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

下面的文本给出了时间沉积池算法的细节，包括伪代码和参数。我们强烈建议您首先阅读空间池算法，并访问其他可用的时间沉积池资源，以便了解时间沉积池在HTM中的高级概念和在生物学映射。您可以在“时间沉积池”一章中找到最新的时间沉积池资源的链接。

时间沉积池算法细节

时间沉积池算法有两个功能：学习空间池算法形成的稀疏分布表征（SDR），并进行预测。具体来说，时间沉积池算法：

1）形成稀疏输入的表征，捕获先前输入的时间上下文

2）形成基于先前输入上下文中当前输入的预测

我们将更详细地讨论这些步骤，但首先我们将介绍一些重要的术语和概念。

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。
* 迷你柱：见“柱”。
* 连通值：一个标量值（0.0到1.0），分配给每个突触，以指示连接的连通值。当连接加强时，其连通值增加。在其他条件下，连通值降低。
* 连通值阈值：如果突触的连通值高于此阈值，则视为完全连接。值域为[0,1]。
* 突触：细胞间的连接点。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。

空间池和时间沉积池算法之间共享的概念

二元权重

HTM突触对突触所在细胞只有0或1的影响；它们的“权重”是二元的，不像许多神经网络模型使用0.0到1.0范围内的变量值

树突片段

突触与树突片段相连。树突有近端和远端两种类型。

-近端树突段与前馈输入形成突触。对这类节段上的活跃突触进行线性求和，以确定圆柱的前馈激活。

-远端树突段与层内其它细胞形成突触。每个细胞都有许多远端树突片段。如果远端部分的活跃突触总和超过阈值，则相关细胞进入预测状态。由于每个细胞有多个远端树突片段，因此一个细胞的预测状态是多个阈值检测器的逻辑或操作。

潜在突触

每个树突段都有一个潜在的突触列表。所有潜在的突触都被赋予一个连通值，如果它们的连通值超过连通值阈值，它们就可能成为功能性突触。

学习

空间池和时间池算法中的学习是相似的：在这两种情况下，学习都涉及在细胞之间建立连接或突触。这里描述的时间池算法学习同一层中细胞之间的连接。空间池算法学习前馈输入和圆柱之间的连接。

学习包括增加或减少树突段上潜在突触的连通值。使突触连通值增加或减少的规则类似于“赫布”学习规则。例如，如果突触通向的细胞由于树突段接收高于其阈值的输入而激活，那么该段上突触的连通值将被修改。那些活跃的突触促进了细胞的活跃，因此它们的连通值增加了。不活跃的突触因为不对细胞的活跃状态有所贡献，因此其连通值降低。在空间池和时间池算法中，突触连通值更新的确切条件不同。时间池算法的细节如下所述。

现在我们将讨论特定于时间池算法的概念。

时间沉积池算法概念

时间池算法学习序列并进行预测。在时间池算法中，当一个细胞变为活跃状态时，它与之前活跃的其他细胞形成连接。然后，细胞可以通过观察它们的连接来预测自身何时变为活跃状态。如果所有的细胞都这样做，它们可以共同存储和回忆序列，并且可以预测接下来可能发生的事情。这里没有存储模式序列的中央存储器；相反，记忆分布在各个单元中。由于记忆是分布式的，系统对噪声和误差具有鲁棒性。单个细胞可能会失败，但是通常只能造成很小的影响甚至没有影响。

系数分布表征的特性

值得注意的是，稀疏分布表征（SDR）的一些重要特性被时间池算法所利用。

设想我们有一个假设层，它通过在10000个单元中选择200个活跃的单元来形成表征（任意时刻至少有2%的单元是活跃的）。我们如何记忆和识别200个活跃细胞的特定模式？一个简单的方法是列出我们关心的200个活跃的细胞。如果我们看到同样的200个细胞再次活跃起来，我们就能识别出这个模式。但是，如果我们只列出200个活跃细胞中的20个，而忽略了其他180个，会怎么样？会发生什么？你可能认为只记住20个细胞会导致很多错误，那20个细胞会以许多与200个细胞不同的模式活跃。但事实并非如此。因为模式大而稀疏（在本例中，10000个细胞中活跃的有200个），记住20个活跃细胞几乎和记住所有200个一样好。在实际系统中出错的可能性非常小，由此我们大大减少了内存需求。

HTM层中的细胞利用了这个特性。每一个细胞的树突部分都有一组与层中其他细胞的连接。树突段形成这些连接，作为在某个时间点识别网络状态的一种手段。附近可能有成百上千的活性细胞，但是树突部分只需要连接到其中的15或20个。当树突段看到这些活性细胞中的15个时，就可以在相当程度上确定更大的模式正在发生。这种被称之为“子抽样”的技术贯穿了HTM算法。

每个细胞参与许多不同的分布模式和序列。一个特定的细胞可能是几十或数百个时间转换的一部分。因此，每个细胞都有多个树突片段，而不仅仅是一个。理想情况下，一个细胞对于它想要识别的每种活跃模式都有一个树突片段。实际上，一个树突片段可以学习几个完全不同的模式的连接，并且仍然能够很好地工作。例如，一个段可以学习4个不同模式中的20个连接，总共80个连接。然后，我们设置一个阈值，使树突段在其任意15个连接处于活跃状态时变为活跃状态。这可能导致出错。有可能，通过混合不同模式的一部分，树突达到15个活性连接的阈值。然而，由于表征的稀疏性，这种错误是不太可能的。

现在我们可以看到一个有一二十树突片段和几千个突触的细胞如何识别数百种不同的细胞活跃状态。

一阶与变阶序列及预测

一个圆柱拥有更多或更少细胞的区别是什么？具体来说，如果每个圆柱只有一个细胞，会发生什么？

正如我们在本章后面所展示的，由100个活跃圆柱和每列4个细胞组成的输入的表示可以用4\*100种不同的方式编码。因此，相同的输入可以出现在许多上下文中而不会混淆。例如，如果输入模式表示单词，那么一个层可以记住许多使用相同单词的句子而不会混淆。像“dog”这样的词在不同的上下文中有独特的表征。此功能允许HTM层进行所谓的“变阶序列”预测。变阶序列预测不仅基于当前发生的情况，而且基于不定量的过去上下文。HTM层是可变长度的序列记忆。

如果每各圆柱增加5个细胞，则示例中任何特定输入的可用编码数都将增加到5\*100，远大于4\*100。在实践中，我们发现在大多数情况下，每圆柱使用10到16个细胞就足够了。

但是，使每个圆柱的单元数变小确实会产生很大的差异。

如果我们使每个圆柱仅有一个细胞，我们将失去在表征中包含上下文的能力。不管上下文如何，相同的输入总是产生相同的预测。对于每个圆柱一个单元，HTM层是“一阶”记忆；预测仅基于当前输入。

一阶预测非常适合大脑解决的一类问题：静态空间推理。如前所述，暴露在短暂视觉图像中的人可以识别出物体是什么，即使曝光太短，眼睛无法移动。对听力而言，你总是需要听到一系列的模式来识别它是什么。而视觉方面，你通常需要处理一系列的视觉图像。但在某些情况下，你也可以通过一次曝光识别出一幅图像。

时间和静态识别可能需要不同的推理机制。一个需要识别模式序列，并根据可变长度的上下文进行预测。另一个需要在不使用时间上下文的情况下识别静态空间模式。每个圆柱有多个细胞的HTM层非常适合识别基于时间的序列，而每个圆柱有一个细胞的HTM层非常适合识别空间模式。

时间沉积池算法步骤

1. 在先前输入的上下文中形成输入的表征

时间池算法的输入是由空间池算法形成的稀疏分布表征序列（SDR）。如本书“空间池算法”一章所述，空间池算法形成了一个输入的稀疏分布式表征（SDR）。这个sdr表示接收最多输入的由细胞构成的圆柱区域（图1）。

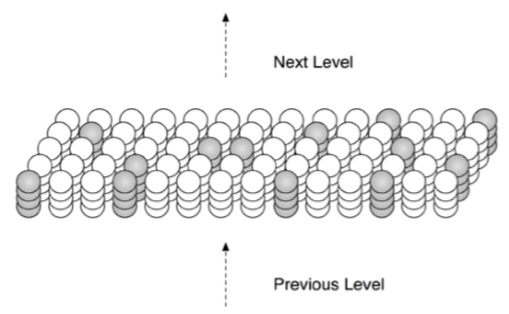


图1描述了空间池算法。HTM层由包括一组细胞的圆柱区域组成。图中只展示了一层的一部分。每个圆柱都从输入的唯一子集接收激活。激活程度最强的柱抑制激活程度较弱的柱。结果是形成了输入的稀疏分布表征。图中显示了浅灰色的活跃圆柱区域。（当没有先前状态时，活跃圆柱区域中的每个细胞都将处于活跃状态，如图所示。）

在空间池之后，时间沉积池算法将输入的圆柱表示转换为一种新的表征，其中包括过去的状态或上下文。新的表征是通过激活每个圆柱中的细胞子集来形成的，通常每个圆柱只有一个细胞（图2）。

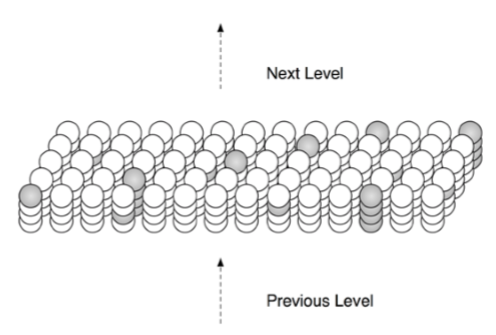


图2通过激活每个圆柱中的细胞子集，HTM层可以在许多不同的上下文中表征相同的输入。圆柱只激活处于预测状态的细胞。没有预测细胞的圆柱将激活圆柱中的所有细胞。图中一些圆柱只有一个细胞处于活跃状态，另外一些圆柱所有细胞都处于活跃状态。

考虑听两个句子，“I ate a pear”和“I have eight pears”。单词“ate”和“eight”是同音词，发音相同。我们可以确定，在大脑的某一点上，神经元对所说的“ate”和“eight”的反应是相同的。毕竟，相同的声音正在进入耳朵。然而，我们也可以确定，在大脑的另一个点上，对这个输入做出反应的神经元在不同的环境中是不同的。当你听到“I ate”和“I have eight”时，声音“ate”的表示会有所不同。想象你已经记住了“I ate a pear”和“I have eight pears”这两个句子。听到“I ate...”会导致与“I have eight...”不同的预测。在听到“I ate”和“I have eight”之后，必须有不同的内部表述。

在不同的上下文中对输入进行不同的编码是感知和动作的通用特性，也是HTM层最重要的功能之一。如何强调这种能力的重要性都不过分。

HTM层中的每一个圆柱都由多个细胞组成。圆柱中的所有细胞都得到相同的前馈输入。圆柱中的每个细胞都可以是活跃的或非活跃的。通过在每个活跃圆柱中选择不同的活跃细胞，我们可以在不同的上下文中表示完全相同的输入。例如，假设每个圆柱有4个细胞，每个输入的表征由100个活跃圆柱组成。如果一次每个圆柱只有一个细胞处于活跃状态，那么我们有4^100种方法来表示完全相同的输入。相同的输入将始终导致相同的100个圆柱处于活跃状态，但在不同的上下文中，这些圆柱中的不同细胞将处于活跃状态。现在我们可以在大量上下文中表示相同的输入，但是这些不同的表示有多独特？从4\*100种可能的模式中，随机选择的一组模式对将相互重叠约25个细胞。因此，一个特定输入在不同的上下文中的两种表征将有大约25个重复的细胞和75个不同的细胞，这使得它们很容易被区分。

HTM层运作的一般规则如下。当圆柱处于活跃状态时，它将查看圆柱中的所有细胞。如果圆柱中的一个或多个细胞已处于预测状态，则只有这些细胞将被激活。如果圆柱中没有处于预测状态的细胞，则所有细胞都将被激活。当输入模式匹配时，系统通过激活处于预测状态的细胞来确认其预测。如果输入没有被预测，则所有的细胞都会被激活，就好像在说所有可能的内部预测都是合法的。

如果没有先前的状态，也因此没有上下文和预测，则当圆柱变为活跃状态时，圆柱中的所有细胞都将变为活跃状态。这种情况类似于听到歌曲中的第一个音符。没有上下文，您通常无法预测接下来会发生什么；所有选项都可用。如果存在先前状态，但输入与预期不匹配，则活跃圆柱中的所有细胞都将变为活跃的。这个确定过程各圆柱是独立的，因此预测不是要么完全匹配要么完全不匹配。

如上述术语部分所述，HTM单元可以处于三种状态之一。如果一个单元由于前馈输入而激活，我们只使用术语“激活”。如果细胞由于与附近其他细胞的横向连接而处于活跃状态，我们称其处于“预测状态”（图3）。

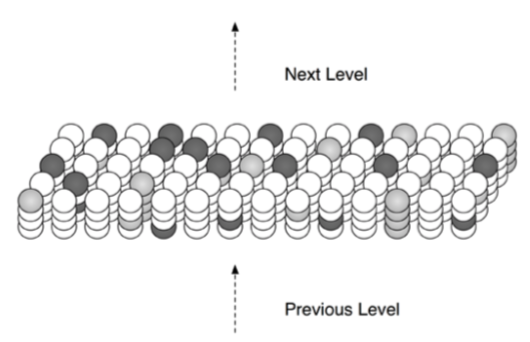


图3 在任何时间点，由于前馈输入，HTM层中的一些细胞将处于活跃状态（以浅灰色显示）。另外一些接收活跃细胞横向输入的细胞将处于预测状态（以深灰色显示）。

1. 根据先前的输入进行预测。

时间池算法的下一步是预测下一步可能发生的事情。预测基于步骤2中形成的表征，其中包括来自所有以前输入的上下文。

当一个层做预测时，它会使所有可能由于未来的前馈输入而变得活跃的细胞去极化（即进入预测状态）。因为一个层中的表征是稀疏的，所以可以同时进行多个预测。例如，如果有2%的圆柱由于输入而处于活跃状态，可以设想，十个不同的预测会导致20%的圆柱具有预测细胞。或者，20种不同的预测可能会导致40%的圆柱具有预测单元。如果每圆柱有10个细胞，进行一次预测，那么4%的细胞将处于预测状态。

稀疏分布表征的章节表明，即使将不同的预测合并（联合）在一起，HTM层也可以非常确信是否预测了特定的输入。

层如何进行预测？当输入模式随时间变化时，不同的圆柱和细胞将依次激活。当一个细胞变为活跃状态时，它与附近单元的一个子集形成连接，而这些单元先前是活跃的。根据应用程序所需的学习速率，可以快速或缓慢地形成这些连接。稍后，一个单元所需要做的就是查看这些连接以确认自身是否将激活。如果连接处于活跃状态，则单元可能会很快变为活跃状态并进入预测状态。因此，前馈激活将导致一组细胞的激活，并由此引发另外一组细胞进入预测状态。把这当作你识别出一首歌并开始预测下一个音符的时刻。

总之，当一个新输入到达时，它会导致一组稀疏的活跃圆柱。每圆柱中的一个或多个细胞变为活跃状态，而这些细胞又通过层中细胞之间的学习到的连接使其他细胞进入预测状态。由层内连接激活的细胞构成对下一步可能发生的事情的预测。当下一个前馈输入到达时，它选择另一组稀疏的活跃圆柱。如果一个新的活跃圆柱没有被预测，即它不含有处于预测状态的细胞，那么它将激活圆柱中的所有细胞。如果一个新的活跃圆柱有一个或多个预测细胞，则只有这些细胞将被激活。

时间沉积池算法伪代码

时间池算法从空间池算法停止的地方开始，现在我们有一组表示前馈输入的活跃圆柱。在下面的伪代码中，算法包括以下步骤：

1。接收一组活性柱，根据细胞的预测状态选择一组活跃的细胞：

a.对于每个活性柱，检查柱中具有活性远端树突段的细胞（即前一时间步骤中处于“预测状态”的细胞），并激活它们。如果没有细胞具有活跃段，则激活圆柱中的所有细胞，将此圆柱标记为“bursting”。活跃细胞的结果集表示当前输入在上下文中的表征。

b.对于每个活跃柱，至少选择一个远端树突段进行学习。对于每一个爆发柱（bursting），选择一个在任何连通值水平上都有一些活跃突触的片段。如果没有这样的片段，则与拥有最少的树突片段的细胞建立连接，随机断开连接。在所有学习片段中，增加每一个活跃的突触的连通值，减少每一个不活跃的突触的连通值，并向先前活跃的细胞生长新的突触。

2。激活一组树突片段：对于层中每个细胞上的每个树突片段，计算与当前活跃细胞相对应的连接的突触数量（在步骤1中计算）。如果数量超过阈值，则该树突段标记为激活。在下一个时间步骤中，具有活跃远端树突片段的细胞集合将成为预测细胞。

在每个时间步结束时，我们增加时间步计数器“t”。

该算法不存储一组预测细胞，而是存储一组活跃的远端树突片段。如果细胞具有活跃的树突片段，则该细胞处于预测状态。

1. 根据预测评估活跃圆柱。选择一组活跃细胞。

将一组活性柱转化为一组活性细胞，并生长、增强、惩罚远端突触，以更好地预测这些柱。首先，对于每一圆柱，判断它是否属于下面某一种状态：

•活跃的且有一个或多个预测细胞

•活跃的且没有预测细胞

•不活跃的且至少有一个预测细胞

使用不同的函数处以每一种情况，如下定义。

1. for column in columns

2. if column in activeColumns(t) then

3. if count(segmentsForColumn(column, activeSegments(t-1))) > 0 then

4. activatePredictedColumn(column)

5. else

6. burstColumn(column)

7. else

8. if count(segmentsForColumn(column, matchingSegments(t-1))) > 0 then

9. punishPredictedColumn(column)

如果其中任何一个细胞有活跃的远端树突段（第3-4行），则该柱被成功预测。如果圆柱中没有一个细胞具有活跃段，则该圆柱处于burst状态（第6行）。

如果非活性柱中的树突片段的活性足以“匹配”，则这些树突片段会受到惩罚（第89行）。

10. function activatePredictedColumn(column)

11. for segment in segmentsForColumn(column, activeSegments(t-1))

12. activeCells(t).add(segment.cell)

13. winnerCells(t).add(segment.cell)

14.

15. if LEARNING\_ENABLED:

16. for synapse in segment.synapses

17. if synapse.presynapticCell in activeCells(t-1) then

18. synapse.permanence += PERMANENCE\_INCREMENT

19. else

20. synapse.permanence -= PERMANENCE\_DECREMENT

21.

22. newSynapseCount = (SYNAPSE\_SAMPLE\_SIZE –

23. numActivePotentialSynapses(t-1, segment))

24. growSynapses(segment, newSynapseCount)

对于每个活跃圆柱，如果具有处于预测状态的细胞，则这些处于预测状态的细胞将变为活跃的（第11-12行）。所有这些都被标记为“赢家”（第13行），使它们在下一个时间步骤中成为突触前细胞的候选细胞。

对于每一个正确激活的片段，加强那些使该片段激活的突触，惩罚那些不起作用的突触（第16-20行）。如果节段的活跃突触小于SYNAPSE\_SAMPLE\_SIZE，则与上一个时间步骤获胜细胞的一个子集建立突触连接，以弥补这种差异（第22-24行）。

25. function burstColumn(column)

26. for cell in column.cells

27. activeCells(t).add(cell)

28.

29. if segmentsForColumn(column, matchingSegments(t-1)).length > 0 then

30. learningSegment = bestMatchingSegment(column)

31. winnerCell = learningSegment.cell

32. else

33. winnerCell = leastUsedCell(column)

34. if LEARNING\_ENABLED:

35. learningSegment = growNewSegment(winnerCell)

36.

37. winnerCells(t).add(winnerCell)

38.

39. if LEARNING\_ENABLED:

40. for synapse in learningSegment.synapses

41. if synapse.presynapticCell in activeCells(t-1) then

42. synapse.permanence += PERMANENCE\_INCREMENT

43. else

44. synapse.permanence -= PERMANENCE\_DECREMENT

45.

46. newSynapseCount = (SAMPLE\_SIZE –

47. numActivePotentialSynapses(t-1, learningSegment))

48. growSynapses(learningSegment, newSynapseCount)

如果圆柱的激活没有被预测到，那么圆柱中的每个细胞都将变为活跃的（第26-27行）。

为圆柱（第29-35行）选择一个获胜的细胞和一个学习段。如果有细胞具有匹配段，选择最佳匹配段及其细胞（第30-31行）。否则，选择最少被使用的细胞，并在其上建立新的树突片段（第33-35行）。

在学习环节，加强对激活树突片段有帮助的突触，惩罚那些不起作用的突触（第40-44行）。然后将新的突触生长到上一个时间步骤中获胜的细胞的一个子集（第46-48行）。

49. function punishPredictedColumn(column)

50. if LEARNING\_ENABLED:

51. for segment in segmentsForColumn(column, matchingSegments(t-1))

52. for synapse in segment.synapses

53. if synapse.presynapticCell in activeCells(t-1) then

54. synapse.permanence -= PREDICTED\_DECREMENT

当带有匹配片段的圆柱没有被激活时，惩罚那些导致这些片段“匹配”的突触。

1. 激活一组树突片段

计算一组活跃和匹配的树突片段。活跃的树突片段指示了处于预测状态的细胞。在匹配的树突段被用于在下一时刻挑选用于学习的细胞。

55. for segment in segments

56. numActiveConnected = 0

57. numActivePotential = 0

58. for synapse in segment.synapses

59. if synapse.presynapticCell in activeCells(t) then

60. if synapse.permanence ≥ CONNECTED\_PERMANENCE then

61. numActiveConnected += 1

62.

63. if synapse.permanence ≥ 0 then

64. numActivePotential += 1

65.

66. if numActiveConnected ≥ ACTIVATION\_THRESHOLD then

67. activeSegments(t).add(segment)

68.

69. if numActivePotential ≥ LEARNING\_THRESHOLD then

70. matchingSegments(t).add(segment)

71.

72. numActivePotentialSynapses(t, segment) = numActivePotential

如果突触的连通值足够高，它就是“连通的”（第60行）。对于每个片段，计算连通的和潜在的活跃的突触（第58-64行）。如果其中一个数字足够高，则该段分别被视为“活跃的”或“匹配的”（第66-70行）。

记录具有有效输入的潜在突触的数量，以便我们在下一个时间步骤（第72行，第23、47行）的学习过程中使用它。

现在，重复一遍。回到第一阶段。把这两个阶段放在一起，就得到了整个算法。

支持函数

这是一些在上面使用的函数。

73. function leastUsedCell(column)

74. fewestSegments = INT\_MAX

75. for cell in column.cells

76. fewestSegments = min(fewestSegments, cell.segments.length)

77.

78. leastUsedCells = []

79. for cell in column.cells

80. if cell.segments.length == fewestSegments then

81. leastUsedCells.add(cell)

82.

83. return chooseRandom(leastUsedCells)

具有最少的树突片段的细胞是被使用最少的细胞。找到所有具有最少的树突片段的细胞（第74-81行），然后随机选择一个（第83行）。

84. function bestMatchingSegment(column)

85. bestMatchingSegment = None

86. bestScore = -1

87. for segment in segmentsForColumn(column, matchingSegments(t-1))

88. if numActivePotentialSynapses(t-1, segment) > bestScore then

89. bestMatchingSegment = segment

90. bestScore = numActivePotentialSynapses(t-1, segment)

91.

92. return bestMatchingSegment

最佳的“匹配”树突片段是具有最多的潜在活跃突触的片段（突触具有有效树突，但连通值没有达到阈值-译者注）。

93. function growSynapses(segment, newSynapseCount)

94. candidates = copy(winnerCells(t-1))

95. while candidates.length > 0 and newSynapseCount > 0

96. presynapticCell = chooseRandom(candidates)

97. candidates.remove(presynapticCell)

98.

99. alreadyConnected = false

100. for synapse in segment.synapses

101. if synapse.presynapticCell == presynapticCell then

102. alreadyConnected = true

103.

104. if alreadyConnected == false then

105. newSynapse = createNewSynapse(segment, presynapticCell,

106. INITIAL\_PERMANENCE)

107. newSynapseCount -= 1

当建立新的突触连接时，与上一时间步的赢家细胞的随机子集建立连接。一个片段只能与突触前细胞建立一次连接（第99-104行）。

时间沉积池实现、数据结构和例程

在本节中，我们描述了时间沉积池的实现细节和术语。

每个时间池细胞都有一个树突片段列表，其中每个片段包含一个突触列表。每个突触都有一个突触前细胞和一个连通值。

潜在突触的实现不同于空间池算法的实现。在空间池算法中，潜在突触的完整列表表示为显式列表。在时间池算法中，每个片段可以有自己的（可能很大的）潜在突触列表。在实践中，为每个段维护一个长的列表在计算上是昂贵的，并且需要大量的内存。因此，在时间池算法中，我们在学习过程中随机添加活跃的突触到每个片段（由参数synapse\_sample\_size的大小控制）。这种优化与保持潜在突触的完整列表有着相似的效果，但是每个片段的列表要小得多，同时仍然保持学习新的时间模式的可能性。

时间池算法允许三种相互排斥的单元状态：“活跃”、“非活跃”和“预测”。但在上面的伪代码中，一个细胞既可以是活跃的，也可以是预测的。这种情况偶尔会发生，例如序列“aabc”，如果输入“a”，时间池算法将预测“a”和“b”，因此此时一些“a”细胞既活跃又具有预测性。

时间池算法允许单个远端树突片段与先前活跃细胞的多个模式形成连接，但上面的伪代码通常不会这样做。它每段学习一个模式。这种方法导致产生与严格必要的相比更多的树突片段，但这对算法没有根本的影响。

|  |  |
| --- | --- |
| t | 当前时间步。在激活树突片段步骤后增长 |
| columns | 圆柱列表 |
| segments | 树突片段列表 |
| activeColumns(t) | T时刻的活跃圆柱列表。时间沉积池的输入 |
| activeCells(t) | T时刻的活跃细胞集合 |
| winnerCells(t) | T时刻的获胜细胞集合 |
| activeSegments(t) | T时刻的活跃树突片段集合 |
| matchingSegments(t) | T时刻的匹配树突片段集合 |
| numActivePotentialSynapses (t, segment) | T时刻给定树突片段的潜在活跃突触数量。潜在突触指连通值未达到阈值 |
| ACTIVATION\_THRESHOLD | 树突片段的激活值阈值。如果一个树突片段的活跃连通突触大于该值，则认为树突片段是活跃的 |
| INITIAL\_PERMANENCE | 突触的初始连通值 |
| CONNECTED\_PERMANENCE | 突触的连通值阈值 |
| LEARNING\_THRESHOLD | 段的学习阈值。如果一个片段中的潜在活跃突触的数量大于或等于这个值，则该片段是“匹配的”，并认为可以与之前活跃的细胞建立或强化突触连接 |
| LEARNING\_ENABLED | True或False。如果为真，则时间沉积池算法在每个时间步生成、强化、惩罚突触连接 |
| PERMANENCE\_INCREMENT | 如果一个树突片段正确的预测了一个细胞的激活，则这个片段的活跃突触的连通值将会增加该值 |
| PERMANENCE\_DECREMENT | 如果一个树突片段正确的预测了一个细胞的激活，则这个片段的非活跃突触的连通值将会减少该值 |
| PREDICTED\_DECREMENT | 如果一个树突片段没有正确的预测一个细胞的激活，则这个片段的活跃突触的连通值将会减少该值 |
| SYNAPSE\_SAMPLE\_SIZE | 树突片段上活跃突触的所需数目。一个学习片段将增长突触达到这个数量。段连接到SDR的一个子集，这是该子集的大小 |

表1。时间池伪码中使用的变量和数据结构。

|  |  |
| --- | --- |
| segmentsForColumn(column, segments) | 给出一组树突片段，返回在该圆柱中某一细胞的片段 |
| count(collection) | 给出集合中的元素数量 |
| chooseRandom(collection) | 从集合中随机选择一个元素 |
| collection.add(element) | 往集合中添加元素。不允许重复 |
| collection.remove(element) | 从集合中移出元素 |
| chooseRandom(collection) | 从集合中随机选择一个元素 |
| min(a, b) | 等价于“if a ≤ b then return a else return b” |

表2。时间沉积池伪代码使用的支持例程。它们在nupic代码库中可能有不同的名称。