Classification Algorithms

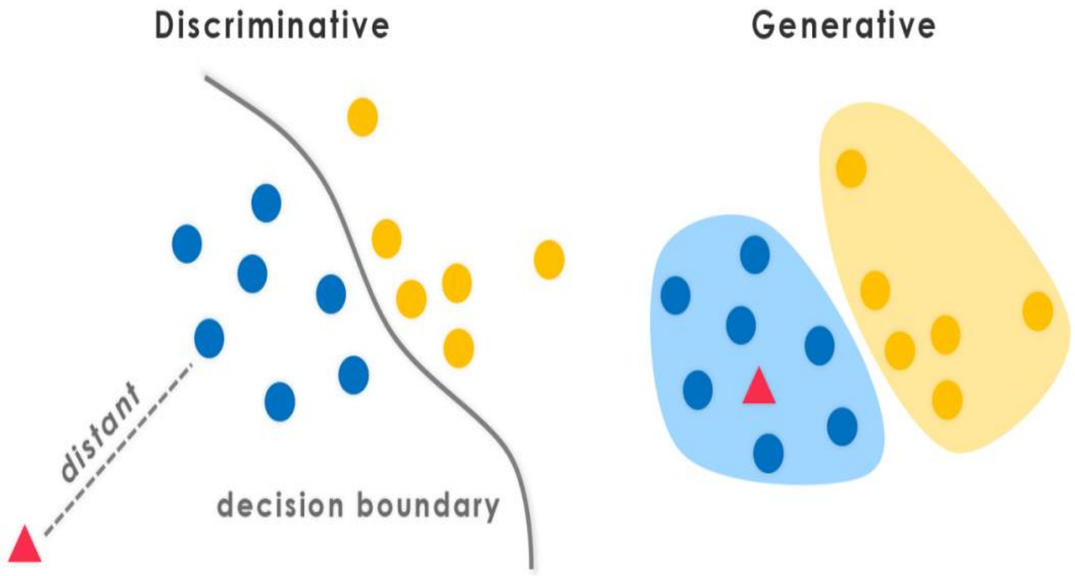
分类算法

分类Classification

- 响应变量 Y 为属性变量，取值范围在一个无序的集合 C 中
  - 图像分类：{是猫，不是猫}
  - 肿瘤分类：{良性，恶性}
  - 信用卡审批：{通过，不通过}
- 给定特征向量 $\boldsymbol{X}$，分类的任务是：构造一个函数 $f(\boldsymbol{X})$ 来对属性响应变量 Y 的值进行预测。
- 对于一个事件发生的情况，往往不能得到100%的预测。因此我们通常也非常关心一个观测属于 C 中每一类的概率是多少。

开始之前

- 分类算法主要分为**生成学习算法**(Generative Learning Algorithms)和**判别学习算法**(Discriminative learning algorithms)
- 生成学习算法假设 x 是由某概率分布生成而来对每一类中的x分别进行建模，最终利用贝叶斯定理来获得$P(y=k|x)$
- 判别学习算法直接对$P(y|x)$进行建模，例如：$P(y=1|x;\beta)=g(\beta^T x)=\frac{1}{1+e^{-\beta^T x}}$



判别学习与生成学习

- 判别学习算法（如logistic回归）同时查看两类数据，利用梯度下降算法，初始化参数，不断迭代最终找到一个分类边界
- 生成学习算法每次查看一类数据，对每类数据的特征分别建模，建立每一类的模型，然后对于新的数据判断其更像是来自于哪一类

Outline

- 线性（高斯）判别分析
- 二次判别分析
- 朴素贝叶斯
- 逻辑回归
- SOFTMAX 回归

贝叶斯定理与分类

- y给定$\boldsymbol{x}$时的后验概率：

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

- $P(y=1)$为y的先验分布
- $P(\boldsymbol{x}|y=1)$为给定$y=1$时$\boldsymbol{x}$的概率密度函数
- $P(\boldsymbol{x})=P(\boldsymbol{x}|y=1)\cdot P(y=1)+P(\boldsymbol{x}|y=0)\cdot P(y=0)$
- 利用$P(y=k|\boldsymbol{x})$来做分类：

$$\underset{k=0,1}{\operatorname{argmax}}\,P(y=k|\boldsymbol{x})=\underset{k=0,1}{\operatorname{argmax}}\frac{P(\boldsymbol{x}|y=k)P(y=k)}{P(\boldsymbol{x})}=\underset{k=0,1}{\operatorname{argmax}}\,P(\boldsymbol{x}|y=k)P(y=k)$$

高斯判别分析

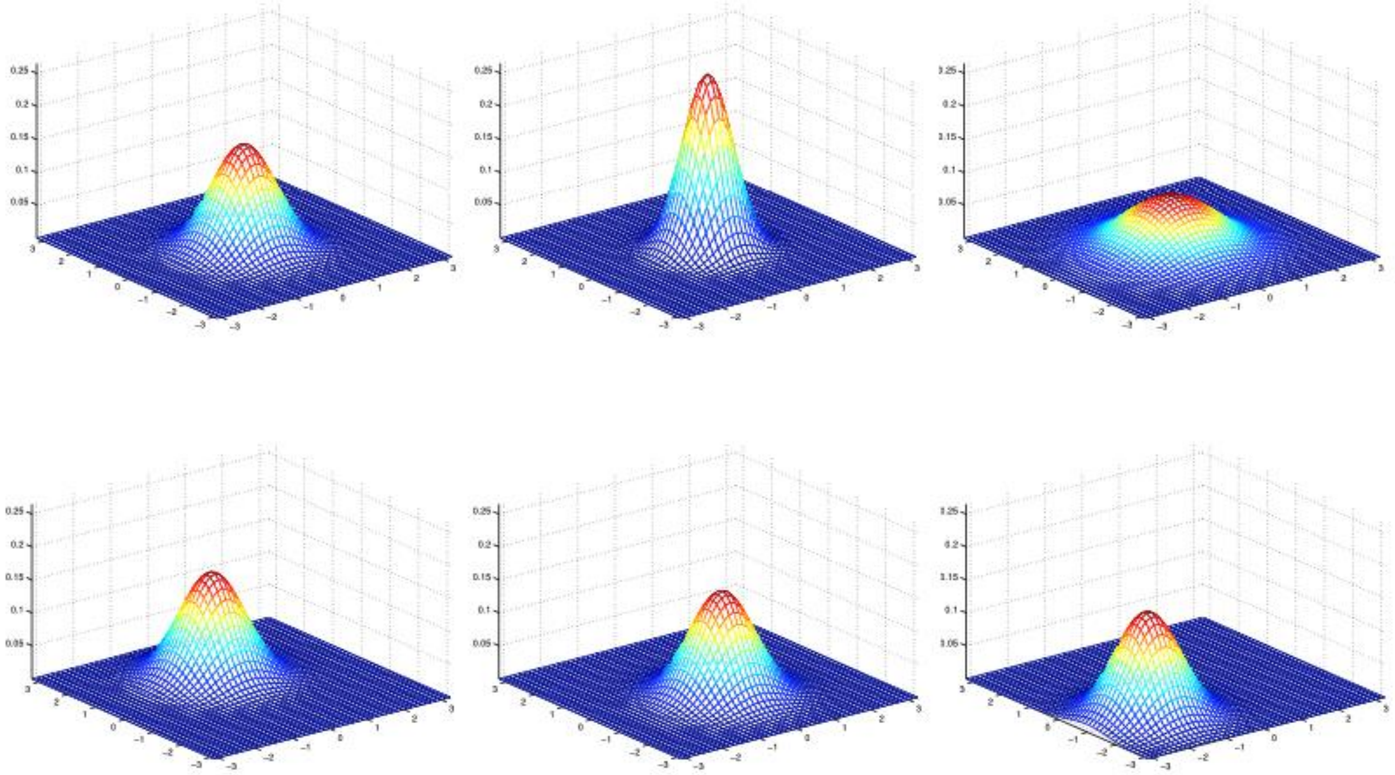
- $\boldsymbol{x} \in \mathbb{R}^d$（此时不再需要$x_0 = 1$）
- 关键假设：　$\boldsymbol{x}|y = k$是多元正态分布

- $\boldsymbol{z} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$，$\boldsymbol{\mu} \in \mathbb{R}^d$，　$\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$：包含多个随机变量的向量
- $E(\boldsymbol{z}) = \boldsymbol{\mu}$：包含各个随机变量均值的向量
- $Cov(\boldsymbol{z}) = E[(\boldsymbol{z} - \boldsymbol{\mu})(\boldsymbol{z} - \boldsymbol{\mu})^T] = \boldsymbol{\Sigma}$：协方差矩阵，描述了各个随机变量之间的相关性。它是一个对称正定矩阵，对角线上的元素是各个随机变量的方差，非对角线上的元素表示不同随机变量之间的协方差。

$$P(\boldsymbol{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{z} - \boldsymbol{\mu})\right)$$

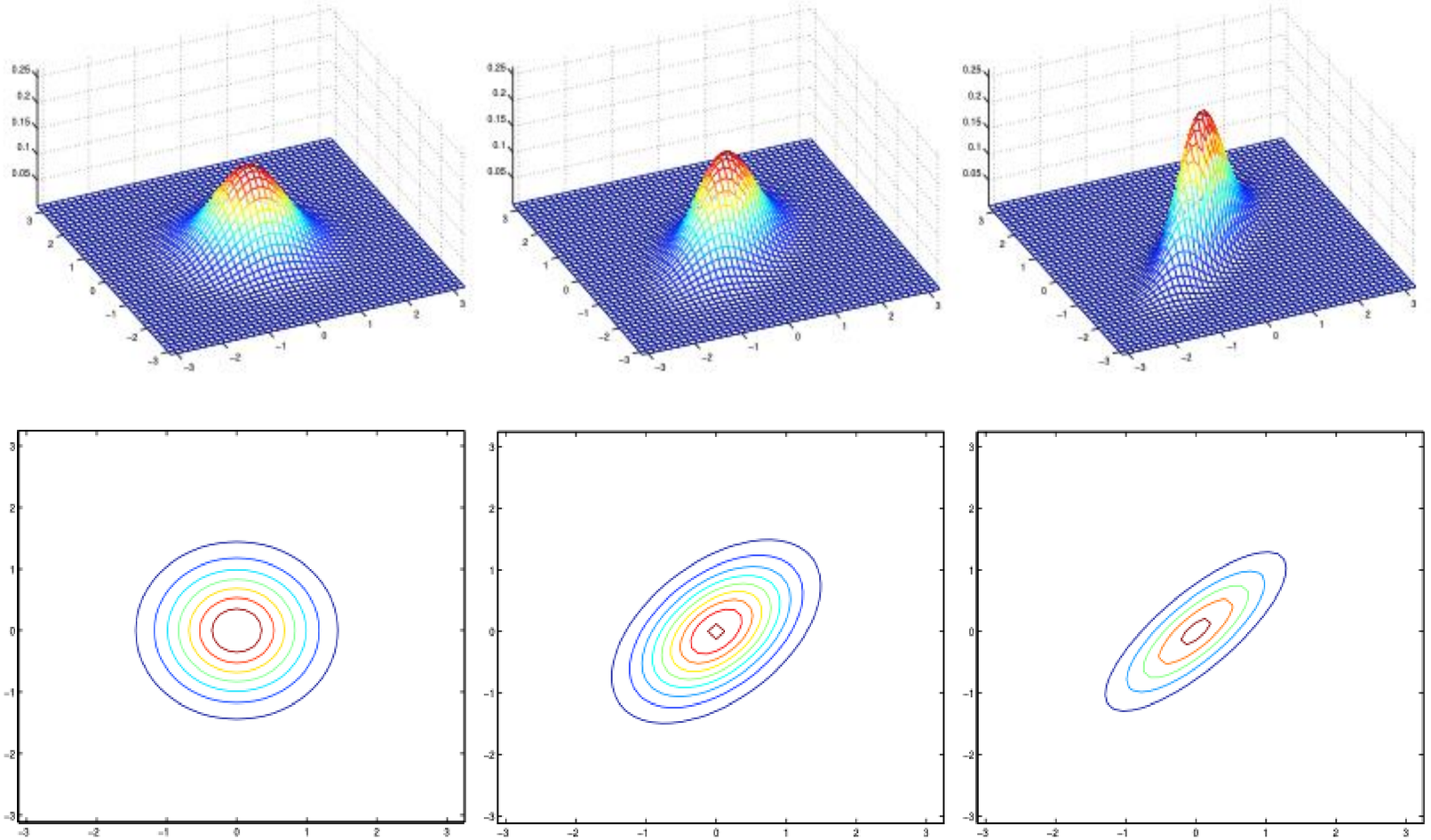
二元正态分布

- 第一行：$\mu = (0,0)^{\mathrm{T}},\ \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$
- 第二行：$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \mu = (0,1.5)^{\mathrm{T}}, \mu = (0,-0.5)^{\mathrm{T}}\ ,\ \mu = (-1.5,-1)^{\mathrm{T}}$



二元正态分布

- 第一行：$\mu = (0,0)^{\mathrm{T}},\ \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$



高斯判别分析Gaussian Discriminant Analysis

- $\boldsymbol{x}|y=0 \sim N(\boldsymbol{\mu_0}, \boldsymbol{\Sigma})$，$\boldsymbol{x}|y=1 \sim N(\boldsymbol{\mu_1}, \boldsymbol{\Sigma})$
- $P(\boldsymbol{x}|y=0)=\frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_0})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_0})\right)$
- $P(\boldsymbol{x}|y=1)=\frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_1})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_1})\right)$
- 两个正态分布均值不同，协方差矩阵相同

- $y \sim Bernoulli(\phi)$
- $P(y)=\phi^{y}(1-\phi)^{1-y}$

- GDA参数：ϕ，$\boldsymbol{\mu_0}$，$\boldsymbol{\mu_1}$，$\boldsymbol{\Sigma}$

GDA参数估计——极大似然

- 和往常一样，我们有训练数据集$\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$
- 最大化参数的联合似然：

$$L(\phi, \boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} P(\boldsymbol{x}^{(i)}, y^{(i)}; \phi, \boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} P(\boldsymbol{x}^{(i)} | y^{(i)}) P(y^{(i)})$$

- $\hat{\phi} = \frac{1}{n}\sum_{i=1}^{n} y^{(i)} = \frac{1}{n}\sum_{i=1}^{n} I(y^{(i)} = 1)$
- $\hat{\boldsymbol{\mu}}_{\boldsymbol{0}} = \frac{\sum_{i=1}^{n} I(y^{(i)}=0)\boldsymbol{x}^{(i)}}{\sum_{i=1}^{n} I(y^{(i)}=0)}$
- $\hat{\boldsymbol{\mu}}_{\boldsymbol{1}} = \frac{\sum_{i=1}^{n} I(y^{(i)}=1)\boldsymbol{x}^{(i)}}{\sum_{i=1}^{n} I(y^{(i)}=1)}$
- $\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n} \left(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_{y^{(i)}}\right)\left(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_{y^{(i)}}\right)^{T}$

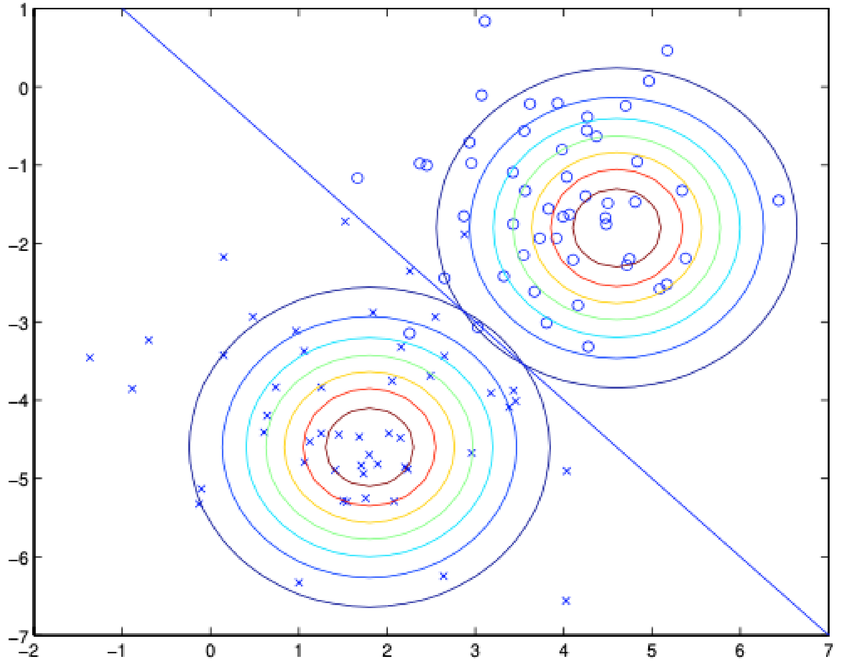
GDA预测

- $\underset{k=0,1}{\operatorname{argmax}} P(y=k|\boldsymbol{x}) = \underset{k=0,1}{\operatorname{argmax}} P(\boldsymbol{x}|y=k)P(y=k)$
- $y=1$需要$\frac{P(\boldsymbol{x}|y=1)P(y=1)}{P(\boldsymbol{x}|y=0)P(y=0)}>1$，即
- $$\frac{\frac{\phi}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_1})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_1})\right)}{\frac{1-\phi}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_0})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_0})\right)} = \frac{\phi}{1-\phi}\exp\left((\boldsymbol{\mu_1}-\boldsymbol{\mu_0})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\frac{1}{2}\boldsymbol{\mu_1}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_1}+\frac{1}{2}\boldsymbol{\mu_0}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_0}\right)>1$$
- 化简得到$(\boldsymbol{\mu_1}-\boldsymbol{\mu_0})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x} > \frac{1}{2}\boldsymbol{\mu_1}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_1}-\frac{1}{2}\boldsymbol{\mu_0}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_0}-\log\left(\frac{\phi}{1-\phi}\right)$
- 决策边界$(\boldsymbol{\mu_1}-\boldsymbol{\mu_0})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\frac{1}{2}\boldsymbol{\mu_1}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_1}+\frac{1}{2}\boldsymbol{\mu_0}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_0}+\log\left(\frac{\phi}{1-\phi}\right)=0$

GDA预测

- $\boldsymbol{\Sigma^{-1}(\mu_1 - \mu_0)}$ 称为判别向量，与决策边界垂直
- 决策边界是线性的，这也是GDA别名LDA的由来
- 导致决策边界线性的主要原因是因为我们对两个正态分布假设了相同的 $\boldsymbol{\Sigma}$
- 如果我们对不同的类假设不同的$\boldsymbol{\Sigma}$，就导出了另一种方法：

二次判别分析（Quadratic Discriminant Analysis）



混淆矩阵Confusion Matrix

- 针对两分类的情形，可能有四种情况：
  - TN-True Negative：真实分类为0，预测分类为0
  - TP-True Positive：真实分类为1，预测分类为1
  - FP-False Positive：真实分类为0，预测分类为1
  - FN-False Negative：真实分类为1，预测分类为0

		真实分类		
		0	1	Total
预测分类	0	TN	FN	N^*
	1	FP	TP	P^*
	Total	N	P	n

- 准确率 Accuracy=$(TN+TP)/n$：整体预测准确率。由于样本不平衡，准确率可能水分较大
- 精准率 Precision=TP/P^*：在所有被预测为 1 的样本中实际为 1 的样本所占比例，我们有多大的把握可以预测正确。
- 敏感性 Sensitivity =TP/P。在实际为 1 的样本中被预测为 1 的样本所占比例。
- 特异性 Specificity= TN/N：在实际为 0 的样本中被预测为 0 的样本所占比例。

例—Credit Card Default

- Training error=(23+252)/10000=2.75%
- Accuracy=(9644+81)/10000=97.25%=1-2.75%
- 看起来还不错，但有两个潜在问题值得注意：
  - Training error往往小于test error，而test error才是我们真正关心的。换句话说，当方法应用在测试集上的时候，预期结果肯定是会更差的，因为我们刻意调整了模型参数来迎合训练数据。
  - 训练集中只有3.33%的个体发生了违约。一个简单却无效的分类器可以无脑预测个体不会违约，最终的training error只有3.33%，只比GDA的2.75%高一点。这就是样本不平衡带来的问题。
- Sensitivity=81/333=24.3%
- Specificity=9644/9667=99.8%
- 此时Accuracy高意义不大，因为Sensitivity 太小：保险公司更想避免对违约用户的错误分类。
- 从Sensitivity来看，对违约用户的分类结果很差。能否调整GDA分类器呢？

		True default status		
		No	Yes	Total
Predicted default status	No	9644	252	9896
	Yes	23	81	104
	Total	9667	333	10000

例—Credit Card Default

- GDA本质上是在近似Bayes分类器：$\underset{k=0,1}{\operatorname{argmax}} P(y = k|\boldsymbol{x})$
- 在两分类的情况下，我们判定是否违约的条件是对违约的后验概率施加临界值。例如：
$$P(\text{default} = \text{Yes} \mid X = x) > 0.5$$
- 如果更关注违约客户的错误分类，我们可以调低临界值，得到更加激进的结果。例如：
$$P(\text{default} = \text{Yes} \mid X = x) > 0.2$$
- 此时，更多的样本被分类为1，更少的样本被分类为0； Sensitivity上升， Specificity下降。

- Accuracy=(9432+195)/10000=96.27% （略降）
- Specificity=9432/9667=97.56% （略降）
- Sensitivity=195/333=58.55% （大涨）

		True default status		
		No	Yes	Total
Predicted default status	No	9432	138	9570
	Yes	235	195	430
	Total	9667	333	10000

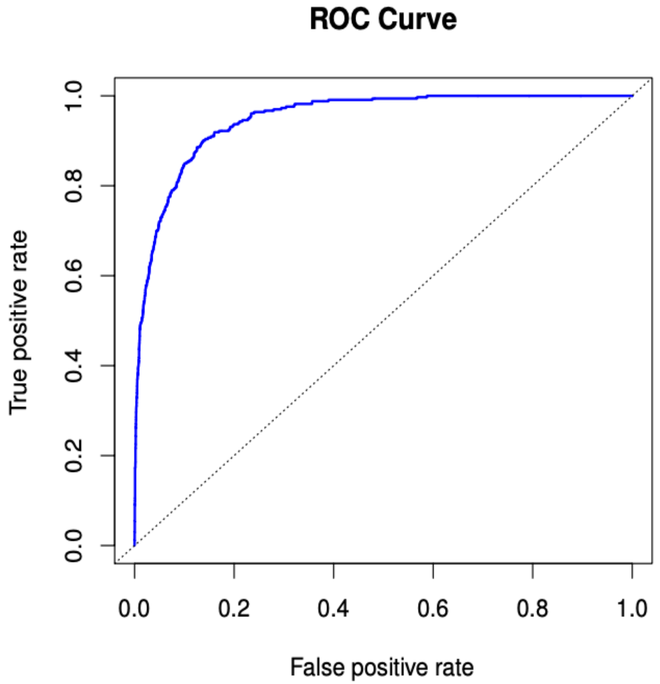
例—Credit Card Default

- 随着临界值的增加，整体的错误率变小，Specificity变大，Sensitivity变小
- 实际中选择最优的临界值离不开领域知识的帮助，例如违约用户所带来的损失的相关详细信息。

- 怎么样对算法的表现作出综合评估呢？

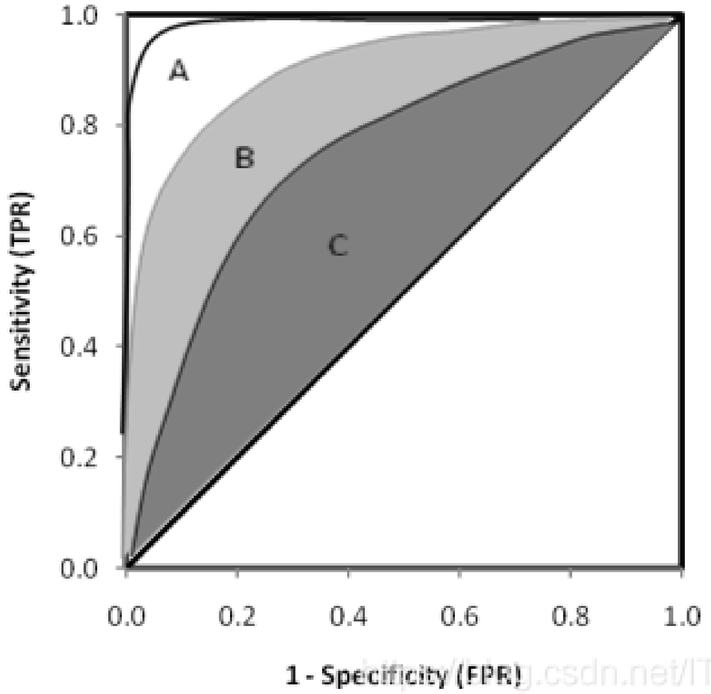
ROC曲线

- ROC，Receiver Operating Characteristic，“**受试者工作特征曲线**”
- 横坐标是1-specificity，纵坐标是sensitivity。
- 随着临界值的变化，两者同增同减。
- 给定分类器，对每个临界值，我们可以根据其在测试数据上的表现得到一个点对。
- 调整分类器的临界值，我们就可以得到一个经过(0, 0)，(1, 1)的曲线，即此分类器的ROC曲线。
- 什么样是好的ROC曲线？



AUC (Area Under Curve)

- AUC被定义为ROC曲线下的面积，取值范围一般在0.5和1之间。
- 一个理想的ROC曲线会顶到左上角(0,1)。此时存在至少一个临界值能得出完美预测。因此AUC越大说明分类器表现越好。
- ROC用来比较不同的分类器非常有用，综合考虑了所有可能的临界值。

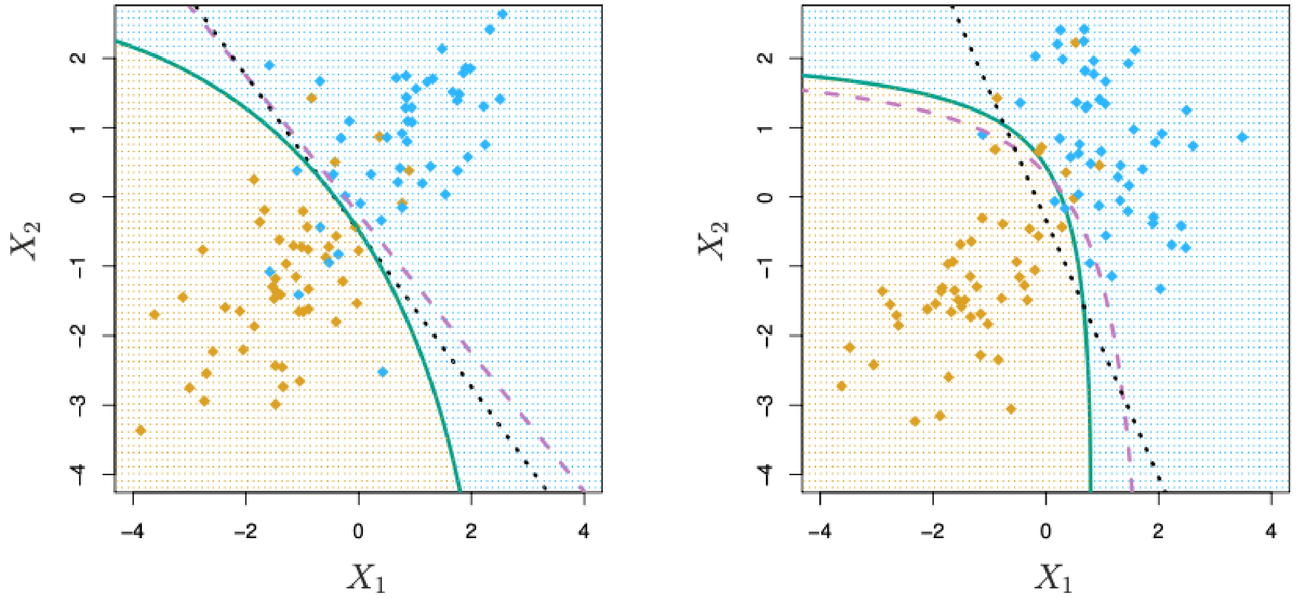


QDA

- $y \sim Bernoulli(\phi)$
- $P(y) = \phi^{y}(1-\phi)^{1-y}$
- $\boldsymbol{x}|y=0 \sim N(\boldsymbol{\mu_0}, \boldsymbol{\Sigma}_0)$，$\boldsymbol{x}|y=1 \sim N(\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})$
- $P(\boldsymbol{x}|y=0) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma_0}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_0})^{\mathrm{T}}\boldsymbol{\Sigma_0^{-1}}(\boldsymbol{x}-\boldsymbol{\mu_0})\right)$
- $P(\boldsymbol{x}|y=1) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma_1}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_1})^{\mathrm{T}}\boldsymbol{\Sigma_1^{-1}}(\boldsymbol{x}-\boldsymbol{\mu_1})\right)$
- LDA参数：ϕ $\boldsymbol{\mu_0}$ $\boldsymbol{\mu_1}$ $\boldsymbol{\Sigma_0}$ $\boldsymbol{\Sigma_1}$
- 两个正态分布均值不同，协方差矩阵也不同

QDA预测

- $\underset{k=0,1}{\operatorname{argmax}} P(y=k|\boldsymbol{x}) = \underset{k=0,1}{\operatorname{argmax}} P(\boldsymbol{x}|y=k)P(y=k)$
- 决策边界是二次的，这也是QDA中Quadratic的由来
- $\underset{k=0,1}{\operatorname{argmax}} \delta_k(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_k})^T\boldsymbol{\Sigma_k}(\boldsymbol{x}-\boldsymbol{\mu_k}) - \frac{1}{2}\log|\boldsymbol{\Sigma_k}| + \log P(y=k)$



GDA & QDA

- QDA由于对不同的类假设了不同的协方差，需要估计更多的参数，当d和K很大时这将是个严重的问题，可能会导致较高的方差
- GDA对不同的类假设了相同的协方差，当这样的假设不成立时（最优决策边界不是线性时），可能会导致较高的偏差
- 粗略的说，当训练样本较少时，减小方差变得十分关键，此时应选择GDA；当训练样本较多时，或者当不同类的协方差明显不同时，应选择QDA

朴素贝叶斯 Naive Bayes

- 在GDA当中，特征 $\boldsymbol{x}$ 是连续的，在很多应用中 $\boldsymbol{x}$ 是离散的。
- 我们基于一个垃圾邮件分类的例子来介绍NB。首要的问题是如何用特征向量来表示一封邮件？
- 假设我们有一组训练数据，由一些被标注为垃圾和非垃圾的邮件组成。
- 我们用一个和英语字典中单词数量同样长度的特征向量来表示邮件。例如，右图代表一封包含”a” “buy”、不包含“aardvark” “aardwolf” ”zygmurgy”的邮件。

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

Naive Bayes

- 有了特征向量$\boldsymbol{x} \in \{0,1\}^d$，我们现在要构建生成模型，即对$P(\boldsymbol{x}|y)$和$P(y)$建模。
- 假设只考虑字典中的前10000个单词，即$d = 10000$
- $\boldsymbol{x}$的可能取值有2^{10000}个，以此为基础的多项分布有2^{10000}个参数需要估计

- 为解决这一问题，我们做一个很强的假设：给定 y 的情况下， x_j之间相互独立

$$\begin{aligned} P(x_1, \dots, x_{10000}|y) &= P(x_1|y)P(x_2|y, x_1) \dots P(x_{10000}|y, x_1, \dots, x_{9999}) \\ &= P(x_1|y)P(x_2|y) \dots P(x_{10000}|y) \\ &= \prod_{j=1}^{d} P(x_j|y) \end{aligned}$$

训练Naive Bayes

- 模型参数：$\phi_y = P(y=1)$, $\phi_{j|y=1} = P(x_j = 1|y=1)$, $\phi_{j|y=0} = P(x_j = 1|y=0)$
- 给定训练数据，我们可以写出数据的联合似然函数：

$$L(\phi_y, \phi_{j|y=1}, \phi_{j|y=0}) = \prod_{i=1}^{n} P(x^{(i)}, y^{(i)}; \phi_y, \phi_{j|y=1}, \phi_{j|y=0})$$

- 极大似然估计：

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

Naive Bayes预测

- 估计好参数之后，我们只需要计算后验概率：

$$
\begin{aligned}
p(y=1|x) &= \frac{p(x|y=1)p(y=1)}{p(x)} \\
&= \frac{\left(\prod_{j=1}^{d} p(x_j|y=1)\right) p(y=1)}{\left(\prod_{j=1}^{d} p(x_j|y=1)\right) p(y=1) + \left(\prod_{j=1}^{d} p(x_j|y=0)\right) p(y=0)},
\end{aligned}
$$

并设置一定的阈值即可。

讨论

- 为什么朴素贝叶斯的假设这么有效？因为我们完全忽视了特征之间的相关性，使问题变得非常简单。
- 我们真的相信这个朴素的假设吗？在绝大多数情况下并不是。但尽管如此，这个假设往往能得到不错的结果，尤其是在 d 很大，n 不大，无法很好的估计特征的联合分布的时候。
- 朴素贝叶斯引入了一些偏差，但是极大的减小了方差，最终的分类器在偏差方差平衡上做的很好。
- All models are wrong, but some are useful.

能否用线性回归来做分类？

- 假设一个人送到了急诊，我们要根据病人的症状判断他的状况（中风、吸毒过量、癫痫）
- 若使用线性回归，我们需要对属性响应变量 Y 进行数值编码：

$$Y = \begin{cases} 1, \text{中风} \\ 2, \text{吸毒过量} \\ 3, \text{癫痫} \end{cases}$$

- 编码假设了几种症状严重程度的高低，同时假设了1-2 ， 2-3 之间差距相等
- 另一方面，若调整编码顺序，可能会得到截然不同的预测结果：

$$Y = \begin{cases} 1, \text{吸毒过量} \\ 2, \text{中风} \\ 3, \text{癫痫} \end{cases}$$

续

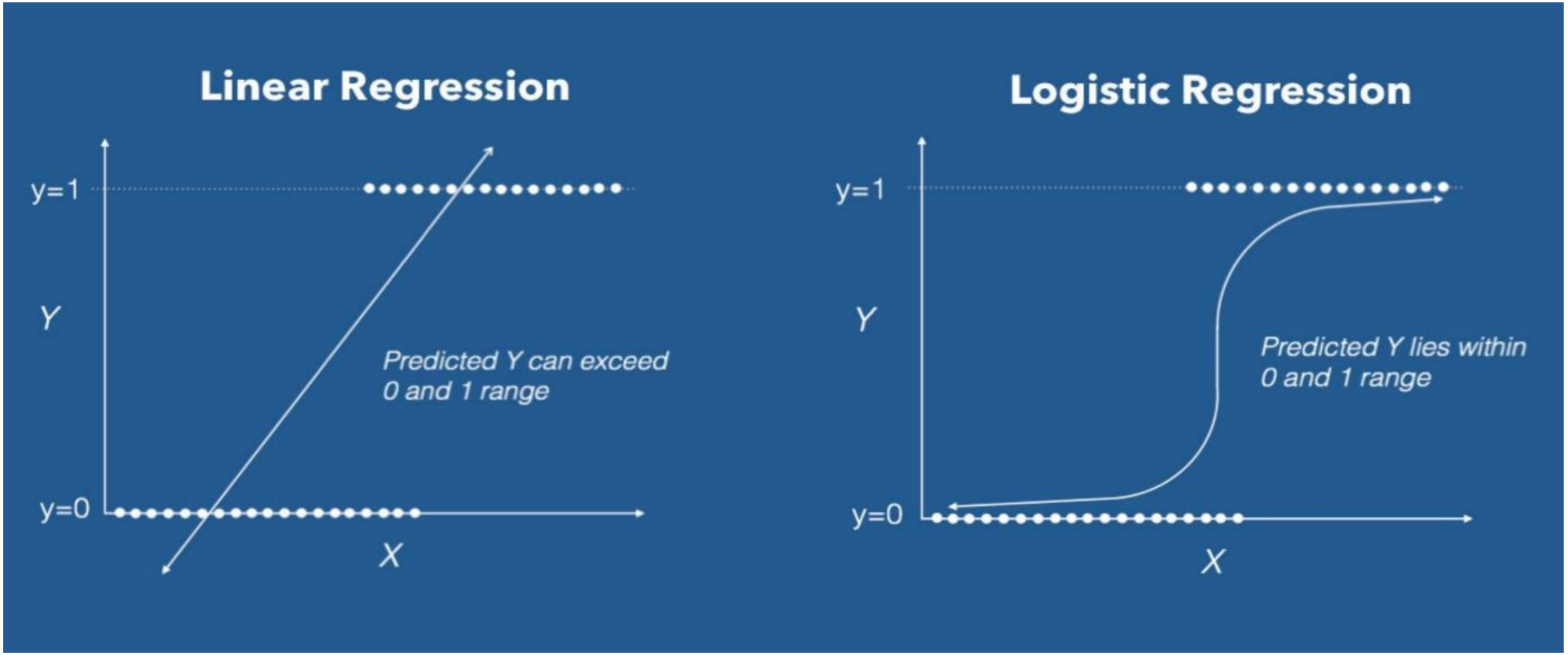
- 假设在信用卡违约的分类任务中，我们对 Y 进行数值编码：

$$Y = \begin{cases} 0, \text{没有违约} \\ 1, \text{违约} \end{cases}$$

我们是否可以对 X 和 Y 构造线性回归，并将 $\hat{Y} > 0.5$ 分类为违约呢？

- 在两分类的场景下，线性回归可以充当分类器的角色
- 但是，线性回归会输出大于1或小于0的值，丧失了概率上的可解释性
- 同时，改变一个观测值可能会极大地影响线性回归的分类结果

线性回归与逻辑回归


Linear Regression
y=1
Y
y=0
X
Predicted Y can exceed
0 and 1 range
Logistic Regression
y=1
Y
y=0
X
Predicted Y lies within
0 and 1 range

逻辑回归

逻辑回归

- 为方便表示，记 $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_d)^T$， $\boldsymbol{X} = (X_0, X_1, \ldots, X_d)^T, X_0 = 1$
- 线性回归：$Y = f_{\boldsymbol{\beta}}(\boldsymbol{X}) = \boldsymbol{\beta}^T \boldsymbol{X}$
- 逻辑回归不再直接对响应变量 Y 进行直接建模，而是对 Y 属于某类的概率进行建模
- 一般令 $Y \in \{0, 1\}$。我们进一步假设：

$$P(Y = 1|\boldsymbol{X}; \boldsymbol{\beta}) = f_{\boldsymbol{\beta}}(\boldsymbol{X}) = g(\boldsymbol{\beta}^T \boldsymbol{X}) = \frac{1}{1 + e^{-\boldsymbol{\beta}^T \boldsymbol{X}}} = \frac{e^{\boldsymbol{\beta}^T \boldsymbol{X}}}{1 + e^{\boldsymbol{\beta}^T \boldsymbol{X}}}$$

$$P(Y = 0|\boldsymbol{X}; \boldsymbol{\beta}) = 1 - P(Y = 1|\boldsymbol{X}; \boldsymbol{\beta}) = \frac{e^{-\boldsymbol{\beta}^T \boldsymbol{X}}}{1 + e^{-\boldsymbol{\beta}^T \boldsymbol{X}}} = \frac{1}{1 + e^{\boldsymbol{\beta}^T \boldsymbol{X}}}$$

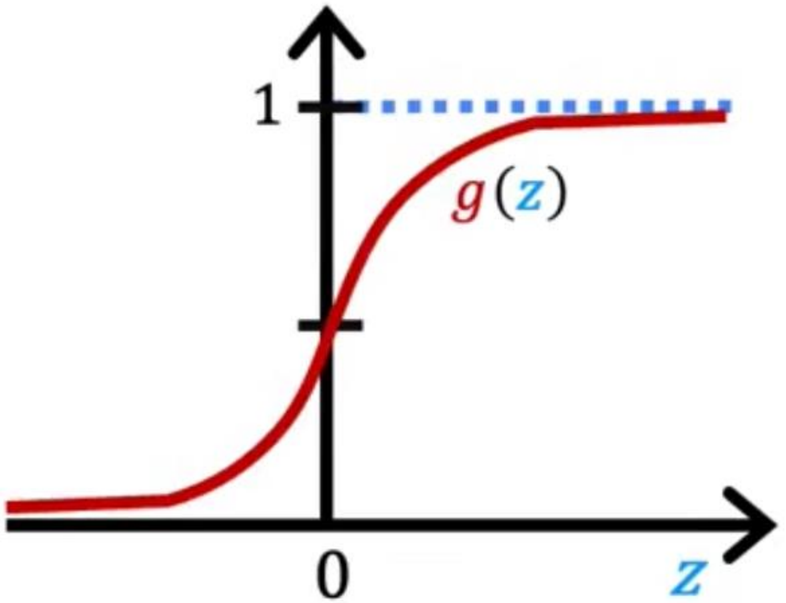
- 另一种形式：

$$\log\left(\frac{P(Y = 1|\boldsymbol{X}; \boldsymbol{\beta})}{1 - P(Y = 1|\boldsymbol{X}; \boldsymbol{\beta})}\right) = \boldsymbol{\beta}^T \boldsymbol{X}$$

等式左边被称作对数比（log odds）或 logit 变换

Logistic函数

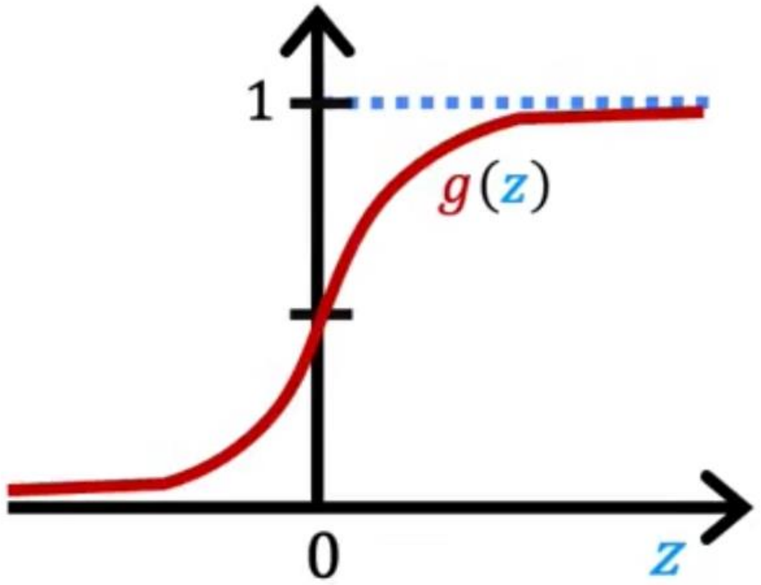
- $g(z)=\frac{1}{1+e^{-z}}$ 被称作 logistic 函数，或 sigmoid 函数
  - 单调递增
  - $z\to+\infty$ 时 $g(z)\to 1$； $z\to-\infty$ 时 $g(z)\to 0$
  - $z=0$ 时 $g(z)=0.5$，并且关于0.5中心对称
- 这些性质使logistic函数很好的符合了两分类的要求
- 后续我们将会看到，logistic函数是在一定的概率假设下的自然结果。



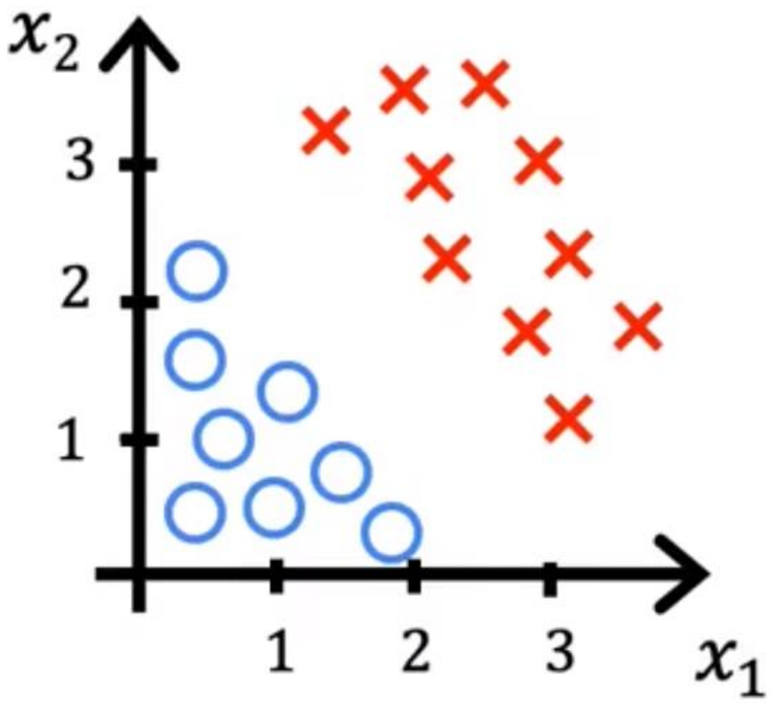
- 导数由其本身构成：$g'(z)=g(z)\big(1-g(z)\big)$

逻辑回归是如何运行的

- 假设参数$\boldsymbol{\beta}$已经训练完成
- $z = \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X} \rightarrow g(z) = \frac{1}{1+e^{-z}}$
- $f_{\boldsymbol{\beta}}(\boldsymbol{X}) = g\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X}\right) = \frac{1}{1+e^{-\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X}}} = P(Y = 1|\boldsymbol{X};\boldsymbol{\beta})$
- 设定临界值为0.5并判定 $f_{\boldsymbol{\beta}}(\boldsymbol{X})$ 是否大于等于0.5
  - 是：$\hat{Y} = 1$　　否：$\hat{Y} = 0$
- 什么时候 $f_{\boldsymbol{\beta}}(\boldsymbol{X}) = g(z) \geq 0.5$ ？
  - $z = \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X} \geq 0$
- 什么时候 $f_{\boldsymbol{\beta}}(\boldsymbol{X}) = g(z) < 0.5$ ？
  - $z = \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X} < 0$

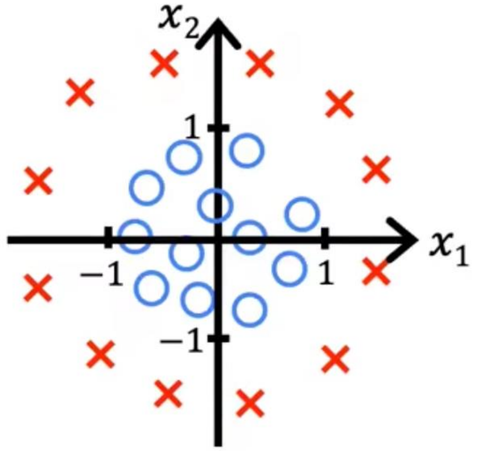


决策边界Decision Boundary

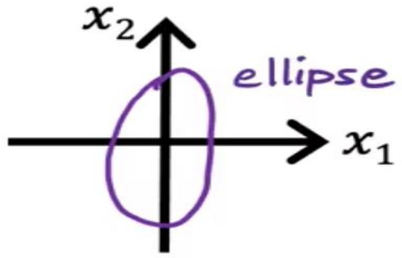


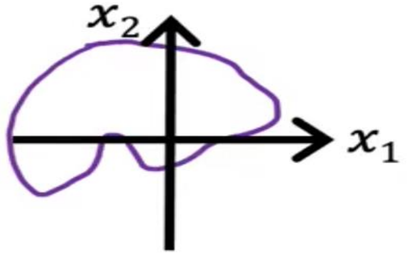
- $f_{\boldsymbol{\beta}}(\boldsymbol{X}) = g(z) = g(\beta_0 X_0 + \beta_1 X_1 + \beta_2 X_2)$
- 假设在本例中， $\beta_0 = -3$，$\beta_1 = \beta_2 = 1$
- **决策边界：** $z = \boldsymbol{\beta}^T \boldsymbol{X} = X_1 + X_2 - 3 = 0$
- 在决策边界上，我们对Y等于0还是1持中立态度。
- 常规的逻辑回归是线性决策边界，也可以构造更加复杂的特征来获得非线性决策边界。

非线性决策边界



- $f_{\boldsymbol{\beta}}(\boldsymbol{X}) = g(z) = g(\beta_0 X_0 + \beta_1 X_1^2 + \beta_2 X_2^2)$
- 假设在本例中， $\beta_0 = -1$，$\beta_1 = \beta_2 = 1$
- **决策边界：** $X_1^2 + X_2^2 - 1 = 0$ 或 $X_1^2 + X_2^2 = 1$





- $f_{\boldsymbol{\beta}}(\boldsymbol{X}) = g(z) = g(\beta_0 X_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_1 X_2 + \beta_5 X_2^2 + \beta_6 X_1^3 + \cdots)$
- 可以得到更加复杂的非线性决策边界
- 此时应注意过拟合问题

损失函数

- 训练数据集 $\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)$, $\boldsymbol{x}^{(i)} = \left(x_0^{(i)}, x_1^{(i)} \ldots, x_d^{(i)}\right)^T$, $y^{(i)} \in \{0,1\}$, $i = 1, \ldots, n$
- 目标：找到一组参数$\boldsymbol{\beta}$，使得$y^{(i)} \approx f_{\boldsymbol{\beta}}\left(\boldsymbol{x}^{(i)}\right) = g\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right) \triangleq \hat{y}^{(i)}$
- 类似线性回归，我们可以定义残差平方和损失：$\frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$
- 问题：**非凸！**梯度下降算法容易陷入局部最优
- 单个样本损失函数：$L\left(y^{(i)}, \hat{y}^{(i)}\right) = -\left(y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)\right)$
  - $y^{(i)} = 1$：$L\left(y^{(i)}, \hat{y}^{(i)}\right) = -\log \hat{y}^{(i)}$，$\hat{y}^{(i)}$趋向于1时$L$趋向于0，反之趋向于无穷
  - $y^{(i)} = 0$：$L\left(y^{(i)}, \hat{y}^{(i)}\right) = -\log\left(1 - \hat{y}^{(i)}\right)$，$\hat{y}^{(i)}$趋向于0时$L$趋向于0，反之趋向于无穷
- **整体损失函数：$J(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)\right)$，是凸函数**

概率解释——损失函数的由来

- 模型假设：$P(Y=1|\boldsymbol{X};\boldsymbol{\beta})=g(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X})$，$P(Y=0|\boldsymbol{X};\boldsymbol{\beta})=1-g(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X})$
- $P(Y|\boldsymbol{X};\boldsymbol{\beta})=\left(g(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X})\right)^{Y}\left(1-g(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X})\right)^{1-Y}$
- 对于训练数据集 $(\boldsymbol{x}^{(i)},y^{(i)}),i=1,\dots,n$，条件似然函数为：

$$L(\boldsymbol{\beta})=\prod_{i=1}^{n}P\left(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\beta}\right)=\prod_{i=1}^{n}\left(g(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(i)})\right)^{y^{(i)}}\left(1-g(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(i)})\right)^{1-y^{(i)}}$$

- 对数似然函数为：

$$l(\boldsymbol{\beta})=\log L(\boldsymbol{\beta})=\sum_{i=1}^{n}y^{(i)}\log\left(g(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(i)})\right)+\left(1-y^{(i)}\right)\log\left(1-g(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(i)})\right)$$

- 最大化对数似然函数 $l(\boldsymbol{\beta})$ 与最小化损失函数 $J(\boldsymbol{\beta})$ 等价

梯度下降算法

- $\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\boldsymbol{\beta})$ simultaneously for $j = 0, \dots, d$

- 如果只有一个训练数据$(\boldsymbol{x}, y)$ ：

$$\begin{aligned}
\frac{\partial}{\partial \beta_j} J(\boldsymbol{\beta}) &= -\left(y \frac{1}{g(\boldsymbol{\beta}^T \boldsymbol{x})} - (1-y) \frac{1}{1 - g(\boldsymbol{\beta}^T \boldsymbol{x})} \right) \frac{\partial}{\partial \beta_j} g(\boldsymbol{\beta}^T \boldsymbol{x}) \\
&= -\left(y \frac{1}{g(\boldsymbol{\beta}^T \boldsymbol{x})} - (1-y) \frac{1}{1 - g(\boldsymbol{\beta}^T \boldsymbol{x})} \right) g(\boldsymbol{\beta}^T \boldsymbol{x}) \left(1 - g(\boldsymbol{\beta}^T \boldsymbol{x})\right) \frac{\partial}{\partial \beta_j} \boldsymbol{\beta}^T \boldsymbol{x} \\
&= -\left(y \left(1 - g(\boldsymbol{\beta}^T \boldsymbol{x})\right) - (1-y) g(\boldsymbol{\beta}^T \boldsymbol{x}) \right) x_j \\
&= -\left(y - g(\boldsymbol{\beta}^T \boldsymbol{x}) \right) x_j
\end{aligned}$$

- 随机梯度下降算法：$\beta_j := \beta_j + \alpha \left(y^{(i)} - g(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}) \right) x_j^{(i)}$

另一种优化方法——牛顿法

- NR算法是十分著名的优化算法，其在统计计算中有着广泛的应用。
- NR算法有着二次的收敛速率(Quadratic convergence rate)。
- NR算法的主要思想是通过构造严格凸或者凹的二次型函数去局部逼近原函数，从而使在每次迭代时，只需要优化该二次型函数即可。

另一种优化方法——牛顿法

- 假设我们考虑如下优化问题：
- $\min\ \mathrm{f}(\boldsymbol{x}), \boldsymbol{x} \in \mathcal{X}$
- 这里要求f为二阶可导的凸函数。
- 那么给定当前的估计$\boldsymbol{x}^t$，有
- $\mathrm{f}(\boldsymbol{x}) \approx\ \mathrm{f}(\boldsymbol{x}^t) + (\boldsymbol{x}-\boldsymbol{x}^t)^T\ \nabla \mathrm{f}(\boldsymbol{x}^t) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^t)^T \nabla^2 \mathrm{f}(\boldsymbol{x}^t)\ (\boldsymbol{x}-\boldsymbol{x}^t)$
- 因而，　可以得到下一步的迭代
- $\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \left(\nabla^2 f(\boldsymbol{x}^t\right)^{-1} \nabla(\boldsymbol{x}^t)$

另一种优化方法——牛顿法

- 对于我们的问题， $\boldsymbol{\beta}$是一个p维向量，我们想要找到$J(\boldsymbol{\beta})$取最小值的$\boldsymbol{\beta}$:

$$\boldsymbol{\beta} := \boldsymbol{\beta} - \boldsymbol{H}^{-1}\nabla_{\boldsymbol{\beta}} J(\boldsymbol{\beta})$$

其中，$\nabla_{\boldsymbol{\beta}} J(\boldsymbol{\beta})$是$J(\boldsymbol{\beta})$关于$\beta_j$的偏导构成的向量， $\boldsymbol{H}$是一个$(p+1)\times(p+1)$维的矩阵，称为*Hessian*:

$$H_{ij} = \frac{\partial^2 J(\boldsymbol{\beta})}{\partial \beta_i \partial \beta_j}$$

另一种优化方法——牛顿法

- 逻辑回归中

$$\frac{\partial J(\beta)}{\partial \beta} = -\sum_{i=1}^{n}(y_i - g(\beta^T \mathbf{X}_i))\mathbf{X}_i$$

$$\frac{\partial^2 J(\beta)}{\partial \beta^T \partial \beta} = \sum_{i=1}^{n} g(\beta^T \mathbf{X}_i) g(1 - \beta^T \mathbf{X}_i)\mathbf{X}_i$$

- 逻辑回归中牛顿法的迭代为

$$\begin{aligned}\beta^{(t+1)} =& \beta^{(t)} + (\mathbf{X}^T \mathbf{W}^{(t)} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{y} - \mathbf{g}^{(t)}) \\ =& (\mathbf{X}^T \mathbf{W}^{(t)} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W}^{(t)} (\mathbf{X}\beta^{(t)} + (\mathbf{W}^{(t)})^{-1}(\mathbf{y} - \mathbf{g}^{(t)}))\end{aligned}$$

其中$g^{(t)}$为$g(\beta^T X_i)$的向量，$W^{(t)}$为对角元素为$g(\beta^T X_i)(1 - g(\beta^T X_i)$的对角矩阵

另一种优化方法——牛顿法

- 可以将迭代分为两步
  1. 计算响应变量

$$\mathbf{z}^{(t+1)} = \mathbf{X}\beta^{(t)} + (\mathbf{W}^{(t)})^{-1}(\mathbf{y} - \mathbf{g}^{(t)})$$

  2. 更新权重

$$\beta^{(t+1)} = (\mathbf{X}^T\mathbf{W}^{(t)}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}^{(t)}\mathbf{z}^{(t+1)}$$

- 这个算法为迭代加权最小二乘(IRLS)，每次迭代解决加权最小二乘问题。
- 收敛速度优于梯度下降，需要的迭代次数更少，但当p很大时计算过于缓慢

信用卡数据分析--一元逻辑回归

- $\hat{P}(\text{default} = \text{yes}|\text{balence} = 1000) = \frac{1}{1+e^{-(-10.6513+0.0055*1000)}} = 0.00576$
- $\hat{P}(\text{default} = \text{yes}|\text{balence} = 2000) = \frac{1}{1+e^{-(-10.6513+0.0055*2000)}} = 0.586$

	Coefficient	Std. error	z-statistic	p-value
Intercept	−10.6513	0.3612	−29.5	<0.0001
balance	0.0055	0.0002	24.9	<0.0001

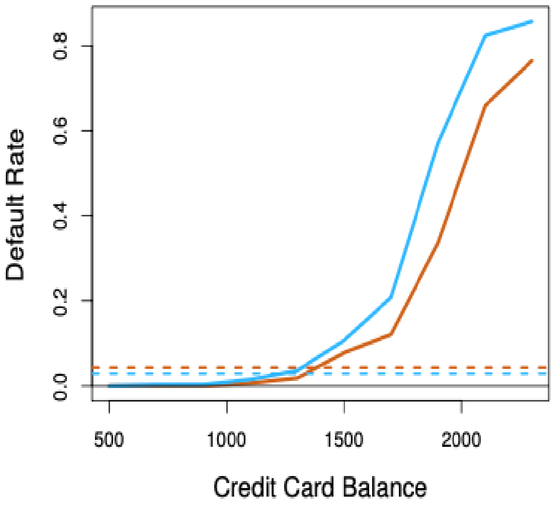
- $\hat{P}(\text{default} = \text{yes}|\text{student} = \text{yes}) = \frac{1}{1+e^{-(-3.5041+0.4049*1)}} = 0.0431$
- $\hat{P}(\text{default} = \text{yes}|\text{student} = \text{no}) = \frac{1}{1+e^{-(-3.5041+0.4049*0)}} = 0.0292$

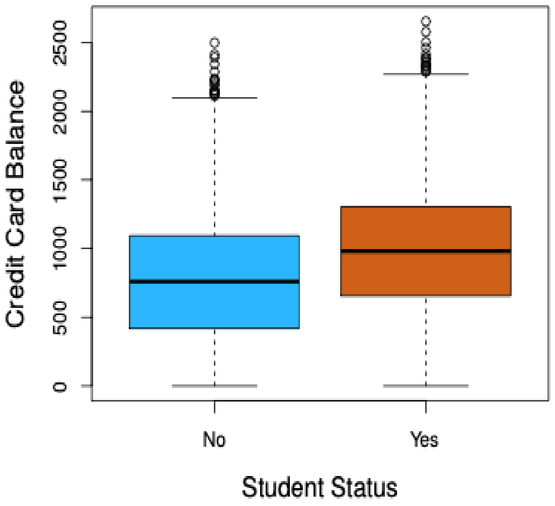
	Coefficient	Std. error	z-statistic	p-value
Intercept	−3.5041	0.0707	−49.55	<0.0001
student[Yes]	0.4049	0.1150	3.52	0.0004

信用卡数据分析-多元逻辑回归

- 等等！Student的系数怎么变成了负的？
- 学生倾向于有着更高的信用卡账单，所以倾向于有着更高的违约率；　　在没有任何信用卡账单信息的情况下，一个学生比一个非学生的风险显然是更高的。
- 但是，对于给定的信用卡账单，学生的违约率总是低于非学生
- 这样的现象叫做未观测混杂(unmeasured confounding），多元逻辑回归可以消除这一问题
- 辛普森悖论

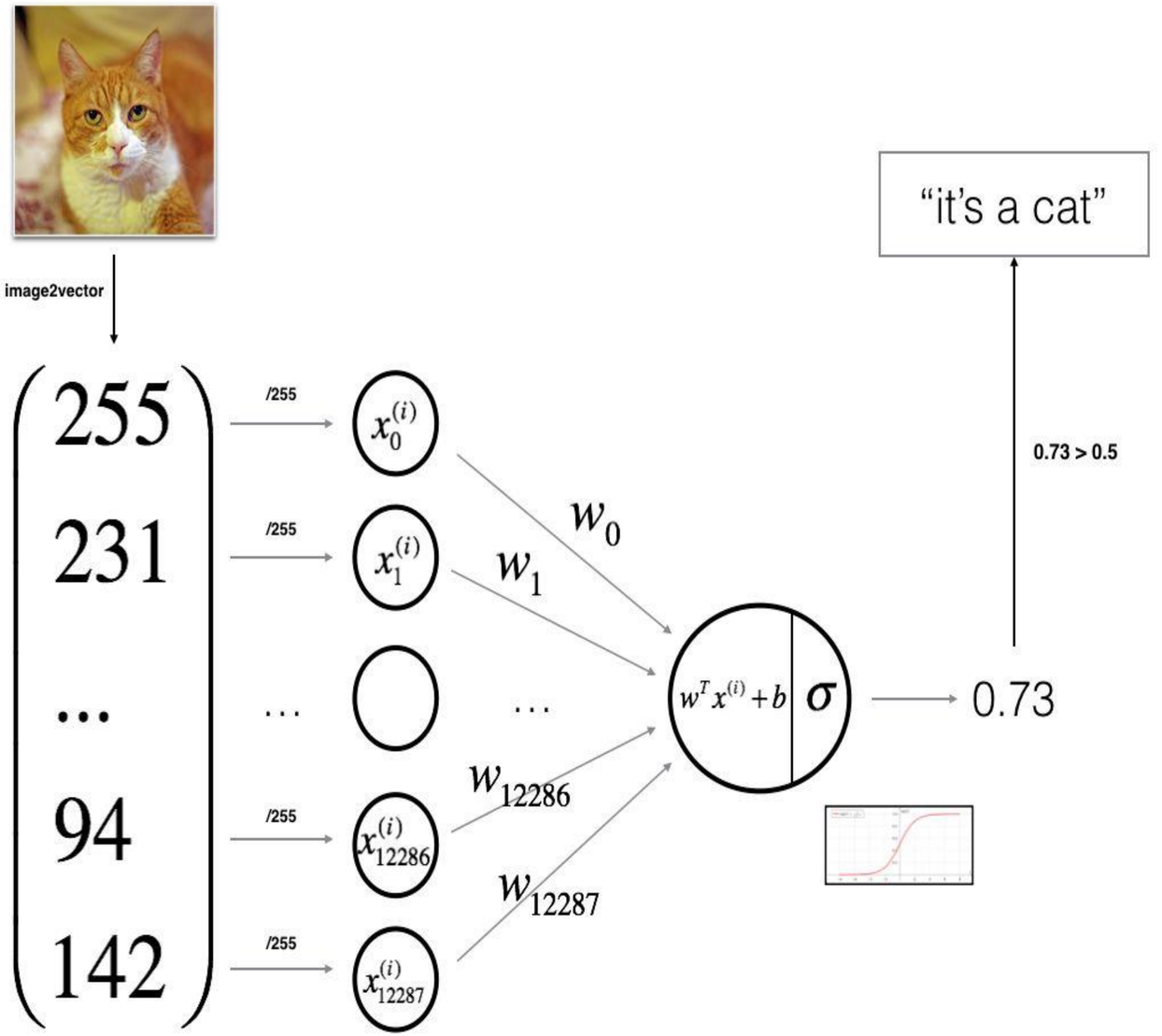
	Coefficient	Std. Error	Z-statistic	P-value
Intercept	-10.8690	0.4923	-22.08	< 0.0001
balance	0.0057	0.0002	24.74	< 0.0001
income	0.0030	0.0082	0.37	0.7115
student[Yes]	-0.6468	0.2362	-2.74	0.0062





逻辑回归是神经网络

- 逻辑回归是最简单的神经网络
- 只有输入层和输出层，没有隐含层
- 蕴含了前向传播和反向传播的思想



GDA & Logistic Regression

- 给定$\phi, \boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \boldsymbol{\Sigma}$，神奇的事情发生了：

$$\frac{P(y=1|\boldsymbol{x};\phi,\boldsymbol{\mu_0},\boldsymbol{\mu_1},\boldsymbol{\Sigma})}{1-P(y=1|\boldsymbol{x};\phi,\boldsymbol{\mu_0},\boldsymbol{\mu_1},\boldsymbol{\Sigma})}$$

$$= (\boldsymbol{\mu_1}-\boldsymbol{\mu_0})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu_1}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_1} + \frac{1}{2}\boldsymbol{\mu_0}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_0} + \log\left(\frac{\phi}{1-\phi}\right)$$

$$= \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{\alpha}$$

- GDA和Logistic Regression使用的是同样的模型！
- 因为参数估计方法的不同，它们会得到不同的决策边界
- GDA假设可以得到Logistic假设，但反之却不成立：说明GDA假设更强

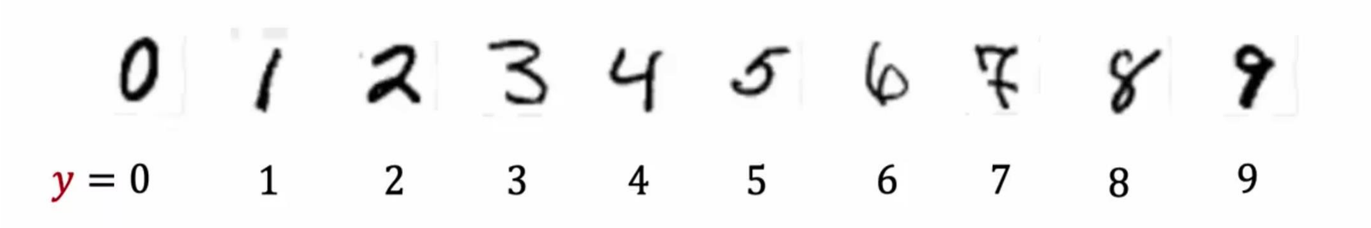
GDA & Logistic Regression

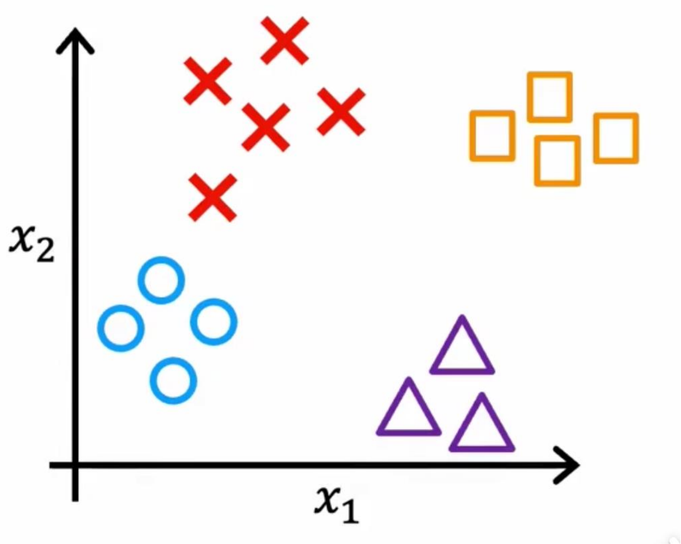
- 当GDA的强假设成立的时候，GDA需要更少的数据来进行有效的学习，此时表现要优于Logistic回归
- 但当模型是Poisson但却指定为Gaussian时，GDA的表现会很差
- 如果不知道分布是什么，Logistic回归的弱假设要更加稳健，不需要担心模型错判的问题。这也是在实际中判别模型要比生成模型用的更多的主要原因
- 数据量很大时，倾向于使用逻辑回归
- GDA有着更高的计算效率

SOFTMAX回归

多分类问题

- 假设Y的取值范围为$1,\dots,K$（或$0,\dots,K-1$）





- Softmax回归：得到一个K维向量 $\begin{pmatrix} P(others|\boldsymbol{x}) \\ P(cats|\boldsymbol{x}) \\ P(dogs|\boldsymbol{x}) \\ P(chicks|\boldsymbol{x}) \end{pmatrix}$，告诉我们属于每一类的概率。
- 向量中每一个元素大于0小于1，所有元素相加等于1。

SOFTMAX回归

- 两分类逻辑回归：

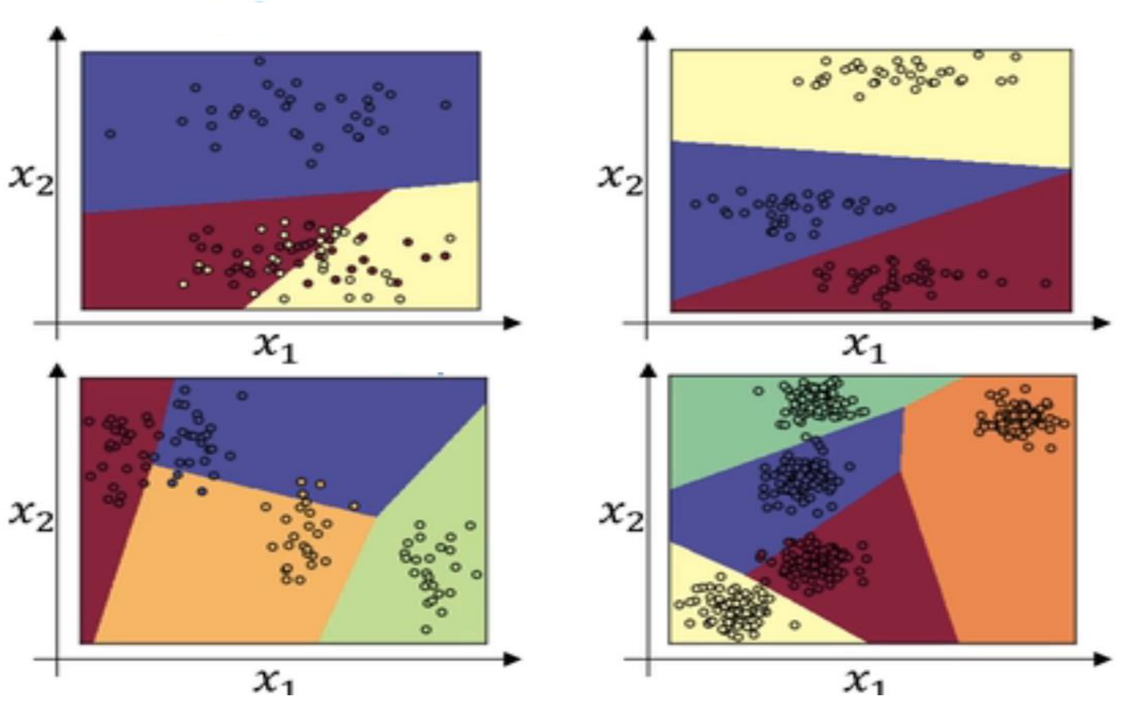
$$z = \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X} \rightarrow a_1 = g(z) = \frac{e^z}{1+e^z} = P(Y=1|\boldsymbol{X}); \quad a_2 = 1 - a_1 = \frac{1}{1+e^z} = P(Y=0|\boldsymbol{X})$$

- 多分类Softmax回归（以3类为例）：
  - $z_1 = \boldsymbol{\beta_1^T X};\ z_2 = \boldsymbol{\beta_2^T X}; \quad z_3 = \boldsymbol{\beta_3^T X};\ \boldsymbol{\beta_k} = (\beta_{k0}, \dots, \beta_{kd})^T, k = 1, \dots, 3$
  - $a_1 = \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}} = P(Y=1|\boldsymbol{X});\ a_2 = \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}} = P(Y=2|\boldsymbol{X});\ a_3 = 1 - a_1 - a_2 = \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}} = P(Y=3|\boldsymbol{X})$
  - $P(Y=k|\boldsymbol{X}) = \frac{e^{\beta_k^T X}}{\sum_{l=1}^{K} e^{\beta_l^T X}}$

- $\boldsymbol{X} \rightarrow \begin{pmatrix} z_1 = \boldsymbol{\beta_1^T X} \\ z_2 = \boldsymbol{\beta_2^T X} \\ z_3 = \boldsymbol{\beta_3^T X} \end{pmatrix} \rightarrow \begin{pmatrix} 5 \\ 2 \\ -1 \end{pmatrix} \rightarrow \begin{pmatrix} e^5 \\ e^2 \\ e^{-1} \end{pmatrix} = \begin{pmatrix} 148.4 \\ 7.4 \\ 0.4 \end{pmatrix} \rightarrow \begin{pmatrix} 148.4/156.2 \\ 7.4/156.2 \\ 0.4/156.2 \end{pmatrix} = \begin{pmatrix} 0.95 \\ 0.047 \\ 0.003 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

- $\log\left(\frac{P(Y=k|\boldsymbol{X})}{P(Y=k'|\boldsymbol{X})}\right) = (\beta_{k0} - \beta_{k'0}) + (\beta_{k1} - \beta_{k'1})X_1 + \cdots + (\beta_{kd} - \beta_{k'd})X_d$

SOFTMAX示例



- 任意两类之间是线性的决策边界

训练SOFTMAX回归

- One-hot encoding：将$y = 1, \ldots, K$转化为只有一个值为1其余值为0的K维向量
- 单个样本的例子：$y = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \hat{y} = \begin{pmatrix} 0.3 \\ 0.2 \\ 0.1 \\ 0.4 \end{pmatrix}$，只为真实的分类分配了0.2的概率值
- 单个样本的损失函数：$L(y, \hat{y}) = -\sum_{l=1}^{K} y_l \log \hat{y}_l$
- 训练数据的损失函数：$J(\boldsymbol{\beta_1}, \ldots, \boldsymbol{\beta_K}) = \frac{1}{n} \sum_{i=1}^{n} L\left(y^{(i)}, \hat{y}^{(i)}\right)$

广义线性回归

广义线性回归

- 回忆：$\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_d)^T$，　$\boldsymbol{X} = (X_0, X_1, \dots, X_d)^T$, $X_0 = 1$
- 线性回归：X服从正态分布

$$\mathrm{E}[Y|\boldsymbol{X};\boldsymbol{\beta}] = \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X}$$

- 逻辑回归：X服从二项分布

$$\mathrm{E}[Y|\boldsymbol{X};\boldsymbol{\beta}] = P(Y = 1|\boldsymbol{X};\boldsymbol{\beta}) = \frac{1}{1 + e^{-\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X}}}$$

- 广义线性回归

$$\eta(\mathrm{E}[Y|\boldsymbol{X};\boldsymbol{\beta}]) = \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X}$$

其中η为联系函数。线性回归中$\eta(\mu) = \mu$，逻辑回归中$\eta(\mu) = \log(\mu/(1-\mu))$。

广义线性回归

- 泊松回归：X服从泊松分布

$$\eta(\mathrm{E}[Y|\boldsymbol{X};\boldsymbol{\beta}]) = \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X}$$

  其中$\eta(\mu) = \log(\mu)$

- 正态、二项、泊松分布都属于指数族
- 对其他指数族，如Gamma，负二项分布，同样可以定义广义线性回归

本章总结

- 在本章中，我们学习到了：
  - 高斯判别分析
  - 二次判别分析
  - 朴素贝叶斯
  - 混淆矩阵、ROC、AUC
  - 逻辑回归
  - SOFTMAX 回归
  - 广义线性回归