基于局部有效性的选择性决策树集成

（申请南京大学理学学士学位论文）

培养单位: 南京大学计算机科学与技术系

专　　业: 计算机科学与技术

本 科 生: 俞　　扬

指导教师: 周 志 华　教 授

二〇〇四年五月

基于局部有效性的选择性决策树集成

俞扬 （001221154）

（南京大学　计算机科学与技术系，南京　210093）

摘要：

集成学习通过为同一个问题训练出多个个体学习器并将结论进行合成，可以显著地提高学习系统的泛化能力。最近，一些研究者提出了选择性集成，即从训练好的学习器中选择一部分进行集成，可能获得比用所有学习器进行集成更强的泛化能力。本文对此进行了研究，并通过在局部样本空间上选择学习器，提出了一种基于局部有效性的选择性集成算法LOVSEN。该算法使用 k 近邻来确定个体学习器在局部样本空间的有效性，从而为待预测的样本选择合适的个体学习器进行集成。实验结果表明，LOVSEN 可以较为稳定地生成泛化能力较强的决策树集成。

关键词：

机器学习; 集成学习; 选择性集成; 决策树; 惰性学习; k 近邻

Local Validity Based Selective Ensemble of Decision Trees

YU Yang (001221154)

(Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

Abstract:

Ensemble learning can significantly improve the generalization ability of learning systems through training a finite number of neural networks and then combing their results. Recently, a new ensemble learning approach named selective ensemble has been proposed, which can further improve learning system's performance through ensembling *many* instead of *all* the base learners at hand. In this paper, the problem of selective ensemble has been investigated. Through ensembling base learners from local instance space, a novel local validity based selective ensemble approach name LOVSEN has been proposed. LOVSEN employs k-nearest neighbor algorithm to determine the local validity of base learners and then ensembles appropriate base learners for unseen instances. Experiments show that LOVSEN can stably generate ensemble of decision trees with better generalization ability.

Keywords:

Machine Learning, Ensemble Learning, Selective Ensemble, Decision Tree, Lazy Learning, k-Nearest Neighbor

1、引言

机器学习（Machine Learning）[26]是对计算机如何通过经验的积累，从而自动提高系统性能的机制的研究。机器学习涉及人工智能、概率统计、计算复杂性理论、哲学、心理学、遗传学、神经生物学等。随着该领域的发展，机器学习在最近 20 年的应用更加成熟而广泛，其成功应用的领域包括生物信息学、计算机辅助医疗诊断、计算金融学、行星地质学、网络安全等。在各种实验科学研究的共同的研究方法的每一步骤：观察、假设、验证、结论，机器学习都能够成为有利的助手[27]。

机器学习的核心问题是如何在计算机上进行有效的学习。根据标记（label）给定的不同形式，目前大致有四种机器学习框架：

- 监督学习（Supervised Learning）：给定一系列训练样本（training samples/instances），其中每个样本都做上了标记，比如说标记出这个样本来自对一个苹果外观的一次观测。学习的目的是从这些带有标记的样本中学习到一些概念，比如说什么样的数据对应苹果而不是香蕉，并且在未来给出新的样本时，能够正确预测新样本的标记。
- 非监督学习（Unsupervised Learning）：给定一系列没有任何标记的训练样本，学习的目的是发现隐藏在这些样本中的某种结构，例如样本的聚集情况。
- 强化学习（Reinforcement Learning）：从训练样本集中学习出状态到行为的映射，这些样本没有直接给出标记，而是给出了在一系列行为发生后导致的奖赏（rewards），也可视为滞后了的标记。
- 多示例学习（Multi-Instance Learning）：训练样本被组织成包（bag），训练样本集包含一系列的包，每个包中又包含一组样本。包被标记为正包（positive）和反包（negative），而包中的样本却没有标记。

本文的讨论仅限于监督学习。1988 年，Kearns 和 Valiant[21]提出了弱学习器与强学习器的等价问题。如果一个多项式级学习算法能够产生高精度的学习器，则这个学习器称为强学习器，而如果产生的学习器只是略好于随机猜测，那么这个学习器就是弱学习器。要设计出强学习器是一件相当有挑战性的工作，然而如果存在简单的方法能够将弱学习器提升为强学习器，就可以以较少的代价获得很高的精度。

Schapire[31] 通过设计出一个集成学习算法（即后来著名的 Boosting 算法），构造性地对上述猜想进行了证明。1990 年 Hansen 和 Salamon[19] 提出了神经网络集成（neural network ensemble），一般认为这是神经网络集成的起源，而集成学习作为神经网络集成的超集，其起源必然在此之前。从今天来看，在正式使用“集成学习”（Ensemble learning）这个名字之前，实际上已经有很多集成学习方面的研究工作，但其起源已经很难考证。

简单地说，集成学习是为同一个问题训练一组学习器，并将这些学习器联合起来执行预测任务。按照个体学习器的生成方式，目前的集成学习方法大致可以分为个体学习器可以并行训练的方法，以及个体学习器只能串行训练的方法。研究表明，集成学习是目前泛化能力

最强的机器学习技术之一。

以往的集成学习方法，如 Bagging[2]，AdaBoost[14]，是将所训练的所有学习器都进行集成。而最近的研究发现[39]，从所训练的学习器中选择一部分进行集成预测，能够得到更好的泛化能力。这种思想称为选择性集成（Selective Ensemble）。

本文对选择性集成进行了研究，提出对待预测样本所属的局部空间进行分析，仅利用在这个局部空间上有效的个体学习器进行集成，从而提出了 LOVSEN（LOcal Validity based Selective ENsemble）算法。具体而言，在训练阶段，产生一批学习器后，LOVSEN 利用 k 近邻来估计出每个学习器最“擅长”的区域，当给出一个测试样本时，选择在其邻域中的最佳学习器构成集成。

本文后续部分组织如下：第二节回顾集成学习算法。在此基础上，第三节展示了新的集成学习算法 LOVSEN，试验对比结果在第四节中展现。第五节总结全文。

2、集成学习

对于传统的监督学习问题，给出属性集（attribute/feature set）$A=\{A_1,A_2,\ldots A_n\}$ 和一个分布 $\mathcal{P}$，在 A 构成的空间上按照分布 $\mathcal{P}$ 抽取 N 个样本，构成一个训练样本集 $S_t=\{(\mathbf{x}_1,y_1),\ldots(\mathbf{x}_N,y_N)\}$。其中成对的 $(\mathbf{x}_i,y_i)$ 中，$\mathbf{x}_i$ 称为一个样本，y_i 是 $\mathbf{x}_i$ 的标记，它由一个未知的函数 $y_i=f(\mathbf{x}_i)$ 决定。$\mathbf{x}_i$ 是一个向量 $[x_{i,1},x_{i,2},\ldots x_{i,n}]^{\mathrm{T}}$，它的每一个分量是在属性集 A 对应分量上的取值。对于分类（classification）问题，y_i 属于一个离散数值集合{1,2,…,C}，集合中的每一个元素代表一个类别；对于回归（regression）问题，y_i 是一个实数。

学习算法的输出是一个学习器（分类器或者回归器），这个学习器就是对未知函数 f 的估计，记作 $\hat{f}$。当给出新样本 $\mathbf{x}$ 时，学习器将给出这个样本的预测值 $\hat{y}=\hat{f}(\mathbf{x})$（类别或实数）。这里，新样本是指独立于训练样本集 S_t，且符合分布$\mathcal{P}$的样本。在实际问题中，由于训练样本数量有限、存在噪音以及学习算法本身的限制等多方面原因，$\hat{f}$ 与 f 之间往往会存在一定的误差。

集成学习的方法首先在训练集上训练出 m 个学习器 $[\hat{f}_1,\ldots,\hat{f}_m]^{\mathrm{T}}$，当给出新样本时，让每一个学习器都进行预测，产生结果 $[\hat{y}_1,\ldots,\hat{y}_m]^{\mathrm{T}}$。然后通过某种方法，例如相对多数投票（majority voting），产生集成的预测结果 $\hat{y}$。

考虑一种简单情况，把每一个学习器看作随机变量 $[X_1,\ldots,X_m]^{\mathrm{T}}$，使用平均值作为集成的结果 $\hat{y}=\sum X_i(x)/m$。当满足两个条件：a) X_i 的精度>0.5，即比随机猜测要好；b) 对于任意 $i\neq j$，X_i 独立于 X_j，这时 $\hat{y}$ 与 y 的误差达到最小。这时，若令 m 趋向于无穷大，就可以得到 $\hat{y}=y$。虽然在实际上无法满足第二个条件，即学习器之间相互独立，但是这个结论直观地阐述了集成要取得更强泛化能力的条件：每个学习器要有一定的精度，并且学习器之间存在一定的差异[19]。

Krogh 和 Vedelsby[23]以回归学习器的集成推导出重要的集成学习的泛化误差公式，这个公式对于分类器的集成有着同样的意义。对于 n 个学习器，它们的集成的误差由(1)所示。

$$E = \bar{E} - \bar{A} \tag{1}$$

其中，$\bar{E}$ 为 n 个学习器的绝对误差的加权平均，$\bar{A}$ 为 n 个学习器相对于集成的误差的加权平均。$\bar{E}$ 指示出学习器固有的误差，$\bar{A}$ 指示出这些学习器之间的差异。这个式子表明了要获得好的集成就需要降低个体学习器的误差并增加学习器间的差异。

2.1 差异产生策略

目前研究者们已经提出了许多集成学习算法，这些算法往往利用不同的策略来获得学习器之间的差异。下面根据 Dietterich[8] 对差异产生机制的分类进行介绍。

2.1.1 对训练样本集的扰动

使用这一策略的方法从原来的训练样本集中产生多个新的训练样本集，在每个新的训练样本集上都训练出一个学习器，最后再将这些学习器聚集起来形成集成。研究者们发现，一些学习算法对训练样本的变动很敏感，例如决策树、神经网络、规则学习，训练集的微小变化会导致学习器较大的变化，这种学习算法称为不稳定的学习算法。而对另外一些算法来说，训练集的微小变化不会对学习结果有大的影响，例如 k 近邻、线性回归，这种算法称为稳定的学习算法。基于样本集扰动形成的集成，对不稳定的学习算法有好的效果，而对稳定的算法则没有明显作用。这一类集成算法中最为著名的有 Bagging 和 Boosting。

Table 1. a) The Bagging algorithm.

Input: training set S, learning algorithm L, iterations I

Output: ensemble leaner L_E

Process:

[1] for i = 1 to I

[2] S_i = bootstrap sample from S

[3] L_i = train a leaner on S_i via L

[4] end for

[5] $L_E = \underset{y \in Y}{\operatorname{argmax}} \sum_{i:L_i(x)=y} 1$

b) The AdaBoost algorithm.

Input: training set S of size m, learning algorithm L, iterations I

Output: ensemble leaner L_E

Process:

[1] for i = 1 to I

[2] normalize the weights so that the total weight is m

[3] S_i = sample from S according to weight distribution

[4] L_i = train a leaner on S_i via L

[5] $e_i = 1/m \sum_{x_i \in S_i : L_i(x_i) \neq y_i} \text{weight}(x_i)$

[6] $\beta_i = e_i / (1 - e_i)$

[7] weight(x_i)=weight(x_i)$\cdot \beta_i$, for all x_i where $L_i(x_i)=y_i$

[8] end for

[9] $L_E = \underset{y \in Y}{\operatorname{argmax}} \sum_{i:L_i(x)=y} \log(1/\beta_i)$

Breiman[2]提出的 Bagging 方法使用了简单的扰动方法，Table 1a)给出了 Bagging 的伪代码描述。为训练 N 个学习器，Bagging 从原始训练样本集中产生 N 个新的训练样本集。原始样本集包含 m 个样本，每个新样本集通过对原始样本集进行 m 次均匀分布的可重复抽样产生

（*bootstrap sampling*[13]）。所产生的每一个新样本集平均覆盖了原始样本集 63.2%的样本。Bagging 使用简单的平均投票、取相对多数的结果作为集成的输出。Clemen[6]的实验说明这种简单的投票方式有很好的鲁棒性。Bagging 是可并行的算法，每个新训练样本集可以并行产生，每个学习器也可以并行训练。

Boosting 中 N 个学习器排成序列，每一个学习器将关注它前面学习器预测错误的那些样本。Freund 和 Schapire[14][15]成功地发展出 AdaBoost（Adaptive Boost）算法，Table 1b)给出了 AdaBoost 算法的伪代码描述。第 i 个学习器 L_i 的训练集通过对原始样本集按照权值确定的概率分布 $p_i(x)$ 进行抽样产生。训练出 L_i 后，按照 h_i 在经过 $p_i(x)$ 加权的原始样本集上的错误率 e_i，产生下一个概率分布 $p_{i+1}(x)$ 用于第 i+1 个学习器的训练。最后，通过学习器的加权投票、取相对多数的结果作为集成的输出。其中，每个学习器的权值由这个学习器在训练样本集上的精度决定。一般来说，精度越高权值越高。

以往研究表明[10]，Bagging 和 AdaBoost 平均都能取得比单个学习器更强的泛化能力，AdaBoost 在多数情况下能够获得比 Bagging 要高的精度，然而在一些情况下 AdaBoost 的表现会差于单个学习器，而 Bagging 则能比较稳定地优于单个学习器。

2.1.2 对输入属性集的扰动

使用这一策略的方法通过对输入属性进行扰动来产生不同的新样本集，从而训练出不同的学习器。从样本空间的角度看，这相当于在不同的样本子空间上产生不同的学习器。此类技术大致可以分为基于属性性质的选择与基于属性效果的选择。

基于属性性质的选择通过衡量属性的某种性质来进行属性选择。一般选择在相关性较小的属性上分别训练学习器，希望使得训练出的学习器之间有较小的相关性。Tumer 和 Oza[35]对于每一种可能的类别，分别计算所有属性与这个类别的相关性。对每一种类别，分别选出与之相关性最高的部分属性，在这些属性构成的子空间中训练一个学习器。最后通过简单投票、取相对多数的结果作为集成的输出。

基于属性效果的选择枚举各种可能的属性组合，通过测试在这些组合上分类器的精度来确定属性的取舍。Bryll 等人[5]首先通过分类精度的测试，选取适合的属性的个数 m。然后从原始属性集中随机抽取 m 个属性，在这些属性构成的子空间中训练一个学习器。最后，通过测试选择那些精度最高的学习器参与集成。集成可以使用简单投票，也可以使用加权投票的结果作为最终的输出。

2.1.3 对输出标记的扰动

第三种策略是通过操纵样本集中样本的标记来产生不同的训练样本集，并以此训练学习器。

Dietterich 等人[9]描述了一种利用纠错输出编码(ECOC)的方法。该方法将训练集复制 N 份，对第 i 份训练集随机选择一个映射函数 $g_i : Y \to \{0,1\}$，把训练集中的每一个样本对 $(\mathbf{x}_i, y_i)$ 的标记 y_i 改为 $g_i(y_i)$。然后在每一份训练集上训练一个学习器。预测时，给出测试样本 $\mathbf{x}$，对于学习器 L_i，将 Y 的子集 $\{y \mid y \in Y, g_i(y) = g_i(L_i(x))\}$ 中的每一种类别投上一票。所有学习器投

票完成后，取票数最多的类别作为集成的输出。这种方法可以等价地看作：每个函数 g_i 使得学习器 L_i 做了一个二值输出，将 N 个学习器排列起来 $[L_1,\ldots,L_N]^{\mathrm{T}}$，它们的输出就构成一个 N 比特长的字。而 Y 中的每一种类别标记都可以通过函数序列 $[g_1,\ldots,g_N]^{\mathrm{T}}$ 映射成 N 比特的字。对于一个测试样本，得到 N 个学习器的输出构成的字 W。选择对应的字与 W 的海明距离最小的类别标记作为集成的输出。ECOC 集成的好坏取决于编码方法的选择，对每个类使用纠错编码，使类别编码间的海明距离最大，同时每个学习器的输出之间的相关性最小，这时可以得到最好的集成效果。ECOC 的原理来源于编码纠错过程，这就是算法名称中 *error-correcting* 的来源。文献[9]的实验表明 ECOC 对 C4.5 和 BP 神经网络的集成可以在多类别问题上获得较高的精度。

Breiman[4]在实验中意外发现，对样本的标记的扰动同样能够取得很好的效果。文献[4]中实现了两种扰动方法：一种对样本标记即 $(\mathbf{x}_i, y_i)$ 中的 y_i 添加高斯噪音；另一种在保持样本集中每种类别的样本数量所占比例没有变化的条件下，对样本标记添加随机噪音。实验结果表明，这种扰动方法生成的集成的泛化能力介于 Bagging 与 AdaBoost 之间。

2.1.4 对学习器内部机制的扰动

第四种策略涉及对具体学习器内部构造的改动。

Kolen 等人[22]考察了初始值对 BP 神经网络对的影响。给定神经网络不同的初始值，BP 算法可能收敛到不同的结果。由此，Parmanto 等人[28]中通过随机设定每个神经网络的初始值，来获得多个有差异的神经网络，形成的集成效果略差于 Bagging。

C4.5[29]也可以很容易地加入扰动[24]。C4.5 算法在决定一个结点所使用的划分属性时，将所有属性按照信息增益率（information gain ratio）排序，选择值最高的属性。Dietterich 等人[11]将 C4.5 改为：从排名较高的一部分属性中随机选择一个用于该结点的划分。由此可以形成多个不同的决策树进行集成。

以上四种策略可以相互组合。Schapire[32]把 ECOC 用于 AdaBoost，产生了 AdaBoost.OC 算法，其能力与比它复杂的 AdaBoost.M2 算法相当。Ricci 等人[30]把 ECOC 与属性选择结合起来，也取得了较好的效果。

2.2 为什么集成学习有效?

前面提到当所有学习器相互独立时，集成可以得到最好效果。但是这个理想条件在现实中无法达到，因此这种解释只能作为类比。Dietterich[8]希望从三个可能的角度说明集成学习的有效性（Fig. 1 直观地展现了这三种解释）。1) 统计的角度： 一些算法，例如 C4.5，当训练样本无穷多时，可以逼近真实目标。但是由于样本集的大小有限，对目标的表达只能达到一定精度，因此学习算法产生的学习器对目标的逼近也只能达到一定的精度。集成相当于对多个逼近程度相当的学习器求平均，减少每个学习器的误差，进一步逼近目标。2) 计算的角度：目标对于一些算法而言太“困难”，例如寻找符合训练样本的最小决策树是 NP-hard 问题[20]，

因此换而是用启发式贪婪搜索。因此学习的结果与目标也存在距离。3) 表示的角度：一些学习算法的搜索范围可能太小，没有包含目标，因此产生的学习器自然也不可能达到目标（虽然如 C4.5、BP 神经网络等算法对整个空间都进行了搜索）。对学习器的集成则可能得到超过搜索空间的解，相当于扩大了搜索空间。需要注意的是，上述说法仅是一些直观的说明或猜测，并不是集成学习有效性的正式的理论解释。到目前为止，集成学习有效性的理论分析仍然是一个吸引了众多研究者的开问题。

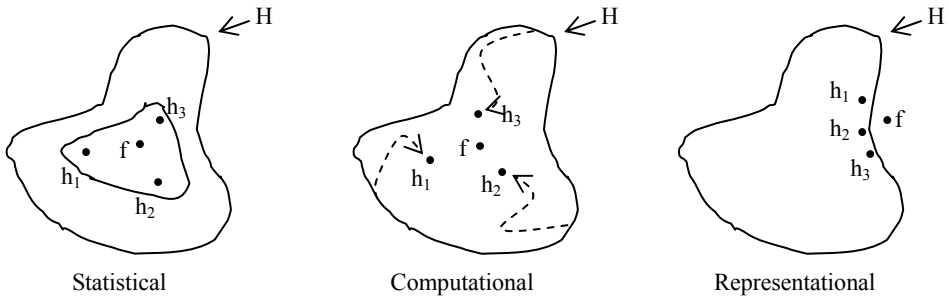


Fig. 1. Three fundamental reasons why an ensemble may works better than a signle learner.

2.3 选择性集成

由于降低学习器之间的相关性可以提高集成的泛化能力，因此研究者们把目光集中在如何通过加入扰动产生这样的学习器上。而 Zhou 等人[39]则把目光放到已经构造出的学习器上：在构造好一组学习器后通过筛选掉其中“坏的”学习器，从而得到高质量的集成。

文献[39]对集成的泛化误差进行了分析。对 N 个学习器构成的集成，以回归任务进行分析，得到(2)式：

$$E = \sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij} \big/ N^2 \tag{2}$$

C_{ij} 表达了第 i 个学习器与第 j 个学习器的相关性，如式(3)所式，其中 $y_{\mathbf{x}}$ 为 $\mathbf{x}$ 对应的标记。

$$C_{ij} = \int d\mathbf{x} p(\mathbf{x})(L_i(\mathbf{x}) - y_{\mathbf{x}})(L_j(\mathbf{x}) - y_{\mathbf{x}}) \tag{3}$$

(2)式与(1)式相似，都指出当学习器间的差异越大，泛化误差越小。一个极端情况，当所有学习器完全一样时，$C_{ij}=e$，即单个学习器的泛化误差，从而 $E=e$，说明集成没有任何效果。另一个极端情况，当所有学习器产生误差的领域完全没有重叠时，$C_{ij}=0$，使得 $E=0$，集成的效果达到最好。

由(2)式可以推导出“坏的”学习器的条件。对于用于集成的第 k 个学习器（其泛化误差为 E_k），如果满足(4)式，则第 k 个学习器就是“坏的”学习器。将这个学习器从集成中排除，有助于提高集成的泛化能力。

$$(2N-1)\sum_{i=1}^{N}\sum_{j=1}^{N}C_{ij} \le 2N^2\sum_{\substack{i=1\\ i\neq k}}^{N}C_{ik} + N^2E_k \tag{4}$$

文献[39]中对分类任务对进行的分析，也得到了判断“坏的”学习器的条件。

为取得学习器的最佳组合，在集成时为每一个学习器设定一个权值 $\mathbf{w}=\{w_1,w_2,\ldots,w_N\}$，得到集成的泛化误差为：

$$E=\sum_{i=1}^{N}\sum_{j=1}^{N}w_iw_jC_{ij} \tag{5}$$

文献[39]中推导出计算最优权 $\mathbf{w}^{\text{opt}}$ 的表达式：

$$w_k^{\text{opt}}=\frac{\sum_{j=1}^{N}(C^{-1})_{kj}}{\sum_{i=1}^{N}\sum_{j=1}^{N}(C^{-1})_{ij}} \tag{6}$$

然而，在实际中由于各个学习器的性能差别很小，使得矩阵 C 中的每一单元很接近，从而 C 往往是一个病态矩阵，有时甚至不可逆。所以(6)式从计算上不可行。

因此，文献[39]中采用了另一种方法来解决这个问题。求解 $\mathbf{w}^{\text{opt}}$ 可以看作是一个优化问题，因此文献[39]使用了遗传算法(genetic algorithm)[17]来寻找 $\mathbf{w}^{\text{opt}}$ 的近似解，形成了 GASEN（Genetic Algorithm based Selective ENsemble）算法，Table 2 描述了 GASEN 算法。

文献[39]的实验表明，GASEN 在回归和分类任务上的泛化能力都强于 Bagging 和 AdaBoost。文献[39]中将 GASEN、Bagging、AdaBoost 在这些数据集上进行 *bias-variance* 分解，分解结果表明 GASEN 的平均 *bias* 和 *variance* 都低于 Bagging 和 AdaBoost。

值得注意的是，在文献[39]的实验中，每个集成初始训练了 20 个神经网络作为单个学习器。而 GASEN 最后只使用了很少的几个进行集成。对回归实验，GASEN 在 10 个数据集上平均使用了 3.71 个神经网络；对分类实验，GASEN 在 10 个数据集上平均使用了 7.10 个。

Table 2. The GASEN algorithm.

Input: training set S, validation set V, learning algorithm L, iterations I, threshold λ

Output: ensemble leaner L_E

Process:

[1] for i = 1 to I

[2] S_i = bootstrap sample from S

[3] L_i = train a leaner on S_i via L

[4] end for

[5] $\mathbf{w}^{\text{opt}}$ = the evolved best weight vector via GA given fitness function $f(\mathbf{w})=1/E_{\mathbf{w}}^{V}$, where $E_{\mathbf{w}}^{V}$ is the estimated generalization error of the ensemble corresponding to the $\mathbf{w}$ on the validation set V

[6] $L_E=\arg\max_{y\in Y}\sum_{\mathbf{w}_i^{\text{opt}}>\lambda:L_i(x)=y}1$ for classification OR $L_E=\text{Avg}\sum_{\mathbf{w}_i^{\text{opt}}>\lambda}L_i(x)$ for regression

3、LOVSEN 算法

3.1 局部化与泛化能力

集成学习器 L_E 的泛化误差 E 可以定义为(7)式：

$$E = \sum_{A_d} \int_{Ac} d\mathbf{x} p(\mathbf{x}) I(L_E(\mathbf{x}) - y_{\mathbf{x}}) \tag{7}$$

其中 A_d 为样本属性 A 中的离散属性集，A_c 为 A 中的连续属性集，y 为 $\mathbf{x}$ 相应的标记，$p(\mathbf{x})$为样本的概率密度， $I(\cdot)$ 为(8)式的函数：

$$I(x) = \begin{cases} x^2 & \text{for regression} \\ \begin{cases} 0 & x = 0 \\ 1 & x \neq 0 \end{cases} & \text{for classification} \end{cases} \tag{8}$$

GASEN 所做的，是通过取得最佳的 L_E 使得(7)式右端最小，得到(9)式：

$$E^{\text{GASEN}} = \sum_{A_d} \int_{Ac} d\mathbf{x} p(\mathbf{x}) I(L_E^{\text{opt}}(\mathbf{x}) - y_{\mathbf{x}}) \tag{9}$$

又注意到和式∑与积分∫的可加性，将样本空间 D 分割为 n 个不交叠的区域$\{D_1, D_2, \ldots, D_n\}$，即 $D=\cup D_i$。从而，(7)式可以等价地写作：

$$E = \sum_{\substack{A_d \\ D_1}} \int_{\substack{A_c \\ D_1}} d\mathbf{x} p(\mathbf{x}) I(L_E(\mathbf{x}) - y_{\mathbf{x}}) + \ldots + \sum_{\substack{A_d \\ D_n}} \int_{\substack{A_c \\ D_n}} d\mathbf{x} p(\mathbf{x}) I(L_E(\mathbf{x}) - y_{\mathbf{x}}) \tag{10}$$

下面，假设在每一个区域 D_i 上，都取得了对于这个区域最优的集成 L_E^{opt,D_i} ，则这时的泛化误差为：

$$E^* = \sum_{\substack{A_d \\ D_1}} \int_{\substack{A_c \\ D_1}} d\mathbf{x} p(\mathbf{x}) I(L_E^{\text{opt},D_1}(\mathbf{x}) - y_{\mathbf{x}}) + \ldots + \sum_{\substack{A_d \\ D_n}} \int_{\substack{A_c \\ D_n}} d\mathbf{x} p(\mathbf{x}) I(L_E^{\text{opt},D_n}(\mathbf{x}) - y_{\mathbf{x}}) \tag{11}$$

[定理 1] $E^* \le E^{\text{GASEN}}$

[证明] 将(9)式分解，得到

$$E^{\text{GASEN}} = \sum_{\substack{A_d \\ D_1}} \int_{\substack{A_c \\ D_1}} d\mathbf{x} p(\mathbf{x}) I(L_E^{\text{opt}}(\mathbf{x}) - y_{\mathbf{x}}) + \ldots + \sum_{\substack{A_d \\ D_n}} \int_{\substack{A_c \\ D_n}} d\mathbf{x} p(\mathbf{x}) I(L_E^{\text{opt}}(\mathbf{x}) - y_{\mathbf{x}}) \tag{12}$$

∵ 由于 L_E^{opt,D_i} 是区域 D_i 上的最优集成，

∴ 有 $\sum\int d\mathbf{x} p(\mathbf{x}) I(L_E^{\text{opt},D_1}(\mathbf{x}) - y_{\mathbf{x}}) \le \sum\int d\mathbf{x} p(\mathbf{x}) I(L_E^{\text{opt}}(\mathbf{x}) - y_{\mathbf{x}})$ ，即得到 $E^* \le E^{\text{GASEN}}$

□

定理 1 说明了在样本空间的子区域上分别优化集成，将取得不坏于在整个空间上进行的优化更强的泛化能力。并且粗糙地说，划分的子区域数量越多，泛化能力越强。但是，值得

注意的是，定理 1 成立的前提是当子区域增多的时候，在各子区域上取得的最优集成的泛化能力没有降低。

3.2 LOVSEN

本文基于分类任务，提出了一种基于局部有效性的选择性集成算法 LOVSEN。Table 3 描述了 LOVSEN 算法。该算法使用 k 近邻[7]来进行局部选择寻优。

Table 3. The LOVSEN algorithm. selword[i] means the ith bit of selword. F is a filter defined as equation (13) or(14).

Input: training set S, learning algorithm L, iterations I, threshold λ, filter F, neighbors k

Output: ensemble leaner $L_{E,\mathbf{x}}$ for unknown sample $\mathbf{x}$

Training Process:

[1] train I learners on bootstrap sample from S via L

[2] S_L := sequence of trained learners $\{L_1, L_2, \ldots, L_I\}$

[3] for each sample $(\mathbf{x}_i, y_i)$ in S %verify learners on each sample

[4] for j := 1 to I

[5] if $L_j(\mathbf{x}_i)$ equals to $F(S_L, \lambda, (\mathbf{x}_i, y_i))$,
then set the jth bit of $\mathbf{x}_i$.word to 1,
else set the jth bit of $\mathbf{x}_i$.word to 0

[6] end for

[7] end for

Prediction Process:

[8] given unknown sample $\mathbf{x}$

[9] $S_{\mathbf{x},k}$:= k nearest neighbors of $\mathbf{x}$ in S, i.e. $S_{\mathbf{x},k} := \{\mathbf{x}_{m_1}, \mathbf{x}_{m_2}, \ldots, \mathbf{x}_{m_k}\}$

[10] selword := 1

[11] for i := 1 to k %select learners

[12] selword := selword AND $\mathbf{x}_{m_i}$.word

[13] end for

[14] if selword = 0, then selword :=1 %when none of learner is selected, all learners will be used.

[15] $L_{E,\mathbf{x}} := \arg\max_{y \in Y} \sum_{L_i(x)=y}$ selword[i] %majority voting

具体来说，LOVSEN 算法分为训练和预测两个阶段：

在训练阶段，首先使用可重复取样（bootstrap）方法产生 N 个不同的训练集并分别生成 N 个学习器，学习器序列 $S_L = [L_1, \ldots, L_N]^{\mathrm{T}}$。对训练样本集 S 中的每一个样本 $\mathbf{x}_i$，附加上一个 N 比特的字，表示为 $\mathbf{x}_i$.word。然后，在每一个训练样本 $\mathbf{x}_i$ 上对所有学习器进行验证，当第 j 个学习器在 $\mathbf{x}_i$ 上预测正确时，将 $\mathbf{x}_i$.word 的第 j 位设为 1。否则，即第 j 个学习器在 $\mathbf{x}_i$ 上预测错误时，将 $\mathbf{x}_i$.word 的第 j 位设为 0。也就是说，$\mathbf{x}_i$.word 记录了在 $\mathbf{x}_i$ 上预测正确的那些学习器。

在预测阶段，当给出一个新样本 $\mathbf{x}$ 时，找出 $\mathbf{x}$ 在训练样本中的 k 近邻 $S_{\mathbf{x},k}=\{\mathbf{x}_{m_1},\mathbf{x}_{m_2},\ldots,\mathbf{x}_{m_k}\}$，设 selword 为对这 k 个样本的字进行按位'AND'运算的结果，也就是说，selword 记录了那些在 $S_{\mathbf{x},k}$ 上都预测正确的学习器。最后，使用 selword 对应的这些学习器构成集成对 $\mathbf{x}$ 进行预测。对没有学习器在 $S_{\mathbf{x},k}$ 上都预测正确的情况，即 selword =0，这时使用所有的学习器进行集成，即设 selword =1。对选择出的学习器进行相对多数投票作为集成的输出。如果投票出现平局，则随机选取。

LOVSEN 算法在有 n 个不相同的样本的训练集上，当取 k 近邻时，样本空间被划分的数量区间为 $[n-k+1,C_n^k]$。

在真实数据中往往存在噪音，当某一个训练样本 $\mathbf{x}_i$ 的标记被噪音污染时，$\mathbf{x}_i$.word 记录的可能不再是在 $\mathbf{x}_i$ 上预测正确的学习器，而是预测错误的学习器。为此，有必要引进噪音处理机制。在 LOVSEN 中，引入校正函数 $F:S_L\times\lambda\times(\mathbf{x},y)\to\hat{y}$，其中 λ 为阈值，$(\mathbf{x},y)$为训练样本，输出 $\hat{y}$ 为校正后的样本标记。由此，当确定 $\mathbf{x}_i$.word 时，比较学习器 $L_j(\mathbf{x}_i)$与 $F(S_L,\lambda,(\mathbf{x}_i,y_i))$ 的值，相同时 $\mathbf{x}_i$.word 的第 j 位设为 1，否则设为 0。以下有两种可选的校正函数：

$$F(S_L,\lambda,(\mathbf{x},y))=\begin{cases}y & support(S_L,\mathbf{x})<\lambda\\ vote(S_L,\mathbf{x}) & support(S_L,\mathbf{x})>\lambda\end{cases} \tag{13}$$

$$F(S_L,\lambda,(\mathbf{x},y))=\begin{cases}y & confidence(S_L,\mathbf{x})<\lambda\\ vote(S_L,\mathbf{x}) & confidence(S_L,\mathbf{x})>\lambda\end{cases} \tag{14}$$

其中，$vote(S_L,\mathbf{x})$函数为 S_L 中学习器对样本 $\mathbf{x}$ 的相对多数投票的结果；$support(S_L,\mathbf{x})$为预测结果与相对多数投票结果相同的学习器在所有学习器中的比例；当每个学习器投的票不是整数 1，而是这个学习器的估计精度，$confidence(S_L,\mathbf{x})$为得票值最高的类别所得的票值。在下面的实验中，将对这两种校正函数进行比较。

4、实验

4.1 实验设置

本节将给出使用不同参数配置的 LOVSEN 算法的对比、LOVSEN 与其他算法的对比以及 LOVSEN 与 GASEN 在集成的个体学习器数量上的对比。实验以 C4.5 决策树[29]作为单个学习器进行集成，每一个集成使用了 20 个决策树。实验的程序在 Weka[37]的基础上建立，实验中使用的 C4.5 决策树是在 Weka 中实现的 J48 树。k 近邻算法使用了 HVDM[36]来计算离散属性间的距离，HVDM 的计算式如(15)所示：

$$\mathrm{HVDM}(x,y)=\sqrt{\sum_{a=1}^{n}d_a^2(x_a,y_a)} \tag{15}$$

其中，$d_a^2(\cdot)$ 由(16)式定义：

$$d_a = \begin{cases} \mathrm{vdm}_a(x,y) & \text{if a is nominal} \\ (x_a - y_a)^2 & \text{if a is linear} \end{cases} \tag{16}$$

$\mathrm{vdm}_a(\cdot)$ 函数用于衡量离散属性之间的距离：

$$\mathrm{vdm}_a(x,y) = \sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right| \tag{17}$$

其中，$N_{a,x}$ 为训练集中第 a 个属性的值为 x 的样本个数；$N_{a,x,c}$ 为训练集第 c 类中第 a 个属性的值为 x 的样本个数。

实验分别在真实数据集和人造数据集上进行。人造数据集将在 5.2 节中给出。真实数据集使用了 20 个 UCI 机器学习数据库[1]中的数据集。数据集中排除了带有缺失属性值的样本。Table 4 列出了使用的数据集的参数。样本个数从小样本集的 101 个到较大样本集的 3196 个不等，有 45%数据集的样本数量在[0,500)范围中，有 40%数据集的样本数量在[500,1000)范围中，有 15%数据集的样本数量在 1000 以上。其中，8 个数据集的属性全部为连续属性，4 个数据集的属性全部为离散属性，其它数据集的属性为混合属性。数据集的属性个数从 4 个到 60 个不等。类别数目从 2 个到 19 个不等，90%数据集的类别数目在[2,7]范围中。

Table 4. Parameters of data sets.

Dataset	Size	Attributes			Class
		Total	nominal	continuous	
anneal	898	38	32	6	6
autos	159	25	10	15	7
balance-scale	625	4	0	4	3
breast-cancer	277	9	9	0	2
breast-w	683	9	0	9	2
credit-a	653	15	9	6	2
credit-g	1000	20	13	7	2
diabetes	768	8	0	8	2
glass	214	9	0	9	7
heart-c	296	13	7	6	5
ionosphere	351	34	0	34	2
kr-vs-kp	3196	36	36	0	2
lymph	148	18	15	3	4
segment	2310	19	0	19	7
sonar	208	60	0	60	2
soybean	562	35	35	0	19
vehicle	846	18	0	18	4
vote	232	16	16	0	2
vowel	990	13	3	10	11
zoo	101	17	16	1	7

Table 5 列出了对 LOVSEN 算法取不同参数配置的表示符号。其中首列中 k 参数是算法寻找最近邻的个数，实验中尝试了 3 和 5 两种配置；首行中指出使用的不同校正函数，实验中尝试了不使用校正函数、使用(13)式作为校正函数和使用(14)式作为校正函数三种配置。在使用(13)式和(14)式作为校正函数时，阈值 λ 都设为 0.7。

Table 5. Notations of LOVSEN algorithm with different parameters.

	do not use filter function	employ equation (13)	employ equation (14)
$k=3$	LOVSEN(0)-3	LOVSEN(1)-3	LOVSEN(2)-3
$k=5$	LOVSEN(0)-5	LOVSEN(1)-5	LOVSEN(2)-5

此外，文中给出的错误率以及标准差均来自于 10 次 10 倍交叉验证（10-Fold Cross Validation）。

4.2 校正函数对算法的影响

为测试不同的校正函数对算法的影响，本文使用了 20 个人工数据集。每一个数据集是由三个在二维平面上正态分布的类别的样本组成，共 450 个样本。20 个数据集中不同类别的分布的中心相同，通过递增正态分布的标准差来产生不同的数据集。标准差的递增使得各个类别的边界相互融合，形成交迭噪音。第 1 个数据集如 Fig. 2a)所示，各个类别能够被清晰地分辨。随着噪音的递增，第 20 个数据集如 Fig. 2b)所示，各个类别间有很大的侵扰，难以划分边界。

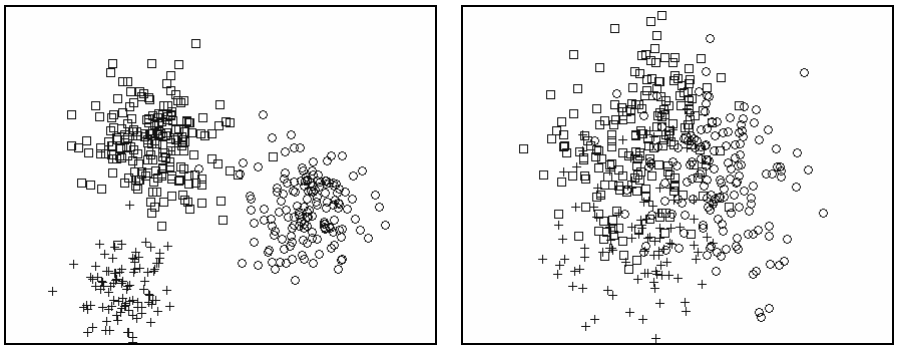


Fig. 2. a) Dataset 1 with lowest overlap noise. b) Dataset 20 with highest overlap noise.

Fig. 3 比较了当 k =3 时，三种配置的 LOVSEN 算法随着交迭噪音递增的表现（k =5 时的表现与此相似）。其中，在每一个数据集上的每一次比较，三种配置的算法都是用了训练好的相同的 20 个学习器。当噪音很小时，即在前 5 个数据集上，各种配置几乎没有差别。当噪音较大时，使用校正函数则有更好的效果。而且随着噪音的增大，LOVSEN(1)-3 与 LOVSEN(0)-3 的差距也在增大。说明校正函数在抗噪上有一定的作用，并且 LOVSEN(1)-3 的抗噪能力最强，LOVSEN(2)-3 次之。

然而在真实数据集中，并没有出现如此高的交迭噪音。Table 6 列出了 LOVSEN 算法的六种配置在 UCI 数据集上的表现，以不同的 k 值分为两组。注意到在所有数据集上，经显著程度为 0.05 的成对双尾 t 检验（pairwise two-tailed t test），在 k =3 的一组，LOVSEN(0)-3 在 10 个数据集（50%）上有显著优势，LOVSEN(1)-3 在 7 个数据集（35%）上有显著优势，LOVSEN(2)-3 在 6 个数据集（30%）上有显著优势；在 k =5 的一组，LOVSEN(0)-5 在 8 个数据集（40%）上有显著优势，LOVSEN(1)-5 在 7 个数据集（35%）上有显著优势，LOVSEN(2)-5 在 5 个数据集（25%）上有显著优势。另外，Table 6 中的数据显示，在 k=3 的组中，LOVSEN(2)-3 的性能介于 LOVSEN(0)-3 和 LOVSEN(1)-3 之间（最少成为其中最差的）在 k=5 的组中也有同样的现象，也就是说使用(14)式作为校正函数的效果较为稳定。

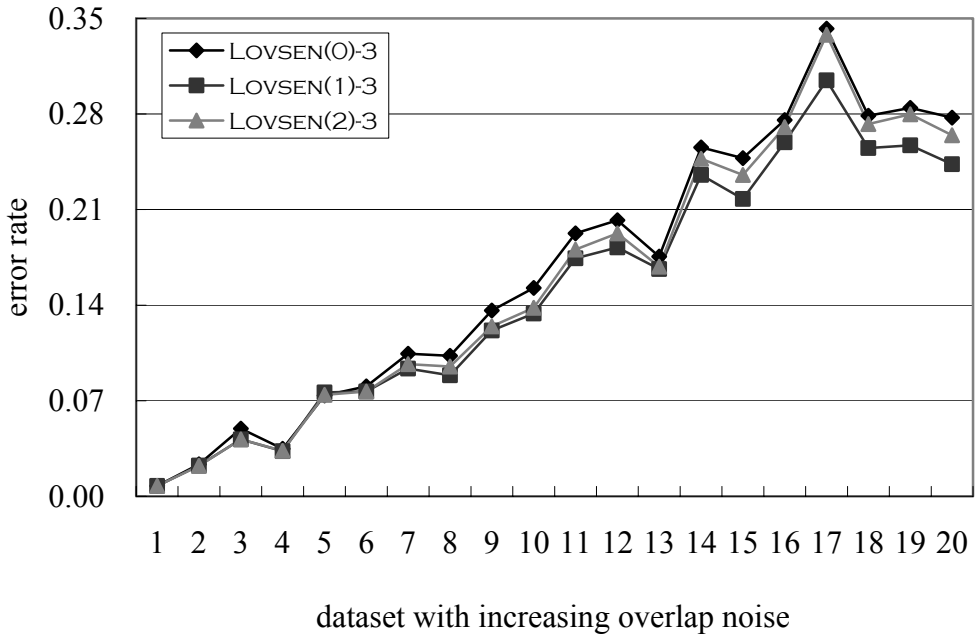


Fig. 3. Error ratios of Lovsen with different filters. The x-values are the datasets, increasing with noise.

4.3 LOVSEN 与其他算法的比较

由于上面的实验得出 LOVSEN(2)-3 和 LOVSEN(2)-5 较为稳定，下面将它们与单个 J48 决策树、Bagging[2]、AdaBoost.M1[14]、GASEN-b[38]进行比较。Bagging、AdaBoost.M1 均使用 Weka 的实现。Bagging、GASEN-b、LOVSEN(2)-3 和 LOVSEN(2)-5 的集成个数都设为 20，AdaBoost.M1 的最大集成个数设为 20，且使用重取样（resampling）方法。对于每一个数据集的每一次比较，Bagging、GASEN-b、LOVSEN(2)-3 和 LOVSEN(2)-5 都使用训练好的相同的 20 个 J48 决策树。

各种算法在 20 个 UCI 数据集上进行了实验。Table 7 列出了这 6 种算法在每个数据集上的交叉验证的平均错误率和标准差。Table 8 列出了 LOVSEN(2)的两种配置与其他算法比较的双尾成对 t 检验的结果，表格中的“win”代表 LOVSEN(2)在 0.05 的显著程度下优于其它算法，

“tie”代表无显著差别，“loss”代表 LOVSEN(2)在 0.05 的显著程度下逊于其它算法。Table 8 中最后一行为对这 20 次比较的符号检验的结果，结果表示算法的相似度，当结果为 1 时两种算法最相似，当结果为 0 时两种算法最相异。

Table 6. Error of Lovsen algorithms with different settings, obtained with 10 times 10-fold cross validation, grouped by the value of *k*. Each data is shown in the form of “mean-error±standard-deviation”. The best one(s) of each group of each row is shown in bold style.

Dataset	LOVSEN(0)-3	LOVSEN(1)-3	LOVSEN(2)-3	LOVSEN(0)-5	LOVSEN(1)-5	LOVSEN(2)-5
anneal	**.0053**±.00130	.0116±.00088	.0105±.00114	**.0049**±.00143	.0110±.00060	.0100±.00141
autos	**.1464**±.01214	.1504±.01263	**.1463±**.01214	.1469±.01219	.1490±.01001	.1469±.01219
balance-scale	.2068±.00757	.2079±.00807	.2073±.00741	.1861±.00610	.1867±.00529	.1863±.00594
breast-cancer	.2861±.01766	**.2641±**.01196	.2842±.01838	.2826±.01167	**.2680**±.01101	.2847±.01369
breast-w	.0393±.00452	**.0354±**.00357	**.0357**±.00392	.0384±.00355	**.0349**±.00278	**.0348**±.00254
credit-a	.1461±.00958	**.1285±**.00665	.1309±.00624	.1467±.00840	**.1298**±.00682	.1328±.00651
credit-g	.2621±.00945	**.2563±**.00747	.2595±.00873	.2667±.00780	**.2607**±.00756	.2647±.00774
diabetes	.2623±.00878	**.2456±**.00815	.2596±.00840	.2695±.00848	**.2547**±.00962	.2665±.00796
glass	**.2762**±.01381	.2836±.01438	.2776±.01446	**.2784**±.02203	.2868±.01890	**.2793**±.02249
heart-c	.2122±.01273	**.2020**±.01793	.2050±.01780	.2152±.01165	**.1988**±.01999	.2039±.01891
ionosphere	**.0711**±.00474	.0726±.00692	**.0703**±.00503	.0714±.00583	.0729±.00803	**.0694**±.00685
kr-vs-kp	**.0061**±.00101	.0070±.00058	.0068±.00043	**.0058**±.00072	.0068±.00064	.0065±.00050
lymph	**.1721**±.02037	.1805±.01814	**.1727**±.02015	**.1875**±.02255	.1934±.02198	**.1854**±.02547
segment	**.0240**±.00154	.0257±.00150	.0252±.00165	**.0251**±.00138	.0268±.00193	.0261±.00165
sonar	.1681±.01942	.1714±.02191	.1680±.01941	.1817±.01325	.1816±.01716	.1816±.01324
soybean	.0787±.00648	.0783±.00566	.0788±.00607	**.0725±**.00398	.0766±.00597	.0753±.00496
vehicle	**.2647**±.00571	.2665±.00592	**.2649**±.00535	.2665±.01142	.2673±.01023	.2666±.01095
vote	.0389±.00325	**.0312**±.00171	.0320±.00283	.0407±.00223	**.0316**±.00208	.0324±.00294
vowel	**.0486**±.00470	.0506±.00446	**.0485**±.00470	**.0640**±.00666	.0670±.00658	**.0640**±.00665
zoo	**.0595**±.01267	.0649±.01255	.0621±.01158	**.0604**±.00987	.0658±.01022	.0630±.00902

与 J48 相比，LOVSEN(2)的两种配置均在 17 个数据集（85%）上（anneal, autos, balance-scale, breast-w, credit-a, credit-g, glass, heart-c, ionosphere, kr-vs-kp, lymph, segment, sonar, soybean, vehicle, vowel, zoo）显著优于 J48，在其他 3 个数据集（15%）上（breast-cancer, diabetes, vote）没有显著差别。符号检验的结果为 LOVSEN(2) 100.00%的优于 J48。

与 Bagging 相比，LOVSEN(2)-3 在 10 个数据集（50%）上（anneal, autos, ionosphere, kr-vs-kp, lymph, segment, sonar, vehicle, vowel, zoo）显著优于 Bagging，在 6 个数据集（30%）上（balance-scale, breast-cancer, credit-g, diabetes, heart-c, vote）显著逊于 Bagging，同时在其他 4 个数据集（20%）上（breast-w, credit-a, glass, soybean）没有显著区别；LOVSEN(2)-5 在 10 个数据集（50%）上（anneal, autos, ionosphere, kr-vs-kp, lymph, segment, sonar, soybean, vowel, zoo）显著优于 Bagging，在 6 个数据集（30%）上（balance-scale, breast-cancer, credit-g, diabetes, heart-c, vote）显著逊于 Bagging，同时在其他 4 个数据集（20%）上（breast-w, credit-a, glass, vehicle）

没有显著区别。符号检验的结果为 LOVSEN(2) 54.55%的优于 Bagging。

Table 7. Errors of each algorithm, obtained with 10 times of 10-fold cross validation. Each data is shown in the form of "mean-error±standard-deviation".

Dataset	J48	Bagging	AdaBoost.M1	GASEN-b	LOVSEN(2)-3	LOVSEN(2)-5
anneal	.0171±.00316	.0146±.00268	.0033±.00071	.0136±.00225	.0105±.00114	.0100±.00141
autos	.3024±.01747	.2151±.01088	.1210±.01988	.1903±.01640	.1463±.01214	.1469±.01219
balance-scale	.2216±.01059	.1523±.00478	.2212±.00701	.1547±.00558	.2073±.00741	.1863±.00594
breast-cancer	.2787±.01626	.2574±.01540	.3108±.02178	.2660±.01601	.2842±.01838	.2847±.01369
breast-w	.0540±.00470	.0359±.00208	.0328±.00297	.0380±.00417	.0357±.00392	.0348±.00254
credit-a	.1424±.00527	.1321±.00428	.1412±.00597	.1341±.00816	.1309±.00624	.1328±.00651
credit-g	.2804±.00443	.2492±.00608	.2742±.01142	.2570±.00952	.2595±.00873	.2647±.00774
diabetes	.2620±.01242	.2377±.01012	.2670±.01327	.2368±.00741	.2596±.00840	.2665±.00796
glass	.3485±.03479	.2822±.01345	.2371±.01405	.2792±.02092	.2776±.01446	.2793±.02249
heart-c	.2317±.01689	.1840±.00863	.1979±.01695	.1860±.01599	.2050±.01780	.2039±.01891
ionosphere	.1047±.01353	.0738±.00680	.0585±.00383	.0801±.00956	.0703±.00503	.0694±.00685
kr-vs-kp	.0092±.00083	.0074±.00047	.0037±.00036	.0064±.00073	.0068±.00043	.0065±.00050
lymph	.2464±.02458	.2141±.01757	.1715±.01761	.2042±.01577	.1727±.02015	.1854±.02547
segment	.0437±.00297	.0322±.00181	.0176±.00157	.0312±.00183	.0252±.00165	.0261±.00165
sonar	.2990±.01429	.2330±.01742	.1892±.01517	.2265±.01889	.1680±.01941	.1816±.01324
soybean	.0980±.00737	.0808±.00383	.0718±.00438	.0826±.00776	.0788±.00607	.0753±.00496
vehicle	.2995±.01035	.2692±.00689	.2346±.01007	.2675±.01223	.2649±.00535	.2666±.01095
vote	.0316±.00185	.0303±.00034	.0446±.00684	.0372±.00445	.0320±.00283	.0324±.00294
vowel	.2742±.01217	.1273±.00798	.0521±.00501	.1309±.00729	.0485±.00470	.0640±.00665
zoo	.0913±.01650	.0759±.01364	.0510±.00877	.0645±.01307	.0621±.01158	.0630±.00902

与 AdaBoost.M1 相比，LOVSEN(2)-3 在 8 个数据集（40%）上（balance-scale, breast-cancer, credit-a, credit-g, diabetes, sonar, vote, vowel）显著优于 AdaBoost.M1，在 10 个数据集（50%）上（anneal, autos, breast-w, glass, ionosphere, kr-vs-kp, segment, soybean, vehicle, zoo）显著逊于 AdaBoost.M1，同时在其他 2 个数据集（10%）上（heart-c, lymph）没有显著区别；LOVSEN(2)-5 在 5 个数据集（25%）上（balance-scale, breast-cancer, credit-a, credit-g, vote）显著优于 AdaBoost.M1，在 12 个数据集（60%）上（anneal, autos, breast-w, glass, ionosphere, kr-vs-kp, lymph, segment, soybean, vehicle, vowel, zoo）显著逊于 AdaBoost.M1，同时在其他 3 个数据集（15%）上（diabetes, heart-c, lymph, sonar）没有显著区别。符号检验的结果为 LOVSEN(2)-3 18.55%的逊于 AdaBoost.M1，LOVSEN(2)-5 85.65%的逊于 AdaBoost.M1。

与 GASEN-b 相比，LOVSEN(2)-3 在 10 个数据集（50%）上（anneal, autos, breast-w, ionosphere, lymph, segment, sonar, soybean, vote, vowel）显著优于 GASEN-b，在 5 个数据集（25%）上（balance-scale, breast-cancer, diabetes, heart-c, kr-vs-kp）显著逊于 GASEN-b，同时在其他 5 个数据集（25%）上（credit-a, credit-g, glass, vehicle, zoo）没有显著区别；LOVSEN(2)-5 在 10 个数据集（50%）上（anneal, autos, breast-w, ionosphere, lymph, segment, sonar, soybean, vote,

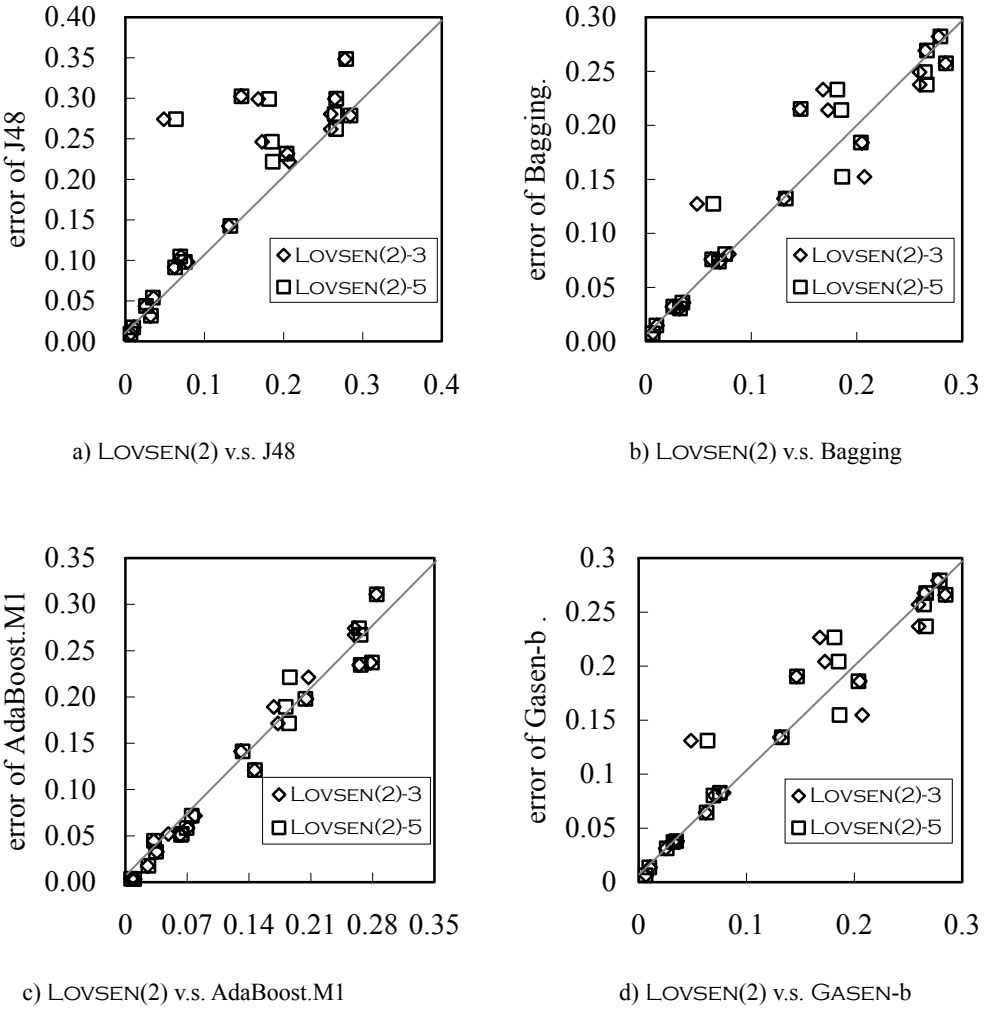
vowel）显著优于 GASEN-b，在 5 个数据集（25%）上（balance-scale, breast-cancer, credit-g, diabetes, heart-c）显著逊于 GASEN-b，同时在其他 5 个数据集（25%）上（credit-a, glass, kr-vs-kp, vehicle, zoo）没有显著区别。符号检验的结果为 LOVSEN(2) 69.82%的优于 GASEN-b。

Table 8. The *win/tie/loss* table comparing the performance of LOVSEN and other algorithms, obtained with pairwise two-tailed *t* test given α=.05. "*win*" means the performance of LOVSEN is significant better, "*tie*" means the performance is statistically equal, while "*loss*" means the performance of LOVSEN is significant worse.

Dataset	LOVSEN(2)-3				LOVSEN(2)-5			
	J48	Bagging	AdaBoost.M1	GASEN-b	J48	Bagging	AdaBoost.M1	GASEN-b
anneal	win	win	loss	win	win	win	loss	win
autos	win	win	loss	win	win	win	loss	win
balance-scale	win	loss	win	loss	win	loss	win	loss
breast-cancer	tie	loss	win	loss	tie	loss	win	loss
breast-w	win	tie	loss	win	win	tie	loss	win
credit-a	win	tie	win	tie	win	tie	win	tie
credit-g	win	loss	win	tie	win	loss	win	loss
diabetes	tie	loss	win	loss	tie	loss	tie	loss
glass	win	tie	loss	tie	win	tie	loss	tie
heart-c	win	loss	tie	loss	win	loss	tie	loss
ionosphere	win	win	loss	win	win	win	loss	win
kr-vs-kp	win	win	loss	loss	win	win	loss	tie
lymph	win	win	tie	win	win	win	loss	win
segment	win	win	loss	win	win	win	loss	win
sonar	win	win	win	win	win	win	tie	win
soybean	win	tie	loss	win	win	win	loss	win
vehicle	win	win	loss	tie	win	tie	loss	tie
vote	tie	loss	win	win	tie	loss	win	win
vowel	win	win	win	win	win	win	loss	win
zoo	win	win	loss	tie	win	win	loss	tie
win/tie/loss	17/3/0	10/4/6	8/2/10	10/5/5	17/3/0	10/4/6	5/3/12	10/5/5
sign test	.0000	.4545	.8145	.3018	.0000	.4545	.1435	.3018

从 Table 8 中可以得出，LOVSEN(2)完全优于 J48，在一定程度上优于 Bagging 和 GASEN-b，并且逊于 AdaBoost.M1。但是，AdaBoost.M1 的表现不稳定，在 2 个数据集（breast-cancer, vote）上，AdaBoost.M1 显著逊于 J48。而同时 LOVSEN(2)在所有数据集上都没有逊于 J48，这说明 LOVSEN(2)比 AdaBoost.M1 更稳定。根据 Table 7 各个算法在 20 个数据集上的错误率，Fig. 4 分别表示出 LOVSEN(2)-3、LOVSEN(2)-5 与单个 J48 决策树、Bagging、AdaBoost.M1、GASEN-b 的比较。其中，纵轴分别为单个 J48 决策树、Bagging、AdaBoost.M1、GASEN-b 的错误率，横轴为 LOVSEN(2)-3 和 LOVSEN(2)-5 的错误率，数据点越靠左表示 LOVSEN(2)越优，靠右超过斜线的数据点表示 LOVSEN(2)错误率高于与其比较的算法。从图

中可以直观地看出，与 J48、Bagging、GASEN-b 相比，LOVSEN(2)有更低的错误率；而 AdaBoost.M1 与 LOVSEN(2)的性能相当。并且，LOVSEN(2)的两种配置之间的比较显示出 k=3 和 k=5 这两种取值没有使 LOVSEN(2)的性能产生显著的变化。



a) LOVSEN(2) v.s. J48　　b) LOVSEN(2) v.s. Bagging

c) LOVSEN(2) v.s. AdaBoost.M1　　d) LOVSEN(2) v.s. GASEN-b

Fig. 4. Comparison of errors of LOVSEN and other algorithms. The errors are normalized by error of J48.

4.4 选择规模的比较

选择性集成算法 GASEN-b 和 LOVSEN 都没有使用全部的学习器进行集成。Table 9 列出了通过 10 次 10 倍交叉验证得到的 GASEN-b、LOVSEN(2)-3 和 LOVSEN(2)-5 的集成规模。其中，规模最小的标记为粗体。在大部分的数据集（80%）上，GASEN-b 的规模最小，在另外 4 个数据集上（20%），LOVSEN(2)-5 的规模最小。需要注意的是，GASEN-b 只保存选择出的个体学习器，而 LOVSEN 需要保存所有的个体学习器，因此，GASEN-b 的实际规模始终小于 LOVSEN。Table 9 所比较的只是在进行预测时，不同算法所平均使用的个体学习器数目。

Table 9. Comparison of ensemble sizes. Least size of each dataset is shown in bold style.

Dataset	GASEN-b	LOVSEN(2)-3	LOVSEN(2)-5
anneal	**8.54**	19.59	19.44
autos	**9.74**	12.16	10.10
balance-scale	**9.76**	13.14	12.21
breast-cancer	**7.80**	12.45	12.25
breast-w	**8.13**	18.77	18.26
credit-a	**7.42**	16.24	15.16
credit-g	9.82	11.76	**9.67**
diabetes	**9.52**	11.60	9.65
glass	**8.71**	11.47	10.09
heart-c	**9.18**	13.52	11.93
ionosphere	**7.42**	17.81	16.62
kr-vs-kp	**7.76**	19.73	19.62
lymph	**8.67**	13.22	11.87
segment	**9.73**	18.68	18.13
sonar	10.92	12.45	**9.84**
soybean	**8.33**	17.87	17.32
vehicle	11.95	11.28	**9.38**
vote	**7.83**	19.62	19.40
vowel	13.33	12.28	**9.42**
zoo	**11.67**	18.11	17.52

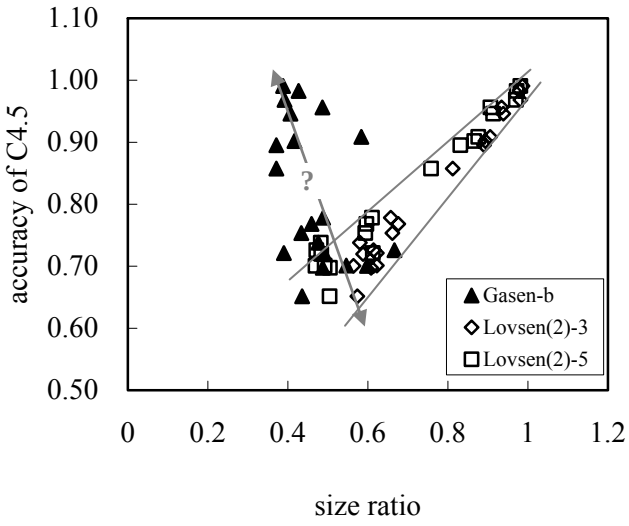


Fig. 5. Size ratio of ensembles compared with accuracy of J48.

有趣的是，观察在每个数据集上的规模和个体学习器的精度发现，LOVSEN 的集成规模与个体学习器的精度有一定的关系。Fig. 5 显示了个体学习器的精度与 GASEN-b、LOVSEN(2)-3 和 LOVSEN(2)-5 的集成规模比率（即集成规模/单个学习器总数）的关系。由图

中可以观察到，LOVSEN 的两种配置都显示出与个体学习器的平均精度成正比关系。这一现象可以从 LOVSEN 算法的选择策略上做出解释。LOVSEN 选择那些在局部上表现优秀的学习器，而当学习器的全局表现也较为优秀时，即学习器的精度较高时，局部表现优秀的学习器也较多，集成规模就较大。

而对于 GASEN，其集成规模始终在大约 0.4 到 0.7 之间，这说明 GASEN 在个体学习器精度与差异度之间达到了比较好的折中，可以比较有效地选择出少量的个体学习器来获得较好的泛化性能。

d da 左地喏惑 中国人芪械咖啡 嗲
不可能是这种情况的啦

5、总结

反馈 地器大吃大喝
口中咕咕

本文基于 Zhou 等人[39]提出的选择性集成思想，通过分析局部化与泛化能力的关系，提出了一种新的选择性集成方法 LOVSEN。LOVSEN 在对具体样本进行预测时，根据该样本的近邻，动态选择合适的学习器构成集成。以 J4.8 决策树作为基学习器的实验表明，LOVSEN 具有较高的泛化能力和较为稳定的性能。

LOVSEN 算法有两个参数需要确定。一个是近邻数 k，用于确定局部区域的范围。在实验中比较了 k=3 和 k=5 两种配置，结果表明这两种配置对算法没有很大的影响。但是不保证其他的 k 值对算法会有较大的影响。另一个参数是校正函数 F，在实验中比较了两种校正函数和不使用校正函数对算法的影响。

以下几个方面的内容值得进一步研究：

1) LOVSEN 使用了 HVDM 来度量离散值之间的距离。利用其他最近发现的离散属性距离度量方法，例如 SDM[25]以及使用样本流形（manifold）上的距离度量[34]，是否能够使算法更准确地寻找出近邻样本。
2) 是否有其他更稳定的校正函数，以及校正函数引入的阈值参数 λ 对算法会造成什么样的影响。
3) 当校正函数不能完全提供无噪音的训练样本时，在 k 近邻上选择完全预测正确的个体学习器这一要求过于苛刻。是否存在其它选择方式，例如在 k 近邻上选择预测“基本正确”的个体学习器。
4) 是否存在其他的局部化方法，例如使用决策树对样本进行划分。

致谢：

感谢我的导师周志华教授对论文标题、组织结构、语言的指导意见以及对文中一些问题的指正，LOVSEN 的主要思想属于周老师。感谢机器学习组张敏灵硕士对本文的语言、内容、形式上的指导意见。

参考文献：

[1] Blake C L, Merz C J. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/ MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[2] Breiman L. Bagging predictors. Machine Learning, 1996, 24(2): 123-140.

[3] Breiman L. Bias, variance, and arcing classifiers. Technical Report, Department of Statistics, University of California at Berkeley, 1996.

[4] Breiman L. Randomizing outputs to increase prediction accuracy. Machine Learning, 2000, 40(3): 229-242.

[5] Bryll R, Gutierrez-Osuna R, Quek F. Attribute Bagging: improving accuracy of classifier ensembles by using random feature subsets. Pattern Recognition, 2003, 36(6): 1291-1302.

[6] Clemen R T. Combining forecasts: a review and annotated bibliography. International Journal of Forecasting, 1989, 5(4): 559-583.

[7] Dasarathy B V. Nearest Neighbor Norms: NN Pattern Classification Techniques, Los Alamos, CA: IEEE Computer Society Press, 1991.

[8] Dietterich T G. Ensemble methods in machine learning. In: Proceedings of the First International Workshop on Multiple Classifier Systems, Cagliari, Italy, 2000, 1-15.

[9] Dietterich T G, Bakiri G. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research, 1995, 2: 263-286.

[10] Dietterich T G. Machine learning research: four current directions. AI Magazine, 1997, 18(4): 97-136.

[11] Dietterich T G, Kong E B. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical Report, Department of Computer Science, Oregon State University, Corvallis, Oregon, 1995.

[12] Dietterich T G, Lathrop R H, Lozano-Pérez T. Solving the multiple-instance problem with axis-parallel rectangles. Artificial Intelligence, 1997, 89(1-2): 31-71.

[13] Efron B, Tibshirani R. An Introduction to the Bootstrap, New York: Chapman & Hall, 1993.

[14] Freund Y, Schapire R E. A decision-theoretic generaliation of on-line learning and an application to boosting. Journal of Computer and System Sciences, 1997, 55(1): 119-139.

[15] Freund Y, Schapire R E. Experiments with a new boosting algorithm. In: Proceedings of the 13th International Conference on Machine Learning, San Francisco, Bari, Italy, 1996, 148-156.

[16] Freund Y, Schapire R E. Discussion of the paper "arcing classifiers" by Leo Breiman. The Annals of Statistics, 1998, 26(3): 824-832.

[17] Goldberg D E. Genetic Algorithm in Search, Optimization and Machine Learning, Boston, MA: Addison-Wesley, Reading, 1989.

[18] Grove A J, Schurmans D. Boosting in the limit: maximizing the margin of learned ensemble. In: Proceedings of the 15th National Conference on Artificial Intelligence, Madison, Wisconsin, 1998, 692-699.

[19] Hansen L K, Salamon P. Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990, 12(10): 993-1001.

[20] Hyafil L, Rivest R L. Constructing optimal binary decision trees is NP-Complete. Information Processing Letters, 1976, 5(1): 15-17.

[21] Kearns M, Valiant L G. Learning boolean formulae or factoring. Technical Report TR-1488, Cambridge, MA: Havard University Aiken Computation Laboratory, 1988.

[22] Kolen J F, Pollack J B. Back propagation is sensitive to initial conditions. In: Lippmann P R, Moody E J, Touretzky D S, Eds. Advances in Neural Information Processing Systems 3, San Francisco, CA: Morgan Kaufmann, 1991, 860-867.

[23] Krogh A, Vedelsby J. Neural network ensembles, cross validation, and active learning. In: Tesauro G, Touretzky D, Leen T, Eds. Advances in Neural Information Processing Systems 7, Cambridge, MA: MIT Press, 1995, 231-238.

[24] Kwok S W, Carter C. Multiple decision trees. In: Schachter R D, Levitt T S, Kannal L N, Lemmer J F, Eds. Uncertainty in Artificial Intelligence 4, Amsterdam: Elsevier Science, 1990, 327-335.

[25] Ludl M C. Symbolic distance measurements based on characteristic subspaces. In: Proceedings of the 7th European Conference for Principles of Data Mining and Knowledge Discovery, Dubrovnik, Croatia, 2003, 315-326.

[26] Mitchell T M. Machine Learning, New York: McGraw-Hill, 1997.

[27] Mjolsness E, DeCoste D. Machine learning for science: state of the art and future prospects. Science, 2001, 293: 2051-2055.

[28] Parmanto B, Munro P W, Doyle H R. Improving committee diagnosis with resampling techniques. In: Touretzky D S, Mozer M C, Hesselmo M E, Eds. Advances in Neural Information Processing Systems 8, Cambridge, MA: MIT Press, 1996, 882-888.

[29] Quinlan J R. C4.5: Programs for Machine Learning, San Mateo, CA: Morgan Kaufmann, 1993.

[30] Ricci F, Aha D W. Extending local learners with error-correcting output codes. Technical Report AIC-97-001, Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, 1997.

[31] Schapire R E. The Strength of weak learnability. Machine Learning, 1990, 5(2): 197-227.

[32] Schapire R E. Using output codes to boost multiclass learning problems. In: Proceedings of the 14th International Conference on Machine Learning, Nashville, Tennessee, 1997, 313-321.

[33] Schapire R E, Freund Y, Bartlett Y, Lee W S. Boosting the margin: a new explanation for the effectiveness of votiong methods. The Annals of Statistics, 1998, 26(5): 1651-1686.

[34] Tenenbaum J B, Silva V, Langford J C. A global geometric framework for nonlinear

dimensionality reduction. Science, 2000, 290: 2319-2323.

[35] Tumer K, Oza N C. Input decimated ensembles. Pattern Anal Applic, 2003, 6: 65-77.

[36] Wilson D R. Martinez T R. Improved heterogeneous distance functions. Journal of Artificial Intelligence Research, 1997, 6: 1-34.

[37] Witten I H, Frank E. Data Mining: Practical Machine Learning Tools With Java Implementations, San Francisco, CA: Morgan Kaufmann, 1999.

[38] Zhou Z-H, Tang W. Selective ensemble of decision trees. In: Wang G, Liu Q, Yao Y, Skowron A, Eds. Lecture Notes in Artficial Intelligence 2639, Berlin: Springer, 2003, 476-483.

[39] Zhou Z-H, Wu J, Tang W. Ensembling neural networks: many could be better than all. Artificial Intelligence, 2002, 137(1-2): 239-263.

da