# The ML-Index: A Multidimensional, Learned Index for Point,Range, and Nearest-Neighbor Queries

摘要：我们提出了ML-Index，一个可以处理点查询、kNN查询和范围查询的内存高效多维学习型结构。使用数据相关的参考点，ML-Index将数据分区并将其转换为相对于与其最近参考点的距离的一维值。缩放后，ML-Index 利用学习模型来有效地近似缩放值的顺序。我们提出了一种新颖的偏移缩放方法，与 iDistance 方法的现有缩放方法相比，它提供了更容易学习的函数。 我们通过使用两个真实数据集进行彻底的实验性能比较来验证可行性并展示我们的方法的优越性。

1. 介绍

处理多维数据查询是一个经典并经过彻底研究的问题。大量的索引结构提供了紧凑地存储和查询多维数据集的机制，应用于无数的应用场景。Kraska 等人最近提出的一个想法。 [4]，建议用机器学习和深度学习模型改进和替换传统索引结构。他们特别成功地用递归学习模型取代了 B 树，该模型将键映射到排序数组中记录的估计位置。通过使用所提出的学习模型，他们能够利用数据分布中的模式，从而比传统索引改善了内存消耗和执行时间。

为了提供多维 B 树的泛化，数据需要按照监督学习模型可以轻松学习的顺序进行排序。排序需要以在回答范围和 KNN 查询时满足正确性保证的方式完成。在实践中，可以利用 Morton 和 Peano-Hilbert 顺序等技术对多维数据进行排序。 然而，深度学习模型无法轻松学习在上述顺序内直接映射多维数据点。Kraska等人。 [3]提出了一种学习顺序的方法，该方法基于对多个维度上的点进行连续排序和划分为大小相等的分区。 然而，当维度数量增加时，仅选择维度的子集可能会导致性能下降，并且分区邻居的推导可能不是一项节省时间的任务。

因此，我们创建了一种新颖的多维学习（ML）索引，它概括了著名的 iDistance 缩放方法 [2] 的思想，并使用缩放排序与两层学习索引相结合，来回答多维查询。不像现有的索引，它通过相对于分布感知参考点有效地分区和缩放数据以及学习排序缩放值的分布来捕捉数据分布。ML-Index 利用深度学习模型的力量，是第一个完整的学习索引，能够高效地回答点、范围和 KNN 查询，同时具有较低的内存消耗。

1. 相关工作

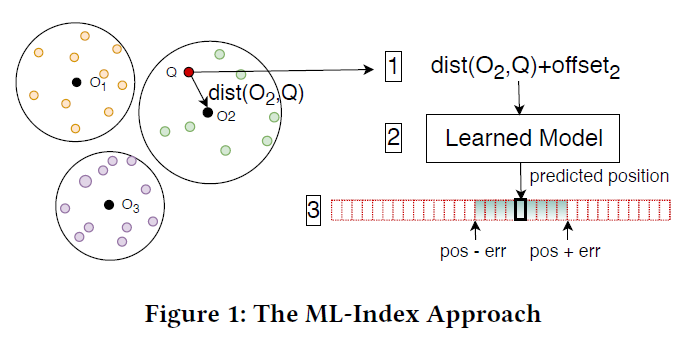
直到最近，通过将数据索引与机器学习结合来改进数据索引的研究还很有限。 第一个分布感知索引 [1] 重点关注 R 树与自组织映射的组合。另一种精确 KNN 算法 [9] 采用 k 均值聚类和三角不等式剪枝来实现高效的查询执行。

最近，Kraska等人[4]聚焦于一种新颖的思想，即用深度学习模型替代索引。在假设每个索引是一个模型的基础上，他们在索引和机器学习世界中的相应物之间建立了类比。例如，B树索引和哈希索引可以被视为将键映射到有序和无序数组中的位置的模型，并且可以很容易地用神经网络模型替代。一种名为SageDB的学习型数据库系统[3]通过沿着一系列维度逐步对点进行分区，将它们划分为相等大小的单元并按照它们所占据的单元对它们进行排序，从而将这些概念扩展到多维数据。尽管该排序产生了一个被称为可学习的布局，但直接学习n维点到排序位置的投影的复杂性随着维度的增加而增加。即使通过限制用于分区的维数可以避免增加的复杂性，但这会导致包括缺失维度的范围查询执行速度较慢。

提供一个可以有意义地排序的多维点的廉价表示已经得到了广泛研究，例如通过Z-order[6]、Hilbert order[5]或与参考点的相应距离对数据进行预排序[2]。学习型的Z-order模型[8]专注于将Z-order缩放与分阶段学得的模型结合起来，以有效地回答空间查询。尽管适用于较小的维度，但在处理更大数量的维度时，无论是Z-order模型还是UB-Tree，在实验和直接比较中都存在一定的限制，这将在实验和直接比较中进行分析。

**3.ML-INDEX**

ML-Index 是两个主要组件的组合，如图 1 所示。它的创建分两个阶段进行，以先前存在的 iDistance 索引 [2] 的思想为指导和推广。上半部分由**一组参考点**组成，负责将多维数据缩放为一维值，便于排序; 下半部分是用于**学习缩放值分布**的学习模型，以及用于搜索和存储数据的排序数组。



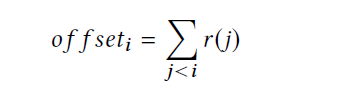
**3.1缩放方法**

考虑数据点集dl∈D,dl=(d0,d1, ...,dn)在n维度量空间. 第一阶段负责创建上部部分，将数据点从 n 映射到一维数据空间。缩放方法旨在将彼此相似的点分组并将它们投影到一维，从而通过一维值的接近度来保留相似性。为了有效地进行缩放，选择 m 个参考点 Oi，每个参考点代表分区 Pi 中数据的标识或质心。 分区 Pi 由最接近参考点 Oi 的点组成。 这意味着点 dl 到参考点的最小距离决定了适当的分区。

无论通过哪种方式寻找参考点，我们都会详细说明不同缩放方法的用法，并讨论它们的优点和缺点。 最简单的情况是通过将 dl 映射到一维值 key = dist (Oi ,dl ) 来相对于单个参考点缩放 dl。这只适用于较小的数据集，因为该方法会产生相当大量的误报，这是通过将彼此相距较远的多维点映射到相同值而产生的。

第二种缩放方法是Jagadish等人的iDistance。这种方法将点集dl映射进一个一维键值基于key=i\*c+dist(Oi,dl),其中i是最近的参考点Oi的下标。常数 c 用于将点划分为预定义的范围，并根据其值以不同的方式扩展范围。通过使用常数c,属于Pi分区的点将会被映射进区间[i\*c,(i+1)\*c]。尽管它提供了良好缩放的值并大大减少了误报的数量，但它高度依赖于参数 c。c的值较小可能会导致分区重叠，从而导致不同分区的多维数据点可能映射为相同的缩放值。因此，在搜索时，不必要的检查点的数量将会增加。另一方面，较大的 c 直接影响 ML 索引第二阶段的创建，我们将在第 3.2 节中详细讨论这一点。

为常数 c 找到正确的值，这将不会导致重叠并且具有完美有序的分区，并且它们之间没有间隙，这是不可能的。 这并不奇怪，因为参考点对应于不同大小的分区。 受前文的启发，我们提出了一种新颖的缩放方法，该方法在不同分区的范围之间提供完美的排序并克服重叠问题。新方法称为偏移方法，给定点 dl 及其最近的参考点 Oi ，将缩放值计算为 key = offseti + dist (Oi ,dl )，其中 offseti 对于每个分区 Pi 都是不同的。给定参考点 O1,O2, ...,Om 的任意顺序及其适当的分区 P1、P2、...、Pm，偏移量计算为其先前分区的半径之和：

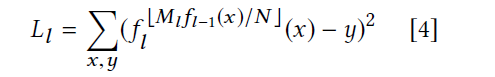


其中 r 是从 Oj 到分区 Pj 中的点的最大距离。 此方法可确保不重叠，并减少分区之间的间隙，这对于 ML-Index 第二阶段的性能至关重要。此外，它省略了调整参数c的问题，以便进行合适的范围创建。与iDistance相比的缺点是由于存储偏移而导致的内存增加不明显。在对原始数据进行缩放之后，每个数据点dl都与一个值键相关联。多个点可以具有相同的缩放值。在执行ML-Index的第二阶段之前，我们根据值键对点进行排序。所得的顺序被用作创建学得模型的起点，该模型能够有效地预测在已排序数组中给定键的位置。

**3.2学习order**

ML-Index的第二阶段代表了一个递归学习型索引，类似于Kraska等人[4]描述的索引。该索引通过将给定的查找键映射到已排序数组中，并保证该键在预测位置[pos - err，pos + err]的接近范围内，模仿传统B-Tree的行为。

正如观察到的[4]，执行这一任务的模型有效地近似了累积分布函数（CDF），建模为p = F(key) \* N，其中N是键的数量，F(key)是估计的CDF，估计了预测小于或等于查找键的键的可能性。学习型索引是以自上而下的方式构建的，其中每个阶段的模型提供一个预测，用于选择下一阶段的模型，或者在达到最终阶段时用于选择键的位置。对于在第l阶段的模型Ml中的模型f(x)，其中输入为x，计算损失：



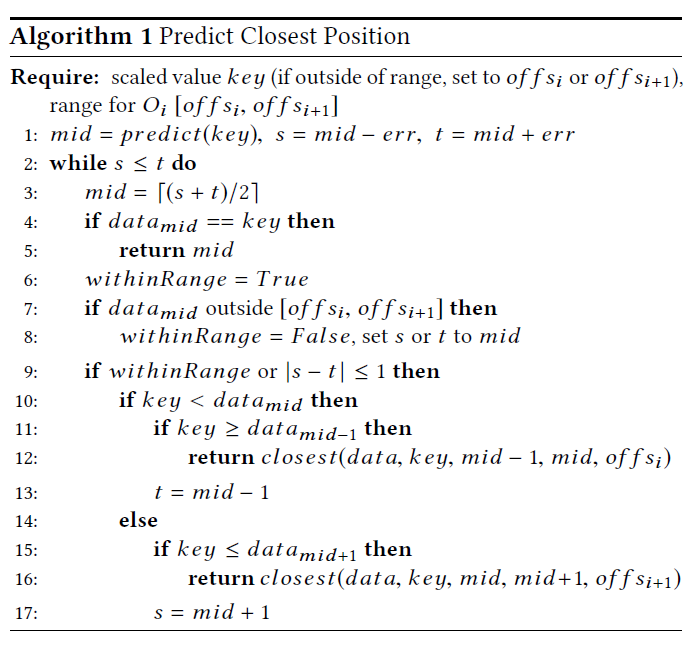
尽管Kraska等人[4]建议使用多个学习型索引阶段，但我们只使用两个。这个决定的动机是，如果第二阶段索引产生了比预期更大的误差，那么在第二阶段增加模型的数量足以减小误差。此外，与他们提出的学习型索引不同，ML-Index中学习模型的第二阶段仅通过使用回归进行构建，目的是平衡构建时间和学习型索引的性能。通过使用一个更简单的模型，搜索时的乘法和加法的数量减少，从而导致更短的搜索时间。学习型模型的最终预测是在已排序数组中的预测位置，其中键（在我们的情况下是缩放后的值）被存储。因为位置是带有一定误差的预测，所以必须在预测周围的误差范围内搜索正确的位置。

如前所述，不同的缩放方法会影响ML-Index的第二阶段。由于朴素的缩放方法不可行，我们仅详细讨论iDistance和offset缩放方法的影响。这两种方法导致需要由学习索引估计的不同函数。在考虑iDistance时，我们区分两种情况，一种是分区之间存在重叠，一种是不存在重叠。当c小于所有分区的最大半径时，分区范围之间的重叠是不可避免的。然而，使用较小的c对数据进行排序会创建一个可以很容易学习的函数。这是由于重叠导致了缩放值的较大密度，从而产生了一个相当连续的函数。第二种情况是使用较大的c，可以避免重叠，但会在函数内创建较大的间隙。因此，神经网络将学习存在于不同范围内的数据。然而，在搜索时可能会搜索到不在这些范围内但位于函数间隙中的点。针对要学习的函数中的缺失值，已经提出了大量研究，然而，这会导致一个可以通过使用offset方法轻松避免的预处理步骤。offset方法规避了由于分区的不同最大尺寸而创建的间隙，并提供了一个更好的“可学习”函数，其中缺失值可能仅出现在聚类中的稀疏性而不是缩放方法。

**4 查询过程**

**点查询:**点查询用于确定索引中是否存在一个多维点，它分为三个步骤，如图1所示。第一步包括搜索查询q最接近的参考点Oi，并计算缩放值key = offseti + dist(ClosestOi, q)，其中offseti为iDistance缩放计算的i \* c，对于offset方法也是相应的。一旦计算出key，就可以用它来预测点q在已排序点数组中的位置。预测值是key在已排序数组中的位置pos，并在其边界[pos - err，pos + err]上进行指数搜索。参考点和d维度的第一步的复杂性为O(m \* d)。学得模型的复杂性取决于神经网络的体系结构。具有宽度h和输入大小N的单隐藏层的神经网络将具有O(hN)的乘法和加法。

**KNN查询**：给定一个查询q和一个参数k，KNN查询找到最接近查询的k个点Sk，使得对于所有的di属于Sk，都存在dj属于D \ Sk，使得dist(q, dj) ≥ dist(q, di)。由于iDistance最初是为执行KNN查询而创建的，我们对ML-Index进行了适应。该算法创建了多个一维范围查询，其选择性扩展，直到结果完整为止。主要的修改是确定范围的起始位置。为此，使用算法1，该算法利用学习模型预测已排序数组中给定键的位置。然而，由于可能提供不在数组中的键，我们需要搜索与初始键的差异最小的键。尽管如此，搜索与初始键最接近的键不仅取决于学习模型误差提供的边界，还取决于参考点所占用的范围的边界。因此，我们修改二分搜索，以包括offseti和offseti+1，从而进一步减小搜索空间。方法closest返回范围[offseti，offseti+1]中最接近键的值的位置。为了描述对第二个边界的需求，我们考虑有两个参考点及其在数组中的相应范围[1, 5][6, 10]。进一步假设第一个范围具有键[1, 3, 5]，第二个范围具有键[7.5, 8, 10]。在搜索时，我们希望找到对于参考点O2，最接近键为key = 6。如果我们不考虑参考点O2具有下界6，我们将检索到键5作为最接近键，但该键不属于区域2，因此在现实中可能与查询点甚至不接近。



范围查询：对于较小的维度而言，范围查询q = q1，q2，...，qn，其中qj = [boundmin，boundmax]，定义了维度j的边界，检索满足对于任意j ∈ n，dij ≥ qj0和dij≤ qj1的数据点di ∈ D。为了在ML-Index中执行范围查询，我们采用了Schuh等人[7]建议的基于数据的范围近似方法。该算法遍历参考点，对于每个参考点，计算给定范围q的最近点和最远点。对于计算得到的最远点和最近点，算法发出一个形式[dist(Oi,pointclosest)+offseti，dist(Oi,pointfurthest)+offseti]的范围查询。由于最接近和最远的点也可能具有不在已排序数组中的键，因此使用Algorithm 1中描述的方法。