十一、Linear Models for Classification 用于分类的线性模型

11.1 Linear Models for Binary Classification 用于二元分类的线性模型

目前叙述的算法模型主要有3类：线性二元分类，线性回归，logistic回归，这三个模型的最主要的相同点在假设函数和错误函数中都出现了线性得分函数（linear scoring function），如公式11-1所示。

$$s = w^T x$$

三类模型与得分s之间的关系如图11-1所示。

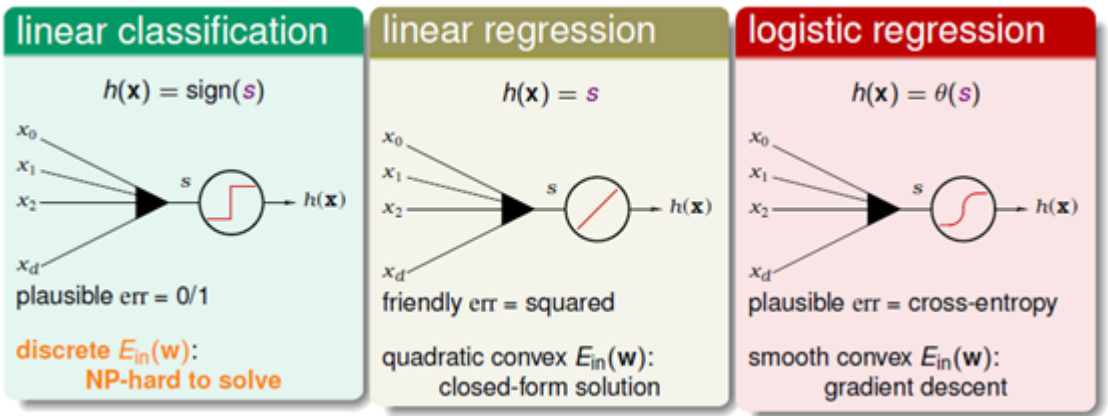

图11-1 三类模型与得分s的关系

最左为线性二元分类，其假设函数为 $h(\mathbf{x}) = \text{sign}(s)$，一般使用0/1错误，通过 $E_{in}(\mathbf{w})$ 求解最优权值向量w比较困难；中间为线性回归模型，其假设函数为 $h(\mathbf{x}) = s$，一般使用平方错误，可直接通过解析解求解最优w；最右为logistic回归模型，假设函数为 $h(\mathbf{x}) = \theta(s)$，使用交叉熵错误，通过梯度下降法求出近似的w。

从上述分析不难看出，线性二元分类问题的求解方式最为困难，但与另外两种模型存在着共同点——得分s，能否利用这两种模型的算法近似求得二分类问题的最优w呢？

回顾10.2节，logistic回归的错误，可用符号 err_{CE} 表示，其中CE为交叉熵（cross-entropy ）的缩写，可以写成公式11-2所示。

$$err_{CE} = err(w, x, y) = err_{CE}(s, y) = \ln(1 + \exp(-yw^T)) = ln(1 + \exp(-ys))$$

是否二元分类模型和线性回归模型的错误函数可以写成关于 ys 的形式？答案是可以的，如图11-2所示。

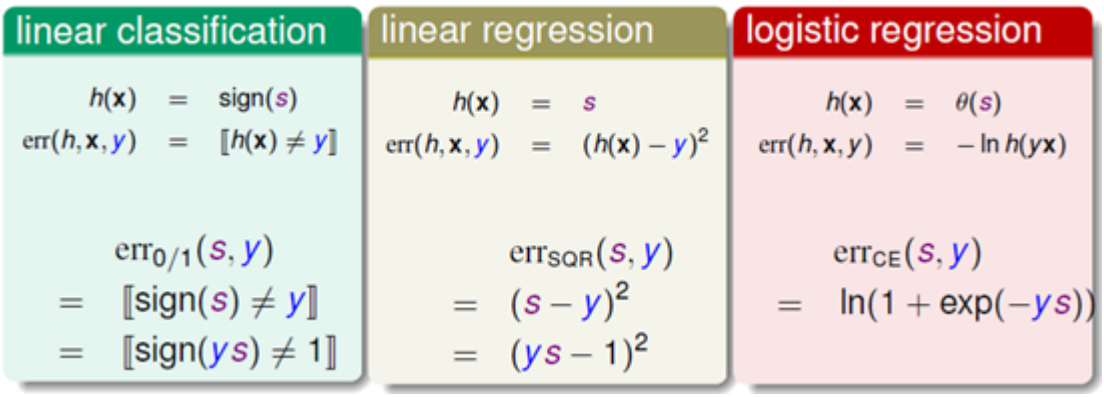

图11-2 三类模型的错误函数

二元分类模型和线性回归模型错误函数中的转换都用到了$y \in \{+1, -1\}$的性质。接着观察三类模型的错误函数与ys之间的关系。本节开头回顾了s的物理意义为得分，此处ys的物理意义是正确的得分，因此ys越大越好，表示两者接近且同号。

根据图11-2中的三类模型的错误函数有关ys的公式，可以得出如图11-3所示的关系图。

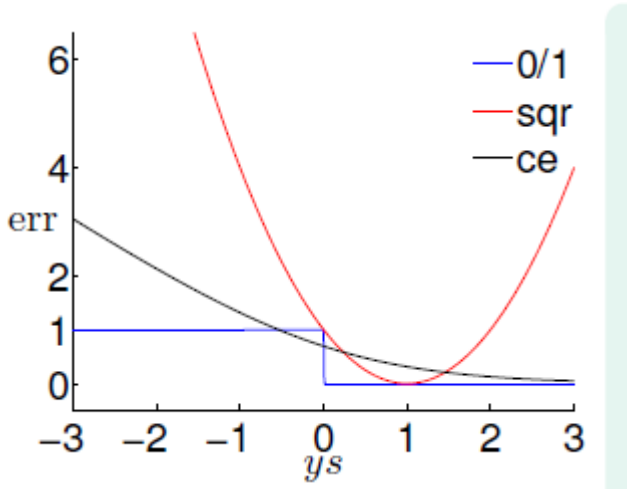

图11-3 三类模型的错误函数与ys的关系图

其中蓝色的折线表示0/1错误$err_{0/1}$，在ys大于0时，$err_{0/1} = 0$，反之$err_{0/1} = 1$；红色的抛物线表示平方错误srr_{SQR}，在$ys \ll 1$时与$err_{0/1}$在该范围内所表现出的特征相似，但是在$ys \gg 1$时与在该范围内所表达的效果相去甚远，因此只有在err_{SQR}很小的情况下，可以使用err_{SQR}取代$err_{0/1}$；墨绿的曲线表示err_{CE}，同样如图11-3所示也只有在err_{CE}很小的情况下，err_{CE}和$err_{0/1}$可互相取代。但是err_{CE}跟想得到的错误曲线还有一些差距，因此略做转变，得到公式11-3。

$$err_{SCE} = log_2(1 + \exp(-ys))$$

其中err_{SCE}表示缩放的（scaled）err_{CE}，即对srr_{CE}做了一个换底，因此可以得到图11-4。

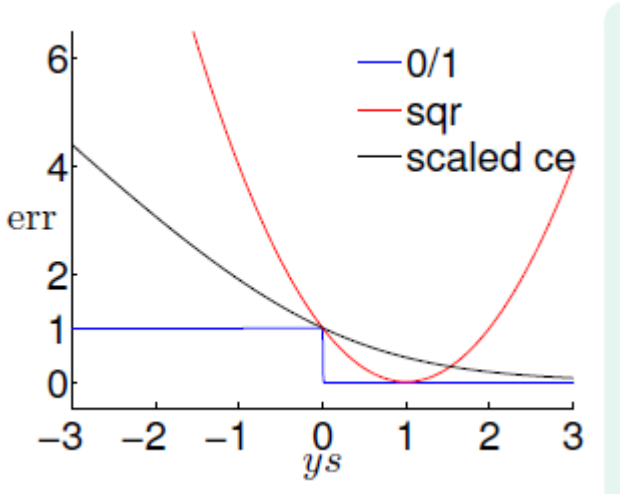

图11-4err_{SCE}关于ys的图

如图11-4中墨绿色的线表示err_{SCE}，从图中可以看出，该错误函数很适合做$err_{0/1}$的上限，在err_{SCE}很小的情况下，err_{SCE}和$err_{0/1}$可互相取代，如公式11-4所示。

$$err_{0/1}(s,y) \leq err_{SCE}(s,y) = \frac{1}{\ln 2} err_{CE}(s,y)$$

通过公式11-4可以得出$E_{in}^{0/1}$和$E_{out}^{0/1}$的上限，如公式11-5和公式11-6所示。

$$E_{in}^{0/1} \leq E_{in}^{SCE}(w) = \frac{1}{\ln 2} E_{in}^{CE}(w)$$

$$E_{out}^{0/1} \leq E_{out}^{SCE}(w) = \frac{1}{\ln 2} E_{out}^{CE}(w)$$

再通过VC限制理论可以得到公式11-7。

$$E_{out}^{0/1} \leq E_{out}^{SCE}(w) + \Omega^{0/1} = \frac{1}{\ln 2} E_{out}^{CE}(w) + \Omega^{0/1}$$

第一个不等号连接的是在VC限制下$E_{out}^{0/1}(w)$和其上界，概念见7.4节, 其中$\Omega^{0/1}$函数是在7.4节中提到过的模型复杂度，在二元分类中可以写成的形$\sqrt{\frac{8}{N}\ln(\frac{4(2N)^{d_{VC}}}{\delta})}$式。

因此得到如下结论：小的$E_{out}^{0/1}(w)$可以通过小的$E_{in}^{CE}(w)$得出。同理可以证明小的$E_{out}^{0/1}(w)$也可以通过小的$E_{in}^{SQR}(w)$得出，即线性回归模型和logistic回归模型可以用作二元分类。

算法流程一般是在输出空间$y \in \{-1, +1\}$ 的情况下，通过线性回归和logistic回归相对应的求解方法求出最优w_{REG}；

将求得的w_{REG}代入公式sign，得到最优假设函数 $g(x) = sign(w_{REG}^T x)$。

三类模型做分类的利弊分析如表11-1所示。

表11-1 三类模型做分类的利弊分析

	二元分类	线性回归	Logistic回归
好处	在线性可分的情况下可以保证完成	最容易的优化算法	容易的优化算法
坏处	在线性不可分的情况，需要使用启发式pocket	在$\|ys\|$非常大时，相对于$err_{0/1}$是一个很宽松的上界	在ys为负时，是一个宽松的上界

线性回归一般只作为PLA、pocket、logistic回归的初始向量w_0；logistic回归经常取代pocket算法。

11.2 Stochastic Gradient Descent　随机梯度下降

如公式11-8为迭代优化算法的通式，学过的PLA的迭代算法如公式11-9，logistic回归中梯度下降的迭代公式如公式11-10。

$$w_{t+1} = w_t + \eta \cdot v$$

$$w_{t+1} = w_t + y_{n(t)} \cdot x_{n(t)}$$

$$w_{t+1} = w_t - \eta \nabla E_{in}(w_t) = w_t - \eta(\frac{1}{N}\sum_{n=1}^{N}\theta(-y_n w_t^T x_n)(-y_n x_n)) = w_t + \eta(\frac{1}{N}\sum_{n=1}^{N}\theta(-y_n w_t^T x_n)(y_n x_n)$$

对比以上两种迭代优化方法：PLA与logistic回归的梯度下降。发现PLA只需要通过一个样本点便可计算出w_{t+1}，即每次迭代的时间复杂度为$O(1)$；logistic回归的梯度下降需要遍历所有的样本点才能计算出w_{t+1}，即每次迭代的时间复杂度为$O(n)$。有无可能将logistic回归每次迭代时间复杂度降为$O(1)$？

观察公式11-10，方向向量v，v≈$-\nabla E_{in}(w_t)$，该梯度是通过所有的样本点加权求和再取平均得到的，如何使用一个样本点的取值近似整体的平均值？

可以将求平均的过程理解为求期望值，此处使用在N个样本中随机抽取一个样本点求出的梯度取代原来的期望梯度，这种随机选取的梯度称为随机梯度（stochastic gradient），可用符号$\nabla_w err(w,x,y)$表示，而真实的梯度与随机梯度的关系如公式11-11。

$$\nabla E_{in}(w_t) = \varepsilon_{random \quad n} \nabla_w err(w, x_n, y_n)$$

随机梯度值可以看做真实的梯度值加上一个噪音，使用随机梯度取代真实梯度做梯度下降的算法称作随机梯度下降（stochastic gradient descent），简称SGD。这种替代的理论基础是在迭代次数足够多的情况下，平均的随机梯度和平均的真实梯度相差不大。

该算法的优点是简单，容易计算，适用于大数据或者流式数据；缺点是不稳定。

Logistic回归的随机梯度下降的迭代如公式11-12所示。

$$w_{t+1} = w_t + \underbrace{\eta(-y_n w_t^T x_n)(y_n x_n)}_{-\nabla_w err(w, x_n, y_n)}$$

是否联想到了其他的迭代算法？PLA，如公式11-13所示。

$$w_{t+1} = w_t + \underbrace{1}_{\eta}\underbrace{[|sign(w_t^T x_n) \neq y_n|]y_n x_n}_{v}$$

因此logistic回归随机梯度下降类似于"软"的PLA，为什么称为软的？原因是它的$y_n x_n$之前的权值并没有那么绝对不是1就是0，而是一个在0~1之间的值。在公式11-12中，如果$\eta = 1$且$w_t^T x_n$始终是一个很大的值，则logistic回归随机梯度下降相当于是PLA。

11.3 Multiclass via Logistic Regression　通过logistic回归实现多类别分类

多类别分类有许多应用场景，特别是在识别（recognition）领域。

如图11-5为，输出空间y为四类别的情况，即$y \in \{\square, \diamond, \triangle, \star\}$。

图11-5 四分类问题

实际多类别问题也可以使用二元分类问题$\{\bigcirc, \times\}$的思路进行分类，如将原四类问题分解为是否为$\square$，即将$\square$与其他的类别分离，生成一个新的二元分类问题，即$\{\square = \bigcirc, \diamond = \times, \triangle = \times, \star = \times\}$，通过此方式得到一个分类超平面，如图11-6所示。

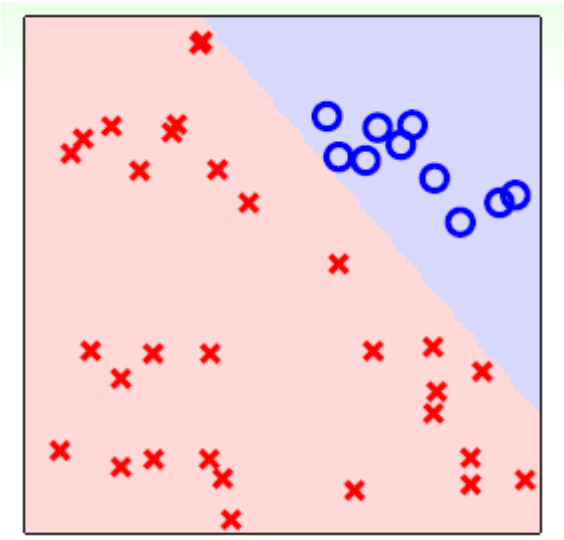

图11-6 以是否为$\square$进行二元分类

同理可以以是否为$\diamond$生成一个新的二元分类问题，即$\{\square = \times, \diamond = \bigcirc, \triangle = \times, \star = \times\}$，该分类超平面如图11-7所示。

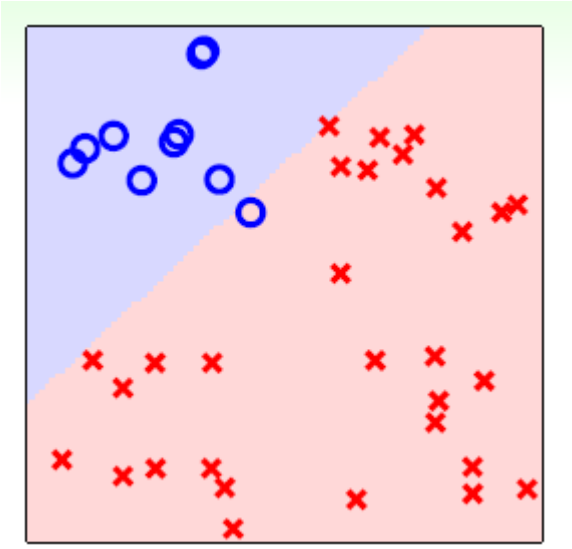

图11-7 以是否为◊进行二元分类

另外两种情况就不一一列举，最终以是否为每个类别得到的二元分类如图11-8。

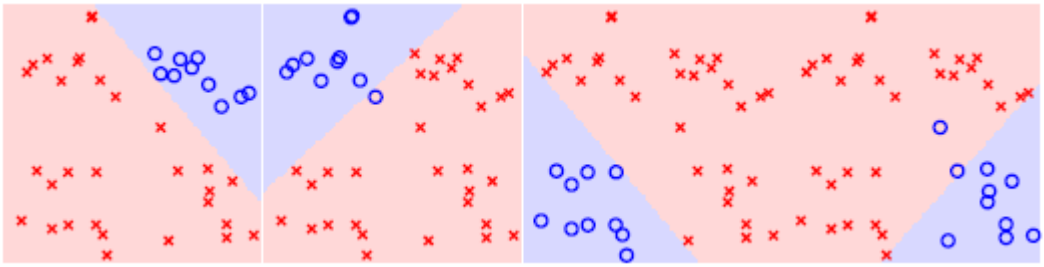

图11-8 四个类别各自的二元分类情况

当将图11-8的四种情况合并在一个图中会发现有一些无法处理的情形，如图11-9所示。

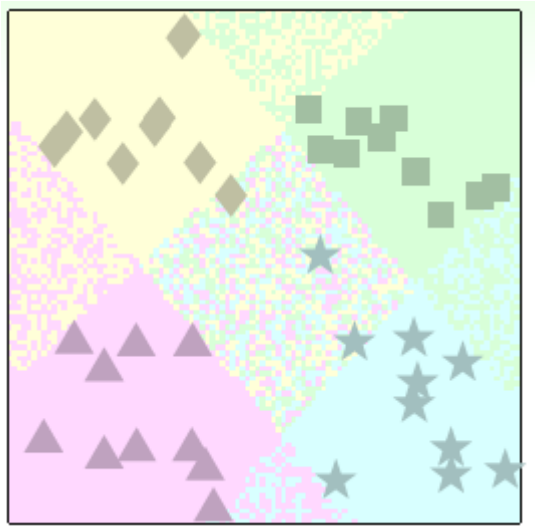

图11-8 四种情况合并图

其中四个边缘的三角阴影所在的区域为相邻两个类别都争夺的区域，如最上方的三角区域是类别□和类别◊重叠的区域；还有图正中的区域又不属于任何类别。这些问题如何解决？

使用以前学过的软性分类，还是关于类别□的二元分类问题，此处不再使用硬划分，而是使用该样本点是□的可能性，即 $P(□|\mathbf{x})$ ，如图11-9所示。

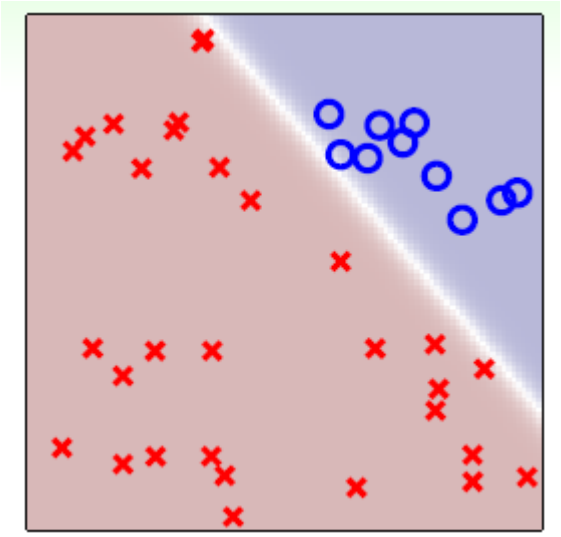

图11-9 关于类别□的软化分

余下三种情况不再一一举例，最终得到的四种类别的分类情况和合并后的情况分别如图11-10和图11-11所示。

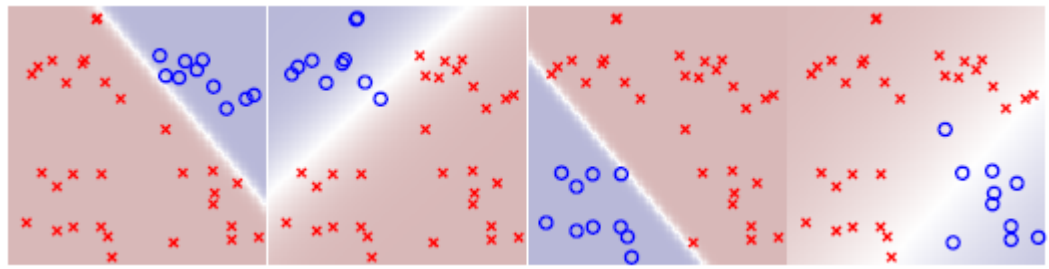

图11-10四个类别各自的软二元分类情况

图11-11四个类别软二元分类合并后情况

如何判断样本点属于哪个类别，可以分别计算样本点在四种软二元分类情况下概率，选择其中概率最大的一个作为所属类别，如公式11-14所示。

$$g(x) = arg\max_{k \in \gamma} \theta(w_{[k]}^T x)$$

其中求概率的公式使用logistic函数θ，k表示类别，注意到logistic函数是一个单调函数，因此可以消去该函数，直接使用个类别的得分值作比较，如公式11-5所示。

$$g(x) = arg\max_{k\in\gamma} w_{[k]}^T x$$

用此种思路设计的算法称作一对多（One Versue All），简称为OVA，表示一个类别对其他所有类别，算法流程如下：

在整个训练数据集D上，$D_{[k]} = \{(x_n, y_n = 2[|y_n = k|] - 1)\}_{n=1}^N$（在y=k时为+1，y≠k时为-1，符号$[|\cdot|]$取1或者0），使用logistic函数计算各个类别的权值向量$w_{[k]}$；

返回假设函数g，$g(x) = arg\max_{k\in\gamma} w_{[k]}^T x$。

该算法的优点是简单有效，易于类似于logistic函数的二元分类问题扩展成多类别分类；缺点是当类别特别多时，产生了不平衡的现象（如类别特别多，则+1的数据量就很少，大部分都是-1，数据量严重不平衡）。

11.4 Multiclass via Binary Classification 通过二元分类实现多类别分类

上一节的最后提到OVA的方式在类别非常多的情况下，出现了训练数据严重失衡的现象，于是本节介绍一种应对这类不平衡问题的方法。

还是上节中使用的四分类问题，不像OVA在整个数据集中计算是否为□的权值向量w，此种方法是任意选择四类中的两类，如类别□和类别◊，将两个类别分别设为+1和-1，形式如$\{□=○, ◊=×, \Delta = nil, ☆ = nil\}$，在包含两类的数据集上计算权值向量w，如图11-12。

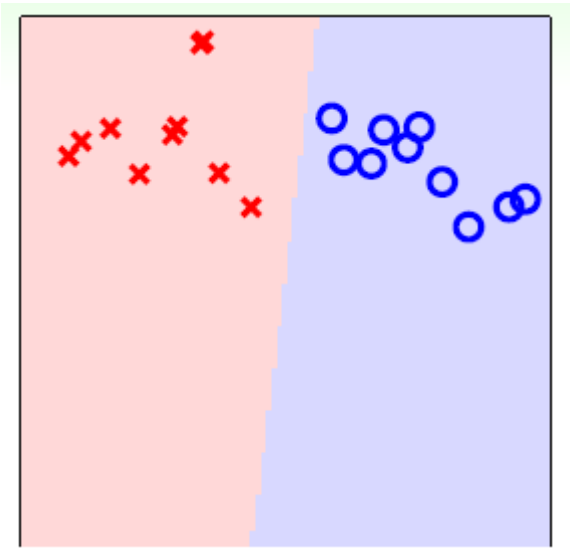

图11-12类别□和类别◊的二分类

如上述情况相同，从四种类别中选取两种做二元分类，一共可得6种对比（$C_4^2 = 6$）,各对比如图11-13所示。

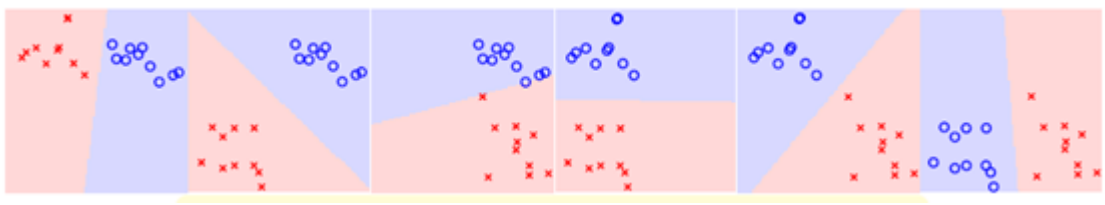

图11-13 6种对比情况

如图11-13得到6个不同的权值向量w，如何判断某新进样本属于哪个分类？如11-14中紫色的样本点在6中情况下所属的类别，前三种属于□，第4种属于◇，后两种属于☆，只需要找出在所有对比中胜利次数最多的类别，因此该点属于□。这种方式如同投票选举，样本点属于所有类别对比中赢得次数最多的那种类别。

这种分类方式称为一对一（one vervuse one），简称OVO。其算法流程如下：

所有类别的任意两个类别做对比，使用二元分类算法，在数据集D，

$D_{[k,l]}=\left\{\left(x_n,y_n^{'}=2[\![y_n=k]\!]-1\right):y_n=k\ or\ y_n=l\right\}$ 求出最佳的权值向量$\mathbf{w}_{[k,l]}$；

通过投票返回假设函数g。

其优点是简单有效，在做两两对比时，每次使用的不是全部训练数据，而是仅属于当前两类的训练数据，能将所有类似于二元分类的算法扩展成多元分类问题； 缺点是对比次数是C_K^2，即$O(\mathbf{K}^2)$，其中K表示类别数，因此就需要花费更多的存储空间、计算时间。