Supervised learning

笔记：Alvin

2017-07-05

目录

1 生成学习算法 Generative Learning algorithms

迄今为止，我们已经讨论了关于 $p(y|x;\theta)$ 模型（给定 x 下的 y 的条件分布）的学习算法。例如当 $h_\theta = g(\theta^T x)$ 时（g 是一个 sigmoid 函数），建立逻辑回归模型 $p(y|x;\theta)$。在这次的讲义中，我们将要讨论一个不同种类的学习算法。

考虑一个分类问题，已知一个动物的一些特征，我们想要去学习分辨出该动物是大象（y=1）还是小狗（y=0）。给定一个训练集，诸如线性回归或者感知器算法基本上是尝试去找到一条直线，这条直线可以看作是一个边界线，将大象和小狗在图像上分开。然后通过检查新的动物落在哪一边从而确定是大象还是小狗，从而使它的预测有据可循。

这里自然是给出另一种方法。首先观察大象，我们可以先建立一个大象看起来是什么样子的模型，然后同样的，观察小狗然后建立一个关于小狗的模型。最后通过与它们匹配来确定到底新的动物是更像大象还是更像小狗。

那些想要直接学习出 $p(y|x)$（如 logistic 回归），或者像学习出从输入集 $\mathcal{X}$ 到标签 $\{0,1\}$ 的映射的算法（如感知器算法）叫做判别学习算法（discriminative learning algorithms）。这里我们讨论的一种建立 $p(x|y)$（和 $p(y)$）的算法，这些算法叫做生成学习算法（generative learning algorithms）。例如，如果 y 代表样本是小狗 (0) 还是大象 (1)，然后 $p(x|y=0)$ 建立了小狗的特征的分布，而 $p(x|y=1)$ 建立了大象的特征的分布。

在建立完 $p(x|y)$ 和 $p(y)$（称作类的先验）后，我们就可以利用贝叶斯规则来得到 x 下 y 的后验分布：

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

因此分母为 $p(x) = p(x|y=0)p(y=0) + p(x|y=1)p(y=1)$，当然，由于我们计算 $p(y|x)$ 是为了预测，因此实际上我们没有必要计算分母值的，因为：

$$\begin{aligned} arg\ \max_y p(y|x) &= arg\ \max_y \frac{p(x|y)p(y)}{p(x)} \\ &= arg\ \max_y p(x|y)p(y) \end{aligned}$$

1.1 高斯判别分析 Gaussian discriminant analysis

我们将要观察的第一个生成的学习算法是高斯判别算法（GDA）。在这个模型中，我们假设 $p(x|y)$ 是一个多元正态分布。在讲到 GDA 之前，先简单介绍一下多元正态分布的特点。

1.1.1 多元正态分布

n 维的多元正态分布也叫做多维高斯分布，有一个参数平均值（期望）向量 $\mu in\mathcal{R}^n$ 以及一个协方差矩阵（covariance matrix）$\sum \in \mathcal{R}^{n\times n}$，并且这里的 $\sum > 0$，是对称正定的。也写成 $\mathcal{N}(\mu, \sum)$，它的等价形式为：

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

在上面的公式中 $|\sum|$ 代表的是矩阵 $\sum$ 行列式。

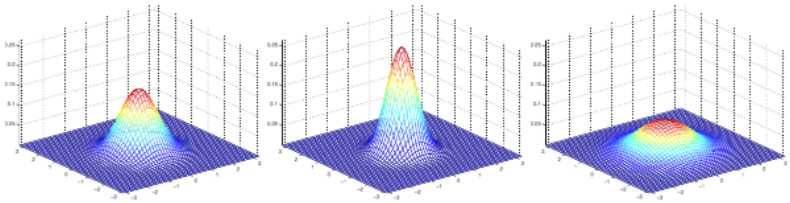
对于分布为 $\mathcal{N}(\mu, \sum)$ 的随机变量 X，期望值为 μ，那么其期望值就是：

$$E[X] = \int_x xp(x;\mu,\sum)dx = \mu$$

而随机变量 Z 的向量值的协方差是 $Cov(Z) = E[(Z - E[Z])(Z - E[Z])^T]$，这就生成了实数随机变量的变量符号。协方差也能用 $Cov(Z) = E[(Z - E[Z])(Z - E[Z])^T]$ 来推导（你要会证明这些与那两个定义是不相同的）。如果有 $X \sim \mathcal{N}(\mu, \sum)$, 则有

$$Cov(X) = \sum$$

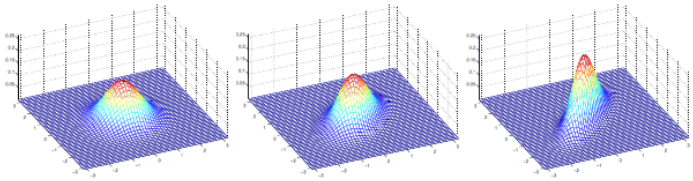
让我们通过这里的例子来看一下高斯分布的密度：



最左边的是期望为 0（是 2×2 零向量）和协方差矩阵 $\sum = I$（2×2 特征矩阵 identity matrix)。中间的图展示了一个期望为 0 并且 $\sum = 0.6I$ 的高斯密度函数；最右边的是 $\sum = 2I$ 的，我们可以看到随着 $\sum$ 的增加，高

斯函数的密度图像就变得扩散（spread-out），也就是变小，那么自然分布就会变得更扁了。

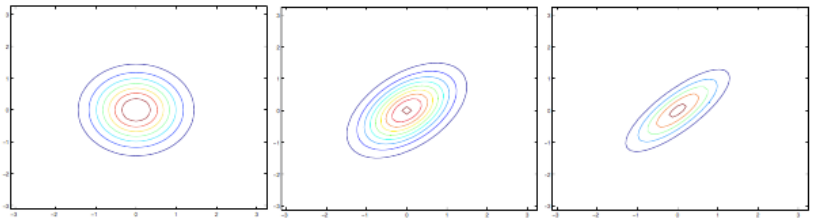
让我们看看更多的例子。



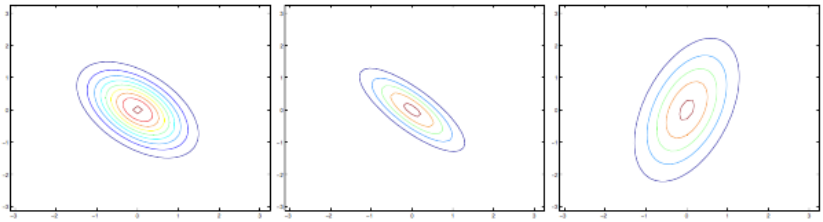
上面的图像的高斯分布的均值期望为 0，其协方差矩阵分别为：

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

最左边的展现了熟知的标准正态分布，当在反对角线上增加数值时，图像在 45 度方向上更扁了（$x_1 = x_2$），我们可以很清楚的在俯视图角度观看这样的变化：



那么如果是减少的话，可以从下图观察：

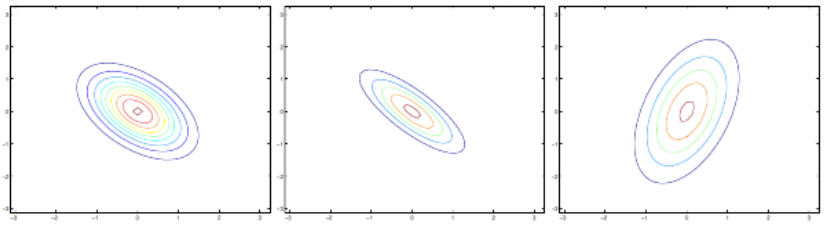


而它们的值分别是

$$\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 3 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

第一、二幅图很容易看出负值的话就是反方向发生扁化，最后一幅，由于将数据更随便了，于是得到的是一个椭圆型的俯视图。

当我们做最后一次修改，使得 $\sum = I$，从而改变 μ 的值，



上面图中的 μ 分别为：

$$\mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mu = \begin{bmatrix} -0.5 \\ 0 \end{bmatrix} \quad \mu = \begin{bmatrix} -1 \\ -0.5 \end{bmatrix};$$

1.1.2 高斯判别分析模型 The Gaussian Discriminant Analysis model

当我们遇到一个分类问题，并且输入的特征值 x 是连续值的随机变量，那么我们就能够用高斯判别分析模型，它会用一个多元正态分布来建立 $p(x|y)$ 模型。该模型为：

$$\begin{aligned} y &\sim Bernoulli(\phi) \\ x|y \sim 0 &\sim \mathcal{N}(\mu_0, \sum) \\ x|y \sim 1 &\sim \mathcal{N}(\mu_1, \sum) \end{aligned}$$

完整写出这个分布的话，有：

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

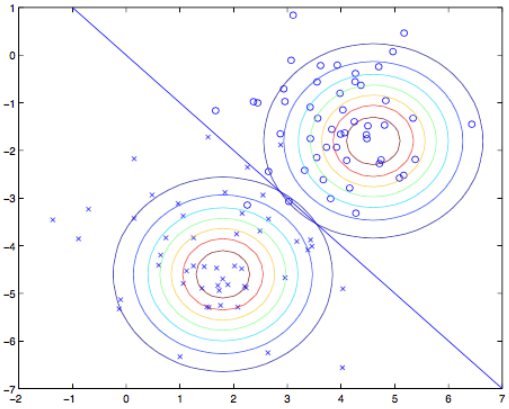
我们模型的参数就是 $\phi, \sim, \mu_0, \mu_1$（注意，这里有两个均值向量，而只使用一个协方差矩阵 $\sum$）。那么数据的对数概率为：

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

通过最大化 ℓ 求参数，我们找到了参数的最大似然估计（作业 1）：

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

这个算法的效果用图表示就是这样子的：



图中的就是训练数据，以及对应的两类的高斯分布的轮廓。注意，这两个高斯分布的轮廓和方向是一样的，因为它们共享一个协方差矩阵 $\sum$ ，但是它们有不同的均值 μ_0 和 μ_1。图中的直线即是一个在 $p(y=1|x)=0.5$ 处的决策边界，在该边界的一侧预测 y=1，另一侧预测 y=0。

1.1.3 讨论：GDA 和逻辑回归 GDA and logistic regression

GDA 模型和逻辑回归有着有趣的关系。如果我们将公式 $p(y = 1|x; \phi, \mu_0, \mu_1, \sum)$ 看作是一个 x 的函数，我们发现它可以表示成形式：

$$p(y = 1|x; \phi, \mu_0, \mu_1, \sum) = \frac{1}{1 + exp(-\theta^T x)}$$

这里的 θ 是合适的关于参数 $\phi, \mu_0, \mu_1, \sum$ 的函数。而这正是用来建立 $p(y = 1|x)$ 模型的判别算法——逻辑回归。

那么我们什么时候挑选一种算法而不用另一种呢？逻辑回归和 GDA 在训练集上得到不同的决策边界，那么哪个较好呢？

我们仅假设如果 $p(x|y)$ 是多元高斯分布（共享 $\sum$），那么 $p(y|x)$ 则会是一个 logistic 函数；反之不成立。$p(y|x)$ 是一个逻辑函数并不能推导出 $p(x|y)$ 是一个多元高斯。这就意味着 GDA 比逻辑回归关于数据做了更强的建模假设。因此，当这些假设这些建模假设正确时，那么 GDA 则可能找到一个更拟合数据的边界，因此它是一个更好的模型。具体来说，如果 $p(x|y)$ 确实是高斯（带有共享的 $\sim$）那么 GDA 则是渐进有效的。非形式化一点，这就意味着在有限的训练集下，没有一个算法比 GDA 更好。因此在这样一个设定下，GDA 将会是一个比逻辑回归更好的算法，并且对小的简单的训练数据也是通用的。我们只需要期待 GDA 会更好。

相反，通过使得有意义的假设更弱，逻辑回归鲁棒性更大，并且对错误的建模假设不是很敏感。有许多种不同的假设集可以得到逻辑函数形式 $p(y|x)$。例如说，如果 $x|y = 0 \sim Poisson(\lambda_0)$，以及 $x|y = 1 \sim Poisson(\lambda_1)$，那么 $p(y|x)$ 将是 logistic。逻辑回归在泊松数据上也有不错的结果。但是如果你在这个数据上用 GDA——也就是在不是高斯分布的数据上拟合高斯分布——那么最后的结果将不是那么可预料，也就是说这里 GDA 也许效果并不怎么好。

总结：GDA 在建模假设正确或者至少大致正确的前提下，它有着更强的建模假设并且数据的使用效率很高（例如只需要较少的训练样本就有好的结果）。Logistic 回归的假设比较弱，更有可能偏离建模假设。具体来说，当数据确实不是高斯分布时，那么由于大数据的局限，逻辑回归往往会比 GDA 的效果更好一些。这也是为什么在实际中逻辑回归用的会比 GDA 更多。

1.2 朴素贝叶斯法 Naive Bayes

在 GDA 中，特征向量 x 是连续的实数值向量。现在我们讨论一种不同的算法，可以应用在当 x 是离散值时。

我们考虑一个有趣的例子——用机器学习建立一个垃圾邮件过滤器。我们要分出邮件是否是商业垃圾邮件，或者是正常的邮件。通过学习算法，我们可以使得我们的邮箱能够自动为接收到的邮件进行分类。邮件分类其实是文本分类领域的一个例子。

我们先有一个训练集，这个集合已经被标签了是否是垃圾邮件。现在我们需要通过具体化用来代表一封邮件的特征参数 x_i 来构造我们的垃圾邮件过滤器。

我们应一个特征向量来表示一封邮件，这个特征向量的长度等于字典中单词的个数。具体地，如果说这封邮件包含在字典中的第 i 个字，那么设置 $x_i = 1$，否则设置 $x_i = 0$。例如这样的一个向量：

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

代表着这个邮件有字“a”和“buy”，但是没有其他的一些字。被编入特征向量中的字叫做词汇（vocabulary），因此 x 向量的维度与词汇的规格是一样的。

选择了我们的特征向量之后，我们就能够建立一个生成模型。因此我们得建立模型 $p(x|y)$。但是如果我们有一个规格为 5000 字的词汇，那么 $x \in \{0,1\}^{50000}$（x 是一个 0，1 的 5000 维的向量），如果假设我们要建模的 x 是一个有着 2^{50000} 可能的输出，那么我们最后就得有一个（$2^{50000} - 1$）维度的特征参数的向量。显然参数的个数过多了。

因此，建立 $p(x|y)$ 我们在这里就要使用一个非常强的假设——x 在 y 下是条件独立的。这个假设叫做朴素贝叶斯假设（Naive Bayes Assumption），而这样的算法就叫做朴素贝叶斯分类器。例如如果 $y = 1$ 代表垃圾邮件；“buy”是字 2087，“price”是字 39831；那么假设定义了一封邮件如果是垃圾邮件，

即 y=1 时，那么“price“是否出现在邮件中不会影响”buy“是否出现在邮件中。形式化地书写就是 $p(x_{2087}|y) = p(x_{2087}|y, x_{39831})$。（注意，这并不代表“buy”和“price”是相互独立的，这里是在条件 y 之下，才有“buy”和“price”条件独立。）那么我们就有：

$$\begin{aligned}
&p(x_1, ..., x_{50000}|y) \\
&\qquad = p(x_1|y)p(x_2|y, x_1)p(x_3|y, x_1, x_2)...p(x_{50000}|y, x_1, .., x_{49999}) \\
&\qquad = p(x_1|y)p(x_2|y)P(x_3|y)...p(x_{50000}|y) \\
&\qquad = \prod_{i=1}^{n} p(x_i|y)
\end{aligned}$$

第一个等式主要是一般的概率公式，第二个等式利用了朴素贝叶斯假设。我们知道虽然朴素贝叶斯是一个比较强的假设，但是它的确是在很多问题上效果不错。

我们模型的参数是 $\phi_{i|y=1} = p(x_i|y = 1), \phi_{i|y=0} = p(x_i|y = 0), \phi_y = p(y = 1)$。像往常一样，给定一个训练集 $\{(x^{(i)}, y^{(i)}); i = 1, ..., m\}$，我们就能够将数据的概率表示为：

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

求其最大值，从而得到最大似然估计：

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

上面的符号 $\wedge$ 代表的是且，$\phi_{j|y=1}$ 意思是垃圾邮件中（y=1），字 j 出现在该邮件的邮件数。

那么，拟合完这些参数之后，我们就可以预测一个新的样本特征 x，我

们计算:

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

然后选择有更高后验概率的那一类。

最后我们要注意刚才我们的朴素贝叶斯算法是特征 x_i 是两种值的那类问题。那么更普遍的是 x_i 可以取 $\{1, 2, ..., k_i\}$ 中的值。这儿我们作为多元分布建模 $p(x_i|y)$，而不是伯努利分布。当然，还有一些输入值是连续的值（例如一开始讲的房价与房屋面积的问题），那么离散化（discretize）它们是很常见的，然后再利用 NB 算法。例如，我们用特征 x_i 来表示居住面积，我们可能需要做如下的离散化操作:

居住面积（英尺）	< 400	400-800	800-1200	1200-1600	> 1600
x_i	1	2	3	4	5

因此，对于一个 890 英方的居住面积，我们设置对应的 $x_i = 3$。然后我们就能根据之前的描述的对于多元分布应用朴素贝叶斯算法来建立模型 $p(x_i|y)$。当初始的连续参数值用多元正态分布建模的效果不好时，可以将参数离散化，并利用朴素贝叶斯法（而不是 GDA）通常会得到一个更好的分类器。

1.2.1 拉普拉斯平滑 Laplace smoothing

NB 算法在许多问题中的效果都非常的好，但是呢这里还有一个简单的修改可以使得它的效果更佳，尤其在文本分类中。下面就简单讨论下一个适用于该类型的问题，并给出如何改进它。

假设有一个邮件分类系统，在完成 cs229 之后你已经很好地完成了这个项目，你打算在 2003 年六月左右向 NIPS 发布你的成果。由于你要开始在邮件中讨论你的论文，因此你肯定会开始写带有“nips”字的 email。但是由于这是你第一次向该会议提交论文，在这之前你的邮箱中并没有字“nips”，换句话说，“nips”之前并没有出现在你的垃圾邮件训练集中。假设“nips”是你的词汇中的第 35000 个字，因此你的朴素贝叶斯邮件过滤器要在参数 $\phi_{35000|y}$ 中挑选它的最大似然估计:

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

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

由于之前的训练样本中从未出现过字“nips”，它认为在任何一类邮件中看到它的概率都是 0。因此，我们想要知道这些包含“nips”的消息的邮件是否是垃圾邮件，那么计算该类的后验概率，显然有：

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

之所以上面的值为 0 是因为每一个项 $\prod_{i=1}^{n} p(x_i|y)$ 都包括了乘数项 $p(x_{35000|y}) = 0$。这样一来我们的算法得到的结果是 0/0 而无法进行预测。

将问题说得更广泛点，估计一些事件的概率为 0 数据上是个不好的注意，因为之前在你的有限的训练集中并没见到过。解决这样的估计一个多元随机变量 z 的问题，且 z 是在 1 到 k 中取值，我们可以用 $\phi_i = p(z = i)$ 作为参数。给定 m 个独立的观测值 $\{z^{(1)}, ..., z^{(m)}\}$，最大似然估计为：

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

就像之前看到的，如果利用这些最大似然估计，有可能 ϕ_j 会有 0 的结果，这就会产生问题。为了避免这一问题，我们采用拉普拉斯平滑（Laplace smoothing），换用下面的估计：

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

注意这样修改的话，$\sum_{j=1}^{k} \phi_j = 1$ 依旧成立（check this yourself!）。同时 $\phi_j \neq 0$ 对于所有的 j 值都成立，这就解决了概率可能为 0 的问题。在这样的条件下（该条件可证明是很强的），拉普拉斯平滑实际上提供了一个对 ϕ_j 最好的最好的估计。

再结合我们的 NB 分类器和拉普拉斯平滑，我们就能得到下面的参数估计：

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} + 2}$$

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} + 2}$$

(实际上，是否在 ϕ_y 上使用拉普拉斯不是那么重要，因为一般都会有一个比较好的值，因此 ϕ_y 会有一个合理的 $p(y = 1)$ 的估计并且离 0 会很远。)

1.2.2 文本分类器的事件模型 Event models for text classification

临近生成学习算法话题结束，现在多讨论一个关于文本分类的模型。在朴素贝叶斯算法在文本分类上得到很好的应用的同时，其实还有一个更好的模型。

在特定的文本分类的上下文中，其实朴素贝叶斯算法用的是多变量伯努利事件模型（multi-variate Bernoulli event model）。在该模型中，我们假设一封邮件产生的方式是，首先它随机决定一封邮件是否会给你发送下一条信息。然后，发送邮件的发送者开始遍历字典，根据概率 $p(x_i - 1|y) = \phi_{i|y}$ 独立决定每一个字 i 是否出现。因此信息的可能性就是由 $p(y)\prod_{i=1}^{n} p(x_i|y)$。

这里又一个不同的模型叫做多重事件模型（multinomial event model）。为了描述这个算法，我们要用不同的记号和特征集来表示邮件。我们令 x_i 代表在邮件中第 i 个字的特点。因此，x_i 是一个整数，从 $\{1, ..., |V|\}$ 中取值，其中 $|V|$ 是我们词汇的规格。现在带有 n 个字的 email 现在由一个向量 $(x_1, x_2, ..., x_n)$ 表示。注意到这里的 n 不太会想通。例如说，如果一个邮件开始是"A NIPS ..."，那么 $x_1 = 1$（a 是在字典中的第一个字），而 $x^2 = 35000$（假如"nips"是字典中的第 35000 个字）。

在多元事件模型，我们假设邮件生成的方式是一个根据 $p(y)$ 首次决定垃圾／正常邮件的随机过程。然后邮件的发送方通过第一次在字 $p(x_1|y)$ 上的一些多元分布生成 x_1，接着，第二个字被选择是独立于第一个字的，但是来自同样的多元分布，x_3，x_4 之后则类似，直到邮件中的 n 个字被生成。因此一个信息的总的概率就通过 $p(y)|\prod_{i=1} np(x_i|y)$。注意到这个公式和多变量伯努利事件模型中的信息概率看起来很相似，但它们是不同的。尤其是，$x_i|y$ 是一个多元分布，而不是伯努利分布。

新模型的参数是 $\phi_y = p(y)$，$\phi_{k|y=1} = p(x_j = k|y = 1)$（对所有的 j）和 $\phi_{i|y=0} = p(x_j = k|y = 0)$。我们假设 $p(x_j|y)$ 对所有的 j 值都一样。(产生的字的分布并不依赖于邮件中它的位置 j)。

如果我们给定一个训练集 $\{(x^{(i)}, y^{(i)}); i = 1, ..., m\}$，并且这里的 $x^{(i)} =$

$(x_1^{(i)}, x_2^{(i)}, ..., x_{n_i}^{(i)})$（这里的 n_i 是在第 i 个训练样本中的字的数量），数据的概率为：

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

将其最大化得到参数的最大似然估计：

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

如果我们使用拉普拉斯平滑（为了更好的性能），来估计 $\phi_{k|y=0}, \phi_{k|y=1}$，那么我们就需要在分子上加上 1，在分母上加上 |V|，从而得到：

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

如果不是要追求最好的分类算法的话，朴素贝叶斯分类通常效果是不错的。由于它的实现简单，它也是一个非常好的可以用来做第一次尝试的算法。