> 原文地址 https://www.pkudodo.com/2018/12/05/1-7/， 作者 Dodo

github：

[GitHub | 手写实现李航《统计学习方法》书中全部算法]()

相关博文：

1. [统计学习方法 | 感知机原理剖析及实现]()
2. [统计学习方法 | K 近邻原理剖析及实现]()
3. [统计学习方法 | 朴素贝叶斯原理剖析及实现]()
4. [统计学习方法 | 决策树原理剖析及实现]()
5. [统计学习方法 | 逻辑斯蒂原理剖析及实现]()
6. [统计学习方法 | 最大熵原理剖析及实现]()

> 哪怕整书已经看过一遍了，我仍然认为最大熵是耗费我时间较多的模型之一。主要原因在于千篇一律的博客以及李航的《统计学习方法》在这一章可能为了提高公式泛化性而简化的关键点。当然了，在阅读上没有问题，可是在程序的编写上可能是我历时最久的。如果读者需要编写最大熵模型，比较建议使用我的博客配合代码学习，忘掉书上的这一章吧。

最大熵的直观理解

为了引出最大熵，我们可能需要举一个所有博客都会举的例子：如果我手里拿着一个骰子，想问你扔下去后是每一面朝上的概率是多少？所有人都能抢答出来是 1/6。那么为什么是 1/6 呢？为什么不是 1/2 或者 1/3？

因为六个面的概率相同呀

emmm.... 我暂时承认你说的有点道理。可是如果这是老千手里的骰子呢？你还会答 1/6 吗？

可是你没说是老千手里的骰子呀

你让我哑口无言了，但为什么大家在猜之前不会去假设是不是老千的骰子这种情况呢？

因为你没说是老千手里的骰子呀

完蛋了，又绕回来了。不过我想想也是，如果要考虑老千，那么也可能要考虑骰子是不是破损了，桌面是不是有问题...... 完蛋了，没头了。

所以 1/6 最保险呀

如果我告诉你 1 朝上的概率是 1/2 呢？

那剩下的就是 1/10 呀

emmm... 我承认在目前的对话中你是明智的，而我像个低龄的儿童。但是我要告诉你，上面几句对话，其实就是最大熵。
当我们在猜测概率时，不确定的部分我们都认为是等可能的，就好像骰子一样，我们知道有六个面，那么在概率的猜测上会倾向于 1/6，也就是等可能，换句话讲，也就是倾向于均匀分布。为什么倾向均匀分布？你都告诉我了，因为这样最保险呀！当我们被告知 1 朝上的概率是 1/2 时，剩下的我们仍然会以最保险的形式去猜测是 1/10。是啊，最大熵就是这样一个朴素的道理：

凡是我们知道的，就把它考虑进去，凡是不知道的，通通均匀分布！

从另一个角度看，均匀分布其实保证的是谁也不偏向谁。我说万一是老千的骰子呢？你说万一骰子有破损呢？骰子好端端地放在那咱俩地里咕噜瞎猜啥呢。咱们与其瞎猜是啥啥啥情况，干脆就均匀分布，谁也不偏向谁。

多朴素的道理啊！

朴素贝叶斯分类器的数学角度（配合《统计学习方法》食用更佳）

> 在最大熵章节中我们在公式上仍然参考《统计学习方法》，除此之外会补充一些必要知识点，建议阅读完毕后浏览一下代码知道详细的用法。

最大熵模型是一个分类模型，这样，我们照旧不管中间过程，统统都不管，我们最终可能会想要这样一个式子： P(Y|X)。是啊，P(Y|X) 就像一个终极的大门一样，扔进入一个样本 x，输出标签 y 为各种值的概率。我们现在手里有一堆训练数据，此外手握终极大式子 P(Y|X)，是不是想到了朴素贝叶斯？我们对训练数据进行分析，获得先验概率，从而得到 P(Y|X) 模型进行预测。最大熵也有数据和相同的终极大式子，那么

最大熵和朴素贝叶斯有什么联系?

可以说在本质上是非常密切的，但具体的联系和区别，我希望读者正式了解了最大熵以后，我们一起放在文章末尾讨论。

文章的最初，我们需要给出最大熵公式，如果对下面这个式子不太熟悉的，可以查看[《统计学习方法 | 决策树原理剖析及实现》]一节：

$$H(P) = -\sum_{x} P(x)\log P(x)$$

H(P) 表现为事件 P 的熵变，也称为事件 P 的不确定性。对于最大熵模型而言，我们一直记得那个终极式子 P(Y|X)，如果放到骰子例子中，P(Y=1|X) 表示扔了一个骰子，1 朝上的概率，P(Y=2|X) 表示 2 朝上的概率。我们之前怎么说的？我们只知道骰子是六面，至于其他情况我们一概不知，在不知道的情况下，我们就不能瞎做推测，换言之，我们就是要让这个 P(Y|X) 的不确定性程度最高。

不确定性程度最高?

可能有些读者这么一绕没回过神来，如果我告诉你点数 1 朝上的概率是 1/2，你是不是对这个概率分布开始有了一点认识，那么会将剩下的面认为是 1/10（注：1 朝上概率是 1/2，那么 1 背面的概率一定会降低，但我们没有考虑这件事情，事实上我们不应该添加任何的主观假设，因为实际上有些骰子 1 的背面是 3，也有些是 4，任何主观假设都有可能造成偏见产生错误的推断）。你推断剩余面是 1/10 的概率是因为我告诉了你信息，你对这个骰子开始有了一些确定的认识，也就是信息的不确定性程度——熵减少了。但事实上我没有告诉你这一信息，你对 P(Y|X) 的分布是很不确定的，那么我们能不能用使用下面这个式子，让 H(P) 最大化来表示我们对 P(Y|X) 的极大不确定呢？事实上当我们令下式中 H(P) 最大时，P(Y|X) 正好等于 1/6。也就是说，如果我们要让熵最大，P(Y|X) 必须为 1/6。换句话说，只有我们把骰子的每一面概率当做 1/6 时，才能充分表示我们对骰子每一面的概率分布的极度不确定性，骰子的概率这件事情的混乱度最高，同时也传递了一个信息：我们对这件事情没有添加任何主观的假设，对任何未知的东西没有偏见和主观假象，这是最保险、最安全的做法。

$$H(p) = -\sum_{y} P(y|x) \log P(y|x)$$

上式可以理解成 x = 托马斯全旋法扔骰子，y 为各类结果的概率。那么我们并不是说 x 只有托马斯全旋法这一种，很可能还有王思聪吃热狗法扔骰子、边骂作者啥也不懂边扔骰子法等等等等.... 我们要使得在所有情况下熵都是最大，所以引出下式（条件熵，可参考决策树章节）：

$$H(P) = -\sum_{x,y} \tilde{P}(x) P(y \mid x) \log P(y \mid x)$$

加了一个 P_(x) 是什么意思？就是说我在求每种 x 对应的 y 时，要考虑到 x 出现的概率进行相乘。换句话说，也可以认为是当前 P(y|x) 的权值。因为我们要保证在所有情况下总的熵是最大的，所以每个部分单独的熵乘以权值后再相加，就是整个事件的熵。上式这个模型，其实就是最大熵模型，我们要让 P(y|x) 的熵最大，等价于让 H(p) 最大，我们好像获得了一个新的终极大式子。目前来看，只要能够让 H(P) 为最大值，那就能获得模型啦。

emmm.... 不过里面的各个子式好像也不是那么好求的。所以我们暂时先忘记上面的式子，好好回想一下咱们手里有哪些信息，好好整理一下。

我认为我手里应该会有一个训练集，包含所有的样本以及样本对应的标签（训练集都没有的话啥模型也训练不了啊，那可太惨了）。

那么我们首先可以得到训练集中 x，y 的联合概率分布以及 x 的边缘分布，数据都在我们手上，这两个很简单就能求出来对吧。

$$\tilde{P}(X = x, Y = y) = \frac{v(X = x, Y = y)}{N}$$

$$\tilde{P}(X = x) = \frac{v(X = x)}{N}$$

在上式中 v 表示频数，第一个式子的分子也就表示 (x, y) 组合出现的次数，处于总数 N 即为 (x, y) 组合出现的概率，也就是 P(X, Y)。第二个式子同理，求得 P(X=x)的概率。那么现在 P(X, Y)和 (X) 我们知道了，有什么用呢？emmm.... 暂时好像还没什么头绪，我们看下式，书中说这是特征函数 f(x, y)关于经验分布 p(x, y)的期望值，f(x, y)在 x 和 y 满足某一事实时结果为 1，反之输出 0。比如说训练集中某一样本记录是托马斯回旋法扔骰子，结果是 5 朝上。那么 $f(x=\text{托马斯回旋法扔骰子}, Y=5)=1$。其实更简单的理解，可以认为如果 (x, y) 出现在训练集中，那么它就是 1，因为一旦出现在训练集中，说明 (x, y) 对已经符合了某一样本的事实了。P 上面的短横表示这个模型是基于训练集得出来的，只能被称为经验分布。

$$E_{\tilde{P}}(f) = \sum_{x,y} \tilde{P}(x, y) f(x, y)$$

于此同时我们可以直接将 P_(x, y) 拆开来，拆成下式：

$$\bar{E}_{\bar{P}}(f) = \sum_{x,y} \bar{P}(x) \bar{P}(y|x) f(x,y)$$

那么我们将 P_(x, y) 转换成了 P_(x)*P_(y|x)，发现什么了没有？里面有 P_(y|x)，这个和我们要得到的最终大式子 P(Y|X) 很像，那么如果我们只是将其替换进去变成下面这样会有什么结果呢？

$$E_P(f) = \sum_{x,y} \tilde{P}(x) P(y \mid x) f(x, y)$$

我们发现，如果我们的预测模型 P(y|x) 是正确的话，那 P(y|x) 应该等于 p_(y, x)，也就是说

$$E_P(f) = E_{\tilde{P}}(f)$$

这两个应该相等！或者说：

$$\sum_{x,y} \tilde{P}(x)P(y|x)f(x,y) = \sum_{x,y} \tilde{P}(x,y)f(x,y)$$

那这个有啥用？emmm.... 我们将我们目前知道的信息整理出来了上式，老实说，目前来看没啥用.....

咦？等等，我们换种思维想一下，我们知道如果做出来的模型足够正确的话，那应该上面的两个式子相等，那.... 这算不算一种约束？约束我一定要得到正确的模型？还记得我们要最大化的那个熵的式子吗？这个代表了我们要求得的模型。

$$H(P) = -\sum_{x,y} \tilde{P}(x)P(y|x)\log P(y|x)$$

那加上我们刚刚整理出来的约束：

$$\begin{aligned}
\max_{P \in \mathbf{C}} \quad & H(P) = -\sum_{x,y} \tilde{P}(x)P(y|x)\log P(y|x) \\
\text{s.t.} \quad & E_P(f_i) = E_{\tilde{P}}(f_i), \quad i = 1,2,\cdots,n \\
& \sum_y P(y|x) = 1
\end{aligned}$$

是不是看起来有点像那么回事了，第三个式子也是一个约束，概率和一定为 1 嘛。在机器学习中不止一次见到了上图的式子形式，但通常是求 min 而非上图的 max，那么我们再转换一下，求最大取个负号就是求最小了嘛。

$$\begin{aligned}
\min_{P \in \mathbf{C}} \quad & -H(P) = \sum_{x,y} \tilde{P}(x)P(y|x)\log P(y|x) \\
\text{s.t.} \quad & E_P(f_i) - E_{\tilde{P}}(f_i) = 0, \quad i = 1,2,\cdots,n \\
& \sum_y P(y|x) = 1
\end{aligned}$$

好了，现在是不是知道该怎么做了？转换成拉格朗日乘子法直接求最小值 H(P) 呀。

$$L(P,w) \equiv -H(P) + w_0\left(1-\sum_y P(y|x)\right) + \sum_{i=1}^n w_i(E_{\tilde{P}}(f_i) - E_P(f_i))$$

$$= \sum_{x,y} \tilde{P}(x)P(y|x)\log P(y|x) + w_0\left(1-\sum_y P(y|x)\right)$$

$$+ \sum_{i=1}^n w_i\left(\sum_{x,y} \tilde{P}(x,y)f_i(x,y) - \sum_{x,y} \tilde{P}(x)P(y|x)f_i(x,y)\right)$$

但是我们并不能直接求到 H(P) 的最优值，在使用拉格朗日乘子法后引入了变量 w，对于 w，我们需要先求得 L(P,W) 的最大值，此时将 P 看成了一个常数。之后再对 P 操作求得 - H(P) 的最小值。也就是说：

$$\min_{P \in C} \max_w L(P,w)$$

一定要注意 min 和 max 下方的参数是不一样的，因为 L 是考虑到约束函数从生成的目标桉树，在求解最优的参数 w 以此使得我们的模型能够符合约束时 P 只是一个定值，此时需要求得最优的 w 以此保证约束条件对模型的约束性。当求得 w 后 P 满足了约束性，此时再求最优的 P 以此使得 - H(p) 最小化，也就是 H(p) 最大化以此得到最优模型。

与此同时，我们可以将上式转换为下式，因为 L 是 P 的凸函数，因此两式等价。

$$\max_w \min_{P \in C} L(P,w)$$

所以我们先求内部的极小化问题，即 min L(P， w)，因为 L 是此时是关于 P 的凸函数，w 是一个定值，所以可以直接对 P 求导求得最小值。H(P) 中的 P 是什么？ P 就是 P(y|x) 啊（前文在 H(P) 的式子中可以显然看到，事实上我们全文也一直在为 P(y|x) 进行准备）。因为 L 是 P 的凸函数，那么直接求导并且令求导结果为 0 就好啦，从而得到下式。

$$P(y|x) = \exp\left(\sum_{i=1}^n w_i f_i(x,y) + w_0 - 1\right) = \frac{\exp\left(\sum_{i=1}^n w_i f_i(x,y)\right)}{\exp(1-w_0)}$$

我们再整理一下：

$$P_w(y|x)=\frac{1}{Z_w(x)}\exp\left(\sum_{i=1}^{n}w_if_i(x,y)\right)$$

$$Z_w(x)=\sum_y\exp\left(\sum_{i=1}^{n}w_if_i(x,y)\right)$$

上式没有什么区别，只是把分母变成了 Zw（因为所有 y 对应的 P(y|x)求和为1，所以 exp(1-wo) = Zw），从而看起来简化不少。那么我们已经得到最优的 P(y|x) 了，将其代入 L 式中再求最优的 w 不就好了嘛，这个步骤也就是求个导，书上不详细展开了，因为是重复性工作，因此在这里我觉得也没有必要再讲求导式写出来，读者可以自己写一下。

因此综上，只要能够得到最优的 Pw(y|x)，随后再求得最优的 w，全部工作也就迎刃而解了。最主要的是 Pw(y|x) 问题。对于如何求解最优的 Pw(y|x)，这里介绍改进的迭代尺度法 (IIS)

改进的迭代尺度法 (IIS)

我们已知目前的解决目标是：

$$P_w(y|x)=\frac{1}{Z_w(x)}\exp\left(\sum_{i=1}^{n}w_if_i(x,y)\right)$$

$$Z_w(x)=\sum_y\exp\left(\sum_{i=1}^{n}w_if_i(x,y)\right)$$

所有的式子连乘取对数转换为对数似然函数为：

（在《统计学习方法》6.2.4 节中说明了为何上文中关于 w 进行拉格朗日极大化等价于对数似然函数最大化，但由于只是一些简单的公式，这里不再展开。唯一难点在于书中给出的对数似然形式，可以查看该链接学习：最大熵模型中的对数似然函数的解释)

$$L(w)=\sum_{x,y}\tilde{P}(x,y)\sum_{i=1}^{n}w_if_i(x,y)-\sum_x\tilde{P}(x)\log Z_w(x)$$

数学家说我们只要求对数似然函数 L(w) 的极大值，就一定能够得到最优解。确实是这样的，但是具体怎么求呢？我们找极大值一般使用的是求导得到导数为 0 的点，我没有试过上式对 w 进行求导能否得到解（应该是不行的，如果可以的话，也就不需要 IIS 法了）。IIS 法的核心思想是每次增加一个量δ，使得 L(w+δ)>L(w)，以此不断提高 L 的值，直到达到极大值。我们将 L(w+δ) – L(w) 写出来：

$$L(w+\delta)-L(w)=\sum_{x,y}\tilde{P}(x,y)\log P_{w+\delta}(y|x)-\sum_{x,y}\tilde{P}(x,y)\log P_w(y|x)$$
$$=\sum_{x,y}\tilde{P}(x,y)\sum_{i=1}^{n}\delta_i f_i(x,y)-\sum_x\tilde{P}(x)\log\frac{Z_{w+\delta}(x)}{Z_w(x)}$$

那么我们要干啥？当然得保证差值大于 0 呀。只有每次都比前一次的大才能保证不停地往上走。除此之外呢？我们还希望每次增加的量都尽可能大一点，这样才能以最快的速度收敛到极大值。利用不等式 -lnα≥1-α我们可以得到下式：

$$L(w+\delta)-L(w)\geqslant\sum_{x,y}\tilde{P}(x,y)\sum_{i=1}^{n}\delta_i f_i(x,y)+1-\sum_x\tilde{P}(x)\frac{Z_{w+\delta}(x)}{Z_w(x)}$$
$$=\sum_{x,y}\tilde{P}(x,y)\sum_{i=1}^{n}\delta_i f_i(x,y)+1-\sum_x\tilde{P}(x)\sum_y P_w(y|x)\exp\sum_{i=1}^{n}\delta_i f_i(x,y)$$

将右端记为

$$A(\delta|w)=\sum_{x,y}\tilde{P}(x,y)\sum_{i=1}^{n}\delta_i f_i(x,y)+1-\sum_x\tilde{P}(x)\sum_y P_w(y|x)\exp\sum_{i=1}^{n}\delta_i f_i(x,y)$$

于是有

$$L(w+\delta)-L(w)\geqslant A(\delta|w)$$

A 也可以称之为该变量的下届，就是对于任意的δ，它们的差值一定是大于等于 A(δ|w) 的。我们应该求一个δ使得尽量增大 A(δ|w)，这样能保证我们每步都会走得尽可能跨度大，程序耗时也就越少。此外由于直接求 A 的最大值并不太好求，因此再将 A 松开一点，建立一个不太紧的方便求极大值的下界 B，书中有详细公式推导，并不太难，因此不再讲解。给出最终的式子：

求 $B(\delta|w)$ 对 δ_i 的偏导数：

$$\frac{\partial B(\delta|w)}{\partial\delta_i}=\sum_{x,y}\tilde{P}(x,y)f_i(x,y)-\sum_x\tilde{P}(x)\sum_y P_w(y|x)f_i(x,y)\exp(\delta_i f^{\#}(x,y)) \tag{6.32}$$

在式 (6.32) 里，除 δ_i 外不含任何其他变量．令偏导数为 0 得到

$$\sum_{x,y}\tilde{P}(x)P_w(y|x)f_i(x,y)\exp(\delta_i f^{\#}(x,y))=E_{\tilde{P}}(f_i) \tag{6.33}$$

于是，依次对 δ_i 求解方程 (6.33) 可以求出 δ．

这就给出了一种求 w 的最优解的迭代算法，即改进的迭代尺度算法 IIS．

对 B 求导并令其为 0，得到最优的δ，从而每次迭代过程中 w=w+δ，以此提高 L 值，得到最优解。这就是算法的全过程。

补充（极其重要——针对 mnist 数据集）：

mnist 是一个手写数字的数据集，里面有很多的样本，每个样本是 28*28 的图片，展开是一个 784 维的向量，代表一个数字，同时给了一个标签告诉你标签时多少。

好的，那么我的问题是：我们全文一直在说 P(y|x)，可是你真正考虑过在 mnist 中里面的 y 和 x 都是什么吗？

有人说 x 是每一个样本，y 是标签。看起来确实是这样的，那 f(x, y) 是什么？就是样本 x 和 y 是否成对出现过吗？那我如果出现一个训练集中没出现的样本，就变成 0 了吗？我们都知道手写每个人都不一样，不可能写出来的样本在训练集中一定有一个一模一样的，那么它就变成 0 了吗？

接下来我要说的事情，只针对 Mnist 数据集，作者进入机器学习时间尚短，不清楚书中是否是为了提高公式在所有情况下的泛化性而简略了写，因此对于其他方向的使用，保留一些意见。

事实上，对于 Mnist 而言所有的 x 都不是样本 x！它表示的是样本的特征。以 f(x, y) 为例，或许应该写成 f0(x, y)、f1(x, y), …., fn(x, y)，其中 n 为特征的数目。再详细下去以 f0(x, y) 为例，f0(x = 0, y = 0) 表示是否存在这样一个事实：训练集所有样本的特征中有样本的第 0 个特征值为 0，与此同时 y 为 0。是不是有点理解了？

我们再以 P(y|x) 为例，例如 P1(y=0|x=0) 表示的是在训练集中所有样本中，第一个特征为 0 的样本其中 y 为 0 的概率。

我当时也是写程序的时候卡在这里总是不知道该怎么写，直到看了别人的实现以后才发现了这个没有提到的地方，关于具体实现可以查看我的代码，有详细注释。

最后还是需要补充一点，我并不太清楚在别的地方（例如 NLP）中最大熵是如何应用的，也不清楚上面提到的这个小 tip 是否只是针对 mnist 而言做出的一个特例修改，因此保留对于这件事情的意见。

关于最大熵和朴素贝叶斯的联系

两者都使用了 P(y|x)，区别在于朴素贝叶斯使用先验概率求解，最大熵利用最大化条件熵求解。

朴素贝叶斯倾向于从基本信息中提取新的信息，而最大熵将提取信息的程度进行了量化（就好像强迫自己获得概率一定是要均匀分布一样）。