，如果接受率为100%，那该多好，算法的收敛速度将达到最快。

真的还可以再优化吗？ 科学家们为了发表论文可真没闲着，因为100%接受率的采样方法真的找到了，掌声欢迎 Gibbs Sampling 算法华丽登场 。

5.2.3.3. Gibbs Sampling 算法

Metropolis-Hastings 算法由于存在接受率的问题，因此对于高维空间的采样效率并不高，能否找到一个转移矩阵Q使得接受率$alpha = 1$呢?

首先看下二维情况，假设有一个概率分布$p(x, y)$，考察x坐标相同的两个点$A(x_1, y_1)$，$B(x_2, y_1)$，我们发现：

$$p(x_1, y_1)p(y_2|x_1) = p(x_1)p(y_1|x_1)p(y_2|x_1)$$
$$p(x_1, y_2)p(y_1|x_1) = p(x_1)p(y_2|x_1)p(y_1|x_1)$$

上面两公式相等，所以我们得到

$$p(x_1, y_1)p(y_2|x_1) = p(x_1, y_2)p(y_1|x_1) \tag{5.4}$$

即

$$p(A)p(y_2|x_1) = p(B)p(y_1|x_1)$$

基于以上等式，我们发现在$x = x_1$这条平行于y轴的直线上，如果使用条件分布$p(y|x_1)$作为任何两个点之间的转移概率，那么任意两个点之间的转移满足细致平稳条件。 同样的，如果我们在$y = y_1$这条直线上任意取两个点$A(x_1, y_1)$，$C(x_2, y_1)$，也有如下等式：

$$p(A)p(x_2|y_1) = p(C)p(x_1|y_1)$$

于是我们可以如下构造平面上任意两点的之间的转移概率矩阵Q：

$$Q(A \rightarrow B) = p(y_B|x_1) \quad if \quad x_A = x_B = x_1$$
$$Q(A \rightarrow C) = p(y_C|x_1) \quad if \quad x_A = x_C = x_1$$
$$Q(A \rightarrow D) = 0 \quad other$$

有了如上的转移矩阵Q，我们很容易验证对于平面上的任意两点X,Y，满足细致平稳条件：

$$p(X)Q(X \rightarrow Y) = p(Y)Q(Y \rightarrow X)$$

于是，这个二维空间的马氏链收敛到平稳分布$p(x,y)$，而这个算法就是 Gibbs Sampling 算法，由物理学家 Gibbs 首次提出。

Algorithm 7 二维Gibbs Sampling 算法

1: 随机初始化$X_0 = x_0 Y_0 = y_0$

2: 对$t = 0, 1, 2, \cdots$ 循环采样

1. $y_{t+1} \sim p(y|x_t)$
2. $x_{t+1} \sim p(x|y_{t+1})$

图 21. Gibbs Sampling算法

如图所示，马氏链的转移只是轮换的沿着坐标轴做转移，于是得到样本$(x_0,y_0),(x_0,y_1),(x_1,y_1),(x_1,y_2),\dots$， 马氏链收敛后，最终得到的样本就是$p(x,y)$的样本。补充说明下，教科书上的 Gibbs Sampling 算法大都是坐标轮转算法，但其实这不是强制要求的。最一般的情况是， 在任意t时刻，可以在x轴和y轴之间随机的选一个坐标轴，然后按照条件概率做转移，马氏链也是一样收敛的。轮换两个坐标只是一种方便的形式。

以上的过程，我们很容易推广到高维的情况，对于n维空间，概率分布$p(x_1,x_2,x_3,\dots,x_n)$可以如下定义转移矩阵：

- 如果当前状态为$x_1,x_2,\dots,x_n$，马氏链转移的过程中，只能沿着坐标轴做转移。沿着x_i这根坐标轴做转移的时候，转移概率由条件概率$p(x_i|x_1,\dots,x_{i-1},x_{i+1},..,x_n)$定义；
- 其他无法沿着单根坐标轴进行的跳转，转移概率都设置为0。

于是，我们可以把二维的 Gibbs Sampling 算法从采样二维的$p(x,y)$推广到采样n维的$p(x_1,x_2,\dots,x_n)$。以上算法收敛后，得到的就是概率分布$p(x_1,x_2,\dots,x_n)$的样本， 在通常的算法实现中，坐标轮转都是一个确定性的过程，也就是说在给定时刻t，在一根固定的坐标轴上转移的概率是1. 高维算法的伪代码如下：

Algorithm 8 n维Gibbs Sampling 算法

1: 随机初始化$\{x_i : i = 1, \cdots, n\}$

2: 对$t = 0, 1, 2, \cdots$ 循环采样

1. $x_1^{(t+1)} \sim p(x_1|x_2^{(t)}, x_3^{(t)}, \cdots, x_n^{(t)})$
2. $x_2^{(t+1)} \sim p(x_2|x_1^{(t+1)}, x_3^{(t)}, \cdots, x_n^{(t)})$
3. $\cdots$
4. $x_j^{(t+1)} \sim p(x_j|x_1^{(t+1)}, \cdots, x_{j-1}^{(t+1)}, x_{j+1}^{(t)}, \cdots, x_n^{(t)})$
5. $\cdots$
6. $x_n^{(t+1)} \sim p(x_n|x_1^{(t+1)}, x_2^{t}, \cdots, x_{n-1}^{(t+1)})$

图 22. n维Gibbs Sampling算法

5.2.4. 第三次思考

看到这里希望大家的思路还是清晰的，让我们再一起思考下：

示例 6. 脑洞大开

- 我们已经知道通过坐标轮转的方式，即 Gibbs Sampling 算法，可以让我们开始时任意指定文章中任意词对应的主题(即主题对应的词分布),经过迭代都可以收敛到平稳状态，也就是得到我们想要的topic-word分布
- 那么，具体到我们的问题中，该如何应用 坐标轮转大法，是几维空间，坐标轴是什么？如何做到只沿着一个坐标轴轮转？

在自然语言处理中，我们经常将词映射到高维空间，因此这里一个很自然的想法就是：同样将词作为高维空间的维度，我们迭代第i个词时，坐标轮转法就意味着：在迭代过程中，固定当前词不变， 考虑条件概率分布$p(z_i = k|\vec{z}_{\neg i}, \vec{w})$,这个条件概率的含义是在已知除了第$i$个词意外所有词的主题分布和可观察到的所有词的前提下，第i个词等于第k个主题的概率，公式中有一个符

号：¬ 表示逻辑关系 非 。

理解上面的思维转换过程是非常重要的！！！ 其实大家只要对照前面讲的Gibbs采样的样本结果还是很容易理解的，每次更新一个坐标，保持其他坐标轴值不变，也就是每次只更新一个词的主题编号，条件是已知其他词的主题编号。

如果知道了条件概率分布$p(z_i = k|\vec{z}_{\neg i}, \vec{w})$，只需要对每个词按照概率抽样对应的主题编号就可以了。问题终于抽象为了一个数学问题，即求解公式(5.5)的值：

$$p(z_i = k|\vec{z}_{\neg i}, \vec{w}) \qquad (5.5)$$

似乎没什么思路，一个大胆的想法就是既然条件概率不好求，是不是可以求出联合概率分布，然后利用贝叶斯公式间接求解？ 这个方法很老套，但不妨试试。既然要计算联合概率分布，首先想到的是 概率图模型 ，把这些变量之间的关系都画出来,瞬间写出联合概率分布.

5.3. 文本建模

关于上节的结论，我还想再多说一句，其实大家从很自然角度去想公式(5.5)，你会发现，它其实就是在说：文本中的某个词的主题分布，是由它周围(或者说文档中其他)词共同决定的，这个很好理解，毕竟文档中所有的词都是在描述同样的事情。既然这个概率公式的解释如此自然， 我们就从文本自然产生的过程出发，来考虑其联合概率分布。

5.3.1. 文档是如何产生的

我们可以看看日常生活中的人是如何构思文章的。如果我们要写一篇文章，往往是要先确定写哪几个主题。譬如构思一篇自然语言处理相关的文章，可能40%会谈论语言学、30%谈论概率统计、20%谈论计算机、还有10%谈论其他主题：

- 说到语言学，我们容易想到的词汇：语法、句子、乔姆斯基、主语...
- 谈到概率统计，我们容易想到一下词：概率、模型、均值、方差、证明、独立、马尔可夫链...
- 谈论计算机，我们容易想到的词是：内存、硬盘、编程、二进制、对象、算法、复杂度...

我们之所以立刻想到这些词，是因为这些词在对应的主题下出现的概率很高。我们可以很自然的看到，一篇文章通常是由多个主题构成，而每一个主题大概可以用与该主题相关的频率高的一些词来描述。

以上的直观想法由 `Hoffmn` 于1999年给出的 `PLSA(Probabilistic Latent Semantic Analysis)` 模型中首先进行了明确的数学化， `Hoffmn` 认为一篇文档可以有多个主题$Topic$混合而成，而每个主题都是词汇上的概率分布， 文章中的每个词都是由一个固定的$Topic$生成的。

5.3.2. PLSA模型

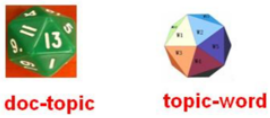
统计学被人们描述为猜测上帝的游戏，人类产生的说有的预料文本我们都可以看成是一个伟大的上帝在天堂中抛掷骰子生成的，我们观察到的都是上帝玩这个游戏的结果，所以在文本建模中，我们希望猜测出上帝是如何玩这个游戏的， 具体一点，最核心的两个问题是：

- 上帝都有什么样的骰子
- 上帝是如何抛掷这些骰子的

`Hoffmn` 认为上帝是按照如下的游戏规则来生成文本的。

Game 11 PLSA Topic Model

1: 上帝有两种类型的骰子，一类是doc-topic 骰子,每个doc-topic 骰子有K 个面，每个面是一个topic 的编号；一类是topic-word 骰子，每个topic-word 骰子有V 个面，每个面对应一个词；



2: 上帝一共有K 个topic-word 骰子, 每个骰子有一个编号，编号从1 到K；

3: 生成每篇文档之前，上帝都先为这篇文章制造一个特定的doc-topic 骰子，然后重复如下过程生成文档中的词

- 投掷这个doc-topic 骰子,得到一个topic 编号z
- 选择K 个topic-word 骰子中编号为z的那个，投掷这个骰子，于是得到一个词

图 23. 游戏：PLSA主题模型

以上 PLSA 模型的文档生成的过程可以图形化的表示为：

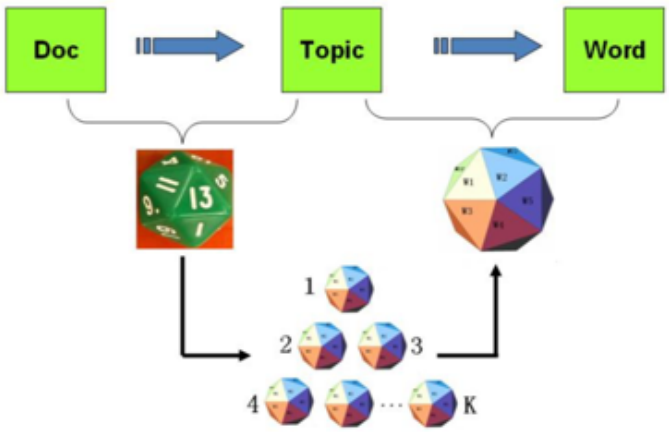


图 24. PLSA模型的文档生成过程

我们可以发现，按照上面的游戏规则，文档和文档之间是独立可交换的，同一个文档内的词也是独立可交换的，这是一个典型的词袋 bag-of-words 模型。 游戏中的K个 topic-word 骰子，我们可以记为$\overrightarrow{\varphi_1}, \dots \overrightarrow{\varphi_K}$, 对于包含$M$篇文档的预料$C = (d_1, d_2, \dots , d_M)$中的每篇文档$d_m$，都会有一个特定的 doc-topic 骰子$\overrightarrow{\theta_m}$, 所以对应的骰子记为$\overrightarrow{\varphi_1}, \dots \overrightarrow{\varphi_M}$,为了方便，我们假设每个词$w$都是一个编号，对应到 topic-word 骰子的面。 于是在 PLSA 模型中，第m篇文档d_m中的每个词的生成概率为：

$$p(w|d_m) = \sum_{z=1}^{K} p(w|z)p(z|d_m) = \sum_{z=1}^{K} \varphi_{zw}\theta_{mz} \qquad (5.6)$$

所以整篇文章的生成概率为：

$$p(\vec{w}|d_m) = \prod_{i=1}^{n}\sum_{z=1}^{K} p(w_i|z)p(z|d_m) = \prod_{i=1}^{n}\sum_{z=1}^{K} \varphi_{zwi}\theta_{mz} \qquad (5.7)$$

由于文本之间相互独立，我们也容易写出整个预料的生成概率。求解 PLSA 这个主题模型的过程中，模型参数容易求解，可以使用著名的 EM 算法进行求得局部最优解，由于该模型的解并不是本章节的重点， 有兴趣的同学可以参考 Hoffmn 的原始论文，此处略。

5.3.3. LDA模型

对于 PLSA 模型，贝叶斯学派显然是有意见的， doc-topic 骰子$\overrightarrow{\theta_m}$和 topic-word 骰子$\overrightarrow{\varphi_k}$都是模型中的参数，参数都是随机变量，怎么能没有先验分布呢？ 于是对于 PLSA 模型的贝叶斯改造，我们可以在如下两个骰子参数前加上先验概率。

应该增加什么样的先验分布呢？ 假定先验概率为X分布，在似然函数为多项分布的情况下，观察到一定的数据后，得到的后验概率Y,然后后验概率Y会作为下一次观察前的先验概率， 因此我们希望先验概率分布和后验概率分布相同，即$X = Y$，在似然函数为多项分布的情况下，什么样的分布满足先验分布和后验分布相同呢?

5.3.3.1. 二项分布

在数学上定义，在指定似然函数下，先验分布和后验分布相同时，先验分布就叫做似然函数的共轭分布，一个容易让人误解的概念就是：共轭先验是先验概率相对于似然函数而言的。 那么究竟什么分布和多项分布共轭?

问题简化下，我们知道多项分布是二项分布在高维度上的推广，我们先研究二项分布的共轭先验分布！

假定上帝和我们做一个游戏：游戏的规则很简单，上帝由一个魔盒，上面有个按钮，你每按一下按钮，就会均匀的输出一个[0~1]之间的随机数，上帝现在按了10下，手上有10个数， 你猜第7大的数是什么。你该如何猜呢?

从数学角度描述这个游戏如下：

Algorithm 1 游戏1

1: $X_1, X_2, \cdots, X_n \overset{iid}{\sim} Uniform(0,1)$,
2: 把这n 个随机变量排序后得到顺序统计量$X_{(1)}, X_{(2)} \cdots, X_{(n)}$,
3: 问$X_{(k)}$ 的分布是什么

图 25. 游戏1

对于上面的游戏而言$n = 10, k = 7$，如果我们能求出$X_{(7)}$的分布的概率密度，那么用概率密度的极值点去做猜测就是最好的策略。那么对于一般的情形，$X_{(k)}$的分布是什么呢？ 我们尝试计算$X_{(k)}$在一个区间$[x, x + \Delta x]$的概率，也就是如下的概率值：

$$p(x \leqslant X_{(k)} \leqslant x + \Delta x) = ?$$

把[0,1]区间分成三段$[0, x), [x, x + \Delta x], (x + \Delta x, 1]$，我们先考虑简单的情形，假定n个数字中只有一个落在区间$[x, x + \Delta x]$内，则因为该区间是第k大的， 则$[0, x)$应该有$k - 1$个数，$(x + \Delta x, 1]$之间应该有$n - k$个数，我们将符合上述要求的事件记为E：

$$\begin{aligned} E = \{ & X_1 \in [x, x + \Delta x], \\ & X_i \in [0, x) \quad (i = 2, \cdots, k), \\ & X_j \in (x + \Delta x, 1] \quad (j = k + 1, \cdots, n)\} \end{aligned}$$

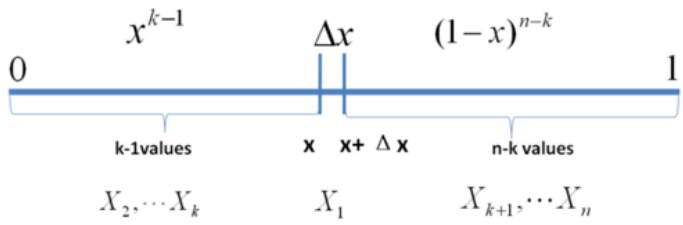


图 26. 事件E

则有：

$$\begin{aligned} P(E) &= \prod_{i=1}^{n} P(X_i) \\ &= x^{k-1}(1 - x - \Delta x)^{n-k} \Delta x \\ &= x^{k-1}(1 - x)^{n-k} \Delta x + o(\Delta x) \end{aligned}$$

其中$o(\Delta x)$表示Δx的高阶无穷小。显然，由于不同的排列组合，即n个数中有一个数落在$[x, x+\Delta x]$区间的有n种取法， 余下$n-1$个数落在$[0, x)$的有C_{n-1}^{k-1}种组合，所以和事件E等价的事件一共有nC_{n-1}^{k-1}个。

刚才我们假定了在$[x, x+\Delta x]$区间只有一个数的情况，再考虑稍微复杂点的情况，假设有两个数落在了区间$[x, x+\Delta x]$,事件记为$E^{'}$,则有

$$P(E^{'}) = x^{k-2}(1-x-\Delta x)^{n-k}(\Delta x)^2 = o(\Delta x)$$

从以上的分析可以很容易看出，只要落在$[x, x+\Delta x]$内的数字超过一个，则对应事件的概率就是Δx，于是：

$$\begin{aligned} P(x \le X_{(k)} \le x+\Delta x) &= nC_{n-1}^{k-1}P(E) + o(\Delta x) \\ &= nC_{n-1}^{k-1}x^{k-1}(1-x)^{n-k}\Delta x + o(\Delta x) \end{aligned}$$

所以，可以得到$X_{(k)}$的概率密度函数为：

$$\begin{aligned} f(x) &= \lim_{\Delta x \to 0} \frac{P(x \le X_{(k)} \le x+\Delta x)}{\Delta x} \\ &= nC_{n-1}^{k-1}x^{k-1}(1-x)^{n-k} \\ &= \frac{n!}{(k-1)!(n-k)!}x^{k-1}(1-x)^{n-k} \quad x \in [0,1] \end{aligned}$$

这面的公式中有阶乘，因此可以用 `Gamma` 函数来表示，我们可以把$f(x)$表达为：

$$f(x) = \frac{\Gamma(n+1)}{\Gamma(k)\Gamma(n-k+1)}x^{k-1}(1-x)^{n-k} \tag{5.8}$$

考虑到可能有同学不了解 `Gamma` 函数，我们接下来会补充介绍一下，了解的同学可以跳过下面一个小节。

5.3.3.2. 神奇的 `Gamma` 函数

学高数的时候，我们都学过如下一个长相有点奇特的 `Gamma` 函数

$$\Gamma(x) = \int_0^{\infty} t^{x-1}e^{-t}dt$$

通过分布积分，我们可以求得

$$\begin{aligned} \Gamma(x) &= \int_0^{\infty} t^{x-1}e^{-t}dt \\ &= \frac{1}{x}\int_0^{\infty} e^{-t}dt^x \\ &= \frac{1}{x}[(e^{-t}t^x)\,|_0^{\infty} - \int_0^{\infty} t^x de^{-t}] \\ &= \frac{1}{x}[0 - e^{-t}(-1)\int_0^{\infty} t^x e^{-t}dt] \\ &= \frac{1}{x}\int_0^{\infty} t^x e^{-t}dt \\ &= \frac{1}{x}\Gamma(x+1) \end{aligned}$$

所以推导出了如下的递归性质

$$\Gamma(x+1) = x\Gamma(x)$$

看到上式，你一定会想到 斐波那契数列 ，它可以表示为数的阶乘的形式！于是很容易证明， Gamma 函数可以当成是阶乘在实数集上的扩展， 具有如下性质：

$$\Gamma(n) = (n-1)!$$

5.3.3.3. Beta 函数

让我们再回到公式(5.8)，我们令$\alpha = k, \beta = n-k+1$ ，于是我们得到

$$f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} \qquad (5.9)$$

这个函数，就是一般意义上的 Beta 分布，上面的例子中$n = 10, k = 7$，所以我们按照如下密度分布的峰值去猜测才是最有把握的。

$$f(x) = \frac{10!}{6!3!} x^6(1-x)^3 \quad x \in [0,1]$$

然而即便如此，我们能做到一次猜中的概率仍然很低，因为我们掌握到的信息量是在是太少了，如果上帝仁慈了点，告诉了我们一些信息， 那么我的对 手头上第7大的数是多少 这件事会得到怎样的后验概率分布呢？ 首先已知的是：在没有任何知识的情况下，这个概率分布为 Beta 分布.

$$\text{先验分布} + \text{数据的知识} = \text{后验分布}(?)$$

以上是贝叶斯分析过程的简单直观的表述，假定上帝给我们的信息为：**上帝按了5下这个机器，你就得到了5个[0,1]之间的随机数，然后上帝告诉你这5个数中的每个数，以及和第7个数相比，谁大谁小，然后再让你猜上帝手头上第7大的数是多少。**

上帝的第二个游戏，数学上形式化一下，就是

Algorithm 2 游戏2

1: $X_1, X_2, \cdots, X_n \overset{iid}{\sim} Uniform(0,1)$，排序后对应的顺序统计量$X_{(1)}, X_{(2)} \cdots, X_{(n)}$，我们要猜测$p = X_{(k)}$；
2: $Y_1, Y_2, \cdots, Y_m \overset{iid}{\sim} Uniform(0,1)$，$Y_i$中有$m_1$个比$p$小，$m_2$个比$p$大；
3: 问$P(p|Y_1, Y_2, \cdots, Y_m)$ 的分布是什么。

图 27. 游戏2

我看网上很多人质疑 Richjin 给出的这几个例子，我觉得质疑者多半是没有看懂这些例子背后的含义。让我们仔细阅读上面的游戏规则， 实际上这个就是一个典型的探索先验分布、似然函数以及后验分布关系的例子，千万别忘了走到这一步我们的目的是什么： 在似然函数为多项分布的时候，什么样的分布满足先验分布和后验分布共轭，为了简化问题，我们将多项分布退化到二项分布来研究这个问题。

好，上面又啰嗦的梳理了下我们的思路，让我们继续考虑第2个游戏。由于$p = X_{(k)}$在$X_1, X_2, \ldots, X_n$中是第k大的数，利用Y_i的信息， 我们很容易得到p在$X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_m$ 这$(m+n)$个独立同分布的随机变量中是第$k+m_1$ 个，于是按照第1个游戏的方式，我们可以得到$p = X_k$的概率密度函数是

$$Beta(p|k+m_1, n-k+1+m_2)$$

讲到这里，我们把以上的过程再整理如下：

- $p = X_k$是我们猜测的参数，我们推导出p的分布为$f(p) = Beta(p|k, n-k+1)$ 称为p的先验分布；
- 数据Y_i中有m_1个比p大，m_2个比p小，Y_i相当于做了m次伯努利实验，所以m_i服从二项分布$B(m,p)$;
- 在给定来自数据提供的(m_1, m_2)的知识后，p的后验分布变为了$f(p|m_1, m_2) = Beta(p|k+m_1, n-k+1+m_2)$

以上总结可以写成如下公式

$$Beta(p|k, n-k+1) + BinomCount(m_1, m_2) = Beta(p|k+m_1, n-k+1+m_2)$$

其中(m_1, m_2)对应的是二项分布$B(m_1+m_2, p)$的计数，更为一般的，对于非负实数α, β，我们有如下关系

$$Beta(p|\alpha, \beta) = BinomCount(m_1, m_2) = Beta(p|\alpha+m_1, \beta+m+2)$$

这个式子描述的就是 `Beta-Binomial共轭` ，此处共轭的意思就是：*数据符合二项分布的时候，参数的先验分布和后验分布都能保持 `Beta` 分布的形式*。

而我们从以上过程可以看到， `Beta` 分布的参数α, β都可以理解为物理计数，这两个参数经常被称为伪计数$(pseudo-count)$，基于以上逻辑， 我们也可以把 `Beta` 分布写成如下的公式来理解

$$Beta(p|1, 1) + BinomCount(\alpha-1, \beta-1) = Beta(p|\alpha, \beta) \qquad (5.10)$$

其中 `Beta(p|1,1)` 恰好就是均匀分布 `Uniform(0, 1)` 。

对于公式(5.10)，我们其实可以从贝叶斯的角度来进行推导和理解。假设有一个不均匀的硬币抛出正面的概率p，抛m次后出现正面和反面的次数分别为m_1, m_2， 那么按传统的频率学派观点，p的估计值应该为$\hat{p} = \frac{m1}{m1+m2}$,而从贝叶斯学派角度来看，开始对硬币的不均匀性一无所知，所以应该假设 $p\ Uniform(0, 1)$，于是有了二项分布的计数(m_{1}, m_{2})之后，按照贝叶斯公式计算p的后验概率

$$\begin{aligned} P(p|m_1, m_2) &= \frac{P(p) \cdot P(m_1, m_2|p)}{P(m_1, m_2)} \\ &= \frac{1 \cdot P(m_1, m_2|p)}{\int_0^1 P(m_1, m_2|t)dt} \\ &= \frac{C_m^{m1} p^{m1}(1-p)^{m2}}{\int_0^1 C_m^{m1} t^{m1}(1-t)^{m2}dt} \\ &= \frac{p^{m1}(1-p)^{m2}}{\int_0^1 t^{m1}(1-t)^{m2}dt} \end{aligned}$$

计算得到的后验概率正好是$Beta(p|m_1+1, m_2+1)$。

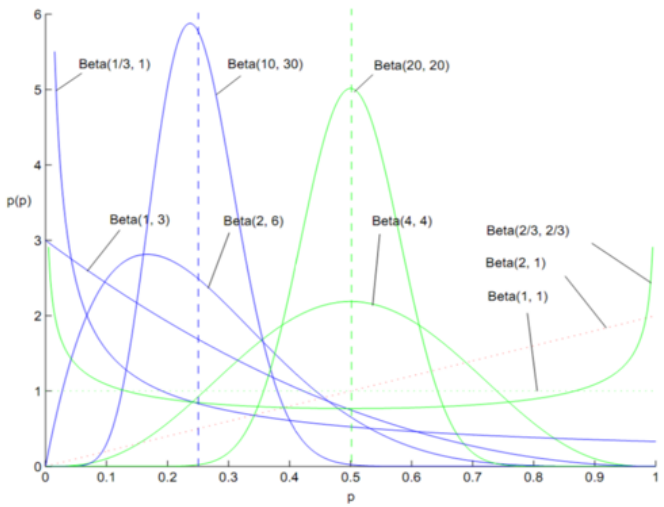


图 28. 百变星君Beta分布

`Beta` 分布的概率密度我们把它画成图，会发现它是百变星君，而均匀分布也是特殊的 `Beta` 分布，由于 `Beta` 分布能够拟合如此之多的形状，因此它在统计数据拟合和贝叶斯分析中被广泛使用。

5.3.3.4. 第四次思考

这一次大家不用脑洞大开，我们一起来思考下面几个问题。

核心概念

1. 如何更直观的理解伪计数，因为似乎后验概率相对于先验概率就是改了这个东西而已
2. 如何直观理解似然函数

首先，举例说明什么是似然函数，以二项分布的最大似然估计为例，假设投硬币实验中，进行了N次独立试验，n次朝上，$N-n$次朝下，假设朝上的概率为p，请使用最大似然估计这个参数p。

在这个N次的独立试验中，结果是服从参数为p的二项分布，因此对参数的求解，可以使用最大似然函数来做点估计，使用对数似然函数作为目标函数

$$
\begin{aligned}
f(n|p) &= log(p^n(1-p)^{N-n}) \\
h(p) &= log(p^n(1-p)^{N-n}) \qquad (5.11) \\
\frac{\partial h(p)}{\partial p} &= \frac{n}{p} - \frac{N-n}{1-p} = 0 \Rightarrow p = \frac{n}{N}
\end{aligned}
$$

公式(5.11)是一个关于参数的函数，就是似然函数，令似然函数导数为0，求出的结果就是参数的点估计。

这种估计方式是典型的频率学派的思想，结果全部由观察到的数据决定，也种方法在观察数据较少时，往往会有很大问题。 比如我们就某学校门口，通过观察通过校门的男女生人数来估计学校里男女比例。 如果观察到2两个都是女生，按照上面的计算公式得到的结论就会是学校里女生占比100%，这显然是不对的，那我们就来思考这个问题。

假设观察到的男生和女生的个数分别为$N_B = 1$和$N_G = 4$，频率学派的计算公式就是

$$
P_B = \frac{N_B}{N_B + N_G} = 0.2 \qquad P_G = \frac{N_G}{N_B + N_G} = 0.8
$$

如果修正这个公式，使得在观察学生的数量较少时仍然不会太离谱，我们假定男生和女生的比例为$1:1$，给出如下公式

$$
P_B = \frac{N_B + 5}{N_B + N_G + 10} = 0.4 \qquad P_G = \frac{N_G + 5}{N_B + N_G + 10} = 0.6
$$

增加了一点先验的知识，使得结果在观察数据量不大的时候，不会太离谱，此时先验的知识会有比较高的权重，当观察的数据量很大时，先验的知识影响力就变得微乎其微。而公式中的5和10就是先验分布的 伪计数 。

务必要理解这个 伪计数 的含义，我们这么做的目的是为了让参数的点估计结果在观察数据量很小的时候仍然可以得到一个不太离谱的数，相当于给要计算的参数增加了先验知识，这里假设参数的先验概率符合均匀分布. 当数据量较少时，其实就是证据不足时我们仍然相信我们的先验概率，只不过根据观察到的证据稍微修正先验知识。 ***贝叶斯理论的核心就是如何根据观察到的数据更新先验知识，参数真正的分布是什么，只有上帝知道，我们永远无法得知，我们只有通过不断地观察学习，来更新我们的认知，使得我们的认知更接近真理!***

下面再来学术性的分析下先验概率和后验概率的关系，投掷一个非均匀硬币，可以使用参数为θ的伯努利模型，θ为正面朝上的概率，那么结果x的分布形式为

$$
p(x|\theta) = \theta^x \cdot (1-\theta)^{1-x} \qquad x = 0, 1
$$

这其实是一个关于参数θ的函数，因此叫做似然函数。我们已经知道了两点分布和二项分布的共轭先验是 `Beta` 分布，它具有两个参数α, β， `Beta` 分布的形式为

$$
p(\theta|\alpha,\beta) = \begin{cases} \frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}, \theta \in [0,1] \\ 0 \qquad\qquad\qquad\qquad , other \end{cases}
$$

有了先验概率和似然函数，使用贝叶斯公式计算下后验概率

$$
\begin{aligned}
p(\theta|x) &= \frac{p(x|\theta)\cdot p(\theta)}{p(x)} \\
&\propto p(x|\theta)\cdot p(\theta) \\
&= (\theta^x\cdot(1-\theta)^{1-x})\cdot(\frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}) \\
&\propto \theta^x\cdot(1-\theta)^{1-x}\cdot\theta^{\alpha-1}\cdot(1-\theta)^{\beta-1} \\
&= \theta^{(x+\alpha)-1}(1-\theta)^{(1-x+\beta)-1}
\end{aligned}
$$

我们发现后验概率的分布和先验概率的分布完全一样，即伯努利分布的共轭先验是 Beta 分布。

参数α,β是决定参数$theta$的参数，常称为超参数。在后验概率的最终表达式中，参数α,β和x一起作为参数θ的指数—后验概率的参数为$(x+\alpha,1-x+\beta)$ 。 而这个指数的实际意义是：投币过程中，正面朝上的次数。参数α,β先验的给出了在没有任何试验的情况下，硬币朝上的概率分布，因此参数α,β被称为 伪计数 。

5.3.3.5. Dirichlet 分布

多项分布是二项分布在高维空间的推广，多项分布的先验分布也是 Beta 分布在高维空间的推广，它就是 Dirichlet 分布 。

此时此刻你一定要🙏感谢上帝，我们终于绕回来了，毕竟我们在说 LDA 模型，毕竟那个 D 代表的就是 Dirichlet ，我们走了两万五千里长征终于切入正题了。关于 Dirichlet 分布 也可以通过 Rickjin 的 LDA数学八卦 的游戏示例推导出相关结论， 但本节我们就免去那些繁琐的推导，毕竟它是二项分布在高维空间的推广，那就以对比推广的形式给出结论。

首先，再看下 Beta分布

$$
f(x)=\begin{cases}\frac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1}, x\in[0,1] \\ 0 \qquad\qquad\qquad\quad , other\end{cases}
$$

其中

$$
B(\alpha,\beta)=\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}
$$

对应的 Dirichlet 分布为

$$
f(\vec{p}|\vec{\alpha})=\begin{cases}\frac{1}{\Delta(\vec{\alpha})}\prod_{k=1}^{K}p_k^{\alpha k-1}, P_k\in[0,1] \\ 0 \qquad\qquad\quad , other\end{cases}
$$

简记为

$$
Dir(\vec{p}|\vec{\alpha})=\frac{1}{\Delta(\vec{\alpha})}\prod_{k=1}^{K}p_k^{\alpha k-1}
$$

其中

$$
\Delta(\vec{\alpha})=\frac{\prod_{k=1}^{K}\Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^{K}\alpha_k)}
$$

对 Dirichlet 分布求积分，我们还可以得到一个$\Delta(\vec{\alpha})$的积分形式

$$\int Dir(\vec{p} \mid \vec{\alpha}) = 1$$
$$\Rightarrow \frac{1}{\Delta(\vec{\alpha})} \int p_k^{\alpha^k - 1} dp_k = 1$$
$$\Rightarrow \Delta(\vec{\alpha}) = \int \prod_{k=1}^{K} p_k^{\alpha^k - 1} dp_k \qquad (5.12)$$

有了后验概率，我们应该如何估计我们的参数呢？合理的方式是使用后验分布的极大值点，或者是参数在后验分布中的平均值，本文中，我们取平均值作为参数的估计值。 抽样分布的数学期望等于总体参数的真值 .

让我们来看下 Beta/Dirichlet 分布 的数学期望到底是什么？ 如果$p\ Beta(t|\alpha,\beta)$，则

$$\begin{aligned} E(p) &= \int_0^1 t \cdot Beta(t|\alpha,\beta) dt \\ &= \int_0^1 t \cdot \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} t^{\alpha-1}(1-t)^{\beta-1} dt \\ &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} t^{\alpha}(1-t)^{\beta-1} dt \qquad (5.13) \end{aligned}$$

上式右边的积分对应概率分布$Beta(t|\alpha+1,\beta)$，对于这个分布，我们有

$$\int_0^1 \frac{\Gamma(\alpha+\beta+1)}{\Gamma(\alpha+1)\Gamma(\beta)} t^{\alpha}(1-t)^{\beta-1} dt = 1$$

代入公式(5.12)，可以得到

$$\begin{aligned} E(p) &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \cdot \frac{\Gamma(\alpha+1)\Gamma(\beta)}{\Gamma(\alpha+\beta+1)} \\ &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha+\beta+1)} \cdot \frac{\Gamma(\alpha+1)}{\Gamma(\alpha)} \\ &= \frac{\alpha}{\alpha+\beta} \end{aligned}$$

这说明，对于 Beta分布 的随机变量，其均值可以用$\frac{\alpha}{\alpha+\beta}$来估计。同理， Dirichlet分布 也有类似的结论，如果$\vec{p} \sim Dir(\vec{t} \mid \vec{\alpha})$, 同样可以证明

$$E(\vec{p}) = \left(\frac{\alpha_1}{\sum_{i=1}^K \alpha_i}, \frac{\alpha_2}{\sum_{i=1}^K \alpha_i}, \ldots, \frac{\alpha_K}{\sum_{i=1}^K \alpha_i} \right) \qquad (5.14)$$

以上两个结论非常重要，因为我们在后面的 LDA 数学推导中需要使用这个结论。简单想下为何后面会遇到，因为我们的条件概率是由贝叶斯公式来求解的，贝叶斯公式中有积分的情况，而高维空间的积分是不好求解的，但是我们知道随机变量和概率密度的积分就是期望！ 哇塞，感觉到一把利剑在手，有么有？ 不懂没关系，后面还会讲解。

差点忘了我们要干嘛，有了多项分布的先验分布是 Dirichlet分布 ，我们的目的是求在已知似然函数为多项分布的情况后验概率分布的参数会怎么更新，仍然对比二项分布来看

$$Beta(p \mid \alpha,\beta) + BinomCount(m_1, m_2) = Beta(p \mid \alpha + m_1, \beta + m_2) \qquad (5.15)$$

来对比看下多项分布

$$Dir(\vec{p} \mid \vec{k}) + MultCount(\vec{m}) = Dir(\vec{p} \mid \vec{k} + \vec{m}) \qquad (5.16)$$

公式(5.16)非常重要，其中k为伪计数，m为观察到的属于p_k的个数。

5.3.3.6. LDA 模型

已经接近终点了，让我们做下来思索下走过的路，有没有迷路呢?

- 我们从最开始了了解到如果想知道一篇文章的主题，那么如果知道每个词对应的主题就好了，也就是需要知道每个主题对应的词分布；
- 然后我们希望给每个词随机赋值一个主题编号，通过迭代的方式，最后可以收敛到主题对应的词分布，当然这个前提不管我们开始给每个词赋值了什么主题编号，都不影响最后的结果；
- 神奇的事情发生了，我们发现马氏链的平稳分布就具有这个特点，怎么才能结合上呢?
- 我们研究了马氏链，发现获得平稳分布的关键在于转换矩阵，转换矩阵可能有很多，我们只需要找到一个就可以了，于是我们取巧，根据对称性发现了 Metropolis Hastings 算法；
- Metropolis Hastings 算法 有个缺陷就是在状态转移时有接受率的限制，接受率较低的话会导致很难遍历完所有马氏链的状态，算法收敛速度很慢，有没有接受率为100%的算法呢；
- 神奇的事情又发生了，沿着坐标轴进行转移时，发现接受率100%，我们无意中发现了 Gibbs Sampling 算法,这个算法的核心是坐标轮转，如何结合到我们的问题场景下呢；
- 我们发现将文档看作一个由词构成的高维空间，在更新每个词的主题时，只需要依据其他词的主题分布，固定其他词不变，只改变当前词的做法就是坐标轮转大法；
- 我们发现这个更新规则是个条件概率，不好求解，于是我们想到了贝叶斯公式，想到了全概率公式来求解的方法，那就开始研究这些变量之间的关系，以便获取变量间的全概率公式；
- 通过将所有变量的关系进行梳理，我们发现本文的生成就是一个上帝掷骰子的过程，写文章时其实首先要确定主题，然后根据主题写选择词汇，这个主题是个隐变量；
- 文章到主题是多项分布，主题到词汇也是多项分布，多项分布的参数模型可以认为是上帝手里的骰子，而既然是参数，是变量，贝叶斯学派是不会放任它太随意了，让这两个多项分布的参数都来个先验分布吧；
- 选择什么样的先验分布合适呢? 贝叶斯学派关注的是在观察数据的基础上，是如何更新先验知识的，换句话说，先验分布加上观察数据变成后验分布，其实又是下一次观察的先验分布，因此我们希望找到先验和后验分布类型相同的分布；
- 为了简化问题，我们避开了直接研究多项分布的先验分布问题，因为那是高维空间的，不好抽象和理解；我们从多项分布的特例，即二项分布出发；
- 通过和上帝玩了几个游戏，我们神奇的发现：二项分布的共轭先验分布 Beta分布 ，于是我们顺势就将二项分布推广到了高维空间，发现多项分布的共轭先验分布是 Dirichlet分布；
- 既然知道了先验分布是 Dirichlet分布 ，返回去求解条件概率时会用到全概率公式，既然有高维积分的形式，那就看看是不是这个分布的期望比较好求呢?
- 神奇的事情又又又发生了， Beta分布 和 Dirichlet分布 的期望值都是如此的简洁，不得不感慨 数学之美啊 ！

上面重新梳理了下思路，让我们按照上面的思路继续解决问题，下面让我们用 概率图模型 表示 LDA 模型的物理过程，如图所示。

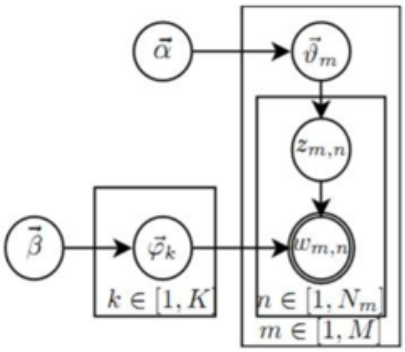


图 29. LDA概率图模型表示

这个概率图模型可以分解为两个主要的物理过程：

- $\vec{\alpha} \rightarrow \vec{\theta} \rightarrow z_{m,n}$,这个过程表示生成第$m$篇文档的时候，先从第一个坛子中抽了一个 `doc-topic` 骰子$\vec{\theta}$,然后投掷这个骰子生成了文档中第n个词的主题编号$z_{m,n}$；
- $\vec{\beta} \rightarrow \overrightarrow{\varphi_k} \rightarrow w_{m,n} \mid k = z_{m,n}$,这个过程表示用如下动作生成语料中第$m$篇文档的第$n$个词：在上帝手头的$K$个 `topic-word` 骰子$\overrightarrow{\varphi_k}$中，挑选编号为$k = z_{m,n}$的那个骰子进行投掷，然后生成单词$w_{m,n}$；

大家都说理解 `LDA` 最重要的就是理解上面这两个物理过程，其实这两个物理过程还是很好理解的。首先我们来看下文章中的主题和词的联合分布

$$p(\vec{w}, \vec{z} \mid \vec{\alpha}, \vec{\beta}) = p(\vec{w} \mid \vec{z}, \vec{\beta}) p(\vec{z} \mid \vec{\alpha}) \qquad (5.17)$$

公式第一项因子是给定主题采样词的过程；后面的因子计算，$n_z^{(t)}$表示词t被观察到分配给主题z的次数，$n_m^{(k)}$表示主题k分配给文档m的次数。

计算第一个因子如下

$$\begin{aligned} p(\vec{w} \mid \vec{z}, \vec{\beta}) &= \int p(\vec{w} \mid \vec{z}, \Phi) \cdot p(\Phi \mid \vec{\beta}) d\Phi \\ &= \int \prod_{z=1}^{K} \cdot \frac{1}{\Delta(\vec{\beta})} \prod_{t=1}^{V} \varphi_{z,t}^{n_z^{(t)} + \beta_t - 1} d\overrightarrow{\varphi_z} \\ &= (continue \ldots) \end{aligned}$$

这一步推导依据公式(5.16)，后面的概率是由参数β控制的 `Dirichlet分布` ，前面的概率是多项分布，多项分布乘以先验 `Dirichlet分布` 得到的结果仍然为 `Dirichlet分布`，其中Φ为词分布，$\overrightarrow{\varphi_z}$为主题为$z$的词分布，因为共有$K$个主题，所以对主题向量$\vec{z}$中的每个主题对应的词分布$\overrightarrow{\varphi_z}$求积分. 根据公式(5.16)后验 `Dirichlet分布` 可以表示为归一化因子和参数的乘积，其中参数的伪计数由原来的伪计数$\beta_t - 1$基础上增加观察量$n_z^{(t)}$ 表示词t被分配为主题z的次数。

$$= \prod_{z=1}^{K} \frac{\Delta(\overrightarrow{n_z} + \vec{\beta})}{\Delta(\vec{\beta})} \quad (5.18)$$

上一步推导依据公式(5.12)，将积分转换为归一化因子即可,其中

$$\overrightarrow{n_z} = \left\{ n_z^t \right\}_{t=1}^{V}$$

同理，计算公式(5.17)联合分布的第二个因子

$$
\begin{aligned}
p(\vec{z} \mid \vec{\alpha}) &= \int p(\vec{z} \mid \Theta) p(\Theta \mid \vec{\alpha}) d\Theta \\
&= \int \prod_{m=1}^{M} \frac{1}{\Delta(\vec{\alpha})} \prod_{k=1}^{K} \theta_{m,k}^{n_m^{(k)}+\alpha_k-1} d\overrightarrow{\theta_m} \\
&= \prod_{m=1}^{M} \frac{\Delta(\overrightarrow{n_m}+\vec{\alpha})}{\Delta(\vec{\alpha})} \qquad ,\overrightarrow{n_m} = \left\{ n_m^{(k)} \right\}_{k=1}^{K} \qquad (5.19)
\end{aligned}
$$

主题分布和词分布的计算思路完全相同，不再赘述。

将公式(5.18)和(5.19)代入第i个词主题编号的概率公式得到：

$$
\begin{aligned}
p(z_i = k \mid \overrightarrow{z_{\neg i}}, \vec{w}) &= \frac{p(\vec{w}, \vec{z})}{p(\vec{w}, \overrightarrow{z_{\neg i}})} = \frac{p(\vec{w} \mid \vec{z})}{p(\overrightarrow{w_{\neg i}} \mid \overrightarrow{z_{\neg i}}) p(w_i)} \cdot \frac{p(\vec{z})}{p(\overrightarrow{z_{\neg i}})} \\
&\propto \frac{\Delta(\overrightarrow{n_z}+\vec{\beta})}{\Delta(\overrightarrow{n_{z,\neg i}}+\vec{\beta})} \cdot \frac{\Delta(\overrightarrow{n_m}+\vec{\alpha})}{\Delta(\overrightarrow{n_{m,\neg i}}+\vec{\alpha})} \\
&= \frac{\Gamma(n_k^{(t)}+\beta_t)\Gamma(\sum_{t=1}^{V} n_{k,\neg i}^{(t)}+\beta_t)}{\Gamma(n_{k,\neg i}^{(t)}+\beta_t)\Gamma(\sum_{t=1}^{V} n_k^{(t)}+\beta_t)} \cdot \frac{\Gamma(n_m^{(k)}+\alpha_k)\Gamma(\sum_{k=1}^{K} n_{m,\neg i}^{(k)}+\alpha_k)}{\Gamma(n_{m,\neg i}^{(k)}+\alpha_k)\Gamma(\sum_{k=1}^{K} n_m^{(k)}+\alpha_k)} \\
&= \frac{n_{k,\neg i}^{(t)}+\beta_t}{\sum_{t=1}^{V} n_{k,\neg i}^{(t)}+\beta_t} \cdot \frac{n_{m,\neg i}^{(k)}+\alpha_k}{[\sum_{k=1}^{K} n_{m,\neg i}^{(k)}+\alpha_k]-1} \qquad (5.20) \\
&\propto \frac{n_{k,\neg i}^{(t)}+\beta_t}{\sum_{t=1}^{V} n_{k,\neg i}^{(t)}+\beta_t} \cdot (n_{m,\neg i}^{(k)}+\alpha_k)
\end{aligned}
$$

上面是邹博老师的推导，感觉这么推导结果不是那么美啊，还是看看 Rickjin 的积分转为期望的推导

$$
\begin{aligned}
p(z_i = k \mid \overrightarrow{z_{\neg i}}, \vec{w}) &\propto p(z_i = k, w_i = t \mid \overrightarrow{z_{\neg i}}, \overrightarrow{w_{\neg i}}) \\
&= \int p(z_i = k, w_i = t, \overrightarrow{\theta_m}, \overrightarrow{\varphi_k} \mid \overrightarrow{z_{\neg i}}, \overrightarrow{w_{\neg i}}) d\overrightarrow{\theta_m} d\overrightarrow{\varphi_k} \\
&= \int p(z_i = k \mid \overrightarrow{\theta_m}) p(\overrightarrow{\theta_m} \mid \overrightarrow{z_{\neg i}}, \overrightarrow{w_{\neg i}}) \cdot p(w_i = t \mid \overrightarrow{\varphi_k}) p(\overrightarrow{\varphi_k} \mid \overrightarrow{z_{\neg i}}, \overrightarrow{w_{\neg i}}) d\overrightarrow{\theta_m} d\overrightarrow{\varphi_k} \\
&= \int p(z_i = k \mid \overrightarrow{\theta_m}) Dir(\overrightarrow{\theta_m} \mid \overrightarrow{n_{m,\neg i}}+\vec{\alpha}) d\overrightarrow{\theta_m} \cdot p(w_i = t \mid \overrightarrow{\varphi_k}) Dir(\overrightarrow{\varphi_k} \mid \overrightarrow{n_{k,\neg i}}+\vec{\beta}) d\overrightarrow{\varphi_k} \\
&= \int \theta_{mk} Dir(\overrightarrow{\theta_m} \mid \overrightarrow{n_{m,\neg i}}+\vec{\alpha}) d\overrightarrow{\theta_m} \cdot \int \varphi_{kt} Dir(\overrightarrow{\varphi_k} \mid \overrightarrow{n_{k,\neg i}}+\vec{\beta}) d\overrightarrow{\varphi_k} \\
&= E(\theta_{mk}) \cdot E(\varphi_{kt}) \\
&= \hat{\theta}_{mk} \cdot \hat{\varphi}_{kt} \\
&= \frac{n_{m,\neg i}^{(k)}+\alpha_k}{\sum_{k=1}^{K} n_{m,\neg i}^{(k)}+\alpha_k} \cdot \frac{n_{k,\neg i}^{(t)}+\beta_t}{\sum_{t=1}^{V} n_{k,\neg i}^{(t)}+\beta_t} \qquad (5.20)
\end{aligned}
$$

公式(5.20)就是 LDA模型 的 Gibbs Sampling 公式，这个概率其实是$doc \rightarrow topic \rightarrow word$的路径概率，由于$topic$有$K$个，所以 Gibbs Sampling 公式的物理意义就是在这K条路径中进行采样。

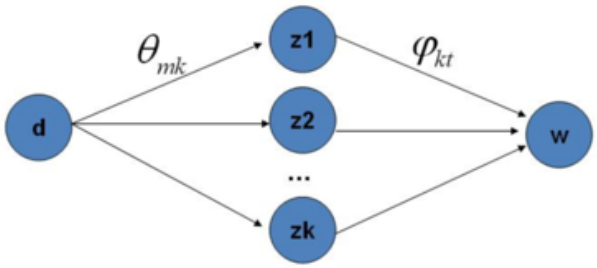


图 30. Gibbs采样路径概率

5.3.3.7. 模型训练和推演

有了 LDA 模型，当然我们的目标有两个

- 估计模型中的参数$\vec{\varphi^1}, \dots, \vec{\varphi^K}$ 和 $\vec{\theta^1}, \dots, \vec{\theta^M}$
- 对新来的一篇文档，我们能够计算这篇文档的主题分布

训练的过程就是通过 Gibbs Sampling 获取预料中的(z, w)样本，而模型中的所有参数都可以基于最终采样得到的样本进行估计，训练的流程为

Algorithm 14 LDA Training

1: 随机初始化：对语料中每篇文档中的每个词w，随机的赋一个topic 编号z;
2: 重新扫描语料库，对每个词w，按照Gibbs Sampling 公式重新采样它的topic，在语料中进行更新;
3: 重复以上语料库的重新采样过程直到Gibbs Sampling 收敛;
4: 统计语料库的topic-word 共现频率矩阵，该矩阵就是LDA的模型;

图 31. LDA训练过程

由这个 topic-word 频率矩阵我们可以计算每一个$p(word|topic)$概率，从而算出模型的参数，通常在 LDA 模型训练的过程中，我们是取 Gibbs Sampling 收敛之后的n个迭代的结果进行平均来做参数估计，这样模型质量更高。

有了模型，对于新来的文档，我们如何做主题语义分布的计算呢？基本上推理和训练的过程完全类似。对于新的文档，我们只要认为 Gibbs Sampling 公式中的$\vec{\varphi_{k,t}}$部分是稳定不变的，是由训练预料得到的模型提供的， 所以采样的过程中我们只要估计该文档的$topic$分布就好了。

Algorithm 15 LDA Inference

1: 随机初始化：对当前文档中的每个词w，随机的赋一个topic 编号z;
2: 重新扫描当前文档，按照Gibbs Sampling 公式，对每个词w，重新采样它的topic;
3: 重复以上过程直到Gibbs Sampling 收敛;
4: 统计文档中的topic分布，该分布就是$\vec{\theta}_{new}$

图 32. LDA推演过程

5.3.4. 变分EM算法求解pLSA模型

由于后面会有独立的章节介绍EM算法，因此本节我们介绍其基本原理，在不影响pLSA算法正常运算逻辑基础上尽可能简化介绍。

5.3.4.1. EM算法

EM算法(Expectation Maximization algorithm),又称期望最大算法，其基本思想是：首先随机选取一个值去初始化待估计的值θ^0，然后不断迭代寻找更优θ^{n+1}的使得其似然函数比原来的要大。 即通过不断迭代更新，提高似然函数的概率值，使得最大似然函数值取极值点的参数就是最优化参数。

所以EM算法的一般步骤为：

- 随机选取或者根据先验知识初始化θ^0;
- 不断迭代下述两步：
  - 给出当前的参数估计θ^n，计算似然函数$L(\theta)$的下界$Q(\theta;\theta^n)$;
  - 重新估计参数θ，即求θ^{n+1},使得$\theta^{n+1}=\underset{\theta}{argmax}\ Q(\theta;\theta^n)$;
- 上述第二步后，如果$L(\theta)$ (即$Q(\theta;\theta^n)$)收敛，则退出算法，否则继续回到第二步。

下面给出推导公式，假设有训练集$\left\{x^1,x^2,\dots,x^m\right\}$,包含$m$个独立样本，希望从中找到该数组的模型$p(x,z)$的参数。其中$z$是隐变量， 然后通过极大似然估计建立目标函数(对数似然函数)

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{m} log\, p(x;\theta) \\ &= \sum_{i=1}^{m} log \sum_{z} p(x;z;\theta) \end{aligned}$$

这里z是隐变量，要找到参数的估计很难，我们的策略是建立$l(\theta)$的下界，并且求该下界的最大值；重复这个过程，直到收敛到局部最大值。

令Q_i为z的某一个分布，$Q_i \geq 0$，根据 Jenssen 不等式，有

$$\begin{aligned} \sum_{i=0}^{m} log\, p(x^i;\theta) &= \sum_{i=0}^{m} log \sum_{z^0}^{z^m} p(x^i;z^i;\theta) \\ &= \sum_{i=0}^{m} log \sum_{z^0}^{z^m} Q_i(z^i)\frac{p(x^i;z^i;\theta)}{Q_i(z^i)} \\ &\geq \sum_{i=0}^{m} \sum_{z^0}^{z^m} Q_i(z^i) log \frac{p(x^i;z^i;\theta)}{Q_i(z^i)} \end{aligned}$$

要理解上面不等式的转换很简单，将$Q_i(z^i)$看作 Jenssen 不等式中的平衡因子，且$\sum_{i=0}^{m} Q_i(z^i)=1$，我们要寻找的下界就是不等式取等号， 若要让 Jenssen 不等式，则必须让两个点重合，点的值为固定常量，即满足

$$\frac{p(x^i;z^i;\theta)}{Q_i(z^i)} = C$$

所以可得

$$\begin{aligned} Q_i(z^i) &= \frac{p(x^i,z^i;\theta)}{\sum_{i=0}^{m} p(x^i,z^i;\theta)} \\ &= \frac{p(x^i,z^i;\theta)}{p(x^i;\theta)} \\ &= p(z^i \mid x^i;\theta) \end{aligned}$$

最后得导EM算法的正题框架

Repeat until convergence {

(E-step) For each i, set

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

(M-step) Set

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

}

图 33. EM算法伪代码

5.3.4.2. 求解pLSA算法

首先尝试从矩阵的角度来描述两个待估计的变量$p(w_j \mid z_k)$和$p(z_k \mid d_i)$。

假定用ϕ_k表示词表V在主题z_k上一个多项分布，则ϕ_k可以表示为一个向量，每个元素$\phi_{k,j}$表示词项w_j在主题z_k上的概率，即

$$p(w_j \mid z_k) = \phi_{k,j}, \qquad \sum_{wj \in V} \phi_{k,j} = 1$$

用θ_i表示主题Z在文档d_i上的一个多项分布，则θ_i也可以表示为一个向量，每个元素$\theta_{i,k}$表示主题z_k出现在文档d_i中的概率，即

$$p(z_k \mid d_i) = \theta_{i,k}, \qquad \sum_{z_t \in Z} \theta_{i,k} = 1$$

这样巧妙的把$p(w_j \mid z_k)$和$p(z_k \mid d_i)$转换为了两个矩阵，换言之我们最终要求解的参数是两个矩阵

$$\Phi = \{\phi_1, \ldots, \phi_K\}, \qquad z_k \in Z$$
$$\Theta = \{\theta_1, \ldots, \theta_M\}, \qquad d_i \in D$$

由于每篇文章的词和词之间是相互独立的，所以整篇文章N个词的分布为

$$p(W \mid d_i) = \prod_{j=1}^{N} p(d_i, w_j)^{n(di,wj)}$$

而由于文章和文章之间也是相互独立的，所以整个预料的词分布为

$$p(W \mid D) = \prod_{i=0}^{M} \prod_{j=1}^{N} p(d_i, w_j)^{n(di,wj)}$$

其中$n(d_i, w_j)$表示词项w_j在文档d_i中的词频，$n(d_i)$表示文档d_i中词的总数,显然$n(d_i) = \sum_{wj \in V} n(d_i, w_j)$。从而得到整个语料库的词分布的对数似然函数

$$\begin{aligned} l(\Phi, \Theta) &= \sum_{i=1}^{M} \sum_{j=1}^{N} n(d_i, w_j) log\, p(d_i, w_j) \\ &= \sum_{i=1}^{M} n(d_i)(logp(d_i) + \sum_{j=1}^{N} \frac{n(d_i, w_j)}{n(d_i)}) log \sum_{k=1}^{K} p(w_j \mid z_k) p(z_k, d_i) \\ &= \sum_{i=1}^{M} n(d_i)(logp(d_i) + \sum_{j=1}^{N} \frac{n(d_i, w_j)}{n(d_i)}) log \sum_{k=1}^{K} \phi_{k,j} \theta_{i,k} \end{aligned}$$

现在，我们的工作就是最大化这个对数似然函数，求解参数$\phi_{k,j}$和$\theta_{i,k}$，对于这种隐含变量的最大似然估计，可以使用EM算法，典型的EM算法分成两步，先E后M。

- E-step:假定参数已知，计算此时隐变量的后验概率，利用贝叶斯公式，可得

$$\begin{aligned} p(z_k \mid d_i, w_j) &= \frac{p(z_k, d_i, w_j)}{\sum_{l=1}^{K} p(z_l, d_i, w_j)} \\ &= \frac{p(w_j \mid d_i, z_k)p(z_k \mid d_i)p(d_i)}{\sum_{l=1}^{K} p(w_j \mid d_i, z_l)p(z_l \mid d_i)p(d_i)} \\ &= \frac{p(w_j \mid z_k)p(z_k \mid d_i)}{\sum_{l=1}^{K} p(w_j \mid z_l)p(z_l \mid d_i)} \\ &= \frac{\phi_{k,j}\theta_{i,k}}{\sum_{l=1}^{K} \phi_{l,j}\theta_{i,l}} \end{aligned}$$

- M-step:这样隐变量的乘积可以用后验概率表示，将其代入似然函数，最大化对数似然，求解相应参数。

由于文档长度$p(d_i) \propto n(d_i)$可单独计算，去掉它不影响最大似然函数，此外，根据 `E-step` 结果，将$\phi_{k,j}\theta_{i,k} = p(z_k \mid d_i, w_j) \sum_{l=1}^{K} \phi_{l,j}\theta_{i,l}$带入似然函数,得到:

$$l = \sum_{i=1}^{M} \sum_{j=1}^{N} n(d_i, w_j) \sum_{k=1}^{K} p(z_k \mid d_i, w_j)\, log(\phi_{k,j}, \theta_{i,k}) \qquad (5.21)$$

这是一个多元函数求极值问题，并且有如下的约束条件:

$$\sum_{j=1}^{N} \phi_{k,j} = 1$$

$$\sum_{k=1}^{K} \theta_{i,k} = 1$$

一般这种带约束条件的极值问题，常用的方法便是拉格朗日乘法，即通过引入拉格朗日乘子将目标函数和约束条件融合到一起，转化为无约束条件的优化问题。

$$H = L^c + \sum_{k=1}^{K} \gamma_k (1 - \sum_{j=1}^{N} \phi_{k,j}) + \sum_{i=1}^{M} \rho_i (1 - \sum_{k=1}^{K} \theta_{i,k})$$

分别对$\phi_{k,j}$和$\theta_{i,k}$求导，令导数为0，得到

$$\phi_{k,j} = \frac{\sum_{i=1}^{M} n(d_i, w_j) p(z_k \mid d_i, w_j)}{\sum_{j=1}^{M} \sum_{i=1}^{N} n(d_i, w_j) p(z_k \mid d_i, w_j)}$$

$$\theta_{i,k} = \frac{\sum_{j=1}^{N} n(d_i, w_j) p(z_k \mid d_i, w_j)}{n(d_i)}$$

5.4. 终篇

走到这里的人，我相信你们一定被 `LDA` 模型深深的吸引，这也是我为何要花费大量时间编写这个章节的原因，LDA原始论文中是基于变分 `EM` 算法进行参数公式推导，当然对应的模型实现也可以采用变分法，本章我们对基于 `Gibbs Sampling` 的 `LDA` 模型进行了详细描述，整章的编写花费了大概两天的时间，是在太累了， 后续有时间再将变分法补上(已经补上)。

6. 最大熵理论和多分类器

7. EM 算法

8. 决策树和集成学习

本章我们将介绍两个非常重要的概念，决策树和集成学习。应该说市面上还没有哪本书会将这两个概念放在一个章节去讲，我觉得主要的考虑应该是这两个概念并没有本质上的联系， 比如决策树不属于集成学习，而集成学习没有限定用到的基模型必须是树模型。

大家可以这样简单理解这个章节，首先决策树具有很多优势：简单、解释性强、分类速度快等，也正式因为这些因素，使得决策树模型很容易过拟合；没办法我们不想让它泛化能力太差， 于是每个决策树就变成了一个弱决策树， 三个臭皮匠胜于诸葛亮 ，所以我们多来几个弱决策树，让他们投票表决结果，这就有了随机森林，而随机森林是典型的集成学习， 所以这就是本章节两大基本概念的关系。

举个例子先来感性的认识下刚才说了啥：

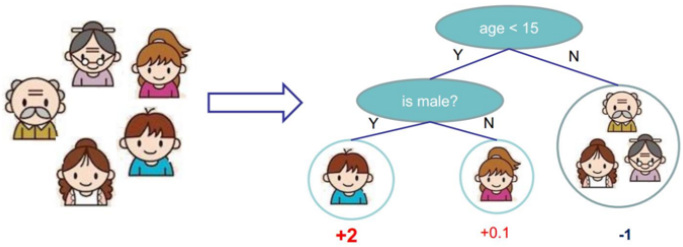


图 34. 分类回归决策树

这是一颗 CART 树，关于什么是 CART 树，后面会单独讲，你可以认为这就是一个决策树，图中给出的是预测样本中的每个人是不是喜欢计算机游戏，这是一个典型的二分类问题， 我们假定了样本共有三个特征，我们的这棵决策树深度为2，对每个样本首先判断年龄是否小于15岁，然后在小于15岁的样本里再判定性别是否为男性，如果样本里只有小明(左侧男孩)一个人爱玩计算机游戏， 那么最左边的决策路径就可以很好的知道爱玩计算机游戏的样本，同时可以看到决策树在这条路径上输出的结果最大。

上面的那颗决策树对我们的例子没什么问题，但是并没有考虑是否每天都有使用计算机这个特征，因此容易欠拟合，但是考虑到其在训练样本中已经有了很高的准确率，因此再增加树的深度，又有过拟合的风险， 因此我们又单独生成了一个决策树，这棵决策树只考虑第三个特征，如下图，我们发现小明和爷爷每天都有使用计算机的习惯，因此在这条决策路径上输出的结果较大。

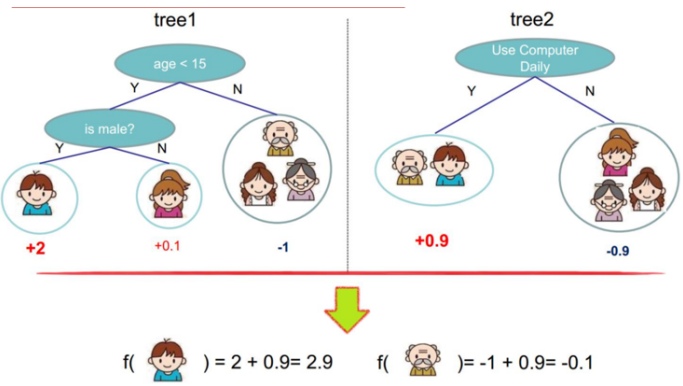


图 35. 随机森林

最后的结果我们可以通过利用两棵树投票加权来决定谁更喜欢玩计算机游戏，很显然小明非常喜欢玩。

8.1. 决策树模型

下面开始介绍第一个概念： 决策树 ，决策树是一种基本的分类和回归方法，既可以用来解决分类问题也可以解决回归问题，因此又可以分为分类树和回归树， 回归树主要优势就是模型简单，可读性强，分类速度快，学习时，利用训练数据，根据损失函数最小化的原则建立决策树模型，决策树的学习主要包括三个步骤:

- 特征选择
- 决策树的生成

- 决策树的剪枝

8.1.1. 决策树思想

决策树模型结构类似于数据结构中的树，虽然大家常看到的决策树是二叉树，但其实决策树并没有这个限制。决策树的节点包括：内部节点和叶子节点，内部节点表示一个特征或属性， 叶子节点表示一个类。

可以将决策树理解为一个 `if-then` 的规则集，决策树的根节点到叶子节点的每一条路径构成一条规则，叶节点的类对应着规则的结论，这个规则集有个很重要的性质：互斥并且完备。 也就是说每个样本都被一条路径或一条规则所覆盖，并且只被一条路径或一条规则覆盖。决策树还可以理解为给定特征条件下的条件概率分布，决策树在分类时将节点的样本强行分类到条件概率大的那一类。

决策树学习本质上就是从训练数据集中归纳出一组分类规则，与训练集不相矛盾的规则有很多，我们需要的是一个与训练集矛盾较小的决策树，同时具有很好的泛化能力。 从另一个角度来看，我们选择的条件概率模型应该不仅对训练数据有很好的拟合，而且对未知数据有很好的理解。

不失一般性，决策树学习也用损失函数表示这一目标，决策树学习的损失函数通常是正则化的极大似然函数。因为所有可能的决策树中选取最优决策树是一个 `NP` 完全问题， 所以现实中决策树学习算法通常采用启发方法，近似求解这一最优化问题，决策树通常是递归的选择最优特征进行生长，因此是个典型的 **贪心算法** 。

决策树学习算法的最大优点是：它可以自学习。在学习过程中不需要使用者了解太多背景知识，只需要对训练实例进行较好的标注，就能够进行学习，属于有监督学习。

8.1.2. 特征选择

由于决策树模型是个典型的树形结构，每层的节点代表一个特征，那么在节点进行分裂的时候，我们应该如何选择本层应该选择的特征呢？ 上节我们提到了这种最优的选择方法是启发式的， 是贪心的，也就是说我们在每次特征选择时都会选择这样的特征，它可以将训练样本尽可能明确的分开到各个类别中去。

这种描述实在是太绕嘴了，我们思索着用什么概念可以表征这种好与不好的差异呢？

8.1.2.1. 信息熵

想想我们的目标是什么，输入给我们的是一堆包含了很多特征的数据，我们对如何根据数据判定其类别一无所知，而结果是要有一系列规则，通过规则可以很容易知道所有数据属于哪类， 我们从一个信息量很大，收敛到一个明确的，完全没有任何信息量的事件，这个信息量其实就是信息论中 **熵** 的概念。

熵的定义：熵是表示随机变量不确定性的度量，假设X是一个可以取有限个值的离散随机变量，其概率分布为:

$$P(X = x_i) = p_i, \qquad i = 1, 2, \ldots, n$$

则随机变量X的熵定义为:

$$H(X) = -\sum_{i=1}^{n} p_i log\, p_i$$

熵值越大，随机变量的不确性越大，由定义可知:

$$0 \leq H(X) \leq log\, n$$

通常对数是以2或者e为底，对应的熵的单位分别为比特或者纳特。

条件熵：随机变量X给定的条件下，随机变量Y的不确定性就叫做条件熵$H(Y|X)$，定义为X给定条件下的概率分布的熵对X的数学期望:

$$H(Y|X)=\sum_{i=1}^{n}p_iH(Y|X=x_i)$$

条件熵也等于(X,Y)发生所包含的熵，减去X单独发生包含的熵:

$$H(Y|X)=H(X,Y)-H(X)$$

下面给出条件熵的推导公式:

$$\begin{aligned}H(X,Y)-H(X)&=-\sum_{x,y}p(x,y)log\,p(x,y)+\sum_{x}p(x)log\,p(x)\\&=-\sum_{x,y}p(x,y)log\,p(x,y)+\sum_{x}(\sum_{y}p(x,y))log\,p(x)\\&=-\sum_{x,y}p(x,y)log\,p(x,y)+\sum_{x,y}p(x,y)log\,p(x)\\&=-\sum_{x,y}p(x,y)log\frac{p(x,y)}{p(x)}\\&=-\sum_{x,y}p(x,y)log\,p(y|x)\end{aligned}$$

根据条件熵的定义，我们还可以推出如下结论:

$$\begin{aligned}H(X,Y)-H(X)&=-\sum_{x,y}p(x,y)log\,p(y|x)\\&=-\sum_{x}\sum_{y}p(x,y)log\,p(y|x)\\&=-\sum_{x}\sum_{y}p(x)p(y|x)log\,p(y|x)\\&=-\sum_{x}p(x)\sum_{y}p(y|x)log\,p(y|x)\\&=\sum_{x}p(x)(-\sum_{y}p(y|x)log\,p(y|x))\\&=\sum_{x}p(x)H(Y|X=x)\end{aligned}$$

当熵和条件熵中的概率由数据估计(特别是极大似然估计)得到时，所对应的熵与条件熵分别成为经验熵和经验条件熵。

8.1.2.2. 信息增益

信息增益：特征A对训练数据集D的信息增益$g(D,A)$,定义为集合D的经验熵$H(D)$与特征A给定条件下D的经验条件熵$H(D|A)$之差，即:

$$g(D,A)=H(D)-H(D|A)$$

决策树学习中的信息增益等价于训练数据集中类与特征的互信息。下面给出信息增益的计算方法:

假设训练数据集为D，$|D|$表示样本的个数，样本共有K类，用C_k表示每个类，其中$k=1,2,\dots,K$,$|C_k|$为属于类C_k的样本个数， 有: $\sum_k|C_k|=|D|$；假设特征A有n个不同的取值$a_1,a_2,\dots,a_n$，根据特征A的取值，将训练数据划分为n个子集，$D_1,D_2,\dots,D_n$，$|D_i|$为D_i的样本个数，有: $\sum_i|D_i|=D$；记D_i中属于类C_k的集合为D_{ik},$|D_{ik}|$为D_{ik}的样本个数。

首先计算样本D的经验熵，为:

$$H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} log \frac{|C_k|}{|D|}$$

然后遍历所有特征，对于特征A:

- 计算特征A对数据集D的经验条件熵$H(D|A)$，方法见下面公式
- 计算特征A的信息增益，$g(D,A) = H(D) - H(D|A)$
- 选择信息增益最大的特征作为当前的分裂特征

$$\begin{aligned} H(D|A) &= -\sum_{ik} p(D_k, A_i) log\, p(D_k|A_i) \\ &= -\sum_{ik} p(A_i) p(D_k|A_i) log\, p(D_k|A_i) \\ &= -\sum_{i}^{n} \sum_{k}^{K} p(A_i) p(D_k|A_i) log\, p(D_k|A_i) \\ &= -\sum_{i}^{n} p(A_i) \sum_{k}^{K} p(D_k|A_i) log\, p(D_k|A_i) \\ &= -\sum_{i}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} log\, \frac{|D_{ik}|}{|D_i|} \end{aligned}$$

8.1.2.3. 信息增益比

使用信息增益进行特征选择有个比较大的问题是：信息增益比较倾向于选择取值较多的特征，比如 `id` 这种，信息熵直接将为0，但实际毫无意义的特征，当然这里只是距离说明， 真实是不会存在这种特征。但是信息增益的这种倾向性，必须进行校正，我们可以使用信息增益比。

信息增益比：特征A对训练数据集D的信息增益比$g_R(D,A)$定义为其信息增益$g(D,A)$与训练数据集D关于特征A的值的熵$H_A(D)$之比，即

$$g_R(D,A) = \frac{g(D,A)}{H_A(D)}$$

其中，$H_A(D) = -\sum_{i=1}^{n} \frac{|D_1|}{|D|} log_2 \frac{|D_i|}{|D|}$，$n$是特征$A$取值的个数。

8.1.2.4. 基尼指数

对于分类树，一般常用基尼指数选择最优特征，同时决定该特征的最优二值切分点。

基尼指数：分类问题中，假如有K个类，样本的点属于第k个类的概率为p_k，则概率分布的基尼指数定义为:

$$Gini(p) = \sum_{k=1}^{K} p_k(1-p_k) = 1 - \sum_{k=1}^{K} p_k^2 = 1 - \sum_{k=1}^{K} (\frac{|C_k|}{|D|})^2$$

如何理解基尼指数的具体含义呢？根据下面的公式可以将基尼指数看作信息熵的一阶近似。

$$H(X) = -\sum_{k=1}^{K} p_k ln p_k \approx \sum_{k=1}^{K} p_k(1-p_k)$$

8.1.3. 决策树生成

决策树的生成主要包括 `ID3算法` 、 `C4.5` 和 `CART` 。

- `ID3`，使用信息增益或者叫互信息，进行特征选择；特征多的属性更容易使数据更纯，其信息增益最大，训练得到的是一棵庞大且深度浅的树，不合理；
- `C4.5`，利用信息增益率
- `CART`，利用基尼指数

8.1.3.1. CART生成

决策树的生成就是递归地构建决策树的过程，对回归树用平方误差最小化准则，对分类树用基尼指数最小化准则，进行特征选择，生成决策树。记住CART树是二叉树！

(1)回归树的生成

使用平方误差来表示回归树对于训练数据的预测误差，用平方误差最小的准则求解每个单元上的最优输出值，很显然这个值就是在该单元上的均值，这样的回顾树通常被称为最小二乘回归树，算法如图:

算法 5.5（最小二乘回归树生成算法）

输入：训练数据集 D；

输出：回归树 $f(x)$.

在训练数据集所在的输入空间中，递归地将每个区域划分为两个子区域并决定每个子区域上的输出值，构建二叉决策树：

（1）选择最优切分变量 j 与切分点 s，求解

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right] \tag{5.21}$$

遍历变量 j，对固定的切分变量 j 扫描切分点 s，选择使式 (5.21) 达到最小值的对 (j,s).

（2）用选定的对 (j,s) 划分区域并决定相应的输出值：

$$R_1(j,s)=\{x\mid x^{(j)}\leqslant s\},\quad R_2(j,s)=\{x\mid x^{(j)}>s\}$$

$$\hat{c}_m=\frac{1}{N_m}\sum_{x_i\in R_m(j,s)}y_i,\quad x\in R_m,\quad m=1,2$$

（3）继续对两个子区域调用步骤 (1)，(2)，直至满足停止条件.

（4）将输入空间划分为 M 个区域 $R_1,R_2,\cdots,R_M$，生成决策树：

$$f(x)=\sum_{m=1}^{M}\hat{c}_m I(x\in R_m)$$

■

图 36. 回归树的生成算法

实在是懒得敲了，毕竟这个算法比较简单，浪费时间没那个必要了，大家可以直接看 **统计学习方法** 一书的相关介绍，但是要记住一点，上面的过程都是基于我们选择了平方误差作为损失函数的前提下， 当我们选择其他损失函数时，不再使用，后面介绍集成学习时会介绍一个很熟悉的概念 **梯度** 。

(2)分类树的生成

分类树的构造过程类似于回归树，每次更新时需要对比所有可能的特征和它们所有可能的切分点，选择基尼指数最小的特征和对应的切分点作为最优特征和最优切分点。

8.1.4. 决策树剪枝

决策树生成算法递归的产生决策树，直到不能继续下去为止。这样产生的树往往对训练数据很准，但对于未知的测试数据的分类却没有那么准，即出现了过拟合现象。 过拟合的原因是因为学习时过多的考虑如何提高对训练数据的正确分类，从而构建出过于复杂的决策树，为了解决这个问题，可以考虑对决策树进行剪枝。

决策树的剪枝往往是通过极小化决策树整体的损失函数或代价函数实现，很显然这里的损失函数是正则化后的。

假设树T的叶子节点个数为$|T|$，t是树T的叶子节点，该叶子节点有N_t个样本点，其中k类的样本点有N_{tk}个，$k=1,2,\dots,K$，$H_t(T)$为叶子节点t上的经验熵，$\alpha\geq 0$为参数，则决策树学习的损失函数可以定义为:

$$C_\alpha(T)=\sum_{t=1}^{|T|}N_tH_t(T)+\alpha|T|$$

其中经验熵为:

$$H_t(K) = -\sum_k \frac{N_{tk}}{N_t} log \frac{N_{tk}}{N_t}$$

我们也可以将损失函数简写成如下形式:

$$C_\alpha(T) = C(T) + \alpha|T|$$

此时$C(T)$表示模型对训练数据的预测误差，$|T|$表示模型复杂度，参数$\alpha \geq 0$控制着两者之间的影响。 较大的α促使选择简单的模型，较小的$alpha$促使选择复杂的模型，而$\alpha = 0$只考虑模型和训练数据的拟合程度，不考虑模型的复杂度。

损失函数的作用就是平衡模型对训练数据的拟合程度以及模型的复杂度。可以看出，决策树生成只考虑了通过提高信息增益(或信息增益比)对训练数据进行更好的拟合， 而决策树剪枝通过优化损失函数还考虑了较小模型复杂度，决策树生成学习局部模型，而决策树剪枝学习整体模型。

剪枝的基本思想:

- 由完全树T_0开始，剪枝部分节点得到T_1,再次剪枝部分节点得到T_2，直到仅剩树根的树T_k
- 在验证数据集上对这k个树分别评价，选择损失函数最小的树T_α

剪枝过程的核心是：如何确定剪枝系数!可以按照如下方式确定剪枝的系数。根据损失函数$C_\alpha(T) = C(T) + \alpha \cdot |T_{leaf}|$

- 假定当前对以r为根的子树剪枝，剪枝后只保留r节点本身，删除掉所有的叶子
- 考察以r为根节点的子树，剪枝后的损失函数为$C_\alpha(r) = C(r) + \alpha$，让其等于剪枝前的损失函数 $C_\alpha(T) = C(T) + \alpha \cdot |T_{leaf}|$
- 求得$\alpha = \frac{C(r) - C(R)}{|R_{leaf}| - 1}$
- α称为节点r的剪枝系数

剪枝算法:

- 根据上面方法，计算所有内部节点的剪枝系数
- 查找最小剪枝系数的节点，剪枝得到决策树T_k
- 重复以上步骤，直到决策树T_k只剩一个节点
- 得到决策树序列$T_0, T_1, \dots, T_K$
- 使用验证样本集选择最优子树

验证集作最优子树的评价标准，可以使用评价函数$C(T) = \sum_{t \in leaf} N_t \cdot H(t)$。

剪枝是为了防止过拟合，提高模型的泛化能力，不过我们还有一种方式可以用来防止过拟合： 随机森林 ，由于随机森林属于集成学习的范畴，在开始介绍前，还是先让我们了解下集成学习的世界吧。

8.2. 集成学习

集成学习通过构建并合并多个学习器来完成学习任务，常可以获得比单一学习器显著优越的泛化能力，这对于 弱学习器 尤为明显，根据个体学习器的生成方式，目前集成学习大致分为两类:

- 个体学习器之间存在强依赖关系，必须窜行生成的序列化方法，如 Boostring方法
- 个体学习器之间不存在强依赖关系，可同时生成的并行化方法，如 Bagging方法 和 随机森林

8.2.1. Bagging和随机森林

Bagging 是并形式集成学习方法最著名的代表，其学习过程是基于一种叫做 Bootstrap 的自助采样法进行的，自助采样法是一种有放回的采样方法：

Bootstrap :假定样本的数据集个数为m，我们先随机的取出一个样本放到采样集中，再把该样本放回初始样本集，使得下次采样时仍有可能选中该样本， 这样经过m次的随机采样就得到了m个样本的采样集。照这样，我们可以采样出T个含m个训练样本的采样集，然后基于每个采样集训练出一个基学习器，再将这些基学习器进行结合，这就是 Bagging 的基本流程。 一般输出结果都是由这些分类器投票决定。

随机森林 (Random Forest)是 Bagging 的一个扩展变体，RF在以决策树为基学习器构建 Bagging 集成的基础上，进一步在决策树的训练过程中引入随机属性选择。

传统决策树在选择划分属性时，是在当前节点的属性集合中选择一个最优的属性，而在RF中，对基决策树的每个节点，先从该节点的属性集合中随机选择一个包含k个属性的子集，然后再从这个子集中选择一个最优属性进行划分。
随机森林的训练效率常优于Bagging，因为在个体决策树的构建过程中，Bagging使用的是"确定型"决策树，在选择划分属性时，要对节点的所有属性进行考察；而随机森林使用的"随机型"决策树则只考虑一个属性子集。

8.2.2. Boosting

首先来说下提升的概念，提升是一种机器学习技术，可用于回归和分类问题，它每一步产生一个弱学习器，并加权累加到总模型中；如果每一步的弱学习器生成都依赖于损失函数的梯度方向， 则称之为梯度提升(Gradient Boosting)。

梯度提升算法首先给定一个目标损失函数，它的定义域是所有可行的弱函数集合(基函数)；提升算法通过迭代的选择一个 负梯度方向 上的基函数来逐渐逼近 局部极小值 。 这种在函数域的梯度提升观点对机器学习的很多领域有深刻影响。

提升的理论意义：如果一个问题存在弱分类器，则 可以通过提升的办法得到强分类器。

8.2.2.1. Adboost详解

大部分的提升方法都是改变训练数据的概率分布(训练数据的权值分布)，针对不同的训练数据分布调用弱学习算法学习一系列弱分类器。这样提升方法就有两个问题需要解决： 一是，在每一轮如何改变训练数据的权重或概率分布；二是如何将弱分类器组合成一个强分类器。

Adboost的做法是：提高那些被前一轮弱分离器错误分类样本的权值，而降低那些被正确分类样本的权值。这样一来，那些没有得到正确分类的数据，由于其权重加大而受到后一轮的弱分类器的更大关注。 Adboost就是根据分类结果动态调整样本分布，巧妙地将这些想法实现在了一种算法里。

下面讲解AdaBoost算法，假定给定一个二分类的训练数据集:

$$T = (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

其中，每个样本由实例和标注组成，标记$y_i \in -1, +1$,AdaBoost算法实现如下:

(1)初始化训练数据的权值分布

$$D_1 = (w_{11}, \ldots, w_{1i}, \ldots, w_{1N}), \;\; w_{1i} = \frac{1}{N}, \;\; i = 1, 2, \ldots, N$$

(2)对$m = 1, 2, \ldots, N$ * 使用具有权值分布D_m的训练数据集学习，得到基本分类器

$$G_m(x) : \chi \to -1, +1, \;\; x_i \in \chi \subseteq R^n$$

- 计算$G_m(x)$在训练数据集上的分类误差率

$$e_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} I(G_m(x_i \neq y_i))$$

- 计算$G_m(x)$的系数

$$\alpha_m = \frac{1}{2} log \frac{1-e_m}{e_m}$$

这里的对数是自然对数。

- 更新训练数据集的权值分布

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2}, \cdots, w_{m+1,N})$$
$$w_{m+1,i} = \frac{w_{mi}}{Z_m} exp(-\alpha_m y_i G_m(x_i))$$

其中,Z_m是归一化因子

$$Z_m = \sum_{i=1}^{N} w_{mi} exp(-\alpha_m y_i G_m(x_i))$$

它使得D_{m+1}成为一个概率分布。

(3)构建基于分类器的线形组合

$$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

得到最终的分类器

$$G(x) = sign(f(x)) = sign(\sum_{m=1}^{M} \alpha_m G_m(x))$$

其中，$w_{m+1,j}$的更新规则可以写成:

$$w_{m+1,j} = \begin{cases} \frac{wmi}{Zm} e^{-\alpha m}, & G_m(x_i) = y_i \\ \frac{wmi}{Zm} e^{\alpha m}, & G_m(x_i) \neq y_i \end{cases}$$

定理: AdaBoost算法是前向分布加法算法的特例。这时，模型是由基本分类器组成的加法模型，损失函数是指数函数，证明参考 统计学习方法 .

8.2.2.2. GBDT详解

GBDT(Gradient Boosting Decision Tree) 又称为 MART(Multiple Additive Regression Tree),因此GBDT的树不仅是 CART 树，而且还是回归树。 GBDT与AdBoost最大的不同是：AdBoost利用前一轮弱学习器的误差率来更新训练集的权重，这样一轮一轮迭代下去；GBDT也是迭代，使用了前向分布算法，但是弱分类器限定只使用CART回归树模型， 同时每轮迭代的目标为找到一个CART回归树模型的弱学习器，让本轮的损失最小，也就是说，本轮迭代的决策树，要让本轮的损失最小。

GBDT采用梯度提升算法，利用最速下降法近似方法，其关键是利用损失函数的负梯度在当前模型的值作为回归问题提升树算法中的残差近似值，拟合一个回归树。

$$-[\frac{\partial L(y, f(x_i))}{\partial f(x_i)}]_{f(x)=f_{m-1}(x)}$$

GBDT算法推导

1)相关负号定义

f决策树，从数学上讲，决策树是一个分段函数，所以它的参数描述了分段方法，我们用R_{j1}^{J} 和 b_{j1}^{J}表示决策树的参数，前者是分段空间(决策树划分出的disjoint空间)，后者是这些空间上f输出的函数值。其中J是叶子节点的数量。

$$f(x;R_j,b_{j1}^{J})=\sum_{j=1}^{J}b_jI(x\in R_j) \qquad (8.1)$$

下文我们用$f(x)$来省略表示$f(x;R_j,b_{j1}^{J})$.

在 `Boosting` 框架中，f理论上可以是很多弱学习器，在 `Gradient Boosting` 框架里，常用且被证明有效的出了决策树外，还有逻辑回归，而xgboost中的gbtree和gblinear就对应这两种实现，下面的讨论仅限于决策树上。

F，决策树的ensemble，定义为:

$$F=\sum_{i=0}^{K}f_i$$

注意：弱学习器不可以是线形函数，因为多个线形函数集成到一起仍然是一个线形函数，等价于一个线形模型，表达能力有限，也不能再叫做集成学习了！

f_0为模型初始值，通常是按照一定原则计算出的常熟，同时定义$F_k=\sum_{i=0}^{k}f_i$.
$D=(x_i,y_i)_1^N$，为训练样本.
L，目标函数，定义为:

$$L=L(y_i,F(x_i)_1^N)=\underbrace{\sum_{i=1}^{N}L(y_i,F(x_i))}_{Trainning\ loss}+\underbrace{\sum_{k=1}^{K}\Omega(f_k)}_{Regularization} \qquad (8.2)$$

第一项L是针对样本的$Loss$，L可以有多种选择：绝对值误差，平方误差，logistic loss，指数误差等.
第二项Ω是正则化函数，它惩罚f_k的复杂度，树结构越复杂它的值越大，其对提升效果非常重要。

2)算法推导

先给出原始GBDT算法的框架

输入: $\{(x_i,y_i)\}_1^N$, K, L, $\cdots$
1. 初始化f_0
for $k=1$ **to** K **do**
- 2.1. $\tilde{y}_i=-\frac{\partial\mathcal{L}(y_i,F_{k-1}(x_i))}{\partial F_{k-1}}$, $i=1,2,\cdots,N$
- 2.2. $\{R_j,b_j\}_1^{J^*}=\arg\min_{\{R_j,b_j\}_1^J}\sum_{i=1}^{N}\left[\tilde{y}_i-f_k(x_i;\{R_j,b_j\}_1^J)\right]^2$
- 2.3. $\rho^*=\arg\min_\rho\mathcal{L}(\{y_i,F_{k-1}(x_i)+\rho f_k(x_i)\}_1^N)$
$=\arg\min_\rho\sum_{i=1}^{N}L(y_i,F_{k-1}(x_i)+\rho f_k(x_i))+\Omega(f_k)$
- 2.4. 令$f_k=\rho^*f_k$, $F_k=F_{k-1}+f_k$

end
输出: F_K

图 37. 基本GBDT算法

下面对基本的GBDT算法步骤进行解释。

- 初始化f_0，常用的方法有:
  - 随机初始化
  - 用训练样本中的充分统计量进行初始化

- 用其他模型的预测值初始化 GBDT很健壮，对初始值并不敏感，但是更好的初始值能够获得更快的收敛速度和质量。

- 2.1中$\tilde{y}_i$被称作 响应 ，它是一个和残差正相关的变量。
- 2.2公式背后表达了，使用平方误差训练一颗决策树f_k，拟合数据$(x_i, \tilde{y}_i)$
- 2.3进行line search,在2.2中的f是在平方误差下学到的，这一步进行一次line search，让f乘以步长ρ后最小化损失L
- 2.4 讲训练出来的f_k叠加到F

总体来说，GBDT就是一个不断拟合响应(残差)并叠加到F上的过程，在这个过程中，Loss不断接近最小值。

8.2.2.3. XGBoost详解

XGBoost可以认为是在GBDT基础上的优化，GBDT每次计算出响应后，都会生成一颗最小二乘CART回归树，然后根据损失函数优化树的步长，XGBoost直接对损失函数泰勒展开到二阶，直接求解。首先我们给出XGBoost的正则化函数:

$$\Omega(f_k) = \frac{\gamma}{2}J + \frac{\lambda}{2}\sum_{j=1}^{J} b_j^2 \qquad (8.3)$$

接上节，下面给出XGBoost的推导过程:

$$\begin{aligned}
L_k &= \sum_{i=1}^{N} L(y_i, F_{k-1}(x_i) + f_k(x_i)) + \Omega(f_k) \\
&= \sum_{i=1}^{N} L(y_i, F_{k-1} + f_k) + \Omega(f_k) \\
&\approx \sum_{i=1}^{N} (L(y_i, F_{k-1}) + \underbrace{\frac{\partial L(y_i, F_{k-1})}{\partial F_{k-1}}}_{:=g_i} f_k + \frac{1}{2}\underbrace{\frac{\partial^2 L(y_i, F_{k-1})}{\partial F_{k-1}^2}}_{:=h_i} f_k^2) + \Omega(f_k) \\
&= \sum_{i=1}^{N} (L(y_i, F_{k-1}) + g_i f_k + \frac{1}{2} h_i f_k^2) + \Omega(f_k) \\
&= \sum_{i=1}^{N} (L(y_i, F_{k-1}) + g_i \sum_{j=1}^{J} b_j + \frac{1}{2} h_i \sum_{j=1}^{J} b_j^2) + \frac{\gamma}{2}J + \frac{\lambda}{2}\sum_{j=1}^{J} b_j^2 \\
&\propto \sum_{i=1}^{N} (g_i \sum_{j=1}^{J} b_j + \frac{1}{2} h_i \sum_{j=1}^{J} b_j^2) + \frac{\gamma}{2}J + \frac{\lambda}{2}\sum_{j=1}^{J} b_j^2 \\
&= \sum_{x_i \in R_j} (g_i \sum_{j=1}^{J} b_j + \frac{1}{2} h_i \sum_{j=1}^{J} b_j^2) + \frac{\gamma}{2}J + \frac{\lambda}{2}\sum_{j=1}^{J} b_j^2 \\
&= \sum_{j=1}^{J} (\sum_{x_i \in R_j} g_i b_j + \sum_{x_i \in R_j} \frac{1}{2} h_i b_j^2) + \frac{\gamma}{2}J + \frac{\lambda}{2}\sum_{j=1}^{J} b_j^2 \\
&= \sum_{j=1}^{J} (\sum_{x_i \in R_j} g_i b_j + \frac{1}{2}(\sum_{x_i \in R_j} h_i + \lambda) b_j^2) + \frac{\gamma}{2}J \\
&= \sum_{j=1}^{J} (G_j b_j + \frac{1}{2}(H_j + \lambda) b_j^2) + \frac{\gamma}{2}J \qquad (8.4)
\end{aligned}$$

对上述推导过程说明如下:

- XGBoost通过在f_k处进行二阶泰勒展开，来近似经验损失

- 令$G_j = \sum_{xi \in R_j} g_i, H_j = \sum_{xi \in R_j} h_i$

现在问题来了，如何同时求解R_j和b_j，为了解决这个问题，把问题分成两个子问题:

- 问题1:如果已经得到了R_j，最小化L_k的b_j是多少?
- 问题2:如果将当前节点R_i分裂，应该在哪一个分裂点使得L_k最小，这一步我们称为树分裂算法

问题1:

对公式(8.4)的b_j求导，令结果为零，容易求得:

$$b_j^* = -\frac{G_j}{H_j + \lambda} \tag{8.5}$$

此时最小的L_k是:

$$L_k^* = -\frac{1}{2}\sum_{j=1}^{J}\frac{G_j^2}{H_j + \lambda} + \frac{\gamma}{2}J \tag{8.6}$$

问题2:

求R_j的方法不同于b_j，因为它是对输入x所属空间的一种划分方法，不连续，无法求导。 精确得到划分是一个NP问题，通常使用贪心算法，即分裂某节点时， 只考虑对当前节点分裂后，哪个分裂方案能得到最小的损失。

GBDT中的CART树使用这种贪心方法时是需要遍历x的每个维度的每个分裂点，XGBoost在这个过程中有所优化，后面详解介绍。那么问题2就可以形式的描述为：将当前节点R_j分裂成R_L和R_R，使得分裂后整个树的损失最小。

由于整棵树的损失等于每个叶子节点上的损失之和，而整个分裂过程仅涉及到三个节点，其他任何节点的损失不变，因此这个问题又等价于下列形式:

$$max_{RLRR}\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma \tag{8.7}$$

其中γ为增加了叶子节点后带来的模型复杂度惩罚项，公式(8.7)表示的是每轮叶子节点分裂时整个树最小损失的降低量，显然这个值越大越好。如果公式(8.7)的值小于0，则不能分裂。

下面给出树分裂算法的处理逻辑:

输入: R_j, 落在R_j的训练样本$\{(x_i^{(j)}, y_i^{(j)})\}_1^N, \cdots,$ 其中$x_i^{(j)} \in \mathbb{R}^m$

decr $= 0$

$G = \sum_{i=1}^N g_i$, $H = \sum_{i=1}^N h_i$

for $k = 1$ **to** m **do**

  $G_L = 0$, $H_L = 0$

  for l *in uniq(sorted*($\{x_{ik}^{(j)}\}$)) **do**

    $G_L = G_L + g_l$, $H_L = H_L + h_l$

    $G_R = G - G_L$, $H_R = H - H_L$

    decr $= \max\left(\text{decr}, \frac{G_L^2}{H_L+\gamma} + \frac{G_R^2}{H_R+\gamma} - \frac{(G_L+G_R)^2}{H_L+H_R+\gamma}\right)$

  end

end

if *decr* < 0 *or* 满足终止条件 **then**

  输出: 保持R_j不分裂

end

输出: 按照最大decr的方案将R_j分裂成R_L和R_R

图 38. 树分裂算法

8.2.2.4. 总结

关于XGBoost可以说的内容很多，下面列举说明部分注意事项:

- XGBoost如何处理缺省值？ 基于公式(8.7)分裂的评分函数，我们还可以用来处理缺失值。处理的方法就是，我们把缺失值部分额外取出来，分别放到IL和IR两边分别计算两个评分，看看放到那边的效果较好，则将缺失值放到哪部分。
- 除了以上提到了正则项以外，我们还有shrinkage与列采样技术来避免过拟合的出现。所谓shrinkage就是在每个迭代中树中，对叶子结点乘以一个缩减权重eta。该操作的作用就是减少每颗树的影响力，留更多的空间给后来的树提升。
- 另一个技术则是采样的技术，按层随机抽样的意思就是，之前每次分裂一个结点的时候，我们都要遍历所有的特征和分割点，从而确定最优的分割点，那么如果加入了列采样，我们会在对同一层内每个结点分裂之前，先随机选择一部分特征，于是我们只需要遍历这部分的特征，来确定最优的分割点。
- 而行采样则是bagging的思想，每次只抽取部分的样本进行训练，而不使用全部的样本，从而增加树的多样性。

8.2.2.5. 深度思考

1. (1)xgboost相比传统gbdt有何不同？xgboost为什么快？xgboost如何支持并行？

- 传统GBDT以CART作为基分类器，xgboost还支持线性分类器，这个时候xgboost相当于带L1和L2正则化项的逻辑斯蒂回归（分类问题）或者线性回归（回归问题）
- 传统GBDT在优化时只用到一阶导数信息，xgboost则对代价函数进行了二阶泰勒展开，同时用到了一阶和二阶导数。顺便提一下，xgboost工具支持自定义代价函数，只要函数可一阶和二阶求导
- xgboost在代价函数里加入了正则项，用于控制模型的复杂度。正则项里包含了树的叶子节点个数、每个叶子节点上输出的score的L2模的平方和。从Bias-variance tradeoff角度来讲，正则项降低了模型的variance，使学习出来的模型更加简单，防止过拟合，这也是xgboost优于传统GBDT的一个特性
- Shrinkage（缩减），相当于学习速率（xgboost中的eta）。xgboost在进行完一次迭代后，会将叶子节点的权重乘上该系数，主要是为了削弱每棵树的影响，让后面有更大的学习空间。实际应用中，一般把eta设置得小一点，然后迭代次数设置得大一点。
- 列抽样（column subsampling）。xgboost借鉴了随机森林的做法，支持列抽样，不仅能降低过拟合，还能减少计算，这也是xgboost异于传统gbdt的一个特性
- 对缺失值的处理。对于特征的值有缺失的样本，xgboost可以自动学习出它的分裂方向
- xgboost工具支持并行。boosting不是一种串行的结构吗?怎么并行的？注意xgboost的并行不是tree粒度的并行，xgboost也是一次迭代完才能进行下一次迭代的（第t次迭代的代价函数里包含了前面t-1次迭代的预测值）。xgboost的并行是在特征粒度上的。我们知道，决策树的学习最耗时的一个步骤就是对特征的值进行排序（因为要确定最佳分割点），xgboost在训练之前，预先对数据进行了排序，然后保存为block结构，后面的迭代中重复地使用这个结构，大大减小计算量。这个block结构也使得并行成为了可能，在进行节点的分裂时，需要计算每个特征的增益，最终选增益最大的那个特征去做分裂，那么各个特征的增益计算就可以开多线程进行
- 可并行的近似直方图算法。树节点在进行分裂时，我们需要计算每个特征的每个分割点对应的增益，即用贪心法枚举所有可能的分割点。当数据无法一次载入内存或者在分布式情况下，贪心算法效率就会变得很低，所以xgboost还提出了一种可并行的近似直方图算法，用于高效地生成候选的分割点

最后奉上一张XGBoost优化点的图，关于XGBoost的话还有很多，后续继续补充.

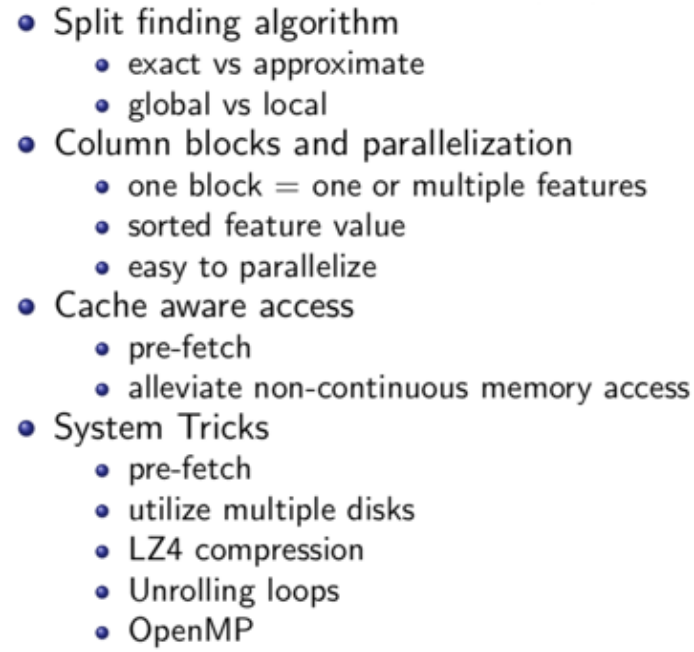


图 39. XGBoost 优化点

9. 推荐系统

10. 聚类和近邻算法

11. 贝叶斯理论

12. 隐马尔可夫模型

13. 条件随机场

14. 深度学习概论

15. 深度学习中的优化算法

16. 深度学习中的正则化

17. 深度神经网络

18. 卷积神经网络

19. 循环神经网络

20. 神经网络番外篇

21. 生成对抗网络

22. 迁移学习

23. 强化学习

24. 数学之美

24.1. 数学中的空间

24.1.1. 空间关系

研究数学的学者对各种空间完全可以驾轻就熟，但是对于计算机专业又是刚开始搞机器学习的同学来说，未必能很好的掌握。比如：什么是赋范线性空间、内积空间，度量空间，希尔伯特空间？

现代数学的一个特点就是以集合为研究对象，这样的好处就是可以将很多不同问题的本质抽象出来，变成同一个问题，当然这样的坏处就是描述起来比较抽象，很多人就难以理解了。

既然是研究集合，每个人感兴趣的角度不同，研究的方向也就不同。为了能有效地研究集合，必须给集合赋予一些“结构”（从一些具体问题抽象出来的结构）。 从数学的本质来看，最基本的集合有两类：线性空间（有线性结构的集合）、度量空间（有度量结构的集合）。

对线性空间而言，主要研究集合的描述，直观地说就是如何清楚地告诉地别人这个集合是什么样子。为了描述清楚，就引入了基（相当于三维空间中的坐标系）的概念， 所以对于一个线性空间来说，只要知道其基即可，集合中的元素只要知道其在给定基下的坐标即可。但线性空间中的元素没有“长度”（相当于三维空间中线段的长度）， 为了量化线性空间中的元素，所以又在线性空间引入特殊的“长度”，即范数。赋予了范数的线性空间即称为赋犯线性空间。但赋范线性空间中两个元素之间没有角度的概念，为了解决该问题，所以在线性空间中又引入了内积的概念。因为有度量，所以可以在度量空间、赋范线性空间以及内积空间中引入极限，但抽象空间中的极限与实数上的极限有一个很大的不同就是， 极限点可能不在原来给定的集合中，所以又引入了完备的概念，完备的内积空间就称为Hilbert空间。

这几个空间之间的关系是：线性空间与度量空间是两个不同的概念，没有交集。赋范线性空间就是赋予了范数的线性空间，也是度量空间（具有线性结构的度量空间）内积空间是赋范线性空间希尔伯特空间就是完备的内积空间。

24.1.2. 希尔伯特空间

做个类比,一般的3D矢量空间(我们最常见的)和Hilbert空间.在3D的矢量空间中, 基底是i,j,k. 维度是3(有限维). 这三个基本的基矢量是完备的(矢量空间中任何一个元素都可以用这3个基底展开,系数唯一), 正交的(不同的基底做点积为0.).

矢量空间中的任意两个元素之间可以定义算符F, 也就是操作. 我们常常对保持元素A长度(自己和自己点积,A*A)不变的操作感兴趣, 这样的操作形象上讲是转动, 抽象些讲是满足$F^2 = 1$的操作,或者叫变换.

好了, 再看看Hilbert空间, 基底一般是函数,常见的是含有各种频率的平面波函数,一种频率对应一个基底 维度是无穷.这些基底, 即平面波函数是完备的(Hilbert空间中的任何元素都可以用平面波函数展开, 其实就是指傅里叶变换), 正交(平面波函数做"点积"为delta函数).

Hilbert空间中任意两元素也可以定义算符G, 也就是操作. 我们常常对保持元素"长度"(自己和自己"点积")不变的操作感兴趣. 由于Hilbert空间是复数域上的,常见的3D矢量空间是实数域上的, 所以$G^2 = 1$ 的G 和F 是不同的, 虽然表达式相同.

建立Hilbert空间的目的是为量子力学中的计算提供强有力的数学基础, 也方便了抽象出其中更本质的运算.包括之后进行的关于对称性的讨论, 都是定义在Hilbert空间上的.注意上面的论述中并没有涉及到矩阵. 因为矩阵其实只是抽象定义的一种表现, 或者说是抽象的定义的一种表示(representation)而已. 这是群论的思想.所有这些后面发展起来的表示理论, 在Hilbert空间中表示后,(尤其是算符的表示)显得非常重要.

25. 深度学习框架

26. 走进互联网

最后更新时间 2018-07-22 22:26:33 CST