C1-4

Machine Learning
by Andrew Ng, Stanford Engineering

Xiaojie Zhou

szxjzhou@163.com

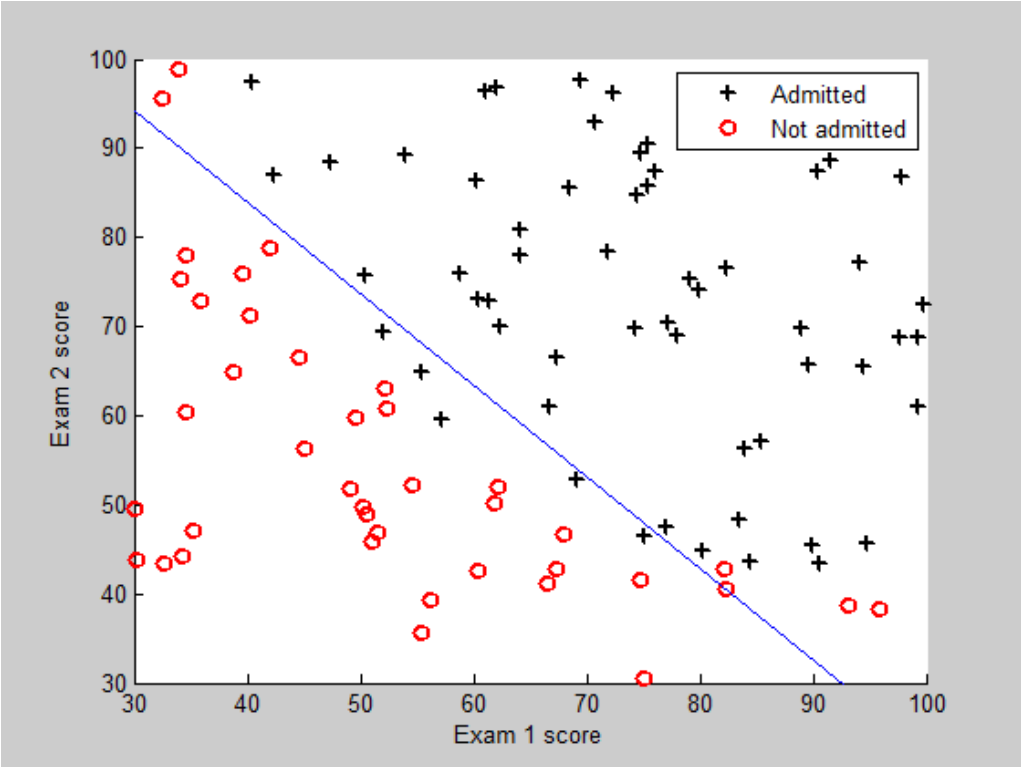
2016.8.10

第五集：生成学习算法

- 生成学习算法(Generative Learning Algorithms)
- 高斯判别分析(Gaussian Discriminant Analysis(GDA))
- 比较：生成学习算法与判别学习算法(Generative Learning Algorithms versus Discriminative Learning Algorithm)
- 朴素贝叶斯(Naive Bayes)
- 拉普拉斯平滑(Laplace Smoothing)

第五集：生成学习算法

- 生成学习算法
  - 对于分类来说我们已经学过了Logistic回归的办法，在Logistic回归中我们需要找到一条直线将数据分开，直线划分的区域即为两个类的数据区域



第五集：生成学习算法

- 生成学习算法
  - 比如说一个很现实的问题，需要根据肿瘤的一些特征（比如说大小、病人年龄等等）判断肿瘤是否恶性，除了Logistic回归直接划分的方法外还有一种间接的做法
    - 我们可以对于恶性肿瘤、良性肿瘤两种样本进行建模。当新的数据过来时，看是对于恶性肿瘤的拟合效果更佳还是对于良性肿瘤的拟合效果更佳
    - 这其实代表着两类学习算法
      - 判别学习算法：目的是学习p(y|x)（对于给定的特征x对应分类为y的概率）或者直接划分h(x)，比如Logistic回归
      - 生成学习算法：目的是学习p(x|y)（对于给定的分类为y，暗示着某种特定特征x的概率，比如对应到上面的例子可能p(x|y = 0)就表示良性肿瘤的特征分布， p(x|y = 1)就表示恶性肿瘤的特征分布

第五集：生成学习算法

- 生成学习算法
  - 由于我们通过极大似然的办法对于问题进行优化，而在极大似然中需要求出p(y|x)
    - 因此我们需要在已知p(x|y)的基础上求出p(y|x)，这时就需要用到贝叶斯公式

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

其中p(x)的计算方法为$\Sigma_{i=1\sim n}p(x|y = yi)p(y = yi)$，但在一般情况下我们并不需要花时间计算p(x)，原因在于对于机器学习的问题来说特征x的集合是给定的，因此p(x)是一个常数（从这个特征集合中抽取某一种特征x是个常数）。因此我们在做极大似然优化的时候跟p(x)是没有关系的，因此将其忽略

第五集：生成学习算法

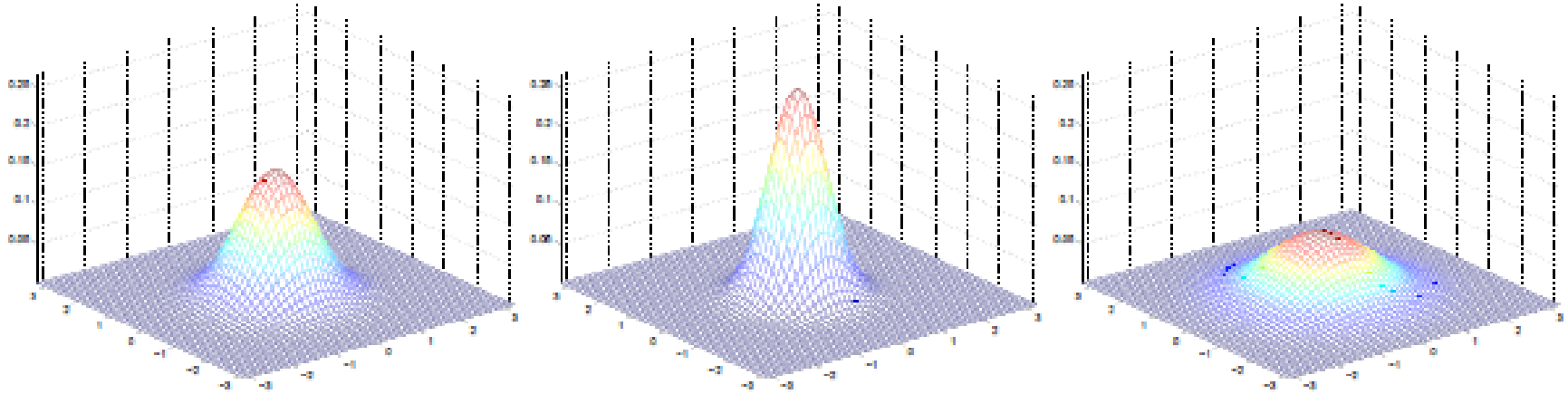
- 高斯判别分析
  - 假设特征x是个连续值，假设p(x|y)满足高斯分布
    - 由于可能不止一个特征，因此需要通过多元高斯进行建模。其中μ为均值、Σ为协方差矩阵（Σii = Var(Xi), Σij = Cov(Xi,Xj) = E[(X − E(X)) (Y − E(Y))]），|Σ|为协方差矩阵Σ的行列式

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

      - 这是一般形式，但是在处理问题上为了减少对于协方差矩阵的行列式和逆的求解的运算量，一般假定协方差矩阵中Σij = 0.经过这样处理后行列式为所有方差的乘积，逆为将主对角线上的元素取倒数后得到的矩阵

第五集：生成学习算法

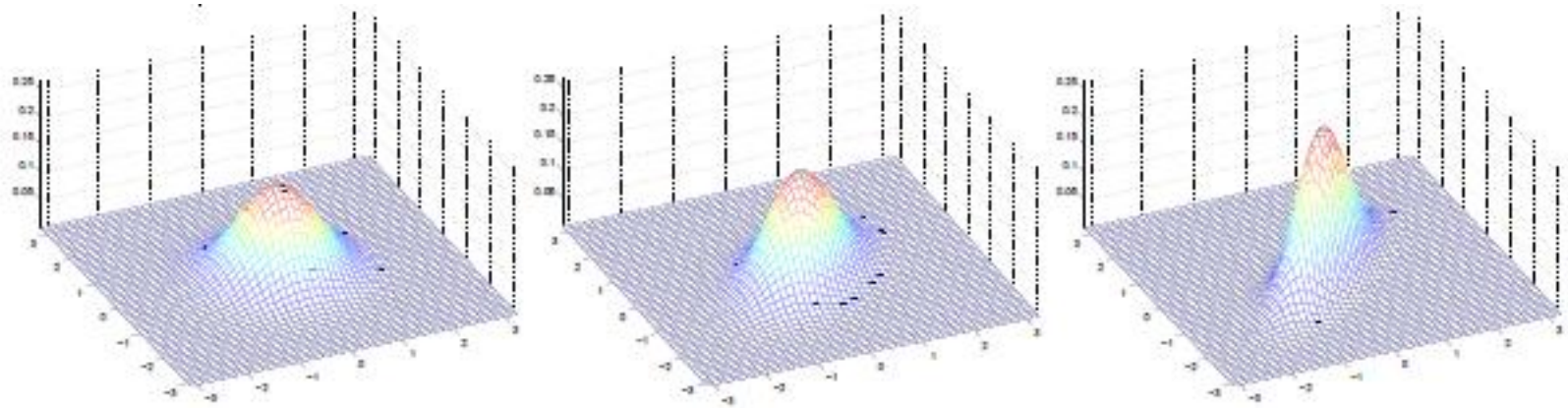
- 高斯判别分析
  - 下面我们来看多元高斯的参数的性质
    - 先来看协方差矩阵Σ中的方差部分是如何影响分布的，首先我们假定均值为0，为二元参数的高斯分布



在这三幅图中，第一幅图对应的方差Σ为2维单位矩阵I（这种均值为0，方差为I的高斯分布也被称为标准正态分布(standard normal distribution)）；第二三幅图的方差分别为0.6I、2I。通过对比这些图发现，随着方差值的增大钟形覆盖范围增大，高度缩小，像是将高峰往下压的结果

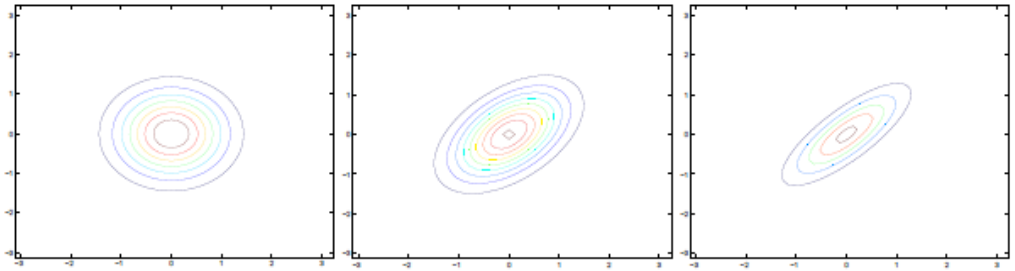
第五集：生成学习算法

- 高斯判别分析
  - 继续来看多元高斯的参数的性质
    - 然后来看协方差矩阵Σ中的协方差部分是如何影响分布的，首先我们假定均值为0，为二元参数的高斯分布



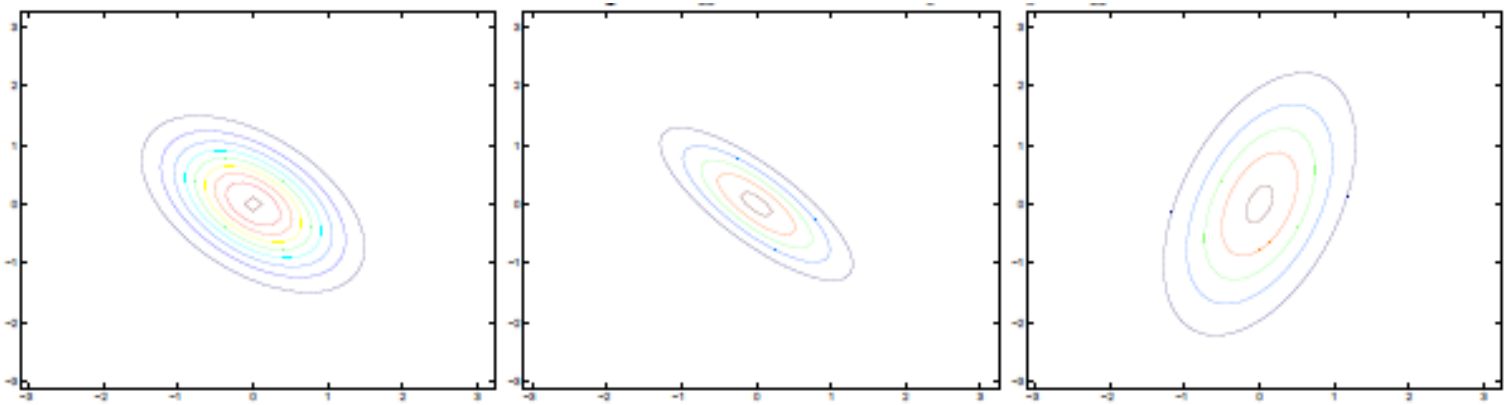
通过对比这些图发现，随着协方差值的增大钟形被从45度挤压的强度越大

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



第五集：生成学习算法

- 高斯判别分析
  - 继续来看多元高斯的参数的性质
    - 然后来看协方差矩阵Σ整体是如何影响分布的，首先我们假定均值为0，为二元参数的高斯分布

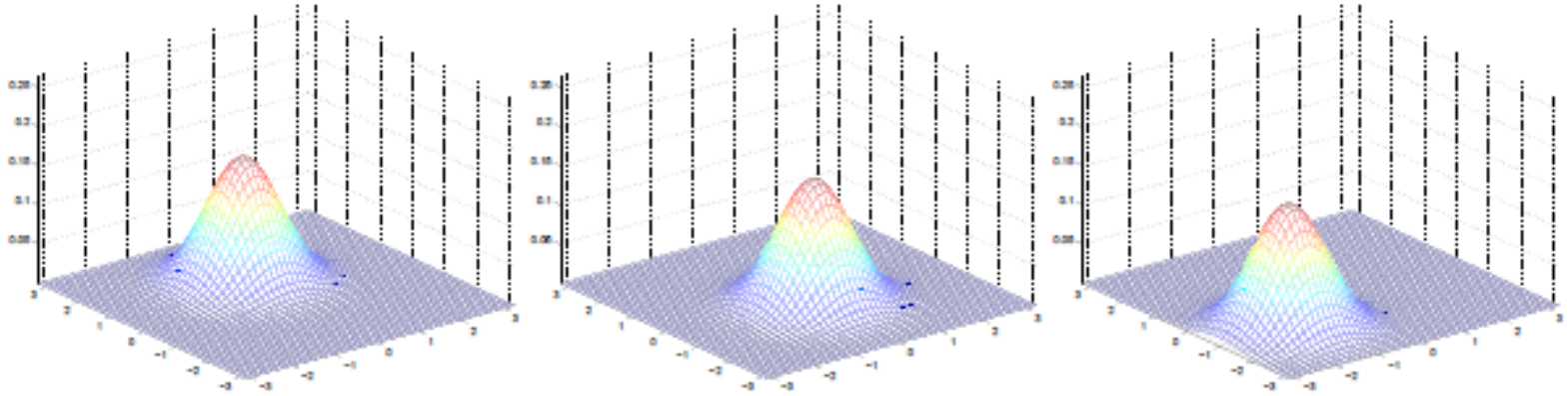


$$\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 3 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

通过对比1图、2图发现，随着协方差值的反向增大钟形被从45度挤压的强度越大（只不过从另一个45度方向进行挤压）；
通过对比2图、3图发现单独改变某些方差值会使图像往该方向处伸展更加剧烈，从而更加椭圆

第五集：生成学习算法

- 高斯判别分析
  - 继续来看多元高斯的参数的性质
    - 最后看一下均值是怎样影响分布的，首先我们假定协方差矩阵为Σ，为二元参数的高斯分布

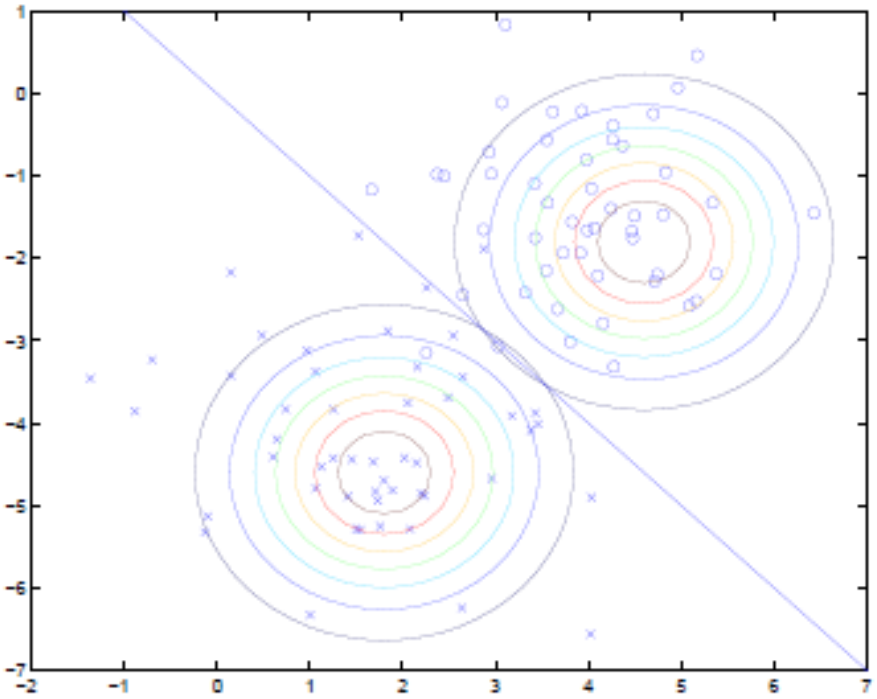


$$\mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad \mu = \begin{bmatrix} -0.5 \\ 0 \end{bmatrix}; \quad \mu = \begin{bmatrix} -1 \\ -1.5 \end{bmatrix}$$

均值直接影响的是钟形中心的所在位置

第五集：生成学习算法

- 高斯判别分析
  - 高斯判别分析的总体过程如下图所示
    - 可以发现在高斯判别分析中我们分别对训练样本中的每一个类进行建模，得到每个类对应的p(x|y=yi)。通过这些模型即可知道几个类别的分割器。



图中将高斯判别分析在数据集上的效果进行了展示。可以看到在高斯判别分析对于两类的训练样本分别训练对应的高斯分布，最终得到那条划分的直线

第五集：生成学习算法

- 高斯判别分析
  - 现在考虑如何通过高斯判别分析解决Logistic回归中要解决的{0,1}分类问题
    - 对于这一类问题来说易得如下的约束
      - 由于是{0,1}分类问题，因此y服从二项分布
      - 由于采用高斯判别分析的方法，因此p(x|y=yi)均通过多元高斯进行建模

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

此时为了简便考虑，假定不同的高斯分布之间仅均值不同方差相同

第五集：生成学习算法

- 高斯判别分析
  - 接下来计算极大似然

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

从1式到2式间使用了贝叶斯公式（但由于贝叶斯公式的分母为常数，因此在此忽略分母而只考虑分子，由此相当于只计算x,y的联合概率p(x,y)），这种用联合概率的极大似然表达方法称为联合极大似然(Joint Likelihood)，区别于用条件概率p(y|x)进行的极大似然表示（后者称为条件极大似然(Conditional Likelihood)）

- 通过极大似然可以得到每个参数的表达方式（这里面由于有4个参数，因此需要对这四个参数求偏微分，令其偏微分为0即可得到对应的关系式）
  - 这里面应用了两个性质：第一，常用的概率密度函数都是对数凹函数（即其对数为凹函数），其本身为拟凹函数（见《凸优化》3.5，例3.40，高斯函数log后相当于二次函数，二项分布log后相当于线性函数）；第二，对数凹性对于乘法是封闭的（即对数凹函数的乘积也是对数凹函数，见《凸优化》3.5.2，log可使得乘法变加法，而非负加权求和是保凸的运算）

第五集：生成学习算法

- 高斯判别分析
  - 对于ϕ的计算

由于

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

可以看到这里面与φ有关的只有右边一项，因此求偏微分时左边一项变为系数。因此令φ的偏微分为 0，相当于让右边一项的偏微分为 0，而不用考虑左边一项，因此有：

$$\begin{aligned}\nabla_{\phi_y} \ell(\varphi) &= \nabla_{\phi_y} \sum_{i=1}^{m} \left(y^{(i)} \log \phi_y + (1 - y^{(i)}) \log(1 - \phi_y) \right) \\ &= \sum_{i=1}^{m} \left(y^{(i)} \frac{1}{\phi_y} - (1 - y^{(i)}) \frac{1}{1 - \phi_y} \right)\end{aligned}$$

Then setting $\nabla_{\phi_y} = 0$ gives us

$$\begin{aligned}0 &= \sum_{i=1}^{m} \left(y^{(i)} \frac{1}{\phi_y} - (1 - y^{(i)}) \frac{1}{1 - \phi_y} \right) \\ &= \sum_{i=1}^{m} \left(y^{(i)} (1 - \phi_y) - (1 - y^{(i)}) \phi_y \right) \\ &= \sum_{i=1}^{m} y^{(i)} - \sum_{i=1}^{m} \phi_y.\end{aligned}$$

Therefore,

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}.$$

第五集：生成学习算法

- 高斯判别分析
  - 对于μ的计算

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

可以看到这里面与μ有关的只有左边一项，因此求偏微分时右边一项变为系数。因此令μ的偏微分为 0，相当于让左边一项的偏微分为 0，而不用考虑右边一项，因此有：

$$\nabla_\mu\ell(\mu)=\nabla_\mu\left[-\frac{1}{2}\sum_{i=1}^{m}(x-\mu)^T\Sigma^{-1}(x-\mu)\log\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\right]$$

$$=-\frac{1}{2}\Sigma^{-1}\log\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\sum_{i=1}^{m}2(x-\mu)$$

令其为 0 有：

$$0=-\frac{1}{2}\Sigma^{-1}\log\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\sum_{i=1}^{m}2(x-\mu)$$

$$\leftrightarrow \mu=x$$

由于不同的样本从属于不同的高斯分布，因此有：

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

第五集：生成学习算法

- 高斯判别分析
  - 对于方差Σ的计算则直接使用了方差计算的公式，由此最终得到了如下的参数表达

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

Φ的含义是所有类标为1的样本个数占样本总数的比值；均值μ0的含义是所有类标为0的样本x的和除以所有类标为0的样本总数；均值μ1的含义是所有类标为1的样本x的和除以所有类标为1的样本总数；方差为所有样本减去对应均值的平方除以样本总数（因为此处两个高斯共享一个方差）

第五集：生成学习算法

- 高斯判别分析
  - 求出参数以后我们已经可以成功表达出p(y),p(x|y=0)和p(x|y=1)，接下来考虑当接收某个测试样本x′，如何预测对应的类标y′
    - 这时最直观的办法就是对于p(y=yi|x)进行计算，这时需要用到贝叶斯公式

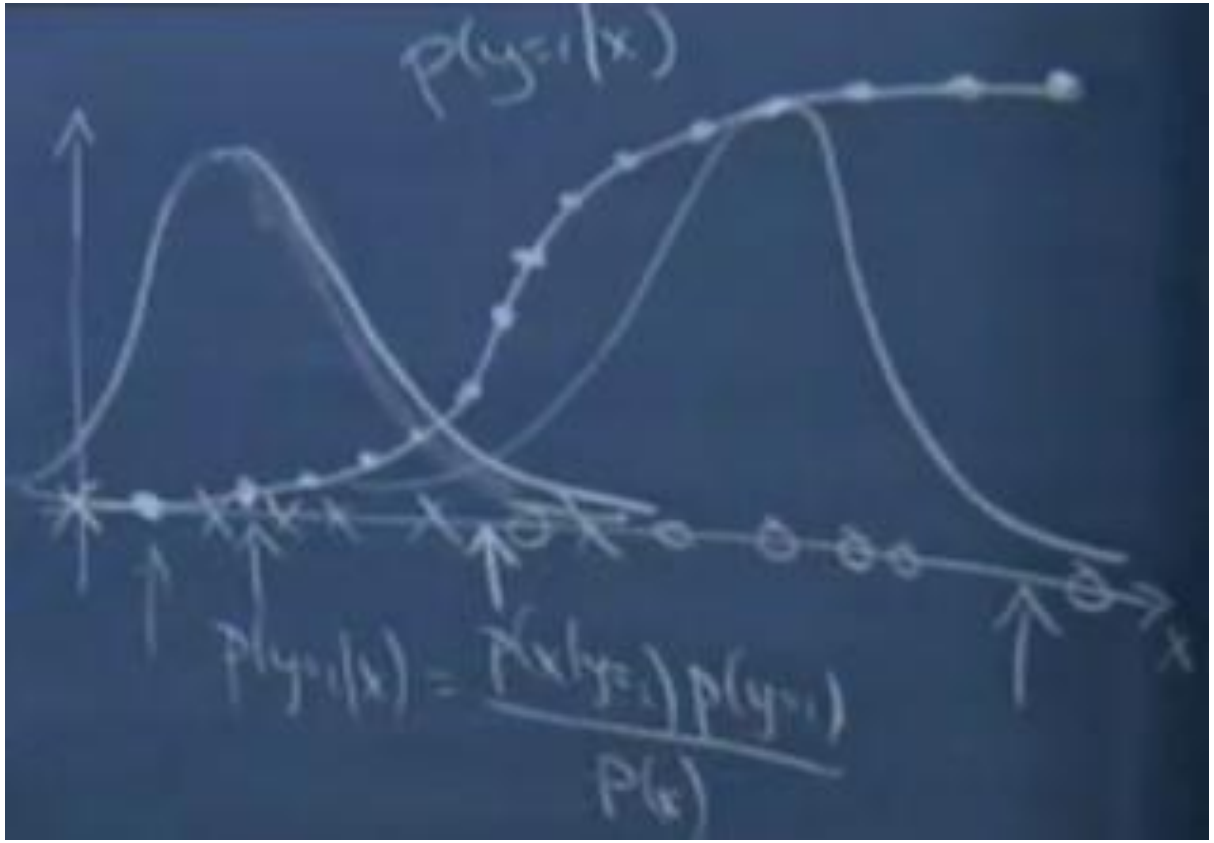
$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

- 由于我们需要找到的是令p(y|x)取得最大值的对应的ymax，在这里我们发现在计算p(y=yi|x)的过程中p(x)的值是不变的，因此我们同样可以忽略掉分母，只考虑分子

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

第五集：生成学习算法

- 比较：生成学习算法与判别学习算法



假设只考虑一维的特征和{0, 1}分类的情况。通过生成学习算法我们发现对于每个类标分别学习出了如图所示的高斯函数图像。当我们判断每个特征所属的分类时需要计算对应的p(y=yi|x)。通过计算后发现p(y=1|x)对应的函数图像形如Sigmoid函数。

由于这里需要计算出p(y=1|x)的确定值，因此在代入贝叶斯公式后不能直接将分母p(x)忽略掉，而是需要通过$\Sigma_{i=1\sim n}p(x|y = yi)p(y = yi)$（在这里是p(x|y=0)p(y=0) + p(x|y=1)p(y=1)进行计算）

第五集：生成学习算法

- 比较：生成学习算法与判别学习算法
  - 接下来继续深入地看生成学习算法与判别学习算法的关系
    - 由于在判别学习算法中直接的输出就是分类结果，比如在Logistic回归中通过极大似然可以得到参数θ的值，进而得到了Sigmoid函数函数的表达式h(x)，代入测试集中的x即可得到对应的分类结果
    - 但是在生成学习算法中我们需要衡量p(y|x)，找出能使得p(y|x)取最大的那个y
      - 因此类比Logistic回归这个判别学习算法，看一下在生成学习算法下p(y=1|x)>p(y=0|x)对应的结论是什么

在开始下面的证明前，先做如下的假设。不失一般性地，我们可以将连续值看成是无穷多个离散值的集合，即用{k1,⋯,k∞}来表示整个无限集合，因此有：

$$p(y=1|x) \geq p(y=0|x) \Leftrightarrow \frac{(\prod_{j=1}^{n} p(xj|y=1))p(y=1)}{(\prod_{j=1}^{n} p(xj|y=0))p(y=0)} \geq 1$$

$$\Leftrightarrow \frac{\left(\prod_{j=1}^{n} p(xj=k1|y=1)^{1\{xj=k1\}} \dots p(xj=k\infty|y=1)^{1\{xj=k\infty\}}\right)\varphi}{\left(\prod_{j=1}^{n} p(xj=k1|y=0)^{1\{xj=k1\}} \dots p(xj=k\infty|y=0)^{1\{xj=k\infty\}}\right)(1-\varphi)} \geq 1$$

$$\Leftrightarrow \sum_{j=1}^{n}\left[1\{xj=k1\}\log\left(\frac{p(xj=k1|y=1)}{p(xj=k1|y=0)}\right)+\cdots\right.$$

$$\left.+1\{xj=k\infty\}\log\left(\frac{p(xj=k\infty|y=1)}{p(xj=k\infty|y=0)}\right)\right]+\log\left(\frac{\varphi}{1-\varphi}\right)\geq 0$$

$$\Leftrightarrow \begin{bmatrix}1\{xj=k1\}\\ \cdots \\ 1\{xj=k\infty\}\end{bmatrix}\left[\log\left(\frac{p(xj=k1|y=1)}{p(xj=k1|y=0)}\right) \quad \dots \quad \log\left(\frac{p(xj=k\infty|y=1)}{p(xj=k\infty|y=0)}\right)\right]$$

$$+\log\left(\frac{\varphi}{1-\varphi}\right)\geq 0 \Leftrightarrow \theta^T\begin{bmatrix}1\\ x\end{bmatrix}\geq 0$$

第五集：生成学习算法

- 比较：生成学习算法与判别学习算法
  - 经过证明表明，假如p(x|y)服从正态分布p(y=1|x)等价于一个Logistic函数，但是这个在反方向上不成立
    - 即p(y|x)是一个Logistic函数时并不能推出p(x|y)服从正态分布
    - 原因在于发现当p(x|y)服从泊松分布（此时p(x|y)=0~Poisson(λ0), p(x|y)=1~Poisson(λ1)）时p(y|x)仍然可对应到Logistic函数（这个其实从之前的证明中就可以看出，在之前的证明中我们在并没有用到p(x|y)服从某一分布的假设的情况下就推出了对应的Logistic函数）
  - 由此表明，假设p(x|y)服从正态分布是比假设p(y|x)服从Logistic分布更为具体的假设，后者更为一般性（后者蕴含前者）

第五集：生成学习算法

- 比较：生成学习算法与判别学习算法
  - 根据之前所述可以得出如下的结论：
    - 如果我们知道数据的大致分布情况（比如说我们知道数据大致满足正态分布、满足泊松分布等），采用生成学习算法会得到更好的效果，因为在生成学习算法中利用了分布的特征
    - 但是对于多种分布或者不确定分布的情况，采用判别学习算法会得到更好的效果，因为判别学习算法更加适应于一般情况的求解，健壮性更好
      - 这里特别澄清一点，在判别学习算法中一般通过指数分布族中的分布推导广义线性模型。在那时我们也考虑了不同的分布，但那时的着眼点在y的特征上（比如说我们要进行{0,1}分类就采用二项分布，多元分类就采取多项式分布...），并没有像生成学习算法那样考虑特征x的分布情况
    - 生成学习算法还有一个好处在于需要的数据量不大就可以拟合出一个比较准确的模型，而判别学习方法所需的样本数要多得多

第五集：生成学习算法

- 朴素贝叶斯
  - 之前讲过的高斯判别分析是一种典型的生成学习算法，可以解决输入数据x是连续随机变量的分类问题
  - 那么对于离散随机变量，朴素贝叶斯是一个很好的求解方法
    - 对于一种文本分类(text classification)问题，比如说我们要对于垃圾邮件进行分类，邮件内容为文本
      - 此时进行的仍然是{0,1}分类，0表示非垃圾邮件，1表示垃圾邮件
    - 判断垃圾邮件的依据是邮件里面是否有令人反感的内容的出现，我们可以用如下的办法对于邮件内容进行表示（表示方法不唯一）

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

可以设立一个字典，向量中的值表示字典中某一词汇是否出现过。如出现过则为1，否则为0（字典的大小根据实际问题和计算机情况进行设定）

第五集：生成学习算法

- 朴素贝叶斯
  - 对于上面那个问题可以通过判别学习方法进行求解
    - 使用判别学习方法：因为这里是{0,1}二元分类问题，因此采取的仍然是Logistic回归。也就是要通过下面的式子进行参数θ的学习，从而得到那条划分的直线

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

      - 但我们发现对于这个问题来说由于x的维数非常大（比如说我们的字典有5000个词，x的维数就为5000，但事实上字典的次数远远不止这个规模），因而导致参数θ的维数也非常大，这样的学习过程非常长
      - 因此只好考虑生成学习算法以避免直接对参数θ进行学习。这里面就需要解决一个问题，输入x的样本特征是什么（因为生成学习算法比起判别学习方法多利用了样本的特征）？如何计算p(x|y)？

第五集：生成学习算法

- 朴素贝叶斯
  - 对于上面的情况可以假设某一个词在邮件中的出现不影响其他词在邮件中的出现，即在给定y的情况下，xi条件独立
    - 实际上这样假设是有问题的，因为在实际情况下由于语境有些词确实会和其他词伴随出现，但实际表明这样假设下的算法对于文本分类问题的效果确实不错
    - 由此我们可以对于p(x|y)进行计算
      - 首先由于概率的链式法则有：

$$\begin{aligned} & p(x_1, \ldots, x_{50000}|y) \\ = \; & p(x_1|y)p(x_2|y,x_1)p(x_3|y,x_1,x_2)\cdots p(x_{50000}|y,x_1,\ldots,x_{49999}) \end{aligned}$$

      - 由于条件独立，进而有：

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

这一假设也被称为朴素贝叶斯假设(Naive Bayes (NB) assumption)，对应得到的算法称为朴素贝叶斯分类器(Naive Bayes classier)

第五集：生成学习算法

- 朴素贝叶斯
  - 下面将对p(x|y)进行求解，表达出极大似然
    - 在这里由于对于每一维的x只能取{0,1}二元值，而y也满足二项分布，因此可做如下的规定：

$$
\begin{aligned}
p(y) &= (\phi_y)^y(1-\phi_y)^{1-y} \\
p(x|y=0) &= \prod_{j=1}^{n} p(x_j|y=0) \\
&= \prod_{j=1}^{n} (\phi_{j|y=0})^{x_j}(1-\phi_{j|y=0})^{1-x_j} \\
p(x|y=1) &= \prod_{j=1}^{n} p(x_j|y=1) \\
&= \prod_{j=1}^{n} (\phi_{j|y=1})^{x_j}(1-\phi_{j|y=1})^{1-x_j}
\end{aligned}
$$

其中：

$$\phi_{j|y=0} = p(x_j = 1|y=0),\ \phi_{j|y=1} = p(x_j = 1|y=1)$$

并由此可得对数极大似然表达形式：

$$
\begin{aligned}
\ell(\varphi) &= \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}; \varphi) \\
&= \log \prod_{i=1}^{m} p(x^{(i)}|y^{(i)}; \varphi) p(y^{(i)}; \varphi) \\
&= \log \prod_{i=1}^{m} \left(\prod_{j=1}^{n} p(x_j^{(i)}|y^{(i)}; \varphi) \right) p(y^{(i)}; \varphi) \\
&= \sum_{i=1}^{m} \left(\log p(y^{(i)}; \varphi) + \sum_{j=1}^{n} \log p(x_j^{(i)}|y^{(i)}; \varphi) \right) \\
&= \sum_{i=1}^{m} \Bigg[y^{(i)} \log \phi_y + (1-y^{(i)}) \log(1-\phi_y) \\
&\qquad + \sum_{j=1}^{n} \left(x_j^{(i)} \log \phi_{j|y^{(i)}} + (1-x_j^{(i)}) \log(1-\phi_{j|y^{(i)}}) \right) \Bigg]
\end{aligned}
$$

第五集：生成学习算法

- 朴素贝叶斯
  - 通过极大化对数极大似然求解出其中的参数
    - 先对于φj|y=0（即p(xj=1|y=0)）进行求解

$$\sum_{i=1}^{m}\left[y^{(i)}\log\phi_y+(1-y^{(i)})\log(1-\phi_y)\right.$$
$$\left.+\sum_{j=1}^{n}\left(x_j^{(i)}\log\phi_{j|y^{(i)}}+(1-x_j^{(i)})\log(1-\phi_{j|y^{(i)}})\right)\right]$$

由于在对数极大似然中左边半式与 φj|y=0 无关，因此对 φj|y=0 求偏微分后结果为 0，可直接忽略，而只对右边半式对 φj|y=0 求偏微分，因此有：

$$\begin{aligned}\nabla_{\phi_{j|y=0}}\ell(\varphi) &= \nabla_{\phi_{j|y=0}}\sum_{i=1}^{m}\left(x_j^{(i)}\log\phi_{j|y^{(i)}}+(1-x_j^{(i)})\log(1-\phi_{j|y^{(i)}})\right)\\ &= \nabla_{\phi_{j|y=0}}\sum_{i=1}^{m}\left(x_j^{(i)}\log(\phi_{j|y=0})1\{y^{(i)}=0\}\right.\\ &\qquad\left.+(1-x_j^{(i)})\log(1-\phi_{j|y=0})1\{y^{(i)}=0\}\right)\\ &= \sum_{i=1}^{m}\left(x_j^{(i)}\frac{1}{\phi_{j|y=0}}1\{y^{(i)}=0\}-(1-x_j^{(i)})\frac{1}{1-\phi_{j|y=0}}1\{y^{(i)}=0\}\right)\end{aligned}$$

令其为 0 有：

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

从而可得到 φj|y=0 为：

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

第五集：生成学习算法

- 朴素贝叶斯
  - 通过极大化对数极大似然求解出其中的参数
    - 然后对于ϕy（即p(y=1)）进行求解

$$\sum_{i=1}^{m}\left[y^{(i)}\log\phi_y+(1-y^{(i)})\log(1-\phi_y)\right.$$
$$\left.+\sum_{j=1}^{n}\left(x_j^{(i)}\log\phi_{j|y^{(i)}}+(1-x_j^{(i)})\log(1-\phi_{j|y^{(i)}})\right)\right]$$

由于在对数极大似然中右边半式与φy 无关，因此对φy 求偏微分后结果为 0，可直接忽略，而只对左边半式对φy 求偏微分，因此有：

$$\begin{aligned}\nabla_{\phi_y}\ell(\varphi) &= \nabla_{\phi_y}\sum_{i=1}^{m}\left(y^{(i)}\log\phi_y+(1-y^{(i)})\log(1-\phi_y)\right)\\ &= \sum_{i=1}^{m}\left(y^{(i)}\frac{1}{\phi_y}-(1-y^{(i)})\frac{1}{1-\phi_y}\right)\end{aligned}$$

令其为 0 有：

$$\begin{aligned}0 &= \sum_{i=1}^{m}\left(y^{(i)}\frac{1}{\phi_y}-(1-y^{(i)})\frac{1}{1-\phi_y}\right)\\ &= \sum_{i=1}^{m}\left(y^{(i)}(1-\phi_y)-(1-y^{(i)})\phi_y\right)\\ &= \sum_{i=1}^{m}y^{(i)}-\sum_{i=1}^{m}\phi_y.\end{aligned}$$

从而得到参数φy 为：

$$\phi_y=\frac{\sum_{i=1}^{m}1\{y^{(i)}=1\}}{m}$$

第五集：生成学习算法

- 朴素贝叶斯
  - 由此最终得到了如下的参数表达

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

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

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}$$

φj|y=1（即p(xj=1|y=1)）的含义是所有类标为1且特征x为1的样本个数占类标为1的样本总数的比值（即所有垃圾邮件中出现词汇j的频率）；φj|y=0（即p(xj=1|y=0)）的含义是所有类标为0且特征x为1的样本个数占类标为0的样本总数的比值（即所有正常邮件中出现词汇j的频率）；Φy（即p(y=1)）的含义是所有类标为1的样本个数占样本总数的比值（即垃圾邮件占全部邮件的频率）

  - 从而可以通过如下所示的方法对于测试样本进行类标判断

$$p(y=1|x) = \frac{p(x|y=1)p(y=1)}{p(x)}$$

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

和以前一样，不论是p(y=0|x)，还是p(y=1|x)，只要x是给定的，这个式子的分母就是常数。因此可以只计算分子，只比较分子的大小。

第五集：生成学习算法

- 朴素贝叶斯
  - 实际上朴素贝叶斯同样可以推广到x取多元离散值的情况，可以用类似的方法获取参数p(xj=k|y=1)（其中k为xj的某个可能取值）

$$\mathrm{p}(\mathrm{xj}=\mathrm{k}|\mathrm{y}=1)=\frac{\sum_{i=1}^{m}1\{x_j^{(i)}=k\wedge y^{(i)}=1\}}{\sum_{i=1}^{m}1\{y^{(i)}=1\}}$$

  - 而且实际上对于x取连续值的情况可以采用分段的方法来将连续值转化为离散值，然后再使用朴素贝叶斯的方法
    - 可以采用信息增益的度量方法来决定如何将连续值转化为离散值的效果

第五集：生成学习算法

- 拉普拉斯平滑
  - 朴素贝叶斯方法有个致命的缺点就是对数据稀疏问题过于敏感。比如说在测试的邮件中出现了一个在训练中从来没有出现过的词（假设这个词是词典里面的第35000个词）
    - 那么当使用朴素贝叶斯方法对该邮件进行是否为垃圾邮件的判断时就会出现下面的情况：

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

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

由于p(y=1|x)和p(y=0|x)的结果均为0/0，因此我们无法判断该封邮件是否为垃圾邮件

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

$$= \frac{0}{0}.$$

第五集：生成学习算法

- 拉普拉斯平滑
  - 在上面那个例子中出现问题的地方在于在训练样本中没出现过的词不代表这个词不存在
    - 因此p(x35000=1|y=1)和p(x35000=1|y=0)的训练结果都是0，导致在计算p(y|x)时掩盖掉了其它词汇的作用，肯定不合理

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

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

    - 既然在训练样本中没出现过的词会出现问题，那最简单的解决办法就是要让所有词都出现过至少1次，这也就引出了称为拉普拉斯平滑(Laplace smoothing)的方法

第五集：生成学习算法

- 拉普拉斯平滑
  - 在平滑之前对于x取多元离散值{k1,k2,...,kn}的情况来说p(x=kj)的计算为

$$\mathrm{p(x=kj)}=\frac{\sum_{i=1}^{m}1\{x^{(i)}=kj\}}{m}$$

  - 解决这个问题最简单的方法就是初始时默认每一种都已经出现了一次，即x取每一种离散值的频数至少为1
    - 但与此同时样本的总数也要随之增加，由此对于p(x=kj)的计算为

$$\mathrm{p(x=kj)}=\frac{\sum_{i=1}^{m}1\{x^{(i)}=kj\}+1}{m+n}$$

第五集：生成学习算法

- 拉普拉斯平滑
  - 对应到之前垃圾邮件分类的例子，需要将参数做如下的改动

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

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

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}$$

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

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

    - 要注意到这里只对x做平滑，并没有要求一定要对y做平滑。由于一般都要求选择的测试集要能够覆盖到所有的可能类标y（否则这个测试集是不完全的），因此没有必要对y做平滑