第三章 线性模型

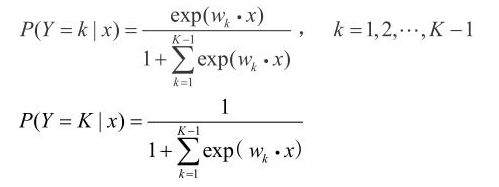
第一部分 本章概要

1.最小二乘法(least square method):基于均方误差最小化进行模型求解的方法。均方误差的几何意义对应了欧几里得距离。在线性回归中，最小二乘法就是找到一条直线，是的所有样本到直线上的欧氏距离之和最小。

2.极大似然法(maximum likehood method)：估计类条件概率的一种常用策略是先假定其具有某种确定的概率分布，在基于训练样本对概率分布的参数进行估计。概率模型的训练过程，就是参数估计（parameter estimation）的过程。频率主义学派认为参数是未知的，但是客观存在的固定值，因此可以优化似然函数等准则来确定参数。贝叶斯学派认为参数是未观察到的随机变量，其自身也可以有分布，因此可假定参数服从一个先验分布，然后基于观测到的数据来计算参数的后验分布。

3.Sigmoid函数：对数几率函数，可以将输出值转化为[0,1]之间的数，实现二分类。

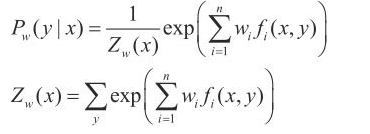
4.逻辑斯谛回归模型是由以下条件概率分布表示的分类模型。逻辑斯谛回归模型可以用于二类或多类分类，是以似然函数为目标函数的最优化问题，目标函数为光滑的凸函数，可以通过迭代尺度法、梯度下降法、牛顿法或拟牛顿法。



这里，x为输入特征，w为特征的权值。

逻辑斯谛回归模型源自逻辑斯谛分布，其分布函数F(x)是S形函数。逻辑斯谛回归模型是由输入的线性函数表示的输出的对数几率模型。

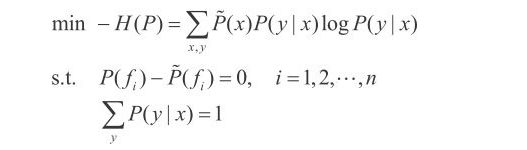
5．最大熵模型是由以下条件概率分布表示的分类模型。最大熵模型也可以用于二类或多类分类。



其中，Zw(x)是规范化因子，fi为特征函数，wi为特征的权值。

6.最大熵模型可以由最大熵原理推导得出。最大熵原理是概率模型学习或估计的一个准则。最大熵原理认为在所有可能的概率模型（分布）的集合中，熵最大的模型是最好的模型。

最大熵原理应用到分类模型的学习中，有以下约束最优化问题：



求解此最优化问题的对偶问题得到最大熵模型。

7.逻辑斯谛回归模型与最大熵模型都属于对数线性模型。

8.逻辑斯谛回归模型及最大熵模型学习一般采用极大似然估计，或正则化的极大似然估计。逻辑斯谛回归模型及最大熵模型学习可以形式化为无约束最优化问题。求解该最优化问题的算法有改进的迭代尺度法、梯度下降法、拟牛顿法。

第二部分 常见考点

**1.简介**

      逻辑回归是一种分类算法。

**2.正式介绍**

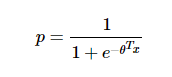
     如何凸显你是一个对逻辑回归已经非常了解的人呢。那就是用一句话概括它！**逻辑回归假设数据服从伯努利分布,通过极大化似然函数的方法，运用梯度下降来求解参数，来达到将数据二分类的目的。**

     这里面其实包含了5个点 1：逻辑回归的假设，2：逻辑回归的损失函数，3：逻辑回归的求解方法，4：逻辑回归的目的，5:逻辑回归如何分类。这些问题是考核你对逻辑回归的基本了解。

* **逻辑回归的基本假设**
  + 任何的模型都是有自己的假设，在这个假设下模型才是适用的。逻辑回归的**第一个**基本假设是**假设数据服从伯努利分布。**伯努利分布有一个简单的例子是抛硬币，抛中为正面的概率是p,抛中为负面的概率是1−p.在逻辑回归这个模型里面是假设 hθ(x) 为样本为正的概率，1−hθ(x)为样本为负的概率。那么整个模型可以描述为



* + 逻辑回归的第二个假设是假设样本为正的概率是



* + 所以逻辑回归的最终形式



* **逻辑回归的损失函数**
  + 逻辑回归的损失函数是它的极大似然函数



* **逻辑回归的求解方法**
  + 由于该极大似然函数无法直接求解，我们一般通过对该函数进行梯度下降来不断逼急最优解。在这个地方其实会有个加分的项，考察你对其他优化方法的了解。因为就梯度下降本身来看的话就有随机梯度下降，批梯度下降，small batch 梯度下降三种方式，面试官可能会问这三种方式的优劣以及如何选择最合适的梯度下降方式。
    - 简单来说 批梯度下降会获得全局最优解，缺点是在更新每个参数的时候需要遍历所有的数据，计算量会很大，并且会有很多的冗余计算，导致的结果是当数据量大的时候，每个参数的更新都会很慢。
    - 随机梯度下降是以高方差频繁更新，优点是使得sgd会跳到新的和潜在更好的局部最优解，缺点是使得收敛到局部最优解的过程更加的复杂。
    - 小批量梯度下降结合了sgd和batch gd的优点，每次更新的时候使用n个样本。减少了参数更新的次数，可以达到更加稳定收敛结果，一般在深度学习当中我们采用这种方法。
  + 其实这里还有一个隐藏的更加深的加分项，看你了不了解诸如Adam，动量法等优化方法。因为上述方法其实还有两个致命的问题。
    - 第一个是如何对模型选择合适的学习率。自始至终保持同样的学习率其实不太合适。因为一开始参数刚刚开始学习的时候，此时的参数和最优解隔的比较远，需要保持一个较大的学习率尽快逼近最优解。但是学习到后面的时候，参数和最优解已经隔的比较近了，你还保持最初的学习率，容易越过最优点，在最优点附近来回振荡，通俗一点说，就很容易学过头了，跑偏了。
    - 第二个是如何对参数选择合适的学习率。在实践中，对每个参数都保持的同样的学习率也是很不合理的。有些参数更新频繁，那么学习率可以适当小一点。有些参数更新缓慢，那么学习率就应该大一点。这里我们不展开，有空我会专门出一个专题介绍。
* **逻辑回归的目的**
  + 该函数的目的便是将数据二分类，提高准确率。
* **逻辑回归如何分类**
  + 逻辑回归作为一个回归(也就是y值是连续的)，如何应用到分类上去呢。y值确实是一个连续的变量。逻辑回归的做法是划定一个阈值，y值大于这个阈值的是一类，y值小于这个阈值的是另外一类。阈值具体如何调整根据实际情况选择。一般会选择0.5做为阈值来划分。

**3.对逻辑回归的进一步提问**

    逻辑回归虽然从形式上非常的简单，但是其内涵是非常的丰富。有很多问题是可以进行思考的

* **逻辑回归的损失函数为什么要使用极大似然函数作为损失函数？**
  + 损失函数一般有四种，平方损失函数，对数损失函数，HingeLoss0-1损失函数，绝对值损失函数。将极大似然函数取对数以后等同于对数损失函数。在逻辑回归这个模型下，对数损失函数的训练求解参数的速度是比较快的。至于原因大家可以求出这个式子的梯度更新



这个式子的更新速度只和相关。和sigmod函数本身的梯度是无关的。这样更新的速度是可以自始至终都比较的稳定。

* + 为什么不选平方损失函数的呢？其一是因为如果你使用平方损失函数，你会发现梯度更新的速度和sigmod函数本身的梯度是很相关的。sigmod函数在它在定义域内的梯度都不大于0.25。这样训练会非常的慢。
* **逻辑回归在训练的过程当中，如果有很多的特征高度相关或者说有一个特征重复了100遍，会造成怎样的影响？**
* 先说结论，如果在损失函数最终收敛的情况下，其实就算有很多特征高度相关也不会影响分类器的效果。
* 但是对特征本身来说的话，假设只有一个特征，在不考虑采样的情况下，你现在将它重复100遍。训练以后完以后，数据还是这么多，但是这个特征本身重复了100遍，实质上将原来的特征分成了100份，每一个特征都是原来特征权重值的百分之一。
* 如果在随机采样的情况下，其实训练收敛完以后，还是可以认为这100个特征和原来那一个特征扮演的效果一样，只是可能中间很多特征的值正负相消了。
* **为什么我们还是会在训练的过程当中将高度相关的特征去掉？**
  + 去掉高度相关的特征会让模型的可解释性更好
  + 可以大大提高训练的速度。如果模型当中有很多特征高度相关的话，就算损失函数本身收敛了，但实际上参数是没有收敛的，这样会拉低训练的速度。其次是特征多了，本身就会增大训练的时间。

**4.逻辑回归的优缺点总结**

    面试的时候，别人也经常会问到，你在使用逻辑回归的时候有哪些感受。觉得它有哪些优缺点。

**在这里我们总结了逻辑回归应用到工业界当中一些优点：**

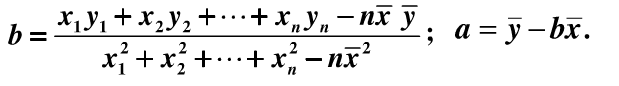
* 形式简单，模型的可解释性非常好。从特征的权重可以看到不同的特征对最后结果的影响，某个特征的权重值比较高，那么这个特征最后对结果的影响会比较大。
* 模型效果不错。在工程上是可以接受的（作为baseline)，如果特征工程做的好，效果不会太差，并且特征工程可以大家并行开发，大大加快开发的速度。
* 训练速度较快。分类的时候，计算量仅仅只和特征的数目相关。并且逻辑回归的分布式优化sgd发展比较成熟，训练的速度可以通过堆机器进一步提高，这样我们可以在短时间内迭代好几个版本的模型。
* 资源占用小,尤其是内存。因为只需要存储各个维度的特征值，。
* 方便输出结果调整。逻辑回归可以很方便的得到最后的分类结果，因为输出的是每个样本的概率分数，我们可以很容易的对这些概率分数进行cutoff，也就是划分阈值(大于某个阈值的是一类，小于某个阈值的是一类)。

**但是逻辑回归本身也有许多的缺点:**

* 准确率并不是很高。因为形式非常的简单(非常类似线性模型)，很难去拟合数据的真实分布。
* 很难处理数据不平衡的问题。举个例子：如果我们对于一个正负样本非常不平衡的问题比如正负样本比 10000:1.我们把所有样本都预测为正也能使损失函数的值比较小。但是作为一个分类器，它对正负样本的区分能力不会很好。
* 处理非线性数据较麻烦。逻辑回归在不引入其他方法的情况下，只能处理线性可分的数据，或者进一步说，处理二分类的问题 。
* 逻辑回归本身无法筛选特征。有时候，我们会用gbdt来筛选特征，然后再上逻辑回归。

第三部分 线性模型

谈及线性模型，其实我们很早就已经与它打过交道，还记得高中数学必修3课本中那个顽皮的“最小二乘法”吗？这就是线性模型的经典算法之一：根据给定的（x，y）点对，求出一条与这些点拟合效果最好的直线y=ax+b，之前我们利用下面的公式便可以计算出拟合直线的系数a,b（3.1中给出了具体的计算过程），从而对于一个新的x，可以预测它所对应的y值。前面我们提到：在机器学习的术语中，当预测值为连续值时，称为“回归问题”，离散值时为“分类问题”。本篇先从线性回归任务开始，接着讨论分类和多分类问题。

[](https://camo.githubusercontent.com/078647820dc355e3ac0ccf9f225bae9a9bdc973e/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373232623036386534382e706e67)

3.1 线性回归

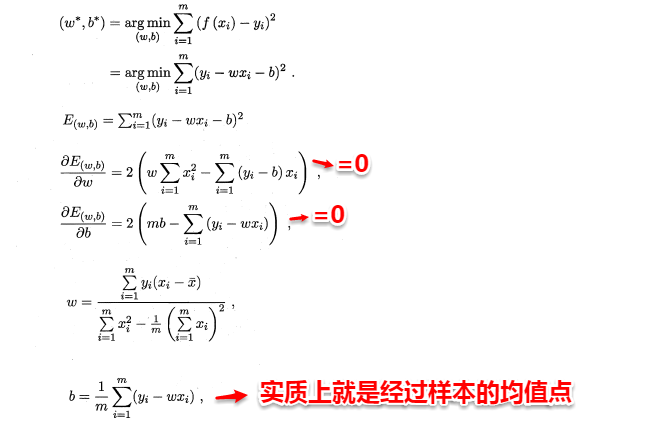
线性回归问题就是试图学到一个线性模型尽可能准确地预测新样本的输出值，例如：通过历年的人口数据预测2017年人口数量。在这类问题中，往往我们会先得到一系列的有标记数据，例如：2000-->13亿...2016-->15亿，这时输入的属性只有一个，即年份；也有输入多属性的情形，假设我们预测一个人的收入，这时输入的属性值就不止一个了，例如：（学历，年龄，性别，颜值，身高，体重）-->15k。

有时这些输入的属性值并不能直接被我们的学习模型所用，需要进行相应的处理，对于连续值的属性，一般都可以被学习器所用，有时会根据具体的情形作相应的预处理，例如：归一化等；对于离散值的属性，可作下面的处理：

若属性值之间存在“序关系”，则可以将其转化为连续值，例如：身高属性分为“高”“中等”“矮”，可转化为数值：{1， 0.5， 0}。

若属性值之间不存在“序关系”，则通常将其转化为向量的形式，例如：性别属性分为“男”“女”，可转化为二维向量：{（1，0），（0，1）}。

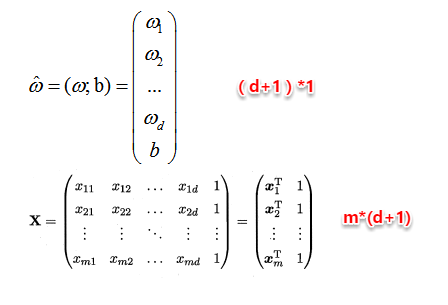
（1）当输入属性只有一个的时候，就是最简单的情形，也就是我们高中时最熟悉的“最小二乘法”（Euclidean distance），首先计算出每个样本预测值与真实值之间的误差并求和，通过最小化均方误差MSE，使用求偏导等于零的方法计算出拟合直线y=wx+b的两个参数w和b，计算过程如下图所示：

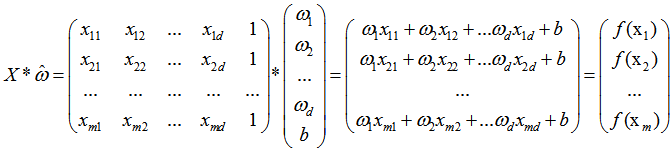
[](https://camo.githubusercontent.com/b9dcb65bfc7a46fb112f98279b8eb6cda42aa6df/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373232623063636563342e706e67)

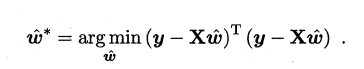
（2）当输入属性有多个的时候，例如对于一个样本有d个属性{（x1,x2...xd）,y}，则y=wx+b需要写成：

[0.png](https://camo.githubusercontent.com/c64712f78a35d82ce42aad44a1b09a520e4f8ce3/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373235363762386263642e706e67)

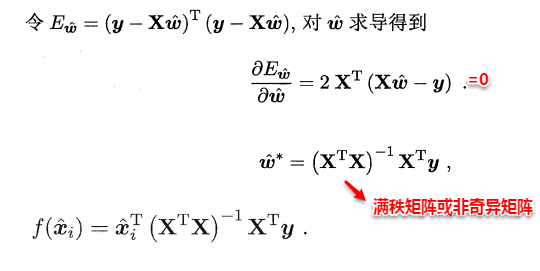
通常对于多元问题，常常使用矩阵的形式来表示数据。在本问题中，将具有m个样本的数据集表示成矩阵X，将系数w与b合并成一个列向量，这样每个样本的预测值以及所有样本的均方误差最小化就可以写成下面的形式：

[](https://camo.githubusercontent.com/fc87813f232ba8ac6524531f193a16355b99c1fe/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373232623061643866372e706e67)

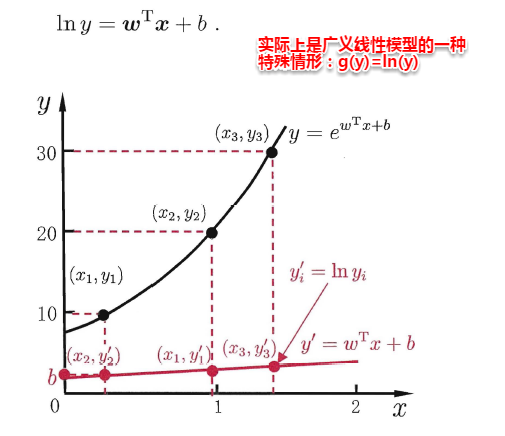
[](https://camo.githubusercontent.com/453dd6709ad6d3bba135aeaae8467bdc4a95b60e/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373232623061663635322e706e67)

[](https://camo.githubusercontent.com/29bd1407dd18d95d85f4dc1e659e9c236a1df5ec/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373232623039303534332e706e67)

同样地，我们使用最小二乘法对w和b进行估计，令均方误差的求导等于0，需要注意的是，当一个矩阵的行列式不等于0时，我们才可能对其求逆，因此对于下式，我们需要考虑矩阵（X的转置\*X）的行列式是否为0，若不为0，则可以求出其解，若为0，则需要使用其它的方法进行计算，书中提到了引入正则化，此处不进行深入。

[](https://camo.githubusercontent.com/646865de4208d00e735d6d72ed2b4f0c94a9d75c/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373232623063646533332e706e67)

另一方面，有时像上面这种原始的线性回归可能并不能满足需求，例如：y值并不是线性变化，而是在指数尺度上变化。这时我们可以采用线性模型来逼近y的衍生物，例如lny，这时衍生的线性模型如下所示，实际上就是相当于将指数曲线投影在一条直线上，如下图所示：

[](https://camo.githubusercontent.com/1a6aa357aacf19c2e3ab6fae816391e5e5db3195/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373232623130336362662e706e67)

更一般地，考虑所有y的衍生物的情形，就得到了“广义的线性模型”（generalized linear model），其中，g（\*）称为联系函数（link function）。

[8.png](https://camo.githubusercontent.com/05d606a470cfd1cb1539a9daedd97f501b8acba8/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373232623061323834312e706e67)

3.2 线性（对数）几率回归（logistic regresion）

3.2.1 逻辑回归简介

什么是 Logistic 回归？

和很多其他机器学习算法一样，逻辑回归也是从统计学中借鉴来的，尽管名字里有回归俩字儿，但它不是一个需要预测连续结果的回归算法。

与之相反，Logistic 回归是二分类任务的首选方法。它输出一个 0 到 1 之间的离散二值结果。简单来说，它的结果不是 1 就是 0。

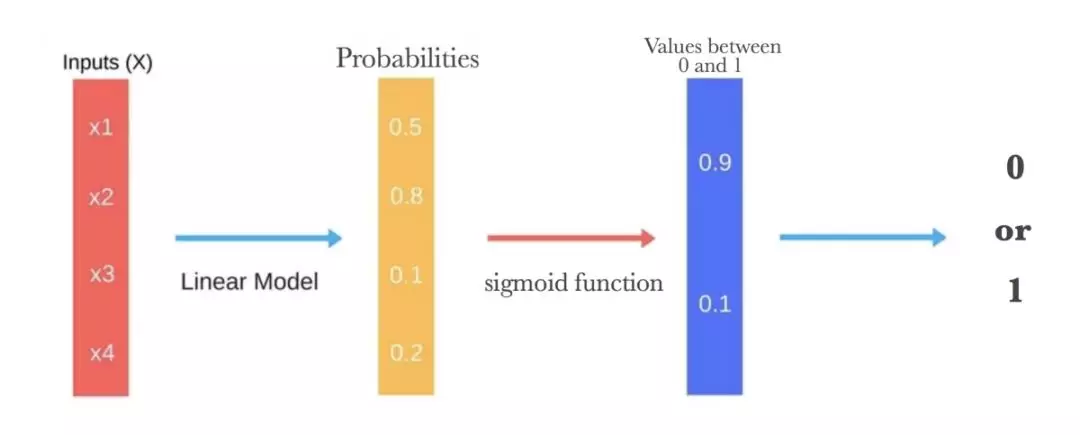
癌症检测算法可看做是 Logistic 回归问题的一个简单例子，这种算法输入病理图片并且应该辨别患者是患有癌症（1）或没有癌症（0）。

它是如何工作的?

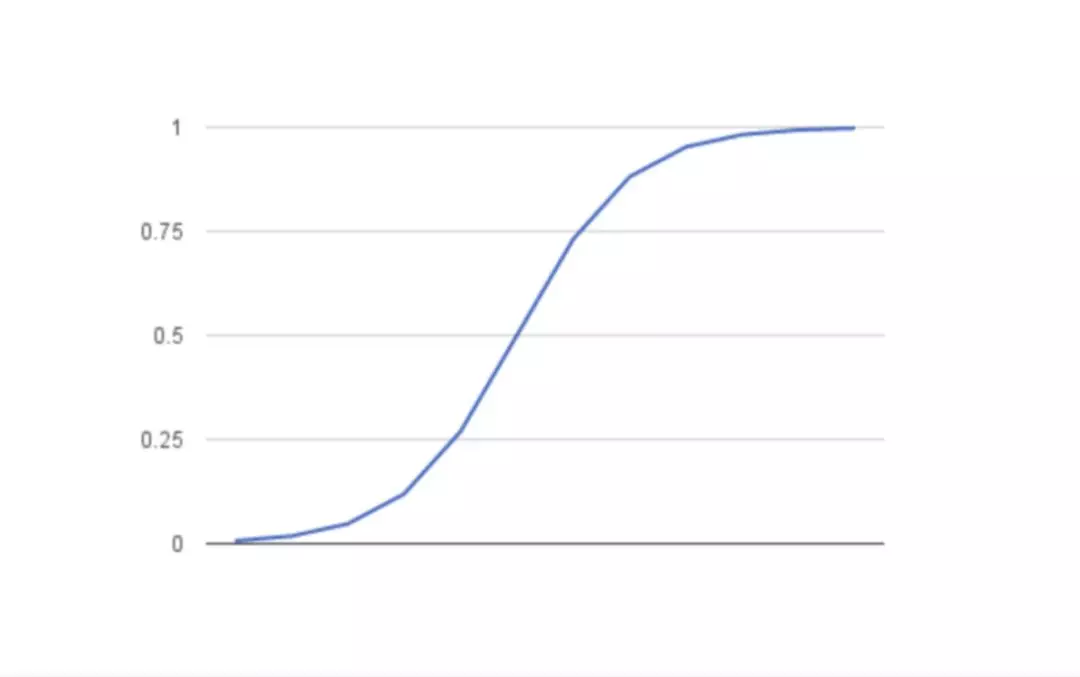
Logistic 回归通过使用其固有的 logistic 函数估计概率，来衡量因变量（我们想要预测的标签）与一个或多个自变量（特征）之间的关系。

然后这些概率必须二值化才能真地进行预测。这就是 logistic 函数的任务，也称为 Sigmoid 函数。Sigmoid 函数是一个 S 形曲线，它可以将任意实数值映射到介于 0 和 1 之间的值，但并不能取到 0或1。然后使用阈值分类器将 0 和 1 之间的值转换为 0 或 1。

下面的图片说明了 logistic 回归得出预测所需的所有步骤。



下面是 logistic 函数（sigmoid 函数）的图形表示：



我们希望随机数据点被正确分类的概率最大化，这就是最大似然估计。最大似然估计是统计模型中估计参数的通用方法。

你可以使用不同的方法（如优化算法）来最大化概率。牛顿法也是其中一种，可用于查找许多不同函数的最大值（或最小值），包括似然函数。也可以用梯度下降法代替牛顿法。

Logistic 回归 vs 线性回归

你可能会好奇：logistic 回归和线性回归之间的区别是什么。逻辑回归得到一个离散的结果，但线性回归得到一个连续的结果。预测房价的模型算是返回连续结果的一个好例子。该值根据房子大小或位置等参数的变化而变化。离散的结果总是一件事（你有癌症）或另一个（你没有癌症）。

优缺点

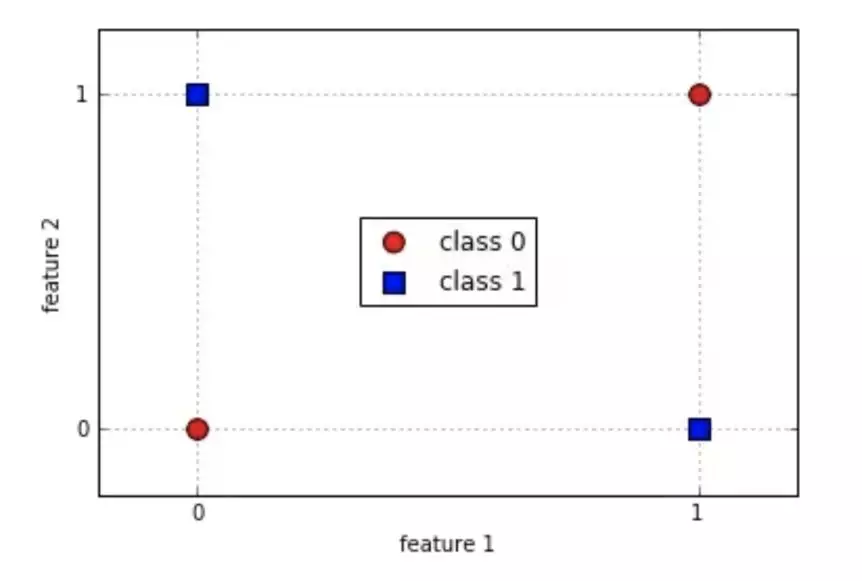
Logistic 回归是一种被人们广泛使用的算法，因为它非常高效，不需要太大的计算量，又通俗易懂，不需要缩放输入特征，不需要任何调整，且很容易调整，并且输出校准好的预测概率。

与线性回归一样，当你去掉与输出变量无关的属性以及相似度高的属性时，logistic 回归效果确实会更好。因此特征处理在 Logistic 和线性回归的性能方面起着重要的作用。

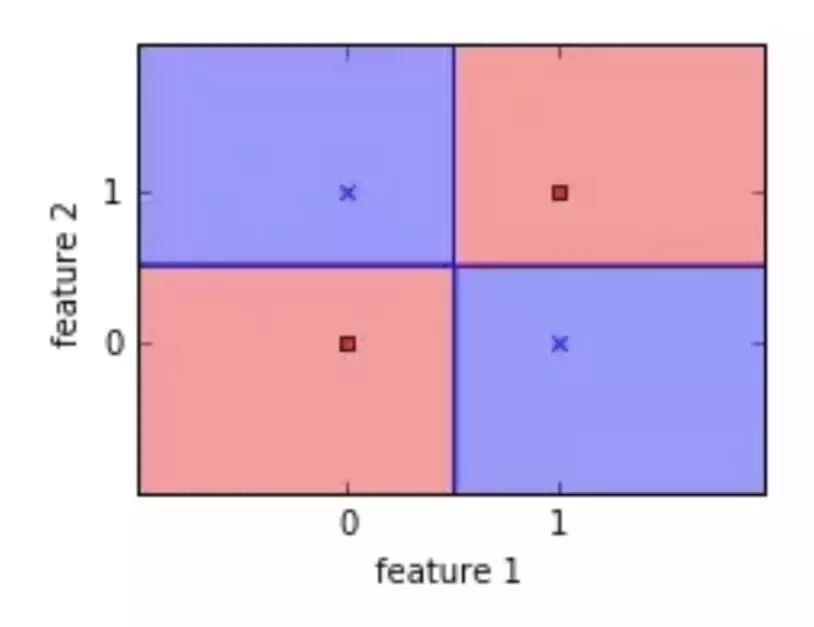
Logistic 回归的另一个优点是它非常容易实现，且训练起来很高效。在研究中，我通常以 Logistic 回归模型作为基准，再尝试使用更复杂的算法。

由于其简单且可快速实现的原因，Logistic 回归也是一个很好的基准，你可以用它来衡量其他更复杂的算法的性能。

它的一个缺点就是我们不能用 logistic 回归来解决非线性问题，因为它的决策边界是线性的。我们来看看下面的例子，两个类各有俩实例。



显然，我们不可能在不出错的情况下划出一条直线来区分这两个类。使用简单的决策树是个更好的选择。

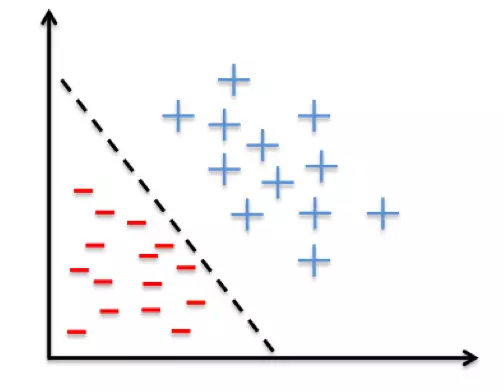


Logistic 回归并非最强大的算法之一，它可以很容易地被更为复杂的算法所超越，另一个缺点是它高度依赖正确的数据表示。

这意味着逻辑回归在你已经确定了所有重要的自变量之前还不会成为一个有用的工具。由于其结果是离散的，Logistic 回归只能预测分类结果。它同时也以其容易过拟合而闻名。

何时适用

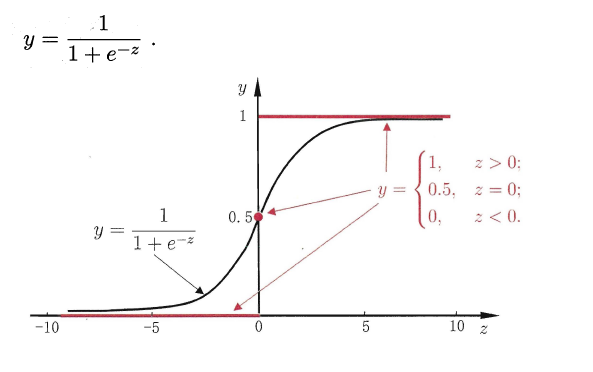
就像我已经提到的那样，Logistic 回归通过线性边界将你的输入分成两个「区域」，每个类别划分一个区域。因此，你的数据应当是线性可分的，如下图所示的数据点：

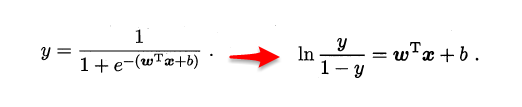


换句话说：当 Y 变量只有两个值时（例如，当你面临分类问题时），您应该考虑使用逻辑回归。注意，你也可以将 Logistic 回归用于多类别分类。

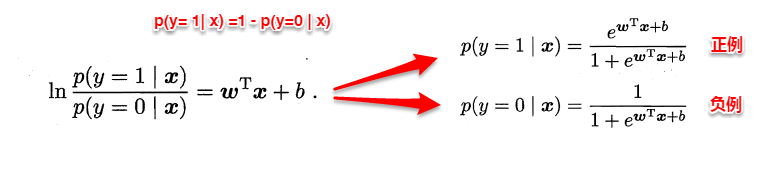
3.2.2 逻辑回归原理以及Loss Function

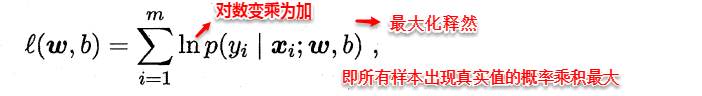
回归就是通过输入的属性值得到一个预测值，利用上述广义线性模型的特征，是否可以通过一个联系函数，将预测值转化为离散值从而进行分类呢？线性几率回归正是研究这样的问题。对数几率引入了一个对数几率函数（logistic function）,将预测值投影到0或1，从而将线性回归问题转化为二分类问题。

[](https://camo.githubusercontent.com/915ac9417380f692b629ff0122c216c98d6db954/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373232623063373734382e706e67)

[](https://camo.githubusercontent.com/c26adeb0b49eadf2822d3e98630c41e7628b4db4/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373232623061363535642e706e67)

若将y看做样本为正例的概率，（1-y）看做样本为反例的概率，则上式实际上使用线性回归模型的预测结果器逼近真实标记的对数几率。因此这个模型称为“对数几率回归”（logistic regression），也有一些书籍称之为“逻辑回归”。下面使用最大似然估计的方法来计算出w和b两个参数的取值，下面只列出求解的思路，不列出具体的计算过程。

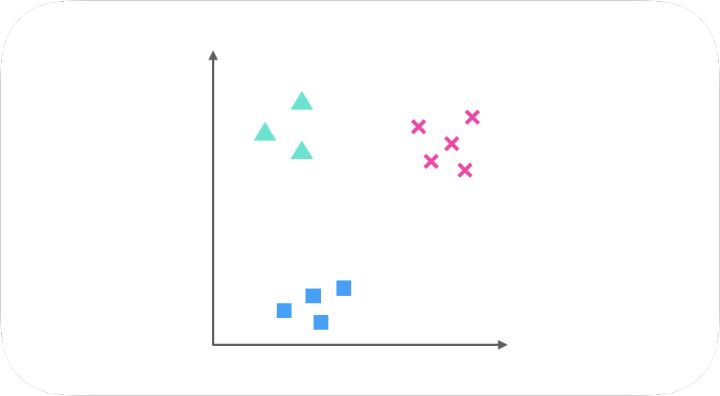
[](https://camo.githubusercontent.com/77590bb23fb0d46091e4f473f862d5d71ded2432/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373233623832346630632e706e67)

[](https://camo.githubusercontent.com/afa702e74ac57f93c0667df5d3d164ec66c39eca/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373233623831373936312e706e67)

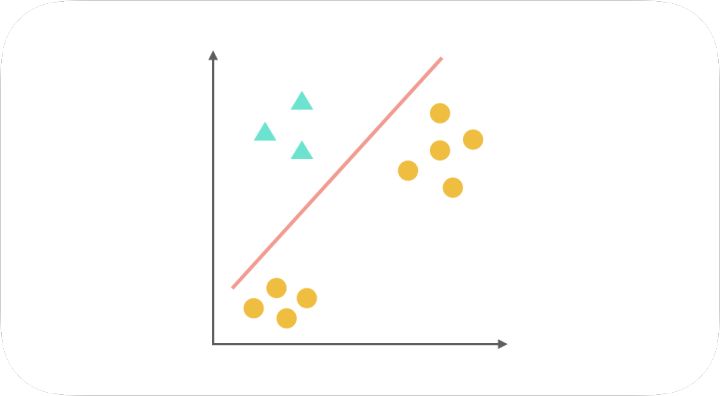
3.2.3 逻辑回归实现多分类

## One-Vs-All

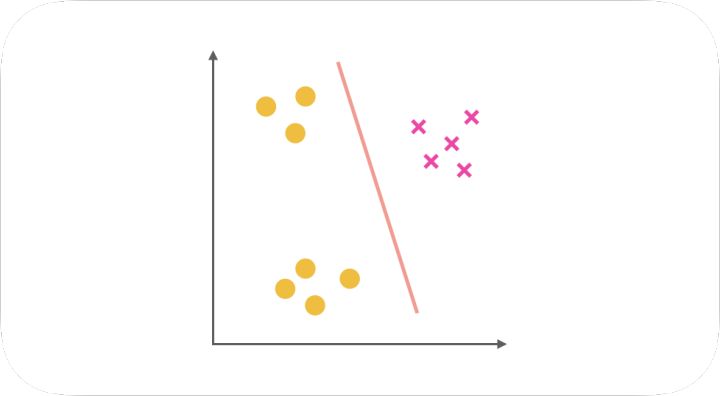
假设我们要解决一个分类问题，该分类问题有三个类别，分别用△，□和×表示，每个实例（Entity）有两个属性（Attribute），如果把属性 1 作为 X 轴，属性 2 作为 Y 轴，训练集（Training Dataset）的分布可以表示为下图：



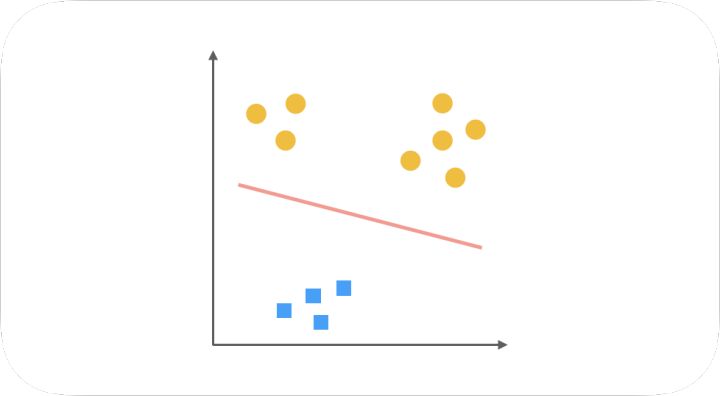
One-Vs-All（或者叫 One-Vs-Rest）的思想是把一个多分类的问题变成多个二分类的问题。转变的思路就如同方法名称描述的那样，选择其中一个类别为正类（Positive），使其他所有类别为负类（Negative）。比如第一步，我们可以将三角形所代表的实例全部视为正类，其他实例全部视为负类，得到的分类器如图：



同理我们把 X 视为正类，其他视为负类，可以得到第二个分类器：



最后，第三个分类器是把正方形视为正类，其余视为负类：



对于一个三分类问题，我们最终得到 3 个二元分类器。在预测阶段，每个分类器可以根据测试样本，得到当前正类的概率。即 P(y = i | x; θ)，i = 1, 2, 3。选择计算结果最高的分类器，其正类就可以作为预测结果。

One-Vs-All 最为一种常用的二分类拓展方法，其优缺点也十分明显。

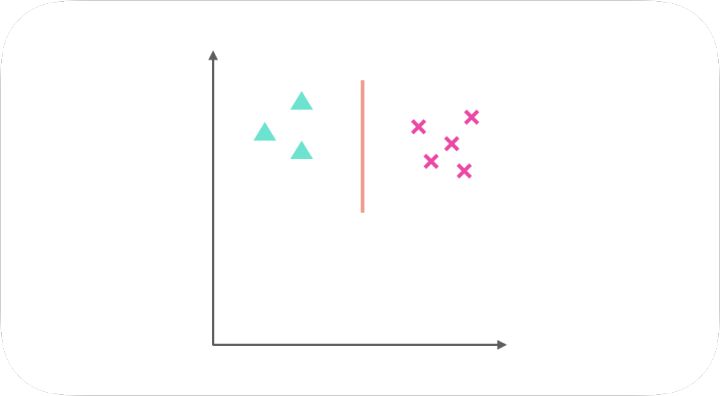
优点：普适性还比较广，可以应用于能输出值或者概率的分类器，同时效率相对较好，有多少个类别就训练多少个分类器。

缺点：很容易造成训练集样本数量的不平衡（Unbalance），尤其在类别较多的情况下，经常容易出现正类样本的数量远远不及负类样本的数量，这样就会造成分类器的偏向性。

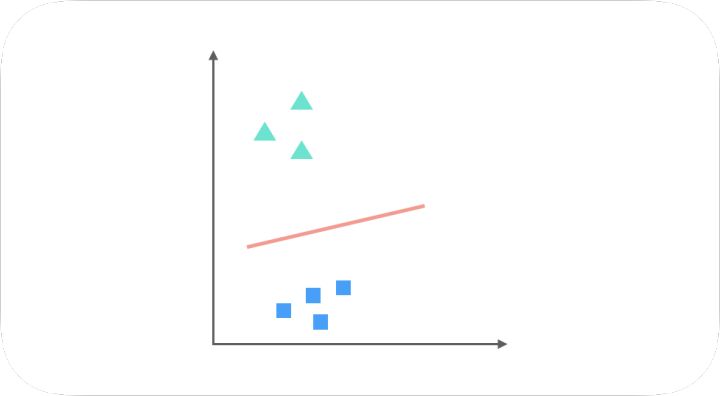
## One-Vs-One

相比于 One-Vs-All 由于样本数量可能的偏向性带来的不稳定性，One-Vs-One 是一种相对稳健的扩展方法。对于同样的三分类问题，我们像举行车轮作战一样让不同类别的数据两两组合训练分类器，可以得到 3 个二元分类器。

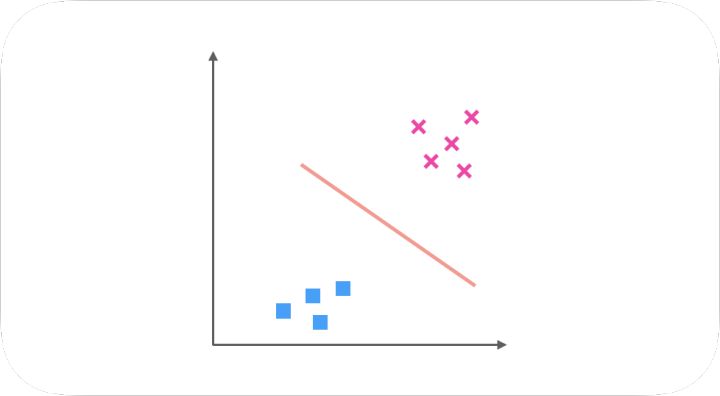
它们分别是三角形与 x 训练得出的分类器：



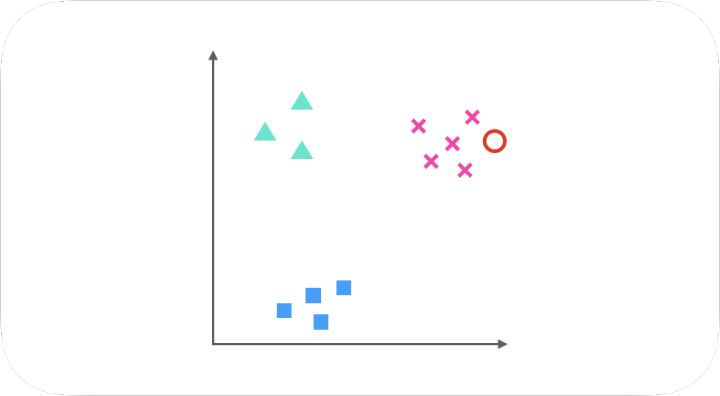
三角形与正方形训练的出的分类器：



以及正方形与 x 训练得出的分类器：



假如我们要预测的一个数据在图中红色圆圈的位置，那么第一个分类器会认为它是 x，第二个分类器会认为它偏向三角形，第三个分类器会认为它是 x，经过三个分类器的投票之后，可以预测红色圆圈所代表的数据的类别为 x。



任何一个测试样本都可以通过分类器的投票选举出预测结果，这就是 One-Vs-One 的运行方式。

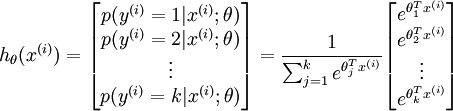
当然这一方法也有显著的优缺点，其缺点是训练出更多的 Classifier，会影响预测时间。

虽然在本文的例子中，One-Vs-All 和 One-Vs-One 都得到三个分类器，但实际上仔细思考就会发现，如果有 k 个不同的类别，对于 One-Vs-All 来说，一共只需要训练 k 个分类器，而 One-Vs-One 则需训练 C(k, 2) 个分类器，只是因为在本例种，k = 3 时恰好两个值相等，一旦 k 值增多，One-Vs-One 需要训练的分类器数量会大大增多。

当然 One-Vs-One 的优点也很明显，它在一定程度上规避了数据集 unbalance 的情况，性能相对稳定，并且需要训练的模型数虽然增多，但是每次训练时训练集的数量都降低很多，其训练效率会提高。

## Softmax

在二元的逻辑回归模型中，我们用 Sigmoid 函数将一个多维数据（一个样本）映射到一个 0 - 1 之间的数值上，有没有什么方法从数学上让一个样本映射到多个 0 - 1 之间的数值呢？答案是通过 Softmax 函数。



https://pic2.zhimg.com/80/v2-231ed6aa6d3f5ab5cac753fd7e710341_hd.jpg

使所有概率之和为 1，是对概率分布进行归一化。

为什么选用指数函数呢？有一些简单的理由：

1. 指数函数简单，并且是非线性的
2. 该函数严格递增
3. 这是一个凸函数

定义了新的假设函数（hypothesis function）之后，我们要得到其对应的代价函数（cost function）。

https://pic3.zhimg.com/80/v2-ab690c0473c9056d1024316661bd89da_hd.jpg

其中的取值规则为大括号内的表达式值为真时，取 1，为假时取 0。

对该代价函数求最优解同样可以使用如梯度下降之类的迭代算法，其梯度公式如下：

https://pic3.zhimg.com/80/v2-0844408e1498b558fa8b3edd4ed24326_hd.jpg

有了偏导数，就可以对代价函数进行优化，最终求解。

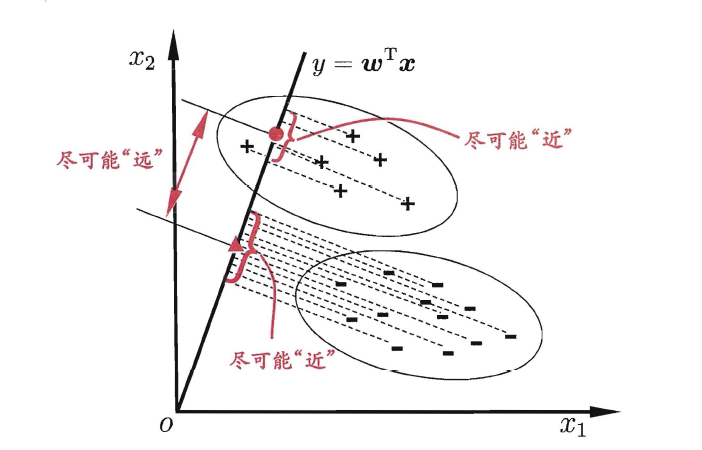
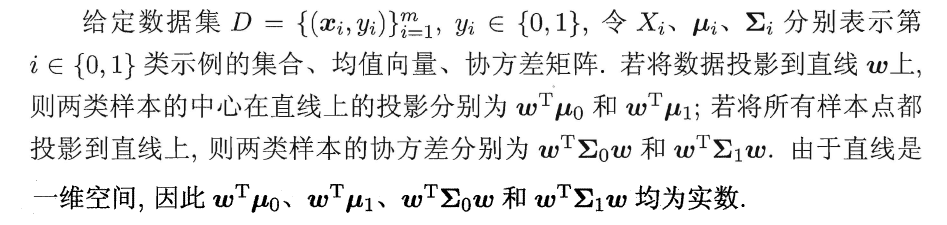
本质上讲，Softmax 回归就是 logistic 回归进行多分类时的一种数学拓展，如果让分类数为 2 带入 softmax 回归，会发现其本质上和 logistic 回归是一样的。

在处理一些样本可能丛属多个类别的分类问题是，使用 one vs one 或 one vs all 有可能达到更好的效果。

Softmax 回归适合处理一个样本尽可能属于一种类别的多分类问题。

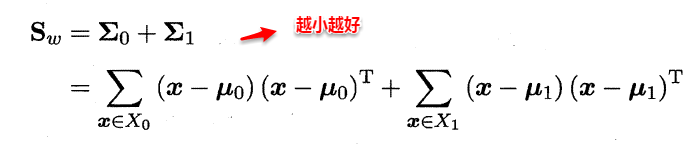
3.3 线性判别分析

线性判别分析（Linear Discriminant Analysis，简称LDA）,其基本思想是：将训练样本投影到一条直线上，使得同类的样例尽可能近，不同类的样例尽可能远。如图所示：

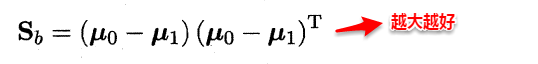
[](https://camo.githubusercontent.com/9493cdaad8be4e794f535d3d63dc51082ab2f5d6/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373233623836336562622e706e67)[](https://camo.githubusercontent.com/62cf12d63c27fac79d395dce64ba648f5725c725/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373233623835626661392e706e67)

想让同类样本点的投影点尽可能接近，不同类样本点投影之间尽可能远，即：让各类的协方差之和尽可能小，不用类之间中心的距离尽可能大。基于这样的考虑，LDA定义了两个散度矩阵。

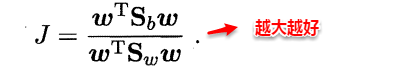
类内散度矩阵（within-class scatter matrix）

[](https://camo.githubusercontent.com/6fcccb86848bf317c7c15f9733cc0bc1eaad7fed/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373233623831353665312e706e67)

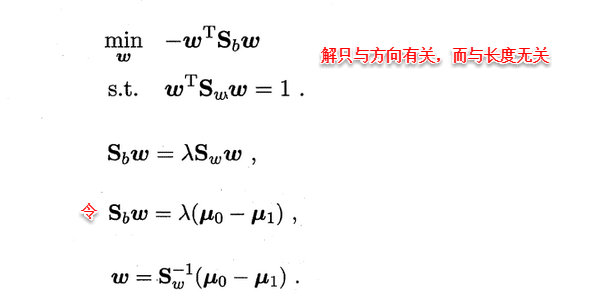
类间散度矩阵(between-class scaltter matrix)

[](https://camo.githubusercontent.com/936f085bd8cfefa919dec271d00897821a13d920/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373233623765396462332e706e67)

因此得到了LDA的最大化目标：“广义瑞利商”（generalized Rayleigh quotient）。

[](https://camo.githubusercontent.com/d49289ddd0dfcf0269db497bf89374f5753e2ee9/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373233623765386136312e706e67)

从而分类问题转化为最优化求解w的问题，当求解出w后，对新的样本进行分类时，只需将该样本点投影到这条直线上，根据与各个类别的中心值进行比较，从而判定出新样本与哪个类别距离最近。求解w的方法如下所示，使用的方法为λ乘子(拉格朗日乘子法)。

[](https://camo.githubusercontent.com/265922e13352fac4dd6a196448a28434a9bcc3b6/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373233623833643565302e706e67)

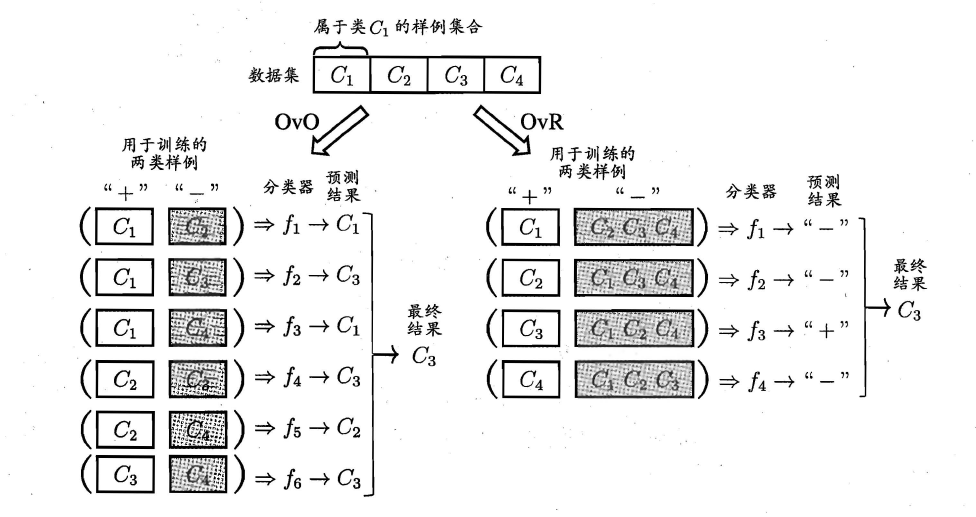
若将w看做一个投影矩阵，类似PCA的思想，则LDA可将样本投影到N-1维空间（N为类簇数），投影的过程使用了类别信息（标记信息），因此LDA也常被视为一种经典的监督降维技术。  
​  
3.4 多分类学习

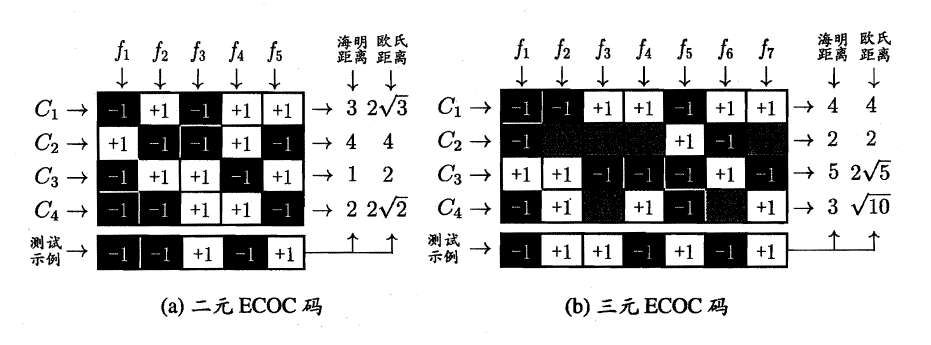
现实中我们经常遇到不只两个类别的分类问题，即多分类问题，在这种情形下，我们常常运用“拆分”的策略，通过多个二分类学习器来解决多分类问题，即将多分类问题拆解为多个二分类问题，训练出多个二分类学习器，最后将多个分类结果进行集成得出结论。最为经典的拆分策略有三种：“一对一”（OvO）、“一对其余”（OvR）和“多对多”（MvM），核心思想与示意图如下所示。

OvO：给定数据集D，假定其中有N个真实类别，将这N个类别进行两两配对（一个正类/一个反类），从而产生N（N-1）/2个二分类学习器，在测试阶段，将新样本放入所有的二分类学习器中测试，得出N（N-1）个结果，最终通过投票产生最终的分类结果。

OvM：给定数据集D，假定其中有N个真实类别，每次取出一个类作为正类，剩余的所有类别作为一个新的反类，从而产生N个二分类学习器，在测试阶段，得出N个结果，若仅有一个学习器预测为正类，则对应的类标作为最终分类结果。

MvM：给定数据集D，假定其中有N个真实类别，每次取若干个类作为正类，若干个类作为反类（通过ECOC码给出，编码），若进行了M次划分，则生成了M个二分类学习器，在测试阶段（解码），得出M个结果组成一个新的码，最终通过计算海明/欧式距离选择距离最小的类别作为最终分类结果。

[](https://camo.githubusercontent.com/ac256a5852b64de313b885410e5a74b41737b72f/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373233623836326266622e706e67)

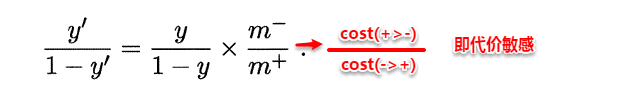
[](https://camo.githubusercontent.com/c9f746bbf30ad6cc8c18900ba1b2cbabd5254d32/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373233623833303064352e706e67)

3.5 类别不平衡问题

类别不平衡（class-imbanlance）就是指分类问题中不同类别的训练样本相差悬殊的情况，例如正例有900个，而反例只有100个，这个时候我们就需要进行相应的处理来平衡这个问题。常见的做法有三种：

在训练样本较多的类别中进行“欠采样”（undersampling）, 这种做法会随机地丢失一些信息。比如从正例中采出100个，常见的算法有：EasyEnsemble（利用集成学习机制，将反例划分为若干个集合供不同学习器使用，这样对每个学习器来看都进行了欠采样，但全局来看不会丢失重要信息）。

在训练样本较少的类别中进行“过采样”（oversampling）,例如通过对反例中的数据进行插值，来产生额外的反例，常见的算法有SMOTE。过采样不能不能简单地对初始正例样本进行重复采样，否则会导致过拟合。

直接基于原数据集进行学习，对预测值进行“再缩放”处理。其中再缩放也是代价敏感学习的基础。[](https://camo.githubusercontent.com/a06f45e443d0de6364532d2a14244c344a2cc74e/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373236666538376165322e706e67)