中国科学: 数学 2015 年 第 45 卷 第 9 期: 1487 ~ 1504
www.scichina.com math.scichina.com
《中国科学》杂志社
SCIENCE CHINA PRESS

综 述

用于神经网络权值稀疏化的 $L_{1/2}$ 正则化方法

献给徐利治教授 95 华诞

吴微*, 杨洁

大连理工大学数学科学学院, 大连 116024
E-mail: wuweiw@dlut.edu.cn, yangjiee@dlut.edu.cn

收稿日期: 2015-02-27; 接受日期: 2015-05-06; * 通信作者
国家自然科学基金 (批准号: 11201051) 资助项目

摘要 在保证适当学习精度前提下, 神经网络的神经元个数应该尽可能少 (结构稀疏化), 从而降低成本, 提高稳健性和推广精度. 本文采用正则化方法研究前馈神经网络的结构稀疏化. 除了传统的用于稀疏化的 L_1 正则化之外, 本文主要采用近几年流行的 $L_{1/2}$ 正则化. 为了解决 $L_{1/2}$ 正则化算子不光滑、容易导致迭代过程振荡这一问题, 本文试图在不光滑点的一个小邻域内采用磨光技巧, 构造一种光滑化 $L_{1/2}$ 正则化算子, 希望达到比 L_1 正则化更高的稀疏化效率. 本文综述了近年来作者在用于神经网络稀疏化的 $L_{1/2}$ 正则化的一些工作, 涉及的神经网络包括 BP 前馈神经网络、高阶神经网络、双并行前馈神经网络, 以及 Takagi-Sugeno 模糊模型.

关键词 神经网络 稀疏化 $L_{1/2}$ 正则化

MSC (2010) 主题分类 68T05

1 引言

人工神经网络研究包含两个主要部分, 其一是权值优化, 即针对给定的某种网络结构, 选取适当的学习方法寻求最优权值, 使得训练误差 (训练样本集上的误差) 和推广误差 (未用于训练的样本的误差) 都足够小 (参见文献 [1,2]); 其二是结构优化, 即选择适当的活化函数、网络层数、单元连接方式、单元数量等 (参见文献 [3–5]). 人工神经网络研究已经得到近几十年的迅猛发展. 相比较而言, 神经网络结构优化研究远不如权值优化研究那样丰富和成熟.

从神经网络精度的角度来看, 结构稀疏化的研究意义在于, 单元数量太少当然无法达到必要的精度, 但是, 单元数量太多则除了增加成本之外, 还容易导致过度训练, 即训练误差很小, 而推广误差很大并且稳健性变坏. 事实上, 神经网络中的单元与权值连接数量很像用多项式作数值逼近时的多项式次数. 达到一定限度之后, 单元与权值连接数量越多 (或多项式次数越高) 则训练精度越高, 但推广精度越低. 主要原因在于, 单元与权值连接数量越多 (或多项式次数越高), 越容易产生振荡 (注意多项式次数决定了其 “拐弯” 的次数), 此即逼近理论中著名的所谓龙格现象. 实际应用中, 神经网络常常要制成相应的电路或光路等硬件. 从这一方面看, 结构稀疏化意味着降低制造成本, 成为神经网络成功应用的重要一环.

英文引用格式: Wu W, Yang J. $L_{1/2}$ regularization methods for weights sparsification of neural networks (in Chinese). Sci Sin Math, 2015, 45: 1487–1504, doi: 10.1360/N012015-00110

神经网络权值优化的研究成果已经相当丰富和成熟. 传统的并且最简单的权值学习算法是梯度法 (BP 算法) [6,7], 以及加上动量项、惩罚项等许多不同机制的各种修正算法 (参见文献 [8,9]). 近些年流行的极端学习机 (ELM) 算法, 是神经网络学习算法研究的重大进展 (参见文献 [10]). ELM 基本解决了困扰多年的 BP 算法收敛慢这一大难题. 为了保证 ELM 的收敛性, 理论上要求神经网络的隐单元个数必须足够多. 但是为了保证推广精度, 隐单元个数又应该尽可能少. 因此很显然, 结构稀疏化又成为 ELM 学习算法得以成功应用的一大关键 (参见文献 [11–13]).

正则化方法也称为 Bayes 正则化[14,15]. 正则项能够防止权值过大以及过度训练, 从而起到改善推广误差的效果; 但是并不能驱使某些权值趋于零, 从而不能起到权值或单元稀疏化的效果. 受压缩感知理论中 Lasso 算法的影响, 人们将正则化用于神经网络权值和单元稀疏化[16,17], 起到不错的稀疏化效果. 最近一两年, 我国学者徐宗本院士所倡导的与 L_1 正则化方法相比有其独特优点的 $L_{1/2}$ 正则化方法[18] 也被应用到神经网络权值和单元稀疏化[19]. 我们进而提出一种光滑化 $L_{1/2}$ 正则化方法, 希望进一步提高正则化效率. 初步研究成果表明, 光滑化 $L_{1/2}$ 正则化方法有效消除了普通 $L_{1/2}$ 正则化方法学习过程中的振荡现象.

本文综述了近年来我们在用于神经网络稀疏化的 $L_{1/2}$ 正则化的部分工作, 涉及的神经网络包括 BP 前馈神经网络、高阶神经网络和双并行前馈神经网络, 以及 Takagi-Sugeno 模糊模型.

2 BP 前馈神经网络

在过去的二十年间, 前馈神经网络的理论与应用研究一直占据神经网络研究领域的中心地位. 主要研究内容为这类网络的性质与学习方法, 其中 BP 方法[20] 是最为流行的一种方法. 基于误差纠正的学习规则, BP 方法能够作为一种一般化的最小均方差 (LMS) 学习方法. 以均方差作为性能指标, BP 方法是一种近似的最速下降方法.

梯度法的应用主要有两种途径, 一个是分批处理的梯度学习方法, 即所有的样本都被输入到网络之后更新权值, 通常被称为离线梯度法; 另外一个是在线梯度学习, 即一个样本输入网络后马上更新权值[21].

2.1 利用 $L_{1/2}$ 正则化方法的离线梯度法收敛性[22]

考虑一个含有 p 个输入节点、q 个隐节点和 1 个输出节点的三层网络. 记 $w_0 = (w_{10}, w_{20}, \ldots, w_{q0})^{\mathrm{T}} \in \mathbb{R}^q$ 为隐藏单元与输出单元之间的权向量, $w_i = (w_{i1}, w_{i2}, \ldots, w_{ip})^{\mathrm{T}} \in \mathbb{R}^p$ 为输入单元与隐单元 i $(i = 1, 2, \ldots, q)$ 之间的权向量. 为了描述方便, 我们把所有的权参数写成一个简写形式

$$W = (w_0^{\mathrm{T}}, w_1^{\mathrm{T}}, \ldots, w_q^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{q+pq},$$

另外再定义一个矩阵 $V = (w_1, w_2, \ldots, w_q)^{\mathrm{T}} \in \mathbb{R}^{q\times p}$. 记 $g : \mathbb{R} \to \mathbb{R}$ 是一个给定的从隐节点到输出节点的传递函数, 特别地, 但不是必要的, 可以是一个 Sigmoid 函数. 对向量 $\boldsymbol{x} = (x_1, x_2, \ldots, x_q)^{\mathrm{T}} \in \mathbb{R}^q$, 我们再定义一个向量函数 $G : \mathbb{R}^q \to \mathbb{R}^q$, $G(\boldsymbol{x}) = (g(x_1), g(x_2), \ldots, g(x_q))^{\mathrm{T}}$. 假设 $\{\xi^j, O^j\}_{j=1}^J \subset \mathbb{R}^p \times \mathbb{R}$ 是一个给定的训练样本集. 具有 $L_{1/2}$ 正则惩罚项的误差函数如下:

$$E(W) = \frac{1}{2}\sum_{j=1}^{J}(O^j - g(w_0 \cdot G(V\xi^j)))^2 + \lambda\sum_{i=1}^{q}\sum_{k=0}^{p}|w_{ik}|^{\frac{1}{2}}$$

$$= \sum_{j=1}^{J} g_j(w_0 \cdot G(V\xi^j)) + \lambda \sum_{i=1}^{q}\sum_{k=0}^{p} |w_{ik}|^{\frac{1}{2}}, \tag{2.1}$$

其中 $|\cdot|$ 表示绝对值,

$$g_j(t) := \frac{1}{2}(O^j - g(t))^2.$$

误差函数的梯度为

$$\begin{aligned} E_W(W) = (&E_{w_{10}}(W), E_{w_{20}}(W), \ldots, E_{w_{q0}}(W),\\ &E_{w_{11}}(W), E_{w_{12}}(W), \ldots, E_{w_{1p}}(W),\\ &E_{w_{21}}(W), E_{w_{22}}(W), \ldots, E_{w_{2p}}(W),\\ &\ldots, E_{w_{q1}}(W), E_{w_{q2}}(W), \ldots, E_{w_{qp}}(W))^{\mathrm{T}}, \end{aligned} \tag{2.2}$$

其中

$$E_{w_{i0}}(W) = \sum_{j=1}^{J} g_j'(w_0 \cdot G(V\xi^j)) g(w_i\xi^j) + \frac{\lambda \mathrm{sgn}(w_{i0})}{2|w_{i0}|^{\frac{1}{2}}}, \tag{2.3}$$

$$E_{w_{ik}}(W) = \sum_{j=1}^{J} g_j'(w_0 \cdot G(V\xi^j)) w_{i0} g'(w_i \cdot \xi^j)\xi_k^j + \frac{\lambda \mathrm{sgn}(w_{ik})}{2|w_{ik}|^{\frac{1}{2}}}, \tag{2.4}$$

这里 $i = 1, 2, \ldots, q$, $k = 1, 2, \ldots, p$.

从一个任意的初始值 W^0 开始, 权 $\{W^n\}$ 通过下式迭代更新,

$$W^{n+1} = W^n + \Delta W^n, \quad n = 0, 1, 2, \ldots, \tag{2.5}$$

其中

$$\Delta w_{i0}^n = -\eta\left[\sum_{j=1}^{J} g_j'(w_0^n \cdot G(V^n\xi^j)) g(w_i\xi^j) + \frac{\lambda \mathrm{sgn}(w_{i0}^n)}{2|w_{i0}^n|^{\frac{1}{2}}}\right], \tag{2.6}$$

$$\Delta w_{ik}^n = -\eta\left[\sum_{j=1}^{J} g_j'(w_0^n \cdot G(V^n\xi^j)) w_{i0}^n g'(w_i^n \cdot \xi^j)\xi_k^j + \frac{\lambda \mathrm{sgn}(w_{ik}^n)}{2|w_{ik}^n|^{\frac{1}{2}}}\right], \tag{2.7}$$

这里 $i = 1, 2, \ldots, q$, $k = 1, 2, \ldots, p$, 学习率 $\eta > 0$ 是一个常数.

前面所述的一般 $L_{1/2}$ 正则项在原点处是非光滑的, 这就导致很难分析收敛性, 而且更重要的是导致数值计算中出现振荡. 为了克服这一缺陷, 通过在原点处光滑化一般的正则项得到了一个改进的 $L_{1/2}$ 正则项, 进而导出以下具有光滑 $L_{1/2}$ 正则惩罚项的误差函数:

$$\begin{aligned} E(W) &= \frac{1}{2}\sum_{j=1}^{J}(O^j - g(w_0 \cdot G(V\xi^j)))^2 + \lambda \sum_{i=1}^{q}\sum_{k=0}^{p} f(w_{ik})^{\frac{1}{2}}\\ &= \sum_{j=1}^{J} g_j(w_0 \cdot G(V\xi^j)) + \lambda \sum_{i=1}^{q}\sum_{k=0}^{p} f(w_{ik})^{\frac{1}{2}}, \end{aligned} \tag{2.8}$$

其中 $g_j(t) := \frac{1}{2}(O^j - g(t))^2$, $|\cdot|$ 表示绝对值, $f(x)$ 是一个近似 $|x|$ 的光滑函数. 为了书写简化, 我们选取 $f(x)$ 为一个分段多项式函数:

$$f(x) = \begin{cases} -x, & x \leqslant -a, \\ -\dfrac{1}{8a^3}x^4 + \dfrac{3}{4a}x^2 + \dfrac{3}{8}a, & -a < x < a, \\ x, & x \geqslant a, \end{cases} \tag{2.9}$$

其中 a 是一个较小的正常数. 不难得出

$$f(x) \in \left[\frac{3}{8}a, +\infty\right), \quad f'(x) \in [-1, 1], \quad f''(x) \in \left[0, \frac{3}{2a}\right].$$

误差函数的梯度可以写为 (2.2), 其中

$$E_{w_{i0}}(W) = \sum_{j=1}^{J} g_j'(w_0 \cdot G(V\xi^j))g(w_i \cdot \xi^j) + \lambda \frac{f'(w_{i0})}{2f(w_{i0})^{\frac{1}{2}}}, \tag{2.10}$$

$$E_{w_{ik}}(W) = \sum_{j=1}^{J} g_j'(w_0 \cdot G(V\xi^j))w_{i0}g'(w_i \cdot \xi^j)\xi_k^j + \lambda \frac{f'(w_{ik})}{2f(w_{ik})^{\frac{1}{2}}}, \tag{2.11}$$

这里 $i = 1, 2, \ldots, q$, $k = 1, 2, \ldots, p$.

具有光滑 $L_{1/2}$ 正则惩罚项的梯度法 (BGMSR) 从一个初始值 W^0 出发, 权 $\{W^n\}$ 通过下式迭代法更新:

$$W^{n+1} = W^n + \Delta W^n, \quad n = 0, 1, 2, \ldots, \tag{2.12}$$

其中

$$\Delta w_{i0}^n = -\eta\left[\sum_{j=1}^{J} g_j'(w_0^n \cdot G(V^n\xi^j))g(w_i^n \cdot \xi^j) + \lambda \frac{f'(w_{i0}^n)}{2f(w_{i0}^n)^{\frac{1}{2}}}\right], \tag{2.13}$$

$$\Delta w_{ik}^n = -\eta\left[\sum_{j=1}^{J} g_j'(w_0^n \cdot G(V^n\xi^j))w_{i0}^n g'(w_i^n \cdot \xi^j)\xi_k^j + \lambda \frac{f'(w_{ik}^n)}{2f(w_{ik}^n)^{\frac{1}{2}}}\right], \tag{2.14}$$

这里 $i = 1, 2, \ldots, q$, $k = 1, 2, \ldots, p$, 学习率 $\eta > 0$ 是一个常数.

下面给出一些关于带光滑 $L_{1/2}$ 正则项 (2.12) 的离线梯度法的收敛性定理. 这些收敛定理的一些充分条件如下:

(A1) 对于 $t \in \mathbb{R}$, $|g(t)|$, $|g'(t)|$, $|g''(t)|$ 一致有界;

(A2) $\|w_0^n\|$ $(n = 0, 1, 2, \ldots)$ 一致有界;

(A3) η 与 λ 满足 $0 < \eta < \frac{1}{M\lambda + C_1}$, 其中 $M = \frac{\sqrt{6}}{4\sqrt{a^3}}$ 与 C_1 是 (3.20) 中定义的常数;

(A4) 存在一个紧集 Φ 使得 $W^n \in \Phi$, 且集合 $\Phi_0 = \{W \in \Phi : E_W(W) = 0\}$ 包含有限个点.

以下是后文用到的一些特别的常数:

$$\begin{aligned}
&C_1 = J(1+C_2)C_3\max\{C_2^2, C_5^2\} + \frac{1}{2}J(1+C_2)C_3 + \frac{1}{2}JC_3^2C_4^2C_5,\\
&C_2 = \max\{\sqrt{q}C_3, (C_3C_4)^2\},\\
&C_3 = \max\Big\{\sup_{t\in\mathbb{R}}|g(t)|, \sup_{t\in\mathbb{R}}|g'(t)|, \sup_{t\in\mathbb{R}}|g''(t)|, \sup_{t\in\mathbb{R},1\leqslant j\leqslant J}|g_j'(t)|, \sup_{t\in\mathbb{R},1\leqslant j\leqslant J}|g_j''(t)|\Big\},\\
&C_4 = \max_{1\leqslant j\leqslant J}\|\xi^j\|,\\
&C_5 = \sup_{n\in N}\|w_0^n\|.
\end{aligned} \tag{2.15}$$

定理 1 令误差函数如 (2.8) 所定义. 对任意一个初始值, 权序列 $\{W^n\}$ 由迭代法 (2.12) 生成. 若假设 (A1)–(A3) 成立, 则

(i) $E(W^{n+1}) \leqslant E(W^n)$, $n = 0, 1, 2, \ldots$;

(ii) 存在一个 $E^* \geqslant 0$ 使得 $\lim_{k\to\infty} E(W^n) = E^*$;

(iii) $\lim_{n\to\infty}\|E_W(W^n)\| = 0$.

更进一步, 若假设 (A4) 也成立, 则有如下强收敛:

(iv) 存在一个点 $W^* \in \Phi_0$ 使得 $\lim_{n\to\infty}(W^n) = W^*$.

2.2 利用 $L_{1/2}$ 正则化方法的在线梯度法收敛性[23]

与离线梯度学习方法相比, 在 BP 训练中, 在线梯度法 (OGM) 更为流行[24]. 从存储空间与计算时间角度分析, 特别是当学习目标不稳定, 只能利用样本序列中最新的训练样本来计算网络参数时, OGM 更为有效. 这个过程本身就具有随机性, 因为每次训练误差计算出来之后都要随机地选取一个新的训练样本. 这一点与离线梯度法形成了鲜明的对比. 而离线梯度法是所有的样本同时决定训练误差, 实际上是一种确定性算法 (参见文献 [25]).

给定初始权 $W^0 \in \mathbb{R}^{q+pq}$, 通常的 $L_{1/2}$ 在线梯度学习方法 (OGL1/2) 利用下式反复更新权 $\{W^n\}$,

$$W^{nJ+j} = W^{nJ+j-1} - \eta_n\Delta_j^n W^{nJ+j-1}, \tag{2.16}$$

其中 $n = 0, 1, 2, \ldots$, $j = 1, 2, \ldots, J$,

$$\Delta_j^n w_{i0}^{nJ+j-1} = g_j'(w_0^{nJ+j-1}\cdot G(V^{nJ+j-1}\xi^j))g(w_i^{nJ+j-1}\xi^j) + \frac{\lambda\mathrm{sgn}(w_{i0}^{nJ+j-1})}{2J|w_{i0}^{nJ+j-1}|^{\frac{1}{2}}}, \tag{2.17}$$

$$\Delta_j^n w_{ik}^{nJ+j-1} = g_j'(w_0^{nJ+j-1}\cdot G(V^{nJ+j-1}\xi^j))w_{i0}^{nJ+j-1}\times g'(w_i^{nJ+j-1}\cdot\xi^j)\xi_k^j + \frac{\lambda\mathrm{sgn}(w_{ik}^{nJ+j-1})}{2J|w_{ik}^{nJ+j-1}|^{\frac{1}{2}}}, \tag{2.18}$$

这里 $i = 1, 2, \ldots, q$, $k = 1, 2, \ldots, p$, η_n 是训练第 n 步的学习率.

与前面类似, 引入具有光滑 $L_{1/2}$ 正则惩罚项的误差函数:

$$\begin{aligned}
E(W) &= \frac{1}{2}\sum_{j=1}^{J}(O^j - g(w_0\cdot G(V\xi^j)))^2 + \lambda\sum_{i=1}^{q}\sum_{k=0}^{p}f(w_{ik})^{\frac{1}{2}}\\
&= \sum_{j=1}^{J}g_j(w_0\cdot G(V\xi^j)) + \lambda\sum_{i=1}^{q}\sum_{k=0}^{p}f(w_{ik})^{\frac{1}{2}}.
\end{aligned} \tag{2.19}$$

给定初始权 $W^0 \in \mathbb{R}^{q+pq}$, 按照以下式子更新权值 $\{W^n\}$,

$$W^{nJ+j} = W^{nJ+j-1} - \eta_n \Delta_j^n W^{nJ+j-1}, \tag{2.20}$$

其中 $n = 0, 1, 2, \ldots$, $j = 1, 2, \ldots, J$,

$$\Delta_j^n w_{i0}^{nJ+j-1} = g_j'(w_0^{nJ+j-1} \cdot G(V^{nJ+j-1}\xi^j)) g(w_i^{nJ+j-1} \cdot \xi^j) + \lambda \frac{f'(w_{i0}^{nJ+j-1})}{2Jf(w_{i0}^{nJ+j-1})^{\frac{1}{2}}}, \tag{2.21}$$

$$\Delta_j^n w_{ik}^{nJ+j-1} = g_j'(w_0^{nJ+j-1} \cdot G(V^{nJ+j-1}\xi^j)) w_{i0}^{nJ+j-1} \times g'(w_i^{nJ+j-1} \cdot \xi^j)\xi_k^j + \lambda \frac{f'(w_{ik}^{nJ+j-1})}{2Jf(w_{ik}^{nJ+j-1})^{\frac{1}{2}}}, \tag{2.22}$$

这里 $i = 1, 2, \ldots, q$, $k = 1, 2, \ldots, p$, η_n 是第 n 步的学习率.

对任意向量 $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, 我们把 x 的 Euclid 范数记作 $\|x\| = \sqrt{\sum_{i=1}^n x_i^2}$. 记 $\Omega_0 = \{W : E_W(W) = 0\}$ 为误差函数 $E(W)$ 的稳定点集. 在算法的收敛性中, 我们作如下假设:

(A1) 对所有的 $t \in \mathbb{R}$, $|g(t)|$ 与 $|g'(t)|$ 均 Lipschitz 连续;

(A2) $0 < \eta_n < 1$, $\sum_{n=0}^{\infty} \eta_n < \infty$.

为了简化标记, 我们令

$$\begin{aligned} C_2 &= \max\Big\{ \sup_{t\in\mathbb{R}} |g(t)|, \sup_{t\in\mathbb{R}} |g'(t)|, \sup_{t\in\mathbb{R}} |g''(t)|, \\ &\qquad \sup_{t\in\mathbb{R}, 1\leqslant j\leqslant J} |g_j'(t)|, \sup_{t\in\mathbb{R}, 1\leqslant j\leqslant J} |g_j''(t)| \Big\}, \\ C_3 &= \max_{1\leqslant j\leqslant J} \|\xi^j\|, \\ C_4 &= \max\{\sqrt{q}C_2, C_2C_3\}. \end{aligned} \tag{2.23}$$

定理 2 误差函数 $E(W)$ 如 (2.19) 所定义, W^0 为任意初始值, 权序列 $\{W^n\}$ 由迭代法 OGSL1/2 (2.20) 对任意一个初始值生成. 假设条件 (A1) 和 (A2) 均成立, 则一定存在唯一的一个点 $W^* \in \Omega_0$, 使得

$$\lim_{n\to\infty} W^n = W^*, \tag{2.24}$$

$$\lim_{n\to\infty} \|E_W(W^n)\| = \|E_W(W^*)\| = 0. \tag{2.25}$$

3 高阶神经网络[26]

3.1 Sigma-Pi-Sigma (Σ-Π-Σ) 神经网络 (SPSNNs)

首先介绍一下 SPSNNs 的网络结构 (图 1), 其包含一个输入层、一个含有求和单元的隐层 (Σ_1 层)、一个含有求积单元的隐层 (Π 层) 以及一个输出层 (Σ_2 层). M, N, Q 和 1 分别表示各层的节点个数. 连接 Σ_2 层与 Π 层的权记为 $\mathbf{w}_0 = (w_{01}, w_{02}, \ldots, w_{0Q})^{\mathrm{T}} \in \mathbb{R}^Q$. 连接 Π 层与 Σ_1 层的权取固定值 1. 连接输入层与 Σ_1 层第 n 个节点的权记作 $\mathbf{w}_n = (w_{n1}, w_{n2}, \ldots, w_{nM})^{\mathrm{T}}$. 记 $\mathbf{W} = (\mathbf{w}_0^{\mathrm{T}}, \mathbf{w}_1^{\mathrm{T}}, \ldots, \mathbf{w}_N^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{Q+NM}$.

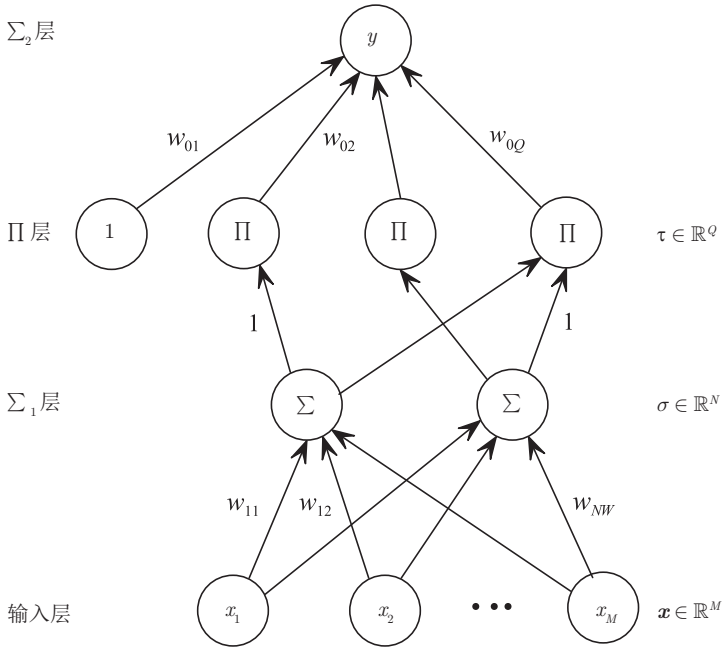

图 1 SPSNNs 的网络结构 ($N = 2, Q = 4$)

我们把网络的输入记作 $\mathbf{x} = (x_1, x_2, \ldots, x_M)^{\mathrm{T}} \in \mathbb{R}^M$, 其中分量 x_M 固定为 1. 这样可以把阈值当作一个权. 假设 $g : \mathbb{R} \to \mathbb{R}$ 是一个给定的 Sigmoid 激活函数, 它可以压缩求和节点的输出值. Σ_1 层的输出向量 $\boldsymbol{\sigma} \in \mathbb{R}^N$ 记作

$$\boldsymbol{\sigma} = (g(\mathbf{w}_1 \cdot \mathbf{x}), g(\mathbf{w}_2 \cdot \mathbf{x}), \ldots, g(\mathbf{w}_N \cdot \mathbf{x}))^{\mathrm{T}}, \tag{3.1}$$

其中 "$\cdot$" 表示内积.

Π 层的输出记作 $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_Q)^{\mathrm{T}} \in \mathbb{R}^Q$. Π 层的激活函数希望是一个关于 Σ_1 层的输出向量 $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_N)^{\mathrm{T}}$ 的各个分量的多项式展开式. 为了达到此目的, 每个求积节点都与 Σ_1 层的某些节点 (如 $\{1\}$ 或者 $\{1, 2\}$) 连接, 且每个节点对应一个特殊的单项式 (如 σ_1 或者 $\sigma_1\sigma_2$). 根据不同的连接单元可以有两种连接方式. 第一种是全连接: Σ_1 层与 Π 层能够全连接, 求积因子的个数为

$$\sum_{n=0}^{N} C_N^n = 2^N.$$

第二种方式是稀疏连接: 进行稀疏连接的求积因子个数少于 2^N. 图 1 给出了一个全连接的 Sigma-Pi-Sigma 网络, Π 层的输出通过 Σ_2 的作用进行加权线性组合.

Λ_q $(1 \leqslant q \leqslant Q)$ 表示 Σ_1 q 个求积节点连接的节点的下标, V_n $(1 \leqslant n \leqslant N)$ 表示 Σ_1 层中所有与第 n 个节点连接的节点的下标, 其中 $\Lambda_q \subseteq \{1, 2, \ldots, N\}$, $V_n \subseteq \{2, 3, \ldots, Q\}$. 例如, 在图 1 中, 与阈值 w_{01} 相对应的第一个节点并没有与 Σ_1 层中的任何节点连接, 所以, $\Lambda_1 = \emptyset$. 而我们有 $\Lambda_2 = \{1\}$, $\Lambda_3 = \{2\}$, $\Lambda_4 = \{1, 2\}$, $V_1 = \{2, 4\}$, $V_2 = \{3, 4\}$. 对 $\{\Lambda_i\}$ 和 $\{V_j\}$ 的不同定义导致了不同的组合. 对任意一个集合 A, 用 $\varphi(A)$ 表示它的元素个数. 那么, 我们有

$$\sum_{q=1}^{Q} \varphi(\Lambda_q) = \sum_{n=1}^{N} \varphi(V_n), \tag{3.2}$$

此式在证明中有用.

Π 层的输出向量 $\boldsymbol{\tau} \in \mathbb{R}^Q$ 通过下式进行计算,

$$\tau_q = \prod_{i \in \Lambda_q} \sigma_i, \quad 1 \leqslant q \leqslant Q, \tag{3.3}$$

这里不妨假定 $\prod_{i\in\Lambda_q} \sigma_i \equiv 1$, 当 $\Lambda_q = \emptyset$ 时.

Σ-Π-Σ 网络的最后输出为

$$y = g(\mathbf{w}_0 \cdot \boldsymbol{\tau}). \tag{3.4}$$

3.2 SPSNN 的批处理梯度学习算法

3.2.1 经典 $L_{1/2}$ 算子

记训练样本集为 $\{\mathbf{x}^j, O^j\}_{j=1}^J \subset \mathbb{R}^M \times \mathbb{R}$, 并记 $y^j \in \mathbb{R}$ $(1 \leqslant j \leqslant J)$ 为输入样本 $\mathbf{x}^j \in \mathbb{R}^M$ 所得到的实际输出. 把传统的不带惩罚项的均方误差函数记作 $\widetilde{E}(\mathbf{W})$, 即

$$\widetilde{E}(\mathbf{W}) = \frac{1}{2}\sum_{j=1}^{J}(y^j - O^j)^2 = \sum_{j=1}^{J} g_j(\mathbf{w}_0 \cdot \boldsymbol{\tau}^j), \tag{3.5}$$

其中

$$g_j(t) = \frac{1}{2}(g(t) - O^j)^2, \quad t \in \mathbb{R}, \quad 1 \leqslant j \leqslant J. \tag{3.6}$$

接下来导出误差函数 $\widetilde{E}(\mathbf{W})$ 的梯度. 注意到

$$\boldsymbol{\tau}^j = (\tau_1^j, \tau_2^j, \ldots, \tau_Q^j)^{\mathrm{T}} = \Big(\prod_{i\in\Lambda_1}\sigma_i^j, \prod_{i\in\Lambda_2}\sigma_i^j, \ldots, \prod_{i\in\Lambda_Q}\sigma_i^j\Big)^{\mathrm{T}}, \tag{3.7}$$

且

$$\sigma^j = (\sigma_1^j, \sigma_2^j, \ldots, \sigma_N^j) = (g(\mathbf{w}_1 \cdot \mathbf{x}^j), g(\mathbf{w}_2 \cdot \mathbf{x}^j), \ldots, g(\mathbf{w}_N \cdot \mathbf{x}^j))^{\mathrm{T}}, \tag{3.8}$$

那么, $\widetilde{E}(\mathbf{W})$ 关于 w_{0q} $(1 \leqslant q \leqslant Q)$ 的偏导数为

$$\widetilde{E}_{w_{0q}}(\mathbf{W}) = \sum_{j=1}^{J} g_j'(\mathbf{w}_0 \cdot \boldsymbol{\tau}^j)\tau_q^j. \tag{3.9}$$

更进一步, 对任意 $1 \leqslant n \leqslant N$, $1 \leqslant m \leqslant M$ 且 $1 \leqslant q \leqslant Q$,

$$\frac{\partial \tau_q}{\partial w_{n,m}} = \begin{cases} \Big(\prod\limits_{i\in\Lambda_q\setminus\{n\}}\sigma_i\Big) g'(\mathbf{w}_n \cdot \mathbf{x}) x_m, & \text{若 } q \neq 1, \quad n \in \Lambda_q, \\ 0, & \text{若 } q = 1, \quad \text{或 } n \notin \Lambda_q. \end{cases} \tag{3.10}$$

根据 (3.3) 和 (3.10), 对任意 $1 \leqslant n \leqslant N$, $1 \leqslant m \leqslant M$, 我们有

$$\widetilde{E}_{w_{n,m}}(\mathbf{W}) = \sum_{j=1}^{J} g_j'(\mathbf{w}_0 \cdot \boldsymbol{\tau}^j) \sum_{q=1}^{Q} w_{0q} \frac{\partial \tau_q^j}{\partial w_{nm}}$$

$$= \sum_{j=1}^{J} g_j'(\mathbf{w}_0 \cdot \boldsymbol{\tau}^j) \sum_{q\in V_n} w_{0q} \Big(\prod_{i\in\Lambda_q\setminus\{n\}} \sigma_i^j \Big) g'(\mathbf{w}_n \cdot \mathbf{x}^j) x_m^j, \tag{3.11}$$

其中 (3.10) 中的 $\frac{\partial \tau_q^j}{\partial w_{nm}}$ 表示 $\frac{\partial \tau_q}{\partial w_{n,m}}$ 在 $\sigma_i = \sigma_i^j$ 且 $\mathbf{x} = \mathbf{x}^j$ 处的值.

带 $L_{1/2}$ 惩罚项的误差函数为

$$E(\mathbf{W}) = \widetilde{E}(\mathbf{W}) + \lambda \Big(\sum_{q=1}^{Q} |w_{0q}|^{\frac{1}{2}} + \sum_{n=1}^{N} \sum_{m=1}^{M} |w_{nm}|^{\frac{1}{2}} \Big). \tag{3.12}$$

网络学习的目的是找到一个 $\mathbf{W}^*$ 使得

$$E(\mathbf{W}^*) = \min E(\mathbf{W}). \tag{3.13}$$

梯度算法是解决这类优化问题的一种常用的简单方法. 从任意一个初始值 $\mathbf{W}^0$ 开始, 当一个学习迭代循环结束后更新权值, 这就是所谓的批处理模式. 所以在迭代过程中, 权值的更新如下:

$$\mathbf{W}^{k+1} = \mathbf{W}^k + \Delta \mathbf{W}^k, \quad k = 0, 1, 2, \ldots, \tag{3.14}$$

其中 $\Delta \mathbf{W}^k = (\Delta w_{01}^k, \ldots, \Delta w_{0Q}^k, \Delta w_{11}^k, \ldots, \Delta w_{NM}^k)^{\mathrm{T}}$, 而

$$\Delta w_{0q}^k = -\eta E_{w_{0q}}(\mathbf{W}^k) = -\eta \Big(\sum_{j=1}^{J} g_j'(\mathbf{w}_0^k \cdot \boldsymbol{\tau}^{k,j}) \tau_q^{k,j} + \frac{\lambda \mathrm{sgn}(w_{0q}^k)}{2|w_{0q}^k|^{\frac{1}{2}}} \Big), \quad 1 \leqslant q \leqslant Q,$$

且

$$\begin{aligned} \Delta w_{nm}^k &= -\eta E_{w_{nm}}(\mathbf{W}^k) \\ &= -\eta \Big(\sum_{j=1}^{J} g_j'(\mathbf{w}_0^k \cdot \boldsymbol{\tau}^{k,j}) \sum_{q\in V_n} w_{0q}^k \Big(\prod_{i\in\Lambda_q\setminus\{n\}} \sigma_i^{k,j} \Big) g'(\mathbf{w}_n^k \cdot \mathbf{x}^j) x_m^j \\ &\quad + \frac{\lambda \mathrm{sgn}(w_{nm}^k)}{2|w_{nm}^k|^{\frac{1}{2}}} \Big), \quad 1 \leqslant n \leqslant N, \quad 1 \leqslant m \leqslant M, \end{aligned}$$

其中学习率 $\eta > 0$.

3.2.2 光滑 $L_{1/2}$ 正则算子

前文我们已经提到, 一般的 $L_{1/2}$ 正则化项在原点处的不光滑性不仅导致收敛性分析困难, 而且我们的数值实验表明, 在 Σ-Π-Σ 也能够导致振荡. 因此, 这里也把改进的 $L_{1/2}$ 正则化项引入网络中. 那么, 相应的带光滑 $L_{1/2}$ 正则化惩罚项的误差函数如下:

$$E(\mathbf{W}) = \widetilde{E}(\mathbf{W}) + \lambda \Big(\sum_{q=1}^{Q} f^{\frac{1}{2}}(w_{0q}) + \sum_{n=1}^{N} \sum_{m=1}^{M} f^{\frac{1}{2}}(w_{nm}) \Big), \tag{3.15}$$

误差函数的梯度为

$$E_{\mathbf{W}}(\mathbf{W}) = (E_{w_{01}}(\mathbf{W}), \ldots, E_{w_{0Q}}(\mathbf{W}), E_{w_{11}}(\mathbf{W}), \ldots, E_{w_{NM}}(\mathbf{W}))^{\mathrm{T}}, \tag{3.16}$$

其中

$$E_{w_{0q}}(\mathbf{W})=\sum_{j=1}^{J}g_j'(\mathbf{w}_0\cdot\boldsymbol{\tau}^j)\tau_q^j+\frac{\lambda f'(w_{0q})}{2f^{\frac{1}{2}}(w_{0q})},\quad q=1,2,\ldots,Q,$$

且

$$E_{w_{nm}}(\mathbf{W})=\sum_{j=1}^{J}g_j'(\mathbf{w}_0\cdot\boldsymbol{\tau}^j)\sum_{q\in V_n}w_{0q}\Big(\prod_{i\in\Lambda_q\setminus\{n\}}\sigma_i^j\Big)g'(\mathbf{w}_n\cdot\mathbf{x}^j)x_m^j+\frac{\lambda f'(w_{nm})}{2f^{\frac{1}{2}}(w_{nm})},$$
$$n=1,2,\ldots,N,\quad m=1,2,\ldots,M.$$

接下来介绍我们的带光滑 $L_{1/2}$ 正则化惩罚项的批处理梯度法: 从任意一个初始值 $\mathbf{W}^0$ 出发, 权 $\{\mathbf{W}^k\}$ 按照下式更新,

$$\mathbf{W}^{k+1}=\mathbf{W}^k+\Delta\mathbf{W}^k,\quad k=0,1,2,\ldots,\tag{3.17}$$

其中

$$\Delta w_{0q}^k=-\eta\Big(\sum_{j=1}^{J}g_j'(\mathbf{w}_0^k\cdot\boldsymbol{\tau}^{k,j})\tau_q^{k,j}+\frac{\lambda f'(w_{0q}^k)}{2f^{\frac{1}{2}}(w_{0q}^k)}\Big),\quad q=1,2,\ldots,Q,\tag{3.18}$$

且

$$\begin{aligned}\Delta w_{n,m}^k&=-\eta E_{w_{n,m}}(\mathbf{W}^k)\\&=-\eta\Big(\sum_{j=1}^{J}g_j'(\mathbf{w}_0^k\cdot\boldsymbol{\tau}^{k,j})\sum_{q\in V_n}w_{0q}^k\Big(\prod_{i\in\Lambda_q\setminus\{n\}}\sigma_i^{k,j}\Big)g'(\mathbf{w}_n^k\cdot\mathbf{x}^j)x_m^j\\&\quad+\frac{\lambda f'(w_{n,m}^k)}{2f^{\frac{1}{2}}(w_{n,m}^k)}\Big),\quad n=1,2,\ldots,N,\quad m=1,2,\ldots,M,\end{aligned}\tag{3.19}$$

这里学习率 $\eta>0$ 是个常数.

3.2.3 我们的主要结果

这里主要给出一些带光滑 $L_{1/2}$ 正则项 (3.17) 的批处理梯度法收敛性结果. 首先给出一些收敛的充分条件:

(A1) 对于 $t\in\mathbb{R}$, $|g(t)|$, $|g'(t)|$ 以及 $|g''(t)|$ 一致有界;

(A2) $\|\mathbf{w}_0^k\|$ $(k=0,1,2,\ldots)$ 一致有界;

(A3) η 与 λ 应满足 $0<\eta<\frac{1}{\lambda\rho+C}$, 其中 $\rho=\frac{\sqrt{6}}{4\sqrt[3]{a^3}}$ 且 C 是 (3.20) 中定义的一个常数;

(A4) 存在一个紧集 $\Phi\in\mathbb{R}^{Q+NM}$ 使得 $\mathbf{W}^k\in\Phi$, 并且集合 $\Phi_0=\{\mathbf{W}\in\Phi:E_{\mathbf{W}}(\mathbf{W})=0\}$ 包含有限个点.

证明中需要用到的一些特别的常数:

$$C_0=\max_{1\leqslant j\leqslant J}\{\|\mathbf{x}^j\|,|O^j|\},$$
$$C_1=\max\Big\{\sup_{t\in\mathbb{R}}|g(t)|,\sup_{t\in\mathbb{R}}|g'(t)|,\sup_{t\in\mathbb{R}}|g''(t)|,\sup_{t\in\mathbb{R},\,1\leqslant j\leqslant J}|g_j'(t)|,\sup_{t\in\mathbb{R},\,1\leqslant j\leqslant J}|g_j''(t)|\Big\},$$
$$C_2=\max\|\mathbf{w}_0^k\|,$$

$$
\begin{aligned}
&C_3 = NQ(C_0)^2(C_1)^{2N},\\
&C_4 = C_0(C_1)^{N-1},\\
&C_5 = JNQ(C_0)^2(C_1)^{N+1}C_2 + \frac{JQ}{2}(C_0)^2(C_1)^{N+1}C_2,\\
&C_6 = \frac{JC_1}{2}\max\{1, C_3\},\\
&C_7 = JC_1\max\{Q(C_1)^{2N}, (C_2)^2C_3\},\\
&C = C_5 + C_6 + C_7.
\end{aligned} \tag{3.20}
$$

定理 3 设误差函数如 (3.15) 中所定义, 且假设 (A1)–(A3) 都成立. 记从任意一个初始值出发, 迭代算法 (3.17) 生成的权序列为 $\{\mathbf{W}^k\}$, 那么, 我们有以下收敛性结果:

(i) $E(\mathbf{W}^{k+1}) \leqslant E(\mathbf{W}^k)$, $k = 0, 1, 2, \ldots$;

(ii) 存在 $E^* \geqslant 0$ 使得

$$\lim_{k\to\infty} E(\mathbf{W}^k) = E^*;$$

(iii) $\lim_{n\to\infty} \|E_{\mathbf{W}}(\mathbf{W}^k)\| = 0$;

更进一步, 若假设 (A4) 也成立, 那么, 我们有以下强收敛:

(iv) 存在一点 $\mathbf{W}^* \in \Phi_0$ 使得

$$\lim_{k\to\infty} (\mathbf{W}^k) = \mathbf{W}^*.$$

4 双并行前馈神经网络[27]

双并行前馈神经网络[28] 是一种并行连接运行机制. 一个简单的双并行前馈神经网络包含三层: 有 p 个输入节点的输入层, 有 L 个隐单元的隐层, 以及只有一个输出节点的输出层. 为了叙述方便, 以下我们将输出空间设定为 1 维. 多维输出空间可以类似导出.

本节将连接隐层与输出层的权向量记为 $W = (w_1, \ldots, w_L)^{\mathrm{T}} \in \mathbb{R}^L$, 连接输入层与隐层的权矩阵记为 $V = (v_{i,j})_{L\times p}$, 其中 $V_i = (v_{i,1}, \ldots, v_{i,p})^{\mathrm{T}}$ 是连接输入层与第 i 个隐节点的权向量. 向量 u 表示连接输入层与输出层的权向量, 即 $u = (u_1, \ldots, u_p)^{\mathrm{T}} \in \mathbb{R}^p$. 令 $g : \mathbb{R} \to \mathbb{R}$ 是隐层与输出层的一个激活函数. 对任意的 $z = (z_1, \ldots, z_L)^{\mathrm{T}} \in \mathbb{R}^L$, 我们记

$$G(z) = (g(z_1), \ldots, g(z_L))^{\mathrm{T}} \in \mathbb{R}^L, \tag{4.1}$$

对任意给定的输入向量 $x \in \mathbb{R}^p$, 可以计算出神经系统的输出 $y \in \mathbb{R}^d$,

$$y = w \cdot G(Vx) + u \cdot x. \tag{4.2}$$

注意到, 通常情形下一个神经系统应该包含阈值项. 然而, 我们把输入向量 x 的最后一个分量记为 -1, 那么, V_i 的最后一个分量就是相应的阈值项. 这个小技巧可以使我们避免明确地在问题描述中写出阈值项.

向网络输入一个给定的训练样本集 $\{x_j, t_j\}_{j=1}^J \subset E^p \times \mathbb{R}$, 误差函数定义为

$$E = \sum_{j=1}^{J} (w \cdot G(Vx_j + u \cdot x_j) - t_j)^2 + \lambda(|w|^{1/2} + |u|^{1/2}), \tag{4.3}$$

其中 λ 是一个调整参数. 我们采用极端学习机算法来最小化误差函数, 即权值矩阵 V 直接随机给定, 而 w 和 u 通过学习确定.

由于 $L_{1/2}$ 正则化算子的引入, 无法直接用最小二乘法 (伪逆运算) 来最小化误差函数, 所以, 我们使用梯度下降法来达到此目的. 在最小化过程中, 权 $W=(u^{\mathrm{T}},w^{\mathrm{T}})^{\mathrm{T}}$ 按照下式迭代更新,

$$W^n=W^{n-1}-\eta\Delta W^n,\quad n=1,2,\ldots, \tag{4.4}$$

其中 $\Delta W^n=((\Delta u^n)^{\mathrm{T}},(\Delta w^n)^{\mathrm{T}})^{\mathrm{T}}$ 是误差函数 (4.3) 的梯度,

$$\Delta u^n=2\sum_{j=1}^{J}(w^n\cdot G(Vx_j)+u^n\cdot x_j)x_j+\lambda\frac{\mathrm{sgn}(u^n)}{2|u^n|^{1/2}}, \tag{4.5}$$

$$\Delta w^n=2\sum_{j=1}^{J}(w^n\cdot G(Vx_j)+u^n\cdot x_j)G(Vx_j)+\lambda\frac{\mathrm{sgn}(w^n)}{2|w^n|^{1/2}}, \tag{4.6}$$

学习率 $\eta>0$.

极端学习机 (ELM) [29] 最初是为只含有一个隐层的前馈网络而提出来的, 是指当隐层与输出层之间的权值由一个伪逆运算决定时, 输入层与隐层之间的权就被随机选取. 针对双并行前馈神经网络, 我们提出的基于极端学习机, 并且利用 $L_{1/2}$ 正则化的方法 ($\mathrm{ELM}_{1/2}$) 描述如下:

(1) 对输入层与隐层之间的权 v_i $(i=1,\ldots,L)$ 随机赋值;

(2) 在零附近随机选取值赋予 W_0, 按照 (4.4) 更新权直到误差 $\sum_{j=1}^{J}(w^n\cdot G(Vx_j)+u\cdot x_j-t_j)^2$ 变得足够小且 $W^n=(u^{\mathrm{T}},w^{\mathrm{T}})^{\mathrm{T}}$ 的许多分量非常接近 0;

(3) 若连接隐层与输出层的权非常小, 则删除相应的隐层中的节点, 重新连接所有权;

(4) 保持输入层与隐层的权 v_i 不变, 再次运行一般的 ELM [30] 得到最终的 w 与 u.

我们注意到, 虽然我们的方法用到了梯度下降法的计算来最小化误差函数, 但是, 这只是对 w 与 u, 而不是针对权矩阵 V, 所以, 计算时间比传统的 BP 算法要少得多. 传统 BP 算法中所有的权值都需要迭代学习.

数值实验表明, 基于极端学习机且利用 $L_{1/2}$ 正则化的方法 ($\mathrm{ELM}_{1/2}$) 在保证精度的前提下, 大大减小了隐单元个数, 达到了很好的结构稀疏化目的 (参见文献 [27]).

5 Takagi-Sugeno 模糊模型[26]

5.1 0 阶 T-S 系统与改进的梯度学习算法

让我们考虑以下两种模糊规则[31,32]:

$$\text{Rule } i:\ \text{IF } x_1 \text{ is } A_{1i} \text{ and } x_2 \text{ is } A_{2i} \text{ and } \ldots \text{ and } x_m \text{ is } A_{mi} \text{ THEN } z \text{ is } y_i, \tag{5.1}$$

其中 i $(i=1,2,\ldots,n)$ 对应于第 i 条模糊规则, n 是模糊规则的数目, y_i 是一个实数.

第一层为输入层, 包含 m 个输入节点, 每个神经元代表 $\mathbf{x}=(x_1,x_2,\ldots,x_m)\in\mathbb{R}^m$ 的一个输入变量. 第二层为隶属层, A_{li} 是 x_l 的一个模糊子集, $A_{li}(x_l)$ 表示模糊判断 "x_l is A_{li}" 的一个 Gauss 隶属函数. 此模糊判断定义为

$$A_{li}(x_l)=\exp\left(-\frac{(x_l-a_{li})^2}{\sigma_{li}^2}\right)=\exp(-(x_l-a_{li})^2b_{li}^2), \tag{5.2}$$

其中 a_{li} 是 $A_{li}(x_l)$ 的中心, σ_{li} 是 $A_{li}(x_l)$ 的宽度, b_{li} 是 $\sigma_{li}(x_l)$ $(i=1,2,\ldots,n,\, l=1,2,\ldots,m)$ 的倒数. 我们分别用以下记号表示 Gauss 隶属函数的中心与宽度的倒数[33]:

$$\mathbf{a}_i=(a_{1i},a_{2i},\ldots,a_{mi}),\quad \mathbf{b}_i=(b_{1i},b_{2i},\ldots,b_{mi})=\left(\frac{1}{\sigma_{1i}},\frac{1}{\sigma_{2i}},\ldots,\frac{1}{\sigma_{mi}}\right),\quad 1\leqslant i\leqslant n, \tag{5.3}$$

连接输入层与隶属层的权可以看作 Gauss 隶属函数的中心与宽度.

第三层是包含 n 个节点的规则层, 第 i 个规则前件通过下式计算,

$$h_i=h_i(\mathbf{x})=A_{1i}(x_1)A_{2i}(x_2)\ldots A_{mi}(x_m)=\prod_{l=1}^{m}A_{li}(x_l),\quad i=1,2,\ldots,n. \tag{5.4}$$

连接第二与第三层的权固定为常数 1.

第四层只含一个输出节点

$$z=\sum_{i=1}^{n}h_iy_i, \tag{5.5}$$

这是规则层输出的线性组合. 我们把结论参数记为

$$\mathbf{a_0}=(y_1,y_2,\ldots,y_n), \tag{5.6}$$

并把它作为连接第三与第四层的权.

5.2 具有光滑 $L_{1/2}$ 正则化的离线梯度神经 - 模糊学习算法

下面的算子 "$\odot$" 将被用来描述我们的学习方法.

定义 1 [33] 设向量 $\mathbf{u}=(u_1,u_2,\ldots,u_n)\in\mathbb{R}^n$, $\mathbf{v}=(v_1,v_2,\ldots,v_n)\in\mathbb{R}^n$. 定义算子 "$\odot$",

$$\mathbf{u}\odot\mathbf{v}=(u_1v_1,u_2v_2,\ldots,u_nv_n)\in\mathbb{R}^n.$$

容易证明该算子的以下性质:

(1) $\|\mathbf{u}\odot\mathbf{v}\|\leqslant\|\mathbf{u}\|\|\mathbf{v}\|$,

(2) $(\mathbf{u}\odot\mathbf{v})\cdot(\mathbf{x}\odot\mathbf{y})=(\mathbf{u}\odot\mathbf{v}\odot\mathbf{x})\cdot\mathbf{y}$,

(3) $(\mathbf{u}+\mathbf{v})\odot\mathbf{x}=\mathbf{u}\odot\mathbf{x}+\mathbf{v}\odot\mathbf{x}$,

其中 $\mathbf{u},\mathbf{v},\mathbf{x},\mathbf{y}\in\mathbb{R}^n$, "$\cdot$" 与 "$\|\cdot\|$" 分别表示普通的内积与模.

设 $\{\mathbf{x}^j,O^j\}_{j=1}^{J}\subset\mathbb{R}^m\times\mathbb{R}$ 是一个应用于网络的训练样本集, J 是训练样本的个数, 模糊规则由 (5.1) 给出. 为了描述方便, 所有的参数都合并为一个权向量

$$\mathbf{W}=(\mathbf{a}_0,\mathbf{a}_1,\ldots,\mathbf{a}_n,\mathbf{b}_1,\ldots,\mathbf{b}_n)\in\mathbb{R}^{(m+1)n}.$$

带 $L_{1/2}$ 正则惩罚项的误差函数定义为

$$\begin{aligned}E(\mathbf{W})&=\widetilde{E}(\mathbf{W})+\lambda\left(\sum_{i=1}^{n}|y_i|^{\frac{1}{2}}+\sum_{i=1}^{n}\sum_{l=1}^{m}|b_{li}|^{\frac{1}{2}}\right)\\&=\frac{1}{2}\sum_{j=1}^{J}(\mathbf{a}_0\cdot\mathbf{h}^j-O^j)^2+\lambda\left(\sum_{i=1}^{n}|y_i|^{\frac{1}{2}}+\sum_{i=1}^{n}\sum_{l=1}^{m}|b_{li}|^{\frac{1}{2}}\right)\end{aligned}$$

$$= \sum_{j=1}^{J} g_j(\mathbf{a}_0 \cdot \mathbf{h}^j) + \lambda\Big(\sum_{i=1}^{n} |y_i|^{\frac{1}{2}} + \sum_{i=1}^{n}\sum_{l=1}^{m} |b_{li}|^{\frac{1}{2}}\Big), \tag{5.7}$$

其中

$$\widetilde{E}(\mathbf{W}) = \frac{1}{2}\sum_{j=1}^{J}(\mathbf{a}_0 \cdot \mathbf{h}^j - O^j)^2,$$

O^j 是第 j 个训练样本 $\mathbf{x}^j$ 的理想输出, $z^j \equiv \mathbf{a}_0 \cdot \mathbf{h}^j$ 是对应的模糊推理结果, 且

$$\mathbf{h}^j = (h_1^j, h_2^j, \ldots, h_n^j) = \mathbf{h}(\mathbf{x}^j), \quad g_j(t) = \frac{1}{2}(t - O^j)^2, \quad t \in \mathbb{R}, \quad 1 \leqslant j \leqslant J, \tag{5.8}$$

对 $q = 1, 2, \ldots, n$, $j = 1, \ldots, J$,

$$\begin{aligned} h_q^j &= \prod_{l=1}^{m} A_{lq}(x_l^j) = \prod_{l=1}^{m} \exp(-(x_l^j - a_{lq})^2 b_{lq}^2) \\ &= \exp\Big(\sum_{l=1}^{m}(-(x_l^j - a_{lq})^2 b_{lq}^2)\Big). \end{aligned} \tag{5.9}$$

需要注意的是, 这里权 y_i 与 b_{li} 被正则化了. 最理想的情形是一些 y_i 与 b_{li} 经过学习 - 正则化过程后变的非常小, 这意味着相应的学习规则 i 与模糊子集 A_{li} 能够从系统里删除掉.

网络的学习目的是找到 $\mathbf{W}^*$ 使得

$$\mathbf{W}^* = \arg\min E(\mathbf{W}). \tag{5.10}$$

梯度下降法通常用来解决优化问题. 误差函数 $E(\mathbf{W})$ 的梯度为

$$\begin{aligned} \frac{\partial E(\mathbf{W})}{\partial \mathbf{W}} =& \Big(\frac{\partial E(\mathbf{W})}{\partial y_1}, \ldots, \frac{\partial E(\mathbf{W})}{\partial y_n}, \frac{\partial E(\mathbf{W})}{\partial a_{11}}, \ldots, \frac{\partial E(\mathbf{W})}{\partial a_{m1}}, \ldots, \\ & \frac{\partial E(\mathbf{W})}{\partial a_{1n}}, \ldots, \frac{\partial E(\mathbf{W})}{\partial a_{mn}}, \frac{\partial E(\mathbf{W})}{\partial b_{11}}, \ldots, \frac{\partial E(\mathbf{W})}{\partial b_{m1}}, \\ & \ldots, \frac{\partial E(\mathbf{W})}{\partial b_{1n}}, \ldots, \frac{\partial E(\mathbf{W})}{\partial b_{mn}}\Big), \end{aligned} \tag{5.11}$$

其中

$$\frac{\partial E(\mathbf{W})}{\partial y_i} = \sum_{j=1}^{J} g_j'(\mathbf{a}_0 \cdot \mathbf{h}^j) h_i^j + \frac{\lambda \mathrm{sgn}(y_i)}{2|y_i|^{\frac{1}{2}}}. \tag{5.12}$$

因为对 $1 \leqslant i \leqslant n$ 有

$$\frac{\partial h_q^j}{\partial \mathbf{a}_i} = \frac{\partial \exp(\sum_{l=1}^{m}(-(x_l^j - a_{lq})^2 b_{lq}^2))}{\partial \mathbf{a}_i} = 0, \quad \forall\, q \neq i, \tag{5.13}$$

所以, 函数 $E(\mathbf{W})$ 关于 $\mathbf{a}_i$ 与 a_{li} 的偏导数分别为

$$\frac{\partial E(\mathbf{W})}{\partial \mathbf{a}_i} = \sum_{j=1}^{J} g_j'(\mathbf{a}_0 \cdot \mathbf{h}^j)\Big(\sum_{q=1}^{n} y_q \frac{\partial h_q^j}{\partial \mathbf{a}_i}\Big) = 2\sum_{j=1}^{J} g_j'(\mathbf{a}_0 \cdot \mathbf{h}^j) y_i h_i^j((\mathbf{x}^j - \mathbf{a}_i) \odot \mathbf{b}_i \odot \mathbf{b}_i), \tag{5.14}$$

$$\frac{\partial E(\mathbf{W})}{\partial a_{li}} = \sum_{j=1}^{J} g_j'(\mathbf{a}_0 \cdot \mathbf{h}^j)\Big(\sum_{q=1}^{n} y_q \frac{\partial h_q^j}{\partial a_{li}}\Big) = 2\sum_{j=1}^{J} g_j'(\mathbf{a}_0 \cdot \mathbf{h}^j) y_i h_i^j (x_l^j - a_{li}) b_{li}^2. \tag{5.15}$$

类似地, 对 $1\leqslant i\leqslant n$, 函数 $\widetilde{E}(\mathbf{W})$ 关于 $\mathbf{b}_i$ 的偏导数为

$$\begin{aligned}\frac{\partial\widetilde{E}(\mathbf{W})}{\partial\mathbf{b}_i}&=\sum_{j=1}^{J}g_j'(\mathbf{a}_0\cdot\mathbf{h}^j)\Big(\sum_{q=1}^{n}y_q\frac{\partial h_q^j}{\partial\mathbf{b}_i}\Big)\\&=-2\sum_{j=1}^{J}g_j'(\mathbf{a}_0\cdot\mathbf{h}^j)y_ih_i^j((\mathbf{x}^j-\mathbf{a}_i)\odot(\mathbf{x}^j-\mathbf{a}_i)\odot\mathbf{b}_i).\end{aligned}\tag{5.16}$$

对 $1\leqslant i\leqslant n, 1\leqslant l\leqslant m$, 函数 $E(\mathbf{W})$ 关于 b_{li} 的偏导数为

$$\begin{aligned}\frac{\partial E(\mathbf{W})}{\partial b_{li}}&=\sum_{j=1}^{J}g_j'(\mathbf{a}_0\cdot\mathbf{h}^j)\Big(\sum_{q=1}^{n}y_q\frac{\partial h_q^j}{\partial b_{li}}\Big)+\frac{\lambda\mathrm{sgn}(b_{li})}{2|b_{li}|^{\frac12}}\\&=-2\sum_{j=1}^{J}g_j'(\mathbf{a}_0\cdot\mathbf{h}^j)y_ih_i^j(x_l^j-a_{li})^2b_{li}+\frac{\lambda\mathrm{sgn}(b_{li})}{2|b_{li}|^{\frac12}}.\end{aligned}\tag{5.17}$$

从一个任意的初始值 W^0 开始, 权 $\{W^t\}$ 按照下式更新,

$$W^{t+1}=W^t+\Delta W^t,\quad t=0,1,2,\ldots,\tag{5.18}$$

其中

$$\Delta y_i^t=-\eta\Big[\sum_{j=1}^{J}g_j'(\mathbf{a}_0^t\cdot\mathbf{h}^{t,j})h_i^{t,j}+\frac{\lambda\mathrm{sgn}(y_i^t)}{2|y_i^t|^{\frac12}}\Big],\tag{5.19}$$

$$\Delta a_{li}^t=-\eta\Big[2\sum_{j=1}^{J}g_j'(\mathbf{a}_0^t\cdot\mathbf{h}^{t,j})y_i^th_i^{t,j}(x_l^j-a_{li}^t)(b_{li}^t)^2\Big],\tag{5.20}$$

$$\Delta b_{li}^t=-\eta\Big[-2\sum_{j=1}^{J}g_j'(\mathbf{a}_0^t\cdot\mathbf{h}^{t,j})y_i^th_i^{t,j}(x_l^j-a_{li}^t)^2b_{li}^t+\frac{\lambda\mathrm{sgn}(b_{li}^t)}{2|b_{li}^t|^{\frac12}}\Big],\tag{5.21}$$

$i=1,2,\ldots,n$, $l=1,2,\ldots,m$, 且学习率 $\eta>0$ 是常数.

5.2.1 光滑 $L_{1/2}$ 正则化

在上一部分中, 最初的 $L_{1/2}$ 规则项是非光滑的, 如前文所述, 这将不可避免地导致很难进行收敛性分析, 以及更重要的是, 数值实验中将会出现振荡. 这在我们的数值实验中是可以观察到的. 为了克服此缺陷, 一个改进的 $L_{1/2}$ 正则项被提出, 从而得到以下的误差函数:

$$\begin{aligned}E(\mathbf{W})&=\frac12\sum_{j=1}^{J}(\mathbf{a}_0\cdot\mathbf{h}^j-O^j)^2+\lambda\Big(\sum_{i=1}^{n}f(y_i)^{\frac12}+\sum_{i=1}^{n}\sum_{l=1}^{m}f(b_{li})^{\frac12}\Big)\\&=\sum_{j=1}^{J}g_j(\mathbf{a}_0\cdot\mathbf{h}^j)+\lambda\Big(\sum_{i=1}^{n}f(y_i)^{\frac12}+\sum_{i=1}^{n}\sum_{l=1}^{m}f(b_{li})^{\frac12}\Big),\end{aligned}\tag{5.22}$$

其中 $f(x)$ 是一个接近于 $|x|$ 的光滑函数.

5.2.2 收敛定理

这里对权值的迭代给出一些收敛性结论. 关于收敛性分析的一些充分条件如下:

(A1) 对所有的 $i = 1, 2, \ldots, n$, $t = 1, 2, \ldots$, 存在一个常数 $C_0 > 0$ 使得 $\|\mathbf{a}_0^t\| \leqslant C_0$, $\|\mathbf{a_i}^t\| \leqslant C_0$ 与 $\|\mathbf{b}_i^t\| \leqslant C_0$ 成立;

(A2) 选取适当的 η 与 λ 使得 $0 < \eta < \frac{1}{Q\lambda + C}$, 其中 $Q = \frac{\sqrt{6}}{4\sqrt{a^3}}$, 且 C 是 (5.23) 中定义的一个常数;

(A3) 集合 $\mathbf{\Omega} = \{\mathbf{W} \in \mathbf{D}_0 : \frac{\partial E(\mathbf{W})}{\partial \mathbf{W}} = 0\}$ 包含有限个点, 其中 $\mathbf{D}_0$ 是一个紧集.

我们列举以下后文收敛分析中将会用到的常数:

$$
\begin{aligned}
&M = \max_{1 \leqslant j \leqslant J} \{\|\mathbf{x}^j\|, \|O^j\|\},\\
&C_1 = \max\{C_0 + M, (C_0 + M)C_0\},\\
&C_2 = JC_0(\sqrt{n}C_0 + C_1)\max\{\sqrt{n}C_1, \sqrt{n}(C_0 + C_1)\max\{C_0, C_1\},\\
&\qquad 2C_1^2(C_0 + C_1)\max\{C_0, C_1\}\},\\
&C_3 = J(\sqrt{n}C_0 + C_1)\max\left\{\frac{1}{2}, 4C_0^2C_1^2, 4C_1^4\right\},\\
&C_4 = \max\{J, 8JC_0^4C_1^2, 8JC_0^2C_1^4\},\\
&C = C_2 + C_3 + C_4,
\end{aligned}
\tag{5.23}
$$

其中 $\mathbf{x}^j$ 是第 j 个给定的训练样本, O^j 是对应的理想输出, n 与 J 分别是模糊规则与训练样本的个数.

定理 4 设误差函数 $E(\mathbf{W})$ 如 (5.22) 所定义, 权序列 $\{\mathbf{W}^t\}$ 按照 (5.18) 由一个任意的初始值 $\mathbf{W}^0$ 更新. 若假设 (A1) 和 (A2) 成立, 则

(i) $E(\mathbf{W}^{t+1}) \leqslant E(\mathbf{W}^t)$, $t = 0, 1, 2, \ldots$;

(ii) 存在 $E^* \geqslant 0$ 使得 $\lim_{t\to\infty} E(\mathbf{W}^t) = E^*$;

(iii) $\lim_{t\to\infty} \|E_{\mathbf{W}}(\mathbf{W}^t)\| = 0$, 进一步, 若假设 (A3) 也成立, 则我们可以得到强收敛;

(iv) 存在一个点 $\mathbf{W}^* \in \Omega$ 使得 $\lim_{t\to\infty} \mathbf{W}^t = \mathbf{W}^*$.

在定理 4 中, (i) 与 (ii) 分别显示了误差函数序列 $E(\mathbf{W}^t)$ 的单调性与收敛性; (iii) 显示了 $E_{\mathbf{W}}(\mathbf{W}^t)$ 的收敛性; (ii) 和 (iii) 也被称为 $\{\mathbf{W}^t\}$ 的弱收敛; $\{\mathbf{W}^t\}$ 的强收敛如 (iv) 所示.

参考文献

1 Baptista D, Morgado-Dias F. A survey of artificial neural network training tools. Neural Comput Appl, 2013, 23: 609–615

2 Xu L, Chen J, Huang D. Analysis of boundedness and convergence of online gradient method for two-layer feedforward neural networks. IEEE Trans Neural Netw Learn Syst, 2013, 24: 1327–1338

3 Mustafa Y. Artificial neural networks pruning approach for geodetic velocity field determination. Bol Cienc Geod, 2013, 19: 558–573

4 Larsen J, Svarer C, Andersen L N. Adaptive regularization in neural network modeling. In: Lecture Notes in Computer Science, vol. 7700. Berlin: Springer-Verlag, 2012, 111–130

5 Huynh H T, Won Y. Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks. Pattern Recognit Lett, 2011, 32: 1930–1935

6 Yu X, Deng F. Convergence of gradient method for training ridge polynomial neural network. Neural Comput Appl, 2013, 22: S333–S339

7 Wu W, Wang J, Cheng M, et al. Convergence analysis of online gradient method for BP neural networks. Neural Netw, 2011, 24: 91–98

8 Xu D, Shao H, Zhang H. A new adaptive momentum algorithm for split-complex recurrent neural networks. Neurocomputing, 2012, 93: 133–136

9 Yu X, Deng F. Gradient algorithm with penalty for training recurrent pi-sigma neural network. Comput Eng Appl, 2013, 49: 43–46

10 Huang G, Wang D. Advances in extreme learning machines. Neurocomputing, 2013, 102: 1–2
11 Miche Y, van Heeswijk M, Bas P, et al. TROPELM: A double-regularized ELM using LARS and Tikhonov regularization. Neurocomputing, 2011, 74: 2413–2421
12 Balasundaram S. Gupta D, Kapil. 1-Norm extreme learning machine for regression and multiclass classification using Newton method. Neurocomputing, 2014, 128: 4–14
13 Fan Y, Wu W, Yang W, et al. A pruning algorithm with $L_{1/2}$ regularizer for extreme learning machine. J Zhejiang Univ Sci C, 2014, 15: 119–125
14 Burden F, Winkler D. Bayesian regularization of neural networks. Meth Mol Biol, 2008, 458: 25–44
15 Ticknor J L. A Bayesian regularized artificial neural network for stock market forecasting. Expert Syst Appl, 2013, 40: 5501–5506
16 Han M, Li D. An norm 1 regularization term ELM algorithm based on surrogate function and Bayesian framework. Acta Auto Sin, 2011, 37: 1344–1350
17 Ling C, Yu Z L, Dong M H. Speech emotion recognition system based on L_1 regularized linear regression and decision fusion. In: Lecture Notes in Computer Science, vol. 6975. Berlin: Springer-Verlag, 2011, 332–340
18 Xu Z B, Zhang H, Wang Y, et al. $L_{1/2}$ regularizer. Sci China Ser F-Inf Sci, 2009, 52: 1–9
19 Xu Z B, Chang X Y, Xu F M, et al. $L_{1/2}$ regularization: A thresholding representation theory and a fast solver. IEEE Trans Neural Netw Learn Syst, 2012, 23: 1013–1027
20 Werbos P J. Beyond regression: New tools for prediction and analysis in the behavioral science. PhD Thsis. Cambridge: Harvard University, 1974
21 Saad D. On-Line Learning in Neural Networks. Cambridge: Cambridge University Press, 1998
22 Wu W, Fan Q W, Jacek M, et al. Batch gradient method with smoothing $L_{1/2}$ regularization for training of feedforward neural networks. Neural Netw, 2014, 50: 72–78
23 Fan Q, Zurada J M, Wu W. Convergence of online gradient method for feedforward neural networks with smoothing $L_{1/2}$ regularization penalty. Neurocomputing, 2014, 131: 208–216
24 Farrell J A. On performance evaluation in on-line approximation for control. IEEE Trans Neural Netw, 1998, 9: 1001–1007
25 Wilson D R, Martinez T R. The general inefficiency of batch training for gradient descent learning. Neural Netw, 2003, 16: 1429–1451
26 Liu Y, Wu W, Fan Q, et al. A modified gradient learning algorithm with smoothing $L_{1/2}$ regularization for Takagi-Sugeno fuzzy models. Neurocomputing, 2014, 138: 229–237
27 Khan A, Yang J, Wu W. Double parallel feedforward neural network based on extreme learning machine with $L_{1/2}$ regularizer. Neurocomputing, 2014, 128: 113–118
28 He M. Theory, application and related problems of double parallel feedforward neural networks. PhD Thesis. Xi'an: Xidian University, 1993
29 Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: Theory and applications. Neurocomputing, 2006, 70: 489–501
30 Yao M C, Li W T, Liu Y. Double parallel extreme learning machine. Energy Procedia, 2011, 13: 7413–7418
31 Campo I, Echanobe J, Bosque G, et al. Efficient hardware/software implementation of an adaptive neuro-fuzzy system. IEEE Trans Fuzzy Syst, 2008, 16: 761–778
32 Jang J S R. ANFIS adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybernet, 1993, 23: 665–685
33 Wu W, Li L, Yang J, et al. A modified gradient-based neuro-fuzzy learning algorithm and its convergence. Inform Sci, 2010, 180: 1630–1642

$L_{1/2}$ regularization methods for weights sparsification of neural networks

WU Wei & YANG Jie

Abstract On the premise of appropriate learning accuracy, the number of the neurons of a neural network should be as less as possible (constructional sparsification), so as to reduce the cost, and to improve the robustness and the generalization accuracy. We study the constructional sparsification of feedforward neural networks by

using regularization methods. Apart from the traditional L_1 regularization for sparsification, we mainly use the $L_{1/2}$ regularization. To remove the oscillation in the iteration process due to the nonsmoothness of the $L_{1/2}$ regularizer, we propose to smooth it in a neighborhood of the nonsmooth point to get a smoothing $L_{1/2}$ regularizer. By doing so, we expect to improve the efficiency of the $L_{1/2}$ regularizer so as to surpass the L_1 regularizer. Some of our recent works in this respect are summarized in this paper, including the works on BP feedforward neural networks, higher order neural networks, double parallel neural networks and Takagi-Sugeno fuzzy models.

Keywords **neural network, sparsification, $L_{1/2}$ regularization**

MSC(2010) **68T05**

doi: 10.1360/N012015-00110