神经网络学习中"附加样本"的技术*

张 铃[1,3] 张 钹[2,3]

[1](安徽大学人工智能研究所 合肥 230039)
[2](清华大学计算机科学与技术系 北京 100084)
[3](清华大学智能技术与系统国家重点实验室 北京 100084)

摘要 本文将网络的先验知识和网络的样本集知识有机结合起来,提出"附加样本"的神经网络新学习算法,其计算复杂性仅为多项式(上界$\leqslant O(n^4)$),用该算法可以设计出性能更好的神经网络.本文第1节简单介绍FP算法以及FP覆盖算法,第2节提出FP统计附加样本算法.最后举一例子说明用该算法可以设计出性能良好的网络.

关键词 神经网络,FP算法,附加样本,最小覆盖.
中图法分类号 TP18

对于符号主义和连接主义的优劣已有无数的讨论,除个别人外,大多数人均认为应将符号主义的方法和连接主义的方法有机地结合起来(或称为综合集成).这种观点显然是正确的,而问题在于应如何进行有机地结合.本文想对这种结合的方法进行探索,文中只针对如何将先验知识(符号信息)和神经网络的学习方法(准符号信息)相结合的问题进行讨论.

神经网络的学习问题应包括两个部分:①网络拓扑的给定;②从样本集中提取有用的信息.于是如何将先验知识与神经网络的学习问题结合起来,也必须从两方面着手,即如何从先验知识中为网络的拓扑结构的给定和样本的学习提供有用的信息.

关于如何将先验知识用于网络的拓扑的确定的问题,将另文讨论.本文只就如何将先验知识与样本集的学习问题相结合,提出我们的解决方法:用设置"附加样本"的技术,将两者结合起来.并以FP算法为例,讨论如何用新方法设计具有更好性能的神经网络.最后给出模拟结果,以资验证.本文第1节讨论在FP算法基础上建立起来的FP覆盖算法,第2节给出FP统计"附加样本"方法,并给出算法复杂性的估计.最后举一个例子,验证新方法的效果.

1 FP覆盖算法

1.1 FP算法[1]

设神经元元件A有n个输入,1个输出.下面先讨论各分量只取$\{-1,1\}$二值的情况.再设A的输入x与输出y的对应关系为:$y=\mathrm{Sgn}(W*x-\theta)$,其中$W$是元件$A$的权系数矩阵,$\theta$是阈值向量,$\mathrm{Sgn}(x)$表示符号函数.给定训练样本集$K=\{r^0=(x^0,y^0),r^1=(x^1,y^1),\ldots,r^{p-1}=(x^{p-1},y^{p-1})\}$.作如图1所示的网络结构.

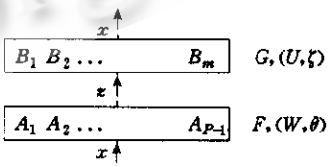

图1

用W^i和θ^i分别表示第1元件层第i神经元的权向量权和阈值,其表达式由公式(1)(2)给出.

$$w^i=(x^i)^T \qquad i=1,2,\ldots,p-1 \tag{1}$$

$$\theta^i=\begin{cases}n-d_i+1 & \text{当 } d_i \text{ 是偶数时}\\ n-d_i & \text{当 } d_i \text{ 是奇数时}\end{cases} \qquad i=1,2,\ldots,p-1 \tag{2}$$

* 本文研究得到国家自然科学基金、国家863高科技项目基金和国家"攀登计划"基金资助.作者张铃,1937年生,教授,主要研究领域为人工智能理论,人工神经网络理论.张钹,1935年生,教授,博士导师,中国科学院院士,主要研究领域为人工智能理论及应用,计算机应用.

本文通讯联系人,张铃,合肥 230039,安徽大学人工智能研究所

本文1997-03-10收到原稿,1997-06-09收到修改稿

$$W=(w^i),\theta=(\theta_i)$$

其中 $d_i=\min_{j\neq i}d(x^i,x^j),i=1,2,\ldots,p-1$. $d(x,y)$表示 x,y 的海明距离.

用 u^i 和 ξ_i 分别表示第 2 层第 i 个神经元的权和阈值,其表达式由(3)和(4)给出. 令

$$u_j^i=\begin{cases}0, & \text{当 } y_j^i=y_j^i \text{ 时}\\ y_j^i, & \text{其他}\end{cases}\quad i=1,2,\ldots,m;j=1,2,\ldots,p-1. \tag{3}$$

$$\xi_i=-(y_i^0+u_1^i+\ldots+u_{p-1}^i) \tag{4}$$

取 $U=(u^i),\xi=(\xi_i)$.

定理 1.1. 当给定训练样本集 $K=\{r^0,r^1,\ldots,r^{p-1}\}$,按式(1)(2)定义 W 和 θ,按公式(3)(4)定义 U 和 ξ. 则图 1 对应的网络将 x^i 变换成 $y^i,i=0,1,\ldots,p-1$,即每个样本输入向量都是吸引中心. 且各样本的吸引域为:$D(x^i)=\{x|\langle x^i,x\rangle>d_i/2\},i=1,2,\ldots,p-1,D(x^0)=\{x|\langle x^i,x\rangle\leqslant d_i/2,i=1,2,\ldots,p-1\}$.

1.2 FP 算法的推广

现将上面的结论推广到输入是一般的实向量的情况. 显然,只要对第 1 层的元件的权、阈值适当地进行修改即可.

先定义

$$\mathrm{Sgn}(x)=\begin{cases}+1, & \text{当 } x>0 \text{ 时}\\ -1, & \text{当 } x\leqslant 0 \text{ 时}\end{cases}$$

设给定训练样本集 $K=\{r^0,r^1,\ldots,r^{p-1}\}$,并设每个样本输入向量均为单位向量.

令

$$d_i=\max_{j\neq i}\{\langle x^i,x^j\rangle\} \tag{5}$$

定义

$$\theta_i=\sqrt{(1+d_i)/2} \tag{6}$$

令

$$W=(w^i),\theta=(\theta_i)\qquad i=1,2,\ldots,p-1 \tag{7}$$

定理 1.2. 按公式(5)~(7)定义第 1 层元件的权矩阵和阈值向量,其第 2 层元件的权矩阵和阈值向量仍按(3)(4)式定义,则网络对每个输入向量 x^i 的吸引域为

$$D(x^i)=\{x|\langle x^i,x\rangle>\sqrt{(1+d_i)/2}\},i=1,2,\ldots,p-1$$

$$D(x^0)=\{x|\langle x^i,x\rangle\leqslant\sqrt{(1+d_i)/2},i=1,2,\ldots,p-1\}$$

且网络没有假吸引中心.

以上两节的详细内容参见文献[1].

1.3 FP 算法的几何意义

FP 算法第 1 元件层的作用是将 p 个样本变换到 $(p-1)$ 维 Z 空间(隐层元空间)中的 p 个最广位置. 然后,第 2 层将 Z 空间中占最广位置的 p 个点映射到 Y 空间(输出空间)中指定的 p 个位置.

若令输入向量的长度相等,如设均为单位长度,则输入向量的定义域就是 n 维空间中的单位球面(设为 S^n),于是由方程 $(w*x-\theta)>0$ 决定的半空间与 S^n 的交就是:S^n 上以 w(设 $w\in S^n$)为中心的某个"球"形领域(设为 $D(w,r)$),于是神经元(由 W,θ 决定的)、半空间(由方程 $\langle w*x-\theta\rangle>0$)、以 w 为中心的领域 $D(w,r)$ 这三者只是同一东西的 3 种不同表示而已. 每个领域覆盖的范围就是以其中心为样本输入的容错范围. 在这样的几何理解下,求隐层的综合问题就变成求某种覆盖的问题,其几何意义十分明显.

我们在文献[1]中已证明,作为通用的联想记忆器,FP 算法给出的网络是最优的. 但是,对具体给定的样本集而言,FP 算法给出的网络的结构未必是最优的,特别在样本输入个数比其对应的不同的输出个数多得多的情况下,更是如此.

一般情况下,最后在 Y 空间的 p 个点,不是完全不同的,往往只有 k 个不同,而且 k 比 p 少得多,如在手写阿拉伯数字识别中,输入的样本数可能是几千甚至几万,但不同的输出只有 10 个. 在这种情况下,若用 FP 算法,得到的网络的隐单元个数可能过多,故需在 FP 算法基础上给出针对这种情况的有效算法. 下面提供一个暂称之为"FP 覆盖法"的算法,可以在一定程度上降低隐层的神经元个数.

1.4 FP 覆盖算法

设样本集 K 为

$$K=\{r^t=(x^t,y^t),t=1,2,\ldots,m\},K\text{ 中不同的 }y^t\text{ 只有 }k\text{ 个}(k\ll m).$$

设样本的输出为 y^t 的样本标号的集合,记为 $I(t)$,其对应的输入集合记为 $P(t),t=1,2,\ldots,k$. 并将输入样本的上标按 $I(0),I(1),\ldots,I(k-1)$ 顺序排序.

固定 t,对 $i\in I(t)$,对应的输入 $x^i(\in P(t))$.

令

$$d_i(t)=\min_{x\notin P(t)}\{d(x,x^i)\}$$

作网络第1层神经元 $A_i:w^i=(x^i)^T$,

$$\theta_i=\begin{cases}n-d_i(t)+1, & \text{当 } d_i(t)\text{为偶数时}\\ n-d_i(t), & \text{当 } d_i(t)\text{为奇数时}\end{cases}$$

令 $D_i(t)=\{x\mid(x,x^i)>d_i/2\}$,称之为以 x^i 为中心的邻域.令 $D(t)=\bigcup D_i(t)$.

由FP算法知,当样本输入 $x\notin P(t)$,则 $x\notin D(t)$;当 $x\in P(t)$,则 $x\in D(t)$,$t=1,2,\ldots,k$.

若我们可以从 $D(t)$ 中选出 $P(t)$ 的子覆盖 $D'(t)$,令 $D'(t)$ 中邻域中心 x^i 的标号的集合为 $I'(t)$,于是在网络第1层只留下 $A_i,i\in\bigcup I'(t)$ 的神经元,就可完成将对应于输出不同的输入变换到 Z 空间中最广位置的任务.

显然,若我们每次能从 $D(t)$ 中选出最小覆盖 $D'(t)$,则在一定意义下就使隐层的神经元个数达到最少.

可惜,求最小覆盖问题是一个典型的NPC问题,目前尚无有效的求解方法.

从以上分析得出,任何一种求(次优)最小覆盖的方法与FP算法相结合,都将给出一个神经网络的学习算法.

现将学习步骤总结如下:

FP覆盖算法:

(1) 按FP算法求各不同输出的样本对应的领域集 $D(t),t=0,1,2,\ldots,k-1$(k 为样本的不同输出的个数);

(2) 按某种求最小覆盖的算法,从 $D(t)$ 中求出 $P(t)$ 的最小(或次小)覆盖 $D'(t)$;

(3) 令 $I'(t)$ 是 $D'(t)$ 中领域对应的指标集,对 $D'(t)$ 中的领域 $D_i(t)$,作对应的神经元 $A_i,i\in\bigcup I'(t)$;

(4) 构造网络:第1层由元件 $A_i(i\in\bigcup I'(t))$ 构成;输出层,取 n(n 是输出向量的维数)个神经元 $B_s,s=1,2,\ldots,n$,然后按公式(3)(4)求其对应的权和阈值.

2 附加样本法

2.1 先验(知识)附加样本法

利用神经网络进行联想学习的目的,不仅要求记住样本集中的对应关系,而且要求能用已学过的网络对未知的输入进行预测,故对网络设计的要求,不但要求网络的结构简单,更主要要求的是所得到的网络具有很好的性能,也许后者比前者更重要.故在求网络的结构时,不但要降低网络元件个数,更重要的是使所求的网络具有良好的性能,为达到这个目的,充分利用先验知识是解决问题的途经之一.

若我们将网络的学习问题看成是求一个满足一定条件的函数问题,假设我们已经从某些先验知识中知道这个函数的一些性质,问是否可以给出一个办法,把所知道的先验知识赋予神经网络(或说是否可将先验知识与神经网络的某种学习方法有机地结合起来),使得学习问题变得容易,或说是加快学习的过程,而且能使最后得到的网络具有所给定的先验知识.

下面给出一个暂且称之为"附加样本"的方法.其主要思想是:设法将这种先验知识用某种方法化成适当的"附加样本"加入到原先的样本集中,然后再进行学习.这样,经过学习之后,所得的函数就具有对应的先验知识,使网络具有更好的性能.

例如设所求的函数 F 是偶函数,则对原来每一样本 (x,y),我们都添加一个附加样本 $(-x,y)$,然后用FP覆盖算法求解.这样所求到的函数也就具有偶函数性质.这就说明,利用"附加样本"法,将先验知识加到原来的样本集中再进行学习能提高网络的性能.

当然,我们也可以用增加隐层元的方法使对应的函数成为偶函数,例如对每个隐元 (w,θ),增加一个对偶的元件 $(-w,\theta)$,第2层各元件的权值按FP算法确定,其阈值按原来公式求出后再乘以2即可.

又如,如果我们预先知道所求的网络具有某种"若 A 则 B"形式的性质,再设 A 已由样本 (a,y) 表达,那么,我们根据"若 A 则 B"的性质,必有性质 B,然后构造一个能表达性质 B 的样本 (b,z) 作为附加样本加入到原样本集中,再进行学习.这样我们就可以将符号主义常用的、表示知识的"若 A 则 B"的形式化为准符号形式(样本形式),与神经网络的学习结合起来.

按照这个想法,我们可以对一些常见的性质进行研究,给出求其对应的"附加样本"的方法,建立一个"附加样本库"以供求解具体网络时之用.这种附加样本法称为先验(知识)附加样本法.

2.2 统计附加样本法

下面以这种思想来求解带噪音的输入的样本学习问题.

上面我们讨论的学习问题都认为每个样本输入、输出都是正确无误的,但实际上,由于噪音等因素,输入、输出总

是有误差的.

例如设正确的输入应为 x^1,但由于噪音等原因,x^1 输入可能被变为 x^2,x^3,x^4 等等.按照 FP 算法是将所有的输入等同看待,即对每个样本的输入都构造对应的领域 $D(t)$,这显然不合理.按理说对 x^1 的容错能力的要求应优于对 x^2,x^3,x^4 的要求,若我们称 x^1 为标准输入,那么,我们的任务就是要设法求出标准输入的位置,然后利用神经网络的学习方法求得对应的网络,使所得的网络本身能保证对于标准输入具有所希望的容错能力,而不去着重考虑其他非标准的输入,同时把那些变化过大(误差过大)的输入(在标准输入吸引区之外的输入)看成是错误的输入,设法把它们去掉,这样得到的网络自身就有一定纠错能力.另一方面,因为不去考虑那些次要的输入的容错能力的多寡,使得学习问题的规模缩小,从而加速了学习速度,一举两得.比如识别手写数字"2",一般"2"字有几种基本的不同写法,其余都是这几种写法的稍微的变形,如果我们能给出一种办法从输入样本集中求出这些标准的输入样本来,那么对于完成识别任务显然是很有利的.

现在,问题在于如何从众多的样本输入(设输入向量的长度是等长的)中求出其中的标准输入向量来.为此,我们借用统计学中方差分析中的一些概念来讨论这个问题.我们将问题提出为:给定一个 n 维空间的点集 $P=\{x_1,x_2,\ldots,x_m\}$,再给定一个常数 $h<m$,求将 P 分成 h 个类,使各类中的点到类的中心的距离的平方和最小.

下面我们用一个移动中心聚类算法(加以改进)来解决上述的问题.

2.2.1 串行移动中心聚类算法

A. 问题的提出

给定一个 n 维空间的点集 $P=\{x_1,x_2,\ldots,x_m\}$,再给定一个常数 $h<m$,求将 P 分成 h 个类,使各类中的点到类的中心的距离的平方和最小.

B. 串行移动中心求聚类的算法

(1) 任取 P 中的 h 个点,记为:$A(0)=\{a_1(0),a_2(0),\ldots,a_h(0)\}$.

(2) 对 P 中元素,实行"关于 $A(j-1)$ 按'就近原则'进行分类".得到的分类集合为:$P(j)=\{P_1(j),P_2(j),\ldots,P_h(j)\}$.开始时,令 $j=1$.

(3) 求 $P_i(j)$ 的重心 $a_i(j)$,得:集合 $A(j)=\{a_1(j),a_2(j),\ldots,a_h(j)\}$.

(4) 对任一个 $x\in P$,执行"关于 $A(j)$ 按'就近原则'进行分类",设 x' 是第 1 个改变类属的点,不妨设 x' 原属于 $P_1(j)$,而现在属于 $P_2(j)$.

(4.1) 若 $d(a_1(j),a_2(j))\geqslant a/m$,则令 $P_1(j+1)=P_1(j)/x'$;$P_2(j+1)=P_2(j)\cup x'$;$P_i(j+1)=P_i(j)$,$j\geqslant 3$.令 $j\leftarrow j+1$,回第(3)步(其中 $2a$ 是 P 中各点间的最短距离).不然,

(4.2) 不妨设 $t_1(j)\leqslant t_2(j)$,令

$$x'=\max_{x_i\in P_1(j)}\{x_i\,|\,d(x_i,a_1(j))\},$$

令 $a_1(j+1)=x'$,$a_i(j+1)=a_i(j)$,$i\geqslant 2$;$P_1(j+1)=\{x'\}$,$P_2(j+1)=P_1(j)\cup P_2(j)/\{x'\}$,$P_i(j+1)=P_i(j)$,$i\geqslant 3$.令 $j\leftarrow j+1$,回第(3)步(其中 $t_i(j)=|P_i(j)|$).

(5) 若 P 中所有的点均不改变类属,成功.$A(j)$ 和 $P(j)$ 为所求.

称 $A(j)$ 为用移动中心法求到的统计附加样本集.

注:"关于 $A(j)$ 按'就近原则'进行分类"是指若 x 距 $a_i(j)$ 最近,则将 x 划归到 $P_i(j)$ 类中.

C. 并行移动中心聚类算法

只要将串行算法中的第(4)步改为:对 P 中的点执行"关于 $A(j)$ 按'就近原则'进行分类",设得到的分类为 $P(j+1)$,若存在某个 i,使得 $P_i(j+1)\neq P_i(j)$,则返回第(3)步;不然,转入第(5)步.

注:关于并行移动中心聚类算法的初等性质可参见文献[2].

D. 算法复杂性的上界估计

定理 2.1. 串行移动中心聚类算法定能在有限步内收敛于一稳定状态,且其计算量的阶为 $O(n^4)$.

证明:设算法到某步得到 x',设 x' 是一改变类属的点,不妨设 x' 原属于 $P_1(j)$,而现在属于 $P_2(j)$.令 $P_1(j+1)=P_1(j)\backslash x'$;$P_2(j+1)=P_2(j)\cup x'$;$P_i(j+1)=P_i(j)$,$i=3,4,\ldots$.

若执行上述中(4.1)步,即 $d(a_1(j),a_2(j))\geqslant a/m$.

先计算 $a_1(j)$ 与 $a_1(j+1)$($a_2(j)$ 与 $a_2(j+1)$)的差.令 $d(x,y)$ 为 n 维向量 x 与 y 的欧氏距离.令 $t_i(j)$ 是 $P_i(j)$ 的元素个数.

因为 $a_1(j+1)=[t_1(j)a_1(j)-x']/(t_1(j)-1)=a_1(j)+(a_1(j)-x')/(t_1(j)-1)$,所以

$$d^2(a_1(j+1),a_1(j))=d^2((a_1(j)-x')/(t_1(j)-1),0)\geqslant\frac{a^2}{4m^4}\tag{8}$$

这里,我们利用了不等式:$d(a_1(j),x')\geqslant a/2m$.

因为 $a_1(j)$是 $P_1(j)$的重心,而 x'不属于 $P_1(j)$,设 $x'\in P_2(j)$,若 $d(a_1(j),x')<a$,则有 $d(a_2(j),x')<d(a_1(j),x')<a$. 由三角不等式,有:$a/m\leqslant d(a_1(j),a_2(j))\leqslant d(a_1(j),x')+d(a_2(j),x')<a/2m-a/2m=a/m$,矛盾! 故得 $d(a_1(j),x')\geqslant a$.

同理可得

$$d^2(a_2(j+1),a_2(j))\geqslant\frac{a^2}{4m^4}\tag{9}$$

若执行上述中(4.2)步,计算对应的目标函数值,则 $P_1(j+1)$中点的目标函数值=零,在 $P_2(j+1)$中点的目标函数值估计如下:

原 $P_2(j)$中的点其值不变,原 $P_1(j)/\{x'\}$中的点有

$$\sum_{x\in P_1(j+1)/\{x'\}}d^2(x,a_2(j+1))\leqslant\sum_{x\in P_1(j+1)/\{x'\}}[d(x,a_1(j+1))+d(a_1(j+1),a_2(j+1))]^2$$

故目标函数在 $x\in P_1(j+1)/\{x'\}$上值之差为

$$\left|\sum_{x\in P_1(j+1)/\{x'\}}d^2(x,a_2(j-1))-\sum_{x\in P_1(j+1)/\{x'\}}d^2(x,a_1(j+1))\right|$$

$$\leqslant\prod_{x\in P_1(j+1)/\{x'\}}\{2|d(x,a_1(j+1))*d(a_1(j+1),a_2(j+1))|+d^2(a_1(j+1),a_2(j+1))\}$$

$$\leqslant(m/2-1)\{2ba/m+(a/m)^2\}<ba-2ba/m+a^2/2m\tag{10}$$

另一方面,$|d^2(x',a_1(j))-d^2(x',x')|=b^2$,

所以,目标函数值总的下降值$\geqslant b^2-(ba-2ba/m+a^2/2m)\geqslant a^2/m$ (11)

注:令 $d^2(x',a_1(j))=b$,在式(10)的不等式估计中用到:当 $x\in P_1(j+1)/\{x'\}$时,有 $d(x,a_1(j+1))\leqslant b$;$|P_1(j+1)/\{x'\}|=t_1(j)-1\leqslant m/2-1$ 以及 $b\geqslant a$.

令
$$l(t)=\sum_{j=1}^{h}\{\sum_{i\in P_j(t+1)}d^2(x_i,a_j(t))\},\qquad L(t)=\sum_{j=1}^{h}\{\sum_{i\in P_j(t)}d^2(x_i,a_j(t+1)\}.$$

下面证明 $l(t)\geqslant l(t+1)$.

由于
$$l(t)=\sum_{j=1}^{h}\{\sum_{i\in P_j(t+1)}(x_i-a_j(t+1))+(a_j(t+1)-a_j(t))^T\cdot(x_i-a_j(t+1))+(a_j(t+1)-a_j(t))\}$$

$$=\sum_{j=1}^{h}\sum_{i\in P_j(t+1)}d^2(x_i,a_j(t+1))+\sum_{j=1}^{h}d^2(a_j(t+1),a_j(t))+$$

$$2\sum_{j=1}^{h}\{\sum_{i\in P_j(t+1)}(x_i-a_j(t+1))^T(a_j(t+1)-a_j(t))\}\tag{12}$$

因为 $a_j(t+1)$是 $P_j(t+1)$的重心,故有 $\sum_{i\in P_j(t+1)}(x_i-a_j(t+1))=0$. 得式(12)的第3项等于零;而其第2项大等于零,于是得

$$l(t)\geqslant L(t)\tag{13}$$

成立.

现在来估计式(12)第2项的值.

第2项为:$\sum_{j=1}^{h}d^2(a_j(t+1),a_j(t))$.

由串行算法的规定得,和式中除前两项外均为零,而前两项由式(8)(9)得其和$\geqslant\frac{a^2}{2m^4}$.

又由 $l(t+1)$和 $L(t)$的定义知,$l(t+1)$是"关于 $A(t+1)$按'就近原则'进行的分类"后,再求其目标函数,而 $L(t)$是按分类 $P_i(t)$中的点与 $a_i(t)$的距离的平方和之和,故有

$$L(t)\geqslant l(t+1)\tag{14}$$

由式(13)(14)得,$l(t)>l(t+1)$. 若执行上述中第(4.1)步,则每迭代一轮,目标函数下降 $O(\frac{a^2}{m^4})$.

若执行上述中第(4.2)步,由公式(13)直接得,每迭代一轮目标函数值下降 a^2/m.

又由于 $l(0)$是有限值,$l(t)\geqslant 0$,故得至多进行 $O(m^4)$步,算法必收敛于某个稳定状态. □

注 1:串行算法,一般只能求到局部极小,未必能求到全局极小.

反例:设在平面上有 4 点:$a_1=(2,1);a_2=(-2,1);a_3=(-2,-1);a_4=(2,-1)$.

若取 $a_1(0)=a_1,a_2(0)=a_4$,得 $P_1(1)=\{a_1,a_2\},P_2(1)=\{a_3,a_4\}$.

$a_1(1)=(0,1);a_2(1)=(0,-1)$. 得 $P_1(2)=\{a_1,a_2\}=P_1(1);P_2(2)=\{a_3,a_4\}=P_2(1)$. 得解为 $A(1)$和 $P(1)$. 其目标函数值=4+4+4+4=16.

若取 $a_1(0)=a_1,a_2(0)=a_2$,得 $P_1(1)=\{a_1,a_4\},P_2(1)=\{a_2,a_3\}$.

$a_1(1)=(2,0),a_2(0)=(-2,0)$. 得 $P_1(2)=\{a_1,a_4\}=P_1(1),P_2(2)=\{a_2,a_3\}=P_2(1)$. 得其解为 $A(1)$和 $P(1)$. 其目标函数值=1+1+1+1=4.

可见前者不是全局最小值. 这个例子说明,是否求到全局最小值与起始集合的取法很有关系. 故通常我们可以利用先验知识帮助我们适当地选取初始值. 当无先验知识可以利用时,我们可取 $A(0)$的点比较均匀地分布在 P 中.

注 2:上面我们对串行的移动中心聚类算法的复杂性的上界进行了估计,从直观上看,似乎并行的移动中心聚类算法将比串行的算法来得更有效,可是我们一时还未能给出其复杂性上界的估计.

2.3 FP 统计附加样本学习算法

设给定一输入样本集 P,设对应有 k 个不同的输出.

(1) 将先验知识化成相应的附加样本(先验附加样本)加入原来样本集中,加入后的样本输入集合仍记为 P.

(2) 现按其对应的不同输出,将 P 分成 k 个子集合 $P(1),P(2),\ldots,P(k)$.

(3) 对每个 $P(t)$用移动中心法,求其对应的统计附加样本集 $A(t)$.

(4) 将 $A(t)$中的点当作"附加样本"加到 $P(t)$中去,得到的集合记为 $P^*(t)$.

(5) 对每个 $P^*(t)$,做对应的领域集合,记为 $D^*(t)$.

(6) 对每个 $D^*(t)$求其最小覆盖集合,记为 $D'(t)$.

注:求最小覆盖可以有如下几种提法:

A. 只要求每个 $D'(t)$盖住 $P(t)$,而不要求盖住 $A(t)$;

B. 给定一个常数 a,只要求 $P(t)$中被 $D'(t)$覆盖的点多于 $a\%$(称之为精度为 $a\%$的覆盖).

(7) 对 $D'(t)$中的每一个领域,按 FP 算法作出对应的神经元 $A_i,i=1,\ldots,s_t$.

(8) 将所有求到的神经元 A_i,放在第 1 层.

(9) 输出层取 n 个神经元,并按 FP 算法构造其对应的权和阈值.

这样我们就得到所要求的网络.

注:(6. A)相当于只利用统计附加样本来降低隐层的元件个数. (6. B)相当于认为每个 $D'(t)$中附加领域(附加样本对应的领域)的中心 x_i 是标准输入,而且不要求覆盖住所有的样本输入点(其物理意义见下文).

原来神经网络的学习问题均假设给定的样本集中的样本都是正确的,但在实际中由于误差等原因,所给的样本其中可能有一小部分是错误的,这时我们当然希望所给的算法能对所给的样本集中的信息自动进行"去伪存真",同时也希望算法能对所学的东西不断进行总结提高,去掉多余的、非本质的信息,并随变化的环境不断地改变自身的状态. 这就是所谓自适应和自学习能力.

下面我们将证明,上面给出的算法具有一定的自适应和自学习的能力.

设对某个子集 $P(t)$,用移动中心法求到统计附加样本集 $A(t)=\{\bar{x}^1,\bar{x}^2,\ldots,\bar{x}^{s_t}\}$,$A(t)$中第 j 个子类的均值为 $\bar{x}^j$,这时,相当于把 $\bar{x}^j$ 看作是第 j 个子类的标准输入,其他的点看成是由 $\bar{x}^j$ 因噪音或其他误差引起变化而得来的,当这些点与标准输入的点误差过大时,理应将它们去掉. 这也就是我们上面提出只要求 $D'(t)$覆盖住 $P(t)$的大部分,而允许有若干 $P(t)$的点(少于$(100-a)\%$)不被覆盖. 这些不被覆盖的点,多处在离标准输入较远的地方,正是这种"不被覆盖"使我们能将那些误差较大的点删去,从而达到"去伪存真"的目的. 另一方面,通过求 $A(t)$集合,就相当于对集合 $P(t)$进行了"归纳提炼"(进行统计处理). 这样,从结构上来看,每进行一轮"附加样本"的学习,就使结构更加精炼(减少隐层元的个数);其次,从样本集的性能来看,每进行一轮"附加样本"的学习(包括进行求最小覆盖算法)后,就留下更有代表性的样本(即进行了知识更新). 当环境发生变化时,通过不断地进行"附加样本"学习,就能通过不断地更新知识和网络结构来适应变化的环境. 以上定性的想法,人们早就有过,本文的贡献在于:用复杂性为 $O(n^4)$的算法精确地体现了这种想法.

人类在学习过程中也有类似的情况. 开始时对新的事物,总是先单纯记忆,记忆到一定数量时,就开始对所学到的知识进行总结、整理、归纳,使之系统化(去掉错误的、多余的信息,留下有代表性的、有规律性的信息),并在变化的过程中不断地修改这些信息.

其他的神经网络学习方法(如 BP 算法等)由于学习速度极慢,故无法从变化的环境中进行实时的学习;而 FP 算

法由于学习速度极快(其学习过程基本是一个直接输入数据的过程),在这样快速学习算法的基础上才有可能提供出一个与变化的环境进行实时交互的学习方法.我们上面提出附加样本的FP算法给出了这种可能性(关于这个问题的详细讨论见另文).

3 实　例

例1:设输入向量是三维(分量取{1,−1}),输入向量 x 的分量中为"1"的个数 $h(x)$.做输出向量 y(三维):

$$y(x)=e^i,\quad \text{其中 } i=h(x)(\text{mod } 3)$$

$$e^0=(1,-1,-1)\qquad e^1=(-1,1,-1)\qquad e^2=(-1,-1,1)$$

这个例子可以看成是XOR问题的推广,因为XOR可看成是二维的(mod 2)问题,上面的例子是三维的(mod 3)问题.

解:将输入按对应的输出分类为:$P(0)=\{(1,-1,-1),(-1,1,-1),(-1,-1,1)\}$,$P(1)=\{(1,1,-1),(-1,1,1),(1,-1,1)\}$,$P(2)=\{(-1,-1,-1),(1,1,1)\}$

先求 $P(1)$的统计附加样本得:$\bar{x}^0=(1,1,1)$(已标准化),令 $x^1=(1,1,-1)$,$x^2=(-1,1,1)$,$x^3=(1,-1,1)$,将 $\bar{x}^1$ 加入 $P(1)$.得 $P'(1)=\{\bar{x}^0,x^1,x^2,x^3\}$.然后按附加样本算法(按6.B进行),求作 $P'(1)$的覆盖(只要求不覆盖 $P(0)$中的点),得对应神经元 A_1,其阈值为0,权向量$=\bar{x}^0$.

同理,再求 $P(2)$的覆盖(要求不能覆盖 $P(0)$和 $P(1)$的点),得元件 A_2 的权向量$=(1,1,1)$,阈值$=2$;最后求覆盖 $P(0)$的元件,得 A_3 其权向量$=(-1,-1,-1)$,阈值$=2$.

直接计算得:$P(0)$中的点经第1层的变换后,均变为(0,0,0)点.$P(1)$中的点经第1层的变换后,均变为(1,0,0)点.$P(2)$中的点经第1层的变换后,变成为(1,1,0)或(0,0,1)点.

最后按FP算法的公式(3)(4)得:输出层的各元件的权和阈值如下:

元件 B_1 权$=(-1,-1,-1)$,$\xi_1=-1/2$;　元件 B_2 权$=(1,-1,-1)$,$\xi_2=1/2$;　元件 B_3 权$=(-1,2,1)$,$\xi_3=1/2$.

于是经输出层后,$P(0)$中的点被映为 Y 空间的(1,−1,−1)点;$P(1)$中的点被映为 Y 空间的(−1,1,−1)点;$P(2)$中的点被映为 Y 空间的(−1,−1,1)点.

若这个网络直接用FP算法求解,其隐层需要7个元件,现改用附加样本的FP算法,所得到的网络其隐层只需3个元件.比原先的少.

可以证明,用同样方法求解 n 维(mod 3)问题,只需 n 个隐层元就能构造出所需的网络,若直接用FP算法求,需 2^n 个隐层元.

参考文献

1 张铃,吴福朝,张钹等.多层前馈神经网络的学习和综合算法.软件学报,1995,6(7):440～448
(Zhang Ling, Wu Fu-chao, Zhang Bo *et al.* A learning and sythesis algorithm of multilayered feed forward neural networks. Journal of Software, 1995,6(7):440～448)

2 胡国定,张润楚.多元数据分析方法——纯代数处理.天津:南开大学出版社,1990
(Hu Guo-ding, Zhang Run-chu. Methods of multivariate data analysis. Tianjin: Nankai University Publisher, 1990)

A New Learning Algorithm Based on Technique Appended Samples

ZHANG Ling[1,3]　ZHANG Bo[2,3]

[1](*Institute of Artificial Intelligence　Anhui University　Hefei*　230039)

[2](*Department of Computer Science and Technology　Tsinghua University　Beijing*　100084)

[3](*State Key Laboratory of Intelligent Technology and Systems　Tsinghua University　Beijing*　100084)

Abstract　A new learning algorithm of neural network, called technique of appended samples, is proposed in this paper, which well combines a priori knowledge with the training samples knowledge to design a neural network. Its complexity is just polynomial ($O(n^4)$). This new method can be used in designing a better performance of neural network. This article is structured as follows: the FP algorithm and the cover algorithm based on FP algorithm are introduced in the section 1. In the next section, the FP statistical technique of appended samples is provided. Finally, a design example is given to illustrate the efficiency of new method.

Key words　Neural network, FP algorithm, technique of appended samples, minimum covering.