Journal of Computers, vol.11.

1988 年 12 月　　计 算 机 学 报　　第 12 期

SLONN: 一个神经元网络的模拟语言及其实现

许 卓 群

(北京大学计算机系)

汪 德 亮

(中国科学院计算技术研究所)

SLONN: A SIMULATION LANGUAGE FOR NEURAL NETS AND ITS IMPLEMENTATION

Xu Zhuoqun

(Peking University)

Wang Deliang

(Institute of Computing Technology, Academia Sinica)

Abstract We have designed and implemented a general purpose simulation language for neural nets (SLONN). By introducing forks to describe connection patterns between neurons and using repetition connection, module type and module arrays to specify large networks, SLONN can be used to specify both small and large neuronal networks conveniently, and to perform simulation experiments effectively. The languange provides an effective tool for neuronal modeling on computers.

摘要 在详细地研究了生物神经元网络的一般性质之后，本文提出和实现了一个通用的神经元网络的描述语言 SLONN. 通过设置参数、引入分叉来描述神经元的连接方式，并且利用重复连接、模块类型和模块数组等说明，用以描述大网络，从而使 SLONN 语言对大、小各类神经元网络都能方便有效地给出描述，并且能对网络进行模拟实验. SLONN 语言为在计算机上进行神经元网络模拟以及人工智能的研究提供了有力的工具.

一、概　述

由神经元所连结成的神经元网络是脑的基本结构，是智能行为的基础. 由于对稍大

本文 1987 年 8 月 27 日收到.

一些的神经元网络直接进行生物学实验的困难，同时也由于人工智能的需要，使得与脑的实验研究的同时，神经元网络的控制论方面的研究也开展了起来.

近年来神经元网络的控制论方面的研究发展是迅速的. 从小神经元网络理论、各种感受器的信息加工模型到小脑模型等在不同动物对象上都有大量的研究工作在进行. 尽管这些研究所表现的神经元网络的行为多种多样，模拟者的目的也各不相同，但它们共同的结构基础是一个由神经元所构成的神经元网络，只不过网络的规模和组织方式不同.

本文提出一个高级的神经元网络的模拟语言 SLONN. 这种通用的高级语言作为工具可以减轻在计算机上建立神经元网络并取得各种模拟实验数据的工作量，使研究者能够专心于研究有效的逻辑网络. 此外，有一个这样的工具也使得从事神经模拟的专家们相互交流更为容易.

本文是"一种用于计算机模拟的神经元模型"论文的续篇. 有关神经元模型的详尽描述将另文描述.

二、神经元网络的描述

1. 神经元描述

SLONN 语言为单个神经元提供两种描述方法:

(1) 提供标准神经元类型 (记之为 **neur**). 所谓标准神经元是指神经元的所有参数均由系统自动赋以缺省值.

(2) 用户定义的神经元类型(冠以名 **neuron**). 用户可以定义神经元的类型名并指定特定的参数值. 例:

```
neuron
  ptype {E = 20}                | ptype 是静息电位 E 为 20mv 的神经
    ... ...                     | 元类型
ptype pear                      | pear 为具有 ptype 类型的一个神经元
```

在词 **neuron** 之后，用户说明自己定义的神经元类型. 在随后的网络描述中，网络中的神经元都要用相应的类型加以说明.

2. 分叉及重复连接

图 1 是一简单的神经元网络. 圆圈内标明的是神经元名，小三角形标识突触，旁边的数字为连接强度，弧线表示神经纤维连接.

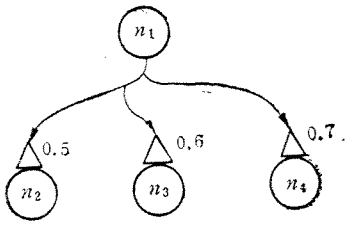


图 1 神经元网络举例

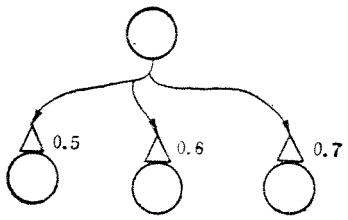


图 2 triple 的连接方式

在 SLONN 中，定义神经元网络分成两步．第一步是定义连接方式，这是通过 **fork**（分叉）来描述的．在连接方式：

fork 3(**to** 0.5, 0.6, 0.7): triple;

中，紧接着 **fork** 的数字 3 表示三股分叉；**to** 表示连接方向，其相反方向 **from**．**triple** 是连接方式类型名，其所形成的连接方式如图 2 所示．

第二步将神经元名与已定义的 **fork** 类型 triple 相联系，从而形成神经元网络．如：

triple (n_1; n_2, n_3, n_4);

其语义是：n_1 为发出神经元；n_2, n_3, n_4 为接受神经元；n_1 到 n_2, n_3, n_4 的连接强度（即突触强度）分别为 0.5, 0.6, 0.7（由 triple 连接方式所指定）；发出神经元和接受神经元之间用分号隔开．这样就描述了图 1 的网络．

triple 作为类型可多次使用．例如：triple (g_1; g_2, g_3, g_4) 则描述图 3 的网络．

如果我们想用相同的分叉来进行多次连接，我们就要重复地使用此分叉．为方便这一目的，SLONN 提供重复连接语句 **for** 语句．

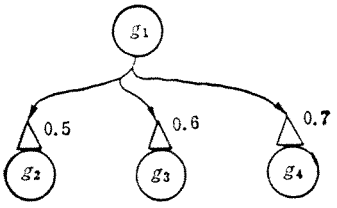


图 3　triple (g_1; g_2, g_3, g_4)

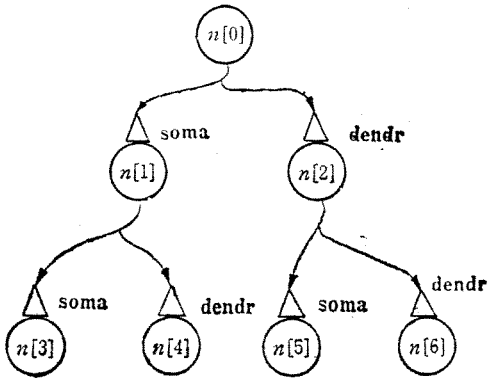


图 4　一神经元网络

例如，图 4 的网络可用以下程序段来描述．

```
integer i;
neur n[7];
fork 2(to soma, dendr): beta;
i = (0 for 2)
  beta (n[i]; n[2i + 1], n[2i + 2]);
```

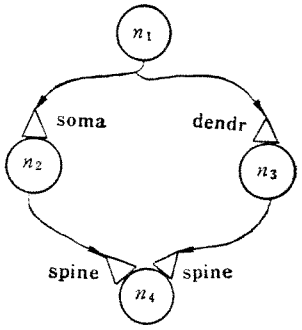
i 为整数
n[7] 为数组说明，它说明了从 n[0] 到 n[6] 的 7 个神经元．

重复语句，即 **for** 语句，表示 beta 要使用三次，i 依次取值为 0, 1, 2．重复语句为规则的连接网络提供了有效的描述工具．在上述 **fork** 语句中的 **soma** 和 **dendr**，以及下面将要出现的 **spine** 都是系统标准连接强度，分别表示轴-体、轴-树、轴-棘连接（体指神经元胞体、树指树突、棘指侧棘）．

3. 模块类型

在网络定义中当需要重复使用某子网络时，可以将该子网络抽象为模块（**module**）类型．

图 5 的子网络是一个要多次出现的结构，我们就可将其抽象成模块，写法如下：




```
module spade
{
  neur n1, n2, n3, n4;
  fork 2(to soma, dendr): alpha;
  fork 1(to spine): omega;
  inner
    alpha (n1; n2, n3);
    omega (n2; n4);
    omega (n3; n4);
}
```

图 5 spade 模块类型

名 spade 为模块类型名，它由四个神经元构成。在 **inner** 下所说明的是属于模块内部的连接.

如果用户有三个 spade 模块类型的子网络 snet1, snet2 和 snet3，则可说明如下：

```
spade snet1, snet2, snet3;
```

模块类型可以很自然地表述类似皮质柱这样的神经系统基本结构[2].

4. 模块数组

模块数组是用数组方式组织的一组模块. SLONN 在处理模块数组时有一些特殊的做法，从而使模块数组成为大神经元网络的有效的说明手段. 这些做法是：

1°. 数组内各模块之间的连接(外部连接)不需要用户说明.

2°. 系统在完成连接时，自动删去那些未指向任何神经元的弧线.

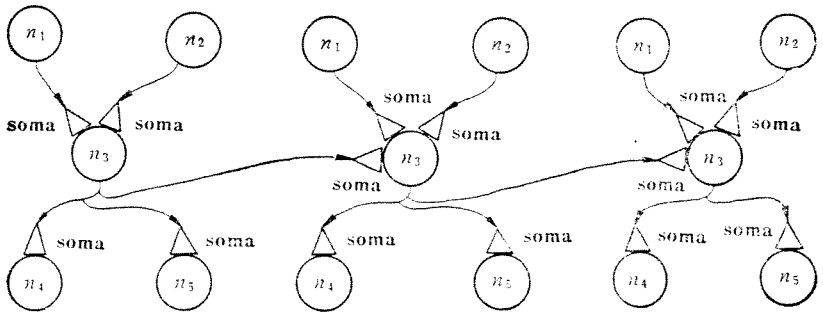


图 6 network[3]

例如，图 6 的网络说明：

```
module star
{
  neur n1, n2, n3, n4, n5;
  fork 1(to soma): alpha;
  fork 2(to soma): beta;
  right
```

本句是 **fork** 2 (**to soma**, **soma**) : beta 的略写形式.

```
    alpha (n3; n4);
  inner
    alpha (n1; n3);
    alpha (n2; n3);
    beta (n3; n4, n5);
}
  ...    ...
star network [3];
  ...    ...
```

其中 **right** 表示模块之间的外部连接方向，即向右. 除 **right** 外，SLONN 还提供了 **left**, **front**, **hind**, **above**, **below** 等连结方向，它们分别为向左、前、后、上、下. 对于一维数组，相应的外部连接方向有：**right**, **left**；对于二维数组，相应的连接方向有：**right**, **left**, **front**, **hind**；对于三维数组，相应的连接方向有：**right**, **left**, **front**, **above**, **below**. network [3] 就是由三个 star 型模块构成的一维模块数组.

在 SLONN 中，模块数组的维数小于或等于三维.

5. 输入描述

对于神经元网络的输入，我们用感受神经元代表各种具体的感受器. 感受神经元是指网络中直接接受外部刺激的那些神经元. 至于如何将外部刺激(如光信号)转换成具有一定频率特性的脉冲发放序列，可由用户指定或编程来模拟.

感受神经元的脉冲发放序列由 0, 1 序列来表征，0 表示无发放脉冲，1 表示有发放脉冲. 按系统规定的节拍指定发放序列有两种方法：

1°. 由用户直接定义输入模式. 例如：

```
neur neu;
string input;                    input 代表一串刺激.
input = {0 0 1 1 1 0 0 1}
stimulate (neu ← input);         此为刺激语句.
```

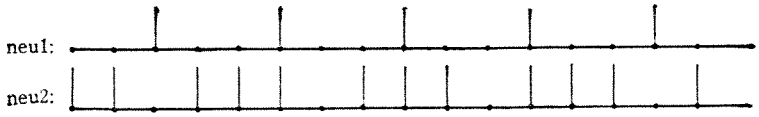
刺激语句将串 input 赋给感受神经元 neu. 按系统节拍，neu 的发放序列可形象地表为：

最小间隔时间由系统规定

再举一例，

```
neur neu1, neu2;
string schema1, schema2;
schema1 = {0 0 1};
schema2 = {1 1 0 1};
stimulate (neu1 ← schema1: 5; neu2 ← schema 2: 4);
```

则将串 schema1 重复 5 次赋给 neu1；串 schema2 重复 4 次赋给 neu2. 在 16 个节拍内，neu1 和 neu2 的发放序列分别为：



2°. 通过函数定义发放模式.

有时，用户需要对脉冲到达的间隔时间规定满足一定函数分布. 这广泛地用于模拟自发放电现象、各种感受器对外部刺激的转换等. 在 SLONN 中，提供了一组随机发放序列的函数定义. 它们有均匀分布、指数分布 (Poisson 过程)、正态分布、β 分布等. 例如:

```
neur n1,
string input;
input = poisson (0.25);
stimulate (n1 ← input);
```

上段描述表示，感受神经元 n_1 的发放序列是到达率为 0.25 (即平均每四个系统节拍有一次发放)的 poisson 分布.

三、神经元网络上的实验操作

1. 运行启动. 使网络中每个神经元都按计算模型动作起来. 网络中的神经元应是并行工作的. SLONN 语言的网络运行调度算法(见第四段实现方法)，保证在每个系统节拍内，所有的神经元均动作一次，而每个系统节拍则模拟了神经元连续发放两脉冲的间隔时间(如 2 毫秒). 在计算机上实际执行时，系统节拍时间将依赖于网络的规模.

2. 结果观察. 用户可以在终端上观察一组神经元的发放序列 (0, 1 串)，并可在绘图仪上将神经元的发放序列连同其膜电位变化一齐绘制出来.

3. 网络内部探测. 使用特定语句可把神经元网络显示及绘制出来. 用户可以指定其中的某些神经元，并利用探测语句获得下列有关的信息: 该神经元突触上的突触强度、记忆突触上的当前记忆值，神经元的其它参数及发放频率等.

4. 时间跳跃. 用于加快对长时记忆的模拟. 为此 SLONN 语言引入时间跳跃语句，神经元网络将按照给出的跳跃时间，模拟遗忘机制，改变记忆神经元突触上的记忆值(这时的神经元网络是处于安静状态).

5. 多次运行. 提供两种实验准备语句. Reset 语句使上次运行已改变状态的神经元网络恢复到最初运行前状态. Initiate 语句则保留上次运行的网络状态，同时为新的输入发放序列进行下一次运行作好准备.

一般而言，对网络的实验操作可提供两种操作方式. 一是程序方式，即将实验操作编入程序内；另一种则是命令方式，用户使用一组实验操作命令干预网络运行，但并不编入程序内. SLONN 系统目前只提供程序方式.

下面我们给出一个已调试通过了的例子(节选)，此程序用以模拟海兔的行为.

```
net
{
    neur skin, sen1, sen2, intn;
    neur fint1, fint2, head, mov1, mov2, jaw;
    fork 2 (to 0.55): branch;
      ...    ...
    branch (skin; sen1, sen2);
      ...    ...
}
begin
    string touch, train;
    touch = {0 0 1 1 1 0 0};
    train = {0 0 1 0 1 1 1};
          /* expr1: a jaw-shrinking reflex */
    stimulate (skin ← touch: 12; head ← {0});
    display (4; jaw);    /* this statement indicates displaying the result,
                            and number 4 means from plotter */
    simulate (80);       /* this statement initiate simulating process. */
        /* expr2: training for short-term memory */
    stimulate (skin ← train: 9; head ← {0});
    simulate (63);
    stimulate (skin ← touch: 12; head ← {0});
    display (4; jaw);
    simulate (80);
      ...    ...
end
```

四、实现方法

SLONN 语言的解释执行系统是在 UNIX 操作系统下用C语言写的. 语言的语法检查,包括词法分析及语法分析,则利用了 UNIX 工具 LEX 及 YACC.

在网络模拟执行过程中,系统的解释执行器在每一个系统节拍要扫视所有的神经元,并对每一神经元进行一次计算. 值得注意的是, 对各神经元的扫视顺序会对运行结果产生一定影响,例如图 7 所示网络,由 10 个神经元互连而成(为便于讨论,网络已画为有向图形式),n_1, n_2 为感受神经元. 很明显, n_1, n_2 谁先计算是重要的: 在计算 n_1 时依赖于 n_2 在同一系统节拍的输出,反之则不然.

为了合理地安排扫视顺序, 需形成神经元网络的运行顺序表 exlist. 利用下列形成算法,并使用神经元信息表、感受神经元表及网络互连信息,可产生出 exlist 表.

表 exlist 的形成算法梗概如下.

1°. 按照离感受神经元的距离(称为输入距),将各神经元从小到大排序. 所谓神经元

A 到 B 的距离指 A 到 B 所有路径中最短距离，即有向边最小路径的边数.

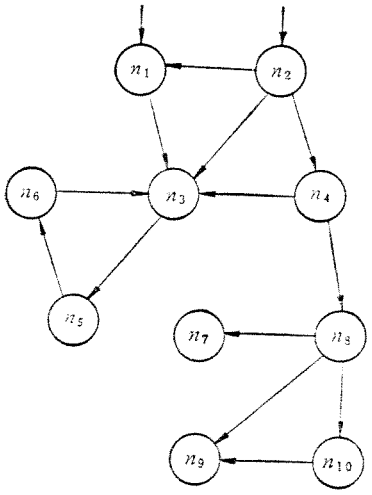


图 7 一神经元网络有向图

2°. 计算导入度. 定义一个神经元 N 的导入度为满足下列条件的所有神经元（称为导入神经元）的个数：有指向 N 的边，且其自身的输入距不小于神经元 N 的输入距.

3°. 将输入距相等的各神经元按导入度的大小排序（从小到大），如果导入度也相同，就按自然顺序，从而形成 exlist.

上图的网络经此算法计算后所形成的 exlist 为：$n_2n_1n_4n_3n_8n_5n_7n_{10}n_6n_9$.

调度算法的目的是，在不考虑神经元之间脉冲传送时间延迟的情况下，计算一个神经元前要尽可能使所有导入神经元在本节拍内的输出已形成. 由于容许神经元网络中有环存在，故不存在一个算法能使在计算任一神经元时，其各导入神经元的输出都已产生. 目前的 exlist 算法只是一种实用算法.

运行顺序对网络的整体行为是否有显著影响，这个问题有待深入研究. 根据我们的观察，这种影响是可以忽略的.

在 SLONN 语言书写的程序中，simulate 语句是启动网络运行的语句. 系统执行该语句时，先利用 exlist 的形成算法以产生 exlist 表（在 simulate 语句前系统已形成此神经元网络的各种信息表）. 然后在每一系统节拍，按 exlist 表顺序依次计算各神经元，计算其时间总和、空间总和以及阈值函数，并根据程序对互连的描述，产生有关神经元的输出-输入传输. simulate 语句将重复多个系统节拍，其节拍重复数由该语句中给出的节拍次数决定.

五、结 语

从语言的角度来说，SLONN 是一种描述网络的语言，只不过描述的是一种特殊的网络——神经元网络. 与描述电路网络的 SPICE 语言[3]，描述进程通讯的 OCCAM 语言[4] 以及描述大神经元网络的 BOSS 系统[5,6] 等相比较就可以发现，SLONN 有两个显著的特点：

1. SLONN 引入分叉来表示连接方式，能自然地反映神经元网络中的分散型连接（即一个神经元发出神经纤维连接到多个神经元上）和集中型连接（即一个神经元接受来自多个神经元纤维的传入）这两种主要的连接方式. 从而赋于 SLONN 程序以很好的可读性.

2. SLONN 中有重复连接、模块和模块数组等三种描述大网络的手段，使得 SLONN 具有很高的网络描述能力.

SLONN 作为高级程序设计语言对于大、小神经元网络均是一种有效的描述工具.

作为应用的例子，我们用 SLONN 描述了海生蜗牛和海兔的部分神经系统. 计算机模拟实验的结果表明，所描述的网络对海生蜗牛的趋光性与条件化学习、海兔的习惯化和敏感化学习等行为均能作出很好的模拟. 有关模拟实验及 SLONN 语言中的神经元模型将在另文中给出.

参　考　文　献

[1] J. F. Stein, An Introduction to Neurophysiology. Blackwell Scientific Publication, Oxford, 1982.

[2] A. Vladimivescu, *et al.*, SPICE User's Guide, 1981 (中译本：SPICE 通用电路模拟程序用户指南，清华大学出版社，北京，1983).

[3] INMOS, Programming Manual in Occam. INMOS limited, Bristol, 1983.

[4] L. D Wittie, Large-scale simulation of brain cortices, Simulation, 31(1978), 73—78.

[5] L. D. Wittie, Large network models using the Brain Organigation Simulation Sistem (BOSS), Simulation, 31 (1978), 117—122.