Online Self-Learning Stochastic Configuration Networks for Nonstationary Data Stream Analysis

摘要

随机配置网络（Stochastic configuration networks，SCN）自 2017 年被提出以来吸引了许多学者的研究，其能够随机给隐含层节点的输入权重和偏置赋值，并利用监督机制来确定最优的输入权重和偏置，有着快速收敛的能力。本次要复现的论文以 SCN 为基础模型，提出了一种自适应在线随机配置网络（online self-learning stochastic configuration network，OSL-SCN），能够在线调整自己的结构，以适应复杂的数据流。本人的工作是复现了 OSL-SCN，并在此基础上加入了类不平衡处理机制。实验表明，本次的改进工作在处理具有类不平衡和概念漂移的数据流分类上，是有效的。

关键词：随机配置网络；在线分类；概念漂移

1 引言

随着现代科技的快速发展，大量的数据从各种应用中产生，挖掘其中隐含的有用信息得到人们的关注与研究。数据流是指带时间戳的数据序列 [1]，是一种新型的数据形式。和传统的静态数据不同，数据流具有无限、变速率、可进化、难存储等特点，使得一些用于静态数据的机器学习方法变得不适用。随着神经网络算法的发展，一些神经网络算法被应用于数据流挖掘任务中，如 BP 算法，但收敛速度较慢，训练时间较长。与其相比，随机化学习方法拥有较好的收敛有效性 [6]。与传统的随机化神经网络不同，SCN [5] 引入了监督机制，使得网络能自主对参数赋值和调整结构。近些年来，针对 SCN 的扩展及变体，得到了许多学者的研究，包括本文所要复现的 OSL-SCN [3]。

本文首先介绍了 SCN 基础模型的工作原理，再介绍了 OSL-SCN 的工作原理，然后介绍 OSL-SCN 的复现细节及提出自己的改进，最后给出实验分析与结论。

2 相关工作

本部分对基础模型 SCN 进行简要的概括与描述。

2.1 SCN 模型结构

SCN 是一种增量式的快速建模方法，和传统的单隐层前馈神经网络 (single-hidden layer feedforward neural network, SFLN) 相似，其结构如图1所示 [6]，包含输入层、隐含层和输出层。

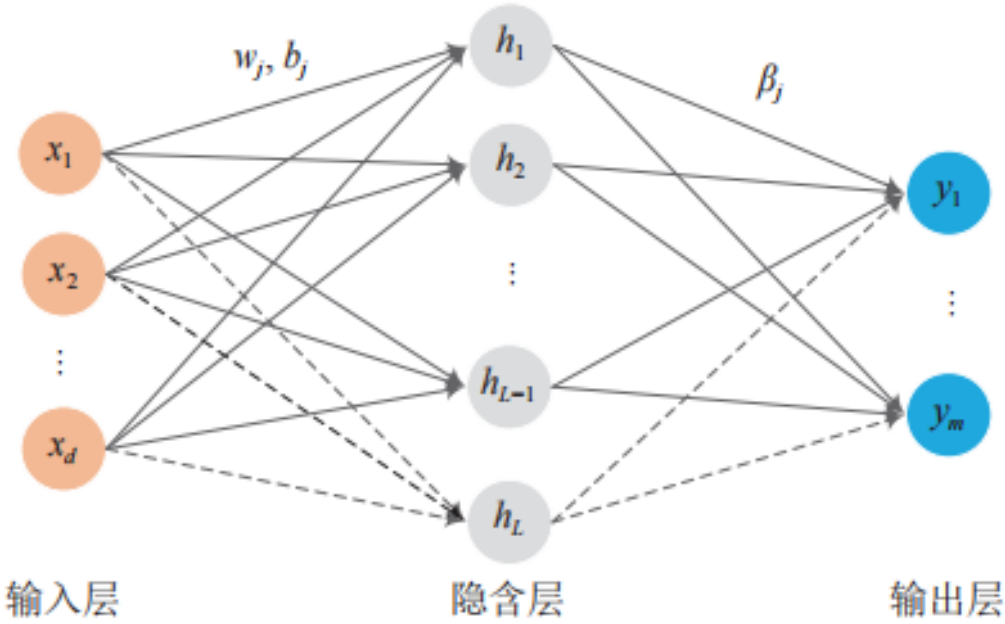

图 1. SCN 结构

2.2 SCN 隐含节点构建原

SCN 能利用监督机制，逐渐增加隐含层的节点个数，使残差达到期望，完成网络的构建。假设当前网络已经生成了 $L-1$ 个隐含节点，f 表示目标函数，m 是输出节点个数，则当前网络的残差可表示为:

$$e_{L-1} = f - f_{L-1} = [e_{L-1,1}, ..., e_{L-1,m}] \tag{1}$$

如果 $\|e_{L-1}\|$ 没有达到预期，则需要生成新的隐含节点。先随机生成权重和偏置，然后利用监督机制来确定最佳权重的偏置。监督机制表示如下:

$$\xi_{L,q} = \frac{\left(e_{L-1,q}^T(X) \cdot h_L(X)\right)}{h_L^T(X) \cdot h_L(X)} - (1 - r - \mu_L) e_{L-1,q}^T(X) e_{L-1,q}(X) \tag{2}$$

其中 h_L 表示新生成的隐含节点的输出矩阵，$q = 1, 2, ...m$，$r \in (0, 1)$，$\{\mu_L\}$ 表示非负实数列表。在所有满足 $\xi_{L,q} \geq 0$、随机生成的权重和偏置中，选择能最大化 $\xi_L = \sum_{q=1}^{m} \xi_{L,q}$ 的权重和偏置，即为新生成的隐含节点 L 的权重和偏置。

3 本文方法

3.1 本文方法概述

OSL-SCN，首先进行初始化，然后在在线阶段，进行训练和自学习。在数据块来到的时候，预测并计算残差，如果残差在可接受的范围内，则进行参数调整；如果超过了容忍度，则

进行网络结构调整。工作流程如图2所示 [3]。

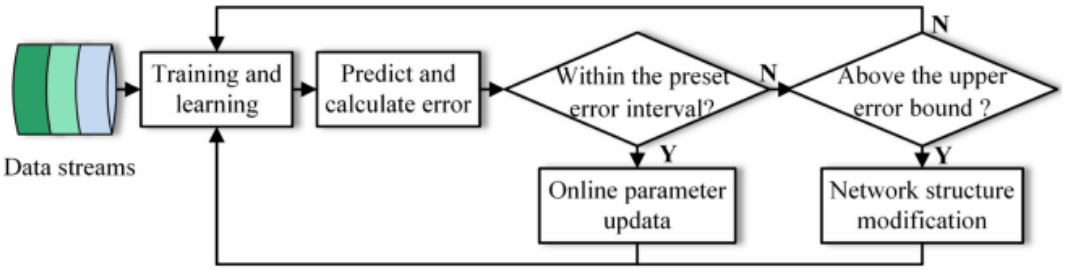

图 2. OSL-SCN 基本工程流程

3.2 在线参数更新机制

SCN 基础模型是假设所有的训练数据都能够一次性拿到，然而在数据流中，数据是一个个或者是一块块到来的，因此需要进行增量学习。当新的数据到来时，能够通过新的数据和标签来增强已经训练好的模型，动态地更新隐含节点的输出权重 β，

3.3 动态结构调整机制

当新样本到达时，计算各个隐含节点对于网络输出的贡献度，贡献度越小说明对当前数据预测越不准确。根据贡献度大小，从大到小排序，根据适应阈值 $\gamma(0 < \gamma < 1)$，将后几个贡献度小的结点进行剪枝，得到一个新的网络。之后再利用 SCN 的添加节点机制，根据当前窗口数据再生成新的节点，直到残差达到预期，完成结构的调整。

4 复现细节

4.1 与已有开源代码对比

OSL-SCN 的复现参考了基础模型 SCN 的开源代码(网络地址:https://deepscn.com/software.html)，在 SCN 开源代码的基础上，改造出 OSL-SCN。最后在 OSL-SCN 的基础上加入了类别不平衡处理机制，以适应数据流中出现的类别不平衡。

4.2 更改的代码

初始化阶段和 SCN 基础模型一样。新增一个在线学习的函数，如图3。

```python
def partial_fit(self, features, targets):
    (numSamples, numOutputs) = (targets.shape[0], targets.T.shape[0])
    assert features.shape[0] == targets.shape[0]
    targets = targets.reshape(-1, 1)
    H = self.activationFun(features)
    Ht = np.transpose(H)


    O = self.getOutput(features)
    E = targets - O
    Error = self.RMSE(E)
    print('==========')
    print(Error)
    if Error > 0.375:
        self.pruned(features, targets)
    else:
        try:
            self.M -= np.dot(self.M, np.dot(Ht, np.dot(pinv(np.eye(numSamples) +
                                                            np.dot(H, np.dot(self.M, Ht))), np.dot(H, self.M))))
            self.Beta += np.dot(self.M, np.dot(Ht, targets - np.dot(H, self.Beta)))
        except np.linalg.linalg.LinAlgError:
            print("SVD not converge, ignore the current training cycle")
```

图 3. 在线学习逻辑

函数逻辑是，首先对数据的合法性进行检查，然后预处理数据，计算当前残差。通过当前残差和预期残差对比，选择参数调整或者结构调整。其中,pruned() 函数是结构调整的核心函数，其实现如图4。

```
def pruned(self, X, T):
    S_list = []
    S_sum = 0
    for i in range(0, self.L):
        w_i = self.W[:, i]
        b_i = self.b[:, i]
        beta_i = self.Beta[i, :]
        t_sum = 0
        for j in range(len(T)):
            H = expit(X[j] @ w_i + b_i)
            t_sum += H @ beta_i
        if t_sum < 0:
            t_sum = -t_sum
        S_list.append([t_sum, i])
        S_sum += t_sum
    S_list.sort(key=lambda ele: ele[0], reverse=True)
    J_list = []
    preSum = 0
    for i in range(0, self.L):
        preSum += S_list[i][0]
        J_i = preSum / S_sum
        J_list.append(J_i)
    print('#剪枝前的L:{}\t'.format(self.L))
    l = self.L
    for i in range(0, self.L):
        if J_list[i] >= self.gamma:
            l = i
            break
    tL = self.L
    dlist = []
    for i in range(l, tL):
        index = S_list[i][1]
        dlist.append(index)
    self.deleteNode(dlist)
```

图 4. 结构调整函数

函数逻辑是，首先计算每个隐含节点的贡献，然后排序后累加，当第一次超过 γ 值的时候，将后面的节点都进行剪枝。在剪枝后，需要添加新的节点，添加新的节点如图5。其实现

逻辑和 SCN 基础模型一致。

```
#添加新节点
while (self.L < self.L_max) and (Error > self.tolChange):
    if self.L % self.verbose == 0:
        print('#L:{}\tRMSE:{:.4f} \tACC:{:.4f}\r'.format(
            self.L, Error, rate))
    # Search for candidate node / Hidden Parameters
    (w_L, b_L, Flag) = self.sc_Search(X, E)
    if Flag == 1:
        # could not find enough node
        break
    self.addNodes(w_L, b_L)
    # Calculate Beta/ Update all
    (otemp, E, Error) = self.upgradeSCN(X, T)
    O = self.getLabel(X)
    rate = metrics.accuracy_score(T, O)
print('添加节点后#L:{}\tRMSE:{:.4f} \tACC:{:.4f}\r'.format(self.L, Error, rate))
```

图 5. 添加新结点

4.3 创新点

在真实数据中，类别不平衡问题和概念漂移问题往往同时存在，少数类错误分类的代价往往更高。因此处理类别不平衡问题具有现实意义。为了方便讨论，这里以二分类为前提。参照 WOS-ELM [4] 的权重添加方法，给 OSL-SCN 添加类别权重矩阵，改进后的模型这里简称为 WOSL-SCN（Weighted OSL-SCN）。在初始化和在线学习阶段都计算类别权重矩阵，在参数更新和计算 β 时，加入权重矩阵，以缓解出现类别不平衡时，学习器偏好多数类的问题。

计算当前数据窗口的类别权重矩阵，处理函数如图6。

```
self.count_1 = sum(targets)
self.count_0 = len(targets) - self.count_1
self.J = np.eye(len(targets))
for i in range(len(targets)):
    if targets[i] == 0:
        self.J[i][i] = self.count_1 / self.count_0
    else:
        self.J[i][i] = 1
```

图 6. 计算类别权重矩阵

这里假设多数类为正类，将正类的权重恒置为 1。而少数类的权重置为：多数类样本数/少数类样本数。同时，参数更新和 β 的计算也要加入权重机制，如图7。

```
try:
    self.M -= np.dot(self.M, np.dot(Ht, np.dot(pinv(pinv(J) +
                                        np.dot(H, np.dot(self.M, Ht))), np.dot(H, self.M))))
    self.Beta += np.dot(self.M, np.dot(np.dot(Ht, J), targets - np.dot(H, self.Beta)))
except np.linalg.linalg.LinAlgError:
```

图 7. 参数更新加入类别权重矩阵

5 实验结果分析

本部分为实验部分，根据 OSL-SCN 和 WOSL-SCN 在具有类别不平衡的数据流的处理能力，设计了系列消融实验。首先介绍了实验所用的数据集，然后介绍了实验所用的评价指标，最后给出实验结果并分析。

5.1 数据集介绍

本次实验采用的数据集是 Sine_CI_IR、Sea_g_CI_IR、Sea_s_CI_IR，其中 IR 表示不平衡率，本次实验取 1、2、4、9，即多数类和少数类的比例为 1、2、4、9。注意，当 IR 等于 1 时，代表数据中没有出现类不平衡的现象。其中 Sine 数据集在 5000、10000 和 15000 三个位置发生突变型反转，Sea_s_CI_IR 则在 5000 和 15000 的位置发生突变型反转，Sea_g_CI_IR 则是发生渐变型概念漂移。

5.2 评论指标

本次实验采用两个指标，一是分类准确率 ACC（Accuracy，记为 ACC），即正确分类的样本数占总样本数的比率；二是几何均值 Gmean [2]，当出现类别不平衡时，Gmean 值可以反映分类器的整体性能，能关注到少数类的分类准确率问题。

5.3 实验结果与分析

Sine_CI_1 实验结果如图8。

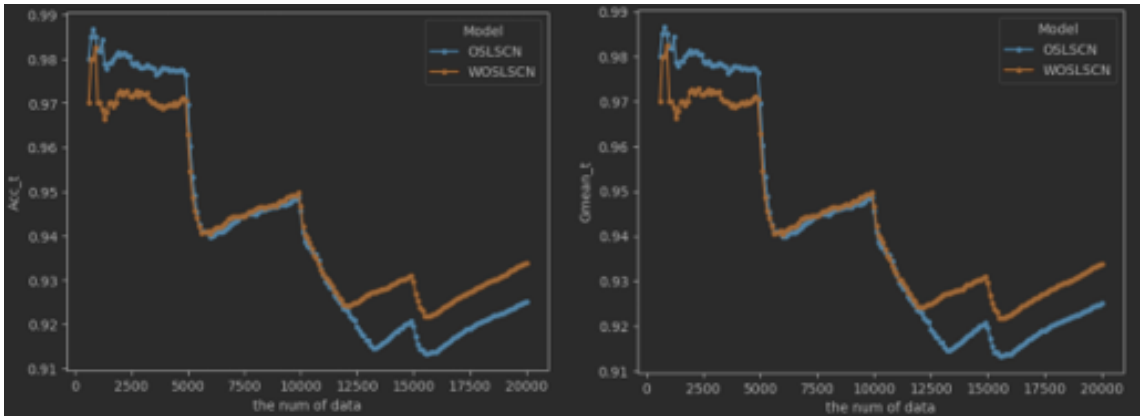

图 8. Sine_CI_1 实验结果

Sine_CI_2 实验结果如图9。

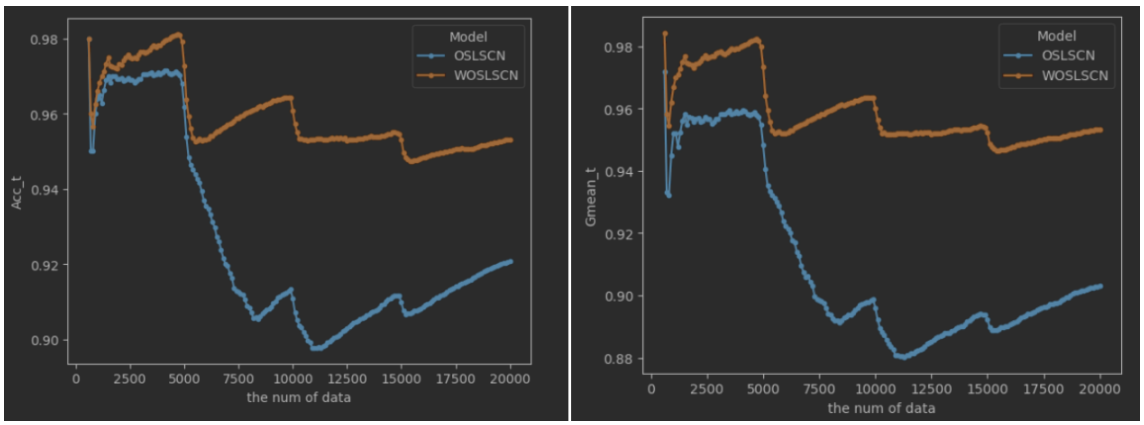

图 9. Sine_CI_2 实验结果

Sine_CI_4 实验结果如图10。

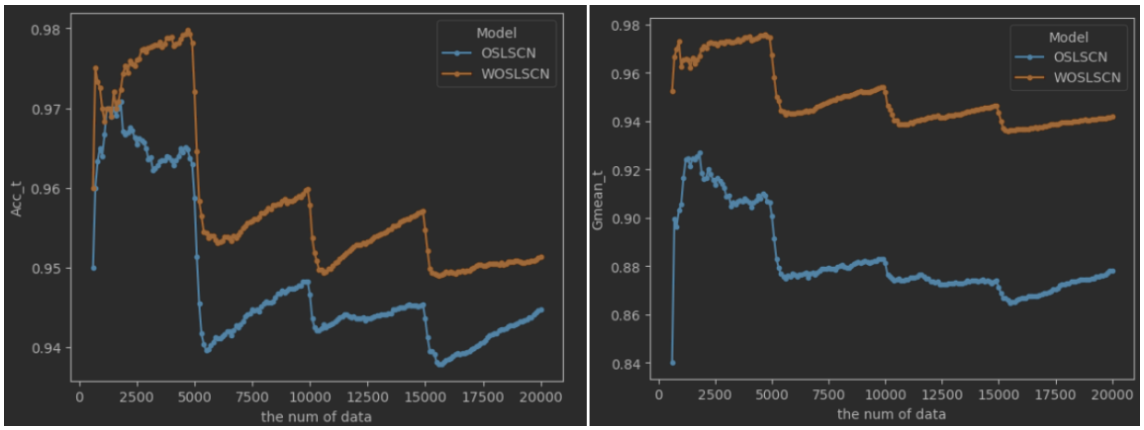

图 10. Sine_CI_4 实验结果

Sine_CI_9 实验结果如图11。

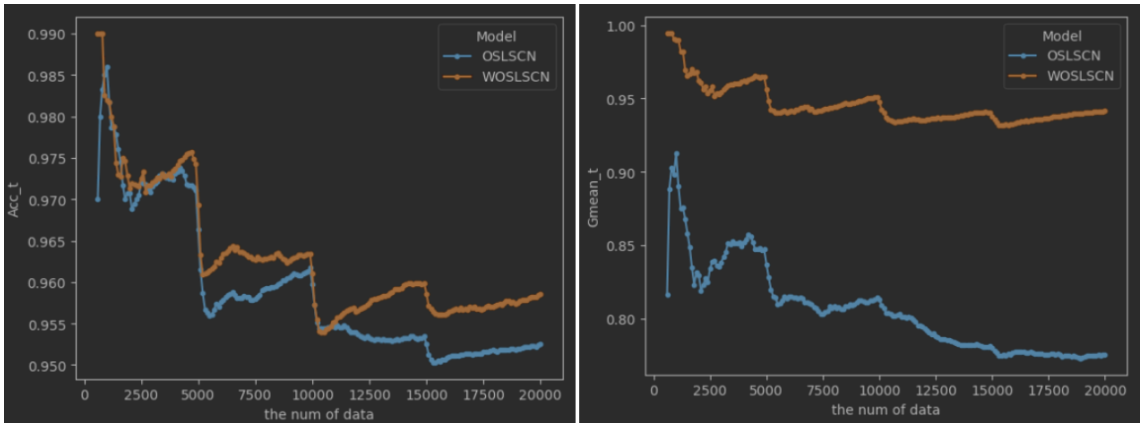

图 11. Sine_CI_9 实验结果

Sea_g_CI_1 实验结果如图12。

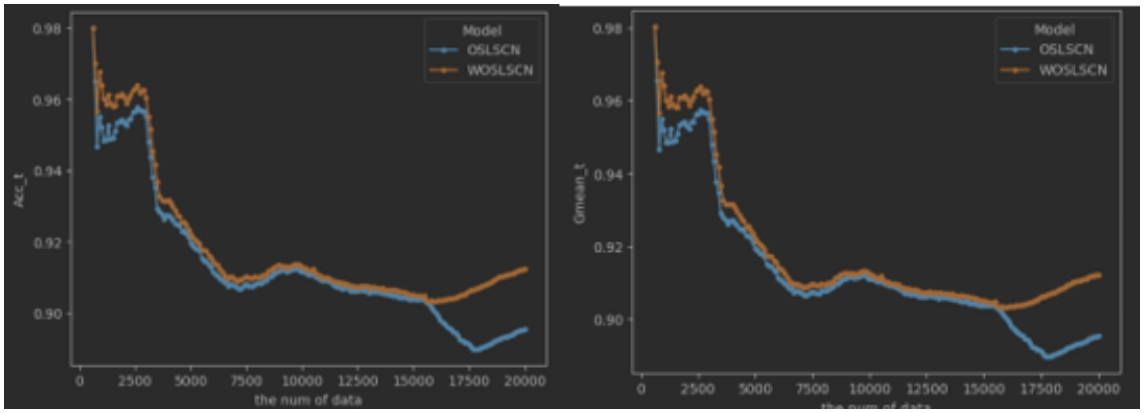

图 12. Sea_g_CI_1 实验结果

Sea_g_CI_2 实验结果如图13。

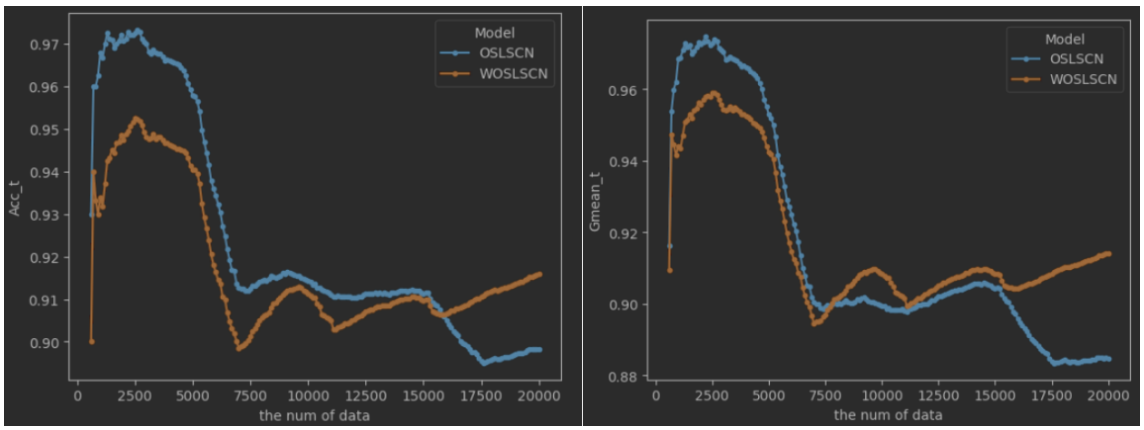

图 13. Sea_g_CI_2 实验结果

Sea_g_CI_4 实验结果如图14。

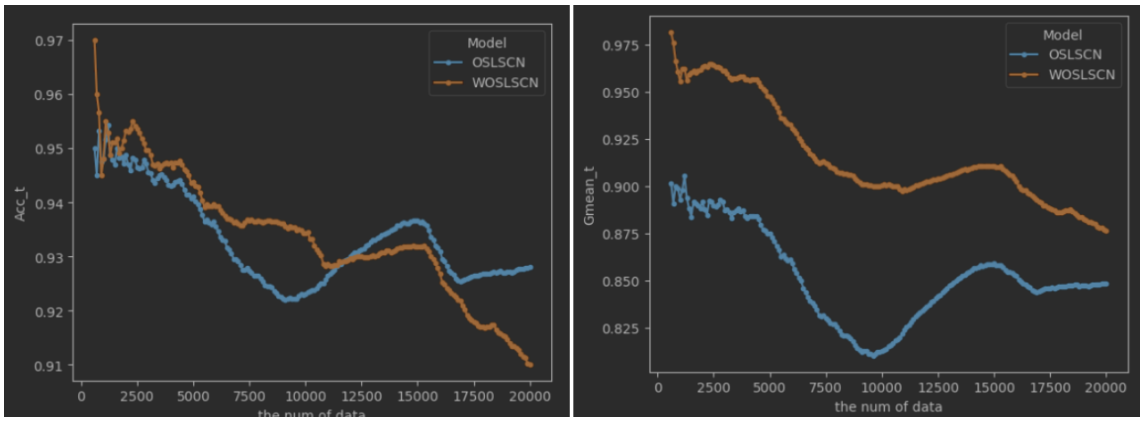

图 14. Sea_g_CI_4 实验结果

Sea_g_CI_9 实验结果如图15。

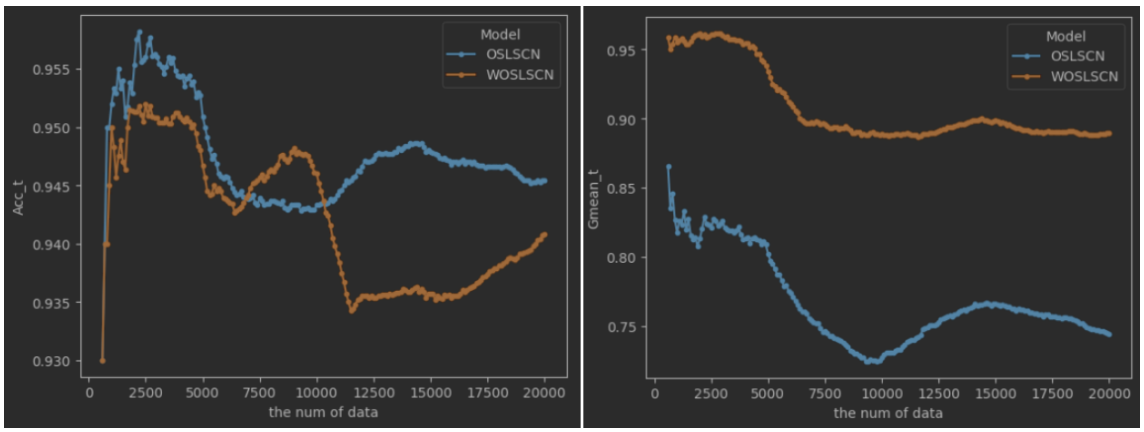

图 15. Sea_g_CI_9 实验结果

Sea_s_CI_1 实验结果如图16。

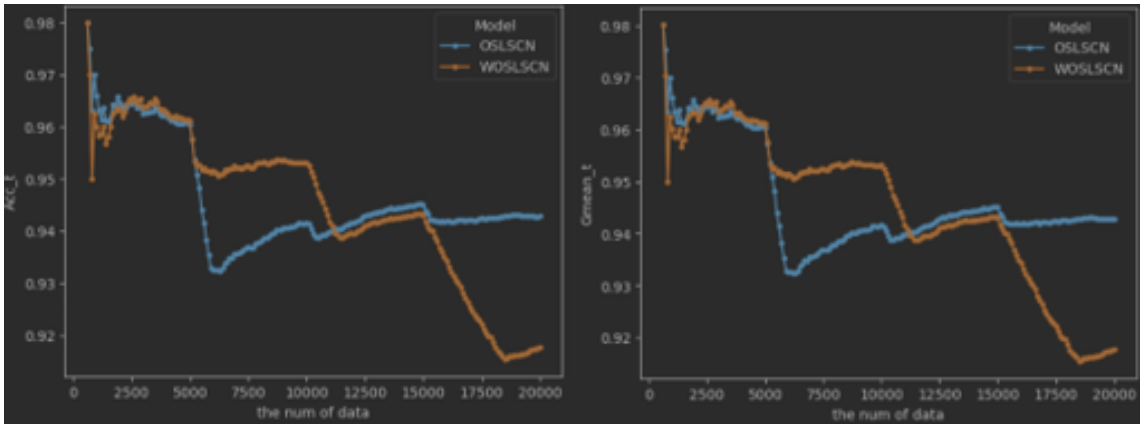

图 16. Sea_s_CI_1 实验结果

Sea_s_CI_2 实验结果如图17。

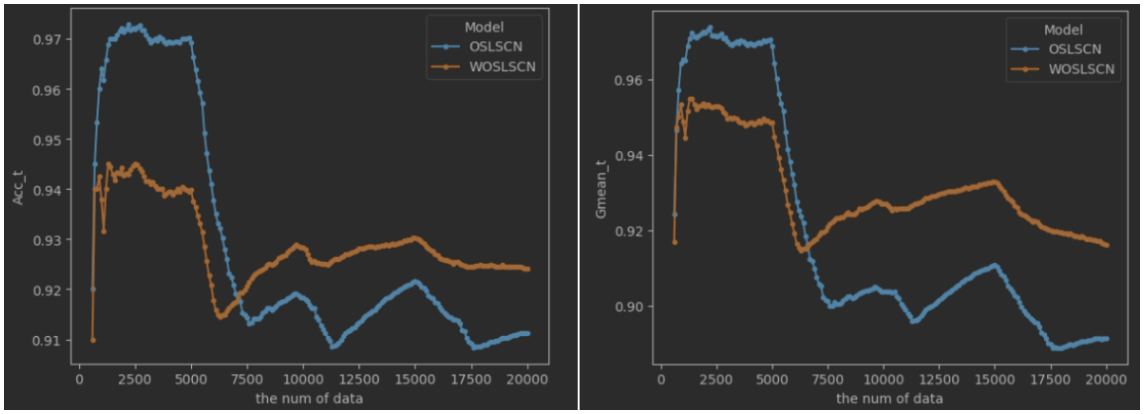

图 17. Sea_s_CI_2 实验结果

Sea_s_CI_4 实验结果如图18。

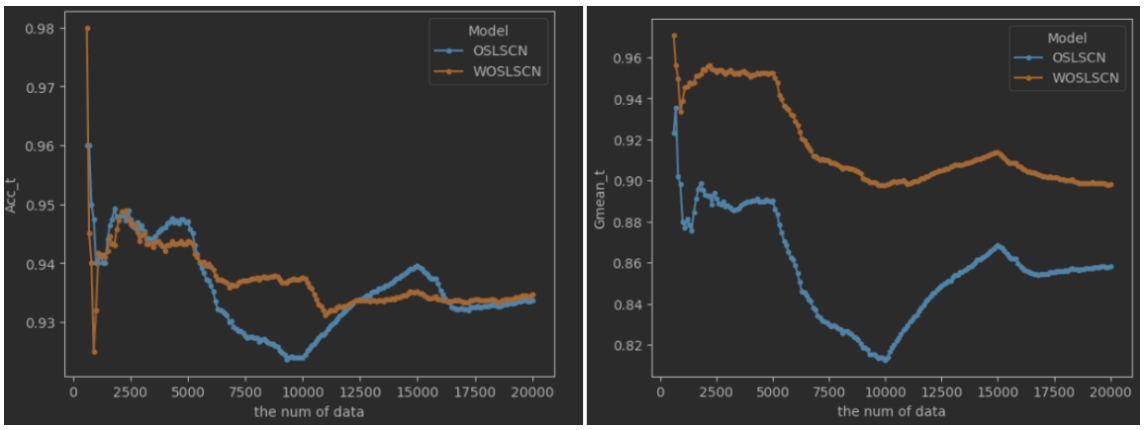

图 18. Sea_s_CI_4 实验结果

Sea_s_CI_9 实验结果如图19。

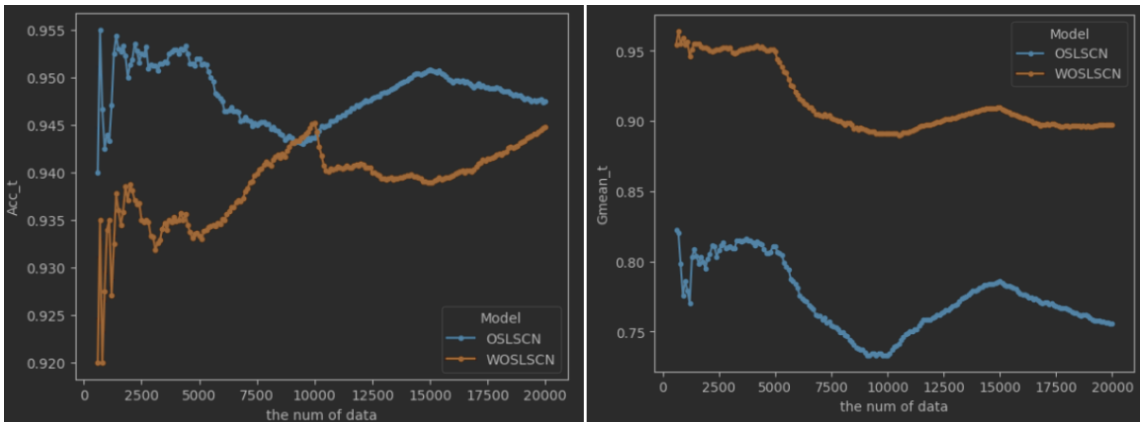

图 19. Sea_s_CI_9 实验结果

从实验结果中可以看出，在没有出现类不平衡时，两者性能差别不大。

但在出现类不平衡时，绝大多数情况下，WOSL-SCN 的 Gmean 值表现都比 OSL-SCN 好；同一数据集下，不平衡率越大，WOSL-SCN 的 Gmean 值表现越好；由于 WOSL-SCN 对多数类的偏好减轻了，因此会出现综合准确率下降的情况，但是也有 90% 以上的正确率，在可接受的范围内。因此可以得出结论，在能保持较高综合准确率的前提下，WOSL-SCN 可以更好地适应类别不平衡问题。

通过消融实验，可以得出结论，本次的工作能让原文的模型能够适应同时具有概念漂移和类不平衡的更复杂数据流分类场景。

6 总结与展望

本文介绍了 SCN 的基础模型，解释了其生成隐含节点的构建原理，并进一步介绍了 OSL-SCN 的工作原理。通过在线参数更新机制和动态结构调整机制，使得 OSL-SCN 在数据流上有效较好的分类性能和较快的收敛速度。在 OSL-SCN 的基础上，本文添加了类别权重矩阵，使得 OSL-SCN 具有处理类别不平衡的能力。最后通过不同种类、不同不平衡率的数据集，进行多次的实验，结果表示，本次的工作能让原文的模型能够适应同时具有概念漂移和类不平

衡的更复杂数据流。但是由于对多数类的偏好减轻了，WOSL-SCN 出现了综合准确率下降的情况，未来也可以着手于提升综合准确率。

除了类别不平衡问题，数据流中还可能出现概念进化、概念偏差、多维数据流等问题，这些问题都会让 OSL-SCN 的性能下降。如何在更复杂数据流中保持较高的学习性能，是未来值得研究的方向。

参考文献

[1] Supriya Agrahari and Anil Kumar Singh. Concept drift detection in data stream mining: A literature review. *Journal of King Saud University-Computer and Information Sciences*, 34(10):9523–9540, 2022.

[2] Miroslav Kubat, Robert Holte, and Stan Matwin. Learning when negative examples abound. In *Machine Learning: ECML-97: 9th European Conference on Machine Learning Prague, Czech Republic, April 23–25, 1997 Proceedings 9*, pages 146–153. Springer, 1997.

[3] Kang Li, Junfei Qiao, and Dianhui Wang. Online self-learning stochastic configuration networks for nonstationary data stream analysis. *IEEE Transactions on Industrial Informatics*, 2023.

[4] Bilal Mirza, Zhiping Lin, and Kar-Ann Toh. Weighted online sequential extreme learning machine for class imbalance learning. *Neural processing letters*, 38:465–486, 2013.

[5] Dianhui Wang and Ming Li. Stochastic configuration networks: Fundamentals and algorithms. *IEEE Transactions on Cybernetics*, 47(10):3466–3479, 2017.

[6] 张成龙, 丁世飞, 郭丽丽, and 张健. 随机配置网络研究进展. 软件学报, pages 1–21, 2022.