章节修订历史

本表格记录修订版本之间的重大改动。类似简单说明或者变更格式这样的细微修改并不会加以记录。

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 改动日期 | 改动内容 | 负责人 |
| 0.4 |  | Initial release | S. Ahmad |
| 0.5 | Feb 2017 | Update to current algorithms | M. Taylor & Y. Cui |

给读者的重要说明

下文给出了包括伪代码和参数在内的空间池算法的详细信息。我们强烈建议您先了解其他可用的空间池资源，以便理解HTM中空间沉积池的高级概念以及生物学映射。您可以在空间沉积池一章中找到最新的空间沉积池资源的链接。

空间沉积池算法细节

我们首先介绍一些重要术语，然后是高度抽象的算法步骤，最后是包括伪代码在内的算法详情。

术语

* (Column)柱：一个HTM区域由包括一些细胞的很多柱状区域组成。SP在柱状区域的层次运行，一组细胞构成一个柱状区域作为一个整体参与计算。
* (Mini-column)迷你柱：见“柱”
* (Inhibition)抑制：维持神经元稀疏激活的机制。在SP中，这表现为一个圆柱区域阻止附近的其它的圆柱区域变得活跃。
* （Inhibition Radius）抑制半径：一个柱的局部邻域的大小，在该邻域内，各柱可能相互抑制活跃状态。
* (Active duty cycle)活跃占空比：表示一个圆柱的激活次数在整个周期上比例。
* (Overlap duty cycle)覆盖占空比：表示一个圆柱在整个周期上的平均覆盖值。该动态均值需大于等于树突段的激活阈值。
* (Receptive field)接收区域：一个圆柱可能连接到的输入空间。
* (Permanence value)连通值增量：表示空间池算法中的一个迷你柱与其接受域中的一位之间的增长量
* (Permanence threshold)连通值阈值：如果一个突触的持久性高于此值，则认为它是完全连接的。值域为[0,1]。
* (Synapse)突触：圆柱和输入位之间的连接。在空间池算法中，圆柱的树突段上的突触连接到输入空间中的位。突触可以处于以下状态：

连通的：连通值高于阈值。

潜在的：连通值低于阈值

未连接：不能连接。

空间沉积池算法步骤

1. 从一个由固定位数比特组成的输入开始。这些位可能代表感官数据，也可能来自HTM系统的其他区域。
2. 通过为接收此输入的区域指定固定数量的圆柱来初始化HTM区域。每个圆柱都有一个相关的树突段，作为输入空间的连接。每个树突段都有一组潜在的突触，代表输入位的一个（随机）子集。每个潜在的突触都有一个连通值。这些值是围绕连通值阈值随机初始化的。那些连通值大于阈值的突触被认为是连通的。
3. 对于任意给定的输入，确定每圆柱上有多少连通的突触连接到活动的（on）输入位。这些突触被定义为活跃的。
4. 活动突触的数量乘以一个“增强”因子，该因子动态地由一个突触所在圆柱相对于其相邻圆柱的活动频率决定。
5. 抑制半径内激活率最高（增强后）的一小部分圆柱变为激活状态，并禁用半径内的其他圆柱。抑制半径本身是由输入比特的扩散动态确定的。现在获得了一组稀疏的活跃圆柱。
6. 该区域现在使用的空间池（赫布理论）学习规则：对于每个活动圆柱，我们调整所有潜在突触的连通值。与活动输入位对齐的突触的连通值增加。与非活动输入位对齐的突触的连通值降低。对连通值的改变可能会改变某些突触的连通状态，从连接到不连接，或反之。
7. 对于后续输入，从步骤3开始重复。

空间池伪代码

本节包含空间沉积池的详细伪代码，分为四个阶段：初始化、覆盖计算、抑制和学习。在初始化（阶段1）之后，空间沉积池算法的计算例程的每次迭代都会经历三个不同的阶段（阶段2到阶段4），这些阶段依次发生。代码中使用的各种数据结构和支持例程在末尾的表X中定义。

阶段1 初始化空间沉积池参数

在接收输入之前，空间沉积池算法通过计算每个圆柱的初始潜在突触列表来初始化。这包括从输入空间（在圆柱的抑制半径内）选择一组随机输入。每个输入由一个突触表示，并分配一个随机的连通值。随机连通值的选择有两个标准。首先，这些值被选择在连通值阈值上下的一个小范围内，即连通值大于该阈值的突触被认为是连通的。这使得潜在的突触在少量的训练迭代后就可以连通（或断开）。第二，每圆柱在输入区域上都有一个自然中心，连通值对这个中心有一个偏差，它们在中心附近有更高的值。

阶段2 对当前输入计算到每个圆柱区域的覆盖值

对于给定输入，该阶段计算每个圆柱区域对于该输入的覆盖值。该覆盖值为每个圆柱连通的突触和活跃的输入位的重合的数量，再乘以增强因子。

1. for c in columns

2. overlap(c) = 0

3. for s in connectedSynapses(c)

4. overlap(c) = overlap(c) + input(t, s.sourceInput)

5. overlap(c) = overlap(c) \* boost(c)

阶段3 再抑制作用完成后计算活跃圆柱

第三阶段计算哪些圆柱在抑制步骤之后仍然是赢家。LocalAreaDensity是一个参数，用于控制局部抑制区域内活跃圆柱的密度。密度也可以由参数numActiveColumnsPerInharea控制。使用此方法时，LocalAreaDensity参数必须小于0。抑制逻辑将确保在每个局部抑制区域中，最多有numActiveColumnsPerinharea圆柱处于活跃状态。例如， numActiveColumnsPerInharea为10，如果一个圆柱的覆盖值为非零，并且其覆盖值在其抑制半径内的圆柱中排名第10或更高，则该圆柱将成为赢家，即活跃的。

6. for c in columns

7. minLocalActivity = kthScore(neighbors(c), numActiveColumnsPerInhArea)

8. if overlap(c) > stimulusThreshold and

9. overlap(c) ≥ minLocalActivity then

10. activeColumns(t).append(c)

阶段4 更新突触连通值和内部变量

最后一个阶段执行学习，根据需要更新所有突触的连通值，以及增强因子和抑制半径。主要学习规则在第14-20行中实现。对于获胜的圆柱，如果一个突触处于活动状态，它的持久值将增加，否则它将减少；连通值被限制在0和1之间。请注意，非获胜圆柱的突触上的连通值将不会被修改。

第21-27行执行增强。有两个单独的机制来帮助圆柱学习连接。如果一个圆柱与它的邻居相比获胜的次数过少（由activeducycle测量），那么它的增强因子设置为大于1（第22-23行）。如果某圆柱比其相邻圆柱更频繁地处于活动状态，则其增强因子设置为小于1。计算增强因子的函数是一个指数函数，取决于一个圆柱的活跃占空比与其相邻圆柱的平均活跃占空比之间的差异。如果一个圆柱连接的突触与任何输入的覆盖程度不够好（通过覆盖占空比循环测量），它的连通值就会增加（第24-27行）。请注意，一旦关闭学习，Boost（C）同样被冻结。

最后，在第4阶段结束时，重新计算抑制半径（第28行）。

11. for c in activeColumns(t)

12. for s in potentialSynapses(c)

13. if active(s) then

14. s.permanence += synPermActiveInc

15. s.permanence = min(1.0, s.permanence)

16. else

17. s.permanence ­= synPermInactiveDec

18. s.permanence = max(0.0, s.permanence)

19. for c in columns:

20. activeDutyCycle(c) = updateActiveDutyCycle(c)

21. activeDutyCycleNeighbors = mean(activeDutyCycle(neighbors(c))

22. boost(c) = boostFunction(activeDutyCycle(c), activeDutyCycleNeighbors)

23. overlapDutyCycle(c) = updateOverlapDutyCycle(c)

24. if overlapDutyCycle(c) < minDutyCycle(c) then

25. increasePermanences(c, 0.1\*connectedPerm)

26. inhibitionRadius = averageReceptiveFieldSize()

算法参数

空间池算法具有许多影响维度、学习机制和总体性能的参数。这里我们详细讨论一些参数，以及各种变量如何影响空间池。参数的作用在之前的伪代码中已经描述过，下面的表1和表2中可以找到算法参数、数据结构和例程的完整列表。

空间池算法结构

空间池算法参数中的圆柱的数量（columnDimensions）定义了HTM区域的维度。如果我们分配4096个圆柱，则区域是一个由4096圆柱构成的数组。然而，对于视觉系统，相同数量的圆柱可以用作二维数组，因此该区域的柱状结构为64x64。我们还可以指定一个三维拓扑。

我们使用圆柱的数量来控制输出维度，圆柱的输入通过指定空间池算法的输入参数来控制。输入维度（inputDimensions）的指定方式与圆柱维度相同，并且维度数必须匹配。一个圆柱可能连接到的输入空间，即该圆柱的接收字段，由潜在半径（potentialRadius）参数控制。此值将确定圆柱在HTM层中的影响范围。一个小的潜在半径将使一个圆柱的接收字段保持在本地，而一个非常大的潜在半径将使该圆柱覆盖到整个输入空间。

抑制

全局抑制（globalinhibition =True）方法从整个层中选择最活跃的那些圆柱激活。否则局部抑制将从与其相邻的圆柱中选择活跃圆柱。我们往往选择前者，因为它提供了显著的性能提升。如果不使用全局抑制，柱间抑制发生在相邻的圆柱之间。圆柱的邻域是抑制半径（inhibitionRadius）的函数，是空间池算法内部的一个动态计算度量，它是所有圆柱连接的接收字段平均大小的函数。圆柱的接收字段可以部分地由上述潜在半径参数控制；由于空间池算法圆柱（通常是HTM单元）的接收字段是动态的，因此无法显式设置。

但是，我们可以使用LocalAreaDensity或NumActiveColumnsPerInhare指定活动圆柱的密度。在抑制过程中，这些参数将用于计算在局部抑制区域内保留的最大柱数。使用前一个参数可以指定密度，活动圆柱的实际数量将随着圆柱更改其接收字段的大小而更改。使用后一个参数，密度将随着接收字段的变化而波动，但活动圆柱的最大数目保持不变。

学习

学习速率可以通过突触的连通值的增加和减少量来确定——通常前者大于后者。

活跃圆柱

虽然一个圆柱可能在竞争中胜出，但它的覆盖值必须大于一个阈值（stimulusThreshold），才能变得活跃。这里的目的是防止噪声激活圆柱，这有助于实现避免琐碎模式的空间池目标。

在驱动圆柱竞争激活时，增强可能会有所帮助。增强由活跃占空比（activeDutyCycle）和覆盖占空比（overlapDutyCycle）监控。抑制后，如果一个圆柱的活跃占空比低于相邻圆柱的活跃占空比，则其内部增强系数（boost）将增加到1以上。如果一圆柱的活跃占空比高于相邻圆柱的活跃占空比，则其增强系数将降低到1以下，这有助于推动圆柱之间的竞争，并实现使用所有圆柱的空间池目标。在抑制之前，如果一个圆柱的覆盖占空比低于其可接受的最小值（由minpctoverlapDutycycle和相邻圆柱的覆盖占空比的函数动态计算），则其所有突触的连通值都会被增加。一个低于标准的占空比意味着一个圆柱以前学习到的输入不再活跃，或者它们中的绝大多数被其他圆柱所“劫持”。通过在抑制前提高所有突触的连通值作为对一个圆柱占空比过低的补偿，我们可以使一个圆柱搜索新的输入。

参数，数据结构和例程

下表总结了空间池算法的数据结构、例程和参数，包括典型用例的推荐参数设置。

|  |  |
| --- | --- |
| Columns | 圆柱列表 |
| ColumnCount | 空间沉积池算法，一个HTM区域的圆柱的数量。该参数任务相关，推荐至少为2048 |
| Input(t,j) | T时刻该层的输入，第j位输入有效时input(t,j)取1 |
| Overlap(c) | 空间沉积池算法中，对于给定的输入模式，圆柱C的覆盖值 |
| ActiveColumns(t) | 由于自下而上的输入而成为赢家的圆柱索引列表。 |
| numActiveColumnsPerInhArea | 一个参数，用于控制在抑制步骤之后将成为赢家的圆柱数。我们通常将其设置为预期抑制半径的2%。对于2048圆柱和全局抑制，此值设置为40。我们建议最小值为25。 |
| inhibitionRadius | 圆柱的平均连接接收字段大小 |
| neighbors(c) | 圆柱C的抑制半径范围内的所有圆柱的列表 |
| stimulusThreshold | 在抑制步骤中一个圆柱要处于活跃状态的所必须最小有效输入。它应当是编码器的输出的背景噪音的大小，通常设置为非常低的值（0到5）。系统对该参数不太敏感；如果不确定，设置为0。 |
| boost(c) | 圆柱c在学习过程中的增强因子，被用于提升不活跃的圆柱的覆盖值 |
| boostStrength | 一个大于或等于0.0的数字，用于控制增强的强度。如果BoostStrength=0，则不使用增强。 |
| Synapse | 一种表示突触的数据结构，包含一个连通值和源输入索引。 |
| potentialPct | 在一个圆柱的潜在半径内，被初始化为该圆柱潜在突触的输入百分比。这应该设置为在初始化空间池算法时，平均至少连接15-20个输入位。例如，假设一个圆柱的输入通常包含40个位，并且连通值被初始化，这样50%的突触开始连接。在这种情况下，由于40\*0.5\*0.75=15，您希望potentialPct至少为0.75。 |
| connectedPerm | 如果一个突触的连通值大于这个值，就称之为连通的。通常设置为0.2。空间池算法对此参数不敏感。 |
| potentialSynapses(c) | 此圆柱的潜在突触及其连通值列表 |
| connectedSynapses(c) | 潜在突触列表的一个子集，其中连通值大于突触的connectedPerm。这些是当前连接到圆柱c的自下而上的输入 |
| synPermActiveInc | 学习过程中活跃突触的连通值增加量。这个参数有点依赖于数据。（数据中的噪声量将决定synPermActiveInc和synPermInactiveDec之间的最佳比率。）通常设置为较小的值，例如0.03 |
| synPermInactiveDec | 在学习过程中不活跃的突触的连通值减小量。通常设置为小于增量的值，例如0.015。 |
| activeDutyCycle(c) | 一个动态均值，表示圆柱c在抑制后的活跃频率（例如在最近1000次迭代中）。 |
| overlapDutyCycle(c) | 一个动态均值，表示圆柱c与其输入（例如，在最近1000次迭代中）有显著重叠（即大于stimulusThreshold）的频率。 |

表1。空间池伪代码和nupic实现中使用的变量和数据结构。对于参数，我们给出了各种情况下都能很好地工作的建议值

|  |  |
| --- | --- |
| kthScore(cols, k) | 给定圆柱列表，返回覆盖值前k大的圆柱列表 |
| updateActiveDutyCycle(c) | 计算柱C在抑制后活动的频率的动态均值。 |
| updateOverlapDutyCycle(c) | 计算圆柱C的覆盖值超过最小活跃阈值的频率的动态均值。 |
| averageReceptiveFieldSize() | 所有圆柱平均连接到的接收字段大小的半径。圆柱实际连通的接收区域的大小只包括连通的突触（那些连通值大于等于connectedPerm的突触）。这用于确定柱间横向抑制的程度。 |
| maxDutyCycle(cols) | 返回给定圆柱列表中圆柱的最大活跃占空比 |
| active(s) | 突触s的活跃状态。例如一个有效输入连接到了突触s则为ON |
| increasePermanences(c, s) | 用比例因子s增加圆柱C中每个突触的连通值。 |
| boostFunction(c) | 返回圆柱的Boost值。boost值是一个正的标量值。如果活跃占空比小于相邻圆柱的平均活跃占空比，则高于1。如果一个圆柱的活跃占空比高于其相邻圆柱，则小于1。 |

表2。空间池伪代码中使用的例程。它们在nupic代码库中可能有不同的名称。