第九章 分类器组合

- 9.0 引言
- 9.1 分类器组合I
- 9.2 分类器组合II

9.0 引言

三个臭皮匠 > 诸葛亮

9.0 引言

- 给定一个训练样本集S，通过学习可以得到一个分类器f_i。他是对真实函数f的一个估计。
- 组合分类器——假设我们有L个分类器$\{f_1,\ldots f_L\}$，如何将它们组合起来得到更好的分类器。（Ensemble Method）

9.0 引言

- 组合分类器比其中单个分类器更精确的充分必要条件——单个分类器是“精确”（accurate）的并且分类器之间是“相异”（diverse）的。
- “精确”——比随机猜测好
- “相异”——错误是独立的

9.0 引言

- 对特征的处理
  - 用同样的特征或表达
  - 用不同的特征或表达
- 对样本的处理
  - 用同样的样本
  - 用不同的样本

9.0 引言

- 对分类器的处理
  - 用同样的分类器
  - 用不同的分类器
  - 对分类器输出的多层处理

9.1 分类器组合I

9.1 分类器组合I

- 对分类器的处理
  - 不同分类器组合
  - 对分类器输出的多层处理——Stacking

9.1 分类器组合I

- 若每个基本分类器的输出是实数（比如属于每类的概率），则通过对这些输出结果相加、相乘等方法取算术或几何平均给出组合结果，也可以取最小、最大等方法给出组合结果

9.1 分类器组合I

- 若分类器只输出类别标号，则可以通过投票的方式，用多数优胜法得到组合结果

9.1 分类器组合I

- 假如基本分类器之间相互“独立”，只要每个分类器的正确率都大于50%，最终结果一定比每个分类器单独的结果好
- 如假设有21个分类器，每个分类器的错误率为0.3并且独立。则多数投票法的错误率为：

$$P_{error}=\sum_{i=11}^{21}C_{21}^{i}0.3^{i}(1-0.3)^{21-i}=0.026<<0.3$$

9.1 分类器组合I

- 加权多数法（Weighted Majority）——每个分类器赋予一定的权重。训练过程中，对分错的分类器进行惩罚，减少其权重。

9.1 分类器组合I

- 问题：实际上很难找到多种完全独立的分类算法。如果基本分类器之间不独立，理论分析变得十分困难。实验结果也显示这种方法并不十分有效，甚至不如从基本分类器中挑出一个正确率最高的。另外，也有人发现独立的分类器组合不一定比不独立的分类器组合好

9.1 分类器组合I

- Wolpert于1992年提出的Stacked Generalization（Stacking）
- Stacking的思想是把基本分类器的输出作为下一级分类器的输入，再进行学习

9.1 分类器组合I

- 假设已有K个基本分类器 $C_1, C_2, \cdots, C_K$
- 采用Leave-one-out思想，每次从训练集中挑出一个样本 x_i，用其它所有样本分别训练 $C_1, C_2, \cdots, C_K$ ，得到的K个分类函数对 x_i 的类别各有判断，记做：$C_1(x_i), C_2(x_i), \cdots, C_K(x_i)$
- 这 K个数实际上是对样本 x_i 的一个全新的描述，或者说新的特征。它还包含了各个分类器以及训练集中其它所有样本的信息

9.1 分类器组合I

- 类似的，对训练集中每个样本都生成一个K 维向量作为特征。
- 这样得到一个新的训练样本集合，每个样本的特征是由基本分类器输出组成的 K 维向量
- 用这个新的训练集训练一个“最终决策”分类器。

9.1 分类器组合I

- 识别：测试样本 x
- 首先将 $C_1, C_2, \cdots, C_K$（用训练集中的全部样本进行训练得到的判别函数）作用于x上得到测试样本的新的特征

  $(C_1(x), C_2(x), \cdots, C_K(x))$
- 将上式作为输入交给“最终决策”分类器产生最后的判断结果。

9.1 分类器组合I

- Stacking推广了交叉检验。实际上，采用一种特殊的“最终决策”分类器就可以得到Leave-one-out交叉检验选取最优分类器的结果。请思考最终决策”分类器应该是什么样的？
- Stacking利用基本分类器的输出作为样本的新的特征，后来被称为meta level learning或meta-learning。

9.1 分类器组合I

- Stacking最大的问题在于缺少有效的理论基础。Wolpert（Stacking的提出者）甚至称之为black art（魔法），不清楚在何种条件下Stacking有效，为什么有效

9.2 分类器组合II

9.2 分类器组合II

- 对样本的处理——样本重采样
- 介绍两种方法：***B***ootstrap ***Agg***regat***ing*** (Bagging)和***Ada***ptive ***Boost***ing (AdaBoost). 这两种方法都有深刻的理论背景，在大量的、针对各种类型的数据集的实验中都取得了显著的效果。

9.21 Bagging

- Bagging是统计学家Breiman于1996年提出的，它的思想根源是统计学中非常重要的Bootstrap理论。

Bootstrap

- 考虑下面的一个很基本但非常深刻的统计问题：有n个i.i.d.抽取的样本 $x_1, x_2, \cdots, x_n$，在只有这些样本的条件下研究均值估计量 $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ 的统计特性。
- 注意：均值估计量本身也是一个随机变量。想要研究它的统计量，比如方差。

Bootstrap

- 问题：研究随机变量的统计特性需要大量样本，但根据 $\overline{x}=\frac{1}{n}\sum_{i=1}^{n}x_i$ 只能得到一个样本，不可能研究。
- 启发性思考：如果我们知道观测样本服从的是什么分布，我们根据这个分布产生出B组，每组包含n个样本的数据，就可以得到B个“均值”的样本。

Bootstrap

- 困难：真实的分布不知道。
- 解决办法：Bootstrap（自举）
  - 思想：模拟真实的分布。按照“模拟”的分布生成许多组，每组包含n个样本的数据。
  - 如何模拟：
    - 不是连续的分布函数，而是一个离散的概率。
    - 在 $x_1, x_2, \cdots, x_n$ 这n个样本点上每点的概率为1/n，其它所有地方为0。

Bootstrap

- Bootstrap数据集的生成：
  - 每组包含n个样本的数据。以1/n的概率从 $x_1, x_2, \cdots, x_n$ 中随机抽取n个数，作为一组。
  - 每组里面很可能有重复的数，也有没被抽到的。
  - 每组计算一个均值，看作均值（作为随机变量）的一个样本。有了大量样本后，应用通常的统计方法可以得到均值估计量的高阶统计特性。

Bootstrap

- 为何这样模拟：它模拟的并不是概率密度函数，而是概率密度函数的积分——概率分布函数 。在n个样本点上每点的概率为1/n作为概率密度函数的积分正是统计学中的经验分布函数，它是真实的概率分布函数的很好的逼近。
- Bootstrap模拟的方法在理论上具有最优性，讨论超出了我们的范围，参文献：Efron 1979

Bootstrap

- Bootstrap给出的均值统计量的二阶矩估计公式：

$$\hat{\bar{x}}^{*(\cdot)} = \frac{1}{B}\sum_{b=1}^{B}\hat{\bar{x}}^{*(b)}$$

均值统计量的数学期望的估计

每组数据得到的一个均值统计量样本

$$Var_{boot}[\bar{x}] = \frac{1}{B}\sum_{b=1}^{B}[\hat{\bar{x}}^{*(b)} - \hat{\bar{x}}^{*(\cdot)}]^2$$

均值统计量的方差

9.21 Bagging

- Bagging把Bootstrap直接应用到模式识别分类器的设计中：
  - 通过n个样本所设计的分类器实际上是有随机性的.
  - 随机性来源于n个样本是按照某个概率分布i.i.d.随机抽取的。
  - 如果有很多组每组n个样本的训练集，我们可以训练出很多个分类函数。

9.21 Bagging

- 直观的想法：对某个测试样本，如果用这些分类函数的结果投票，以多数为优胜，将得到一个“平均”的结果，这个平均的结果可能比只用一个随机的n个样本的训练集得到的判别函数要好，或者说更“稳定”。

9.21 Bagging

- Breiman利用boostrap方法生成很多组，每组n个样本的训练集，求出一个投票的"平均"结果。他称这种方法为***B***ootstrap ***Agg***regat***ing***，简称Bagging。

9.21 Bagging

- 大量实验结果表明，对于“不稳定”的分类器（也就是说如果训练样本稍加改变，得到的判别函数就会产生较大变化），Bagging能显著提高它的正确率。

9.22 AdaBoost

- AdaBoost产生于计算学习理论（Computational Learning Theory，Valiant 1984）
- 核心：PAC（ Probably Approximately Correct）学习。某个模型如果称为PAC learnable，如果存在一种算法能够以很大的概率，学到一个很高的精确度，并且在多项式时间内完成（时间是输入变量、概率、和精确度的多项式函数）。

9.22 AdaBoost

- 例：爱国者导弹的精确度是13米，但这种精确性并不是百分之百的，而是以一个很大的概率保证的(probably)，研究系统的学者称之为可靠性。爱国者导弹的可靠性是95%：以95%的概率保证命中距目标13米的范围内

9.22 AdaBoost

- PAC学习的本意是对各种模型进行分类，找出哪些模型是PAC可学习的。实际上由于需要对任意高的概率和精确度都可学习，只有很少的模型是属于PAC可学习范畴的; “强” 可学习模型。
- “弱” 可学习模型。它只能保证以任意高的概率学到比随机猜好一点（在多项式时间内完成）。两类问题，随机猜的正确率为50%。
- 一般的想法“弱” 可学习模型应该比“强” 可学习模型范围更大

9.22 AdaBoost

- Schapire（1990）证明了一个令人惊奇的结果："弱" 可学习模型与"强" 可学习模型这两个概念是等价的。即：只要模型是弱可学习的，就能找到一种算法在多项式时间内，以任意高的概率和精确度学习出这个模型。
- 关键：Boosting技术。对于一个弱可学习的模型，我们Boosting（增强）它的学习算法。改造后的算法就能满足强学习的要求。

Boosting方法

- 首先根据已有的训练样本集设计一个子分类器，使其分类效果比随机猜好一点，依次训练一组子分类器，其中每个子分类器的训练集都由之前各个子分类器所给出的包含信息量最大的样本点组成，最终的判决结果由各子分类器的结果共同决定，并且对训练样本集的准确率能够任意的高。

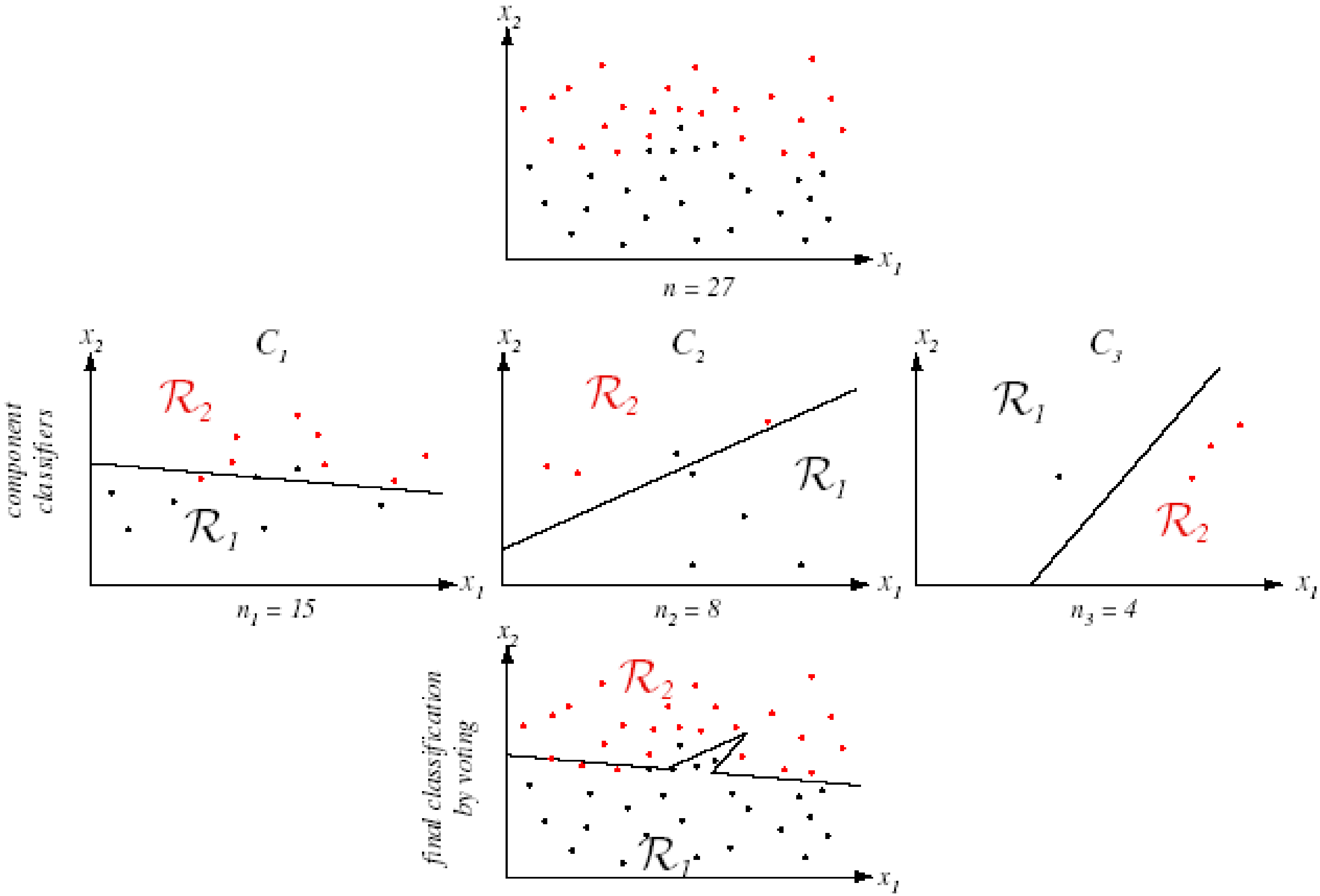
Boosting方法

- 例，两类问题如何使用Boosting方法创建三个子分类器
  - 首先，从大小为 n 的原始样本集 D 中随机选取 n_1 个样本点（不放回），组成样本集 D_1
  - 根据样本集 D_1 训练出第一个子分类器 C_1 比随机猜好一点。
  - 从 D 剩余的样本点中挑选出 n_2 个组成 D_2 使 D_2 中一半的样本能被 C_1 正确分类，另一半被 C_1 错分

Boosting方法

- 根据样本集 D_2 训练出第二个子分类器 C_2
- 在 D 剩余的样本点中选取样本点，用 C_1 和 C_2 进行分类，如果 C_1 和 C_2 判决的结果不同，就把这个样本加入 D_3
- 根据样本集 D_3 训练出第三个子分类器 C_3

用三个子分类器对一个新样本 x 分类，若 C_1 和 C_2 的判决结果相同，标记为这个类别，若不同，标记为 C_3 得到的类别



x_2
x_1
$n = 27$
component classifiers
C_1
$\mathcal{R}_2$
$\mathcal{R}_1$
$n_1 = 15$
C_2
$\mathcal{R}_2$
$\mathcal{R}_1$
$n_2 = 8$
C_3
$\mathcal{R}_1$
$\mathcal{R}_2$
$n_3 = 4$
final classification by voting
$\mathcal{R}_2$
$\mathcal{R}_1$

9.22 AdaBoost

- Boosting的理论证明假定有无穷多的样本，这在实际中是不可能的。
- 如何在固定的训练样本集上将Boosting实用化1996年，Freund和Schapire两人提出了AdaBoost（***Ada***ptive ***Boost***ing）算法，在很多实验中都表现出非常好的性能。

AdaBoost算法

- AdaBoost利用一个基本分类器（看作弱分类器），根据它每次的输出结果，自适应地改变样本权重（已被正确分类的样本权值较小，被错分的样本权重提高），训练出一系列分类器，根据分类器的表现加权投票给出最终判决结果。
- x^i表示原样本集D中的样本，y_i表示其类别标号，$W_k(i)$表示第k次迭代时全体样本的权重分布，AdaBoost算法如下：

AdaBoost算法

- 1 begin initialize $D=\{x^1,y_1,\cdots,x^n,y_n\}$,
  $$k_{\max},W_1(i)=1/n,i=1,\cdots,n$$
- 2 $k\leftarrow 0$
- 3 do $k\leftarrow k+1$
- 4 训练使用按照 $W_k(i)$ 采样的 D 的弱分类器 C_k
- 5 对使用 $E_k\leftarrow W_k(i)$ 的 D 测量 C_k 的的训练误差

AdaBoost算法

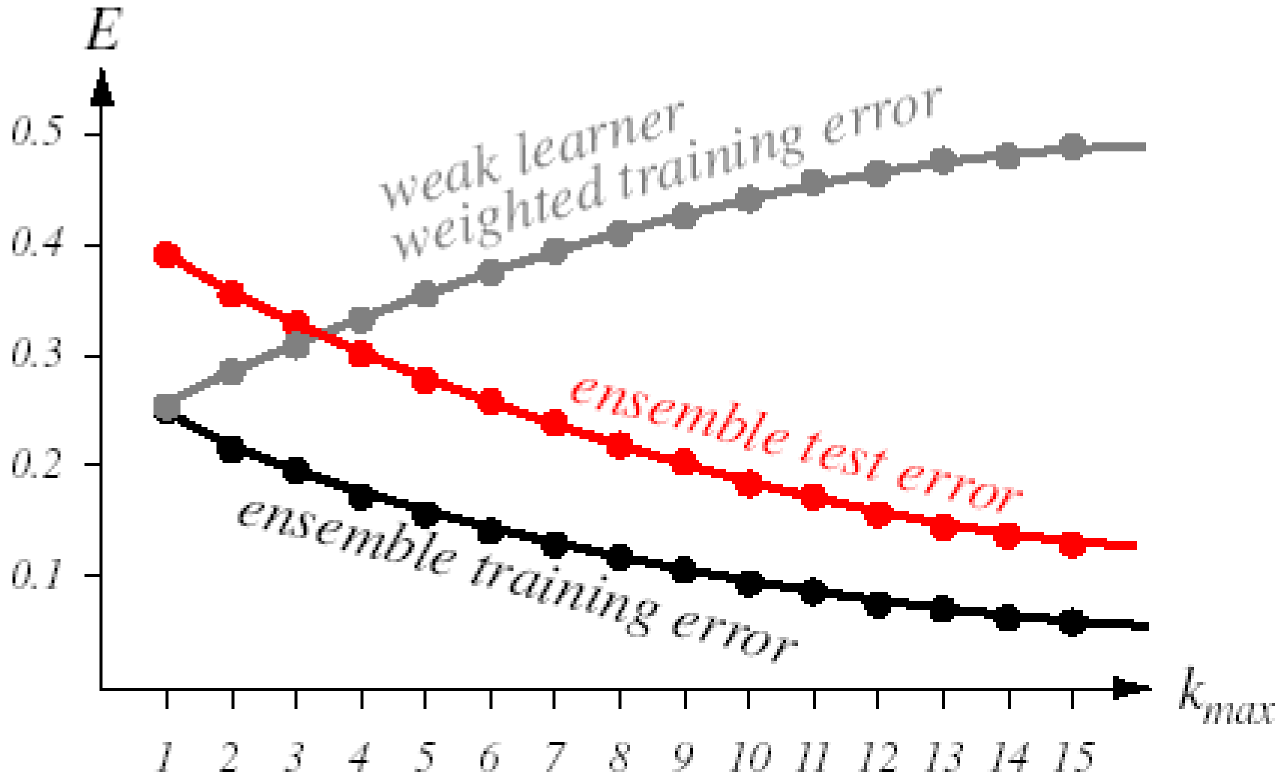
- 6 $\alpha_k \leftarrow \frac{1}{2}\ln[(1-E_k)/E_k]$
- 7 $W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k} & \text{if } h_k(x^i) = y_i \\ e^{\alpha_k} & \text{if } h_k(x^i) \neq y_i \end{cases}$
- 8 until $k = k_{\max}$
- 9 return C_k and α_k , $k = 1, \cdots, k_{\max}$
- 10 end

AdaBoost算法

- 其中 Z_k 为归一化系数，$h_k(x^i)$ 为 C_k 给出的对 x^i 的标记
- 总分类器由各子分类器加权平均得到

$$g(x)=\left[\sum_{k=1}^{k_{\max}}\alpha_k h_k(x)\right]$$

- 多数情况，若每个子分类器都是弱学习器，$k_{\max}$ 足够大，总分类器的训练误差率能够任意的小


E
0.5
0.4
0.3
0.2
0.1
weak learner
weighted training error
ensemble test error
ensemble training error
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
k_max

9.22 AdaBoost

- 尽管Boosting有着坚实的理论基础，但它是在理想条件下成立的。解释AdaBoost在实际问题中的优异表现一直是学者们的目的，但是至今还没有完美的答案。
- 目前最合理的解释：
  - Boosting the margin：以两类分类问题为例，每次迭代得到的组合结果使得训练数据的margin增大。

9.22 AdaBoost

- AdaBoost有一个特点：即使经过若干次迭代，训练数据的识别率已经达到100%，继续迭代还能提高组合分类器在测试样本上的识别率。增大margin给这种现象一个比较合理的解释，它不断提高分类器的推广能力。

参考文献

- [1] D.H. Wolpert, “Stacked Generalization”, Neural Networks, Vol. 5, pp. 241-259, 1992.
- [2] R. E. Schapire, “The Strength of Weak Learnability,” Machine Learning, Vol. 5(2), pp. 197-227, 1990.
- [3]L. Breiman, “Bagging Predictors”, Machine Learning, Vol.24, pp. 123-140, 1996.
- [4]L. G. Valiant, “A Theory of the learnable,” Communications of the ACM, Vol. 27(11), pp. 1134-1142, 1984.
- [5]Y. Freund, and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” Journal of Computer and System Sciences, 55(1), pp. 119-139, 1997.