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. Purdy

HTM 系统的编码数据

本章，我们会介绍如何将数据编码为系数分布表征（SDR），从而用于 HTM 系统。我们会讲解现有的几种编码器，这些都可以通过访问开源项目 NuPIC[1]获取，还会对创建面向新的数据类型的编码器的需求进行论述。

什么是编码器?

层次时序记忆 （HTM）提供了一个灵活和贴近生物学的框架，用于解决涉及众多数据类型的预测、分类和异常检测问题【Hawkins and Ahmad, 2015】。HTM 系统要求输入数据的格式是稀疏分布表征（SDRs）【Ahmad and Hawkins, 2016】。SDR 与标准的计算机表征形式很不一样，比如面向文本的 ASCII 码，含义是直接编码到表征里面的。一条 SDR 由大量的比特组成，其中大部分是 0 而小部分是 1。每个比特都携带一些语义含义，所以如果两条 SDR 在相同位置都有不少的置 1 比特，那么这两条 SDR 就有相似的含义。

可以转化成 SDR 的任何数据都可以用于采用 HTM 系统的各类应用程序。所以，运用 HTM 系统的第一步是通过我们称之为"编码器"的东西把数据源转化成 SDR。编码器把数据的原生格式转化成可以输入到 HTM 系统的 SDR。对应给定的输入数据，编码器负责确定哪些输出比特应该是 1，哪些应该是 0，从而捕获数据的重要语义特征。相似的输入信息应该产生彼此高度重叠的 SDR。

编码过程

编码过程类似于人类和其他动物的感觉器官的功能。比如，耳蜗是一种专门的结构，它将环境中声音的频率和幅度转换成稀疏的活跃神经元【Webster et al, 1992; Schuknecht, 1974】。这个过程（图 1）的基本机制是通过一组排列成多排的内毛细胞构成的，每排内毛细胞对不同的声音频率敏感。当出现合适的声音频率时，内毛细胞会刺激神经元，将信号发送到大脑。以这种方式触发的一组神经元把这些声音编码组合成稀疏分布表征。

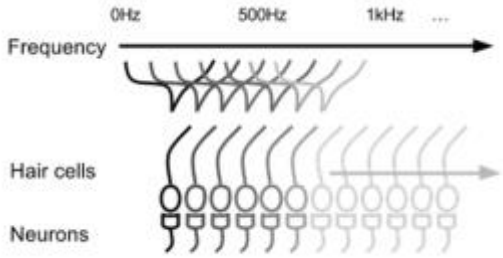


图 1 耳蜗内毛细胞基于声音的频率刺激一组神经元。

[1] http://numenta.org

耳蜗编码过程的一个重要方面是，每个内毛细胞都能响应一系列频率，频率范围与其他邻近内毛细胞的相互重叠。这种特性通过冗余防止某些毛细胞受损，但是也意味着某个特定频率会刺激多个细胞。两个相似频率的声音刺激的内毛细胞会有一些重叠。表征之间的重叠表现了所捕获的数据之间的语义相似程度。这也意味着语义是分布在一组活跃细胞的，使得表征能够容忍噪声和子采样。

不同动物的耳蜗响应不同频率的声音，它们使用不同的分辨率来区分声音频率之间的差异。虽然频率很高的声音对某些动物来说很重要，但对其他动物而言可能没什么用。同样，编码器的设计取决于数据的类型。编码器必须捕获那些对应用程序重要的数据的语义特征。在 NuPIC 项目中的很多编码器的实现里面，设定了可以允许它们为广泛的应用程序工作的范围或者分辨率参数。

在编码数据时，有一些重要的方面需要考虑：

1. 语义相似的数据应该产生活跃比特彼此重叠的 SDR
2. 相同的输入数据应该总是产生相同的输出结果
3. 对所有的输入信息，输出信息应该有相同的维度（比特总数）
4. 对所有的输入信息，输出信息应该有相似的稀疏度，并且有足够的置 1 比特处理噪声和子采样

在接下来的几节，我们会细致地考究每个特性，然后介绍如何对几种不同类型的数据编码。要注意，已经有一些现成的 SDR 编码器了，大多数人不需要自己去创造一个。如果有自行创造编码器的需要，应该仔细考虑上面的标准。

1）语义相似的数据应该产生活跃比特彼此重叠的 SDR

要创造一个有效的编码器，你必须理解数据中有利于相似性的方面。在上面的耳蜗例子中，编码器的设计意图是使音高相近的声音有相似的表现，但是并没有考虑声音的响度，因为这个需要不同的方式。

设计编码器的第一步是确定所要捕获的数据特征。对声音而言，关键特征是音高和振幅；对日期而言，可能就是那一天是不是周末。

在数据中我们所选的一个或多个特征中，对彼此类似的输入信息，编码器创造的表征应该是重叠的。所以对周末编码器，属于周六和周日的日期应该相互重叠，但是重叠部分没有属于工作日的日期那么多。

Preserving Semantics: A Formal Description

Here we formalize the encoding process by defining a set of rules that relate the semantic similarity of two inputs with the number of overlapping one-bits in the corresponding encoded SDRs.

设 $\mathcal{A}$ 为一个任意的输入空间，设 $S(n,k)$ 为长度为 n 且有 k 个 ON 比特的 SDR 集合。编码器 f 仅仅是一个函数 $f:\mathcal{A}\to S(n,k)$ 。输入空间 $\mathcal{A}$ 的距离评分 $d_{\mathcal{A}}$ 是函数 $d_{\mathcal{A}}:\mathcal{A}\times\mathcal{A}\to\mathbb{R}$，满足下面三个条件：

1. $\forall x,y\in\mathcal{A}, d_{\mathcal{A}}(x,y)\geq 0$
2. $\forall x,y\in\mathcal{A}, d_{\mathcal{A}}(x,y)=d_{\mathcal{A}}(y,x)$
3. $\forall x\in\mathcal{A}, d_{\mathcal{A}}(x,x)=0$

等式 1 要求语义相似度度量给出的距离值大于等于 0；等式 2 要求距离度量具备对称性；等式 3 要求完全相同的值之间的距离值为 0。

给定一个输入空间和距离评分，我们可以通过比较输入数据对的距离评分和它们编码的重叠程度来评价编码器。距离评分低的输入数据对应该对应重叠程度高的两条 SDR，反之亦然。此外，如果某个输入数据对比另一个输入数据对的重叠程度更高，那么前者的预编码的距离评分应该低于后者的。下面我们给出正式的规定。

对于相同长度的两条 SDR s 和 t，设 $O(s,t)$ 为重叠部分的比特数目（也就是两者取交 s&t 中 ON 比特的数目）。那么对编码器 $f:\mathcal{A}\to S(n,k)$ 和 $\forall w,x,y,z\in\mathcal{A}$，

4. $O(f(w),f(x))\geq O(f(y),f(z))\Leftrightarrow d_{\mathcal{A}}(w,x)\leq d_{\mathcal{A}}(y,z)$

等式 4 规定，有更多重叠的置 1 比特的编码对意味着它们的语义相似度更大，并且反过来说，语义相似度更大的编码对会有更多重叠的置 1 比特。创建一个满足这种要求的编码器并不总是可能的，但是这个公式可以作为一种启发式的标准来评估编码器的质量。

2）相同的输入数据应该总是产生相同的输出结果

编码器应该是确定性的，这样相同的输入数据每次都会产生相同的输入结果。如果没有这个特性，HTM 系统学到的序列将会随着值的编码表征的变化而过时。要避免去构建包含随机或者自适应成分的编码器。

创建自适应的编码器来调整表征以便处理未知范围的输入数据的做法很吸引人。后面的“一种更灵活的编码器方法”一节中会介绍一种不用改变输入数据的表征就能处理这种情形的设计编码器的方法。这种方法允许编码器处理输入范围无限或者未知的情形。

3）对所有的输入信息，输出信息应该有相同的维度（比特总数）

编码器的输出必须总和它的输入有相同的比特数量。SDR 之间的比较和其他操作是逐比特进行的，这样表示某个特定“含义”的比特总是在同一个位置。如果编码器产生的 SDR 长度不同，那么比较和其他操作就不可行了。

4）对所有的输入信息，输出信息应该有相似的稀疏度，并且有足够的置 1 比特处理噪声和子采样

编码器中 ON 比特所占比重从 1%到 35%不等，但是对某个编码器的特定应用而言，稀疏程度应该是相对固定的。虽然稀疏程度保持一致应该作为一条规则，但是稀疏度的细微变化不会带来负面影响。

此外，必须要有足够的置 1 比特来处理噪声和子采样。根据一般经验，应该至少有 20-25 个置 1 比特。产生的表征只有不到 20 个置 1 比特的编码器在 HTM 系统中表现不佳。因为它们会对少量噪声或者非确定性所导致的误差极其敏感。

范例 1——对数字编码

最常见的需要编码的数据类型是数字。它可以是任何类型的数值——比如 82 度的角度、145 美元的销售额、34%的容量等等。下面的几节会介绍面向单个数值的越来越高级的编码器。在每一节，我们会改变想要捕获的语义特征，同时更新编码器进而达到新的目标。

简单的用于数值的编码器

最简单的做法是，我们可以模仿耳蜗编码声音频率的方式。耳蜗的内毛细胞对不同但是重叠的频率范围有反应。特定的声音频率会刺激其中的一些细胞。我们可以通过把实际值的重叠范围编码为活跃比特来模仿这个过程（图 2）。这样，当值属于从 0 到 5 时第一个比特会激活，当值属于从 0.5 到 5.5 时下一个比特会激活，依此类推。如果我们设定编码总共有 100 个比特，那么最后一个比特会表示属于从 49.5 到 54.5 的值。这样可以表示的整个范围就是从 0 到 54.5 的值。这种方法可以表示的最小值和最大值是固定不变的。超出允许范围两端的值会分别给出对应最大值或者最小值的表征。

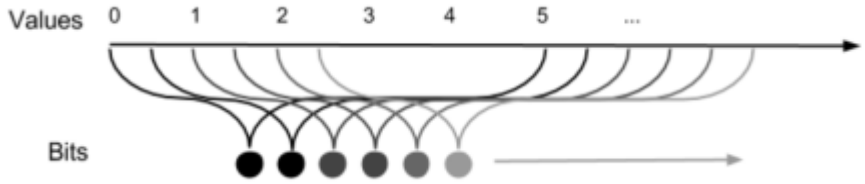


图 2 表征中的每个比特对应着与其相邻比特所对应的重叠的值的范围

采用上面的编码器参数，这里分别是对值 7.0（图 3A）、值 10.0（图 3B）和值 13.0（图 3C）的编码示意。注意相近的数字（7.0 和 10.0，以及 10.0 和 13.0）互相共有一些置 1 比特，但是没那么近的数字（7.0 和 13.0）并不共有任何置 1 比特。

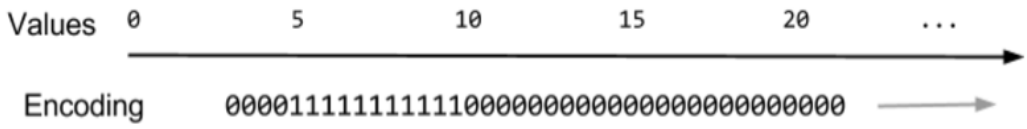


图 1A 值 7.0 对应的编码表征。采用的编码参数是总共 100 个比特，最小值是 0，每个比特对应的范围是 5.0，每个比特增大 0.5。

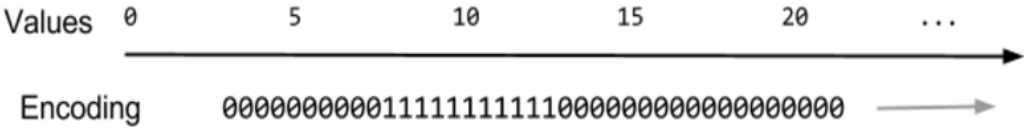


图 3B 值 10.0 对应的编码表征。采用的编码参数是总共 100 个比特，最小值是 0，每个比特对应的范围是 5.0，每个比特增大 0.5。与值 7.0 的表征共有一些置 1 比特。

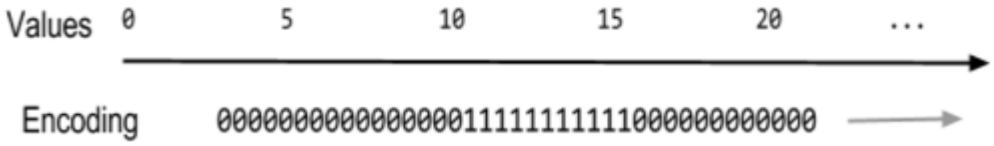


图 3C 值 13.0 对应的比特。采用的编码参数是总共 100 个比特，最小值是 0，每个比特对应的范围是 5.0，每个比特增大 0.5。与值 10.0 的表征共有一些置 1 比特，但是与值 7.0 的表征没有任何共有的置 1 比特。

当我们要实现这种编码器时，首先把值的范围划分成多个桶、然后把这些桶映射到一组活跃细胞。下面是用这种方法对值进行编码的步骤：

1. 选取你要表示的值的范围，也就是 $minVal$ 和 $maxVal$
2. 用 $range = maxVal - minVal$ 计算范围的大小
3. 选取一些桶来表示你要划分的值
4. 选取每个表征中活跃比特的数目 w
5. 计算比特总数 n： $n = buckets + w - 1$
6. 对给定值 v，确定对应的桶 i，满足： $i = floor[buckets * (v - minVal)/range]$
7. 通过把 n 个初始比特中从第 i 个开始的连续 w 个比特翻转成置 1 比特来创造编码表征

请注意，这种编码方案有四个参数：最小值、最大值、桶的数目以及活跃比特的数目（w）。另外，你还可以选取比特总数 n 来代替选取桶的数目，桶的数目可以通过 $\text{buckets} = \text{n} - \text{w} + 1$ 确定。

这里是对某地的室外温度进行编码的例子，当地的气温从 0F° 到 100F° 不等。

1. 最小值是 0F° 并且最大值是 100F°
2. 值的范围是 100
3. 我们选取把范围划分到 100 个桶
4. 我们选取每个表征采用 21 个活跃比特
5. 比特总数通过 $n = buckets + w - 1 = 120$ 确定
6. 现在我们可以挑选出对应 72F° 的桶为 $i = floor[100 * (72 - 0)/100] = 72$
7. 对应的表征是 120 个初始比特中从第 72 个比特开始为 21 个连续的活跃比特:

000000000 … 000011111111111111111111100000000 … 00000000
72nd bit ↗

这种编码方式很简单，也比较灵活，但是需要你知道数据的准备范围。如果你的数据低于预设的最小值或者高于预设的最大值，编码器就不太好用了。一般最小值的桶会处理范围以下的值，最大值的桶会处理范围以上的值。因此所有范围以下的值都会得到同样的表征，所有范围以上的值情况与之类似。下面是温度为 100F° 时对应的表征。这也同样会是温度为 110F° 以及所有其他温度高于 100F° 时对应的表征。

000000000000000 … 000000000000000000001111111111111 … 1111
100th bit ↗

可以根据应用程序中的固有噪声和预测所需的质量来选取桶的数量。如果是处理很嘈杂的信号，需要的桶数目可能比较小。这样 HTM 会得到更稳定的输入数据，但是就不能做出非常精准的预测。如果是处理很干净的信号，需要的桶数目可能比较大。这样 HTM 可以做出非常精准的预测。

更灵活的用于数值的编码器

像耳蜗这样的生物感知器官有一个可以编码的特定范围。人类可以听到频率大约 20Hz 到 20000Hz 的振动。如果世界上的声音偏移到更高的频率，我们的耳朵没有用武之地了。前面描述的编码器机理也有一个特定的范围。编码器的设计者选择范围，但是一旦选定并且系统开始学习，那么就不能再更改了。但是，有办法克服这种限制。借助哈希函数[2]，可以设计出更灵活的编码器，有固定的比特数目，但是编码范围基本上是没有限制的。通过这种设计，一条 SDR 中的每个比特可以代表多个范围的值。如果通过哈希函数来分配这些范围，那么迥然的两个值对应的 SDR 可能会有一两个比特重叠，但是对 HTM 来说这种小重叠不会引起任何问题。我们现在会在数值编码器上演示这种方法，但是本章后面会介绍的地理空间编码器也用到了同样的原理。

如果我们回顾前一部分的步骤 6，就会发现每个桶都由一个特定的比特确定。我们随后为接下来的 w-1 个桶选择比特以便完成表征。每个桶都和它的邻居有重叠的比特。要实现更灵活的数值编码器，我们同样做步骤 1-6，但是改变表征中比特的挑选方式。明确地说，我们可以使用哈希函数从桶的索引中明确地选中输出表征的某个比特。我们对每个比特都单独地做这样的选择，这样就不是简单地设置 w 个连续比特为活跃比特，而是根据哈希函数来确定哪些是活跃比特。哈希操作看起来就像这样：

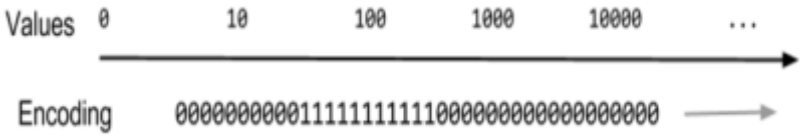
这种方法的优势在于你不需要把值限制到某个整体的范围。相反，你只需要选择每个桶负责的范围。可能会有一些不必要的碰撞。换句话说，两个不应该重叠的桶可能会有一两个比特发生重叠。在实践中，这个问题没有影响。只要你仔细地选择哈希函数，并且设置足够大的 w 值和 n 值。

[2] https://en.wikipedia.org/wiki/Hash_function

如果你从表征中挑出一个比特，你不会知道原来的值是什么，因为很多不同的值会产生包含这个比特的编码。但是如果你把所有的活跃比特都挑出来，这个表征就特别匹配你的输入数据了。这种编码器和很多生物学上的编码器不同，但是它的效果很好，因为它满足我们前面提到的编码特性。

数值对数编码器

一些应用程序需要那种根据数量级的不同来描述数字之间相似性的数值编码器。换句话说，值 4000 和 5000 之间和值 4 和 5 之间是被同等对待的。使用对数函数的编码方式看起来就像这样：



这种编码器对小数值上的细微变化很敏感。从 3 到 15 会产生截然不同的两种编码。但是对大数值上的变化就不会这么敏感。比如，从 1000 到 2000 对编码结果的影响会小很多，尽管在数值上的绝对变化要大很多。

增量编码器

增量编码器的设计意图是捕获值的语义变化，而非值本身。这种技术有助于对具有不同值范围的模式数据进行建模，与常规的数值编码器一起使用可能也会很有用。增量编码器的简单应用场景是对某个持续增长的值进行预测。这种场景的独特之处在于预测模式不在于值本身，而是值的变化——值是在增大还是减小，变化了多少？相比常规的数值编码器是对每个新值给出不同的表征，增量编码器是以相似的幅度对变化的值产生重叠编码。

如果你在某台机器上有一块温度传感器，增量编码器会识别出机器行为引起的温度变化模式，即使周边天气情况造成了绝对温度数值的范围不同。某天的室外温度可能是 60 华氏度，第二天可能是 70 华氏度。对常规的数值编码器来说，60 多华氏度所对应的表征不会和 70 多华氏度所对应的表征有多少重叠，如果有的话。但是当数值以类似的方式变化，增量编码器会产生相似的编码，即使数值本身从来没有出现过。

这种编码器不同于之前的编码器，因为它根据当前的和以往的输入信息来确定输出信息。这种编码器的实现和其他的标量编码器相同，但是你要把它应用到处理当前的输入数据相比于以往数据的差值上。

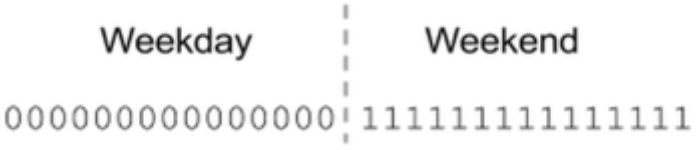
范例 2——对类别编码

很多数据集包含类别信息。有时候，数据是离散的、完全无关的类别（比如商店中产品的型号）。其他时候，数据是相互有些关联的类别（比如一周每日的种类）。下面的例子展示了如何对这些数据编码。

日期和时间的某些特征实际上是有类别之分的。比如：

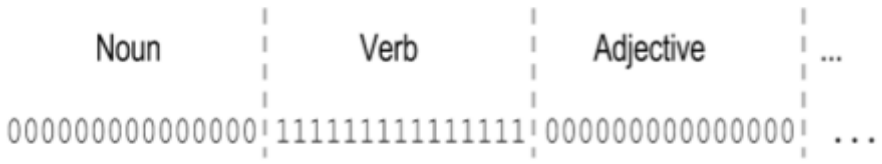
- 工作日 vs 周末
- 节假日 vs 非节假日
- 白天 vs 黑夜
- 用餐时间 vs 非用餐时间
- 单词的词性

很多情况下，把这些特征编码为完全离散的类别是有用的。在这些场合，编码应该尽量减少任何类别的编码之间的重叠。最简单的做法是为每个选项分配一定数量的比特。任何选项的编码都有专门的活跃比特，剩下的是不活跃的比特。这是星期六的工作日/周末类别的编码样例：



把这种编码添加到数据流中很有用，在工作日和在周末的数据流模式是不同的。添加这种编码可以确保 HTM 系统在工作日和在周末是收到了不同的输入模式，这样允许它们可以更容易地学到单独针对两类时间段的预测。

类别编码也可以应用到词性上。这里是一个看起来像是动词的编码：



有序循环类别

有时候，模式是不断变化的，而非完全离散的类别。

模式之间的差异并没有一条严格、离散的界限。中午的模式有时可能比较像早晨的模式，有时又可能比较像傍晚的模式。这些情况下，你可以简单地把它们转化成数值，并且使用标量编码器来生成 SDR。

但是，特别是在日期和时间上，类别不仅仅是连续的，也是循环的。比如，周五是工作日，但是和周四有些不同。周日傍晚是周末，但是不同于周六傍晚。刚开始你可以把一周每日表示成从 0 到 6 的数字，其中 0 代表周日，6 代表周六。数值编码器可以创造出不错的表征，除了周六和周日的编码仅有一点或者根本没有重叠，因为它们分别在编码范围相对的两端。在这些循环的情况下，编码必须“缠绕”。对于本例而言，我们会在编码中用少量的比特。在真正的实现中，你可能需要更多的活跃比特以及更多的比特作为结果。结合示意图来理解它最简单（图 4）：

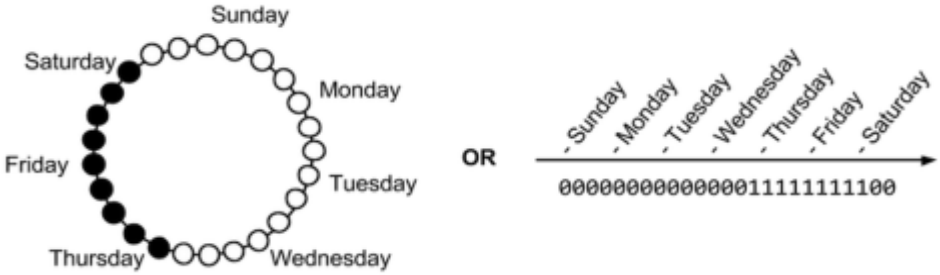


图 4 属于周五的日期的编码

图 4 的两张图展示了周五的编码表征。左图展示了每天是如何参与循环以及如何与前一天、后一天重叠的。右图展示了周五的编码表征以及对每个编码中心的标注。周日的编码既包括最开始的比特，也包括最末尾的比特（图 5）。

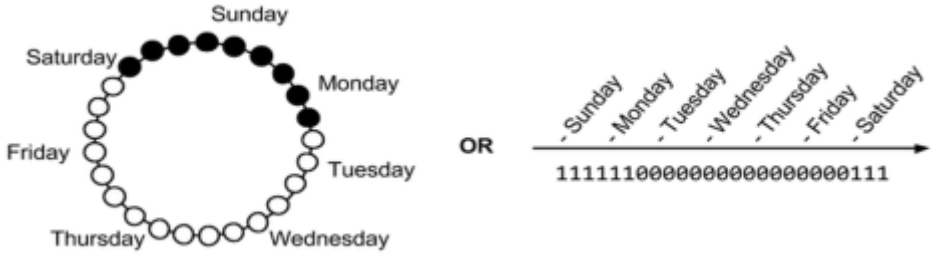


图 5 属于周日的日期的编码

你可能需要捕获的其他关于日期和时间的数值特征是：

- 一年中的每月
- 一月中的每天
- 一天中的每小时
- 一小时中的每分钟

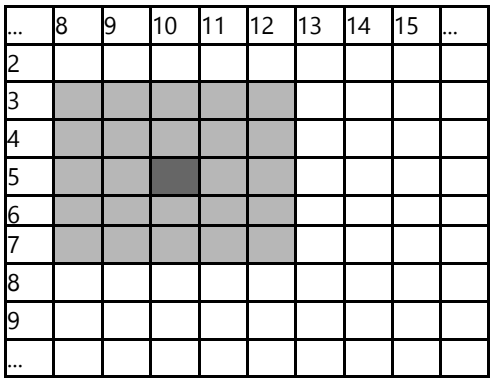
范例 3——对地理空间数据编码

简单的用于地理空间数据的编码器

这个例子展示了编码器如何捕获地理空间数据。最明显的方面是，彼此靠得近的位置应该看作相似，离得远的位置不应该看作相似。要按照这个思路编码，我们首先必须编码过程应该使用的分辨率。对这个例子而言，假设使用精确到大约 10 英尺的 GPS 坐标。我们采用二维坐标，但是进一步推广到三维坐标非常直接。

第一步是把 GPS 坐标转化成平面空间，可以划分成 10 英尺见方的方块。然后我们建立索引系统，这样就可以通过 X 和 Y 的数字坐标[3]确定每个部分。

现在我们需要把位置编码成包含 w 个活跃比特的 n 个比特。在这个例子中，我们设定 n=100 同时 w=25，但是在实际应用中需要设定更大的数字，像 n=1000。我们先选择想要编码的位置的坐标，比如 x=5 和 y=10，然后确定周边位置的坐标。它们会形成一组由 3<=x<=7 和 8<=y<=12 确定的方形位置。这个过程产生了图中灰色区域所示的 25 对位置坐标：



现在我们可以使用确定性的哈希函数来把每对坐标映射到编码中的 100 个比特：

$$Hash(x, y) = ix,y$$

正如前面所解释的，通过哈希函数，我们可以使用固定数量的比特表示无界空间中的任何位置。

因为使用的是确定性的哈希函数，所以我们可以在需要的时候再去计算这些值，而不需要存储它们。特定的 GPS 坐标编码看起来就像这样：

00000010010000000001010000000000000000011000100000000000...

注意，由于哈希碰撞，最终的编码可能只有不足 25 个置 1 比特。当 n 和 w 的值都充分大时，这种情况就不太可能发生，在任何情况下，HTM 系统不会因为它出现问题。

如果我们对 x=6 和 y=10 的位置编码，大多数（25 个中有 20 个）选中的坐标会与 x=5 和 y=10 的位置编码重叠，产生我们期望的 SDR 语义重叠现象。

更灵活的用于地理空间数据的编码器

如果你期望编码能有地理空间系统所允许地那么好，那么之前描述的编码方法效果还不错。如果精确到 10 英尺，那么移动 20 英尺会产生略微不同的 SDR，移动 1000 英尺会产生完全不同的 SDR。但是这可能不是你想要的。你可能想要这样的编码器，当物体缓慢移动时，对细微的变化编码，而在物体快速移动时，对更大的距离编码。比如，如果某人在行走，那么我们会期望编码器去表示像 10 英尺这么小的位置变化，但是如果某人在高速行驶的汽车里，那么 10 英尺就显得微不足道了。高速情况下，200 英尺以内的位置最好用相同的 SDR 表示，相隔 1000 英尺外的位置应该相互重叠。许多地理空间应用都需要这种可变编码。通常有方法设计出能够满足任意需求的编码器。对于这种情况，可以借助所有可能位置的子采样以及通过物体的速度来确定子采样的粒度，进而形成相应的 SDR，从而满足这种需求。

---

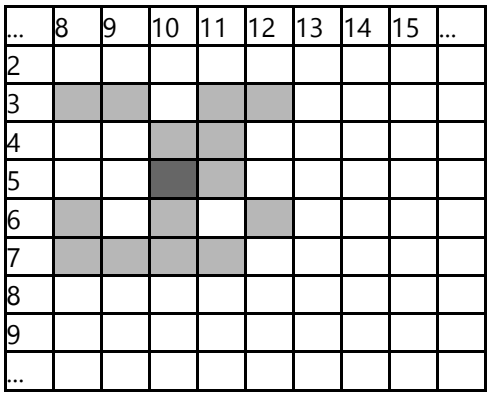
[3] The spherical mercator is one projection that can be used to transform GPS coordinates into two dimensional coordinates: https://en.wikipedia.org/wiki/Mercator_projection

前面描述的地理空间编码需要通过你想要彼此重叠的两个位置的距离来决定活跃比特的数目。你可能并不总希望有很多活跃比特。我们可以从编码范围做子采样，但是需要确定哪些比特做子采样，以便保留所需要的编码器特性。具体地说，我们想要空间相邻的数据点的编码中共有一些比特。如果我们在空间半径内任选 50%比特做子采样，可能和附近位置的编码没有任何共有比特，尽管它们的空间半径是重叠的。

解决这个问题的一个办法是对所有的坐标给出严格的次序，由这个次序决定对那个比特做子采样。为了做到这一点，我们可以用一个确定性的哈希函数把每个坐标都映射为 0.0 到 1.0 之间的浮点数：

$$\text{Hash}(x, y) = w_{x,y}$$

因为我们使用的是确定性的哈希函数，这些权重可以在需要的时候计算，不需要存储到存储器中。假设我们的编码中要有 15 个活跃比特，那么之前的编码看起来就像这样：



权重方案有助于对相同的坐标产生偏好，这样附近位置的编码也会选中一些相同的坐标，但是这一点并不能保证。因此，挑选充分高的比特总数和活跃比特数对确保出现适当的重叠是必要的。

把速度引入地理空间编码器

坐标或者地理空间编码器的一个有前景的应用是对行人或者车辆位置的异常检测。知道行人或者车辆的速度对确定两个位置不再共享重叠时的距离临界值很重要。一种自动化这个过程的方法是动态地调整半径，根据当前速度选择重叠比特，相对于你的速度来改变距离参数，而不是把它作为绝对的度量标准。

接下来的几张图（图 6A、图 6B 和图 6C）展示了三个位置的编码表征。前两个编码表示实体缓慢移动时的位置，第三个编码表示实体加速后的位置。注意，尽管第三个位置已经离得很远了，但是它和前一个编码共有的比特数目与前面两个共有的比特数目差不多。

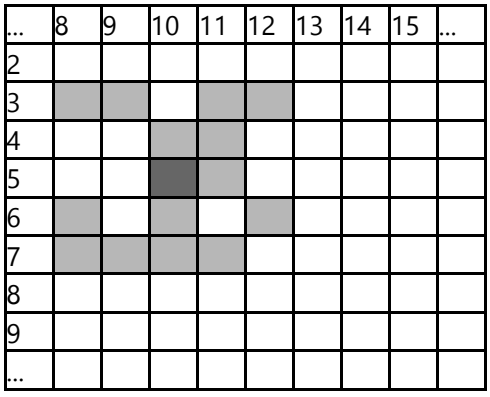
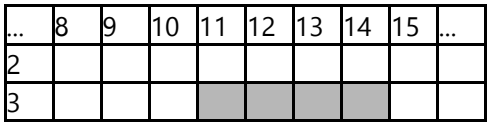


图 6A 这是实体缓慢移动时的位置编码。深灰色的方块代表当前的位置 X。浅灰色的方块代表半径为 2 的范围中挑选为部分表征的比特。这 15 个方块是半径内 25 个方块中权重最高的。



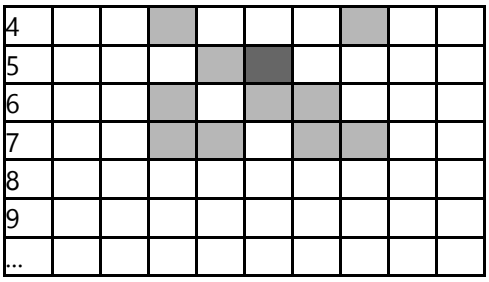


图 6B 这是 X 附近位置的编码，实体以原来的速度继续移动。实体已经向右移动了两个单位。在同时处于 X、Y 半径内的方块中，大多数相同的方块被选中作为部分表征，因为编码中每个方块的权重是固定的。并且半径和 X 的半径一样，因为实体是以相同的速度运动。

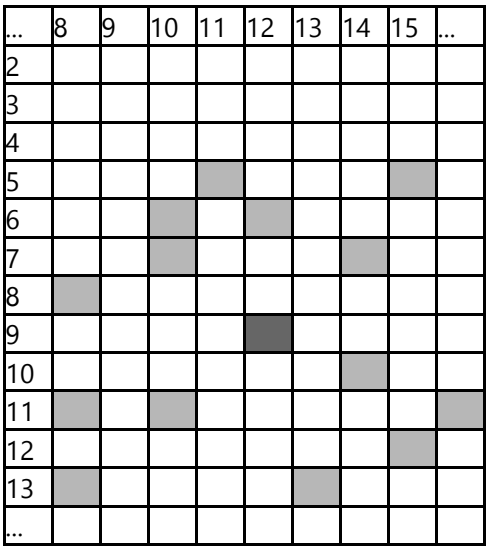


图 6C 这是实体已经继续向下移动四个单位的编码。因为实体的移动得更快了，所以半径也增大了。半径范围内挑选为部分表征的方块比重比之前小，因为我们还是选出同样数目（15 个）的比特，不过可供选择的数目（81 个）更大了。尽管新位置 Z 离 Y 相当远，并且是选择一组稀疏的坐标，权重方案确保了仍然有一些被选中作为部分表征的坐标同时在 Y 和 Z 中。

范例 4——对自然语言编码

单词、句子和文档也可以编码成 SDR。有很多现成的技术可以创造语言的向量编码，包括 Cortical.io 的办法。他们的技术在白皮书【Webber, 2015】里面有详细的描述。

选择新编码器的尺寸

无论编码器是表征什么的，都应该创造出有特定的比特数目 n 和特定的置 1 比特数目 w 的 SDR。你怎么知道 n 和 w 的值设定成什么？为了保持稀疏的特性，w 不能占据 n 的很大比重。但是如果我们设定得太小，我们会失去分布表征的特性。一般来说，w 的值至少要有 20，这样才能处理噪声和子采样，n 的值至少要有 100，这样才有足够的分辨能力区分很多不同的数字。

这里有一些值，是在实际应用中表现得很成功的。

- 在一个从服务器观察数值的度量值的实验中，某个优化算法选择了 n=134 和 w=21 作为理想值。
- 在另一个实验中，n=2048 和 w=41 对地理空间编码器较为理想。
- 当对类别编码时，w 可以占据 n 较大的比重。考虑某个极端情况，如果你对二进制值编码，那么 n 可以取 100 同时 w 可以取 50 。

当创造新的编码器时，采用概括性的准则来初始化 n 和 w 通常是很好的策略。在编码器调试后同时有精度测试就绪时，可以通过仔细选择 n 和 w 来改进编码器的性能。很多关于选择 n 和 w 的合适值的背景来自 Ahmad 和 Hawkins 在 SDR 特性上做出的工作【Ahmad and Hawkins, 2016】。

多值编码

一些应用场景需要在单一的 HTM 模型上编码多个值。单独的值可以自主编码，然后级联形成组合式编码。这么做时，保证每个单独的值编码都有相似数目的置 1 比特非常重要，这样其中任何一个编码都不会占据主导地位。每个单独的编码器可以有不一样的比特总数 n 值。

小结

有很多现成的编码器实现，它们可以覆盖大多数的应用场景。但是如果你需要为某种新的数据类型构建编码器，你可以利用下面的几条简单规则：

1. 语义相似的数据应该产生包含重叠的活跃比特的 SDR
2. 相同的输入数据总是应该产生相同的 SDR 作为输出结果
3. 所有的输入数据产生的输出结果都应该有相同的维度（比特总数）
4. 所有的输入数据产生的输出结果都应该有相似的稀疏度，同时有足够的置 1 比特处理噪声和子采样

参考文献

Hawkins, J. & Ahmad, S. (2016). Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex. Frontiers in Neural Circuits, March 2016, Volume 10. Retrieved from http://dx.doi.org/10.3389/fncir.2016.00023

Ahmad, S., & Hawkins, J. (2016). How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites. arXiv, 1601.00720. Neurons and Cognition; Artificial Intelligence. Retrieved from http://arxiv.org/abs/1601.00720

Webster D, Popper A, Fay R editors (1992). The Mammalian Auditory Pathway: Neuroanatomy. Springer handbook of auditory research, v1. Springer-Verlag New York, Inc

Schuknecht, H.F. (1974). Pathology of the Ear. Cambridge, MA: Harvard University Press

Webber, F. D. S. (2015). Semantic Folding Theory And its Application in Semantic Fingerprinting, 57. Artificial Intelligence; Computation and Language; Neurons and Cognition. Retrieved from http://arxiv.org/abs/1511.08855