第5节 非线性分类问题

——软分隔及多类分类

5.4 软分隔方法

- 改进：加入松弛变量 ζ_i 和惩罚因子 C

- 松弛变量允许实际分类中一定的不准确性的存在，引入松弛变量后原先的约束条件变为：

$$y_i \cdot (W^{\mathrm{T}} \cdot X_i + b) \geq 1 - \zeta_i, \quad \zeta_i \geq 0$$

- 惩罚因子 C 则是为了避免系统轻易放弃一些重要的数据，减小系统损失。引入 C 后目标函数变为：

$$\min \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{n}\zeta_i$$

5.4.1 近似线性可分样本集问题：

假设线性可分的样本集里多了一个异类样本，如图 5.5 所示黄色那个点，它是方形的，与周围圆形的点不一样。这样原来线性可分的问题就变成了非线性的，或者叫做“近似线性可分”的问题。

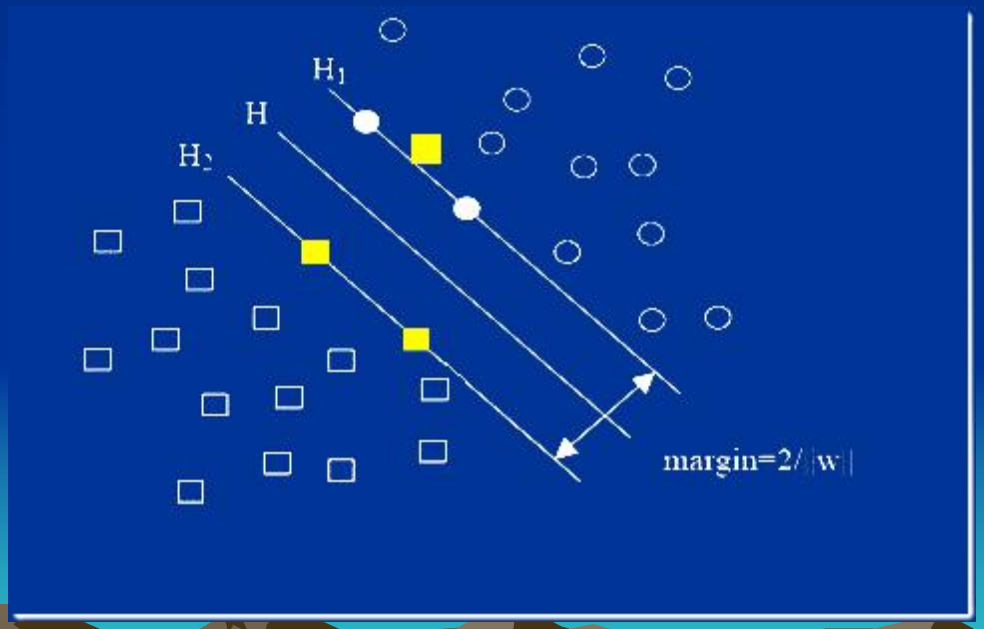


图5.5 “近似线性可分”样本集

5.4.2 “硬间隔” 分类法：

这个孤立的样本点可能就是噪声，但在原来的优化问题的表达式中，程序没有容错性，要考虑所有的样本点，像上面这种有噪声的情况会使得整个问题无解。

因此，我们原来的SVM优化解法也叫做“硬间隔”分类法，因为该方法硬性的要求所有样本点都满足和分类平面间的距离必须大于某个值。

5.4.3 “软间隔” 分类法：

由上面的例子中也可以看出，“硬间隔”的分类法其结果容易受少数点的控制，这是很危险的。

为了解决这个问题，可以仿照人类思维的习惯，允许一些点到分类平面的距离不满足原先的要求。由于不同的训练集各点的间距尺度不太一样，因此用“软间隔”来衡量有利于我们表达形式的简洁。

5.4.4 松弛变量系数 ζ_i 的引入：

为此，引入一个非负的松弛项，有两种常用的方式：

$$\sum_{i=1}^{n} \zeta_i$$

另一种是：

$$\sum_{i=1}^{n} \zeta_i^2$$

分别叫做一阶和二阶软间隔分类器。

给原约束： $y_i \cdot (W^{\mathrm{T}} \cdot X_i + b) \geq 1$ 中的这个硬性阈值“1”加入松弛变量ζ_i，成为：

$$y_i \cdot (W^{\mathrm{T}} \cdot X_i + b) \geq 1 - \zeta_i \ , \quad \zeta_i \geq 0$$

即允许样本点间隔可以比1小。但这意味着放弃了对这些点的精确分类，可以说是一种损失。但是也带来了好处，那就是使分类面不必向这些点的方向移动，因而可以得到更大的几何间隔。

松弛变量ζ_i的特性：

① 只有“离群点”才有松弛变量与其对应，所有没离群的点松弛变量都等于 0；

② 松弛变量 ζ_i的值对应于“离群点”到底离群有多远，点越远，ζ_i的值就越大。

5.4.5 “软间隔”下的目标函数惩罚因子C：

“硬间隔”分类对应的优化问题就是希望目标函数越小越好，而加入松弛系数 ζ_i 带来的损失就是使目标函数变大了。那如何来衡量损失？

把损失加入到目标函数里的时候，就需要一个惩罚因子，原来的优化问题就变成了下面这样：

$$\min \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{n}\zeta_i$$

$$s.t. \quad y_i\left[(W^{\mathrm{T}}\cdot x_i)+b\right]\geq 1-\zeta_i \qquad \zeta_i \geq 0$$

惩罚因子C的取值：

惩罚因子 C 的大小决定你有多重视离群点带来的损失。

① 当 C 越大，对目标函数的损失也越大，暗示着你非常不愿意放弃这些离群点。

此时，ζ_i 只能趋近于0，即算法对误差点的容忍度很低，稍有一个点离群，目标函数的值马上变成无限大，让问题变成无解。这样对样本的拟合性较好，但是泛化性能不一定好。

惩罚因子C的特性：

② 当C取值较小时，处于两个支持边界面间的样本变多，错分的可能性变大，对样本的拟合性能下降，但却可能更合理，因为样本往往可能带有噪声。

因此，惩罚因子C的取值权衡了经验风险和结构风险：C越大，经验风险小，结构风险大，容易出现过拟合；C越小，模型复杂度低，容易出现欠拟合。

惩罚因子C与数据集偏斜问题：

数据集偏斜 (unbalanced)指的是参与分类的两个类别样本数量差异很大。比如说正类有10,000个样本，而负类只给了100个，如图5.6所示。

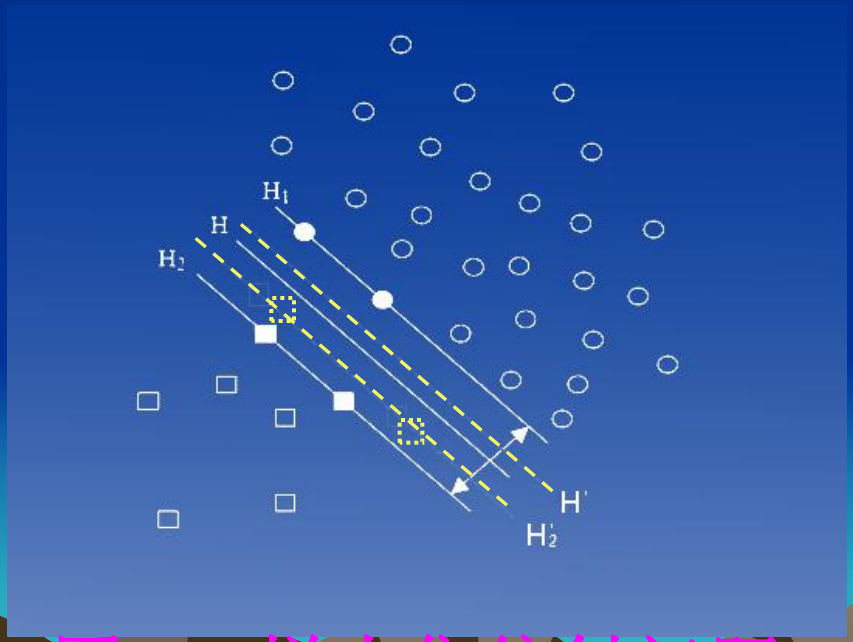


图5.6 样本集偏斜问题

如图5.6所示，方形点是负类。由于负类的样本很少很少，而且有一些本来是负类的样本点没有提供，比如图中两个虚线的方形点。如果这两个点有提供的话，那算出来的分类面应该是两条虚线。

这种由于数据集数量偏斜现象的存在，使得数量多的正类可以把分类面向负类的方向“推”的现象叫做数据集偏斜。它影响了分类结果的准确性。

解决数据集偏斜问题的方法之一就是在惩罚因子 C 的取值上进行调整，给样本数量少的负类更大的惩罚因子，表示我们重视这部分样本。因此我们的目标函数中因松弛变量而损失的部分就变成了：

$$C_+ \sum_{i=1}^{p} \zeta_i + C_- \sum_{j=p+1}^{p+q} \zeta_j \qquad \zeta_{i,j} \geq 0$$

其中 $i = 1, \ldots, p$ 都是正样本；$j = p + 1, \ldots, p + q$ 都是负样本。

C_+ 和 C_- 的大小是试出来的(参数调优)，但是他们的比例可以有些方法来确定。假定$C_+ = 5$，那确定 C_- 的一个很直观的方法就是使用两类样本数的比来算。对应图5.6的例子，C_- 就可以定为500 (因为 10,000 :100 = 100 : 1)。libSVM就直接使用这样的样本数量比。

但是，如果分类错误的原因是因为负类的样本分布的不够广(没扩充到负类本应该有的区域)，比如给政治类和体育类的文章做分类，政治类文章很多，而体育类只提供了几篇关于篮球的文章，这时分类会明显偏向于政治类。

这时，如果给体育类文章增加的样本全都是关于篮球的(也就是说，没有足球，排球，赛车，游泳等等)，虽然体育类文章在数量上可以达到与政治类一样多，但过于集中了，结果仍会偏向于政治类。

所以，给 C_+ 和 C_- 确定比例更好的方法，应该是衡量他们分布的程度。比如可以算算他们在空间中占据了多大的体积，例如给负类找一个超球——就是高维空间里的球——它可以包含所有负类的样本，再给正类找一个，比比两个球的半径，就可以大致确定分布的情况。显然半径大的分布就比较广，就给小一点的惩罚因子。

惩罚因子C的特性：

惩罚因子 C 不是一个变量，而是一个事先指定的值。在指定这个 C 值以后，解一下，得到一个分类器，然后用测试数据看看结果怎么样；如果不够好，换一个C 的值，再解一次优化问题，得到另一个分类器，再看看效果…… 。如此就是一个参数寻优的过程，但这和优化问题本身不是一回事，优化问题在解的过程中，C 一直是定值。

加入松弛变量ζ_i与惩罚因子C后的软分隔求解：

尽管加了松弛变量和惩罚因子，但其优化问题与原始的“硬间隔”求解，没有任何更加特殊的地方。

软分割的求解过程，就是先试着确定一下 w^*，也就是确定了线性分界面及支持向量面的斜率，这时看看间隔有多大，又有多少点离群，把目标函数的值算一算，再换另一组三个面的斜率，再把目标函数的值算一算，如此往复(迭代)，直到最终找到目标函数最小时的 w^*。

5.4.6 核函数与松弛变量作用的区别：

- 核函数的作用是将低维线性不可分的数据，映射到高维空间，同时避开高维直接映射计算，使其达到近似线性可分的状态。
- 此时，再利用松弛变量和惩罚函数来处理那些少数的离群点，在高维空间实现数据的“软分隔”，以达到很好的分类效果。

5.5 多类分类问题

- SVM是一典型的两类分类器，但现实中要解决的往往是多分类问题。

- 几种方法：

  1. 一对多方法 (One-Against-The Rest)
  2. 一对一方法 (One-Against-One)
  3. SVM决策树 (SVM Decision Tree)

5.5.1 一对多方法 (One-Against-The Rest)

如果有 k 类，在其中某一类和其它 $k-1$ 类间建立最优分类面。接着，另选一类和其它的 $k-1$ 类间建立最优分类面……。以此类推，最后将建立 k 个SVM分类器，每个SVM分类器将某一类的数据从其它类中分离出来。

但是，这种方法有“数据集偏斜”问题，而且每次训练都要用到所有样本。

例如：有5个类别，第一次分类就把类别1的样本定义为正样本，其余2、3、4、5类的样本合起来定义为负样本，得到第一个SVM分类器，它能够指出测试样本是不是第1类的。

第二次分类把类别2的样本定义为正样本，把1，3，4，5类的样本合起来作为负样本，得到第二个SVM分类器……，如此下去。

但这种方法容易导致分类重叠现象和不可分类现象，并人为造成“数据集偏斜”问题。

5.5.2 一对一方法 (One-Against-One)

每次只选一个类的样本作为正类样本，而负类样本也是只是从剩下的选其中一类。因此，第一个SVM分类器只需要回答“是第1类还是第2类”，而第二个SVM分类器只会打“是第1类还是第3类”……，以次类推，直至最后一类作为负样本。

假设共有 k 类，这种方法需要建立的SVM分类器数目为 $k(k-1)/2$。

在实际分类过程中，把测试样本扔给所有的分类器，让每个SVM分类器都投上自己的一票。最后统计票数，哪个类别得到的票数最多，就将测试样本分入该类。

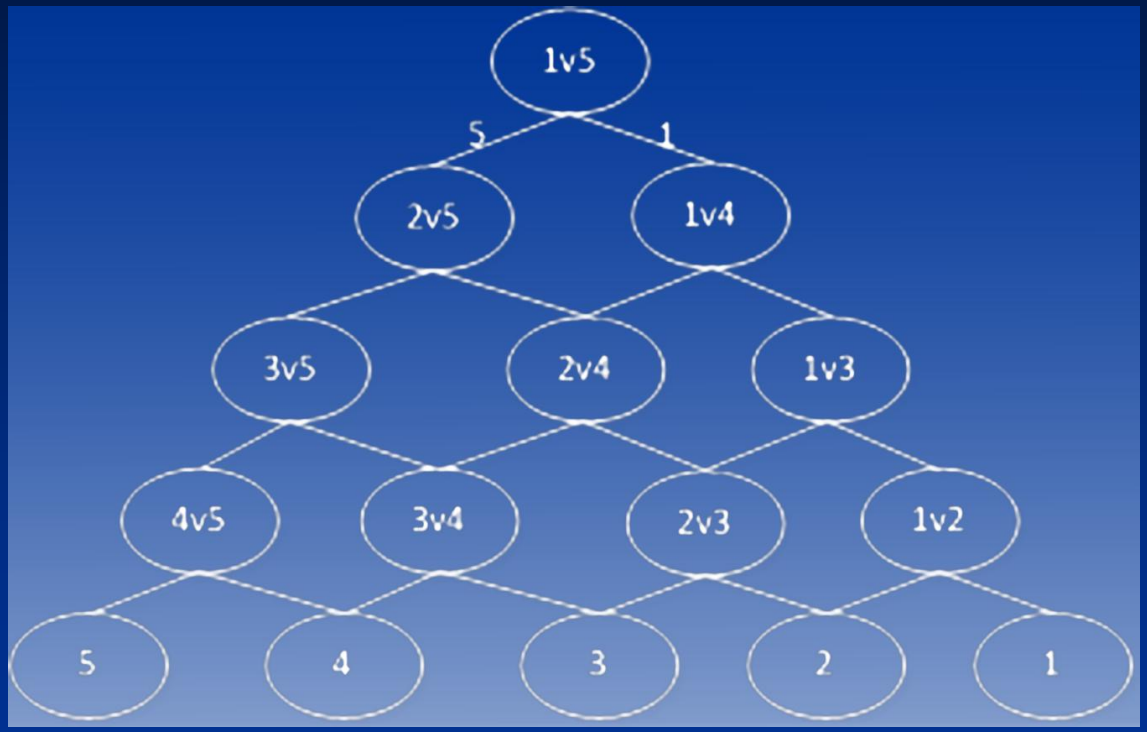
假设 $k = 1000$，则要建立的SVM分类器的数量会上升至约500,000个，过于复杂。

5.5.3 SVM决策树(SVM Decision Tree)

将SVM与二叉树结合，构成多类分类器，形似有向无环图，也叫DAG SVM。它只需要调用 $k-1$ 个SVM分类器，速度快，且没有分类重叠和不可分类现象。5类别的SVM决策树分类器如图5.7所示。

缺点是如果某个节点出错，那么后面的分类器无法纠正它的错误，这个错误就会在后续节点中不断累积。

图5.7 SVM决策树分类器



分类时，先问分类器“1v5”(意思是第1类还是第5类)，如果回答5则继续往左走，再问“2v5”这个分类器，如果回答还是5，则继续往左走……，这样一直下去，就可以得到分类结果。

SVM决策树分类器的改进：

1. 参与第一次分类的两个类别，选择差别最大的两类，或者取在两类分类中正确率最高的那个分类器作为根节点。
2. 分类器不光输出类别标签，还输出一个类似“置信度”的参数，这样就可以按照一定的概率走向另一分支。

5.6 LibSVM工具箱

- LIBSVM是台湾大学林智仁教授2001年开发的一套支持向量机的库，运算速度快，可以很方便的对数据做分类或回归。由于LIBSVM程序小，运用灵活，输入参数少，并且是开源的，易于扩展，因此成为目前国内应用最多的SVM的库。这套库目前已经发展到3.23版 (release date: 2018-07-15)。

5.6.1 安装方法：

解压文件，把当前工作目录调整到libsvm所在的文件夹下，再在set path里将libsvm所在的文件夹加到里面。然后，在命令行里输入：

```
>>    mex -setup  %选择一下编译器
>>    make
```

就可以了。

5.6.2 Libsvm带的SVM函数：

- svmtrain —— 通过训练集来训练模型；
- svmpredict —— 对测试集进行预测；
- svmsclale —— 对原始样本进行缩放；
- svmtoy —— 可视化展示训练数据和分类界面。

5.6.3 Libsvm的数据格式

Label 1:value 2:value …

Label是类别的标识，比如上节train.model中提到的1 -1，也可以自己随意定。

Value就是要训练的数据，从分类的角度来说就是特征值，与序号之间用空格隔开，比如：

-15 1:0.708 2:1056 3:-0.3333

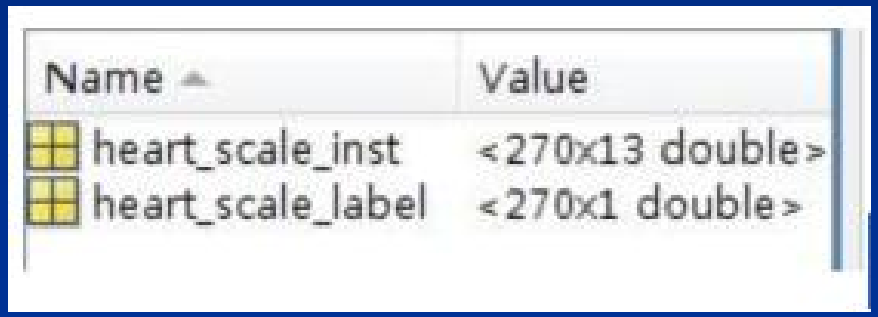
如果特征值为0，特征冒号前面的可以不连续。如：

-15 1:0.708 3:-0.3333

表明第2个特征值为0。

5.6.4 Libsvm实例研究:

- load heart_scale.mat %工具箱里自带的数据，如图:



其中 heart_scale_inst是样本, heart_scale_label是样本标签

- **model = svmtrain(heart_scale_label, heart_scale_inst, '-c 1 -g 0.07');**

  %训练样本，具体参数的调整请看帮助文件

- [predict_label, accuracy, dec_values] =svmpredict(heart_scale_label, heart_scale_inst, model);

%分类预测，这里把训练集当作测试集，验证效果如下:

% test the training data，
Accuracy = 86.6667% (234/270) (classification)

5.6.5 Libsvm工具箱下载地址：

- 该软件包可在http://www.csie.ntu.edu.tw/~cjlin/免费获得。