ELM 网络结构自适应正交搜索算法

徐 睿[1] 梁 循[1] 马跃峰[2] 齐金山[3]

[1](中国人民大学信息学院 北京 100872)

[2](曲阜师范大学信息科学与工程学院 山东 日照 273100)

[3](淮阴师范学院计算机科学与技术学院 江苏 淮安 223300)

摘 要 由于具有灵活的非线性建模能力和良好的模式识别能力,单隐藏层前馈神经网络(Single Hidden Layer Feedforward Neural Network,SLFN)一直是机器学习和数据挖掘领域关注的焦点.众所周知,网络结构是影响 SLFN 泛化能力的重要因素之一.给定一个具体应用,如何在训练过程中自动选取最优的隐节点个数,仍是一大挑战.极限学习机(Extreme Learning Machine,ELM)通过随机生成隐藏层节点参数,并利用最小二乘法求解输出层权值的方式来训练 SLFN,在一定程度上克服了传统的基于梯度类学习方法收敛速度慢、容易陷入局部最小值等问题.然而,ELM 仍需要人为确定隐节点个数,不仅过程繁琐,而且无法保证得到最优或者次优的网络结构.在不影响泛化能力的前提下,为了进一步降低网络的复杂度,本文对 ELM 进行了改进,通过将网络结构学习转化为子集模型选择,提出了一种隐节点自适应正交搜索方法.首先,利用标准 ELM 构建隐节点候选池.然后,采用正交前向选择算法选择与网络期望输出相关度最大的候选隐节点加入到模型中.同时,每向前引入一个新的隐节点,就要向后对已选入的隐节点进行逐个检查,将变得不重要的隐节点从网络中删除.最后,设计了一种增强的向后移除策略来纠正前面步骤中所犯的错误,进一步剔除模型内残留的冗余隐节点.本文方法充分考虑了隐节点间的内在联系和相互影响,实验结果表明,该方法不仅具有良好的泛化性能,而且能够产生比较紧凑的网络结构.

关键词 子集模型选择;紧凑网络结构;极限学习机;正交前向选择;正交后向移除;颜色恒常性计算

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2021.01888

Adaptive Orthogonal Search for Network Structure Learning of ELM

XU Rui[1] LIANG Xun[1] MA Yue-Feng[2] QI Jin-Shan[3]

[1](*School of Information, Renmin University of China, Beijing* 100872)

[2](*School of Information Science and Engineering, Qufu Normal University, Rizhao, Shandong* 273100)

[3](*School of Computer Science and Technology, Huaiyin Normal University, Huaian, Jiangsu* 223300)

Abstract In the past several decades, Single Hidden Layer Feedforward Neural Network (SLFN) has drawn a large amount of attention in the field of machine learning, data mining and pattern recognition, due to its unique characteristics, i.e., learning capability from the input samples, and universal approximation capability for complex nonlinear mappings. Although SLFN has been investigated extensively from both theoretical and application aspects, it is still quite challenging to automatically determine a suitable network architecture for solving a specific task so that the resulting learner model can achieve sound performance for both learning and generalization. Extreme Learning Machine (ELM) is a powerful learning scheme for generalized SLFN with fast learning speed and has been widely used for both regression and classification.

收稿日期:2020-03-10;在线发布日期:2021-04-02.本课题得到国家自然科学基金(62072463,71531012)、国家社会科学基金重大项目(18ZDA309);北京市自然科学基金(4172032)、京东商城电子商务研究项目(413313012)、北大方正集团有限公司数字出版技术国家重点实验室开放课题、江苏省高校自然科学基金项目(19KJB520024)、江苏省大学生实践创新训练计划项目(201910323057Y)资助.

徐 睿,博士研究生,主要研究方向为图像处理、机器学习. E-mail: xurui1064@ruc.edu.cn. **梁 循**(通信作者),博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为数据挖掘、神经网络、社会计算. E-mail: xliang@ruc.edu.cn. **马跃峰**,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为支持向量机、神经网络. **齐金山**,博士,副教授,主要研究方向为社会计算、数据挖掘.

The hidden node parameters of ELM need not be exhaustively tuned during training, but assigned with random values simply, and the output weights are then analytically determined by solving a linear equation system using the generalized inverse method. However, for ELM, the suitable number of hidden nodes is usually pre-determined by trial and error, which may be tedious in some applications and does not guarantee that the selected network size will be close to optimal or will generalize well. Therefore, how to choose a parsimonious structure for ELM and to present a good capacity of generalization is the main objective of this paper. By formulating the learning problem as a subset model selection, we present an adaptive orthogonal search method to address the architectural design of ELM (referred to as AOS-ELM) for regression problems. In AOS-ELM, the hidden nodes can be deleted or recruited dynamically according to their significance to network performance, so that the network architecture can be self-configurable. More precisely, we first randomly generate a large number of hidden nodes using preliminary ELM as the candidate reservoir. Then, the hidden node output vector that has the highest correlation with the target output is selected from the candidates and added to the existing network by orthogonal forward selection in each step. Meanwhile, after a new hidden node is added to the set of selected variables, orthogonal backward elimination is commenced to see if any of the previously selected hidden nodes can be deleted without appreciably increasing the squared error. The procedure stops when no further additions or deletions are possible which satisfy the criteria. Finally, an enhanced backward refinement is implemented to correct mistakes made in earlier steps, so that the redundant hidden nodes are able to be deleted from the model as much as possible, and then the network complexity can be further reduced. To sum up, the proposed method can take into account the intrinsic connections and interactions between the hidden nodes, therefore offers a potential for finding the parsimonious network solutions that will fit the data. We demonstrate effective performance and superiority of the proposed method with experiments on several benchmark regression problems as well as two different color constancy tasks. Simulation results show that our method not only obtains a similar or higher learning accuracy than the preliminary ELM and other well-known constructive and pruning ELMs with a small number of hidden nodes, but also achieves better or comparable illuminant estimates over most of the test error metrics in comparison to several state-of-the-art color constancy algorithms.

Keywords　subset model selection; parsimonious network structure; extreme learning machine; orthogonal forward selection; orthogonal backward elimination; color constancy computation

1　引　言

单隐藏层前馈神经网络由于结构简单且具有较强的学习能力，只要隐藏层节点足够多，就能够在闭区间上以任意精度逼近任何一个多元非线性函数[1]，因此在工业界和学术界获得了广泛应用和深入研究. 过去的几十年里基于梯度下降的学习方法被广泛用于训练神经网络，如 BP 神经网络，其权值的修正是沿着误差性能函数梯度的反方向进行的. 这种学习方法在训练过程中需要同时对网络的输入和输出权值进行迭代调整，导致算法的收敛速度慢，容易陷入局部最小值，影响整个网络的泛化性能. 为了解决上述问题，Huang 等人[2]提出了一种简单高效的 SLFN 学习算法，称为极限学习机(Extreme Learning Machine，ELM). ELM 的训练过程主要分为两个阶段. 第一阶段，随机选择输入权值和隐藏层偏置，利用隐藏层神经元的激活函数将输入向量空间转换到隐藏层空间，又称为 ELM 特征空间. 在 ELM 中，隐藏层到输出层的映射是线性的，因此在第二阶段只需使用最小二乘法计算输出权值. 文献[3]从理论上证明了具有非线性分段连续激活函数的 SLFN，即使随机生成隐藏层节点参数，ELM 仍然能够保持其通用逼近能力. ELM 倾向于以极快的

学习速度和较少的人为干预提供更好的泛化性能，已被广泛应用于图像处理和生物医学等领域[4-6].

然而，在一些实际应用中，由于隐藏层节点参数随机生成，ELM 通常需要更多的隐节点才能达到和传统的神经网络学习方法相当的性能. 研究发现，ELM 网络中的一些隐藏节点在网络输出中所起到的作用非常小，导致网络结构的复杂度增加[7]. 从另一个角度来看，在 ELM 中隐藏层节点个数是唯一需要人为确定的参数. 隐藏层节点数太少会导致欠拟合，太多的隐藏层节点容易导致过拟合，从而影响模型的泛化能力. 因此，针对不同的任务，如何选取最优的隐节点组合，使得网络结构紧凑的同时仍保持良好的泛化性能，成为当前神经网络最活跃的研究方向之一.

标准 ELM 的隐节点数通常是通过反复试验得出的. 这种方法虽然简单直观，但不仅计算量大，而且无法保证所选网络的结构达到或者接近最优. 近年来，一些研究人员从增量和剪枝两个方面对 SLFN 网络结构优化问题进行了研究[8-9]. 一般而言，增量构造法从一个小的初始网络开始，然后在训练过程中不断增加新的隐节点，直到满足一定的停止准则. 文献[3]提出了一种增量式极限学习机(Incremental ELM，I-ELM)，该算法将随机生成的隐节点逐个加入到隐藏层中，当隐藏层节点数达到最大值或训练误差小于用户预先设定的期望值时停止. I-ELM 在添加新的隐节点时会保持现有隐节点的输出权值固定不变. 凸增量极限学习机(Convex I-ELM，CI-ELM)[10]在 I-ELM 的基础上，结合 Barron 凸优化理论，在添加新的隐节点后对已存在的隐节点的输出权值进行更新. 因此，CI-ELM 比 I-ELM 具有更快的收敛速度. 文献[11]提出了一种误差最小化极限学习机(Error Minimized ELM，EM-ELM)，与 I-ELM 和 CI-ELM 逐个添加隐节点不同，EM-ELM 允许隐节点逐块增加，并利用分块矩阵增量地更新网络中所有隐节点的输出权值，从而进一步加快了网络的收敛速度. 文献[12]将施密特正交化过程引入到 I-ELM 中，将其扩展到多分类任务，提出了一种正交化的增量极限学习机(Orthogonal I-ELM，OI-ELM). 由于 OI-ELM 每次新增的隐节点输出向量需要和已存在的隐节点输出向量正交，因此隐藏层节点的个数必须小于训练样本数. 文献[13]使用改进的粒子群优化算法来选择 EM-ELM 的输入权值和隐藏层偏置，进一步提高了模型的数值稳定性. 由于改进的粒子群优化算法中引入了大量的超参数需要调整，所以不可避免地增加了模型的计算复杂度. 文献[14]在 OI-ELM 的基础上提出了一种基于驱动量的正交增量极限学习机(Driving Amount OI-ELM，DAOI-ELM)，通过将网络的训练误差反馈到新增隐节点的输出向量中来减小网络的规模，能够有效提高田纳西伊—斯特曼过程故障诊断的精度. 然而，在上述所有的增量 ELMs 的实现中，隐节点一经加入就不能删除，当新的隐节点被添加到网络中时，可能会导致之前选中的隐节点变得不重要. 此外，隐节点的数量会随着迭代步骤单调递增. 这样，在需要多次迭代的情况下，最终会产生一个大规模网络，而其中一些隐节点可能在网络输出中起到非常小的作用.

相比之下，剪枝法为动态确定适当的网络结构提供了另外一种途径. 剪枝法首先训练一个大于实际需求的网络，然后在学习过程中逐步删除冗余的隐节点，直到模型性能趋于稳定. 文献[15]提出了一种用于模式分类任务的剪枝极限学习机(Pruned ELM，P-ELM)，该算法首先训练一个具有大量隐节点的网络，然后使用卡方统计准则和信息增益剔除与类标签相关性较低的隐节点. 针对不相关训练数据集对网络性能的影响，文献[16]提出了最优剪枝极限学习机(Optimally Pruned ELM，OP-ELM). 该算法首先利用标准 ELM 构建一个 SLFN，然后通过多响应稀疏回归对隐节点进行排序，最终的模型由留一交叉验证法确定. 文献[17]提出了一种基于遗传算法的剪枝极限学习机，通过构造基于最小化分类错误率和剪枝代价的多目标适应度函数来控制模型的复杂度，其中剪枝代价由剪枝后隐藏神经元的个数与候选隐藏神经元总数的比率度量. 针对类别不平衡问题，文献[18]提出了一种基于相关系数的剪枝极限学习机. 该算法的基本思想是利用马修斯相关系数作为评价指标来反映每个隐节点的重要程度，按照重要性对隐节点输出向量进行降序排序，并删除小于指定阈值的隐节点输出向量. 然而，对于上述剪枝类算法，在实际应用中人们通常不知道应该指定多大的初始网络. 其次，剪枝法的大部分训练时间都花在了不必要的大规模网络上，并且很容易产生过拟合[19].

本文从子集模型选择的角度出发，提出了一种 ELM 网络结构自适应正交搜索算法(AOS-ELM)，在保持良好泛化性能的同时，进一步降低网络的复杂度. 首先，使用标准 ELM 随机生成一组隐节点构建候选集；然后，正交化的前向选择和后向移除交

替执行，使重要的隐节点组合都被引入到优化模型中，同时尽可能地剔除不重要的隐节点；最后，采用增强的后向移除算法对活跃集中所有的隐节点逐个进行检查，进一步删除网络中残留的冗余隐节点，从而获得一个更加紧凑的网络结构. 本文的主要贡献如下：(1) 提出了一种前向选择和后向移除相结合的隐节点动态调整方法，能够反映隐节点间的内在联系和相互作用的动态关系；(2) 设计了一种增强的后向移除策略，具有对前一步错误进行修正的能力，从而获得相对最优的隐节点组合；(3) 该方法具有较大的推广空间，可以很容易地应用到广义线性回归和非线性回归的变量筛选这类组合优化问题中.

2　相关工作

2.1　标准极限学习机

给定任意 N 个不同的训练样本 $\{(\boldsymbol{x}_j,\boldsymbol{t}_j)\}_{j=1}^{N}\subset R^n\times R$，其中 $\boldsymbol{x}_j$ 和 $\boldsymbol{t}_j$ 分别为特征向量和与之对应的期望输出向量. 一个含有 L 个隐节点的 SLFN 关于输入向量 $\boldsymbol{x}_j$ 的输出可以表示为

$$f_L(\boldsymbol{x}_j)=\sum_{i=1}^{L}\beta_i G(\boldsymbol{w}_i,b_i,\boldsymbol{x}_j) \tag{1}$$

其中，$\boldsymbol{w}_i\in R^n$，$j=1,2,\cdots,N$，$G(\boldsymbol{w}_i,b_i,\boldsymbol{x}_j)$ 表示与输入 $\boldsymbol{x}_j$ 相对应的第 i 个隐节点的输出，$\beta_i\in R$ 表示连接第 i 个隐藏层节点和输出层节点的输出权值.

当隐藏层神经元为加性节点时，激活函数 $g(x)$ 定义如下：

$$G(\boldsymbol{w}_i,b_i,\boldsymbol{x}_j)=g(\boldsymbol{w}_i\cdot\boldsymbol{x}_j+b_i),\ b_i\in R \tag{2}$$

其中，$\boldsymbol{w}_i=[w_{i1},w_{i2},\cdots,w_{in}]^{\mathrm{T}}$ 表示连接第 i 个隐藏层节点和输入层节点的输入权值，b_i 表示第 i 个隐藏层神经元的偏置，$\boldsymbol{w}_i\cdot\boldsymbol{x}_j$ 表示 $\boldsymbol{w}_i$ 和 $\boldsymbol{x}_j$ 之间的内积.

当隐藏层神经元为 RBF 节点时，激活函数 $g(x)$ 定义如下：

$$G(\boldsymbol{w}_i,b_i,\boldsymbol{x}_j)=g(b_i\|\boldsymbol{x}_j-\boldsymbol{w}_i\|),\ b_i\in R^+ \tag{3}$$

其中，$\boldsymbol{w}_i=[w_{i1},w_{i2},\cdots,w_{in}]^{\mathrm{T}}$ 和 b_i 分别表示第 i 个节点的中心和宽度，R^+ 表示正实数集合.

当上述 SLFN 能够零误差地逼近 N 个训练样本时，则有 $f_L(\boldsymbol{x}_j)=t_j$，$j=1,2,\cdots,N$，表示成矩阵的形式为

$$\boldsymbol{H}\boldsymbol{\beta}=\boldsymbol{T} \tag{4}$$

其中，$\boldsymbol{T}=[t_1,t_2,\cdots,t_N]^{\mathrm{T}}$ 表示训练样本期望输出矩阵，$\boldsymbol{\beta}=[\beta_1,\beta_2,\cdots,\beta_L]^{\mathrm{T}}$ 表示隐藏层和输出层之间的权值矩阵，$\boldsymbol{H}$ 为隐藏层输出矩阵[20]，其元素满足

$$\begin{aligned}\boldsymbol{H}&=\boldsymbol{H}(\boldsymbol{w}_1,\cdots,\boldsymbol{w}_L,b_1,\cdots,b_L,\boldsymbol{x}_1,\cdots,\boldsymbol{x}_N)\\&=\begin{bmatrix}G(\boldsymbol{w}_1,b_1,\boldsymbol{x}_1)&\cdots&G(\boldsymbol{w}_L,b_L,\boldsymbol{x}_1)\\\vdots&\ddots&\vdots\\G(\boldsymbol{w}_1,b_1,\boldsymbol{x}_N)&\cdots&G(\boldsymbol{w}_L,b_L,\boldsymbol{x}_N)\end{bmatrix}\\&=[\boldsymbol{h}_1,\cdots,\boldsymbol{h}_L]\end{aligned} \tag{5}$$

上式中，$\boldsymbol{H}$ 的第 i 列表示第 i 个隐节点关于输入空间样本 $\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N$ 的隐藏层输出，H 的第 j 行表示第 j 个输入向量 $\boldsymbol{x}_j$ 对应的所有隐藏层节点输出.

根据 ELM 学习理论[3]，隐藏层节点参数 $(\boldsymbol{w}_i,b_i)$ 可以基于连续抽样概率分布随机生成. 这样，在给定训练集之后，隐藏层输出矩阵 $\boldsymbol{H}$ 实际上是已知的，训练 SLFN 就转化为求解线性系统 (4) 关于 $\boldsymbol{\beta}$ 的最小范数最小二乘解

$$\hat{\boldsymbol{\beta}}=\boldsymbol{H}^{\dagger}\boldsymbol{T} \tag{6}$$

其中，$\boldsymbol{H}^{\dagger}$ 表示隐藏层输出矩阵 $\boldsymbol{H}$ 的 Moore-Penrose 广义逆[21].

2.2　问题定义和基础理论

当隐藏层输出矩阵 $\boldsymbol{H}$ 确定之后，SLFN 网络也就变成一个线性系统. 在这种情况下，隐节点的选择可以看作多元线性回归中子集模型的选择. 从代数的观点来看，式 (4) 可以改写为

$$\boldsymbol{T}=\boldsymbol{H}\boldsymbol{\beta}=\sum_{i=1}^{L}\beta_i\boldsymbol{h}_i \tag{7}$$

其中，$\boldsymbol{h}_i=[G(\boldsymbol{w}_i,b_i,\boldsymbol{x}_1),\cdots,G(\boldsymbol{w}_i,b_i,\boldsymbol{x}_N)]^{\mathrm{T}}$ 表示第 i 个隐节点关于输入向量 $\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N$ 的特征映射.

定理 1[22]. 给定一个广义线性系统 $\boldsymbol{T}=\boldsymbol{H}\boldsymbol{\beta}$，其中 $N\times L$ 矩阵 $\boldsymbol{H}$ 的列向量 $\boldsymbol{h}_1,\boldsymbol{h}_2,\cdots,\boldsymbol{h}_L$ 线性无关，$\boldsymbol{\beta}\in R^L$，若令 $\{\tilde{\boldsymbol{h}}_i\}_{i=1}^{L}$ 是由 $\{\boldsymbol{h}_i\}_{i=1}^{L}$ 所生成矩阵 $\boldsymbol{H}$ 的列空间 (记为 Col$\boldsymbol{H}$) 的一个标准正交基，则线性模型 $\boldsymbol{T}=\sum_{i=1}^{L}c_i\tilde{\boldsymbol{h}}_i$ 中的权值可以由 $c_i=\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_i$ $(i=1,2,\cdots,L)$ 计算.

定理 2[23]. 给定一个广义线性系统 $\boldsymbol{T}=\boldsymbol{H}\boldsymbol{\beta}$，其中 $\boldsymbol{H}\in R^{N\times L}$，Rank$(\boldsymbol{H})>0$，$\boldsymbol{\beta}\in R^L$. 如果 $\{\tilde{\boldsymbol{h}}_1,\tilde{\boldsymbol{h}}_2,\cdots,\tilde{\boldsymbol{h}}_k\}$ 是 Col$\boldsymbol{H}$ 的任意一个标准正交基，$1\le k\le L$，且 $\hat{\boldsymbol{T}}=\sum_{i=1}^{k}\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_i\tilde{\boldsymbol{h}}_i$ 为 $\boldsymbol{T}$ 在 Col$\boldsymbol{H}$ 上的正交投影，那么 $\boldsymbol{T}$ 可以唯一表示为 $\boldsymbol{T}=\hat{\boldsymbol{T}}+\boldsymbol{E}$，其中，$\hat{\boldsymbol{T}}\in$ Col$\boldsymbol{H}$，$\boldsymbol{E}\in$ (Col$\boldsymbol{H}$)$^{\perp}$，(Col$\boldsymbol{H}$)$^{\perp}$ 称为 Col$\boldsymbol{H}$ 的正交补.

定理 3[23]. 给定一个广义线性系统 $\boldsymbol{T}=\boldsymbol{H}\boldsymbol{\beta}$，其中 $\boldsymbol{H}\in R^{N\times L}$，Rank$(\boldsymbol{H})>0$，$\boldsymbol{\beta}\in R^L$. 若 $\hat{\boldsymbol{T}}$ 是 $\boldsymbol{T}$ 在 Col$\boldsymbol{H}$ 上的正交投影，那么 $\hat{\boldsymbol{T}}$ 是 Col$\boldsymbol{H}$ 中元素对 $\boldsymbol{T}$ 的最佳逼近，即对任意的 $\boldsymbol{T}'\in$ Col$\boldsymbol{H}$ 且 $\boldsymbol{T}'\neq\hat{\boldsymbol{T}}$，不等式

$\|\boldsymbol{T}'-\hat{\boldsymbol{T}}\|<\|\boldsymbol{T}-\boldsymbol{T}'\|$成立.

定理1～3的详细证明可以参考文献[22-23].定理1表明,若期望输出$\boldsymbol{T}$属于隐藏层输出矩阵$\boldsymbol{H}$的列空间,则$\boldsymbol{T}$可以表示为$\boldsymbol{T}=\boldsymbol{H}\boldsymbol{\beta}=\sum_{i=1}^{L}\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_i\tilde{\boldsymbol{h}}_i$,其中$\{\tilde{\boldsymbol{h}}_1,\tilde{\boldsymbol{h}}_2,\cdots,\tilde{\boldsymbol{h}}_L\}$是$R^N$中子空间$\mathrm{Col}\boldsymbol{H}$的一个标准正交基.特别地,当隐藏层节点的个数和训练样本的个数相等时,即$L=N$,矩阵$\boldsymbol{H}$可逆,$\boldsymbol{T}\in\mathrm{Col}\boldsymbol{H}$成立,线性系统能够零误差地逼近所有训练样本.定理2表明,当隐藏层节点的个数远小于训练样本的个数时,若$\boldsymbol{T}\notin\mathrm{Col}\boldsymbol{H}$,则可以通过正交分解得到$\boldsymbol{T}$的唯一表示$\boldsymbol{T}=\hat{\boldsymbol{T}}+\boldsymbol{E}$,且正交投影$\hat{\boldsymbol{T}}$仅依赖于$\mathrm{Col}\boldsymbol{H}$,并不依赖于$\hat{\boldsymbol{T}}=\sum_{i=1}^{k}\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_i\tilde{\boldsymbol{h}}_i$中使用的特殊基.定理3表明,正交投影$\hat{\boldsymbol{T}}$是矩阵$\boldsymbol{H}$列空间中的元素对$\boldsymbol{T}$的最佳逼近,此时线性模型的误差$\boldsymbol{E}$在$\hat{\boldsymbol{T}}$处取得最小值.

3 子集模型选择

首先使用标准ELM生成$\bar{L}$个候选隐节点$\boldsymbol{H}_{\bar{L}}=[\boldsymbol{h}_1,\boldsymbol{h}_2,\cdots,\boldsymbol{h}_{\bar{L}}]_{N\times\bar{L}}$.对每个特征映射向量$\boldsymbol{h}_l$和期望输出$\boldsymbol{T}$中心化,使得$\sum_{j=1}^{N}G(\boldsymbol{w}_l,b_l,\boldsymbol{x}_j)=0$,$\sum_{j=1}^{N}t_j=0$,$l=1,2,\cdots,\bar{L}$.为了方便起见,将候选隐节点的下标放入到候选集$\Lambda$内,该候选集初始化为$\Lambda=\{1,2,\cdots,\bar{L}\}$.同时,将每一步选中的隐节点对应的下标添加到活跃集Γ中,该活跃集初始化为$\Gamma^{(0)}=\varnothing$.设$\hat{\boldsymbol{T}}$为$\boldsymbol{T}$的最小二乘估计并初始化$\hat{\boldsymbol{T}}^{(0)}=\boldsymbol{0}$.根据上一节的讨论,一个含有$\bar{L}$个隐节点的SLFN可以表示为

$$\boldsymbol{T}=f_{\bar{L}}(\boldsymbol{x})=\boldsymbol{H}_{\bar{L}}\boldsymbol{\beta}_{\bar{L}}+\boldsymbol{E}=\sum_{l=1}^{\bar{L}}\beta_l\boldsymbol{h}_l(\boldsymbol{x})+\boldsymbol{E}\tag{8}$$

其中,$\boldsymbol{T}=[t_1,t_2,\cdots,t_N]^{\mathrm{T}}$为期望输出向量,$\boldsymbol{\beta}=[\beta_1,\beta_2,\cdots,\beta_{\bar{L}}]^{\mathrm{T}}$表示输出权值,$\boldsymbol{E}=[e_1,e_2,\cdots,e_N]^{\mathrm{T}}$为残差向量.

子模型选择的最终目标是从候选集中筛选出一组最优的隐藏节点$\boldsymbol{H}_m=(\boldsymbol{H}_{\bar{L}})_{\Gamma^{(m)}}=[\boldsymbol{h}_{l_1},\boldsymbol{h}_{l_2},\cdots,\boldsymbol{h}_{l_m}]$,$\Gamma^{(m)}=\{l_1,l_2,\cdots,l_m\}$,并确定相应的输出权值$\boldsymbol{\beta}_m=[\beta_{l_1},\beta_{l_2},\cdots,\beta_{l_m}]^{\mathrm{T}}$,以此构建一个具有良好泛化性能的紧凑型网络.

3.1 基于ELM的正交前向选择

正交前向选择(Orthogonal Forward Selection,OFS)算法的基本思想是从空集开始学习,每次从候选集中选择一个与当前残差相关度最高的隐节点加入到活跃集中[24].具体步骤如下:在初始步$k=0$时,对候选集中的每个隐节点输出向量计算其与当前残差的相关系数

$$c_l^{(k+1)}=\boldsymbol{h}_l^{\mathrm{T}}(\boldsymbol{T}-\hat{\boldsymbol{T}}^{(k)})=\boldsymbol{h}_l^{\mathrm{T}}\boldsymbol{T},\ l=1,2,\cdots,\bar{L}\tag{9}$$

将相关系数绝对值最大的候选隐节点输出向量的下标添加到活跃集中

$$\begin{cases}l_{k+1}=\arg\max\limits_{l\in\Lambda\backslash\Gamma^{(k)}}|c_l^{(k+1)}|\\ \Gamma^{(k+1)}=\Gamma^{(k)}\cup\{l_{k+1}\}\end{cases}\tag{10}$$

具有一个隐节点的线性系统的最小二乘估计为:

$$\begin{aligned}\hat{\boldsymbol{T}}^{(k+1)}&=\boldsymbol{H}_{k+1}\boldsymbol{H}_{k+1}^{\dagger}\boldsymbol{T}\\&=\boldsymbol{H}_{k+1}(\boldsymbol{H}_{k+1}^{\mathrm{T}}\boldsymbol{H}_{k+1})^{-1}\boldsymbol{H}_{k+1}^{\mathrm{T}}\boldsymbol{T}\\&=\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{l_{k+1}}\tilde{\boldsymbol{h}}_{l_{k+1}}\end{aligned}\tag{11}$$

其中,$\boldsymbol{H}_{k+1}=(\boldsymbol{H}_{\bar{L}})_{\Gamma^{(k+1)}}=[\boldsymbol{h}_{l_1}]$,$\boldsymbol{H}_{k+1}^{\dagger}=(\boldsymbol{H}_{k+1}^{\mathrm{T}}\boldsymbol{H}_{k+1})^{-1}\boldsymbol{H}_{k+1}^{\mathrm{T}}$,$\tilde{\boldsymbol{h}}_{l_{k+1}}=\boldsymbol{h}_{l_{k+1}}/\|\boldsymbol{h}_{l_{k+1}}\|_2$为$\boldsymbol{h}_{l_{k+1}}$的单位向量.

在第$k(1\le k\le\bar{L})$步,隐节点的选择过程如下:首先对候选隐节点$\boldsymbol{h}_l$关于当前活跃集中下标对应的向量组做正交化和单位化

$$\begin{cases}\boldsymbol{h}_l'=\boldsymbol{h}_l-\boldsymbol{H}_k(\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{H}_k)^{-1}\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{h}_l\\ \tilde{\boldsymbol{h}}_l=\boldsymbol{h}_l'/\|\boldsymbol{h}_l'\|_2\end{cases}\tag{12}$$

其中,$\boldsymbol{H}_k=(\boldsymbol{H}_{\bar{L}})_{\Gamma^{(k)}}=[\boldsymbol{h}_{l_1},\boldsymbol{h}_{l_2},\cdots,\boldsymbol{h}_{l_k}]$,$l\in\Lambda\backslash\Gamma^{(k)}$.

计算候选隐节点与当前残差的相关系数,将绝对值最大的相关系数对应的隐节点的下标加入到活跃集中

$$\begin{cases}c_l^{(k+1)}=\tilde{\boldsymbol{h}}_l^{\mathrm{T}}(\boldsymbol{T}-\hat{\boldsymbol{T}}^{(k)})=\tilde{\boldsymbol{h}}_l^{\mathrm{T}}\boldsymbol{T}\\ l_{k+1}=\arg\max\limits_{l\in\Lambda\backslash\Gamma^{(k)}}|c_l^{(k+1)}|\\ \Gamma^{(k+1)}=\Gamma^{(k)}\cup\{l_{k+1}\}\end{cases}\tag{13}$$

实际输出$\hat{\boldsymbol{T}}$在第$k+1$次的更新公式为

$$\begin{aligned}\hat{\boldsymbol{T}}^{(k+1)}&=\boldsymbol{H}_{k+1}\boldsymbol{H}_{k+1}^{\dagger}\boldsymbol{T}\\&=\boldsymbol{H}_{k+1}(\boldsymbol{H}_{k+1}^{\mathrm{T}}\boldsymbol{H}_{k+1})^{-1}\boldsymbol{H}_{k+1}^{\mathrm{T}}\boldsymbol{T}\end{aligned}\tag{14}$$

其中,$\boldsymbol{H}_{k+1}=(\boldsymbol{H}_{\bar{L}})_{\Gamma^{(k+1)}}=[\boldsymbol{H}_k\boldsymbol{h}_{l_{k+1}}]$.

为了提高计算效率,可以将式(14)中的矩阵$\boldsymbol{H}_{k+1}$进行QR分解$\boldsymbol{H}_{k+1}=\tilde{\boldsymbol{H}}_{k+1}\boldsymbol{U}_{k+1}$并代替

$$\begin{aligned}\hat{\boldsymbol{T}}^{(k+1)}&=\tilde{\boldsymbol{H}}_{k+1}\boldsymbol{U}_{k+1}(\boldsymbol{U}_{k+1}^{\mathrm{T}}\tilde{\boldsymbol{H}}_{k+1}^{\mathrm{T}}\tilde{\boldsymbol{H}}_{k+1}^{\mathrm{T}}\boldsymbol{U}_{k+1})^{-1}\boldsymbol{U}_{k+1}^{\mathrm{T}}\tilde{\boldsymbol{H}}_{k+1}^{\mathrm{T}}\boldsymbol{T}\\&=\tilde{\boldsymbol{H}}_{k+1}\tilde{\boldsymbol{H}}_{k+1}^{\mathrm{T}}\boldsymbol{T}\\&=\hat{\boldsymbol{T}}^{(k)}+\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{l_{k+1}}\tilde{\boldsymbol{h}}_{l_{k+1}}\end{aligned}\tag{15}$$

其中,$\tilde{\boldsymbol{H}}_{k+1}=[\tilde{\boldsymbol{h}}_{l_1},\tilde{\boldsymbol{h}}_{l_2},\cdots,\tilde{\boldsymbol{h}}_{l_{k+1}}]$是一个$N\times(k+1)$矩阵,其列形成$\mathrm{Col}(\boldsymbol{H}_{k+1})$的一个标准正交基,$\boldsymbol{U}_{k+1}$是一个$k+1$维的上三角可逆矩阵.

定理4. 在OFS中,误差项序列$\{\|\boldsymbol{E}_k\|\}$单调递减有下界,并且误差减小的速度是最快的,如果每

次选取与期望输出绝对相关度最大的隐节点加入到模型中.

证明.　记 $\xi=\|\boldsymbol{E}_k\|^2-\|\boldsymbol{E}_{k+1}\|^2$，由 $\|\boldsymbol{E}_{k+1}\|=\|\boldsymbol{T}-(\hat{\boldsymbol{T}}^{(k)}+\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{l_{k+1}}\tilde{\boldsymbol{h}}_{l_{k+1}})\|$，$\tilde{\boldsymbol{h}}_{l_{k+1}}\in(\mathrm{Col}(\boldsymbol{H}_k))^{\perp}$，$\hat{\boldsymbol{T}}^{(k)}\in\mathrm{Col}(\boldsymbol{H}_k)$，有

$$\begin{aligned}\xi&=\|\boldsymbol{E}_k\|^2-\|\boldsymbol{T}-(\hat{\boldsymbol{T}}^{(k)}+\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{l_{k+1}}\tilde{\boldsymbol{h}}_{l_{k+1}})\|^2\\&=\|\boldsymbol{E}_k\|^2-\|\boldsymbol{E}_k-\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{l_{k+1}}\tilde{\boldsymbol{h}}_{l_{k+1}}\|^2\\&=2\boldsymbol{E}_k^{\mathrm{T}}(\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{l_{k+1}}\tilde{\boldsymbol{h}}_{l_{k+1}})-(\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{l_{k+1}})^2\\&=2(\boldsymbol{T}-\hat{\boldsymbol{T}}^{(k)})^{\mathrm{T}}(\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{l_{k+1}}\tilde{\boldsymbol{h}}_{l_{k+1}})-(\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{l_{k+1}})^2\\&=(\tilde{\boldsymbol{h}}_{l_{k+1}}^{\mathrm{T}}\boldsymbol{T})^2>0,\end{aligned}$$

由此得 $\|\boldsymbol{E}_{k+1}\|<\|\boldsymbol{E}_k\|$，这表明误差项序列 $\{\|\boldsymbol{E}_k\|\}$ 单调递减，下界为 0.

由式(13)可知，当 $l_{k+1}=\arg\max\limits_{l\in\Lambda\backslash\Gamma^{(k)}}|c_l^{(k+1)}|$ 时，$|\tilde{\boldsymbol{h}}_{l_{k+1}}^{\mathrm{T}}\boldsymbol{T}|$ 取最大值，于是可得 $\xi=\xi_{\max}=(\tilde{\boldsymbol{h}}_{l_{k+1}}^{\mathrm{T}}\boldsymbol{T})^2$，即 $\|\boldsymbol{E}_{k+1}\|=\|\boldsymbol{T}-(\hat{\boldsymbol{T}}^{(k)}+\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{l_{k+1}}\tilde{\boldsymbol{h}}_{l_{k+1}})\|$ 的取值最小，表明沿着与期望输出绝对相关度最大的方向搜索隐节点，误差减小的速度最快.　　证毕.

基于ELM的前向选择算法OFS-ELM归结如下.

算法 1.　OFS-ELM 算法.

输入：训练集 $\{(\boldsymbol{x}_j,\boldsymbol{t}_j)\}_{j=1}^N\subset R^n\times R$，激活函数 $g(x)$，候选隐节点个数 $\bar{L}$，期望学习精度 ϵ

输出：实际输出 $\hat{\boldsymbol{T}}$，活跃集 Γ

初始化：随机生成隐藏层节点学习参数 $(\boldsymbol{w}_i,b_i)$，$i=1,2,\cdots,\bar{L}$. 根据式(5)计算隐藏层输出矩阵 $\boldsymbol{H}_{\bar{L}}=[\boldsymbol{h}_1,\boldsymbol{h}_2,\cdots,\boldsymbol{h}_{\bar{L}}]$. 设置候选集 $\Lambda=\{1,2,\cdots,\bar{L}\}$，初始化实际输出 $\hat{\boldsymbol{T}}^{(0)}=\boldsymbol{0}$，活跃集 $\Gamma^{(0)}=\varnothing$. 将期望输出 $\boldsymbol{T}$ 和特征映射向量 $\boldsymbol{h}_l$ 中心化，$l=1,2,\cdots,\bar{L}$.

1. FOR $k=0$ TO $k=\bar{L}-1$
2. 计算 $\begin{cases}\boldsymbol{h}_l'=\boldsymbol{h}_l-\boldsymbol{H}_k(\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{H}_k)^{-1}\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{h}_l\\\tilde{\boldsymbol{h}}=\boldsymbol{h}_l'/\|\boldsymbol{h}_l'\|_2\\l\in\Lambda\backslash\Gamma^{(k)}\end{cases}$
3. 计算 $l_{k+1}=\arg\max\limits_{l\in\Lambda\backslash\Gamma^{(k)}}|\tilde{\boldsymbol{h}}_l^{\mathrm{T}}\boldsymbol{T}|$
4. 更新活跃集 $\Gamma^{(k+1)}=\Gamma^{(k)}\cup\{l_{k+1}\}$
5. 更新实际输出 $\hat{\boldsymbol{T}}^{(k+1)}=\hat{\boldsymbol{T}}^{(k)}+\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{l_{k+1}}\tilde{\boldsymbol{h}}_{l_{k+1}}$
6. IF $\|\hat{\boldsymbol{T}}^{(k+1)}-\hat{\boldsymbol{T}}^{(k)}\|\leq\epsilon$
7. BREAK
8. END IF
9. END FOR

3.2　基于 ELM 的正交后向移除

与 OFS 算法相反，正交后向移除(Orthogonal Backward Elimination，OBE)算法首先建立一个包含所有候选隐节点的全模型，然后逐个剔除与估计残差相关性最小的隐节点，直到模型中没有可剔除的变量为止[24].

算法 2 给出了基于 ELM 的正交后向移除 OBE-ELM 的详细训练过程，其中 $|\Gamma^{(k+1)}|$ 表示第 $k+1$ 步时活跃集中的元素个数，$\boldsymbol{H}_{k-\boldsymbol{h}_d}=(\boldsymbol{H}_{\bar{L}})_{\Gamma^{(k)}\backslash\{d\}}$ 表示从矩阵 $\boldsymbol{H}_k=\boldsymbol{H}_{\Gamma^{(k)}}$ 中删除列向量 $\boldsymbol{h}_d$. 当 $k=\bar{L}-1$ 时，规定 $\boldsymbol{H}_{k-\boldsymbol{h}_d}(\boldsymbol{H}_{k-\boldsymbol{h}_d}^{\mathrm{T}}\boldsymbol{H}_{k-\boldsymbol{h}_d})^{-1}\boldsymbol{H}_{k-\boldsymbol{h}_d}^{\mathrm{T}}\boldsymbol{h}_d=\boldsymbol{0}$.

算法 2.　OBE-ELM 算法.

输入：训练集 $\{(\boldsymbol{x}_j,\boldsymbol{t}_j)\}_{j=1}^N\subset R^n\times R$，激活函数 $g(x)$，候选隐节点个数 $\bar{L}$，最小隐节点数 $\tilde{L}$

输出：实际输出 $\hat{\boldsymbol{T}}$，活跃集 Γ

初始化：随机生成隐藏层节点学习参数 $(\boldsymbol{w}_i,b_i)$，$i=1,2,\cdots,\bar{L}$. 根据式(5)计算隐藏层输出矩阵 $\boldsymbol{H}_{\bar{L}}=[\boldsymbol{h}_1,\boldsymbol{h}_2,\cdots,\boldsymbol{h}_{\bar{L}}]$. 初始化实际输出 $\hat{\boldsymbol{T}}^{(0)}=\boldsymbol{H}_{\bar{L}}(\boldsymbol{H}_{\bar{L}}^{\mathrm{T}}\boldsymbol{H}_{\bar{L}})^{-1}\boldsymbol{H}_{\bar{L}}^{\mathrm{T}}\boldsymbol{T}$，活跃集 $\Gamma^{(0)}=\{1,2,\cdots,\bar{L}\}$. 将期望输出 $\boldsymbol{T}$ 和特征映射向量 $\boldsymbol{h}_d$ 中心化，$d=1,2,\cdots,\bar{L}$.

1. FOR $k=0$ TO $k=\bar{L}-1$
2. $\begin{cases}\boldsymbol{h}_d'=\boldsymbol{h}_d-\boldsymbol{H}_{k-\boldsymbol{h}_d}(\boldsymbol{H}_{k-\boldsymbol{h}_d}^{\mathrm{T}}\boldsymbol{H}_{k-\boldsymbol{h}_d})^{-1}\boldsymbol{H}_{k-\boldsymbol{h}_d}^{\mathrm{T}}\boldsymbol{h}_d\\\tilde{\boldsymbol{h}}_d=\boldsymbol{h}_d'/\|\boldsymbol{h}_d'\|_2\\d\in\Gamma^{(k)}\end{cases}$
3. 计算 $d_{k+1}=\arg\min\limits_{d\in\Gamma^{(k)}}|\tilde{\boldsymbol{h}}_d^{\mathrm{T}}\boldsymbol{T}|$
4. 更新活跃集 $\Gamma^{(k+1)}=\Gamma^{(k)}\backslash\{d_{k+1}\}$
5. 更新实际输出 $\hat{\boldsymbol{T}}^{(k+1)}=\hat{\boldsymbol{T}}^{(k)}-\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{d_{k+1}}\tilde{\boldsymbol{h}}_{d_{k+1}}$
6. IF $|\Gamma^{(k+1)}|<\tilde{L}$
7. BREAK
8. END IF
9. END FOR

3.3　OFS 和 OBE 各自的不足

OFS 算法的主要缺点是隐节点一旦引入到模型中，就不能被剔除，没有考虑新加入的隐节点对已选入的隐节点的影响. 如图 1 所示，$\boldsymbol{T}$ 可以由 $\boldsymbol{h}_1$ 和 $\boldsymbol{h}_2$ 的线性组合来表示，但 $\boldsymbol{h}_3$ 与 $\boldsymbol{T}$ 的夹角最小，因此 OFS 算法会首先选择 $\boldsymbol{h}_3$，之后即使发现 $\boldsymbol{h}_1$ 和 $\boldsymbol{h}_2$ 的组合最优，也无法剔除 $\boldsymbol{h}_3$. 另一方面，OBE 算法考虑了隐节点间的组合作用，可以纠正这样的错误. 因为 $\boldsymbol{h}_3$ 关于 $\mathrm{Span}(\boldsymbol{h}_1,\boldsymbol{h}_2)$ 正交化后与 $\boldsymbol{T}$ 正交，所以 $\boldsymbol{h}_3$ 在

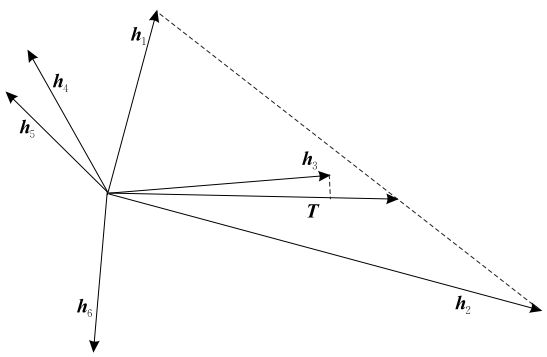


图 1　隐节点选择过程示意图

第一步就会被 OBE 算法删除. 然而,OBE 算法的根本问题是,在 $\bar{L}\gg N$ 的情况下,该方法不能阻止模型过拟合[25]. 相比之下,OFS 算法在这种情况下不受任何影响,即不要求 $\bar{L}<N$. 因此,恰当地将两者相结合,就可以吸收各自的优点,克服彼此的不足.

4 AOS-ELM 算法

4.1 算法描述

本文充分考虑了隐节点之间的内在联系和相互影响,提出了一种隐节点自适应正交搜索算法(AOS-ELM),在保持模型泛化能力的同时进一步降低 ELM 网络结构的复杂度. AOS-ELM 算法的核心步骤主要分成前后交替学习和后向审查. 在前后交替学习阶段,每次使用前向选择从候选集中引入一个新的隐节点后,就要执行后向移除对已选入的隐节点进行逐个检查,将变得不重要的隐节点从模型中剔除. 为了防止重要的隐节点组合被误删,该阶段的删除力度不能太强,这样可能会残留一些冗余的隐节点. 于是需要加大力度的后向移除进行最后审查,使冗余的隐节点尽可能地从模型中移除.

AOS-ELM 算法的流程归结如下.

算法 3. AOS-ELM 算法.

输入:训练集 $\{(\boldsymbol{x}_j,\boldsymbol{t}_j)\}_{j=1}^{N}\subset R^n\times R$,激活函数 $g(x)$,候选隐节点个数 $\bar{L}$,最小隐节点数 $\tilde{L}$

输出:实际输出 $\hat{\boldsymbol{T}}$,活跃集 Γ

初始化:随机生成隐藏层节点学习参数 $(\boldsymbol{w}_i,b_i)$,$i=1,2,\cdots,\bar{L}$. 根据式(5)计算隐藏层输出矩阵 $\boldsymbol{H}_{\bar{L}}=[\boldsymbol{h}_1,\boldsymbol{h}_2,\cdots,\boldsymbol{h}_{\bar{L}}]$. 设置候选集 $\Lambda=\{1,2,\cdots,\bar{L}\}$,初始化实际输出 $\hat{\boldsymbol{T}}^{(0)}=\boldsymbol{0}$,活跃集 $\Gamma^{(0)}=\varnothing$,$0<\tau_1,\tau_2<1$,$k=s=0$,$\eta=\eta^+=\eta^-=0$. 将期望输出 $\boldsymbol{T}$ 和特征映射向量 $\boldsymbol{h}_l$ 中心化,$l=1,2,\cdots,\bar{L}$.

/* 前后交替学习 */

1. 计算
$$\begin{cases} l_{k+1}^{+}=\arg\max\limits_{l\in\Lambda\backslash\Gamma^{(k)}}|\boldsymbol{h}_l^{\mathrm{T}}\boldsymbol{T}| \\ \Gamma^{(k+1)}=\Gamma^{(k)}\cup\{l_{k+1}^{+}\} \\ \tilde{\boldsymbol{h}}_{l_{k+1}^{+}}=\boldsymbol{h}_{l_{k+1}^{+}}/\|\boldsymbol{h}_{l_{k+1}^{+}}\|_2 \\ \hat{\boldsymbol{T}}^{(k+1)}=\hat{\boldsymbol{T}}^{(k)}+\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{l_{k+1}^{+}}\tilde{\boldsymbol{h}}_{l_{k+1}^{+}} \\ k=k+1 \end{cases}$$
2. WHILE $\Lambda\backslash\Gamma^{(k)}\neq\varnothing\,\&\&\,|\Gamma^{(k)}|<N$

/* 前向选择 */

3. 与算法 1 的第 2 行计算相同
4. 计算 $l_{k+1}^{+}=\arg\max\limits_{l\in\Lambda\backslash\Gamma^{(k)}}|\tilde{\boldsymbol{h}}_l^{\mathrm{T}}\boldsymbol{T}|$
5. 更新活跃集 $\Gamma^{(k+1)}=\Gamma^{(k)}\cup\{l_{k+1}^{+}\}$
6. 更新 $\hat{\boldsymbol{T}}^{(k+1)}=\hat{\boldsymbol{T}}^{(k)}+\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{l_{k+1}^{+}}\tilde{\boldsymbol{h}}_{l_{k+1}^{+}}$
7. 计算 $\xi_{k+1}=(\|\boldsymbol{T}-\hat{\boldsymbol{T}}^{(k)}\|_2^2-\|\boldsymbol{T}-\hat{\boldsymbol{T}}^{(k+1)}\|_2^2)/N$
8. 更新 $\eta=\eta+\xi_{k+1}$,$\eta^+=\eta^++\xi_{k+1}$,$k=k+1$
9. 设置 $counter=0$

/* 后向移除 */

10. WHILE TRUE
11. 与算法 2 的第 2 行计算相同
12. 计算 $d_k=\arg\min\limits_{d\in\Gamma^{(k)}}|\tilde{\boldsymbol{h}}_d^{\mathrm{T}}\boldsymbol{T}|$
13. 更新 $\hat{\boldsymbol{T}}=\hat{\boldsymbol{T}}^{(k)}+\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{d_k}\tilde{\boldsymbol{h}}_{d_k}$
14. 计算 $\xi^-=(\|\boldsymbol{T}-\bar{\boldsymbol{T}}\|_2^2-\|\boldsymbol{T}-\hat{\boldsymbol{T}}^{(k)}\|_2^2)/N$
15. 更新 $\eta^-=\eta^-+\xi^-$
16. IF $\eta^->\tau_1\eta^+$
17. IF $counter\neq0$
18. 更新 $\eta^-=\eta^--\xi^-$
19. END IF
20. BREAK
21. END IF
22. 更新 $\eta=\eta-\xi^-$,$s=s+1$,$l_s^-=d_k$
23. 更新 $\Gamma^{(k+1)}=\Gamma^{(k)}\backslash\{l_s^-\}$,$\hat{\boldsymbol{T}}^{(k+1)}=\bar{\boldsymbol{T}}$
24. 更新 $k=k+1$,$counter=counter+1$
25. END WHILE
26. IF $counter\neq0\,\&\&\,l_s^-=l_{s-1}^-$
27. $\Lambda=\Lambda\backslash\{l_s^-\}$
28. EDN IF
29. END WHILE

/* 后向审查 */

30. WHILE $|\Gamma^{(k)}|>\tilde{L}$
31. 与算法 2 的第 2 行计算相同
32. 计算 $d_k=\arg\min\limits_{d\in\Gamma^{(k)}}|\tilde{\boldsymbol{h}}_d^{\mathrm{T}}\boldsymbol{T}|$
33. 更新 $\bar{\boldsymbol{T}}=\hat{\boldsymbol{T}}^{(k)}-\boldsymbol{T}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{d_k}\tilde{\boldsymbol{h}}_{d_k}$
34. 计算 $\xi^-=(\|\boldsymbol{T}-\bar{\boldsymbol{T}}\|_2^2-\|\boldsymbol{T}-\hat{\boldsymbol{T}}^{(k)}\|_2^2)/N$
35. IF $\xi^->\tau_2\eta$
36. BREAK
37. END IF
38. 更新 $\Gamma^{(k+1)}=\Gamma^{(k)}\backslash\{d_k\}$,$\hat{\boldsymbol{T}}^{(k+1)}=\bar{\boldsymbol{T}}$
39. 更新 $\eta=\eta-\xi^-$,$k=k+1$
40. END WHILE

4.2 算法分析

AOS-ELM 在第一阶段交替执行正交化的前向选择和后向移除,每次使用 OFS 增加一个新的隐节点,就要使用 OBE 对前面引入的隐节点逐个检查,并通过 $\eta^->\tau_1\eta^+$ 来判断是否需要剔除. 这里 η^- 表示算法执行到当前步,每删除一个隐节点后均方误差的累计增加量. 相应地,η^+ 表示引入一系列隐节点后均方误差的累计减少量. 如果删除某个隐节点所导致的均方误差累计增加量大于之前均方误差

累计减少量的 τ_1 倍，则认为代价过高，不予删除. 此外，为了防止 η^- 不必要的增长，在算法第 18 行对其进行了适当的修正. 还是以图 1 为例，假设候选隐节点为 $\{\boldsymbol{h}_1,\boldsymbol{h}_2,\boldsymbol{h}_3,\boldsymbol{h}_4,\boldsymbol{h}_5,\boldsymbol{h}_6\}$，$\boldsymbol{T}$ 可以表示为 $\boldsymbol{h}_1$ 和 $\boldsymbol{h}_2$ 的线性组合，隐节点的选取次序是 $\{+3,+2,+5,+4,+1,-4,-5,-3\}$. 在此过程中，$\eta^+$ 累积了前向选择步 $\{+2,+5,+4,+1\}$ 所引起的均方误差减少量，η^- 累积了后向移除步 $\{-2,-5,-4,-4,-5,-3\}$ 所产生的均方误差增加量. 最后三步 $\{-4,-5,-3\}$ 引起的均方误差增加量要小于 $\{+3,+2,+5,+4,+1\}$ 引起的均方误差减少量，才能成功删除 $\{\boldsymbol{h}_4,\boldsymbol{h}_5,\boldsymbol{h}_3\}$. 在删除 $\boldsymbol{h}_3$ 后，后向移除还会对 $\boldsymbol{h}_1$ 进行检查，而删除 $\boldsymbol{h}_1$ 会使均方误差急剧增大，导致删除失败，所以不能将删除 $\boldsymbol{h}_1$ 造成的均方误差增加量计入 η^-. 在 $\{+3,+2,+5,+4,+1,-4,-5,-3\}$ 之后，前向选择还会引入新的隐节点 $\boldsymbol{h}_6$，由于最优隐节点 $\boldsymbol{h}_1$ 和 $\boldsymbol{h}_2$ 都已被选入，$\boldsymbol{h}_6$ 随后可能会被剔除、再引入、再剔除. 为了避免死循环，将此类隐节点从候选集 Λ 中永久删除.

在前后交替学习阶段，隐节点逐个引入，某一步时某个隐节点单独看不重要，但之后该隐节点由于新隐节点的引入可能会变得非常重要，因此不能过早剔除. 也就是说，为了保证重要的隐节点组合都被选入，该阶段的剔除力度不能太大，从而可能会有冗余的隐节点残留下来. 于是，我们设计了后向移除策略对前后选择阶段引入的隐节点做最后的检查，其中隐节点的剔除与否由条件 $\xi^- > \tau_2\eta$ 来决定. 这里，ξ^- 表示删除一个隐节点后引起的均方误差增加量，η 表示算法执行到当前步，均方误差累计净减少量. 如果删除某个隐节点引起的均方误差增加量大于之前均方误差累计净减少量的 τ_2 倍，则认为代价过高，不予删除. 显然，该删除条件比 $\eta^- > \tau_1\eta^+$ 更难满足，因此删除力度更大.

最后，AOS-ELM 在 $\bar{L} > N$ 的情况下依然可以工作. 根据算法第 2 行的循环条件 $|\Gamma^{(k)}| < N$ 可知，前后交替学习阶段最多引入 $N-1$ 个隐节点. 因为，通常隐节点数大于训练样本数会发生过拟合，此时重要隐节点输出向量对应的系数会变得非常小，很容易被后向审查阶段删除.

5　实验结果和分析

5.1　实验数据集

本文首先使用 12 个基准回归数据集[26]对 AOS-ELM 的性能进行评估，并与标准 ELM[2]、I-ELM[3]、CI-ELM[10]、EM-ELM[11]、DAOI-ELM[14]和 OP-ELM[16]进行了比较. 对每个数据集进行 50 次实验，每次实验前训练集和测试集的划分都是独立不重叠的，根据表 1 中指定的大小从原始数据集中随机抽取. 实验数据的输入和输出分别被归一化到 $[-1,1]$ 和 $[0,1]$.

表 1　基准回归数据集描述信息

数据集	#训练样本	#测试样本	#特征
Abalone (AB)	2089	2088	8
Delta ailerons (DA)	3565	3564	5
Delta elevators (DE)	4759	4758	6
Laser (LA)	497	496	4
Electrical-maintenance (EM)	528	528	4
Daily electricity energy (DEE)	183	182	6
Forest fires (FF)	259	258	12
Treasury (TR)	525	524	15
Mortgage (MO)	525	524	15
Baseball salaries (BS)	169	168	16
Auto MPG (AM)	196	196	7
Weather Ankara (WA)	805	804	9

为了进一步验证本文算法的有效性，我们还将其应用到图像颜色恒常性计算问题中. 物体在不同的光源下会呈现出不同的颜色，颜色恒常性计算的目标就是要消除场景中光照差异对图像成像带来的影响，准确地反映出物体表面真实的颜色. 颜色恒常性计算被认为是目标识别、跟踪等许多计算机视觉任务的重要组成部分. 本文主要关注于单一光源下图像的光照估计问题，并假设整个场景中的光照是均匀分布的. 此外，为了得到颜色的色度值，需要将 RGB 颜色空间归一化到 rg 色度空间：$r=R/(R+G+B)$，$g=G/(R+G+B)$，$b=1-r-g$. 我们选用两个广泛使用的标准颜色恒常性数据集进行实验，数据集描述如下：

(1) Gehler-Shi 数据集. 该数据集最初由 Gehler 等人[27]提供，它包含 568 张由 Canon 1D 和 Canon 5D 拍摄的来自室内和室外场景的图像，并以 RAW 格式保存. Shi 等人[28]对原始数据集进行了重新处理，以获取高动态范围线性图像. 此外，由设备引起的黑电平偏移必须从线性图像中减去. 对于 Canon 1D，黑电平为 0；对于 Canon 5D，黑电平为 129. 每幅图像的已知位置都放置了一张 Macbeth Color Checker 色卡以获取场景的真实光照信息. 文献[29]对该数据集的真实场景光照颜色进行了重新计算. 为了防止色卡对实验结果带来偏差，将其所在的区域设置为 RGB=(0,0,0) 进行屏蔽.

(2) Gray Ball 数据集. 该数据集最初由 11 346

幅从近 2 个小时的 15 个不同视频片段中提取的 240×360 非线性图像组成,包括各种光照条件下的室内、室外场景[30]. 场景中的光源颜色利用图像右下角的灰色小球测量得到. 为了消除灰色小球对光照估计产生的影响,在实验过程中需要将其从图像中移除,裁剪后图像的大小调整为 240×240. 由于同一视频片段的图像之间高度相关,Bianco 等人[31]从原始数据集中提取了 1135 幅相关性较低的图像组成代表性子集. 此后,Gijsenij 等人[32]又利用伽马校正(gamma=2.2)将这些图像从 NTSC-RGB 颜色空间转化到线性 RGB 颜色空间,同时重新计算了线性图像的真实光照颜色.

为了使实验结果比较公平客观且具有实际意义,本文遵循以往研究工作针对上述图像数据集设定的测试标准[32]. 对于 Gehler-Shi 和 Gray Ball 数据集,分别使用 3-折交叉验证和 15-折交叉验证来评估不同颜色恒常算法的性能.

5.2 评价指标

在基准回归数据集上,本文使用测试集的均方根误差(Root Mean Square Error,RMSE)和隐节点数来评估模型的泛化性能和复杂度. 实验结果包括测试集的均方根误差的平均值和相应的标准差,网络中隐节点数的平均值和相应的标准差.

在颜色恒常性计算研究中最常用的误差度量方法是角度误差[33]. 角度误差测量的是图像的真实光照色度 $\boldsymbol{e}_u=(r_u,g_u,b_u)$ 与算法得到的估计光照色度 $\boldsymbol{e}_w=(r_w,g_w,b_w)$ 之间的夹角,其计算公式如下:

$$\Theta_A(\boldsymbol{e}_u,\boldsymbol{e}_w)=\cos^{-1}\left(\frac{\boldsymbol{e}_u\cdot\boldsymbol{e}_w}{\|\boldsymbol{e}_u\|\times\|\boldsymbol{e}_w\|}\right)\times\frac{180^\circ}{\pi} \quad (16)$$

本文在整个测试图像集上通过计算角度误差的均值(mean)、中值(median)、三均值(trimean)、最低 25%误差均值(best 25%)和最高 25%误差均值(worst 25%)来全面评价各个颜色恒常性算法的性能[34]. 其中,中值和三均值角度误差通常是大多数文献的首选评价指标,其值越低说明真实光照和估计光照之间的差异越小,算法的准确性也就越高.

此外,为了比较不同颜色恒常性算法之间的误差是否具有显著性差异,本文对测试图像集上的角度误差分布进行 Wilcoxon 符号秩检验[32]. 在实验中,显著性水平参数的值设为 0.05.

5.3 基准回归数据集测试

5.3.1 对比算法和实验设置

为了使算法的比较评价结果更为客观、准确,本文使用网格搜索和交叉验证来确定最优的模型参数. ELM①、I-ELM②、CI-ELM②、EM-ELM②、DAOI-ELM 和 AOS-ELM 网络的隐节点选用 Sigmoid 函数:$G(\boldsymbol{w},b,\boldsymbol{x})=1/(1+\exp(-(\boldsymbol{w}\cdot\boldsymbol{x}+b)))$ 和 RBF 函数:$G(\boldsymbol{w},b,\boldsymbol{x})=\exp(-b\|\boldsymbol{x}-\boldsymbol{w}\|^2)$,并采用 5-折交叉验证对模型进行检验. 隐藏层节点参数 $\boldsymbol{w}$ 和 b 分别从区间[-1,1]和[0,1]内随机选取. OP-ELM③利用多响应稀疏回归和留一交叉验证确定网络结构,为了得到更好的泛化性能,按照文献[13]的建议,使用 Linear、Sigmoid 和 Gaussian 三种核函数的组合作为激活函数.

ELM 隐藏层节点个数的取值范围设为{5,10,…,495,500}. 对于 I-ELM、CI-ELM 和 EM-ELM,最大隐节点个数设为 200,期望学习精度的取值范围设为{0.01,0.02,…,0.19,0.2}. 在 FF 数据集上,为了防止期望学习精度大于初始残差,I-ELM 和 CI-ELM 的期望学习精度的取值范围应该设为{0.001,0.002,…,0.009,0.01}. 由于 EM-ELM 允许逐个或逐块增加隐节点,该算法中初始网络的隐节点个数设为 5,然后在训练过程中逐个添加新的隐节点. 与文献[14]相同,DAOI-ELM 的最大隐节点个数设为 300,期望学习精度在 TR 和 AM 数据集上的取值范围设为{0.01,0.02,…,0.19,0.2},其余 10 个回归问题中期望学习精度的取值范围设为{0.001,0.002,…,0.019,0.02}. 对应 TR 和 MO 这两个数据集的 OP-ELM 网络的初始隐节点个数设为 150,其余 10 个数据集上,初始网络的隐节点数设为 100. 在 AOS-ELM 中,参数 τ_1 除非特别说明,默认取值 0.5. 在 BS、AM、FF、DEE 这 4 个数据集上,τ_2 的取值范围设为{0.01,0.02,…,0.39,0.4},其余 8 个数据集上,τ_2 的取值范围设为{0.001,0.002,…,0.019,0.02}. 对于 FF、DEE 这两个数据集,最小隐节点数 $\tilde{L}$ 设为 2,其余 10 个数据集上,$\tilde{L}$ 设为 5. 对于 DEE、FF、AM 这 3 个数据集,候选隐节点数 $\bar{L}$ 设为 50. 对应 TR 和 MO 数据集的 $\bar{L}$ 分别设为 100 和 120. 在其余 7 个数据集上,$\bar{L}$ 的取值为 80.

5.3.2 算法性能比较和分析

表 2 和表 3 分别比较了不同网络结构学习算法在 12 个基准回归数据集上的测试精度和需要的隐节点个数. 下面根据不同类型的激活函数对实验结果进行讨论和分析.

① https://www.ntu.edu.sg/home/egbhuang/elm_codes.html
② https://github.com/labcisne/ELMToolbox/tree/master/Incremental%20ELMs
③ https://research.cs.aalto.fi/aml/software.shtml

表 2　采用不同类型隐节点的算法的测试误差的均值和标准差比较

数据集	激活函数	ELM	I-ELM	CI-ELM	EM-ELM	DAOI-ELM	AOS-ELM	OP-ELM (L+S+G)
		Testing RMSE	Testing RMSE	Testing RMSE	Testing RMSE	Testing RMSE	Testing RMSE	Testing RMSE
		Mean±Dev	Mean±Dev	Mean±Dev	Mean±Dev	Mean±Dev	Mean±Dev	Mean±Dev
AB	Sigmoid	0.0903±0.0179	0.1006±0.0086	0.0922±0.0070	0.0880±0.0108	0.1580±0.0243	0.0882±0.0153	0.0943±0.0366
	RBF	0.0903±0.0073	0.0991±0.0093	0.0878±0.0055	0.0855±0.0061	0.1518±0.0220	**0.0831±0.0058**	
DA	Sigmoid	0.0445±0.0050	0.0606±0.0080	0.0459±0.0049	0.0456±0.0061	0.2389±0.0358	**0.0435±0.0052**	0.0447±0.0048
	RBF	0.0467±0.0051	0.0703±0.0138	0.0477±0.0051	0.0463±0.0061	0.2108±0.0473	0.0449±0.0055	
DE	Sigmoid	0.0581±0.0055	0.0732±0.0089	0.0596±0.0045	0.0605±0.0061	0.2506±0.0279	0.0576±0.0043	0.0585±0.0055
	RBF	0.0606±0.0049	0.0849±0.0142	0.0605±0.0049	0.0592±0.0050	0.2218±0.0320	0.0576±0.0035	
LA	Sigmoid	0.0405±0.0146	0.1020±0.0078	0.0977±0.0095	0.0404±0.0099	0.1989±0.0390	0.0369±0.0148	0.0323±0.0122
	RBF	0.0736±0.0124	0.0839±0.0108	0.0782±0.0105	0.0464±0.0156	0.1811±0.0415	0.0329±0.0086	
EM	Sigmoid	0.0142±0.0013	0.0454±0.0068	0.0393±0.0038	0.0192±0.0012	0.1852±0.0559	0.0142±0.0021	0.0143±0.0040
	RBF	0.0494±0.0204	0.0483±0.0085	0.0314±0.0041	0.0204±0.0037	0.1534±0.0366	0.0143±0.0028	
DEE	Sigmoid	0.1002±0.0091	0.1039±0.0076	0.1008±0.0062	0.1011±0.0065	0.2922±0.0286	0.1003±0.0079	0.1010±0.0082
	RBF	0.1224±0.0122	0.1323±0.0154	0.1054±0.0064	0.1131±0.0120	0.2775±0.0311	0.1057±0.0088	
FF	Sigmoid	0.1428±0.1448	0.1204±0.1598	0.1240±0.1502	0.1523±0.1612	0.1473±0.1188	**0.1025±0.1354**	0.1270±0.1248
	RBF	0.1095±0.1087	0.1321±0.1346	0.1301±0.1285	0.1253±0.1327	0.1379±0.1063	0.1092±0.1154	
TR	Sigmoid	0.0130±0.0013	0.0424±0.0077	0.0316±0.0042	0.0129±0.0013	0.1685±0.0417	0.0123±0.0010	0.0130±0.0012
	RBF	0.0230±0.0026	0.0517±0.0077	0.0353±0.0055	0.0146±0.0022	0.1756±0.0334	0.0144±0.0019	
MO	Sigmoid	0.0066±8.2e-04	0.0362±0.0072	0.0247±0.0038	0.0105±9.4e-04	0.2078±0.0489	0.0061±7.2e-04	0.0062±7.3e-04
	RBF	0.0211±0.0022	0.0588±0.0117	0.0328±0.0046	0.0128±0.0017	0.2216±0.0394	0.0090±0.0011	
BS	Sigmoid	0.1384±0.0181	0.1387±0.0229	**0.1335±0.0204**	0.1405±0.0230	0.2218±0.0518	0.1426±0.0197	0.1409±0.0214
	RBF	0.1638±0.0245	0.1775±0.0263	0.1518±0.0213	0.1722±0.0313	0.2008±0.0398	0.1666±0.0256	
AM	Sigmoid	0.0838±0.0078	0.1045±0.0097	0.0981±0.0083	0.0872±0.0099	0.2276±0.0354	0.0831±0.0066	0.0834±0.0078
	RBF	0.1140±0.0117	0.1196±0.0154	0.0962±0.0085	0.0951±0.0103	0.2406±0.0317	0.0897±0.0091	
WA	Sigmoid	0.0211±0.0051	0.0516±0.0137	0.0295±0.0032	0.0216±0.0023	0.3205±0.0390	**0.0203±0.0067**	0.0268±0.0241
	RBF	0.0359±0.0048	0.0977±0.0227	0.0420±0.0068	0.0275±0.0058	0.3284±0.0609	0.0253±0.0038	

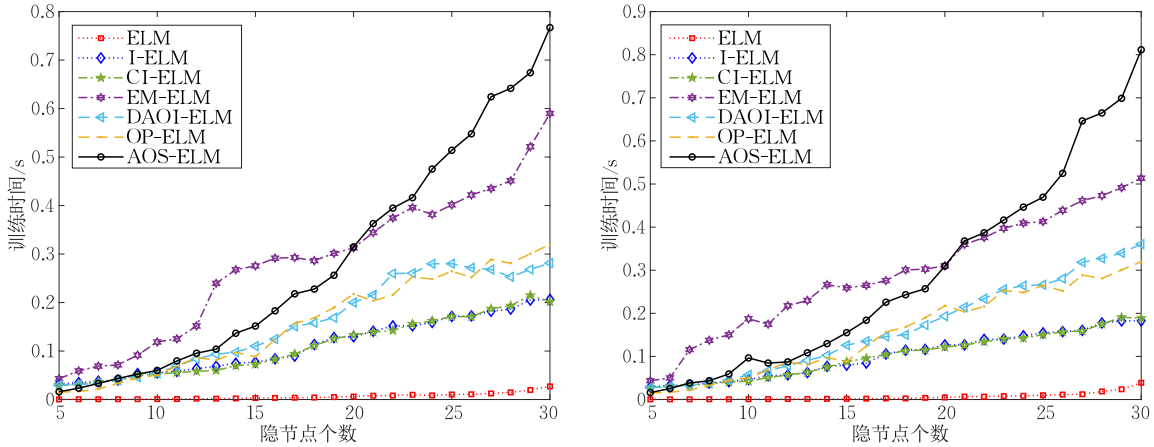
表 3　采用不同类型隐节点的算法的隐节点个数的均值和标准差比较

数据集	激活函数	ELM	I-ELM	CI-ELM	EM-ELM	DAOI-ELM	AOS-ELM	OP-ELM (L+S+G)
		#隐节点	#隐节点	#隐节点	#隐节点	#隐节点	#隐节点	#隐节点
		Mean±Dev	Mean±Dev	Mean±Dev	Mean±Dev	Mean±Dev	Mean±Dev	Mean±Dev
AB	Sigmoid	44.4±13.50	200±0	200±0	32.9±27.68	**13.32±5.03**	25.84±4.07	39.6±15.20
	RBF	146.4±35.89	200±0	198.54±7.65	41.7±22.87	49.52±20.46	25.7±4.63	
DA	Sigmoid	49.30±14.74	200±0	200±0	15.78±14.69	**7.74±2.69**	35.86±7.40	53.6±14.74
	RBF	169.3±36.60	199.64±6.48	200±0	82.78±75.93	23.82±16.73	35.78±4.65	
DE	Sigmoid	60.9±20.04	197.68±16.40	200±0	153.12±76.54	**10.08±5.78**	35.8±7.29	52.6±21.68
	RBF	203.3±47.81	198.42±11.17	200±0	174.36±54.55	49.02±49.64	28±5.42	
LA	Sigmoid	45.4±12.32	200±0	200±0	**25.24±5.00**	26.68±2.13	59.28±9.53	48.2±11.80
	RBF	149.8±41.17	200±0	197.78±10.99	43.56±15.91	71.76±4.04	40.04±6.85	
EM	Sigmoid	40.3±8.17	200±0	199.8±1.41	10.14±2.63	**8.44±2.32**	47.54±12.96	55.4±11.78
	RBF	299.4±141.05	194.7±22.53	196.98±14.68	30.92±20.47	24.80±6.72	42.18±6.45	
DEE	Sigmoid	17.9±7.63	197.96±14.43	192.38±28.29	12.94±8.15	13.08±7.32	12.64±3.06	19.36±11.65
	RBF	164.8±204.02	200±0	195.72±21.49	30.5±15.13	47.20±26.14	**11.8±3.33**	
FF	Sigmoid	5.9±2.19	200±0	200±0	5±0	24.56±19.37	4.94±1.94	**2.32±3.29**
	RBF	7.2±4.76	200±0	200±0	5±0	106.36±47.02	3.5±0.68	
TR	Sigmoid	83.4±26.58	192.28±31.44	199.08±4.62	63.34±16.38	**5.28±3.09**	37.68±5.27	74.4±26.49
	RBF	166.3±23.16	198.34±11.74	196.08±16.57	92.18±12.56	17±11.27	43.68±4.49	
MO	Sigmoid	101.4±21.76	197.92±12.05	200±0	23.74±3.43	**13.92±6.32**	43.02±5.89	82.8±25.97
	RBF	175.3±27.56	199.36±2.81	194±20.06	65.78±7.54	47.40±31.94	48.98±5.35	
BS	Sigmoid	25.4±7.41	196.74±16.25	197.9±14.85	20.46±10.39	39.36±24.24	12.36±3.60	22.78±12.08
	RBF	442.7±75.99	182.34±43.32	178.4±45.78	32.14±13.97	102.70±32.97	**9.58±2.26**	
AM	Sigmoid	25.3±6.73	200±0	197.4±12.91	29.38±12.96	**5.04±3.50**	13.28±3.41	26.8±10.35
	RBF	401.2±161.61	198.08±11.38	199.8±1.41	41.78±15.31	22.28±16.58	18.78±3.34	
WA	Sigmoid	68.2±17.08	196.12±16.08	200±0	23.6±4.53	**12.48±5.12**	27.76±4.29	63.5±20.44
	RBF	237.9±38.31	197.16±15.64	199.34±3.89	119.56±61.37	63.38±42.57	38.08±4.44	

当使用 Sigmoid 作为激活函数时，从表 2 可以看出，除 LA 和 BS 数据集外，AOS-ELM 在大部分数据集上取得了和其他 6 种对比算法非常接近或者更小的测试 RMSE，说明其具有良好的泛化性能. 标准 ELM 在大多数据集上的测试 RMSE 比 OP-ELM 小或者与其接近. 与 I-ELM、CI-ELM、EM-ELM、DAOI-ELM 相比，OP-ELM 通常能够达到更小的测试误差. 相应地，从表 3 可以看出，AOS-ELM 所需要的隐节点个数要远远小于 I-ELM 和 CI-ELM，因此具有更加紧凑的网络结构. 虽然在 DA、LA、EM、MO、WA 这 5 个数据集上 EM-ELM 的隐节点个数比 AOS-ELM 小，但是由于网络结构过度简化使得 EM-ELM 的测试精度下降较为剧烈. 与 OP-ELM 和标准 ELM 相比，AOS-ELM 在大部分数据集上能够得到一个结构更紧凑的网络，并且表现出相对稳定的性能. 在 DEE 和 BS 数据集上 AOS-ELM 能够产生优于 DAOI-ELM 的网络结构和预测性能，其余例子中 DAOI-ELM 所需的隐节点数虽然最少，但是所选择的模型复杂度过低导致其泛化能力不足，预测误差较大.

当使用 RBF 作为激活函数时，本文算法的测试 RMSE 小于或者接近标准 ELM、I-ELM、CI-ELM 和 EM-ELM，且在大多数据集上这种性能优势较为明显. 另一方面，从测试 RMSE 的标准差可以看出，本文提出的算法性能相对稳定. 更重要的是，对于大多数回归问题，AOS-ELM 需要的隐节点个数较少，而标准 ELM、I-ELM 和 CI-ELM 更容易产生较大的网络结构. 在 EM 数据集上，EM-ELM 的网络结构比本文算法的网络结构更加紧凑，但是其测试精度相对较差. 同样，对于 FF 数据集，在比较本文算法和 OP-ELM 的网络结构和泛化性能时可以得出相似的结果. 此外，对于 DEE、TR、MO、BS、AM 这 5 个数据集，OP-ELM 相较于其他对比算法甚至本文提出的算法均取得了最低测试 RMSE，而在其余 7 个数据集上，AOS-ELM 的测试 RMSE 小于 OP-ELM 或者相差无几. 从网络结构的复杂度来看，在 12 个回归问题中，除 DA、EM、TR 和 MO 外，AOS-ELM 所需要的隐节点个数均小于 DAOI-ELM. 此外，在大部分测试数据集上，DAOI-ELM 的隐节点数的标准差远远高于 AOS-ELM. 从预测精度来看，AOS-ELM 的测试误差在所有数据集上都要明显小于 DAOI-ELM. 最后，值得注意的是，本文算法能够获得较为稳定的网络结构，隐藏层节点个数的标准差较小，表明其对随机隐节点具有较强的鲁棒性.

为了评价算法在计算时间方面的性能，本文以 DEE 数据集为测试对象，分别比较了不同类型神经元下各种算法的训练时间随隐藏层中节点数目的变化，结果如图 2 所示. 其中，I-ELM、CI-ELM、EM-ELM、DAOI-ELM 的期望学习精度统一设为 0.01，训练过程中隐节点逐个增加，OP-ELM 采用组合激活函数. 本文使用了 Intel Core i7-6500U 2.50 GHz CPU 和 8 GB RAM 作为实验的硬件环境，所有程序都是在 MATLAB R2017b 上完成的. 可以看出，在隐节点个数相同的情况下，标准 ELM 的训练时间最短；I-ELM 和 CI-ELM 的运算速度相当；OP-ELM 的训练速度和 DAOI-ELM 相差不大；EM-ELM 和 AOS-ELM 的运算复杂度都比较高，而且随着隐节点数目的增加运算速度显著降低. 总体上，本文算法的计算复杂度要高于其他对比算法，这是由于每向前引入一个新的隐节点后，就要使用后向移除法对已选入的隐节点进行逐个检查. 这种



(a) 采用Sigmoid激活函数　　(b) 采用RBF激活函数

图 2　DEE 数据集上本文算法与其他网络学习算法在不同隐节点个数下的训练时间比较

贪婪的搜索策略虽然能够有效地减少冗余的隐节点，降低网络结构的复杂度并减少引发过拟合的机会，但同时也会导致模型的训练速度变慢. 在其他数据集上也能够得到相同的变化趋势，此处不再赘述. 最后，如何在后向移除过程中实现隐节点的逐块检验和删除，进一步提高本文算法的网络学习效率，将是我们下一阶段要重点研究的内容.

综上所述，由本小节的各种实验结果可以看出，相比较其他方法，AOS-ELM 在网络结构的复杂度和预测精度上均获得了较好的结果，在两者之间取得很好的平衡，因此具有较强的泛化能力，同时也能够产生非常稳定紧凑的网络结构.

5.4　颜色恒常性计算

5.4.1　对比算法和实验设置

颜色恒常性计算一般分为两个步骤：首先从输入图像中估计出场景的光照色度，然后通过 von Kries 对角变换[32]将不同光照条件下的图像映射到标准光照下的图像. 本文采用 D65 作为标准光源. 由于第二步计算过程比较简单，因此光照估计是颜色恒常性研究的主要目标. 现有的光照估计算法大体上可以分为无监督的方法、有监督的方法和融合性方法.

无监督的光照估计方法依赖于各种假设条件并利用图像自身的底层颜色特征来得到光源的颜色. 本文使用文献[35]提出的颜色恒常性计算框架①产生一系列常见的无监督光照估计算法作为对比实验，包括 White-Patch (WP)、Grey-World (GW)、Shades-of-Gray (SoG)、general Grey-World (gGW)、1st-order Grey-Edge (GE1)、2nd-order Grey-Edge (GE2). 此外，本文还选取了最近几年提出的基于视觉生理机制的自适应环绕调制 ASM 算法[36]以及基于灰色像素和均值漂移聚类的 MSGP 算法[37]加入对比实验. 对于 Gehler-Shi 图像集，根据原文的建议，MSGP② 的聚类带宽设为 0.001，从灰度值中选取的像素百分比设为 0.1%. 在 Gray Ball 图像集上，这两个参数的取值分别设为 0.001 和 10%. ASM③ 具有动态自适应性，因此没有参数需要调整.

有监督的光照估计方法通常需要一组具有真实光照信息标记的图像，利用统计学习或者机器学习技术构建颜色恒常性计算模型对图像进行光照校正. 实验中涉及到的此类算法分别是④：基于神经网络的 NN 算法[38]、基于支持向量回归的 SVR(2D) 和 SVR(3D)算法[39]、基于贝叶斯理论的 BCC 算法[27]、基于空间-频谱统计的 SSS 算法[40]、基于像素值的 GMP 色域映射算法[41]、基于图像导数的 GMD_V 和 GMD_{VV} 色域映射算法[42]、基于样例的 Exemplar 算法[43]、基于树结构分组联合稀疏表示的多线索 MC 算法[44]. NN、BCC、SSS、GMP、GMD_V、GMD_{VV}、Exemplar、MC 采用和文献[43-45]相同的实验设置. SVR(2D)使用二值化的色度直方图特征作为输入向量，其中 rg 色度空间被均匀划分成 50×50 个等面积的方格. SVR(3D)的输入向量在二值化的色度向量基础上加入了颜色的强度信息 $(R+G+B)$并将其等距划分成 15 个小格. SVR(2D)和 SVR(3D)算法采用 Gaussian 核函数，基于 MATLAB 软件嵌套 LIBSVM 工具箱实现[46]. 在基于支持向量回归的颜色恒常性算法中，SVR 的核参数 γ 和正则化系数 C 的取值范围分别设为 $\gamma \in \{0.01, 0.025, 0.05, 0.1, 0.2, 1, 2, 5, 10, 20, 50\}$，$C \in \{0.005, 0.01, 0.1, 1, 2, 5, 10\}$. 由于颜色恒常性计算关心的是对图像成像时的光照色度(r, g)进行估计，因此需要使用一维输出的 SVR 分别为 r 分量和 g 分量单独建模.

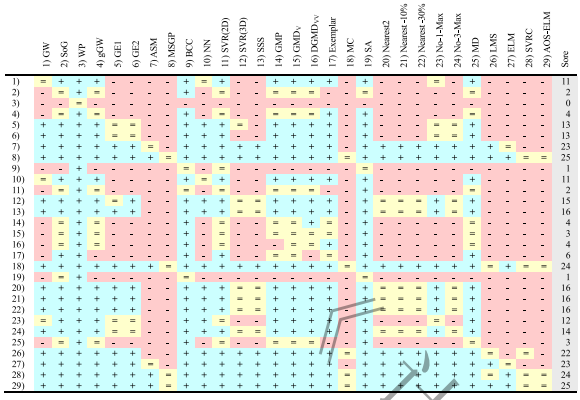
融合光照估计方法利用已有的颜色恒常性算法构建候选算法集合，针对特定的图像对所有候选算法估计得到的光照结果进行融合或者从候选集中选择一个最优的算法来预测场景的光照. 我们使用本文算法学习各个候选颜色恒常性算法估计出的光照色度值与图像的真实光照色度值之间的映射关系，并与几种有代表性的融合方案进行了比较，包括无监督的融合算法：简单平均(Simple Averaging, SA)[47]、最近邻平均 Nearest2[48]、Nearest-10%[48]、Nearest-30%[48]、No-1-Max[48]、No-3-Max[48]、中位数法(Median, MD)[48]；有监督的融合算法：基于最小二乘的 LMS 融合[47]、基于标准极限学习机的 ELM 融合[49]以及基于支持向量回归的 SVRC 融合[50]. 为了便于比较，本文使用 12 种典型的无监督和有监督单一颜色恒常性算法组成候选集合 $E=\{\text{WP}, \text{GW}, \text{SoG}, \text{gGW}, \text{GE1}, \text{GE2}, \text{NN}, \text{SVR(2D)}, \text{SVR(3D)}, \text{BCC}, \text{SSS}, \text{GMP}\}$. LMS、ELM、SVRC 和本文算法的输入向量 $\boldsymbol{V}=[\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_{|E|}]^{\mathrm{T}}$，其中 $|E|$ 表示候选算法的个数，$\boldsymbol{c}_i=(r_i, g_i)$表示第 i 个候选算法的光照色度估计值. 这些算法对应的输出向量也是光照色度 r 和 g. LMS 没有超参数需要调

① http://www.cat.uab.cat/downloads/
② https://github.com/yanlinqian/Mean-shifted-Gray-Pixel
③ https://github.com/ArashAkbarinia/ColourConstancy
④ https://colorconstancy.com/source-code/index.html

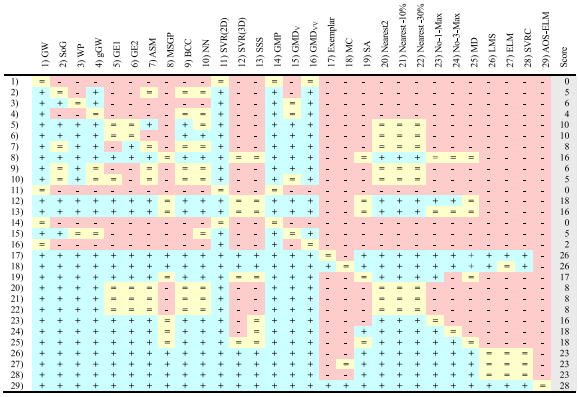
整. ELM 和本文算法的激活函数为 Sigmoid 函数. ELM 隐藏层节点个数的取值范围设为{10,20,…,200}. SVRC 采用 Gaussian 核函数,γ 和 C 的取值范围分别设为 $\gamma\in\{0.01,0.025,0.05,0.1,0.2,1,2,5,10,20,50\}$ 和 $C\in\{0.005,0.01,0.1,1,2,5,10\}$. 在本文提出的融合算法中,参数 τ_2 的取值范围设为{0.001,0.002,…,0.02},最小隐节点个数 $\hat{L}$ 设为 10,参数 τ_1 默认取 0.5,对应 Gray Ball、Gehler-Shi 数据集的候选隐节点个数 $\bar{L}$ 分别设为 80 和 60.

5.4.2 算法性能比较和分析

表 4 和表 5 分别给出了各种颜色恒常性计算方法在 Gehler-Shi 数据集和 Gray Ball 数据集上的实验结果,相应的 wilcoxon 符号秩检验的结果如图 3(a)和图 3(b)所示.



(a) Gehler-Shi数据集



(b) GrayBall数据集

图 3 各种颜色恒常性算法的 wilcoxon 符号秩检验结果((i,j)位置上的(+)号表示在 95%置信水平下,算法 i 明显优于算法 j,(−)号表示相反的意思,(=)号表示两种算法在统计意义上性能相当,最后一列的分数表示该行对应的算法优于其他所有对比算法的次数)

表 4 Gehler-Shi 数据集上各种颜色恒常性算法的性能比较

算法	Mean	Median	Trimean	Best-25%	Worst-25%
GW	4.77°	3.63°	3.94°	0.98°	10.49°
SoG	6.42°	4.48°	5.20°	0.79°	14.98°
WP	10.26°	9.15°	9.48°	1.56°	20.48°
gGW	6.35°	3.90°	4.76°	0.78°	15.79°
GE1	4.19°	3.28°	3.54°	1.11°	8.73°
GE2	4.23°	3.35°	3.62°	1.21°	8.59°
ASM	3.84°	2.42°	2.70°	0.56°	9.64°
MSGP	3.45°	**2.00°**	2.36°	0.43°	8.47°
BCC	6.74°	5.14°	5.55°	1.31°	15.00°
NN	5.16°	3.77°	4.06°	1.25°	11.49°
SVR(2D)	6.15°	5.15°	5.39°	1.64°	12.34°
SVR(3D)	4.14°	3.23°	3.53°	0.95°	9.03°
SSS	3.99°	3.24°	3.46°	1.64°	7.61°
GMP	6.00°	3.98°	4.53°	1.00°	14.27°
GMD$_V$	5.99°	4.03°	4.52°	1.08°	14.36°
GMD$_{VV}$	6.25°	4.25°	4.85°	1.25°	14.74°
Exemplar	4.94°	4.36°	4.56°	2.41°	8.28°
MC	3.25°	2.20°	2.55°	**0.30°**	8.13°
SA	6.44°	6.09°	6.10°	2.97°	10.63°
Nearest2	4.29°	3.00°	3.22°	0.78°	10.13°
Nearest-10%	4.29°	2.98°	3.21°	0.75°	10.12°
Nearest-30%	4.26°	2.95°	3.20°	0.75°	10.03°
No-1-Max	4.46°	3.51°	3.87°	1.22°	9.05°
No-3-Max	4.18°	3.26°	3.54°	1.14°	8.54°
MD	5.46°	4.86°	5.02°	1.85°	10.11°
LMS	3.16°	2.51°	2.67°	0.87°	**6.54°**
ELM	3.74°	2.49°	2.82°	0.76°	8.69°
SVRC	3.26°	2.08°	2.39°	0.68°	7.78°
AOS-ELM	**3.13°**	2.06°	**2.35°**	0.73°	7.13°

表 5 Gray Ball 数据集上各种颜色恒常性算法性能的比较

算法	Mean	Median	Trimean	Best-25%	Worst-25%
GW	12.98°	10.76°	11.33°	3.27°	26.20°
SoG	11.61°	10.39°	10.60°	3.55°	22.16°
WP	12.74°	10.34°	11.29°	2.30°	26.41°
gGW	12.12°	10.58°	10.94°	3.63°	23.15°
GE1	11.13°	9.15°	9.70°	3.09°	22.12°
GE2	11.07°	9.55°	9.89°	2.91°	22.02°
ASM	11.35°	9.41°	9.94°	3.20°	22.59°
MSGP	10.27°	7.76°	8.58°	2.08°	22.19°
BCC	11.98°	10.14°	10.56°	2.36°	24.57°
NN	11.81°	9.75°	10.25°	3.23°	23.85°
SVR(2D)	13.58°	11.80°	12.67°	3.92°	25.67°
SVR(3D)	9.99°	8.39°	8.74°	2.76°	20.02°
SSS	10.43°	8.74°	9.20°	3.06°	20.62°
GMP	14.19°	11.98°	12.72°	3.18°	28.64°
GMD$_V$	12.54°	10.36°	11.01°	2.90°	25.65°
GMD$_{VV}$	13.21°	10.96°	11.62°	2.89°	27.22°
Exemplar	8.00°	6.44°	6.76°	1.94°	16.73°
MC	8.81°	5.61°	6.78°	**1.50°**	19.1°
SA	10.12°	8.95°	9.20°	2.94°	19.40°
Nearest2	11.42°	9.25°	9.83°	2.63°	23.63°
Nearest-10%	11.34°	9.14°	9.76°	2.62°	23.50°
Nearest-30%	11.32°	9.27°	9.83°	2.66°	23.30°
No-1-Max	10.43°	9.07°	9.40°	2.97°	20.14°
No-3-Max	10.46°	8.95°	9.33°	3.06°	20.27°
MD	10.14°	8.80°	9.17°	2.89°	19.56°
LMS	8.98°	7.41°	7.74°	2.44°	18.27°
ELM	8.94°	7.09°	7.62°	2.20°	18.58°
SVRC	9.02°	6.75°	7.43°	2,10°	19.93°
AOS-ELM	**7.31°**	**5.39°**	**5.87°**	1.83°	**15.93°**

从表 4 可以看出，在 Gehler-Shi 图像集上本文提出的方法在 mean、trimean 这 2 项指标上均优于其他所有对比算法. 在有监督的单一颜色恒常性算法中，基于稀疏表示学习的 MC 算法表现最好. 在无监督的融合颜色恒常性算法中，Nearest2、Nearest-10% 和 Nearest-30% 的性能相差无几，但是都明显优于 SA 和 MD. 本文方法的均值角度误差和三均值角度误差分别是 3.13°和 2.35°，比表现最好的有监督融合颜色恒常性算法 SVRC 降低了 3.99%和 1.67%，比表现最好的有监督单一颜色恒常性算法 MC 降低了 3.69%和 7.84%. WP 在此数据集上的表现最差，这是因为该算法假设 RGB 颜色通道的最大响应是由场景中的白色表面引起的，而这种假设条件在很多实际情况下难以得到满足. 由于 GE1 和 GE2 分别利用了图像的一阶导数和二阶导数信息，因此其性能要显著优于经典的 GW 算法及其衍生算法 SoG 和 gGW. Exemplar 算法通过测试图像中场景物体表面与训练样本中最近邻表面的直方图匹配来估计光源的颜色，在不具备图像相似性的 Gehler-Shi 数据集上，其性能并不比无监督的 gGW、GE1、GE2、ASM 和 MSGP 好. 相比较于大多数非生理机制的无监督颜色恒常性算法（比如 GW、SoG、WP、gGW、GE1、GE2），根据初级视皮层（V1 区）和高级视皮层（V4 区）中神经元感受野机制建立的 ASM 模型在光照估计的精度上有了显著的提升. 此外，MSGP 将场景光照估计任务转化为灰色像素检测结合均值漂移聚类技术，在此数据集上取得了最小的中值角度误差 2.00°，略小于 MC、SVRC 和本文方法的角度误差中值 2.20°、2.08°和 2.06°. 另一方面，从图 3(a)可知，在 5%的显著性水平下，本文方法、MSGP、MC 和 SVRC 相互之间的误差分布没有显著差异，说明这几种方法的光照估计性能基本相当. 除此之外，与其他 25 种颜色恒常性对比算法相比，本文方法得分最高，因此具有较为明显的性能优势. 最后，本文方法在最高 25%角度误差均值指标上的结果几乎比其他所有对比算法都要好（除了 LMS），这说明了本文方法具有较强的鲁棒性.

表 5 比较了本文提出的方法和其他颜色恒常性对比算法在 Gray Ball 图像集上的光照估计性能，可以看出，本文方法在多项指标上均取得了最好的结果. 其中，本文方法的中值角度误差和三均值角度误差分别达到了 5.39°和 5.87°，比表现最好的无监督单一颜色恒常性算法 MSGP 下降了 30.5%和 31.6%，比表现最好的有监督单一颜色恒常性算法 MC 下降了 3.9%和 13.4%，比表现最好的无监督融合颜色恒常性算法 MD 下降了 38.8%和 36%，比表现最好的有监督融合颜色恒常性算法 SVRC 下降了 20.1%和 21%. 此外，本文方法的最高 25%角度误差均值要比其他所有对比算法都要小，说明其具有很好的鲁棒性. 无监督的单一颜色恒常性算法在此数据集上都没有能够取得较为理想的光照估计结果，这是因为此类方法大都是建立在对光源光谱分布或者场景图像颜色分布的假设条件下进行的，因而只对某些符合条件的图像有效，对于不同光照和不同场景内容的图像，其应用范围具有局限性. 例如，GW 认为 RGB 颜色通道反射率的平均是无色差的，但是当场景中出现大面积纯色物体时，该算法估计的光源颜色就会偏向于物体的颜色. 此外，尽管 MSGP、MC 和 SVRC 在 Gehler-Shi 图像集上取得了非常好的光照估计结果，但在 Gray Ball 图像集上本文方法相比于它们具有更为明显的性能优势. MSGP 利用图像中的灰色像素来估计光源的颜色，其性能依赖于灰度点检测的准确性，然而判断一个像素点是否为灰色本身就是一个病态的问题. BCC 假设物体表面反射率和光照的概率分布相互独立且服从高斯分布，通过最大后验概率或最大似然函数估计图像成像时的光照颜色，但是这种假设在现实场景中一般很难得到满足，从而导致光照估计结果不够准确. GMP 及其衍生出的 GMD_V 和 GMD_{VV} 则认为在特定的光照条件下场景中物体表面能够观察到的颜色有限并且构成一个封闭的凸包，又称之为观察色域. 通过学习待估计图像的观察色域与标准色域之间的映射关系来计算光源的颜色. 但是，如果图像中的颜色比较单一或者图像的颜色不够丰富，其光照估计性能就会下降. 有监督的融合颜色恒常性算法将不同算法的光照估计结果作为图像的特征，性能不仅明显优于传统的使用高维二值化色度直方图特征的 NN、SVR(2D)和 SVR(3D)，而且显著优于直接对候选算法的结果进行加权平均的无监督融合颜色恒常性算法（比如 SA、Nearest2、Nearest-10%、Nearest-30%、No-1-Max、No-3-Max、MD）. 在输入特征向量相同的情况下，AOS-ELM 的各项性能明显优于 LMS、ELM 和 SVRC，这也说明了本文方法具有较好的泛化能力. 虽然 SVRC 的角度误差中值比 ELM 和 LMS 小，但是从图 3(b)中

可以看出这三种融合算法的误差分布在置信度95%的情况下不存在明显的差异.最后,在所有对比算法中,本文方法取得了最高得分,这也说明其具有较为明显的性能优势.

图4对比显示了本文提出的方法与基于ELM的光照估计融合算法以及基于支持向量回归的SVRC光照估计融合算法的隐藏层节点个数.这三种方法的输入和输出向量维度均相同.从实验结果来看,本文提出的方法需要的隐节点个数最少,表明了所提算法具有非常紧凑的网络结构.同时,还可以看出SVRC需要的支撑向量的个数远远高于ELM和本文方法.

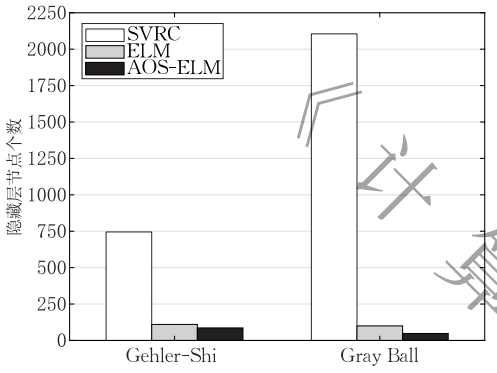


图4 本文方法和SVRC、ELM在Gehler-Shi数据集和Gray Ball数据集上使用的隐节点个数比较

5.4.3 与基于深度学习的颜色恒常性计算方法的对比实验

在Gehler-Shi图像集上,我们将本文方法与几种基于卷积神经网络的颜色恒常性计算方法进行了比较,主要有DS-Net[51]、SqueezeNet-FC4[52]以及Qiu等人[53]提出的光照估计方法.相对于传统机器学习方法,基于深度学习的颜色恒常性计算方法最大的优势是可以利用大规模数据集和高性能GPU训练一个从图像到场景光照的端到端的多层网络结构来直接估计光源的颜色,而不需要手动提取特征.从表6可以看出,与本文方法相比,

表6 Gehler-Shi数据集上本文方法与3种基于卷积神经网络的颜色恒常性方法的参数数量和模型性能的比较

方法	♯参数	Mean	Median	Trimean	Best-25%	Worst-25%
DS-Net	≈17.3 M	1.90°	1.12°	1.33°	0.31°	4.84°
SqueezeNet-FC4	≈1.9 M	1.64°	1.18°	1.27°	0.38°	3.78°
Qiu	≈189 K	1.85°	1.31°	1.37°	0.44°	4.14°
本文	≈2 K	3.13°	2.06°	2.35°	0.73°	7.13°

DS-Net、SqueezeNet-FC4和Qiu等人提出的深度学习光照估计方法在模型性能上取得了较大的提高,这也说明使用深度卷积神经网络从图像中学习到的特征要优于传统的人工设计的颜色恒常性融合特征.

5.4.4 融合光照估计算法的图像颜色校正示例

为了更直观地表现实验结果,图5对比展示了几组采用本文提出的光照估计融合方法和其他3种融合算法对图像进行颜色校正后的实例.其中,上面两幅室内和室外场景图像取自Gehler-Shi数据集,下面两幅室内和室外场景图像取自Gray Ball数据集.第1列为原始图像,第2列是利用真实光照得到的校正结果,第3至6列分别是利用本文方法、LMS、ELM和SVRC的光照色度估计值进行校正得到的结果.图像的左下角给出了各个算法在此图像上估计出的光照色度值与成像时场景中的真实光照色度值之间的角度误差.可以看出,不管是室内场景还是室外场景的图像,本文方法的视觉效果与标准光照下的图像都非常接近,角度误差最小,进一步通过主观实验效果验证了本文方法的有效性.

总的来说,传统的无监督或者有监督单一颜色恒常性方法的性能不是受限于各种物体表面反射率或者成像的假设条件约束就是对图像训练集自身有很强的依赖性,因此只对某些符合特定条件的场景有效,并且在同一幅图像上不同算法的光照估计结果往往差异也很大.融合颜色恒常性方法为不同类型的图像选择合适的光照估计算法,在一定程度上弥补了单一颜色恒常性算法的局限性,从而进一步提高了图像光照估计的精度.无监督的融合颜色恒常性计算直接对候选算法的光照估计结果进行加权平均,这种融合方案虽然简单直观易于实现,但是通常无法得到最优的算法组合,对性能的提升有限.有监督的融合颜色恒常性计算利用机器学习技术寻找最优的加权融合权值,进一步提高了融合算法的准确度.由中值角度误差和隐节点数这两项指标的实验结果可以发现,基于AOS-ELM的颜色恒常性融合方案相比于传统的基于SVRC和ELM的融合颜色恒常性方法,在光照估计精度和模型复杂度之间取得了非常好的平衡,能够以较少的隐藏层节点实现很好的光照估计效果.

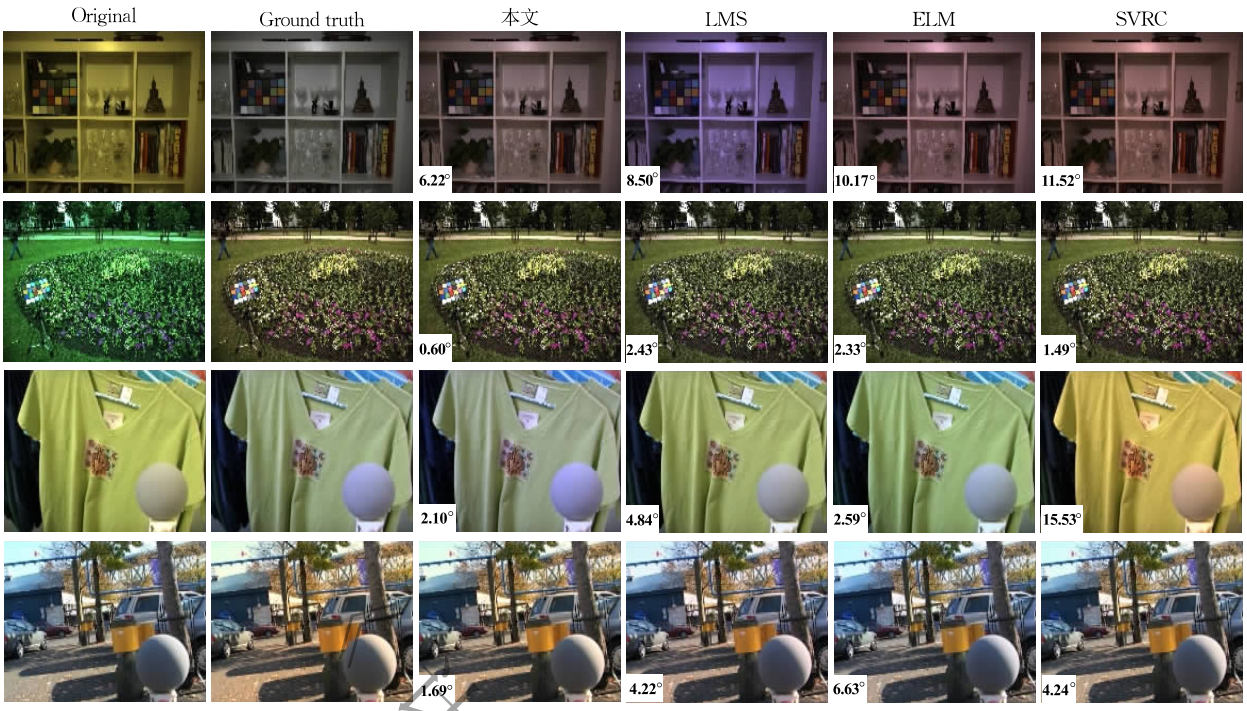


图 5　本文方法和其他 3 种融合颜色恒常性算法在 Gehler-Shi 数据集和 Gray Ball 数据集上的图像光照校正效果比较

6　结束语

本文提出了一种自适应正交搜索算法来学习 ELM 网络结构. 该方法首先利用标准 ELM 随机生成一组候选隐节点；然后交替执行正交化的前向选择和后向移除，在保证最优隐节点组合都被选入的前提下，尽可能地剔除不重要的隐节点；最后通过一个精心设计的向后移除过程对已选入的隐节点进行逐个检查，并将前一阶段残留的冗余节点从模型中进一步删除. 与传统的增量构造法和剪枝构造法在训练过程中只能增加隐节点或只能删除隐节点不同的是，本文方法基于隐节点输出向量和网络期望输出向量的绝对相关程度动态地增加或删除隐节点. 此外，本文方法吸收了前向选择和后向移除各自的优点，克服了彼此的不足，不仅能够反映隐藏层节点之间的相互影响与作用，而且具有对前面步骤中所犯的错误进行修正的能力.

本文提出的 AOS-ELM 算法为解决单隐层前馈神经网络的结构选取问题提供了一种新的思路. AOS-ELM 作为一种浅层网络学习方法具有较为坚实的理论基础. 我们不仅在 12 个基准回归数据集上比较了本文方法与 I-ELM、CI-ELM、EM-ELM、DAOI-ELM、OP-ELM 和标准 ELM 的网络复杂度和泛化性能，而且还将其应用于颜色恒常性计算建模，从角度误差的均值、中值、三均值等方面将本文方法与大量的颜色恒常性计算方法在两个标准颜色恒常性图像集上进行实验和分析，计算并比较了基于 ELM、SVRC 和本文方法的融合光照估计算法需要的隐节点个数. 通过分析大量实验结果可以看出，本文方法在解决小样本、非线性模式回归问题中能够产生一个较为紧凑的网络结构，同时表现出良好的泛化性能.

本文的工作主要集中在回归问题中 ELM 网络结构设计，下一步研究工作计划将其扩展到分类问题中，构建一种能够直接用于回归和多分类应用的统一学习框架. 如何将本文方法推广到多隐藏层前馈神经网络的模型选择中，更大程度地提高网络的学习效率和泛化性能，将是另一个值得研究的问题.

致　谢　中国科学院自动化研究所模式识别国家重点实验室李兵研究员在颜色恒常性计算实验部分提供了诸多指正与学术讨论，评审专家和《计算机学报》编辑部的老师提出了宝贵的意见与建议，籍此致上由衷的谢忱！

参　考　文　献

[1] Cybenko G. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems, 1989, 2(4): 303-314

[2] Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: Theory and applications. Neurocomputing, 2006, 70(1-3): 489-501

[3] Huang G B, Chen L, Siew C K. Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Transactions on Neural Networks, 2006, 17(4): 879-892

[4] Lu Hui-Juan, An Chun-Lin, Ma Xiao-Ping, et al. Disagreement measure based ensemble of extreme learning machine for gene expression data classification. Chinese Journal of Computers, 2013, 36(2): 341-348(in Chinese)
(陆慧娟, 安春霖, 马小平等. 基于输出不一致测度的极限学习机集成的基因表达数据分类. 计算机学报, 2013, 36(2): 341-348)

[5] Zhang Jing, Chen Yi-Qiang, Ji Wen. Two-tie image super-resolution based on CNN and ELM. Chinese Journal of Computers, 2018, 41(11): 2581-2597(in Chinese)
(张静, 陈益强, 纪雯. 基于 CNN 与 ELM 的二次超分辨率重构方法研究. 计算机学报, 2018, 41(11): 2581-2597)

[6] Xu Rui, Liang Xun, Qi Jin-Shan, et al. Advances and trends in extreme learning machine. Chinese Journal of Computers, 2019, 42(7): 1640-1670(in Chinese)
(徐睿, 梁循, 齐金山等. 极限学习机前沿进展与趋势. 计算机学报, 2019, 42(7): 1640-1670)

[7] Wang N, Er M J, Han M. Parsimonious extreme learning machine using recursive orthogonal least squares. IEEE Transactions on Neural Networks and Learning Systems, 2014, 25(10): 1828-1841

[8] Kwok T Y, Yeung D Y. Constructive algorithms for structure learning in feedforward neural networks for regression problems. IEEE Transactions on Neural Networks, 1997, 8(3): 630-645

[9] Parekh R, Yang J, Honavar V. Constructive neural-network learning algorithms for pattern classification. IEEE Transactions on Neural Networks, 2000, 11(2): 436-451

[10] Huang G B, Chen L. Convex incremental extreme learning machine. Neurocomputing, 2007, 70(16-18): 3056-3062

[11] Feng G, Huang G B, Lin Q, et al. Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Transactions on Neural Networks, 2009, 20(8): 1352-1357

[12] Ying L. Orthogonal incremental extreme learning machine for regression and multiclass classification. Neural Computing and Applications, 2016, 27(1): 111-120

[13] Han F, Zhao M R, Zhang J M, et al. An improved incremental constructive single-hidden-layer feedforward networks for extreme learning machine based on particle swarm optimization. Neurocomputing, 2017, 228: 133-142

[14] Zou W, Xia Y, Li H. Fault diagnosis of Tennessee-Eastman process using orthogonal incremental extreme learning machine based on driving amount. IEEE Transactions on Cybernetics, 2018, 48(12): 3403-3410

[15] Rong H J, Ong Y S, Tan A H, et al. A fast pruned-extreme learning machine for classification problem. Neurocomputing, 2008, 72(1-3): 359-366

[16] Miche Y, Sorjamaa A, Bas P, et al. OP-ELM: Optimally pruned extreme learning machine. IEEE Transactions on Neural Networks, 2009, 21(1): 158-162

[17] Alencar A S C, Neto A R R, Gomes J P P. A new pruning method for extreme learning machines via genetic algorithms. Applied Soft Computing, 2016, 44: 101-107

[18] De Campos Souza P V, Araujo V S, Guimaraes A J, et al. Method of pruning the hidden layer of the extreme learning machine based on correlation coefficient//Proceedings of the IEEE Latin American Conference on Computational Intelligence. Guadalajara, Mexico, 2018: 1-6

[19] García-Pedrajas N, Hervás-Martínez C, Muñoz-Pérez J. COVNET: A cooperative coevolutionary model for evolving artificial neural networks. IEEE Transactions on Neural Networks, 2003, 14(3): 575-596

[20] Huang G B, Zhou H, Ding X, et al. Extreme learning machine for regression and multiclass classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2011, 42(2): 513-529

[21] Golub G H, Van Loan C F. Matrix Computations. 3rd Edition. Maryland, USA: Johns Hopkins University Press, 2012

[22] Strang G. Introduction to Linear Algebra. 5th Edition. Massachusetts, USA: Wellesley Cambridge Press, 2016

[23] Meyer C D. Matrix Analysis and Applied Linear Algebra. Philadelphia, USA: Siam Press, 2000

[24] Weisberg S. Applied Linear Regression. 4th Edition. Hoboken, USA: Wiley Press, 2013

[25] Zhang L, Li K. Forward and backward least angle regression for nonlinear system identification. Automatica, 2015, 53: 94-102

[26] Alcalá-Fdez J, Fernández A, Luengo J, et al. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic & Soft Computing, 2011, 17(2-3): 255-287

[27] Gehler P V, Rother C, Blake A, et al. Bayesian color constancy revisited//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Alaska, USA, 2008: 1-8

[28] Shi L, Funt B. MaxRGB reconsidered. Journal of Imaging Science and Technology, 2012, 56(2): 1-10

[29] Hemrit G, Finlayson G D, Gijsenij A, et al. Rehabilitating the colorchecker dataset for illuminant estimation//Proceedings of the 26th Color and Imaging Conference. Vancouver, Canada, 2018: 350-353

[30] Ciurea F, Funt B. A large image database for color constancy research//Proceedings of the 11th Color and Imaging Conference. Arizona, USA, 2003: 160-164

[31] Bianco S, Ciocca G, Cusano C, et al. Improving color constancy using indoor-outdoor image classification. IEEE Transactions on Image Processing, 2008, 17(12): 2381-2392

[32] Gijsenij A, Gevers T, Van De Weijer J. Computational color constancy: Survey and experiments. IEEE Transactions on Image Processing, 2011, 20(9): 2475-2489

[33] Hordley S D, Finlayson G D. Reevaluation of color constancy algorithm performance. Journal of the Optical Society of America A, 2006, 23(5): 1008-1020

[34] Barron J T. Convolutional color constancy//Proceedings of the IEEE International Conference on Computer Vision. Santiago, Chile, 2015: 379-387

[35] Van De Weijer J, Gevers T, Gijsenij A. Edge-based color constancy. IEEE Transactions on Image Processing, 2007, 16(9): 2207-2214

[36] Akbarinia A, Parraga C A. Colour constancy beyond the classical receptive field. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, 40(9): 2081-2094

[37] Qian Y, Pertuz S, Nikkanen J, et al. Revisiting gray pixel for statistical illumination estimation//Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. Prague, Czech Republic, 2019: 36-46

[38] Cardei V C, Funt B, Barnard K. Estimating the scene illumination chromaticity by using a neural network. Journal of the Optical Society of America A, 2002, 19(12): 2374-2386

[39] Xiong W, Funt B. Estimating illumination chromaticity via support vector regression. Journal of Imaging Science and Technology, 2006, 50(4): 341-348

[40] Chakrabarti A, Hirakawa K, Zickler T. Color constancy with spatio-spectral statistics. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 34(8): 1509-1519

[41] Forsyth D A. A novel algorithm for color constancy. International Journal of Computer Vision, 1990, 5(1): 5-35

[42] Gijsenij A, Gevers T, Van De Weijer J. Generalized gamut mapping using image derivative structures for color constancy. International Journal of Computer Vision, 2010, 86(2-3): 127-139

[43] Joze H R, Drew M S. Exemplar-based color constancy and multiple illumination. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014, 36(5): 860-873

[44] Li B, Xiong W, Hu W, et al. Multi-cue illumination estimation via a tree-structured group joint sparse representation. International Journal of Computer Vision, 2016, 117(1): 21-47

[45] Li B, Xiong W, Hu W, et al. Evaluating combinational illumination estimation methods on real-world images. IEEE Transactions on Image Processing, 2013, 23(3): 1194-1209

[46] Chang C C, Lin C J. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2011, 2(3): 1-27

[47] Cardei V C, Funt B. Committee-based color constancy//Proceedings of the 7th Color and Imaging Conference. Scottsdale, USA, 1999(1): 311-313

[48] Bianco S, Gasparini F, Schettini R. Consensus-based framework for illuminant chromaticity estimation. Journal of Electronic Imaging, 2008, 17(2): 1-9

[49] Li B, Xiong W, Xu D, et al. A supervised combination strategy for illumination chromaticity estimation. ACM Transactions on Applied Perception, 2010, 8(1): 1-17

[50] Li B, Xiong W, Hu W, et al. Evaluating combinational color constancy methods on real-world images//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Colorado, USA, 2011: 1929-1936

[51] Shi W, Loy C C, et al. Deep specialized network for illuminant estimation//Proceedings of the 14th European Conference on Computer Vision. Amsterdam, Netherlands, 2016: 371-387

[52] Hu Y, Wang B, et al. FC^4: Fully convolutional color constancy with confidence-weighted pooling//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Hawaii, USA, 2017: 4085-4094

[53] Qiu J, Xu H, et al. Color constancy by reweighting image feature maps. IEEE Transactions on Image Processing, 2020, 29: 5711-5721

XU Rui, Ph. D. candidate. His research interests include image processing and machine learning.

LIANG Xun, Ph. D., professor, Ph. D. supervisor. His research interests include data mining, neural networks, and social computing.

MA Yue-Feng, Ph. D., associate professor. His research interests include support vector machine, neural networks.

QI Jin-Shan, Ph. D., associate professor. His research interests include social computing, data mining.

Background

A critical step in learning a single hidden layer feedforward network for a given problem is the choice of its architecture, since the selection of hidden neurons has a profound impact on generalization capability and complexity of the network.

Too large a network will lead to overfitting and poor generalization performance, but too small a size cannot learn the problem well. A simple and common method to find an appropriate network size is by trial and error. This approach, although straightforward, is not likely to yield optimal or nearly optimal structures especially when adopted by inexperienced users. Research on constructive and destructive algorithms is an effort made towards the automatic design of architectures. Roughly speaking, a constructive algorithm starts with a small network and gradually add new hidden neurons until a satisfactory result is obtained while a destructive algorithm starts with a large network and then deletes redundant hidden neurons until there is only one hidden neuron left, or until some stopping criterion is satisfied. For both constructive and destructive approaches, one of the major issues which has not been addressed much is that every step of the structure adjusting strategy is only capable of adding or pruning neurons from the network, while some of the hidden neurons may play a very minor role in the network output. Therefore, none of these methods can guarantee that the selected network architecture will be close to optimal or will generalize well.

The objective of this work is to produce a parsimonious network structure with a good capacity of generalization for solving a specific task. To this end, by formulating the whole problem as a subset model selection, an adaptive orthogonal search algorithm for network structure learning of ELM is presented. The proposed method can be summarized as the synergy of orthogonal forward selection and backward elimination that searches for topologies, based on the influence of adding or removing each hidden neuron on the error, aiming to concurrently optimize performance while minimizing network complexity. Simulation results concerning both data regression and color constancy indicate that the proposed AOS-ELM can strike an excellent balance between the accuracy and compactness, and thereby contributing to remarkable generalization.

This paper is supported by the National Natural Science Foundation of China under Grant Nos. 62072463 and 71531012, the National Social Science Foundation of China under Grant No. 18ZDA309, the Natural Science Foundation of Beijing under Grant No. 4172032, the Jingdong Mall Research No. 413313012, the Opening Project of State Key Laboratory of Digital Publishing Technology of the Founder Group, the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant No. 19KJB520024, and the Higher School in Jiangsu Province College Students' Practice Innovation Training Programs under Grant No. 201910323057Y.