# 3. 模型介绍：

本文提出的模型，是在综合考虑了查询准确率，内存占用，以及查询时间三个重要指标的前提下，结合可能面对的现实世界中各种大小的工作负载，以及数据集内部可能存在的函数依赖关系，对于来自同一数据源的不同查询，采用模型自动选择技术，保证每次选取的模型都是最优的，从完成从查询数据输入的key值，到key的最小相关字段position的映射；再从该最小相关字段position到该键值对中与key相匹配的value所实际存放的下标index的位置的映射。从而，我们模型的整体架构分为两大部分。从key到最小相关字段position的映射，称为学习索引部分；从position到index部分的映射，称为索引查询部分。这两大部分采取的核心技术，都是自动模型选择技术。利用自动选择模型，可以只在内存中存储参数，以达到减少内存占用，并在保证准确率的前提下，尽可能减少查询时间。

3.1 自动模型选择技术介绍：

3.1.1 总体流程

本技术旨在通过对于数据的学习，考虑数据实际情况以及数据间可能存在的复杂关系问题，建立数据分布与最适配该数据的模型之间的关系。该技术的主要流程包括：

1. 判断当前状态时训练状态还是自动选择状态，如果是训练状态进入（2）中。如果是训练结束，进入自动选择状态，则进入（7）中
2. 对于读取的数据进行处理，包括分段切片，重分布等操作
3. 处理后的数据进入所有模型用于训练，记录下训练后得出的模型参数以及评估参数，例如均方误差，绝对中值误差等
4. 参数进入打分函数模块计算分数，每个模型对应一个分数。打分函数部分综合考虑了模型方差，误差等因素。通过对于分数进行排序，得到模型的性能排名。
5. 对于当前读入进来进行训练的这部分数据，将其对应的最优模型的序号通过打标签的方式，标注在数据上。
6. 读取下一轮数据，重复前4步，直到所有数据处理完毕，更改当前状态为训练完成，进入自动选择状态。
7. 自动选择状态等待接收经过（2）处理后的数据类型，通过解析数据，可以得到（2）中已经标注好的数据的标签，就是自动选择出来的模型序号，得到序号后，进入对应的模型，利用存储的模型的参数，得到模型的输出结果。结果类型依赖于利用该自动选择模型所进行的映射，映射不同，得到的结果格式也不同。
8. 附加步骤：自动选择完毕后，可能根据自动选择的结果，进一步对于数据进行分类，以便进行同一数据源中其他类型数据的预测。支持的操作比如提供数据接口，以便底层分类器进行训练等。

3.1.2 底层模型简介与优化策略

考虑到该技术实际应用于数据库中查询这一操作，我们访问到的数据，实际上存储在数据库中的相关字段上，与其附近的数据处于一种连续状态，所以我们所需要选择的模型均为回归模型。

对于回归模型的选择需要结合现实世界中实际数据情况进行优化。例如，在本模型中，我们提供了线性回归模型，逻辑回归，多项式回归模型，弹性网络回归，岭回归等模型。这些模型的功能与适用范围有部分交查，但都是必须的。以线性回归，多项式回归与岭回归为例，对于数据之间无关，无函数依赖的数据集而言，三者最终效果差别不大。然而，现实生产生活中的数据，尤其是电子商务等领域的数据表单，属性之间有明显的函数依赖关系。我们通过调研当当网上2018-2019年度售书数据集发现，对于某一种类的图书而言，其售卖数量与销售总金额以及净利润之间，存在着具有可解释性的强依赖关系，在数据集中体现为对于属性，，之间的数据，在进行训练时，其特征变量间存在着高共线性关系。当采用线性回归模型与多项式回归模型进行拟合时，会发现拟合效果并不好，同时为了进行拟合，引起了性能的大幅下降。

对于这一现象，我们采用了加入正则项的方法加以限制。通过在模型的损失函数中加入这一正则项，其中为待优化项，为其系数，损失函数优化为最小时应对进行求导，即，，当且仅当前式取0时，取极小值，从而获取最优解，解的形式为与的代数表达式。通过这种方式向所修改的模型中引入了少量偏差，从而大幅减少模型的方差。当越大，整体方差越小，产生偏差越大。因此，需要人为设定，权衡方差与模型偏差之间的关系。

3.1.3 打分系统简介

为了进一步观测模型的方差与准确率，部分解决上述中的方差与误差的优化决策问题，我们引入了一套打分系统。打分系统的作用包括：1. 通过观测影响模型性能的重要指标，为我们对于超参数的赋值调优提供方向；2. 利用打分，对于模型性能进行排序，以便筛选出针对当前数据源的，相对最优的模型。

打分函数设计：为了综合考虑方差，中值，误差等对于模型性能的影响，我们采取了集几个重要的模型评估指标，进而计算整体的对于单一模型的打分。这些评估指标包括：均方误差RMSE，绝对中值误差MAD，拟合优度等。对于每个指标，我们首先给出他们各自的评分，然后对于每项进行调整，需要归一化的加入正则项，需要调整单调性的如RMSE等，就用表示。然后对于每一项赋予其权重，得到的总体表达式如下所示：

其中：

或：

需要注意的是，我们调整了分数的单调性，使得总体分数越小，说明偏差越小，整体模型效果越好。所以在优化模型时，我们可能会注意到打分的变化，我们应向总体得分最小化的方向进行优化与参数调优。

打分函数计算后，把每个模型的打分存入到列表中。在每轮训练结束后，我们将分数列表进行排序，同时寻找评分最小的下标对应的模型。需要注意的是，在某些情形下，我们可能需要保留多于一个的优秀模型，便于日后的推荐；或者处于标记的考虑，尽可能让各种模型都有被标记的可能。例如，在一轮实验中，我们可能会观察到，使用极限树模型和使用多项式回归模型，二者的得分仅有0.002的差别，那么此时我们应该也保留下多项式回归模型，防止因为过拟合行为对于最终位置的寻找产生影响，同时便于如果出现问题后的核对。因此我们还设置了一个阈值，用于接纳得分与最优模型相近的模型。阈值的设置与选择需要经过人为的参数调优与结合实际情况确定。我们实验中采取的阈值为0.2.即，与最优模型得分差距0.2之内的可以进行进一步考虑。这些被留下的模型的序号将会被标注在数据上。

3.1.4 数据标注方法：

数据标注是实现自动模型选择的一个重要环节，他是从训练完成，到自动选择查询的一个过渡操作。通过将选择好的模型标注在数据上，我们可以在查询时，通过直接解析数据中模型序号标签的位置，获取当前数据应该选择的模型，进而直接采取对应的模型即可完成任务。这一过程的主要操作主要包括对于数据的扩列，分组，标注位置，具体标注的策略等。

首先应该进行数据的分组操作，我们将读取到的数据分成若干组，每次读取read\_line行数（这一参数可以调优）。这样一方面可以避免由于一次性读入大量数据，造成机器性能利用率低，可以高效利用机器缓存；另一方面，标注得到的数据不应该都是选择同一个模型的，应该是面对上文中提及的不同场景，都有对应的不同的适配的模型，但是每一轮训练得到的数据，只能进行一次标注。所以这种分组投入数据的操作是必要的。

在分组之后，数据需要扩列。扩列时有多种选择：可以选择在对应数据中的结尾处，或者开头处进行标注。为了方便处理，我们设定在数据第一列位置处增加一列，用于存放选择的模型序号。处理后，对于读入的数组X，选取read\_line参数和扩列操作后，对于当前读取到第temp行时，读取下一组数据的数据处理为：

需要注意的是，temp+read\_line不能超出数组边界，需要加以约束。同时，读取进来后，进入每个个体模型进行训练之前，需要在当前组的数据中划分出训练集与测试集。选取的read\_line数值应该满足划分的需求，应该是其整数倍。

具体标注阶段，本文制定了相应的策略，使得在保证模型性能的基础上，尽可能的标注更多类型的模型。我们记录下每种模型被标记次数（百分比），并且设定一个阈值，超出这个阈值则表明该模型已经被过量标注了。对于标记数量占据绝对优势的模型，我们应适当减少他们再被标记的数量。我们采用之前模型训练部分介绍过的，保证模型性能的阈值,在此阈值的约束范围内，如果对于现在正在处理的第i组数据，有：

那么就将当前score列表中，排名往下顺延一名的模型的序号，填入到需要填充的位置。填充完成后，至此，所有的训练操作均已结束，下一阶段只需要利用已有的训练好的模型，以及处理后的数据进行快速高效的查询工作即可。

3.1.5 查询过程的具体流程：

与训练时需要寻找源数据与最优适配模型序号的映射不同，查询过程是从已标注数据到模型输出的映射。首先读取工作负载中的查询数据，结束数据获取其第一列，表征为我们标注的模型序号。然后跳转进入对应序号的模型，加载已有的模型的参数，数据查询结果即可。具体的查询过程需要结合我们的key to position模型和position to index模型具体情况加一分析。

3.2 模型中key to value 的两个阶段

从key 到value，需要先后哦经历key to position，确定key所在的相关字段的最小长度，然后在此范围内，精确锁定value所在的下标index，即position to index阶段。

3.2.1 key to position阶段

这一阶段用于完成从查询数据输入的key值，到key的最小相关字段position的映射。我们通过将需要训练的key的集合，投入自动模型选择部分，自动选择技术会自动完成数据处理，标注等工作。在查询阶段，数据利用标注的模型，将用户查询的key-value键值对中，value所在的相关字段的范围，进行缩小。这一过程逐步确定value所在的精确位置，实际上进行的是学习索引工作。该阶段查询后的输出，是一个基于误差上界的，包含index\_of\_value的区间。

3.2.2 position阶段

这一阶段需要在position的区间内，精确锁定value的下标：index的位置。我们首先还是采用自动模型选择技术，通过训练position和index的关系，学习到各个组的position应该选取哪种模型会产生最佳适配，并且用3.1阶段中介绍的同样的方式标注在数据集的第一列。注意，这里的数据集是继承自上一阶段处理后的数据集，本身第一列为第一阶段选取的模型的序号。这里需要再增加一列。读取时从第二列开始读取即可。

对于训练过程的优化：与第一阶段不同的是，我们每训练完一次之后，正如开篇的3.1.1中（8）介绍的，我们在底层附加了一个随机森林分类器。该分类器接收经过第二次标注后的数据，按照其在position to index阶段标注的标签进行分类操作，目的是将采用相同类型的数据归属于同一类。根据局部性原理，在工作负载中，相邻的查询的位置很可能是相近的，也就是说，我们在中训练后匹配到的位置，在位置上很可能是是相近的；另一方面，由于我们进行的是同一数据源下的查询工作，来自同一数据源的相邻的查询，即相邻的相关字段，其查询数据的数据分布也是相似的。综上，我们也有理由相信这种用训练相邻的查询所需要的最优模型是相同的。通过进行分类分析，我们可以学习到数据分布与key值所属于的类型，和value所在的相关字段之间的映射关系。

进而，依据这种映射关系，我们可以对于现有的查询进行拓展：对于来自同一数据源的不同数据，我们通过底层的随机森林分类器，根据学习到的数据特征，自动匹配最适的，用于第二阶段的模型。这种方法就可以无需对于这种同源的新数据进行第二轮标注，节省了训练的时间开销的同时，其查询的性能也可以得到保障。

本阶段采用随机森林分类器有以下两点原因：

1. 随机森林可以自动学习特征。结合数据库中数据具有高维，多属性特征复杂，难以抽取，数据间可能存在函数依赖等复杂的数据特征。利用随机森林优化这一特征选择过程，改用调参，例如等参数，优化我们的训练过程
2. 面对现实世界中的工作负载，通常会存在噪声。对于普通的随机森林分类器而言，这种噪声会造成过拟合现象。然而，面向数据库的查询操作，过拟合会使得我们总是采用效果最优的模型，查询到的结果也是更加精确的。因此，这种过拟合现象对于我们当前的应用场景有正反馈作用。

该阶段的查询过程是在接收到position后，经过随机森林分类器，得到该position适配的模型序号，利用选择出的模型，得到到一个预测的index。此时，根据3.1.3中介绍过的，我们会给出一个预测的误差，作为评估标准之一。当这个误差大于了人为设定的误差阈值时，我们就假设认定我们的通过模型得到的查询结果可能出现了错误。此时需要进行回表操作，该阶段的初始数据集中，根据这个value回推key，查看是否相同。

综上，我们两阶段模型的查询流程如下：

1. 读入数据，进入阶段一，解析当前数据，获取最优模型序号
2. 进入最优模型，利用已经存储的模型参数，进行学习索引操作，得到最小相关字段position
3. 进入阶段二，position进入底层随机森林分类器中，输出为该position应该采取的回归模型的序号
4. 数据进入,利用训练好的，得到最终精确位置index。并检查是否超出误差阈值，如果超出了就需要进行纠错操作，例如回表等。

4. 实验

本部分，我们介绍进行实验的一些方法与策略。我们将本文提出的单层的自动选择模型，与2017年kraska在《The Case for Learned Index Structures》中提出的rmi递归索引模型。实验中比较的重要指标包括：索引学习准确率，内存占用，查询时间。其中：

准确率的度量采取打分的模式进行衡量。上述提到的两种学习索引方法，都是基于各种回归模型以及一些样条插值函数进行实现的。为了衡量模型效果，我们设计了打分函数，针对于一些用于分析回归模型的经典评估参数，例如中值，方差等进行了设计，其中包括：均方误差，绝对中值误差等。对于每一参数Si，我们设计打分函数score(i)进行分数评估。在最后计算每个模型Mi整体评分时，在每个参数已有得分的基础上，赋予每个参数权重，调整各项，使得当求和后总体得分越小（接近于0的正数）时，模型效果越好。我们将求得的和作为损失的度量，准确率则是。这一指标是一切指标的前提保障。

内存占用部分，我们对于查询过程进行了内存占用的监视计算。内存占用主要分为两大类：1. 数据读取；2. 利用训练好的模型中已有参数，输出查询结果。其中kraska模型较为特殊，它支持人为选定各层模型，或者由自身选择最优搭配。因此在这两种模式下，内存占用不同。实际上，采取最优搭配的方法在提升了自身准确率的前提下，同时提高了自身的内存占用以及需要额外的查询时间。

查询时间，记录下从数据读取完毕，进入模型进行学习索引过程，到最终得出被查询位置的最小相关字段（学习到的索引）所用的时间。在保障了查询准确率的基础上，必须使得查询时间最小化。需要注意的是，某些优化方法在保证准确率的前提下，牺牲了额外的查询时间。这一操作牺牲部分了正常正确查询的时间，当需要进行纠错，例如回表等对于误差进行处理时，能优化纠错时间。但是在接下来的实验中可以看到，对于大容量的工作负载，绝大多数查询仍是正确的，现有的方法需要在模型性能，查询准确率，纠错方法的选择，以及查询时间四者之间制定合理的权衡策略。

实验环境准备：本对比试验在cpu为i7-9050处理器，GTX1650 GPU，windows10环境下进行实验测试。其中对于kraska的rmi模型的测试，需要配置rust语言环境，在命令行模式下利用cargo命令进行测试并进行实验相关优化参数的设置。

实验数据的处理方法：我们采用了某网站上2018-2019年各种类书籍销售数据，包括书籍名字，书籍单价，书籍所属种类，书籍出售数目，总体销售金额等。数据大小约为300M，涵盖10000000条以上数据，并且处理为整形数据集和浮点型数据集，涵盖了整形以及浮点型的查询需求。我们还提取了wiki百科中2008-2012年间用户查询的记录，数据类型为字符串类型查询，大小约为1.4G。我们对于不同大小的工作负载，针对不同的数据分布生成了不同的数据集，保证涵盖高斯分布，指数分布，均匀分布，随机分布等基本分布类型。至此，我们用于训练输入的数据就从数据类型，数据分布，数据量上，尽可能测试现实世界中的数据集的各种情形。其中，为了增加数据真实性，我们在自主生成数据的过程中也加入了噪声。

不同模型对于数据的处理方式：本文中提出的单层模型学习索引，首先读取上文中提到的全部数据用于训练。我们将输入的数据分成多组，分别进行训练。对于每组读取进来的数据，分别训练我们提供的所有模型。这些模型包括：线性回归，逻辑回归，多项式回归，弹性网络回归，决策树回归，极限树回归，梯度提升树回归，还有一些其他的bagging族的一些回归模型。数据进入模型进行训练，得出每个模型的一系列评估参数。利用这些评估参数，进行打分的计算。对于一轮输入的数据，会得到该轮的所有模型的打分列表。进行排序之后，选取小于人为设定阈值的，几个排序后表现优秀的模型，对于这组数据打上取的模型标号的标签。对于下一轮数据，重复上述过程，直到标注完所有数据。修改后的数据被保留，当查询到来时，直接根据数据的标签，进入对应模型序号的，训练好的模型。利用已有的模型参数，迅速计算出最优模型的输出。

对于kraska的rmi模型，读入数据类型为二进制数据。数据内部有特殊符号作为key（X）和相关字段postion（y）的分隔符。其中X与y是用于拟合模型，进行fit（X，y）操作的两个输入。这种二进制文件读取速度较快，文件处理过程占用内存较小，但是修改，纠错代价消耗更高，需要额外的数据还原过程，回到例如csv等文件形式，进行数据的标注，修改，或者查询后的回表核对，纠错等操作。

查询过程对比实验：

实验中为了与kraska的rmi模型相对比，同时获取多种不同大小的数据集，我们将kraska公开给出的700M的二进制数据集进行还原操作，并从中提取出不同大小的数据集用于训练，大小从10M到100M不等。Kraska中的rmi模型，我们采用其优化参数-optimizer进行优化选择训练的模型，进行查询操作。经过大量实验后发现，采用自主优化模式下，训练得到的rmi模型，每层中的节点所选择的模型主要包括：线性回归模型，三阶样条函数，多项式回归模型（主要为三阶）。并且层数基本稳定在4层。

对于本文中提到的自动模型选择技术，提供了包括线性回归，多项式回归，岭回归，弹性网络回归，梯度提升树，极限树，决策树回归等10种模型以供选择。实验发现：在绝大多数情况下，本文中的模型会选择极限树与决策树回归；多项式回归与弹性网络回归，岭回归其次；排在后面的包括梯度提升树，随机森林回归等，也在少量情况下会被选择。

1. 内存占用对比试验：

影响内存占用的因素主要包括：数据处理，模型大小，结构等。经过内存监视我们发现：对于大小在200M至1500M之间的数据集，本文提出的模型比rmi模型的内存占用低，差值约为154M至47M不等。对于大小超过700M的数据集，这种差值开始增加，从84M逐渐增加，从差值图像上可以看出这种内存占用差值呈现一种单调不减的趋势。

分析可知：

由于rmi在此自动优化模式下训练出来的模型是是一种4层的模型，层内部存储的参数，层之间数据进行传递时的内存占用，会随着数据量的增加而逐渐增大。本文中提到的自动模型选择是一种单层的模型，采用的有较为基础的，内存占用较小的线性回归模型与岭回归模型等，但是也有几个单体占用量较大的模型，主要是bagging族的回归模型。Bagging族的回归模型由于底层决策树的存在，单体内存占用相对较大，这包括极限树模型，梯度提升树等。在数据集大小小于700M时，这种单体模型内存占用较大的现象占据主导地位，导致实验结果发现本模型与rmi的差值并不大；但是在700M之后，大体积的数据集导致rmi的层次结构在传递数据以及数据进入模型，利用参数获取结果的过程中，需要占用更多的内存，从而使得其与本模型内存占用的差值逐渐增大。此外，本实验设计了一个flag点进行校验，即采用未进行自动优化的rmi模型，其层间选择的模型均为线性样条函数和线性回归模型。经过实验发现，这种选择的内存占用，由于模型过于简单，在相同大小的数据集下，内存占用与本文中的模型基本接近。然而，在后续实验中我们可以观测到，这种未进行优化的模型，其查询时间几乎是本文中模型的二倍！这种结果大大降低了其性能，所以总体上，本文中提出的模型在内存占用上具有优势。

1. 准确率对比试验：

我们采用大小从10M至1.4G不等的数据集进行准确率的对比试验，观察的指标为对数平均绝对误差LMAE（avg\_log2\_loss）。我们利用这一误差进行打分，获取其误差分数，然后做差得到其准确率的表达式：，通过实验后发现：在数据集大小约为10M到1100M之间时，本文提出的模型的准确率显著优于rmi模型，最多可高出0.016%。在数据大小为1100M至1500M之间，二者的准确率差值并不明显，不到0.00003。

分析可知：

从个体模型学习效果来看，本文中提出的模型是个体学习能力较强的模型；而rmi中提出的递归层次结构方法，是利用多个个体学习能力较弱的基学习起，进行组合优化的结果。因而，从实验数据角度看，数据量未超过1100M时，数据量较小，所以采用本文中提出的个体学习效果较好的模型，表现更加出色。准确率也明显高于rmi。（这里的明显是相对于已有的高准确率而言的，因为正常情况下一个优秀的学习索引模型，其准确率需要达到0.9986及以上，而本实验中的模型最终在调参优化后，准确率峰值可达到0.99963）。当数据量较大，本文中的模型与rmi模型学习效果均表现优秀，前者比后者越高出0.12‰至0.16‰.

此外，对于部分的误差，我们也需要进行处理，这部分本文中的模型具有优势。原因在于rmi模型读取的数据为二进制数据，需要首先对于数据进行还原，然后进行回表核对，查看key值是否匹配等。这种操作需要额外的时间与内存占用。因而针对rmi模型而言，在对于准确率的补正方面，本文中的模型具有一定优势

1. 查询时间对比试验：

最后，我们对已有的RMI架构以及本文所提出的自动选择模型查询相同数据所花费的时间进行对比实验。已有的RMI架构是一种多层次每层多模型的结构，输入的训练数据将迭代进入各层进行训练学习，每层的训练输出将作为下层的训练输入。我们所提出的自动选择模型是一种单层级多模型结构，查询数据不需要代入自动选择模型中各个回归模型里训练，而只需要在对应分类器中找到最优匹配效果的回归模型，再代入回归模型中，利用已有的模型参数进行计算即可，并且本模型中各个回归模型的性能优于RMI架构中的回归模型。

我们选取大小从10M到750M的数据进行实验。实验发现，在10M至250M之间，二者查询时间差别不大，本文中的模型比rmi快大约250微秒。250M至500M之间，前者与后者的查询时间差值逐渐增大，时间差约有700微秒。500M至700M之间，查询时间差基本稳定在700微秒至800微秒之间，表示二者时间性能已经达到瓶颈。

特别的，对于实验一中提及的设计的flag点处的查询时间，该处数据大小为350M，rmi查询时间为1221微秒，本文中模型查询时间约为540微秒，前者在为rmi未采用优化参数下的运行结果，其查询时间是本文中的模型的二倍。由于在现实世界的生产生活中，查询时间对于整体运行性能至关重要，因此我们有理由相信，利用优化参数下训练的rmi与本文中模型进行对比试验是有意义的。