改进CS\_BP\_Adaboost算法在空气质量判断中的应用

**摘要：**针对CS算法采用的随机产生初始值表达能力不足的问题，采用tent混沌初始化结合反向初始化的方式优化布谷鸟算法的初值。又考虑了tent的初始值不同，所产生的CS初始种群也差异明显的情况，引入了Adaboost算法来组合多个差异足够大的tent初始值和阈值产生的CS-BP模型，这种方式在解决了tent初值确定的问题以外，还能产生多个具有较大差异的模型，为后续Adaboost算法产生的强分类器提供了良好的差异明显的多个弱分类器。最终的测试结果表明，该方式产生的强分类器在性能上优于CS\_BP模型，得到了明显提升。

**关键字：**布谷鸟算法（CS）；BP神经网络；Adaboost算法；tent混沌初始化；反向初始化；

# 0 引言

与人们生产生活越来越便捷、智能相反，环境恶化的问题却越加严重。为了更好的监测环境情况，在综合考虑了设备功能性、适用性以及价格等重要因素之后，本文将使用STM32系列设备来完成数据的采集以及判断功能，并将神经网络模型移植到STM32上，以此来优化网络传输的数据量、降低设备传输功耗以及摆脱网络依赖等情况。

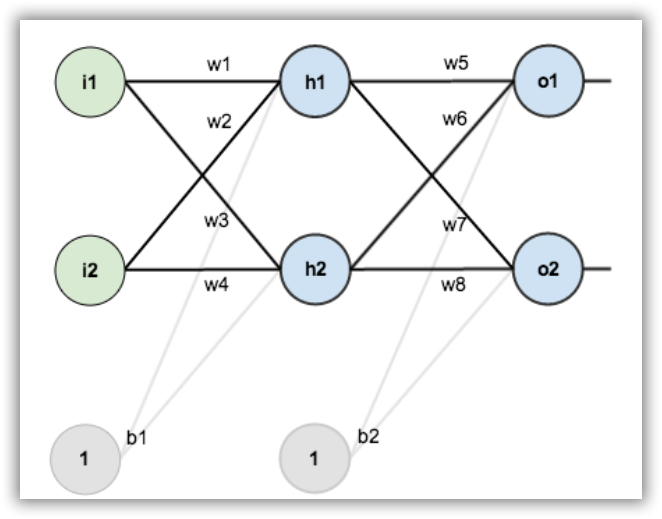
在考虑到STM32系列单片机的RAM和Flash容量较小，难以支撑复杂的神经网络模型之后，本文将移植三层的BP神经网络到STM32上，并使用改进的布谷鸟算法优化BP神经网络，使其能达到更好的效果。最后，针对改进的布谷鸟算法的特点（tent初始化的参数确定），引入Adaboost算法组合多个优化的CS\_BP神经网络，得到最终的强分类器，本文将针对贵阳2018年-2019年的空气质量数据作为模型的训练数据，完成该应用的验证。

（待补充）

# 相关算法描述

## 1.1 BP神经网络

反向传播算法（BackpropagationAlgorithm）是深度学习中重要的思想基础，通过该方式对输入参数的权值进行更新，逐渐逼近一个局部最优值或全局最优值。该方法不需要事先知道输入和输出之间的映射关系，而是通过大量的数据来学习这种映射关系。如下图1所示，一个简单的BP神经网络拓扑结构分为三个层次，分别是输入层、隐藏层、输出层。



**图1 BP神经网络结构**

## 1.2 CS算法

布谷鸟算法是群体智能算法的一种，算法思想来源于布谷鸟特殊的生活习性。这种鸟会将生下来的蛋放到其他鸟的鸟巢去，这在算法的寻优当中就是全局最优化，该算法采用莱维飞行来模拟全局随机寻优。但鸟蛋有概率被寄养的鸟发现，发现之后该蛋会被丢弃，布谷鸟则会在蛋被丢弃之后在附加随机选择一个位置重新安置一个新蛋，这里则对应算法的局部寻优部分，采用局部随机行走算法来完成。布谷鸟算法比GA、PSO这些算法要简单，但已有研究表明CS的效果是要优于这些算法。

## 1.2.1 布谷鸟算法的实现过程

在对上述算法思想的总结之后，具体算法的步骤如下：

Step1：在最初可行域内随机生成一组解（布谷鸟）。

Step2：记录这些点的适应度值，并单独记录最优解的位置及其适应度值。

Step3：通过莱维飞行更新这些解的位置，并计算新解的位置，与之前的解对比适应度值，留下更优的解。更新位置的计算公式如下：

(1)

式（1）中为更新的位置，为之前的位置，为步长，代表点乘。

（2）

式（2）中u服从N（0，）正态分布，v服从N（0，）正态分布。

(3)

(4)

式（3）中表示标准伽玛函数,。

Step4：更新后的解有一定的概率被抛弃，被抛弃的解将在附近寻找一个新的位置，没被抛弃的保存原样。这里采用局部随机行走来完成这个过程，更新公式如下：

（5）

式（5）中为更新的位置，为之前的位置，为步长系数，H为跃迁函数（x大于等于1时数值为1，否则为0），和为t时刻随机选取的两个解。

Step5：重新计算适应度值，并保存这一代最好的解和适应度。

Step6：得到新的一组解从Step3重新开始迭代，直到达到适应度要求或是迭代次数完成。

## 1.3 Adaboost算法

Adaboost（Adaptive Boosting）算法的核心思想是使用多个具有一定差异的弱分类器组合，来实现更加有效的效果。它会将前一个分类器分类错误的样本权重增大，分类正确的样本权重减少，这样当后面的分类器在同一个错误样本上再次分类错误的时候将会得到更大的误差，从而导致这次的分类器所占权重减少。后续的分类器将重复这个过程，最后得到的强分类器将具有修正弱分类器分类错误样本的能力。本文中弱分类器就是一个三层BP神经网络，强分类器则为它们的组合。

## 1.3.1 Adaboost算法的实现过程

Step1：初始化训练样本权值D，最开始所有样本权值相同，总和为1。

(6)

式（6）中N为样本总数。

Step2：训练一个弱分类器，并计算该分类器的误差

(7)

式（7）中，表示第m个弱分类器的误差，表示第m个弱分类器的第i个样本权值，I为指示函数，为第m个弱分类器。

Step3：计算该弱分类器权重。

(8)

式（8）中表示第m个弱分类器的权重。

Step4：更新样本权重分布。

(9)

(10)

式（10）中是规范化因子，目的是使中所有元素和为1。

Step5：用更新的权重分布区计算下一个弱分类器的权重。

Step6：构建最终的强分类器。

(11)

式（11）中最终强分类器G(x)为所有弱分类器的线性加权和。

# 2 空气质量判断模型构建

## 2.1 改进的CS算法优化BP神经网络

CS算法相比较其他智能搜索算法有着结构简单、搜索效果好的优势。传统的CS算法使用伪随机数生成器的方式产生初始解，这种方式虽然有着简单理解与实现的特点，但它生成的初始解却不够均匀，尤其是当数据维度较高的时候缺陷更加明显。本文在生成CS算法初始解的时候采取了Tent混沌初始化和反向初始化相结合的方式来产生在解空间中更加均匀，表达能力更强的初始解。该方式在Tent算法产生的初始解的基础上取其反向解，并计算这两组解的适应度，依据贪婪法则留下其中的最优，以此构成初始解。

Tent初始化的表达式为：

反向初始化的表达式为：

在经过多次的实验后发现混沌初始化虽然在多样性、收敛性、解空间表达能力上提高CS算法的性能，但对于初值以及参数的选择却对算法的稳定性造成了较大的影响。不同的初值和参数带来的最终效果差异很大。针对这个问题，本文引入了Adaboost算法。

## 2.2 改进的CS\_BP\_Adaboost算法

对于混沌初始化的初值和参数问题，本文采用Adaboost构建多个弱分类器的方式来解决。通过生成多组具有足够差异的Tent初值和参数，分别构建多个CS-BP模型，之后按照Adaboost算法的过程求解这些弱分类器的权值，得到最终的强分类器。由于Tent的初值和参数具有足够的差异，所以最终产生的CS-BP模型也具备差异性。而差异性越大的弱分类器构建的强分类器效果通常越好，所以该方法具备一定的适用性。

这里由于Adaboost算法是用来完成二分类情况的，所以对相应的部分作出调整，使其能够适应多分类，具体改动如下：

1）：模型权值计算公式改动。

（12）

式（12）中在原来的基础上增加了log（K-1）部分，其中的K对应分类类别，在该公式下因为弱分类器是完成多分类，所以对弱分类器的要求的准确率大于1/k即可。

2）：更新样本权重计算公式改动，公式（10）中的在多分类情况不再适用，将先计算模型预测误差，样本预测正确保存为0，样本预测错误保存为1，得到一个样本预测向量Err，最终公式为：

（13）

这样预测正确的样本权重不变，预测错误的样本权重增加。在经过规范化之后预测正确的样本权重减小，预测错误的样本权重增加。

# 3实验结果与模型移植效果分析

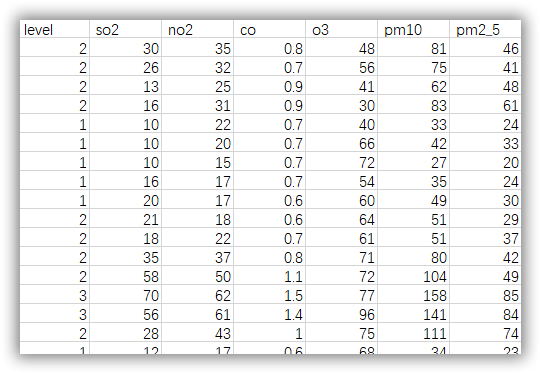
## 3.1 实验环境

实验电脑操作系统为windows，软件部分采用python语言开发，采用TensorFlow2.0构建神经网络。模型移植到STM32部分采用MDK5、STM32CubeMx6.2.0以及ST提供的X-CUBE-AI库完成。

## 3.2 数据选择与预处理

数据采用青悦公开数据提供的贵州省2021年1月到9月的城市每日空气质量数据。

数据预处理首先排除掉一部分不需要的数据属性，删除有样本元素为None的样本，去除非数值属性之后得到的最终数据如下图所示：



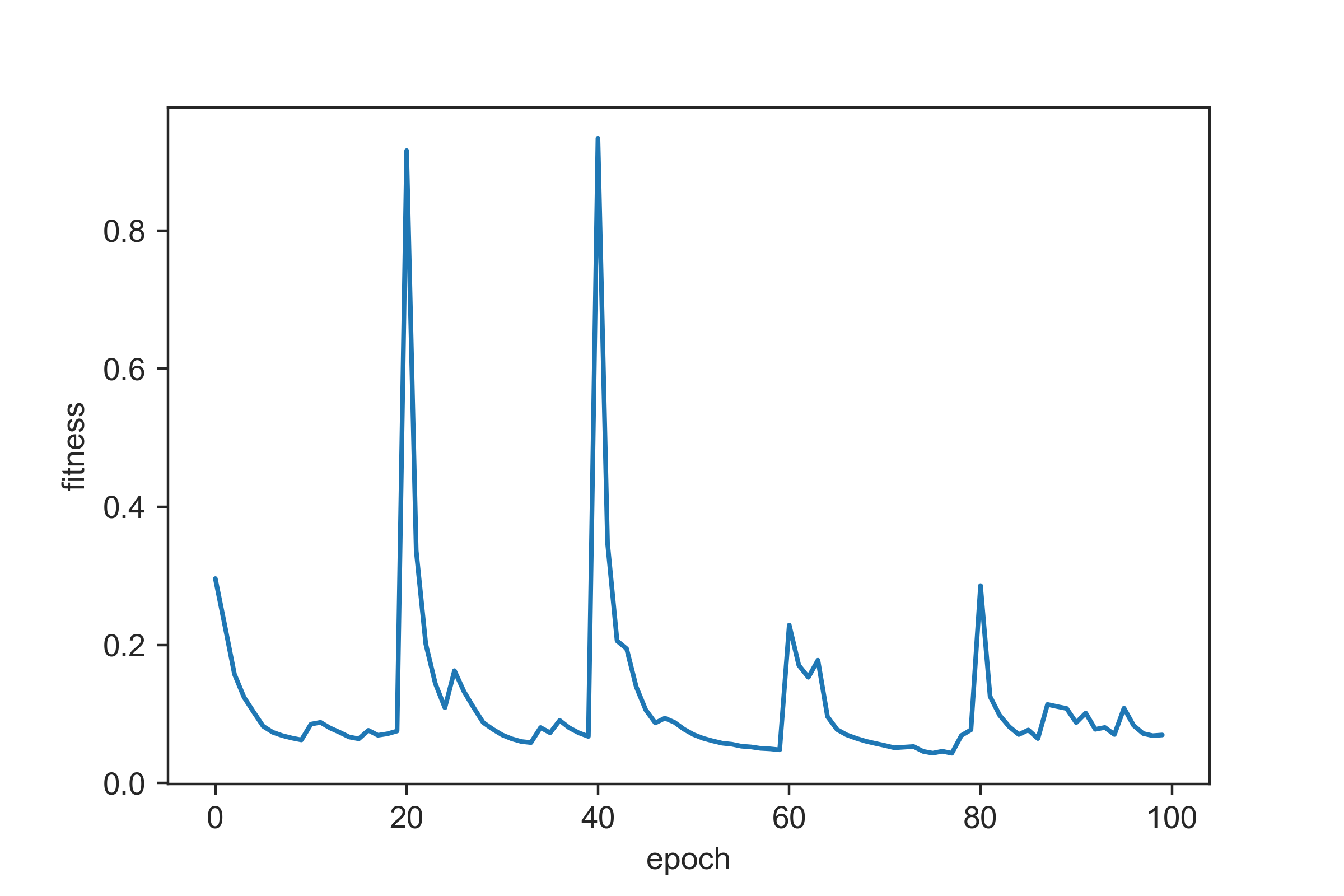
**图2 部分数据展示**

之后进行逐维零均值处理，以及逐维归一化处理。之后将数据转化为TensorFlow张量，在转化为Dataset，方便后续处理。

## 3.3 实验过程

本次实验使用布谷鸟算法产生20组初始解，迭代20次，Adaboost算法采取5个弱分类器结合，适应度函数选择mse。

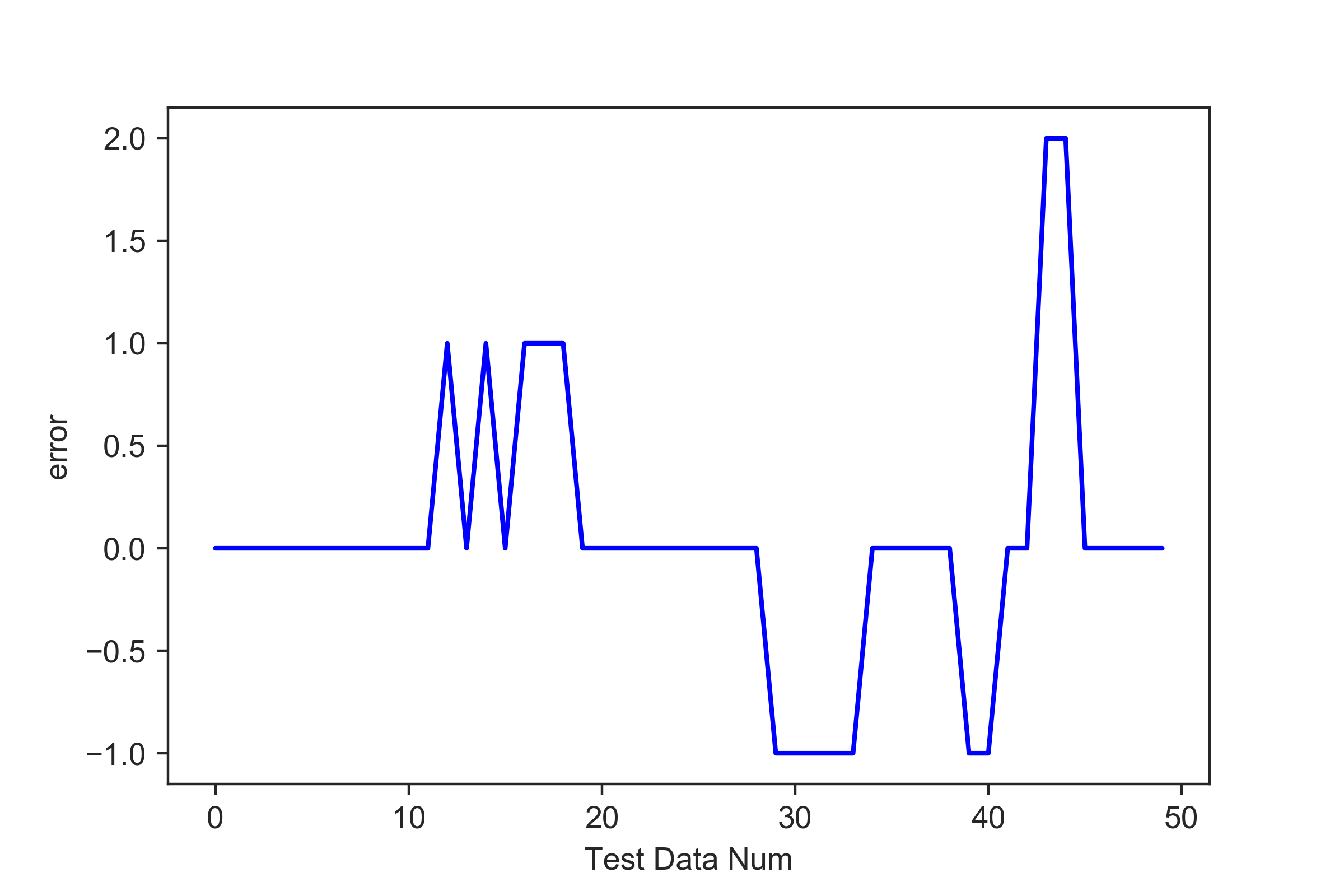
在经过了Tent混沌初始化结合反向初始化后的布谷鸟算法性能上得到了较大的提升，如下图3是5个弱分类器的20次CS算法迭代过程中适应度函数的进化情况，每20个epoch为1个弱分类器结果。



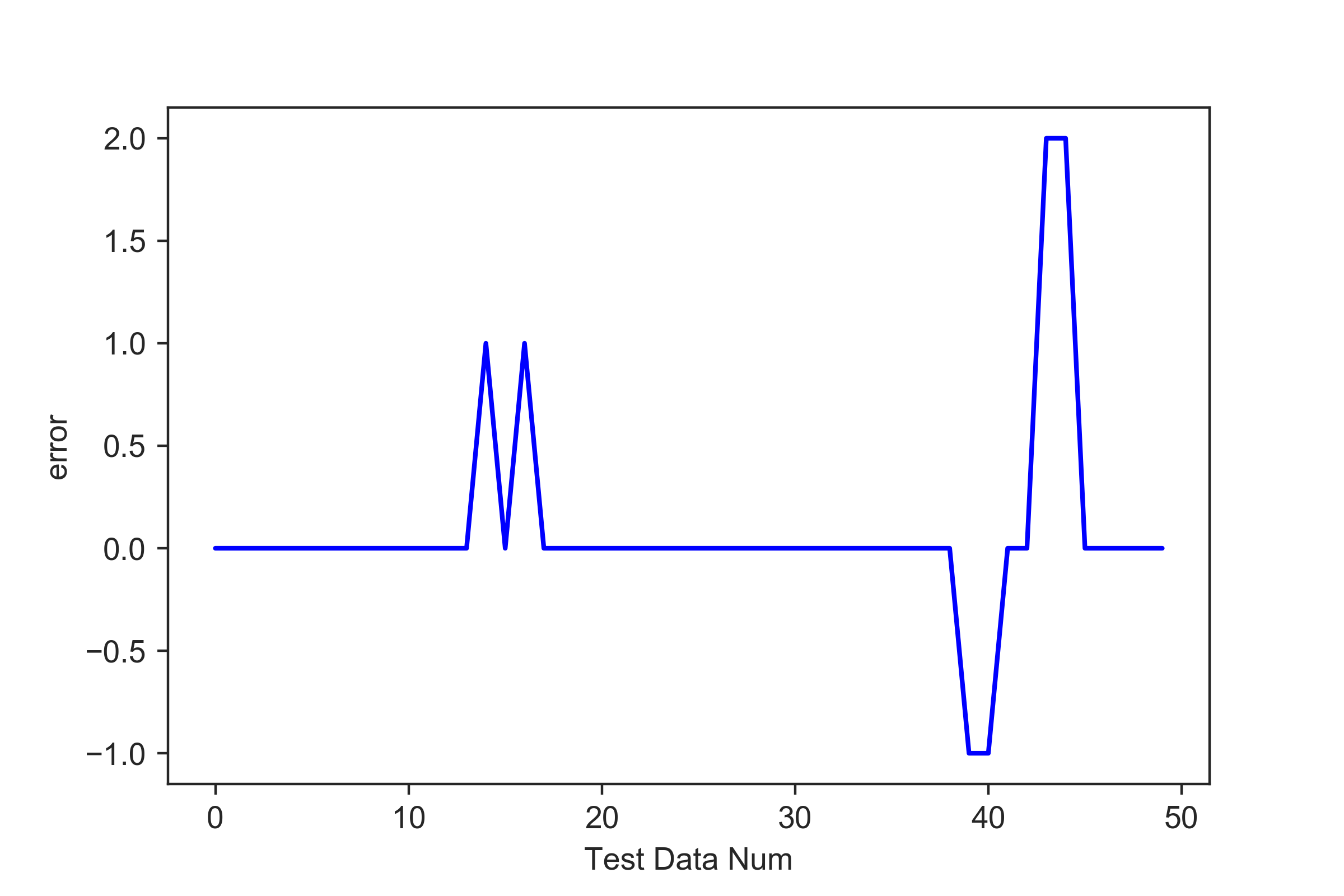
**图3 优化CS算法进化效果**

可以看到改进效果还是比较稳定而且明显的。每个弱分类器的最优解适应度值都是很小的。

下图4是改进CS-BP在50组测试数据上的预测效果。下图5是改进CS-BP-Adaboost算法的预测效果，可以明显观察到，改进CS-BP-Adaboost算法修正了部分改进CS-BP算法的误差，取得了不错的效果，证明了算法的有效性。



**图4 优化CS-BP算法预测效果**



**图5 优化CS-BP-Adaboost算法预测效果**

## 3.4 实验结果分析

## 3.5 模型移植效果分析

移植效果稳定，该方式具备一定的实用性，可以应用在一些现实问题的解决当中。

# 4结语