像松鼠一样跳跃：面向任意时刻随机森林推理的优化执行步骤顺序

1st Daniel Biebert 2nd Christian Hakert 3rd Kay Heider 4th Daniel Kuhse多特蒙德工业大学 多特蒙德工业大学 多特蒙德工业大学 多特蒙德工业大学德国多特蒙德 德国多特蒙德 德国多特蒙德 德国多特蒙德 daniel.biebert@tu-dortmund.de christian.hakert@tu-dortmund.de kay.heider@tu-dortmund.de daniel.kuhse@tu-dortmund.de

5th Sebastian Buschjäger多特蒙德工业大学德国多特蒙德 sebastia n.buschjaeger@tu-dortmund.de

6th Jian-Jia chen多特蒙德工业大学德国多特蒙德 jian-jia.chen@cs.tu-dortmund.de

arXiv:2603.01588v1 [cs.LG] 2 Mar 2026

摘要—**由于决策树和随机森林效率高、体积小，因此它们是广受欢迎的机器学习模型，常用于资源受限系统中的分类。在这类系统中，随机森林推理可用的执行时间可能不足以完成整个模型的执行。理想情况下，应保留已获得的预测置信度。随时算法旨在能够随时中止，同时随着时间推移提供质量不断提升的结果。以往的方法将随机森林实现为以树为粒度的随时算法，在执行完森林中的部分树、但尚未执行全部树时停止。然而，由于决策树在每一步都会细分样本空间，因此在一棵树中每增加一步，预测质量都会有所提升。本文将决策树和随机森林实现为以树中单步为粒度的随时算法。这种方法开辟了定义森林中步骤顺序的设计空间，有望优化平均准确率。我们提出了**最优顺序，**它以指数级运行时间寻找平均准确率最高的步骤顺序；同时提出多项式运行时间的启发式方法**正向松鼠顺序 **和**反向松鼠顺序，**分别通过贪心方式最大化沿树向下和向上执行每个额外步骤时的准确率。评估结果表明，**反向松鼠顺序**的表现**∼ 94% **与**最优顺序 **相当，且 ∼ 99% 与所有其他步骤顺序相当。**

索引词—**随时预测、随机森林**

I. 引言

决策树和随机森林是传统的机器学习模型，经过了广泛研究并得到广泛应用，尤其擅长高效解决表格数据上的分类问题 [11]。它们有许多针对特定用途的变体、实现方式和性能优化版本 [4], [6]。通过剪枝等方法可以缩小模型规模 [5] ，以进一步缩短执行时间。然而，尽管可以通过优化和缩小模型规模来满足任意执行时间预算，但如果无法确定合适的预算，这些方法就会失效。如果没有保证的执行时间预算，就无法得知应将模型规模缩减到何种程度。随时算法旨在确保在这种情况下仍能返回结果。随时算法的核心思想是，无论何时停止执行，都能返回有用的响应。此外，运行时间越长，响应质量越高。已有研究提出了让随机森林以随时算法方式执行的方案[12], [21]。这类实现能够在保留完整模型的同时提前停止。其工作方式是逐棵执行森林中的树，并在执行停止时返回所有已完成树的预测结果。因此，这些方案以树为粒度运行。当前正在执行的树中获得的所有信息都会丢失。

由于决策树的训练方式，预测置信度会随着每个步骤的执行而提高。用于评估数据的时间越长，通常得到的分类就越准确。通过在内部节点中保留有关预测的信息，决策树可以按单步粒度停止，而不必等到整棵树执行完毕。这样，无论执行何时被提前停止，都不会丢失已经获得的信息。

不受必须执行完整决策树这一限制后，便可以在到达叶节点之前在不同树之间跳转。这为优化遍历各棵树的顺序、从而获得更高准确率提供了设计空间。我们假设执行被停止的概率在时间上均匀分布。因此，我们希望优化所有步骤上的平均准确率。在第一种方法中，我们将其称为最优顺序，通过在样本集上最大化平均准确率来确定步骤顺序。在此基础上，我们提出了两种启发式方法，分别称为正向松鼠顺序 和反向*Squirrel*顺序；它们分别以贪心方式最大化下一步或上一步执行时的准确率。

简而言之，我们的贡献如下：

- 我们实现了随机森林的通用任意时刻执行，支持任意步骤组合。
- 我们提出了三种用于优化平均准确率的任意时刻随机森林步骤顺序生成器，分别称为最优顺序、正向松鼠顺序和反向松鼠顺序。
- 我们在多种数据集上进行了实验评估，并与直观的解决方案进行了比较。

II. 相关工作

决策树和随机森林已在文献中得到广泛研究。许多研究致力于优化决策树和随机森林的学习过程 [14]。集成剪枝是一种常用方法，可缩小随机森林的规模，同时保持或提高其预测准确率（近期综述见 [5]）。最常见的剪枝方法分为基于排序的剪枝方法和贪心剪枝方法。基于排序的剪枝方法会为森林中的所有决策树评分，根据分数对树进行排序，并选取评分最高的前 n 棵树 [13], [18], [15], [19]。贪心剪枝策略会在每一轮从剩余的子森林中选出最优的树，以贪心方式最大化准确率，直到选出指定数量的树 [19], [16]。

针对任意时刻决策树和随机森林学习，已有一些专门的研究。Esmeir 等人 [9], [10] 提出了一个训练决策树的框架，训练过程可以在任意时刻停止，同时仍能获得一棵经过训练的树。后续研究则致力于训练适用于任意时刻场景的树。Denninger 等人 [7] 提出了一种逐步训练随机森林的学习算法。Grubb 等人 [12] 进一步创建了一个框架，从一组弱预测器中选择预测器（例如决策树）来执行；即使在执行任意一个入选预测器后停止，也能取得良好性能。Wang 等人[21], 提出了一种类似的方法，该方法会在随机森林中生成树序列，并为早停准则设定阈值。

这些任意时刻策略要么侧重于任意时刻学习算法，要么以树为粒度执行随机森林，而非以树内执行步骤为粒度。这意味着一棵树必须完整执行后，才能将其纳入预测结果。据我们所知，决策树和随机森林尚未适配为能够以单棵树中的单个步骤为粒度，在任意时刻进行预测。

III. 决策树与随机森林

本节介绍决策树和随机森林[3], [2], （由多个决策树组成的集成），并对其进行调整，使其能够随时返回预测结果。

A. 决策树

决策树在样本集上进行训练和测试。每个样本都包含一个特征向量，并被赋予一个类别。给定一个数据集，通常将其划分为训练集 $\mathcal{S}_r$ 和测试集 $\mathcal{S}_t$。与用于超参数调优的验证集类似，我们进一步将数据集划分出第三个排序集$\mathcal{S}_o$，供之后使用 [22]。

更具体地说，决策树是有向无环图的一种，每个节点要么有两个后续节点（称为子节点），要么没有后续节点。每棵树都必须恰好有一个根节点。具有子节点的节点称为内部节点，而没有子节点的节点称为叶节点。内部节点与一对特征索引和分割值相关联。将内部节点的两个子节点分别称为左子节点和右子节点，据此可以定义一条执行规则。对样本进行推理时，如果该特征索引处的特征值小于或等于分割值，下一个节点就是左子节点；否则，下一个节点就是右子节点。当执行到达叶节点时，与该叶节点关联的预测决定分类结果。

B. 随机森林

随机森林由一组决策树构成，各个树的预测结果会合并为一个整体预测。每个叶节点都会返回一个概率向量，将每个类别与其对应的概率关联起来。将所有决策树中所有叶节点的概率向量相加，并选择概率最高的类别，即可得到整体预测。本文的方法将前文所述结构的随机森林作为已知输入。本文使用 sklearn 中常用的标准配置训练森林 [20]。

C. 任意时刻预测

在推理时，完全训练好的决策树通常只在叶节点保留预测向量。然而，这会严重限制其在任意时刻约束下的预测能力，因为只有当树到达叶节点时才能进行预测。因此，我们扩展了这类设计，保留决策树诱导过程中通常产生的预测向量。按照CART算法进行训练时， [3],在每个节点，训练数据集都会沿特征向量某一维上的分割值进行划分。由此得到两个子数据集，分别位于左子节点和右子节点。因此，每个节点都对应于完整训练集的一个子数据集。与叶节点的做法相同，我们利用该子数据集推导预测向量，即其中各类别的经验概率向量。此信息可直接从CART算法中提取 [3]。按照第III-B节所述方法，将多个内部节点的概率向量相加，并返回概率最高的类别，即可得到这些节点的综合预测。因此，任何内部节点也都可用于返回预测结果。

IV. 随机森林中的调度

将随机森林实现为以步骤为粒度的随时算法，开辟了广阔的设计空间。假设每个步骤发生提前中止的概率相同。因此，步骤顺序应最大化平均准确率。下文中的准确率及所有其他指标均使用排序集 $\mathcal{S}_o$ 计算。

A. 直观步骤顺序

一种将森林作为随时算法执行的直观方式，是遵循标准实现通常采用的执行过程。也就是说，执行一棵树，直到到达一个叶节点。之后，推理继续评估下一棵树，直到所有树都到达叶节点。由于这种步骤顺序会走

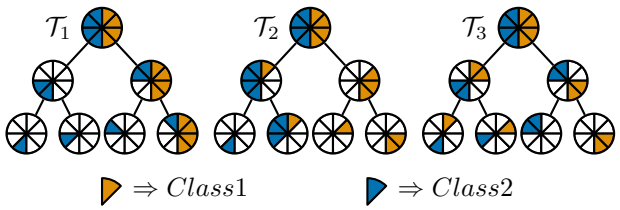


图 1. 标记了样本的示例森林

每棵树的全部深度，下文将其称为深度顺序 。一种同样直观的方法是逐层遍历各棵树，而不是在每棵树中走完全部深度。这类步骤顺序被称为 广度顺序。

为了定义深度顺序和广度顺序的树序列，我们从文献中选取了用于对树对准确率的影响进行排序的方法。Wang 等人提出的一种方法[21], ——*QWYC* 顺序（能停则停）假设，在森林中仅评估部分树后，就能有把握地对某些样本进行分类。*QWYC* 算法会生成一个用于评估树的序列。*QWYC* 算法仅针对二元分类数据集设计。

另一种选择是采用现有的树序列来剪枝森林，如第 II 节 [15], [19], [16]所述。在排序剪枝中，会根据某项指标为每棵树赋分，并选取得分最高的树 [15]。就我们的用途而言，我们按得分对所有树进行排序，并使用这一序列。在采用贪心剪枝时，会根据某项指标（例如个体误差）[19], [16]迭代地选取树。也可以利用树的选取顺序来推导步骤顺序。我们将这样的步骤顺序称为剪枝顺序。需要注意的是，我们会保留所有树，仅使用这些指标来推导序列。生成的序列既可与深度顺序搭配，也可与广度顺序搭配。

B. 最优顺序

直观步骤顺序没有充分发挥以单步粒度执行任意时刻随机森林的全部潜力。森林可以按照任意步骤顺序进行遍历。 $\frac{(d \cdot t)!}{d!^t}$ 种不同的步骤顺序，其中 d 是最大深度，t 是树的数量。每种步骤顺序都有不同的平均准确率。因此，在搜索最优步骤顺序时，穷举搜索是不可行的。不过，可以显著缩小搜索空间。通过将搜索空间解释为加权有向无环图，我们可以使用 Dijkstra 算法 [8]来寻找平均准确率最高的步骤顺序。下面，我们将说明如何构建该图，并使用 Dijkstra 算法寻找最优步骤顺序。为此，我们将使用图 1 所示的示例森林。该示例森林由 3 棵深度为 2 的树组成。排序集 $\mathcal{S}_o$ 由来自两个类别的 8 个样本组成。每个节点中的切片展示了在一次推理中到达该节点的样本。颜色表示类别。

我们首先构建该图。示例森林的图如图2所示。我们将图的顶点建模为森林在推理过程中可能处于的状态。一个状态是

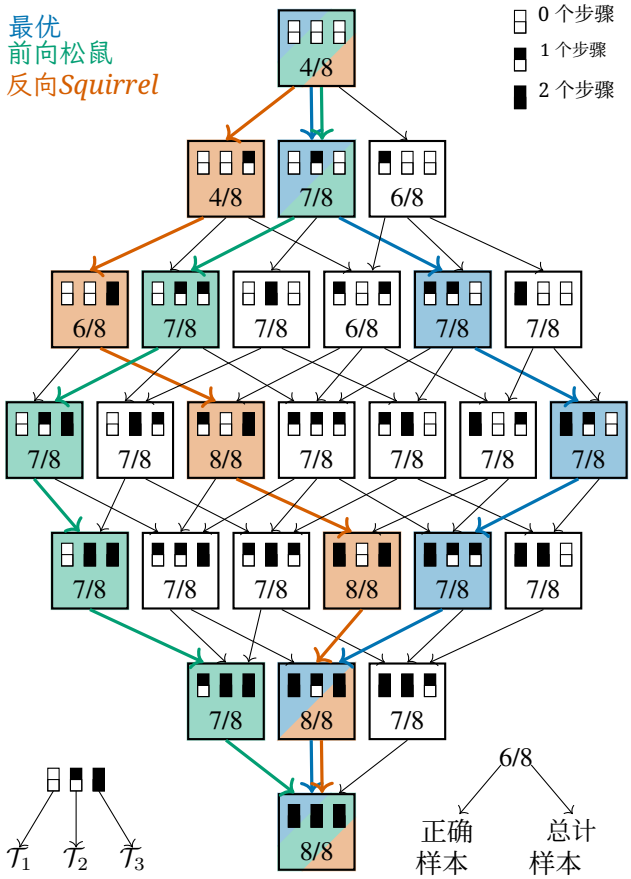


图2。示例森林图

由森林中每棵树已采取的步骤数所定义。在图2中，最左侧顶点所示状态为：第一棵树已采取零个步骤，第二棵树已采取一个步骤，第三棵树已采取两个步骤。所有顶点均通过出边连接到表示在某一棵树中再采取一个步骤的顶点。在该示例中，最左侧顶点连接到第一棵树或第二棵树再多采取一个步骤后的状态。一条边的权重由其指向的状态的准确率决定。状态的准确率使用排序集 $\mathcal{S}_o$计算。在该示例中，最左侧顶点的状态正确分类了八个样本中的七个。因此，所有指向该顶点的边的权重均为$_{7\ 8}$。该图建模了遍历该树的所有不同方式。

最大化平均准确率等同于最小化平均错误率。由于所有步骤顺序的步骤数都相同，我们可以直接使用错误率之和，而不是平均错误率。我们的目标可以改写为最小化所有步骤的错误率之和。

为了找到最优顺序，我们现在使用 Dijkstra 算法[8] ，其中起始顶点表示尚未执行任何步骤，终止顶点表示已执行所有步骤。我们将权重改为准确率的倒数，以最小化错误率。由于 Dijkstra 算法会在图中找到最短路径，因此我们可以保证找到平均错误率最低、从而平均准确率最高的步骤顺序。

最优顺序 在示例中以蓝色标出，见图2。平均准确率为 $\frac{6}{7}$ 或 $\sim 86\%$。图的大小为 $(d+1)^t$。因此，Dijkstra算法的最坏情况运行时间为 $\mathcal{O}((d+1)^t \cdot \log((d+1)^t))$ [8]。

C. *Squirrel*顺序

虽然最优顺序能保证排序集中获得最大的平均准确率，但对于规模不断增大的森林，无法在可行的时间内计算出该顺序。因此，我们提出了一种多项式运行时间的启发式方法*Squirrel*顺序。我们区分前向松鼠顺序和反向松鼠顺序。两者遵循相同的原理，不同之处在于生成步骤顺序的方向。

*Squirrel*顺序的核心思想是对图进行贪心深度优先搜索。前向松鼠顺序从初始状态开始，找出准确率最高的下一个状态。接着，前向松鼠顺序从该状态继续搜索，再次找出准确率最高的后续状态。如此迭代，直到到达最终状态。在图2中，前向松鼠顺序以绿色标出。第一步选择在第二棵树中执行，因为这样得到的准确率最高，为$\frac{7}{8}$。反向松鼠顺序的运作方式非常相似，只是从最终状态开始，沿着树反向搜索，直到到达初始状态。反向松鼠顺序以红色标出。在这个示例中，反向搜索时，它选择将第二棵树作为最后执行一步的树，因为这样倒数第二步的准确率最高，为$\frac{8}{8}$。

尽管*Squirrel*顺序的两种变体在逻辑层面上都作用于图，但关键在于，无需构建完整的图。相反，深度优先搜索会找出 t 下一个状态。只需计算这些状态的准确率。此外，无需存储图，从而减少内存占用。因此，运行时间为 $\mathcal{O}(d \cdot t^2)$[1]。需要注意的是，最优顺序和 *Squirrel* 顺序这两种变体只需在推理前生成一次。生成后，顺序便保持不变，也不会增加推理运行时间。

V. 任意时刻随机森林的实现

虽然在任意时刻约束下确定森林中的步骤顺序，是从随机森林的逻辑视角进行的，但本节讨论的是如何在技术上于正常终止前停止决策树的执行。决策树通常以原生树的形式实现 [1]。在原生树中，树数据存储在数组中，并通过循环遍历数据来构成推理算法。实现任意时刻随机森林需要满足两个条件：

1) 完成前离开并返回树的能力 2) 当前状态需要能够转换为预测

为了在原生树中实现这些条件，推理的状态通过一个索引数组来保存，该数组存储每棵树中当前节点的索引。推理通过遍历一个编码步骤顺序的数组来实现：在下一棵树中执行一步，并更新该树的节点索引。如果提前中止，可使用索引数组加载所有树相应的预测向量，并将它们合并为总体预测。因此，最终实现会紧密循环遍历步骤顺序，每一步推进一棵树。此外，它还支持随时获取预测。

另一种具有竞争力的随机森林实现范式是 if-else 树 [1]。在 if-else 树中，树结构直接编码在源代码中，并通过 if-else 语句实现分支决策。if-else 树也可以实现为随时可用实现。然而，与原生树相比，这需要付出大得多的努力，因为离开并返回树意味着要跳回代码中的相应位置。因此，实现 if-else 树的优化型随时随机森林是一个值得未来研究的课题。

VI. 评估

为了评估以单步粒度将随机森林作为随时算法执行的有效性，以及不同步骤顺序对准确率的影响，我们开展了实验。从 UCI 机器学习资料库 [17] 中选取了 9 个数据集，并使用 sklearn [20]进行训练。这些数据集包括 adult、covertype、letter、magic、MNIST、satlog、sensorless-drive、spambase 和 wearable-body-postures。将数据集随机划分为三个子集：训练集（50%）、排序集（25%）和测试集（25%）。训练子集用于训练随机森林。排序子集用于生成步骤顺序。测试子集用于评估中的推理和准确率计算。对于每个数据集的每种树数量和最大深度组合，使用 5 个不同的随机种子重复实验。实验在两套系统上运行：一套配备 AMD EPYC 7742（简称 EPYC）和 251 GiB 内存，另一套配备 ESP32-S3-WROOM-2（简称 ESP32）。

实验采用了以下步骤顺序：

- 最优顺序。
- 正向松鼠顺序 和反向松鼠顺序。
- 深度剪枝顺序 和 广度剪枝顺序。具体而言，对于基于排名的剪枝，我们使用个体误差 (IE) 和误差歧义(EA) [15]；对于贪心剪枝，我们使用约简误差 (RE) 和 drep (D) [19], [16]。
- *QWYC* 深度顺序 和 *QWYC* 广度顺序 [21]。此外，我们还使用了两种朴素步骤顺序进行比较：
- 非最优顺序，其工作方式与最优顺序 相同，但目标是使平均准确率最小化。
- 随机顺序，它会随机选择下一个步骤。需要注意的是，所有步骤顺序的实现都

u使用以单步为粒度的提前中止。

A. 时间与步骤之间的关系

在本文前面的部分中，森林中单棵树的步骤被视为衡量进度的核心单位。

[1]假设准确率计算的耗时为常数。实际上，准确率计算的开销会随样本数量和树的数量增加。

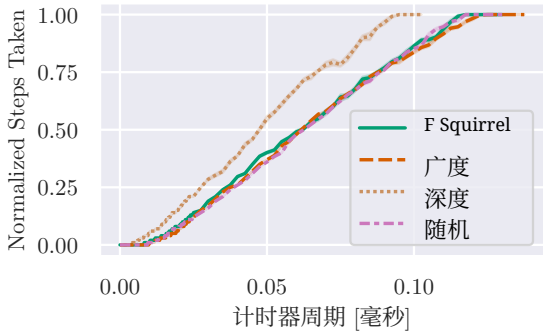


图3. 到期时间与步骤的关系——adult（10棵树，深度10）

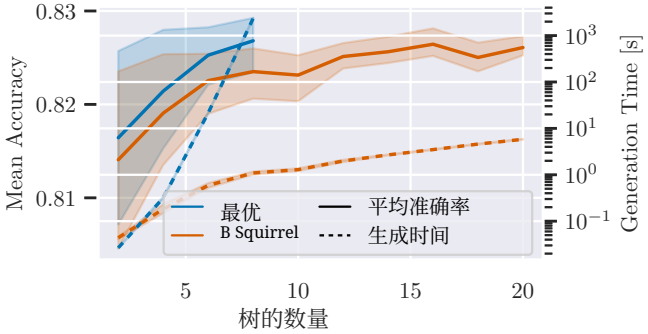


图 4. 步骤顺序运行时间与准确率——adult（深度 8）

步骤这一单位虽然能够与随机森林的结构建立精确联系，却与执行时间没有直接关系。在随时算法的背景下，算法可能会在运行一定时间后被中止。本小节将讨论随机森林中的步骤与执行时间之间的关系。

为了估算步骤与执行时间之间的关系，我们在 ESP32 上进行了实验。ESP32 是一种嵌入式平台。实验中，我们配置了一个计时器，使其在设定的到期时间后中断随机森林的执行。计时器由高精度时钟驱动。我们将该计时器中断视为一个外部事件，它会导致随机森林随时中止。我们记录了计时器中断前森林执行的步骤数，并将其表示为不同计时器周期下执行步骤总数的归一化数值。图 3 展示了横轴上的计时器周期，以及纵轴上对应的已执行步骤归一化数值，并选取了若干种步骤顺序进行比较。所有实验均重复 10 次。图中所示的配置是在 adult 数据集上训练的，包含 10 棵树，最大深度为 10。

实验表明，在平均情况下，步骤数与时间之间基本呈线性关系，因此可以将步骤抽象为时间的度量。虽然不同步骤顺序的执行速度存在细微差异，但这些差异不会影响最优顺序或*Squirrel*顺序的生成，因为两个步骤的执行速度绝不会快于一个步骤。

过度侧重执行时间的评估容易产生受噪声影响的结果。虽然平均而言，计时结果是稳定的，但它无法反映单次执行情况。对比计时器周期相同的两次实验，报告的进度状态可能相差很大。因此，无法随着进度的增加来评估森林趋势。本评估的其余部分将以已执行步骤的数量作为进度指标，研究森林的表现。

B. 步骤顺序生成运行时间

为了分析步骤顺序生成运行时间，我们在adult数据集上进行了一项实验，测量生成步骤顺序所需的时间。我们将最大深度设为 8，并将树的数量从 2 增至 20，步长为 2。我们生成了最优顺序和反向*Squirrel*
顺序 在 EPYC 上，尽可能并行化生成

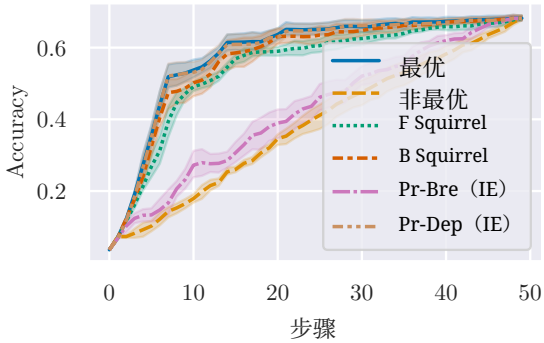


图 5. 步骤数与准确率——letter（7 棵树，深度 7）

可能。在图4中，我们将树的数量绘制在横轴上。我们将平均准确率绘制在左侧纵轴上，将生成运行时间绘制在右侧纵轴上，采用对数刻度。

最优顺序呈指数级生成运行时间。树的数量超过 8 棵后，运行时间和内存占用都变得过大，无法生成最优顺序。反向松鼠顺序的运行时间显著更低，同时平均准确率相近（稍后我们会更仔细地考察准确率）。由于运行时间显著更低，反向松鼠顺序可用于为规模更大、深度更深的森林生成步骤顺序。值得再次指出的是，步骤顺序只需在推理前生成一次，不会影响随机森林推理的运行时间。

C. 步骤与准确率之间的关系

为了深入评估步骤顺序对准确率的影响，我们进行了实验，改变推理中止前执行的步骤数。我们测试了所有步骤顺序，并将树的数量和最大深度分别设为 4 至 7，涵盖所有组合。为了评估较大森林中除最优顺序 之外的所有步骤顺序，我们还进行了未采用最优顺序的额外实验，树的数量分别设为 5、10 和 20，最大深度分别设为 2、5、10 和 20。

首先，我们比较特定配置下步骤数量与准确率的关系。图5展示了在 letter 数据集上训练的、包含 7 棵树且最大深度为 7 的森林的结果。横轴表示步骤数量，纵轴表示所得准确率。目前，我们仅展示部分步骤顺序。

自然地，所有步骤顺序都从相同的准确率出发，并收敛到相同的准确率。对于大多数步骤顺序，只需执行少量步骤，准确率便会迅速提高。*Squirrel*顺序的两个版本、*OptimalOrder*以及剪枝深度顺序 使准确率提升得最快。非最优顺序 和深度剪枝顺序 的斜率则较为平缓。

结果表明，随机森林非常适合以随时算法的方式执行。随着执行步骤的增加，准确率也会提高。这意味着，再执行一步通常就能提高准确率，而花更多时间执行森林则会带来更准确的预测。所选的步骤顺序会对准确率曲线的斜率产生很大影响。

为了比较所有步骤顺序，以下使用平均准确率作为比较指标。我们增加了一个归一化步骤，使不同配置之间具有可比性。我们将平均准确率除以总步骤数与最终准确率的乘积，即假设每一步都达到最终准确率。所得结果称为归一化平均准确率（NMA）。NMA 越高越好。我们在图6中绘制了所有运行配置的 NMA。横轴表示 NMA，纵轴表示不同的数据集。同一树序列的所有深度顺序和广度顺序变体均以相同颜色标示，分别用斜纹和竖纹区分。二元分类数据集带有下划线。

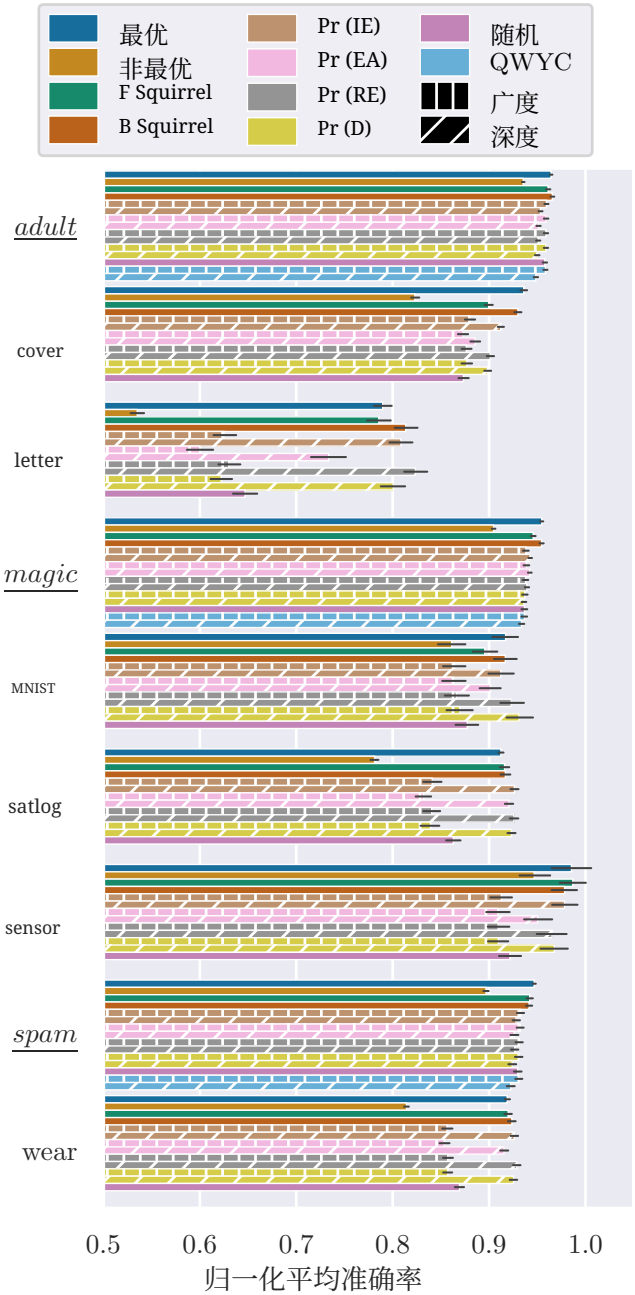


图 6. 所有数据集的平均归一化平均准确率

NMA在不同数据集之间差异很大。因此，斜率的性质不仅取决于步骤顺序，还取决于数据集和森林。在大多数数据集中，最优顺序 表现最佳，而非最优顺序 表现最差。在某些情况下，其他步骤顺序的表现优于最优顺序。虽然最优顺序 使排序集合 $\mathcal{S}_o$的平均准确率达到最优，但它在测试数据集 $\mathcal{S}_t$上的表现可能有所不同。与最优顺序相比，反向松鼠顺序 的表现同样出色，而且通常优于正向松鼠顺序。

在非二分类数据集中，深度变体的表现明显优于广度变体；而在二元分类数据集中则相反。此外，二元分类数据集中的NMA总体范围较小。两类数据集之间的差异可能源于：在非二分类数据集中，要实现置信分类需要更多步骤。因此，这类数据集更倾向于深度变体，因为只有在某一深度，所有类别都已区分。

无论数据集如何，最优顺序 和反向松鼠顺序 都表现良好。它们能够可靠地找到可实现较高 NMA 的步骤顺序。与最优顺序相比，反向松鼠顺序 的表现同样出色，同时生成步骤顺序的速度显著更快。在使用最优顺序的实验中，最优顺序 平均达到最高 NMA 的 ～ 97%。反向松鼠顺序达到最高 NMA 的 ～ 94%。在不使用最优顺序的实验中，反向松鼠顺序 达到最高 NMA 的～ 99%。

VII. 结论

本文表明，保留内部节点中的预测信息后，决策树可以作为随时算法执行。我们进一步表明，可以按单棵树中的单步粒度遍历随机森林，同时仍能返回有意义的预测结果。这由此提出了一个优化问题：寻找一种遍历树的步骤顺序，使平均准确率最大化。我们展示了多种直观策略，用于从文献中类似问题的解推导步骤顺序。我们还提出了如何推导出能使排序数据集上的平均准确率最大化的步骤顺序（最优顺序）。我们将顺序生成运行时间降低至多项式级别

通过启发式方法*Squirrel*顺序，也可以尝试最大化平均准确率。该方法既可沿着叶节点方向应用（正向松鼠顺序），也可朝着根节点方向应用（反向松鼠顺序）。我们表明，随时算法可以高效实现。评估结果表明，*Squirrel*顺序的多项式运行时间使得能够在更短时间内为规模大得多的森林生成步骤顺序。我们还发现，随着执行的步骤增多，任意时刻随机森林的预测准确率会提高，这证明了以单步粒度执行随机森林的有效性。最后，评估结果表明，尽管不同的直观策略在不同数据集上的表现各有优劣，但无论使用哪个数据集，最优顺序和反向松鼠顺序都表现良好。反向松鼠顺序取得了与最优顺序相近的NMA，同时生成它只需多项式运行时间。我们总结认为，随机森林非常适合作为随时算法，并且可以采用反向松鼠顺序，以可靠且高效的方式找到平均准确率较高的步骤顺序。

参考文献

[1] Nima Asadi、Jimmy Lin 和 Arjen P. de Vries。基于树的机器学习模型的运行时间优化。*IEEE Trans. Knowl. Data Eng.*，26(9):2281–2292，2014。[2] Leo Breiman。随机森林。*Mach. Learn.*，45(1):5–32，2001。[3] Leo Breiman、J. H. Friedman、Richard A. Olshen 和 C. J. Stone。分类与回归树。Wadsworth，1984。[4] Sebastian Buschjäger、Kuan-Hsun Chen、Jian-Jia Chen 和 Katharina Morik。通过树框架化实现随机森林的实时评估。载于*IEEE*国际数据挖掘会议，*ICDM 2018*，新加坡，*2018*年*11*月*17*日至*20*日，第19–28页。IEEE计算机学会，2018。[5] Sebastian Buschjäger 和 Katharina Morik。通过 L1 正则化进行联合叶节点细化与集成剪枝。*Data Min. Knowl. Discov.*，37(3):1230–1261，2023。[6] Kuan-Hsun Chen、Chiahui Su、Christian Hakert、Sebastian Buschjäger、Chao-Lin Lee、Jenq-Kuen Lee、Katharina Morik 和 Jian-Jia Chen。决策树实时推理的高效实现。*ACM Trans.Embed. Comput. Syst.*，21(6):68:1–68:26，2022。[7] Maximilian Denninger 和 Rudolph Triebel。对未见类别对象的持续随时学习。载于*2018*年*IEEE/RSJ*智能机器人与系统国际会议，*IROS 2018*，马德里，西班牙，*2018*年*10*月*1*日至*5*日，第4075–4082页。IEEE，2018。[8] Edsger W. Dijkstra。关于与图相关的两个问题的注记。*Numerische Mathematik*，1:269–271，1959。[9] Saher Esmeir 和 Shaul Markovitch。决策树的随时学习。*J. Mach. Learn. Res.*，8:891–933，2007。[10] Saher Esmeir 和 Shaul Markovitch。任意成本分类器的随时学习。*Mach. Learn.*，82(3):445–473，2011。[11] Léo Grinsztajn、Edouard Oyallon 和 Gaël Varoquaux。为什么基于树的模型在典型表格数据上仍优于深度学习？载于 Sanmi Koyejo、S. Mohamed、A. Agarwal、Danielle Belgrave、K. Cho 和 A. Oh 编，神经信息处理系统进展第*35*卷：*2022*年神经信息处理系统年会，*NeurIPS 2022*，*9*，*2022*，202*228* 美国路易斯安那州新奥尔良，*11*月至*12*月。[12] Alexander Grubb 和 Drew Bagnell。Speedboost：具有一致近似最优性的随时预测。载于 Neil D. Lawrence 和 Mark A. Giro- lami 编，第十五届人工智能与统计国际会议论文集，*AISTATS 2012*，*2012*年*4*月*21*日至*23*日，西班牙加那利群岛拉帕尔马岛，《JMLR会议论文集》第22卷，第458–466页。JMLR.org，2012。[13] Huaping Guo、Hongbing Liu、Ran Li、Chang-an Wu、Yibo Guo 和 Mingliang Xu。基于间隔与多样性的集成剪枝排序。*Neurocomputing*，275:237–246，2018。

[14] Tin Kam Ho。随机决策森林。载于第三届文档分析与识别国际会议，*ICDAR 1995*，*1995*年*8*月*14*日至*15*日，加拿大蒙特利尔。第一卷，第278–282页。IEEE计算机学会，1995。[15] Zhengshen Jiang、Hongzhi Liu、Bin Fu 和 Zhonghai Wu。用于分类的广义歧义分解及其在主动学习和无监督集成剪枝中的应用。载于 Satinder Singh 和 Shaul Markovitch 编，第三十一届*AAAI*人工智能会议论文集，*2017*年*2*月*4*日至*9*日，美国加利福尼亚州旧金山，第2073–2079页。AAAI出版社，2017。[16] Nan Li、Yang Yu 和周志华。多样性正则化集成剪枝。载于 Peter A. Flach、Tijl De Bie 和 Nello Cristianini 编，数据库中的机器学习与知识发现——欧洲会议，*ECML PKDD 2012*，*2012*年*9*月*24*日至*28*日，英国布里斯托尔。论文集，第一部分，《计算机科学讲义》第7523卷，第330–345页。施普林格，2012。[17] M. Lichman。UCI 机器学习资料库，2013。[18] Zhenyu Lu、吴信东、Xingquan Zhu 和 Josh C. Bongard。通过个体贡献排序进行集成剪枝。载于 Bharat Rao、Balaji Krishnapuram、Andrew Tomkins 和杨强编，第*16*届*ACM SIGKDD*知识发现与数据挖掘国际会议论文集，*2010*年*7*月*25*日至*28*日，美国华盛顿哥伦比亚特区，第871–880页。美国计算机协会，2010。[19] Dragos D. Margineantu 和 Thomas G. Dietterich。自适应提升剪枝。载于 Douglas H. Fisher 编，第十四届国际机器学习会议论文集（*ICML 1997*），*1997*年*7*月*8*日至*12*日，美国田纳西州纳什维尔，第211–218页。摩根·考夫曼，1997。[20] Fabian Pedregosa、Gaël Varoquaux、Alexandre Gramfort、Vincent Michel、Bertrand Thirion、Olivier Grisel、Mathieu Blondel、Peter Pret- tenhofer、Ron Weiss、Vincent Dubourg、Jake VanderPlas、Alexandre Passos、David Cournapeau、Matthieu Brucher、Matthieu Perrot 和 Edouard Duchesnay。Scikit-learn：Python 中的机器学习。*CoRR*，abs/1201.0490，2012。[21] Serena Lutong Wang、Maya R. Gupta 和 Seungil You。能停则停：通过优化排序高效评估集成。*ACM J. Emerg. Technol. Comput. Syst.*，17(4):55:1–55:20，2021。[22] Yun Xu 和 Royston Goodacre。关于训练集与验证集的划分：比较交叉验证、自助法和系统抽样在估计监督学习泛化性能方面的研究。《分析与测试期刊》，2(3):249–262，2018年7月。