机器学习基础

----万物充满智慧，一切皆可学习

知山知水 树木树人

机器学习基础（二）

- 生成模型和判别模型
- 概率生成模型
- 性能对比
- 其他生成模型
- 补充知识

机器学习基础（二）

- 生成模型和判别模型
- 概率生成模型
- 性能对比
- 其他生成模型
- 补充知识

机器学习中的模型分类

机器学习（统计学习）当中，按照模型的产生方式，可以分为两大类，分别是：

1）判别模型（ Discriminative Model ）

目前，我们所见到的模型当中，判别模型居多，回归（感知机）、 logistic回归、SVM、 KNN、决策树、前馈神经网络、 DNN中的CNN等。

2）生成模型（ Generative Model ）

以概率、随机过程和贝叶斯理论基础，主要有概率生成模型（ logistic回归）、朴素贝叶斯分类、贝叶斯网络、高斯混合模型（ GMM ）隐马尔科夫模型（ HMM ）、GAN等。

相关概念

判别模型：由数据直接学习函数Y=f(X)或者条件概率分布P(Y|X)作为预测的模型，即判别模型。基本思想是有限样本条件下建立判别函数（模型），不考虑样本的概率特性特性，直接产生预测模型。

生成模型：由数据学习联合概率密度分布P(X,Y)，然后求出条件概率分布P(Y|X)作为预测的模型，即生成模型：P(Y|X)= P(X,Y)/ P(X)。基本思想是首先建立样本的联合概率概率密度模型P(X,Y)，然后再得到后验概率P(Y|X)，从而产生预测模型，然后进行预测。

一个典型示例

以一个分类模型为例---来自维基百科

我们有一组数据

y中0代表‘NO’，1代表‘YES’

Input	Output
x	y
0	No
0	No
1	No
1	Yes

Input	Output
x	y
0	0
0	0
1	0
1	1

当一种模型同时有两种形式的时候，生成模型可以导出判别模型，而判别模型不能反推出生成模型。因此，从随机数学角度来看生成模型更“原始”，利于衍生出更多的模型和算法。

判别模型只关注条件分布

$P(y|x)$ 其中$(\sum_{y} P(y|x) = 1)$

		y		
		0	1	sum
x	0	1	0	1
	1	1/2	1/2	1

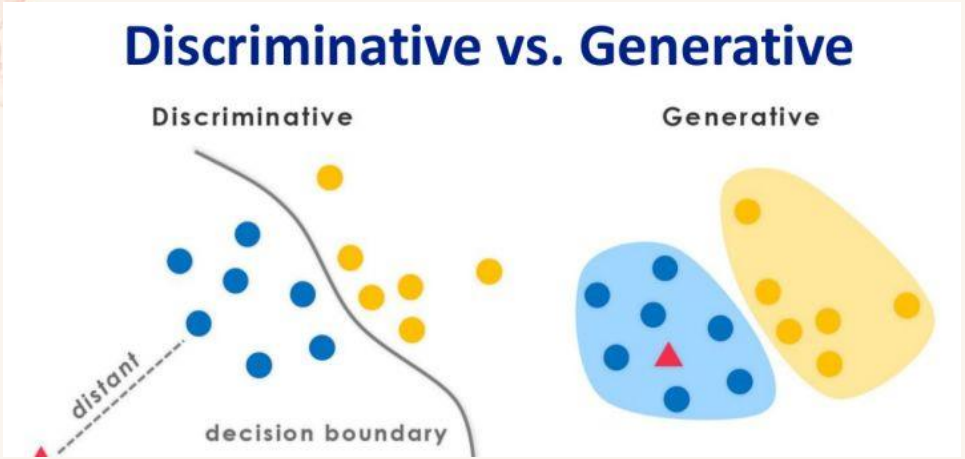
生成模型从关注联合分布*开始！！！*

$P(x,y)$ 其中$(\sum_{x}\sum_{y} P(x,y) = 1)$

		y		
		0	1	sum
x	0	1/2	0	1/2
	1	1/4	1/4	1/2
	sum	3/4	1/4	1

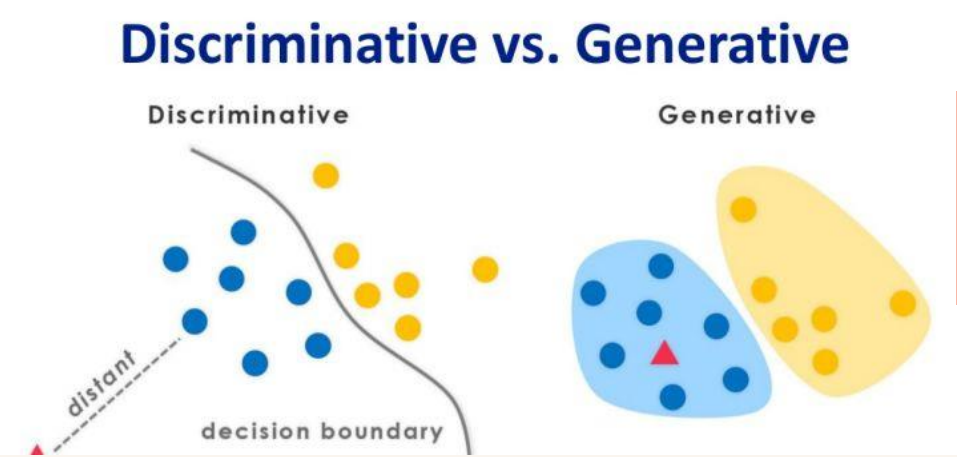
模型对比

对比	判别式模型	生成式模型
特点	寻找不同类别之间的最优分类面，反映异类数据之间的差异	以概率的角度表示数据的分布情况，能够反映同类数据本身的相似度
区别（输入x,输出y）	估计的是条件概率分布：P(y|x)	估计的是联合概率分布 P(x,y)
联系	由判别式模型不能得到生成式模型	由生成式模型可以得到判别式模型（贝叶斯公式）
优势	(1)能清晰地分辨出多类或某一类与其他类之间的差异特征；（2）适用于较多类别的识别；（3）模型更简单	（1）研究单类问题比判别式模型更灵活；（2）模型可以通过增强学习得到；（3）能用于数据不完整的情况。
缺点	不能反映训练数据本身的特性；	学习和计算过程比较复杂
预测性能	较好（因为利用了训练数据的类别标识信息）	比判别模型要差？？？？
常见模型举例	KNN，SVM，决策树，线性回归，逻辑回归，boosting，线性判别分析（LDA），感知机，传统神经网络	概率生成模型，朴素贝叶斯，隐马尔科夫模型，高斯混合模型，限制玻尔兹曼机



判别方法直接学习的是函数Y=f(X)或者条件概率分布P(Y|X)。不能反映训练数据本身的特性。但它寻找不同类别之间的最优分类面，反映的是异类数据之间的差异。直接面对预测，往往学习的准确率更高。由于直接学习P(Y|X)或P(X)，可以对数据进行各种程度上的抽象、定义特征并使用特征，因此可以简化学习问题。

对于判别式模型来说求得P(Y|X)，对未知观测X，根据P(Y|X)可以求得标记Y，即可以直接判别出来，如上图的左边所示，实际是就是直接得到了判别边界，所以传统的、耳熟能详的机器学习算法和模型都是判别式模型这些模型的特点都是输入属性X可以直接得到Y，对于二分类任务来说，实际得到一个score（概率），当score大于某个阈值（threshold）时则为正类，否则为负类。



首先建立样本的联合概率概率密度模型P(X,Y)，然后再得到后验概率P(Y|X)，再利用它进行预测。因此，生成模型可以导出判别模型，反之不成立。！！！

生成方法学习联合概率密度分布P(X,Y)，所以就可以从概率的角度表示数据的分布情况，能够反映同类数据本身的相似度。但它不关心到底划分各类的那个分类边界在哪。生成方法可以还原出联合概率分布P(Y|X)，而判别方法不能。生成方法的学习收敛速度更快，即当样本容量增加的时候，学到的模型可以更快的收敛于真实模型，当存在隐变量时，仍可以用生成方法学习。此时判别方法就不能用。

生成式模型求得P(X, Y)，对于未知观测X，你要求出X与不同类之间的联合概率分布，然后大的类获胜，如上图右边所示，并没有什么边界存在，对于未知观测（红三角），分别求两个联合概率分布（有两个类），比较一下，取概率大的那个分布，也就是相应的类别或值。

如何区分两种建模方式（approach）---通俗来讲

生成算法尝试去找到底这个数据是怎么生成的（产生的），然后再对一个信号进行分类。基于你的生成假设，那么那个类别最有可能产生这个信号，这个信号就属于那个类别。判别模型不关心数据是怎么生成的，它只关心信号之间的差别，然后用差别来简单对给定的一个信号进行分类。

假如你的任务是识别一个语音属于哪种语言。例如对面一个人走过来，和你说了一句话，你需要识别出她说的到底是汉语、英语还是法语等。那么你可以有两种方法达到这个目的：

1、学习每一种语言，你花了大量精力把汉语、英语和法语等都学会了，我指的学会是你知道什么样的语音对应什么样的语言。然后再有人过来对你哄，你就可以知道他说的是什么语音。

2、不去学习每一种语言，你只学习这些语言之间的差别，然后再进行区分。意思是指我学会了汉语和英语等语言的发音是有差别的，我学会这种差别就好了。

第一种方法就是生成式，第二种方法是判别式。

两种方式的对比

1）生成给出的是**联合分布**，不仅能够由联合分布计算条件分布（反之则不行），还可以给出其他信息，比如可以使用**P(x)=求和(P(x|ci))** 来计算边缘**分布P(x)**。如果一个输入样本的边缘分布**P(x)**很小的话，那么可以认为学习出的这个模型可能不太适合对这个样本进行分类，分类效果可能会不好，这也是所谓的outlier detection。生成方法可以原出联合概率分布分布**P(X,Y)**，而判别方法不能。
2）生成模型收敛速度比较快，即当样本数量较多时，生成模型能更快地收敛于真实模型。
3）生成模型能够应付存在**隐变量**的情况，比如**混合高斯模型**就是含有隐变量的生成方法。此时判别方法就不能用。

生成模型

1）天下没有免费午餐，联合分布是能提供更多的信息，但也需要更多的样本和更多计算，尤其是为了更准确估计类别条件分布，需要增加样本的数目，而且类别条件概率的许多信息是我们做分类用不到，因而如果我们只需要做分类任务，就浪费了计算资源。
2）另外，实践中多数情况下判别模型效果更好。

1）不具备更加细致的附加能力，只能进行预测，虽然效果更好，但是容易过拟合。

2）没有额外的诊断功能

判别模型

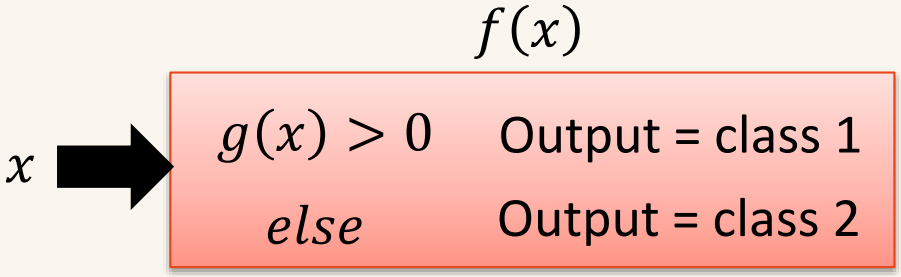
1）由于直接学习P(Y|X)或f(X)，可以对数据进行各种程度上的抽象、定义特征并使用特征，因此可以简化学习问题。与生成模型缺点对应，首先是节省计算资源，另外，需要的样本数量也少于生成模型。
2）直接面对预测，准确率往往较生成模型高。
3）判别方法直接学习的是决策函数Y=f(X)或者条件概率分布P(Y|X)，不能反映训练数据本身的特性。但它寻找不同类别之间的最优分类面，反映的是异类数据之间的差异。所以允许我们对输入进行抽象（比如降维、构造等），从而能够简化学习问题。

机器学习基础（二）

- 生成模型和判别模型
- 概率生成模型
- 性能对比
- 其他生成模型
- 补充知识

概率生成模型(Probabilistic Generative Model)

- Function (Model):



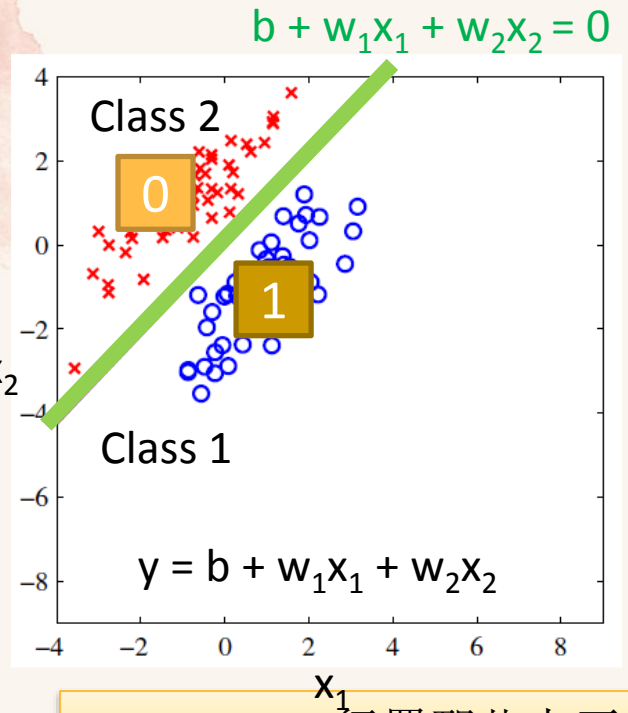
- 损失函数:

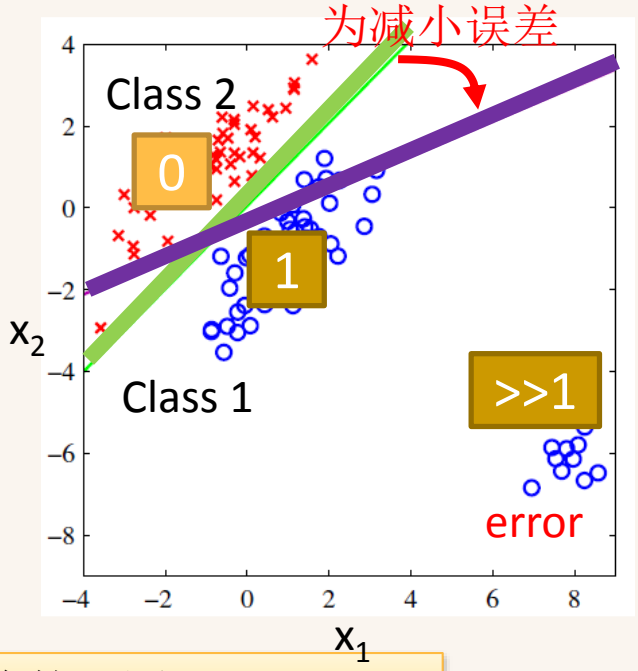
$$L(f) = \sum_n \delta(f(x^n) \neq \hat{y}^n)$$

The number of times f get incorrect results on training data.

- 获得最优模型:
  - Example: 感知器(Perceptron), SVM

像回归那样进行分类



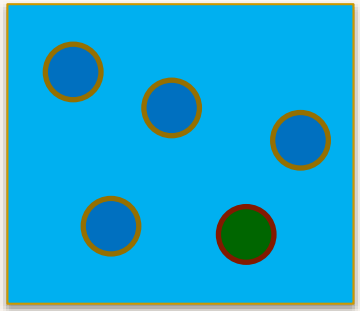


惩罚那些太正确的观测

两个盒子

Box 1

$P(B_1) = 2/3$

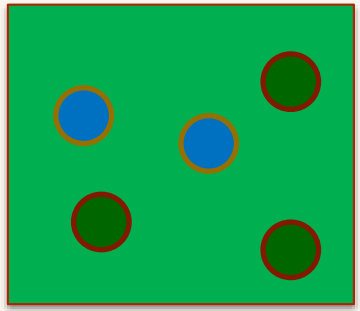


$P(\text{Blue}|B_1) = 4/5$

$P(\text{Green}|B_1) = 1/5$

Box 2

$P(B_2) = 1/3$



$P(\text{Blue}|B_2) = 2/5$

$P(\text{Green}|B_2) = 3/5$

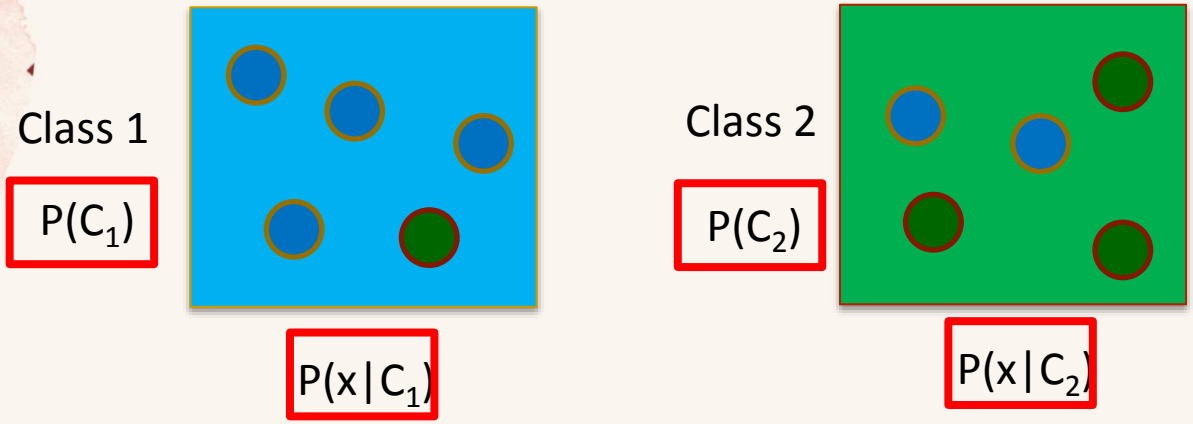
来自于其中一个盒子

那个呢?

$$P(B_1 \mid \text{Blue}) = \frac{P(\text{Blue}|B_1)P(B_1)}{P(\text{Blue}|B_1)P(B_1) + P(\text{Blue}|B_2)P(B_2)}$$

两个类别

从训练数据中估计概率。

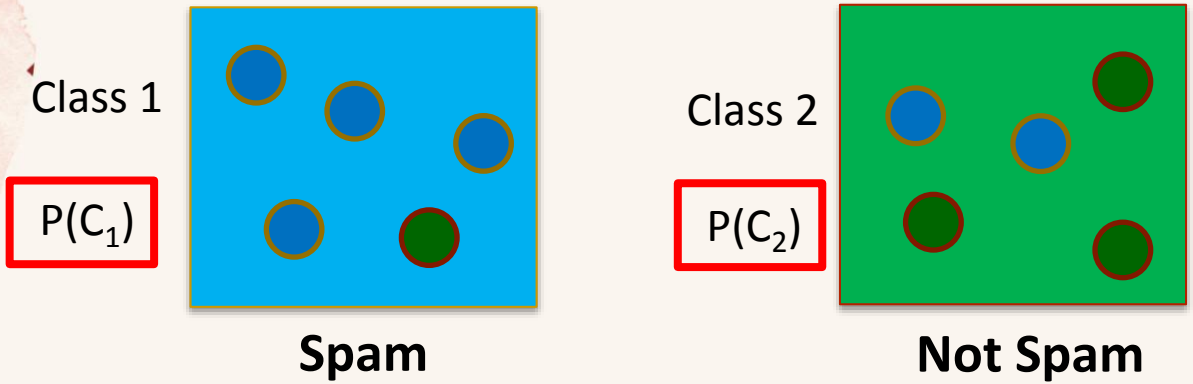


给一个x, 他属于那个类别呢?

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

生成模型(Generative Model)　$P(x) = P(x|C_1)P(C_1) + P(x|C_2)P(C_2)$

先验信息(Prior)



Spam和Not Spam 的数据中一部分作为训练集，
其他为测试集。

Training: 79 Spam, 61 Not Spam

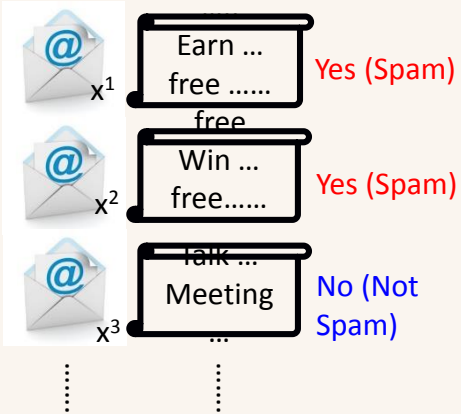
$P(C_1) = 79 / (79 + 61) = 0.56$

$P(C_2) = 61 / (79 + 61) = 0.44$

来自类别的概率

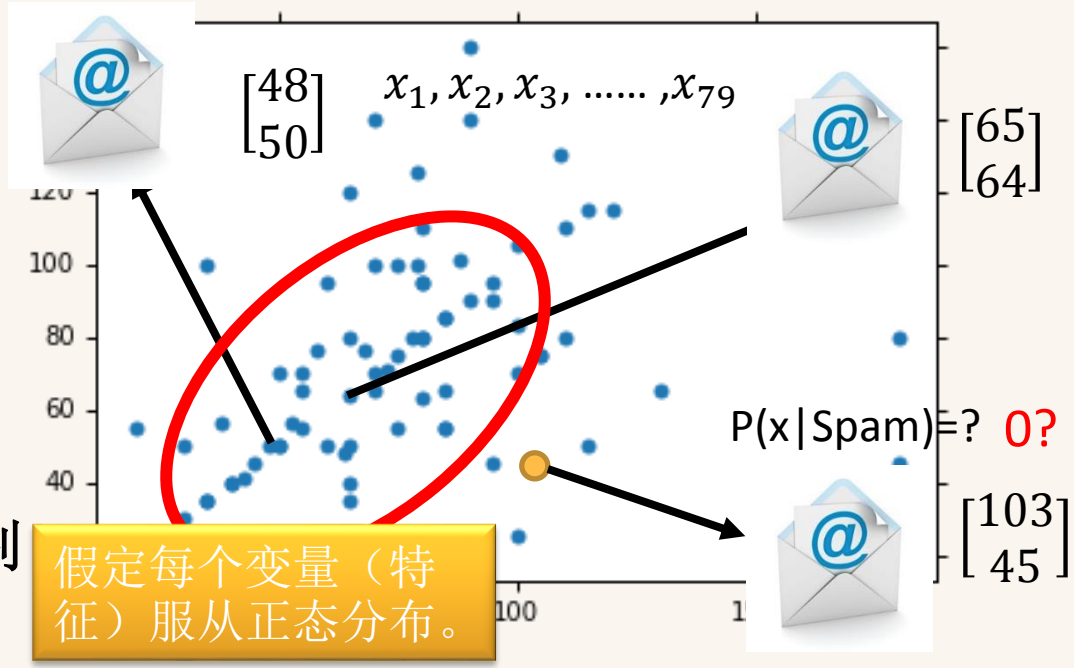
$P(x|C_1) = ?$　　P( |Spam) = ?

邮件的各个特征用向量表示 ➡ 特征（变量）
feature



类别的概率- Feature

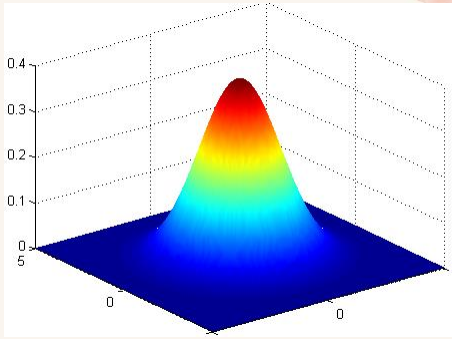
- 考虑两个特征

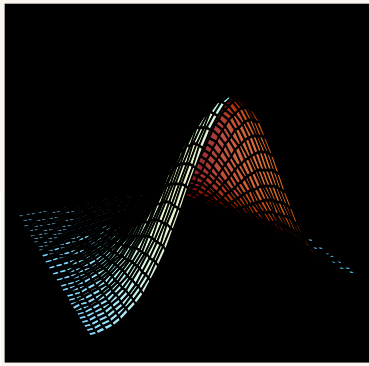


正态分布

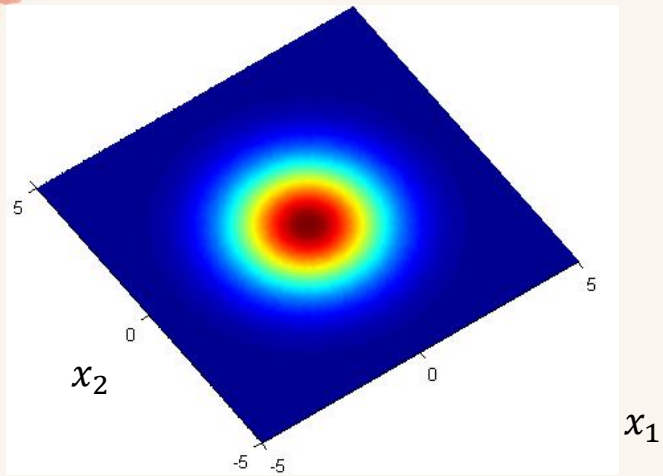
$$f_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\}$$

正态分布的形态由均值 $\boldsymbol{\mu}$ 和协方差矩阵 $\boldsymbol{\Sigma}$ 决定。

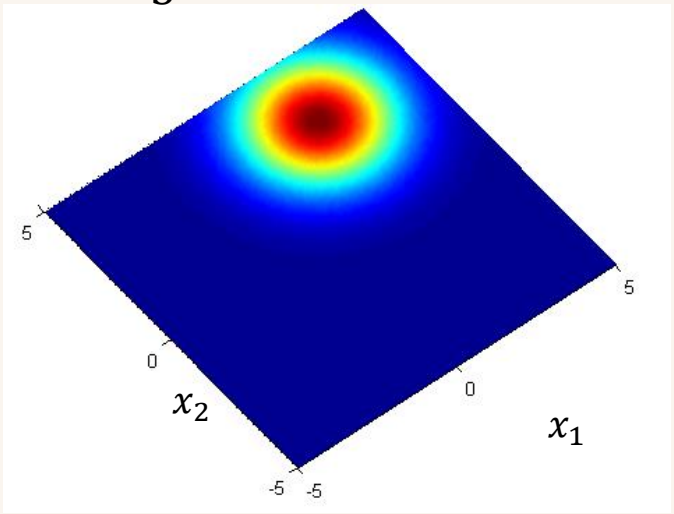




$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \sum = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \quad \sum = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

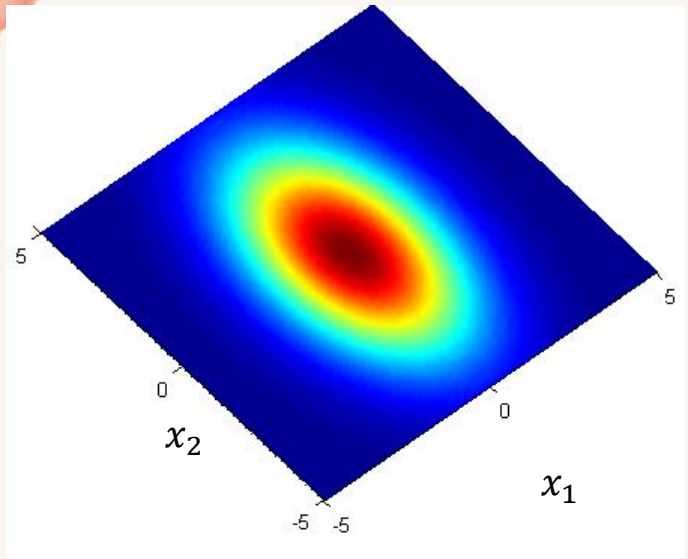


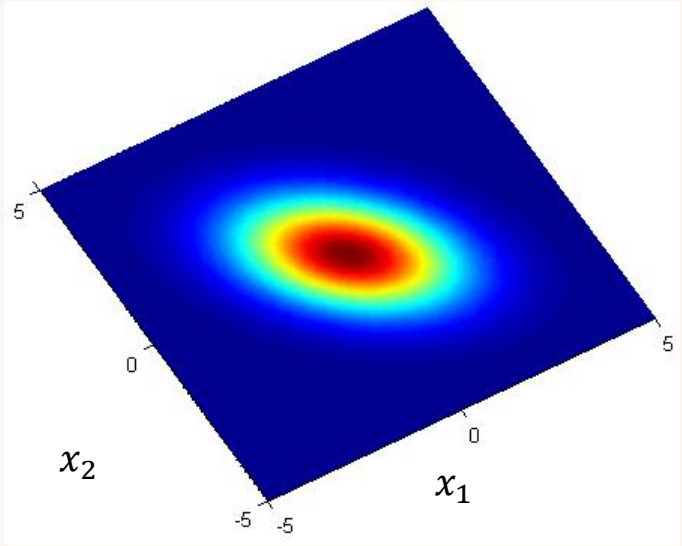
正态分布

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

正态分布的形态由均值 $\boldsymbol{\mu}$ 和协方差矩阵$\boldsymbol{\Sigma}$决定。

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \sum = \begin{bmatrix} 2 & 0 \\ 0 & 6 \end{bmatrix} \qquad \mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \sum = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

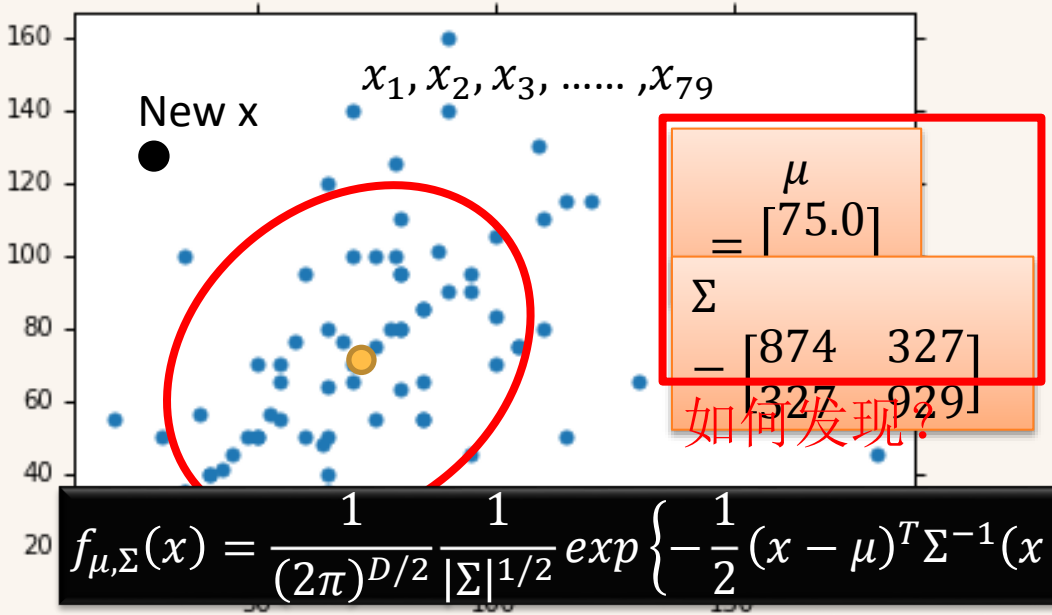




类别中计算概率

假定数据来自正态分布。

在新数据来之前先预测到它。 ➡ 新观测的概率



Spam

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

极大似然Maximum Likelihood

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

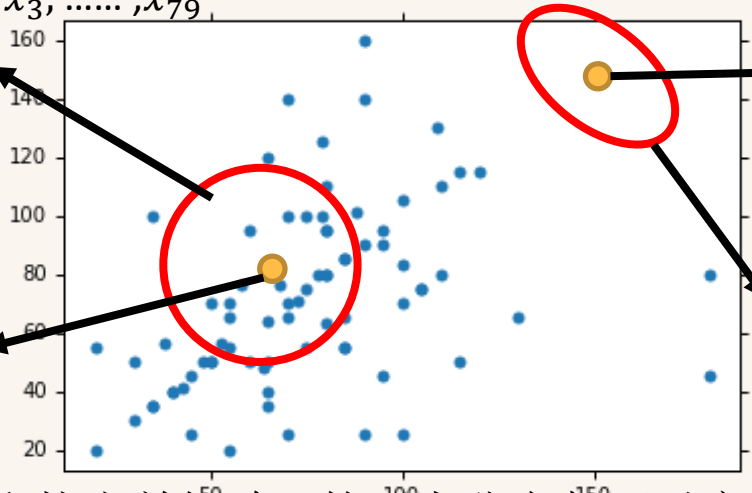
$x_1, x_2, x_3, \ldots\ldots, x_{79}$

$$\sum = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

$$\mu = \begin{bmatrix} 150 \\ 140 \end{bmatrix}$$

$$\mu = \begin{bmatrix} 75 \\ 75 \end{bmatrix}$$

$$\sum = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$



任何均值 μ 和协方差矩阵 Σ的正态分布都可以产生这些观测。

利用样本$x_1, x_2, x_3, \ldots\ldots, x_{79}$，采用极大似然法来对正态分布的参数进行估计，$\mu$和$\Sigma$

$$L(\mu,\Sigma) = f_{\mu,\Sigma}(x^1) f_{\mu,\Sigma}(x^2) f_{\mu,\Sigma}(x^3) \ldots\ldots f_{\mu,\Sigma}(x^{79})$$

极大似然(Maximum Likelihood)

我们有spam类的观测79个： $x_1, x_2, x_3, \ldots\ldots, x_{79}$

假定$x_1, x_2, x_3, \ldots\ldots, x_{79}$来自正态分布$(\mu^*, \Sigma^*)$，采用极大似然估计

$$L(\mu, \Sigma) = f_{\mu,\Sigma}(x_1) f_{\mu,\Sigma}(x_2) f_{\mu,\Sigma}(x_3) \ldots\ldots f_{\mu,\Sigma}(x_{79})$$

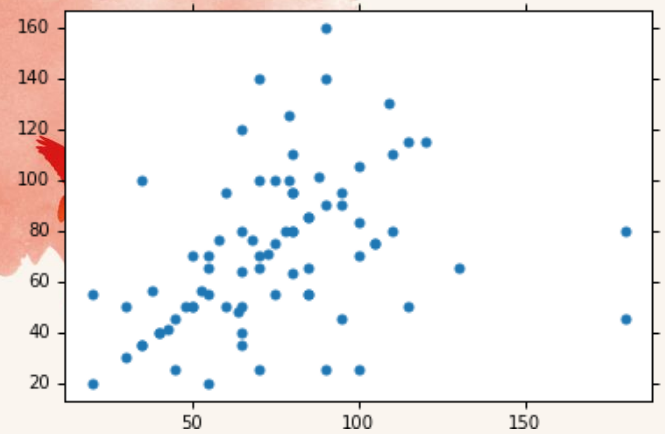
$$f_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\}$$

$$\mu^*, \Sigma^* = arg \max_{\mu,\Sigma} L(\mu, \Sigma)$$

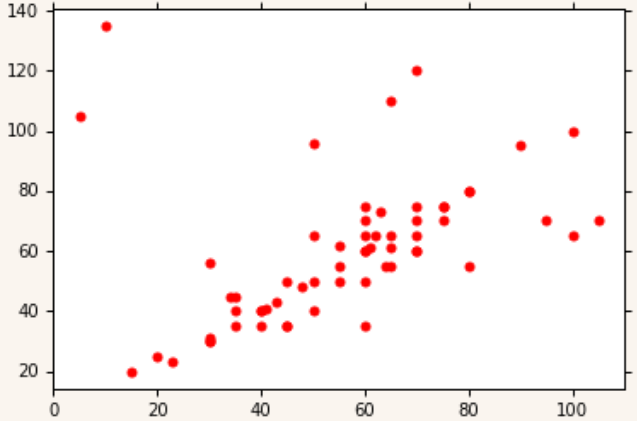
$$\mu^* = \frac{1}{79}\sum_{n=1}^{79} x_n \qquad \Sigma^* = \frac{1}{79}\sum_{n=1}^{79} (x_n - \mu^*)(x_n - \mu^*)^T$$

极大似然

Class 1: Spam



Class 2: Not Spam



$$\mu_1 = \begin{bmatrix} 75.0 \\ 71.3 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 874 & 327 \\ 327 & 929 \end{bmatrix} \qquad \mu_2 = \begin{bmatrix} 55.6 \\ 59.8 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 847 & 422 \\ 422 & 685 \end{bmatrix}$$

通过计算概率做分类

$$f_{\mu_1,\Sigma_1}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma_1|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu_1)^T(\Sigma_1)^{-1}(x-\mu_1)\right\}$$

$$\mu_1 = \begin{bmatrix}75.0\\71.3\end{bmatrix} \quad \Sigma_1 = \begin{bmatrix}874 & 327\\327 & 929\end{bmatrix}$$

P(C1)
= 79 / (79 + 61)
=0.56

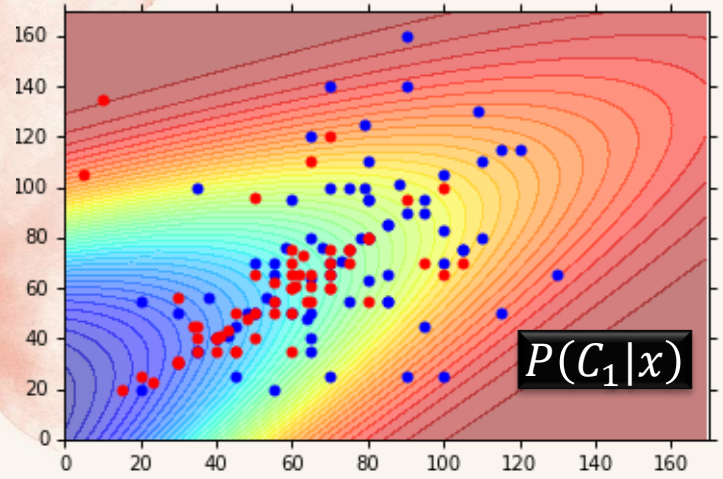
$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

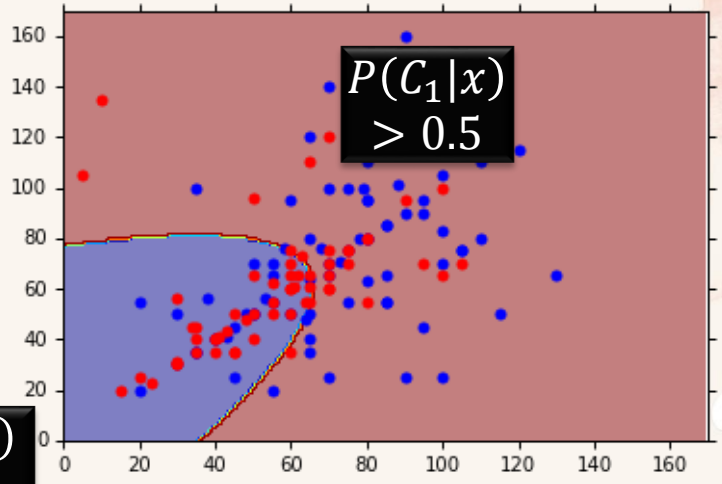
$$f_{\mu_2,\Sigma_2}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma_2|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu_2)^T(\Sigma_2)^{-1}(x-\mu_2)\right\}$$

P(C2)
= 61 / (79 + 61)
=0.44

$$\mu_2 = \begin{bmatrix}55.6\\59.8\end{bmatrix} \quad \Sigma_2 = \begin{bmatrix}847 & 422\\422 & 685\end{bmatrix}$$

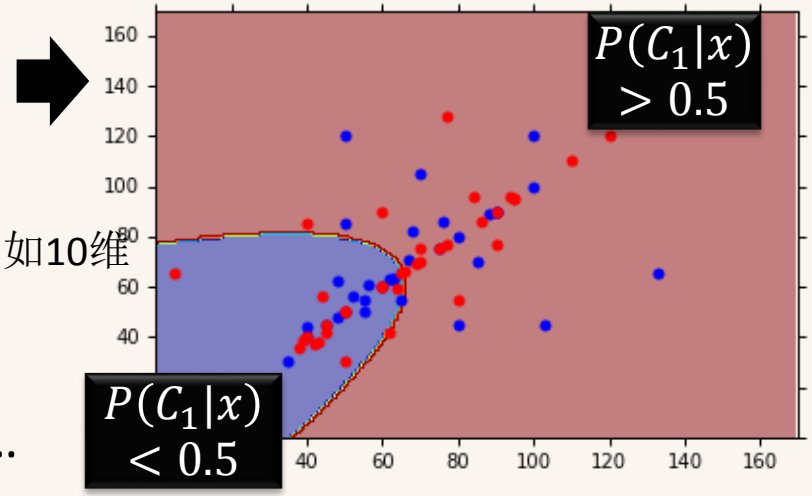
If $P(C_1|x) > 0.5$ ➡ x 属于class 1 (Spam)





蓝点: C_1 (Spam), 红点: C_2 (Not Spam)

Testing data: 47% accuracy

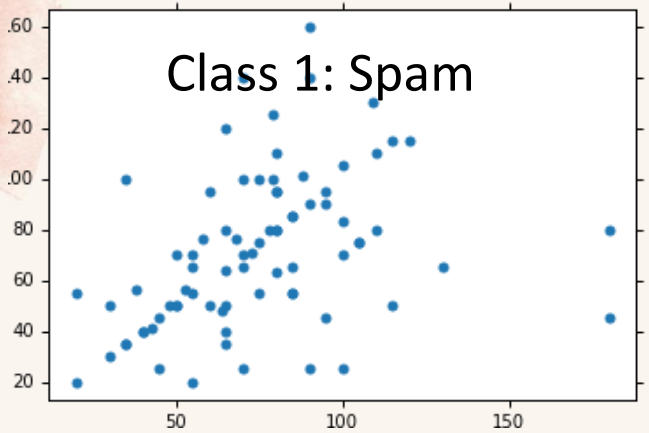


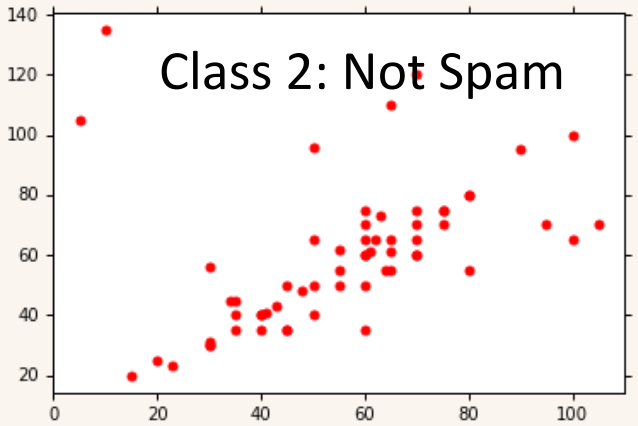
μ_1, μ_2: 向量，几个输入变量就是几维，比如10维

Σ_1, Σ_2: 10 x 10 矩阵

64% accuracy …

模型修正(Modifying Model)





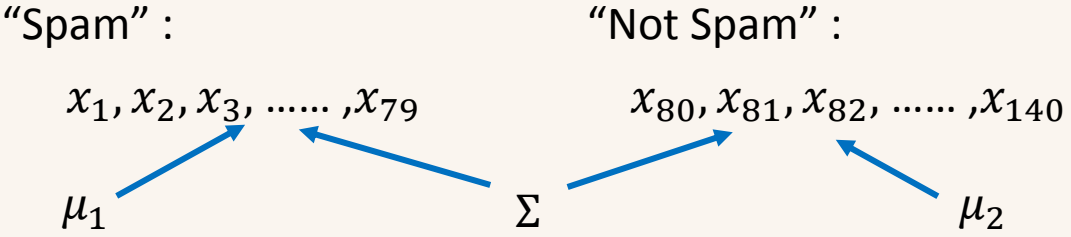
$$\mu_1 = \begin{bmatrix} 75.0 \\ 71.3 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 874 & 327 \\ 327 & 929 \end{bmatrix} \qquad \mu_2 = \begin{bmatrix} 55.6 \\ 59.8 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 847 & 422 \\ 422 & 685 \end{bmatrix}$$

相等的Σ

减少参数个数

模型修正(Modifying Model)

- 极大似然

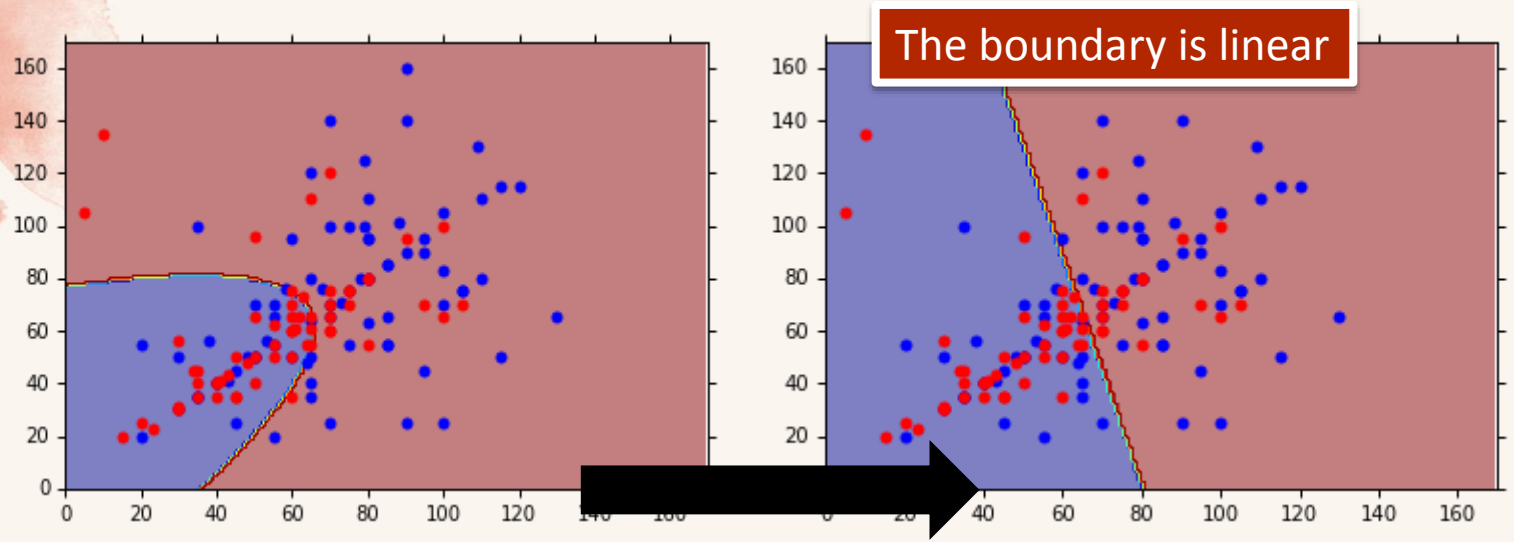


最小化似然函数$L(\mu_1,\mu_2,\Sigma)$ 获得μ_1,μ_2,Σ

$$L(\mu_1,\mu_2,\Sigma) = f_{\mu_1,\Sigma}(x_1)f_{\mu_1,\Sigma}(x_2)\cdots f_{\mu_1,\Sigma}(x_{79}) \times f_{\mu_2,\Sigma}(x_{80})f_{\mu_2,\Sigma}(x_{81})\cdots f_{\mu_2,\Sigma}(x_{140})$$

如果 μ_1和μ_2相同 $\Sigma = \frac{79}{140}\Sigma_1 + \frac{61}{140}\Sigma_2$

模型修正(Modifying Model)



相同的协方差矩阵

54% accuracy → 73% accuracy

三部曲

- 函数集(模型):

x ➡

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

If $P(C_1|x) > 0.5$, output: class 1

否则, output: class 2

- 损失函数(Goodness of a function) :
  - 极大似然估计，μ，Σ 等
- 获得最优模型: 水到渠成

分布的选择

- 根据实际情况和你的经验，反复尝试，获得适合的分布

$$P(x|C_1) = P(x_1|C_1)\ P(x_2|C_1) \quad \cdots\cdots \quad P(x_k|C_1) \quad \cdots\cdots$$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \\ \vdots \\ x_K \end{bmatrix}$$

一维正态分布？

如果是离散数据，根据数据的具体情况，选择分布。

如果你的每个特征（变量）之间完全独立，那么模型就变成朴素贝叶斯分类(Naive Bayes Classifier)。

朴素贝叶斯分类(Naive Bayes Classifier)

sklearn.naive_bayes.**GaussianNB**(*, priors=*None*, var_smoothing=*1e-09*)

```
>>> import numpy as np
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> Y = np.array([1, 1, 1, 2, 2, 2])
>>> from sklearn.naive_bayes import GaussianNB
>>> clf = GaussianNB()
>>> clf.fit(X, Y)
GaussianNB()
>>> print(clf.predict([[-0.8, -1]]))
[1]
>>> clf_pf = GaussianNB()
>>> clf_pf.partial_fit(X, Y, np.unique(Y))
GaussianNB()
>>> print(clf_pf.predict([[-0.8, -1]]))
[1]
```

sklearn.naive_bayes.**MultinomialNB**(*, alpha=*1.0*, fit_prior=*True*, class_prior=*None*)

```
>>> import numpy as np
>>> rng = np.random.RandomState(1)
>>> X = rng.randint(5, size=(6, 100))
>>> y = np.array([1, 2, 3, 4, 5, 6])
>>> from sklearn.naive_bayes import MultinomialNB
>>> clf = MultinomialNB()
>>> clf.fit(X, y)
MultinomialNB()
>>> print(clf.predict(X[2:3]))
[3]
```

sklearn.naive_bayes.**ComplementNB**(*, alpha=*1.0*, fit_prior=*True*, class_prior=*None*, norm=*False*)

```
>>> import numpy as np
>>> rng = np.random.RandomState(1)
>>> X = rng.randint(5, size=(6, 100))
>>> y = np.array([1, 2, 3, 4, 5, 6])
>>> from sklearn.naive_bayes import ComplementNB
>>> clf = ComplementNB()
>>> clf.fit(X, y)
ComplementNB()
>>> print(clf.predict(X[2:3]))
[3]
```

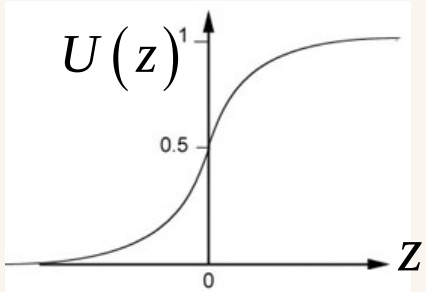
后验概率(Posterior Probability)

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

$$= \frac{1}{1 + \frac{P(x|C_2)P(C_2)}{P(x|C_1)P(C_1)}} = \frac{1}{1 + exp(-z)} = U(z)$$

Sigmoid function

$$z = ln\frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}$$



后验概率(Posterior Probability)

$$P(C_1|x) = U(z)$$ sigmoid $$z = ln\frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}$$

$$z = ln\frac{P(x|C_1)}{P(x|C_2)} + ln\frac{P(C_1)}{P(C_2)} \Rightarrow \frac{\frac{N_1}{N_1+N_2}}{\frac{N_2}{N_1+N_2}} = \frac{N_1}{N_2}$$

$$P(x|C_1) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma_1|^{1/2}}exp\left\{-\frac{1}{2}(x-\mu_1)^T(\Sigma_1)^{-1}(x-\mu_1)\right\}$$

$$P(x|C_2) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma_2|^{1/2}}exp\left\{-\frac{1}{2}(x-\mu_2)^T(\Sigma_2)^{-1}(x-\mu_2)\right\}$$

$$z = ln\frac{P(x|C_1)}{P(x|C_2)} + ln\frac{P(C_1)}{P(C_2)} = \frac{N_1}{N_2}$$

$$P(x|C_1) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma_1|^{1/2}}exp\left\{-\frac{1}{2}(x-\mu_1)^T(\Sigma_1)^{-1}(x-\mu_1)\right\}$$

$$P(x|C_2) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma_2|^{1/2}}exp\left\{-\frac{1}{2}(x-\mu_2)^T(\Sigma_2)^{-1}(x-\mu_2)\right\}$$

$$ln\frac{\frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma_1|^{1/2}}exp\left\{-\frac{1}{2}(x-\mu_1)^T(\Sigma_1)^{-1}(x-\mu_1)\right\}}{\frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma_2|^{1/2}}exp\left\{-\frac{1}{2}(x-\mu_2)^T(\Sigma_2)^{-1}(x-\mu_2)\right\}}$$

$$= ln\frac{|\Sigma_2|^{1/2}}{|\Sigma_1|^{1/2}}exp\left\{-\frac{1}{2}[(x-\mu_1)^T(\Sigma_1)^{-1}(x-\mu_1) - (x-\mu_2)^T(\Sigma_2)^{-1}(x-\mu_2)]\right\}$$

$$= ln\frac{|\Sigma_2|^{1/2}}{|\Sigma_1|^{1/2}} - \frac{1}{2}[(x-\mu_1)^T(\Sigma_1)^{-1}(x-\mu_1) - (x-\mu_2)^T(\Sigma_2)^{-1}(x-\mu_2)]$$

$$z = ln\frac{P(x|C_1)}{P(x|C_2)} + ln\frac{P(C_1)}{P(C_2)} = \frac{N_1}{N_2}$$

$$= ln\frac{|\Sigma_2|^{1/2}}{|\Sigma_1|^{1/2}} - \frac{1}{2}[(x-\mu_1)^T(\Sigma_1)^{-1}(x-\mu_1) - (x-\mu_2)^T(\Sigma_2)^{-1}(x-\mu_2)]$$

$$(x-\mu_1)^T(\Sigma_1)^{-1}(x-\mu_1)$$

$$= x^T(\Sigma_1)^{-1}x - x^T(\Sigma^1)^{-1}\mu_1 - (\mu_1)^T(\Sigma_1)^{-1}x + (\mu_1)^T(\Sigma_1)^{-1}\mu_1$$

$$= x^T(\Sigma_1)^{-1}x - 2(\mu_1)^T(\Sigma_1)^{-1}x + (\mu_1)^T(\Sigma_1)^{-1}\mu_1$$

$$(x-\mu_2)^T(\Sigma_2)^{-1}(x-\mu_2)$$

$$= x^T(\Sigma_2)^{-1}x - 2(\mu_2)^T(\Sigma_2)^{-1}x + (\mu_2)^T(\Sigma_2)^{-1}\mu^2$$

$$z = ln\frac{|\Sigma_2|^{1/2}}{|\Sigma_1|^{1/2}} - \frac{1}{2}x^T(\Sigma_1)^{-1}x + (\mu_1)^T(\Sigma_1)^{-1}x - \frac{1}{2}(\mu_1)^T(\Sigma_1)^{-1}\mu^1$$
$$+\frac{1}{2}x^T(\Sigma_2)^{-1}x - (\mu_2)^T(\Sigma_2)^{-1}x + \frac{1}{2}(\mu_2)^T(\Sigma_2)^{-1}\mu_2 + ln\frac{N_1}{N_2}$$

$$P(C_1|x) = U(z)$$

$$z = ln\frac{|\Sigma_2|^{1/2}}{|\Sigma_1|^{1/2}} - \frac{1}{2}x^T(\Sigma_1)^{-1}x + (\mu_1)^T(\Sigma_1)^{-1}x - \frac{1}{2}(\mu_1)^T(\Sigma_1)^{-1}\mu_1$$
$$+\frac{1}{2}x^T(\Sigma_2)^{-1}x - (\mu_2)^T(\Sigma_2)^{-1}x + \frac{1}{2}(\mu_2)^T(\Sigma_2)^{-1}\mu_2 + ln\frac{N_1}{N_2}$$

$$\Sigma_1 = \Sigma_2 = \Sigma$$

$$z = \underbrace{(\mu_1 - \mu_2)^T\Sigma^{-1}}_{\boldsymbol{w^T}}x - \underbrace{\frac{1}{2}(\mu_2)^T\Sigma^{-1}\mu_1 + \frac{1}{2}(\mu_2)^T\Sigma^{-1}\mu_2 + ln\frac{N_1}{N_2}}_{\mathrm{b}}$$

$$P(C_1|x) = U(w \cdot x + b)$$

我们可以直接获得参数值—判别模型

在生成模型中，我们估计 $N_1, N_2, \mu_1, \mu_2, \Sigma$

然后就可以获得**w**和b

机器学习基础（二）

- 生成模型和判别模型
- 概率生成模型
- 性能对比
- 其他生成模型
- 补充知识

Discriminative v.s. Generative

$$P(C_1|x) = \sigma(w \cdot x + b)$$

直接估计**w** 和 b

估计 μ_1, μ_2, Σ

$$w^T = (\mu_1 - \mu_2)^T \Sigma^{-1}$$

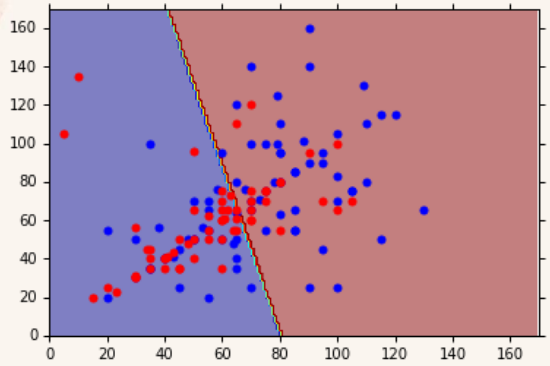
$$b = -\frac{1}{2}(\mu_1)^T (\Sigma_1)^{-1} \mu_1 + \frac{1}{2}(\mu_2)^T (\Sigma_2)^{-1} \mu_2 + ln\frac{N_1}{N_2}$$

同样的数据会获得相同的参数吗?

由于估计方式等技术问题，经常会出现同样的样本数据获得的最终模型不同。

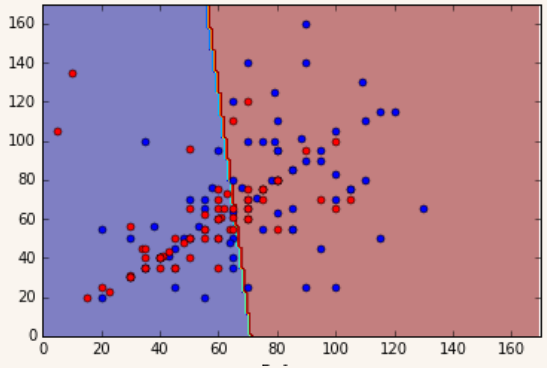
Generative v.s. Discriminative

Generative



73% accuracy

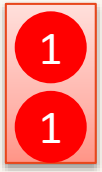
Discriminative



79% accuracy

Generative v.s. Discriminative

Training Data



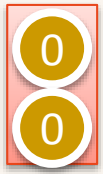
Class 1

X 4

Class 2

X 4

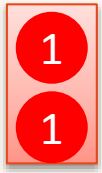
Class 2



X 4

Class 2

Testing Data



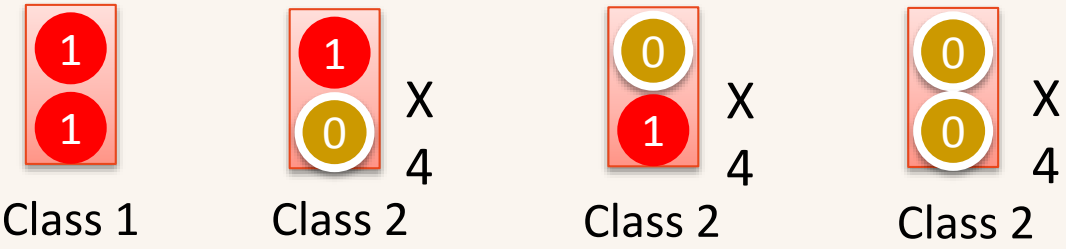
Class 1?
Class 2?

朴素贝叶斯怎样?

$$P(x|C_i) = P(x_1|C_i)P(x_2|C_i)$$

Generative v.s. Discriminative

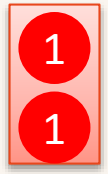
Training Data



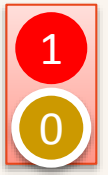
$$P(C_1) = \frac{1}{13} \qquad P(x_1 = 1|C_1) = 1 \qquad P(x_2 = 1|C_1) = 1$$

$$P(C_2) = \frac{12}{13} \qquad P(x_1 = 1|C_2) = \frac{1}{3} \qquad P(x_2 = 1|C_2) = \frac{1}{3}$$

Training Data

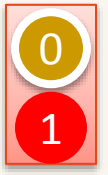


Class 1



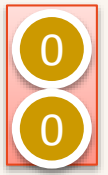
X 4

Class 2



X 4

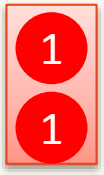
Class 2



X 4

Class 2

Testing Data



<0.5

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

$P(x|C_1)$: 1×1 — $P(C_1)$: $\frac{1}{13}$

$P(x|C_2)$: 1×1 — $P(C_2)$: $\frac{1}{13}$

$P(x|C_2)$: $\frac{1}{3} \times \frac{1}{3}$ — $P(C_2)$: $\frac{12}{13}$

$$P(C_1) = \frac{1}{13} \qquad P(x_1 = 1|C_1) = 1 \qquad P(x_2 = 1|C_1) = 1$$

$$P(C_2) = \frac{12}{13} \qquad P(x_1 = 1|C_2) = \frac{1}{3} \qquad P(x_2 = 1|C_2) = \frac{1}{3}$$

Generative v.s. Discriminative

- 通常情况下，判别式模型的预测效果更好
- 生成模型的优势在于
  - 对于特定的分布来说
    - 对噪声更加稳健
  - 先验分布的获取更加灵活
  - 由于拟合过程更加精细，便于演化出其他的模型和算法，比如GAN（生成对抗网络）和其他深度生成模型。

机器学习基础（二）

- 生成模型和判别模型
- 概率生成模型
- 性能对比
- 其他生成模型
- 补充知识

拉普拉斯平滑

朴素贝叶斯模型可以在大部分情况下工作良好。但是该模型有一个缺点：对数据稀疏问题敏感。

比如在邮件分类中，假设NIPS这个词在词典中的位置为35000，然而NIPS这个词从来没有在训练数据中出现过，这是第一次出现NIPS，于是算概率时：

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

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

由于NIPS从未在垃圾邮件和正常邮件中出现过，所以结果只能是0了。于是最后的后验概率：

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

拉普拉斯平滑

对于这样的情况，我们可以采用拉普拉斯平滑，对于未出现的特征，我们赋予一个小的值而不是0。具体平滑方法为：

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

变为

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

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

机器学习基础（二）

- 生成模型和判别模型
- 概率生成模型
- 性能对比
- 其他生成模型
- 补充知识

补充知识

- 范数
- 熵
- 模型评估
- 常用定理

向量和矩阵的范数

- 为了进行损失函数优化和误差估计，以及迭代法的收敛性的分析，我们需要对R^n(n维向量空间)中的向量或R^{nxn}中矩阵的“大小”引入一种度量——向量和矩阵的范数。

在一维数轴上，实轴上任意一点x到原点的距离用|x|表示。而任意两点x_1，x_2之间距离用| x_1-x_2 |表示。

向量范数

设任一向量$X \in R^n$,按某一确定的

法则对应于一非负实数$\| X \|$,且满足:

1)非负性:$\| X \| \geq 0$, 当且仅当$X = 0$时,$\| X \| = 0$;

2)奇次性:$\| kX \| = | k | \| X \|, k \in R$;

3)三角不等式: 对任意$X, Y \in R^n$,都有

$\| X + Y \| \leq \| X \| + \| Y \|$,

则称$\| x \|$为向量x的范数。

常见的向量范数

设向量$X=(x_1,x_2,...,x_n)^T$

$$\|X\|_1=\sum_{i=1}^{n}|x_i|$$

$$\|X\|_2=(\sum_{i=1}^{n}|x_i|^2)^{\frac{1}{2}}=(X,X)^{\frac{1}{2}}=(X^TX)^{\frac{1}{2}}$$

$$\|X\|_\infty=\max_{1\le i\le n}\{|x_i|\}$$

向量范数性质

性质1　对任意X，$y \in R^n$有　$\left\|\|x\| - \|y\|\right\| \le \|x - y\|$。

性质2　设$X \in R^n$,则向量范数$\| X \|$是分量

$x_1, x_2, ..., x_n$的一致连续函数。

性质3　对R^n中定义的任意两种范数$\| \cdot \|_\alpha, \| \cdot \|_\beta$,

则必存在两正数m, M,使得

$$m \| X \|_\beta \le \| X \|_\alpha \le M \| X \|_\beta \qquad \forall X \in R^n$$

矩阵范数

设任意$A \in R^{n\times n}$,若按某一确定的法则对应于一非负实数$\| A \|$,且满足:

1)非负性:$\| A \| \geq 0$，当且仅当$A = 0$时,$\| A \| = 0$;

2)奇次性:$\| kA \| = | k | \| A \|$，$k \in R$;

3)三角不等式:$\| A + B \| \leq \| A \| + \| B \|, \forall A, B \in R^{n\times n}$;

4)相容性:$\|AB\| \leq \|A\| \ \|B\|$，$\forall A, B \in R^{n\times n}$,

则称$\| A \|$为$R^{n\times n}$的一种范数。

矩阵范数

设$\|x\|,\|A\|$分别为R^n和$R^{n\times n}$

的一种范数,如果

$$\|Ax\|\leq\|A\|\,\|x\|$$

则称该矩阵范数$\|A\|$与此向量范数$\|x\|$是相容的。

设$x\in R^n, A\in R^{n\times n}$,并在$R^n$上

定义向量范数$\|x\|$,则

$$\|A\|=\max_{x\neq 0}\frac{\|Ax\|}{\|x\|}=\max_{\|x\|=1}\|Ax\|$$

为$R^{n\times n}$上的矩阵范数,且称其为算子范数。

常见的矩阵范数

设矩阵$A=(a_{ij})_{n\times n}\in R^{n\times n}$，$X\in R^n$，

则于向量范数$\|X\|_p$ $(p=1,2,\infty)$相容的矩阵范数是

$$1-\text{范数}:\|A\|_1=\max_{1\le j\le n}\sum_{i=1}^{n}|a_{ij}|$$

$$2-\text{范数}:\|A\|_2=\sqrt{\lambda_{\max}(AA^T)}$$

$$\infty-\text{范数}:\|A\|_\infty=\max_{1\le i\le n}\sum_{j=1}^{n}|a_{ij}|$$

$$F-\text{范数}:\|A\|_F=(\sum_{j=1}^{n}\sum_{i=1}^{n}a_{ij}^2)^{\frac{1}{2}}$$

一般称$\|A\|_1$为矩阵的列范数，$\|A\|_\infty$为矩阵的行范数，$\|A\|_2$为矩阵的谱范数或欧几里德范数。

补充知识

- 范数
- 熵
- 模型评估
- 常用定理

熵（Entropy）

- 在信息论中，熵用来衡量一个随机事件的不确定性。
  - 自信息（Self Information）
  - 熵
  - 熵越高，则随机变量的信息越多；熵越低，则随机变量的信息越少。.
- 熵的特性
  - 连续性
  - 对称性
  - 极值性 $H_n(p_1, p_2, \dots\dots p_n) \leq H_n(\frac{1}{n}, \frac{1}{n}, \dots\dots \frac{1}{n})$
  - 可加性

$$I(x) = -\log(p(x))$$

$$\begin{aligned} H(X) &= \mathbb{E}_X[I(x)] \\ &= \mathbb{E}_X[-\log p(x)] \\ &= -\sum_{x \in X} p(x) \log p(x) \end{aligned}$$

微分熵和联合熵

- 连续形式熵的定义。

$$H(x) = -\int p(x) log p(x) dx$$

  - 将熵原始定义的求和积分替代。

- 如果分布为高维，则为联合熵

$$H(x,y) = -\sum_{x \in X} \sum_{y \in Y} p(x,y) log p(x,y)$$

  - 或者

$$H(x,y) = -\iint p(x,y) log p(x,y) dxdy$$

条件熵

可理解为给定X的值前提下随机变量Y的随机性程度。

$$H(Y \mid X) = \sum_{x \in X} p(x) H(Y \mid X = x)$$

$$= \sum_{x \in X} p(x)[-\sum_{y \in Y} p(y \mid x) \log_2 p(y \mid x)]$$

$$= -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(y \mid x)$$

也可以写成

$$H[\boldsymbol{y} \mid \boldsymbol{x}] = -\iint p(\boldsymbol{y}, \boldsymbol{x}) \ln p(\boldsymbol{y} \mid \boldsymbol{x}) \, \mathrm{d}\boldsymbol{y} \, \mathrm{d}\boldsymbol{x}$$

将 (2)式：$H(X,Y) = -\sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 p(x,y)$ 中的 $\log_2 p(x,y)$ 根据概率公式展开：

$$H(X,Y) = -\sum_{x \in X} \sum_{y \in Y} p(x,y) [\log p(x) p(y \mid x)]$$

$$= -\sum_{x \in X} \sum_{y \in Y} p(x,y) [\log p(x) + \log p(y \mid x)]$$

$$= -\sum_{x \in X} \sum_{y \in Y} p(x,y) \log p(x) - \sum_{x \in X} \sum_{y \in Y} p(x,y) \log p(y \mid x)$$

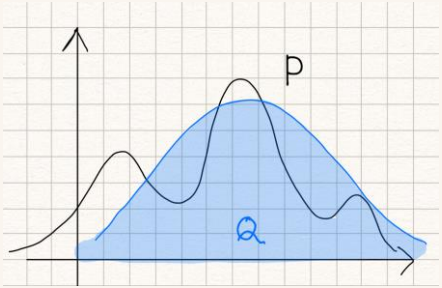
$$= -\sum_{x \in X} p(x) \log p(x) - \sum_{x \in X} \sum_{y \in Y} p(x,y) \log p(y \mid x)$$

$$= H(X) + H(Y \mid X) \qquad (4) \quad (\text{连锁规则})$$

根据和联合熵的关系，描述 x 和 y 所需的信息是描述 x 自己所需的信息,加上给定 x 的情况下具体化 y 所需的额外信息

相对熵(relative entropy)

- 相对熵又叫KL散度(Kullback-Leibler Divergence)是用概率分布q来近似p时所造成的信息损失量。
  - KL散度是按照概率分布q的最优编码对真实分布为p的信息进行编码，其平均编码长度（即交叉熵）H(p,q)和p的最优平均编码长度（即熵）H(p)之间的差异。



$$\mathrm{KL}(p,q) = H(p,q) - H(p)$$

$$= \sum_x p(x) \log \frac{p(x)}{q(x)}$$

交叉熵（Cross Entropy）

- 交叉熵是按照概率分布q的样本对真实分布为p的差异。

$$H(p,q) = \mathbb{E}_p[-\log q(x)]$$
$$= -\sum_x p(x) \log q(x)$$

  - 在给定 q 的情况下，如果 p和 q 越接近，交叉熵越小；
  - 如果 p 和 q 越远，交叉熵就越大 .

其他熵

困惑度(perplexity)

$$PP(S) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{p(w_1 w_2 \dots w_N)}} = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{p(w_i | w_1 w_2 \dots w_{i-1})}}$$

或者等价地，

$$PP(S) = 2^{-\frac{1}{N} \sum log(P(w_i))}$$

互信息(mutual information)

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) log \frac{p(x,y)}{p(x)p(y)}$$

点互信息PMI（Pointwise Mutual Information）

$$PMI(x;y) = log \frac{p(x,y)}{p(x)p(y)} = log \frac{p(x|y)}{p(x)} = log \frac{p(y|x)}{p(y)}$$

Renyi熵 Renyi entropy

$$H = \frac{1}{1-q} \ln \sum_{i=1}^{N} P_i^q$$

diversity

$${}^qD = \frac{1}{M_{q-1}} = \frac{1}{\sqrt[q-1]{\sum_{i=1}^{R} p_i p_i^{q-1}}} = \left(\sum_{i=1}^{R} p_i^q \right)^{1/(1-q)}$$

补充知识

- 范数
- 熵
- 模型评估
- 常用定理

如何选择一个合适的模型？

- 模型选择
  - 拟合能力强的模型一般复杂度会比较高，容易过拟合。
  - 如果限制模型复杂度，降低拟合能力，可能会欠拟合。
- 偏差与方差分解
  - 期望错误可以分解为

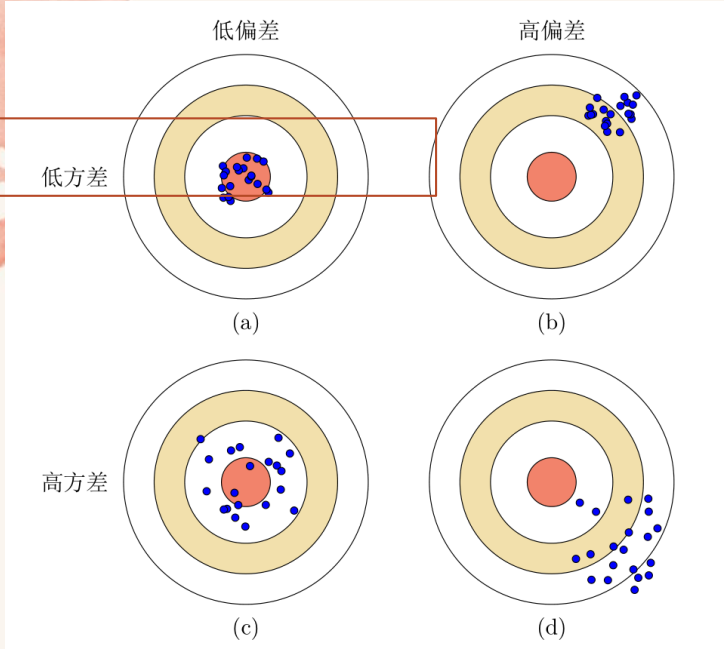
$$\mathcal{R}(f) = (\text{bias})^2 + \text{variance} + \varepsilon.$$

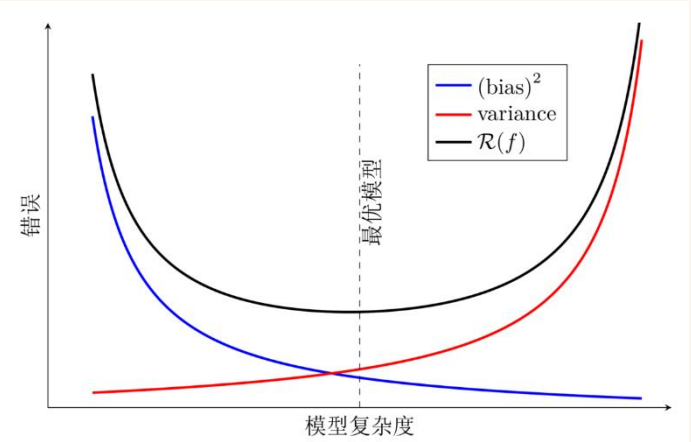
$$\mathbb{E}_{\mathbf{x}}\left[\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathcal{D}}(\mathbf{x})\right] - f^*(\mathbf{x})\right)^2\right]$$

$$\mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}\left[\left(f_{\mathcal{D}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\left[f_{\mathcal{D}}(\mathbf{x})\right]\right)^2\right]\right]$$

$$\mathbb{E}_{(\mathbf{x},y)\sim p_r(\mathbf{x},y)}\left[\left(y - f^*(\mathbf{x})\right)^2\right]$$

模型选择：偏差与方差





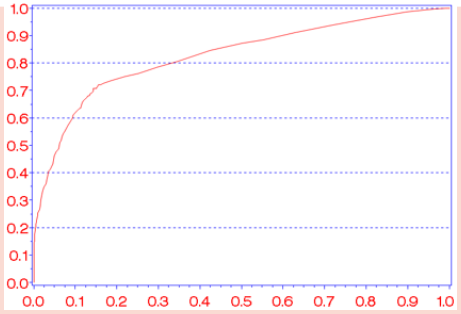
集成模型：有效的降低方差的方法

排序类模型的评估指标

该类模型的需求是二分类目标。比如预测一下客户违约的概率、营销响应的概率。

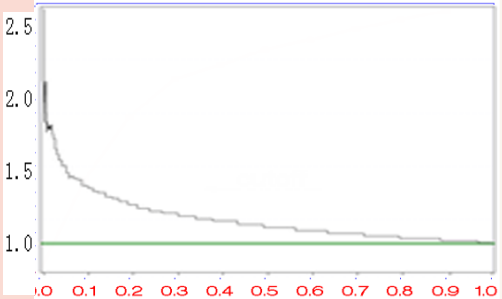
ROC曲线：用来描述模型分辨能力,对角线以上的图形越高模型越好

X:1-特异度
Y：灵敏度



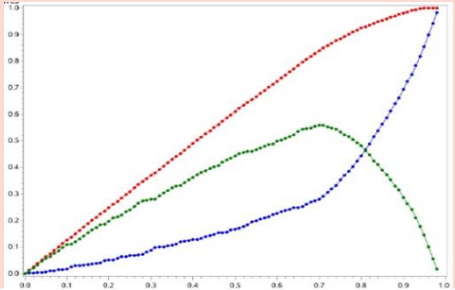
累积提升曲线：由于展示使用模型预测结果与随机情况下获取显性样本的能力比较

X:深度
Y：正例的累积密度除以基准概率



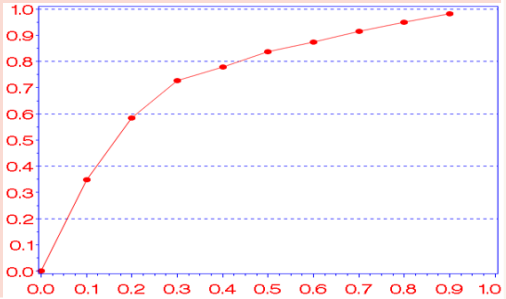
K-S曲线：用来描述模型对违约客户的分辨能力

X:深度
Y红：正例的累积密度
Y蓝：负例的累积密度
Y率：K-S值



洛伦兹曲线：用来描述预期违约客户的分布

X:深度
Y：正例的累积密度



混淆矩阵（Confusion Matrix）

Confusion Matrix		真实值	
		0(Secondary)	1(Primary)
预测值	0(Secondary)	TN(真阴性)	FN(伪阴性)
	1(Primary)	FP(伪阳性)	TP(真阳性)

$$\text{Accuracy} = \frac{TN+TP}{TN+FN+FP+TP} \quad \text{Misclassification} = \frac{FN+FP}{TN+FN+FP+TP}$$

$$\text{Precision} = \frac{TP}{FP+TP} \quad \text{Recall} = \frac{TP}{FN+TP}$$

$$\text{F-Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

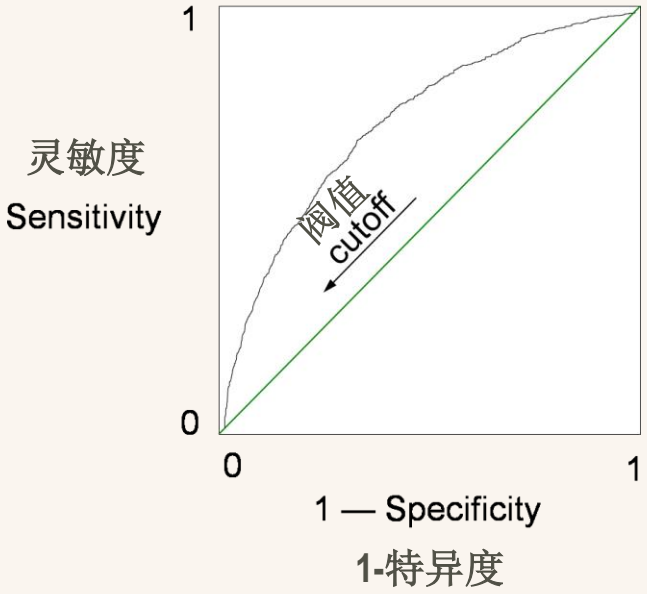
$$\text{Sensitivity} = \text{True Positive Rate} = \text{Recall}$$

$$\text{Specificity} = \text{True Negative Rate} = \frac{TN}{TN+FP}$$

ROC图形

	阈值	敏感度	特异度
1	0.96	0.003	0.998
2	0.91	0.022	0.997
3	0.89	0.038	0.996
4	0.83	0.082	0.989
5	0.75	0.173	0.959
6	0.72	0.227	0.948
7	0.69	0.280	0.929
8	0.61	0.438	0.846
9	0.60	0.467	0.834
10	0.52	0.670	0.721
11	0.48	0.730	0.666
12	0.41	0.829	0.517
13	0.36	0.880	0.412
14	0.35	0.882	0.399
15	0.30	0.908	0.323
16	0.21	0.953	0.200
17	0.17	0.967	0.152
18	0.11	0.983	0.092
19	0.05	0.991	0.048
20	0.00	0.998	0.005

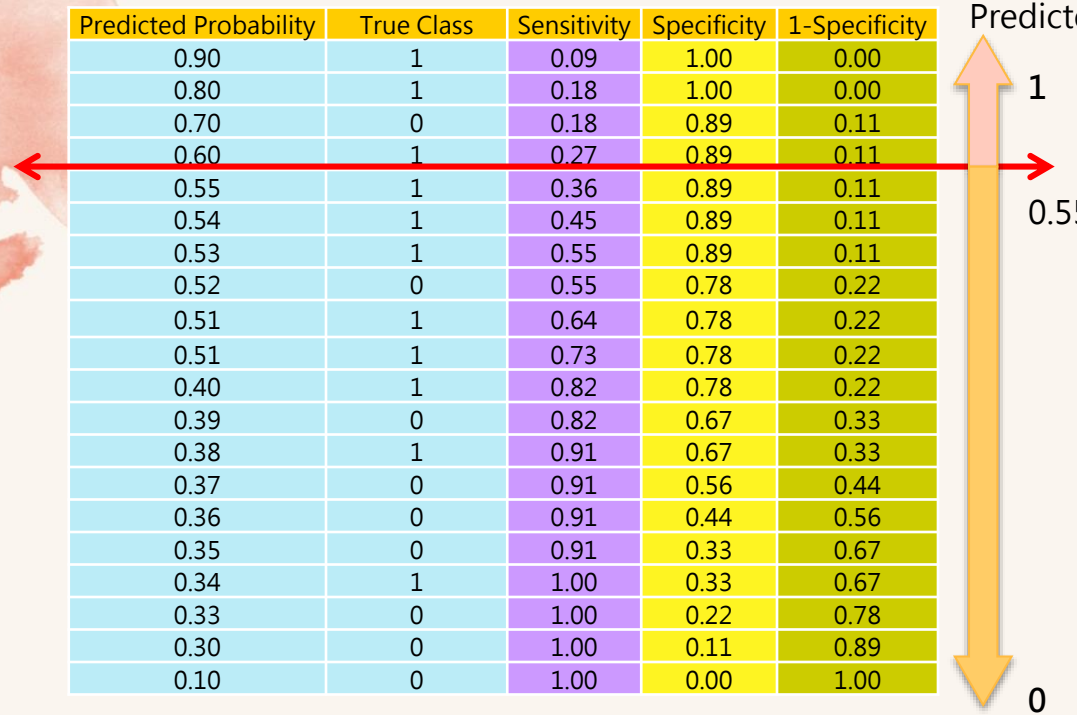
阀值下降



随着阀值的下降，灵敏度在升高，特异度在降低。

ROC 曲线

Predicted Probability	True Class	Sensitivity	Specificity	1-Specificity
0.90	1	0.09	1.00	0.00
0.80	1	0.18	1.00	0.00
0.70	0	0.18	0.89	0.11
0.60	1	0.27	0.89	0.11
0.55	1	0.36	0.89	0.11
0.54	1	0.45	0.89	0.11
0.53	1	0.55	0.89	0.11
0.52	0	0.55	0.78	0.22
0.51	1	0.64	0.78	0.22
0.51	1	0.73	0.78	0.22
0.40	1	0.82	0.78	0.22
0.39	0	0.82	0.67	0.33
0.38	1	0.91	0.67	0.33
0.37	0	0.91	0.56	0.44
0.36	0	0.91	0.44	0.56
0.35	0	0.91	0.33	0.67
0.34	1	1.00	0.33	0.67
0.33	0	1.00	0.22	0.78
0.30	0	1.00	0.11	0.89
0.10	0	1.00	0.00	1.00



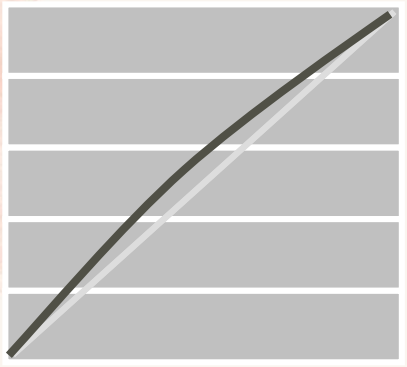
CM		True	
		0	1
P	0	8	8
	1	1	3

Accuracy = 0.55
Precision = 0.75
Recall = 0.27

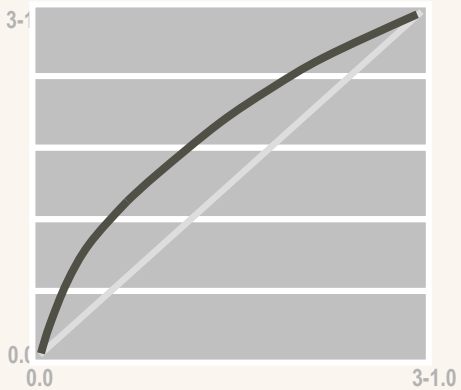
Sensitivity = Recall

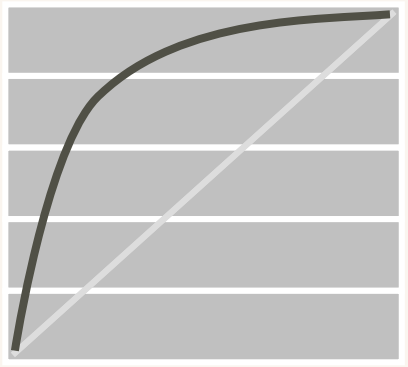
Specificity
= True Negative Rate
= 0.89

Area Under the Curve (AUC)



弱的模型





强的模型

ROC曲线结果的取值在[0.5,1]。

一般来说，

[0.5,0.7)表示效果较低,但是预测股票已经很不错了；

[0.7,0.85) 表示效果一般；

[0.85,0.95)表示效果良好;

[0.95,0.1]社会科学建模中不大可能出现。

注意：

①有时ROC曲线可能会落入对角线以下，这时需检查检验方向与状态值的对应关系

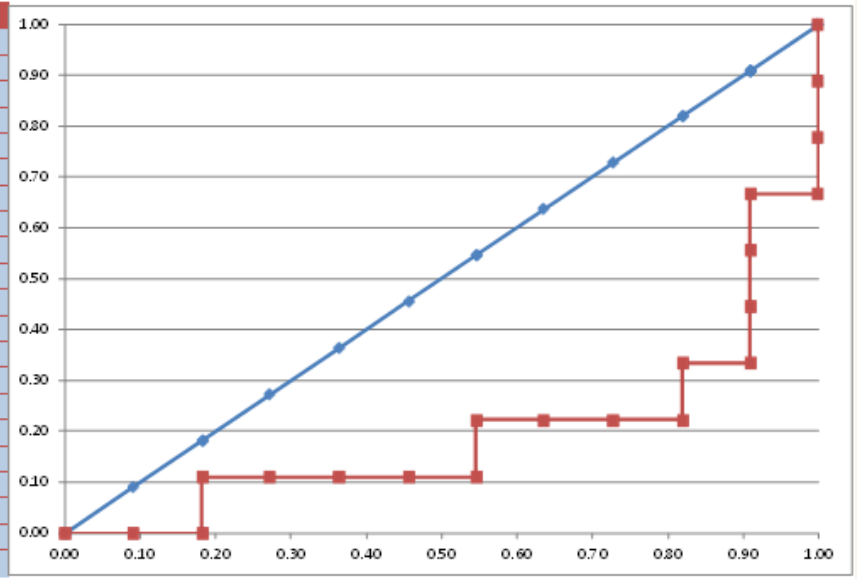
②如果某ROC曲线在对角线两边均有分布，需检查数据或专业背景。

Gini 系数

Predicted Probability	True Class	Cumulative for 1	Cumulative for 0	X	Y
0.90	1	0.09	0.00	0.09	0.00
0.80	1	0.18	0.00	0.18	0.00
0.70	0	0.18	0.11	0.18	0.11
0.60	1	0.27	0.11	0.27	0.11
0.55	1	0.36	0.11	0.36	0.11
0.54	1	0.45	0.11	0.45	0.11
0.53	1	0.55	0.11	0.55	0.11
0.52	0	0.55	0.22	0.55	0.22
0.51	1	0.64	0.22	0.64	0.22
0.51	1	0.73	0.22	0.73	0.22
0.40	1	0.82	0.22	0.82	0.22
0.39	0	0.82	0.33	0.82	0.33
0.38	1	0.91	0.33	0.91	0.33
0.37	0	0.91	0.44	0.91	0.44
0.36	0	0.91	0.56	0.91	0.56
0.35	0	0.91	0.67	0.91	0.67
0.34	1	1.00	0.67	1.00	0.67
0.33	0	1.00	0.78	1.00	0.78
0.30	0	1.00	0.89	1.00	0.89
0.10	0	1.00	1.00	1.00	1.00

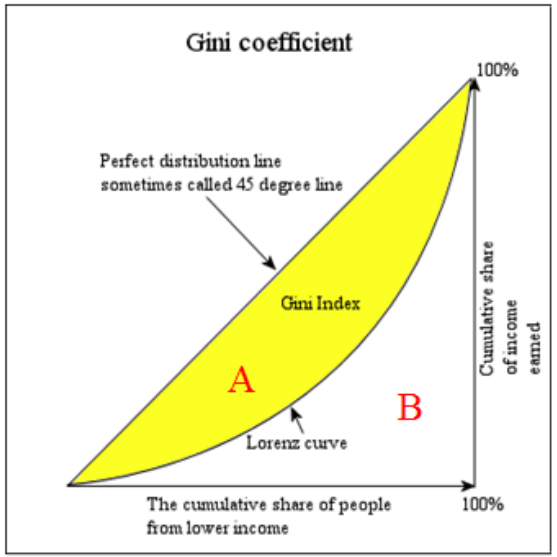
Gini 系数

Cumulative for 1	Random	Cumulative for 0
0.00	0.00	0.00
0.09	0.09	0.00
0.18	0.18	0.00
0.18	0.18	0.11
0.27	0.27	0.11
0.36	0.36	0.11
0.45	0.45	0.11
0.55	0.55	0.11
0.55	0.55	0.22
0.64	0.64	0.22
0.73	0.73	0.22
0.82	0.82	0.22
0.82	0.82	0.33
0.91	0.91	0.33
0.91	0.91	0.44
0.91	0.91	0.56
0.91	0.91	0.67
1.00	1.00	0.67
1.00	1.00	0.78
1.00	1.00	0.89
1.00	1.00	1.00



Gini 系数 vs. ROC Index

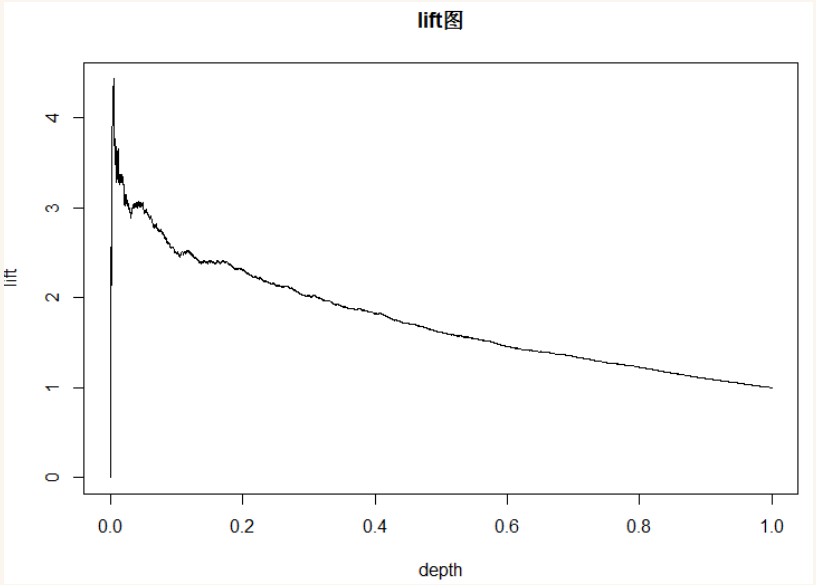
- The Gini coefficient is calculated as a ratio of the areas on the Lorenz curve diagram
  - A : the area between the line of perfect equality and Lorenz curve
  - B : the area underneath the Lorenz curve
  - Gini coefficient is A/(A+B)



ROC Index = A + 0.5
Gini Coefficient = A / (A+B) = 2A = 2 * (ROC Index – 0.5)

累积提升度

深度	为真的组中的累积百分比	随机模型	累积提升度
0.0500	0.0900	0.0500	1.8000
0.1000	0.1800	0.1000	1.8000
0.1500	0.1800	0.1500	1.2000
0.2000	0.2700	0.2000	1.3500
0.2500	0.3600	0.2500	1.4400
0.3000	0.4500	0.3000	1.5000
0.3500	0.5500	0.3500	1.5714
0.4000	0.5500	0.4000	1.3750
0.4500	0.6400	0.4500	1.4222
0.5000	0.7300	0.5000	1.4600
0.5500	0.8200	0.5500	1.4909
0.6000	0.8200	0.6000	1.3667
0.6500	0.9100	0.6500	1.4000
0.7000	0.9100	0.7000	1.3000
0.7500	0.9100	0.7500	1.2133
0.8000	0.9100	0.8000	1.1375
0.8500	1.0000	0.8500	1.1765
0.9000	1.0000	0.9000	1.1111
0.9500	1.0000	0.9500	1.0526
1.0000	1.0000	1.0000	1.0000

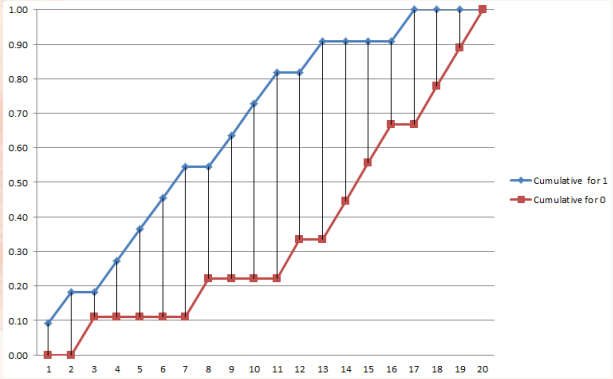


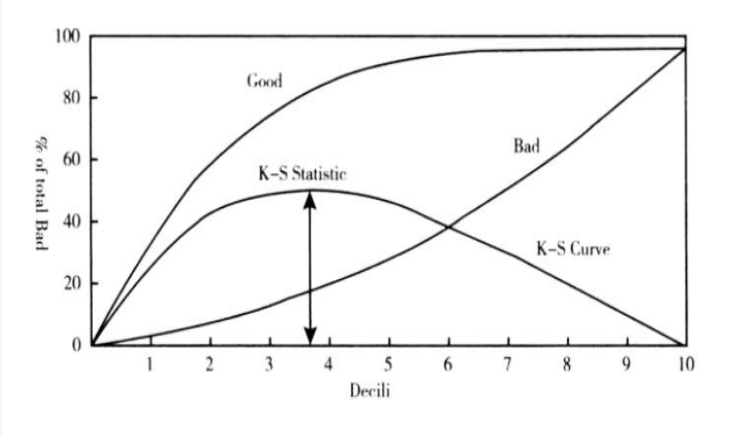
K-S统计量

Predicted Probability	True Class	Cumulative for 1	Cumulative for 0	Difference*100
0.90	1	0.09	0.00	9
0.80	1	0.18	0.00	18
0.70	0	0.18	0.11	7
0.60	1	0.27	0.11	16
0.55	1	0.36	0.11	25
0.54	1	0.45	0.11	34
0.53	1	0.55	0.11	44
0.52	0	0.55	0.22	33
0.51	1	0.64	0.22	42
0.51	1	0.73	0.22	51
0.40	1	0.82	0.22	60
0.39	0	0.82	0.33	49
0.38	1	0.91	0.33	58
0.37	0	0.91	0.44	47
0.36	0	0.91	0.56	35
0.35	0	0.91	0.67	24
0.34	1	1.00	0.67	33
0.33	0	1.00	0.78	22
0.30	0	1.00	0.89	11
0.10	0	1.00	1.00	0

(K-S = 60)

K-S曲线





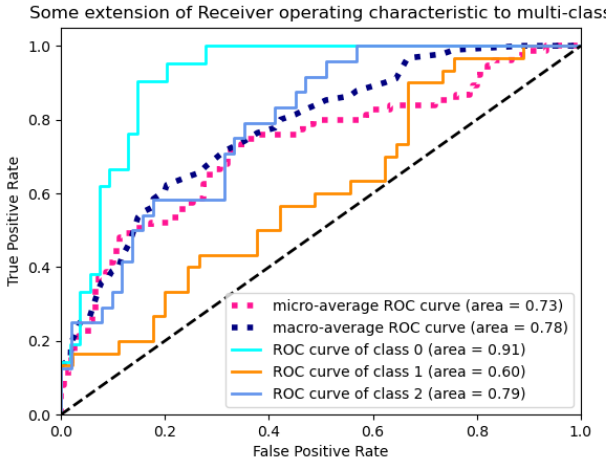
K-S 统计量

- ✓ 小于20：模型无鉴别能力
- ✓ 20~40之间：模型勉强接受
- ✓ 41~50之间：模型具有区别能力
- ✓ 51~60之间：此模型有很好的区别能力
- ✓ 61~75之间：此模型有非常好的区别能力
- ✓ 大于75：此模型异常，可能有问题

metrics.accuracy_score(y_true, y_pred, *[, ...])	Accuracy classification score.
metrics.auc(x, y)	Compute Area Under the Curve (AUC) using the trapezoidal rule.
metrics.average_precision_score(y_true, ...)	Compute average precision (AP) from prediction scores.
metrics.balanced_accuracy_score(y_true, ...)	Compute the balanced accuracy.
metrics.brier_score_loss(y_true, y_prob, *)	Compute the Brier score loss.
metrics.classification_report(y_true, y_pred, *)	Build a text report showing the main classification metrics.
metrics.cohen_kappa_score(y1, y2, *[, ...])	Cohen's kappa: a statistic that measures inter-annotator agreement.
metrics.confusion_matrix(y_true, y_pred, *)	Compute confusion matrix to evaluate the accuracy of a classification.
metrics.dcg_score(y_true, y_score, *[, k, ...])	Compute Discounted Cumulative Gain.
metrics.det_curve(y_true, y_score[, ...])	Compute error rates for different probability thresholds.
metrics.f1_score(y_true, y_pred, *[, ...])	Compute the F1 score, also known as balanced F-score or F-measure.
metrics.fbeta_score(y_true, y_pred, *, beta)	Compute the F-beta score.
metrics.hamming_loss(y_true, y_pred, *[, ...])	Compute the average Hamming loss.
metrics.hinge_loss(y_true, pred_decision, *)	Average hinge loss (non-regularized).
metrics.jaccard_score(y_true, y_pred, *[, ...])	Jaccard similarity coefficient score.
metrics.log_loss(y_true, y_pred, *[, eps, ...])	Log loss, aka logistic loss or cross-entropy loss.
metrics.matthews_corrcoef(y_true, y_pred, *)	Compute the Matthews correlation coefficient (MCC).
metrics.multilabel_confusion_matrix(y_true, ...)	Compute a confusion matrix for each class or sample.
metrics.ndcg_score(y_true, y_score, *[, k, ...])	Compute Normalized Discounted Cumulative Gain.
metrics.precision_recall_curve(y_true, ...)	Compute precision-recall pairs for different probability thresholds.
metrics.precision_recall_fscore_support(...)	Compute precision, recall, F-measure and support for each class.
metrics.precision_score(y_true, y_pred, *[, ...])	Compute the precision.
metrics.recall_score(y_true, y_pred, *[, ...])	Compute the recall.
metrics.roc_auc_score(y_true, y_score, *[, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
metrics.roc_curve(y_true, y_score, *[, ...])	Compute Receiver operating characteristic (ROC).
metrics.top_k_accuracy_score(y_true, y_score, *)	Top-k Accuracy classification score.
metrics.zero_one_loss(y_true, y_pred, *[, ...])	Zero-one classification loss.

sklearn.metrics.**roc_auc_score**(y_true, y_score, *, average='*macro*', sample_weight=*None*, max_fpr=*None*, multi_class='*raise*', labels=*None*)

sklearn.metrics.**roc_curve**(y_true, y_score, *, pos_label=*None*, sample_weight=*None*, drop_intermediate=*True*)



Model Diagnostic Statistics

- R-Square:

$$R^2 = 1 - \sum_{t=1}^{n} (Y_t - \hat{Y}_t)^2 \Big/ \sum_{t=1}^{n} (Y_t - \bar{Y})^2$$

- Root Mean Squared Error:

$$\mathrm{MSE}^* = \frac{1}{n-k} \sum_{t=1}^{n} (Y_t - \hat{Y}_t)^2$$

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}}$$

* For test data

Model Diagnostic Statistics

- Mean Absolute Percent Error:

$$\mathrm{MAPE} = \frac{1}{n}\sum_{t=1}^{n} | Y_t - \hat{Y}_t | / Y_t$$

- Mean Absolute Error:

$$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} | Y_t - \hat{Y}_t |$$

Model Diagnostic Statistics

- Symmetric Mean Absolute Percent Error:

$$\mathrm{SMAPE} = \frac{1}{n}\sum_{t=1}^{n} | Y_t - \hat{Y}_t | / [(Y_t + \hat{Y}_t)/2]$$

- Mean Absolute Error/Mean:

$$\mathrm{MAE/Mean} = \frac{\frac{1}{n}\sum_{t=1}^{n} | Y_t - \hat{Y}_t |}{\frac{1}{n}\sum_{t=1}^{n} Y_t}$$

Model Diagnostic Statistics

- Likelihood-Based Information Criteria
- General Formula:
- IC = Accuracy + Penalty
- Akaike's A Information Criteria:

$$\mathrm{AIC} = -2\log(L) + 2k$$

- Schwarz's Bayesian Information Criteria:

$$\mathrm{SBC} = -2\log(L) + k\log(n)$$

Model Diagnostic Statistics

- Error-Based Information Criteria
- Akaike's A Information Criteria:

$$\mathrm{AIC} = n\log(\mathrm{SSE}/n) + 2k$$

- Schwarz's Bayesian Information Criteria:

$$\mathrm{SBC} = n\log(\mathrm{SSE}/n) + k\log(n)$$

Regression metrics

See the Regression metrics section of the user guide for further details.

`metrics.explained_variance_score`(y_true, ...)	Explained variance regression score function.
`metrics.max_error`(y_true, y_pred)	max_error metric calculates the maximum residual error.
`metrics.mean_absolute_error`(y_true, y_pred, *)	Mean absolute error regression loss.
`metrics.mean_squared_error`(y_true, y_pred, *)	Mean squared error regression loss.
`metrics.mean_squared_log_error`(y_true, y_pred, *)	Mean squared logarithmic error regression loss.
`metrics.median_absolute_error`(y_true, y_pred, *)	Median absolute error regression loss.
`metrics.mean_absolute_percentage_error`(...)	Mean absolute percentage error regression loss.
`metrics.r2_score`(y_true, y_pred, *[, ...])	R^2 (coefficient of determination) regression score function.
`metrics.mean_poisson_deviance`(y_true, y_pred, *)	Mean Poisson deviance regression loss.
`metrics.mean_gamma_deviance`(y_true, y_pred, *)	Mean Gamma deviance regression loss.
`metrics.mean_tweedie_deviance`(y_true, y_pred, *)	Mean Tweedie deviance regression loss.

Multilabel ranking metrics ¶

See the Multilabel ranking metrics section of the user guide for further details.

`metrics.coverage_error`(y_true, y_score, *[, ...])	Coverage error measure.
`metrics.label_ranking_average_precision_score`(...)	Compute ranking-based average precision.
`metrics.label_ranking_loss`(y_true, y_score, *)	Compute Ranking loss measure.

```
from statsmodels.regression.linear_model import OLS
from statsmodels.tools import add_constant

regr = OLS(y, add_constant(X)).fit()
print(regr.aic)
```

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2, 7]
>>> y_pred = [2.5, 0.0, 2, 8]
>>> mean_absolute_error(y_true, y_pred)
0.5
>>> y_true = [[0.5, 1], [-1, 1], [7, -6]]
>>> y_pred = [[0, 2], [-1, 2], [8, -5]]
>>> mean_absolute_error(y_true, y_pred)
0.75
>>> mean_absolute_error(y_true, y_pred, multioutput='raw_values')
array([0.5, 1. ])
>>> mean_absolute_error(y_true, y_pred, multioutput=[0.3, 0.7])
0.85...
```

sklearn.metrics.**mean_absolute_error**(y_true, y_pred, *, sample_weight=*None*, multioutput='*uniform_average*')

补充知识

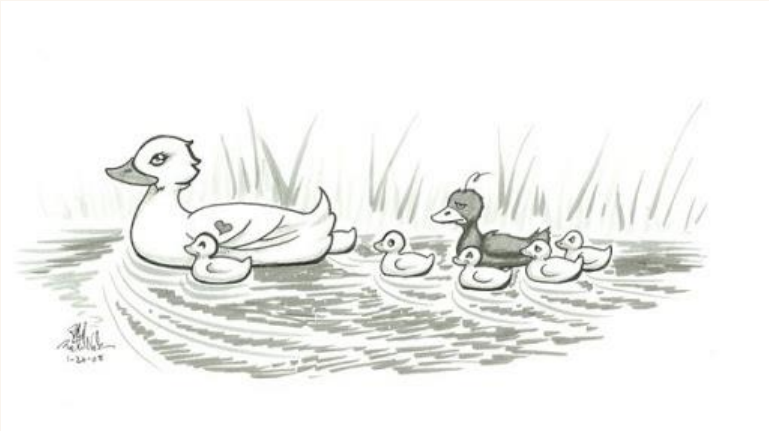
- 范数
- 熵
- 模型评估
- 常用定理

常用的定理

- 没有免费午餐定理（No Free Lunch Theorem，NFL）
  - 对于基于迭代的最优化算法，不存在某种算法对所有问题（有限的搜索空间内）都有效。如果一个算法对某些问题有效，那么它一定在另外一些问题上比纯随机搜索算法更差。

常用的定理

- 丑小鸭定理(Ugly Duckling Theorem)
  - 丑小鸭与白天鹅之间的区别和两只白天鹅之间的区别一样大.



常用的定理

- 奥卡姆剃刀原理(Occam's Razor)
  - 如无必要，勿增实体

归纳偏置(Inductive Bias)

- 很多学习算法经常会对学习的问题做一些假设，这些假设就称为**归纳偏置**。
  - 在最近邻分类器中，我们会假设在特征空间中，一个小的局部区域中的大部分样本都同属一类。
  - 在朴素贝叶斯分类器中，我们会假设每个特征的条件概率是互相独立的。
  - 归纳偏置在贝叶斯学习中也经常称为先验（Prior）。

END

(Q&A)