流式大数据随机样本划分模型

赵凌翔

摘要

本文提出一种流式数据随机样本划分（Streaming Random Sample Partition，SRSP）模型。该模型模型可以将一定时间批次内的流式数据表达成一系列 SRSP 数据块的集合，分布式地存储在集群的节点上，这些 SRSP 数据块具有跟该时间批次数据整体相似的统计分布和时序性，可以用于代表该时间段内的数据用于大数据的建模和分析。本文首先介绍了所复现论文的主要工作，然后介绍基于该工作提出的流式大数据处理框架，之后基于该框架详细介绍 SRSP 数据块的生成方法，然后通过真实数据集验证该方法的有效性并基于生成的 SRSP 数据块进行算法实验并分析结果，最后总结本文的工作以及对未来工作的展望。

关键词：大数据；随机样本划分；流式数据

1 引言

在互联网时代，随着数据规模逐渐增大，难以提供足够的存储和计算资源进行大数据的分析，解决这个问题的常用方法是近似计算，通过对大数据进行随机采样获得和原始数据具有相似概率分布的代表数据块，通过用少量的数据可以获得接近全量数据的建模效果，以此来减轻计算压力。信息技术的发展使得数据的产生越来越快，流数据是一组按时空顺序、大量、快速、连续到达的数据序列，在互联网的应用场景中，有很多数据驱动的应用要求系统具备对流数据进行处理的能力，社交网络、电商交易等实时流数据都蕴含着一定的数据价值，对这些数据进行分析可以用于实时决策并提高业务效率，但随着互联网的快速发展，这些场景下实时产生的数据量也在急速增长，故对实时数据进行近似计算分析也可以有效减少计算负担而提升数据分析的效率。

2 相关工作

本节对所复现论文的主要工作以及相关技术的当今研究现状进行介绍。

2.1 随机样本划分模型

目前大数据的储存方法一般是把大数据文件切分成一个个小的数据块存储在分布式文件系统上，而对于大数据分析则是一般采用分而治之的方法来进行分布式并行处理，Hadoop MapReduce 和 Spark 是目前主流的分布式计算框架，MapReduce 理论上可以计算无限大的数据，但是由于 map 操作和 reduce 操作之间的 shuffle 阶段有着大量的磁盘读写操作，导致其迭代算法的效率低。而 Spark 框架由于由于其计算和中间结果均在内存中进行，所以他的计算速度比较快，但当数据量超过内存容量时，算法执行效率将被大大降低，甚至无法运行。面对内存瓶颈问题，采用随机样本数据对大数据进行近似计算是一个有效的途径。作者提出大数据随机样本划分（Random Sample Partition，RSP）模型[1]。RSP 数据模型同样是将大数据划分成小的数据块分布式存储在集群的节点上，但不同于 HDFS 数据块，每个 RSP 块的概率分布与整个数据集的概率分布是高度近似的，因此 RSP 数据块可以用于估计整个数据

的统计信息和建立预测模型，目前该论文已经实现了基于 HDFS 对静态数据进行块级采样生成 RSP 数据块用于数据计算和分析。

2.2 流式数据采样

传统的流式数据采样是通过把流数据全部读进内存中然后使用随机算法进行采样[2]，但随着当今时代流数据越来越多，把所有数据读入内存显然不是很现实。研究[3]提出了蓄水池采样方法，其思想是从 N 个元素中随机地等概率地抽取 k 个元素，对于流数据而言，N 无法确定。研究[4]提出了使用滑动窗口模型进行随机采样，在内存中只保存最近一段时间的数据，提出了对于固定窗口和可变窗口进行随机采样的方法。本文基于复现论文所提出的 RSP 模型，根据分布式流数据处理框架提出了一种流式大数据随机样本划分模型 SRSP，可以分布式地对流数据进行采样。

3 本文方法

3.1 流数据处理框架

随着互联网的发展，在社交媒体、搜索引擎和自动驾驶等领域，不同类型的大规模数据在短时间内持续产生，传统的数据处理系统在性能、延迟和灵活性上很难有效地处理大规模的实时流数据，因此需要一种并行流数据计算框架来对爆炸式增长的流式数据进行处理。基于该现状，本文基于 Apache Spark 提出了如图所示的分布式流数据处理框架，如图 1所示：

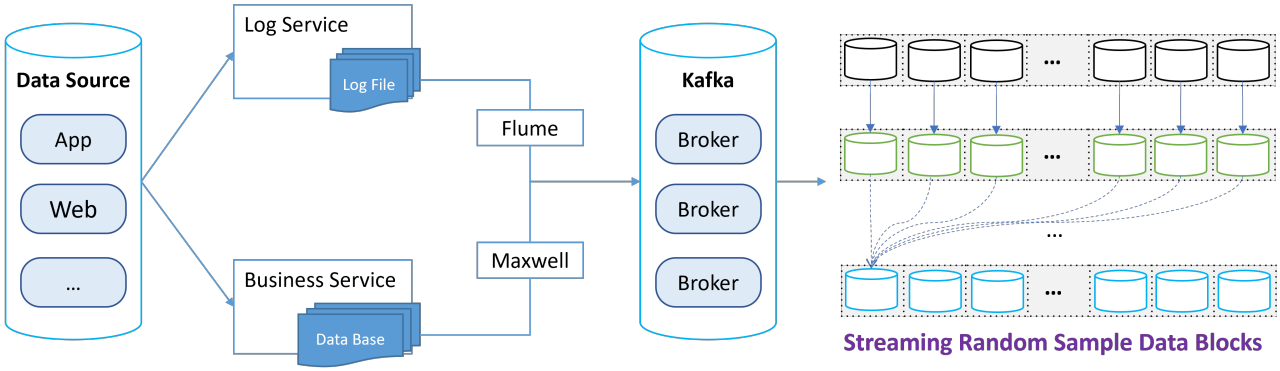


图 1: 流数据处理框架

实际场景中流数据可能来源于不同的设备终端，数据产生之后会来到服务端，根据数据的类型进行划分，流数据可能是实际业务的数据，也有可能是日志数据，这些数据产生后都是先储存在设备的本地文件中，通过使用数据采集框架对这些文件进行监控，例如图中的 Maxwell 和 Flume，监控到文件发生变化就可以将增量数据读取到到 Kafka 消息队列中，通过消息队列与数据处理引擎 Spark 的接口，流数据从数据源最终最后到达采样框架。

3.2 SRSP 采样框架

本框架进行采样的实质是将流数据转化成一定时间批次的数据来处理，工作流程如图 2所示。Spark Streaming 是基于 Spark 的流式批处理引擎，其基本原理是将实时流数据以时间片（秒级）为单位进行拆分，每一个拆分的数据批次都能转换成 Spark 中的 RDD（Resilient Distributed Dataset）数据结构，每个 RDD 含有一段时间间隔内的数据，得到 RDD 之后，Spark 就可以将其转换成 SRSP 数据块存储在 HDFS 中。

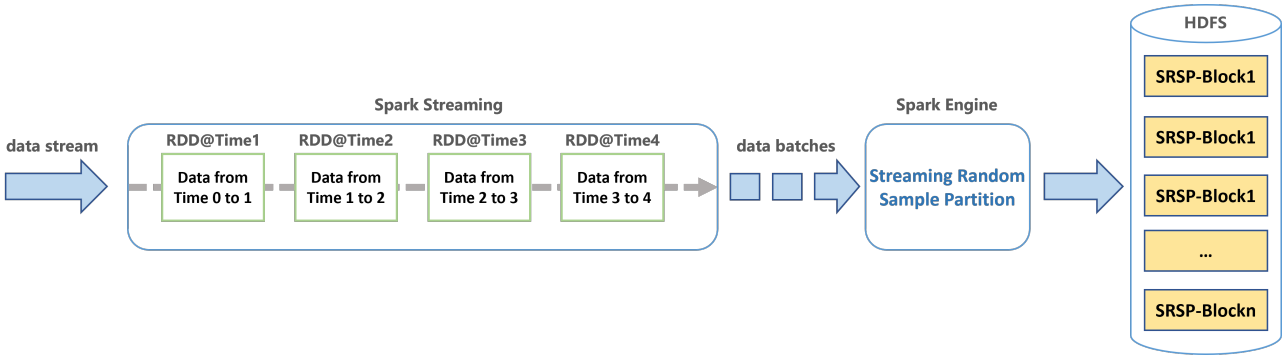


图 2: SRSP 采样框架

4 复现细节

本节对所提出的 SRSP 模型的转换方法进行详细介绍。

4.1 与已有开源代码对比

本文没有引用所复现论文的代码，而是通过其思想将其复现并实现进一步的功能，本文实现流数据的随机样本划分伪代码如下：

Procedure 1 Streaming Random Sample Partition.

Input: streaming files X, partition number N
Output: random sample partition file Y
file number $j = 0$
file rdd set $s = null$
for f ***in*** X **do**
  file rdd r = makeRDD(f)
  for *line* ***in*** r **do**
    random number k = Random(N) $line$ = Tuple(k, Tuple(j, *line*))
  end
  add rdd r *to* s
end
let r' be the union of all the rdds in s
for *line* ***in*** r' **do**
  repartition *line* by the tuple key $line =$ the tuple value of *line*
end
for *partition* ***in*** r' **do**
  sort *partition* by the tuple key of each line in the *partition*
  for *line* ***in*** *partition* **do**
    $line =$ the tuple value of *line*
  end
end
save r' as Y
return Y

通过一个数据流和规定要采样的 SRSP 块数目，可以转换成最终的数据文件 Y，Y 中的分区数就是需要采样的 SRSP 块的数目，每个分区都是该数据流的一个随机样本划分。

4.2 流数据采样过程

本文的实现的流数据采样简化了上述框架中的步骤，跳过中间 Spark Streaming 的 RDD 转换过程，以多个数据块的文件作为多个不同时间批次的流数据进行采样实现，文件依次进行读取成为 RDD 和 Spark Streaming 转 RDD 的本质是一样的。如图 3是采样的实现过程：

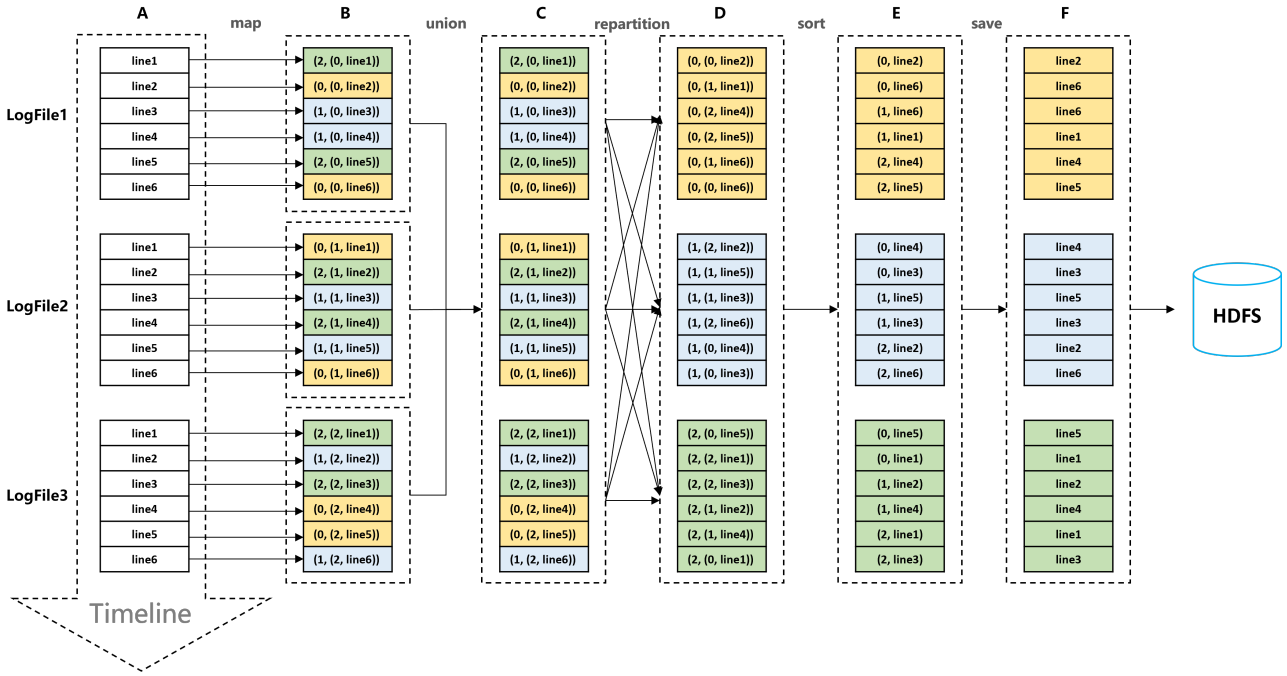


图 3: 流数据采样过程示意图

以采样三个 RSP 数据块为例，图中 A 部分的 LogFile1-3 是三个时间批次的流数据文件，假设 LogFile1 是最早的时间段的数据，LogFile2 和 LogFile3 其次，那么首先先将 LogFile1 进行读入作为 RDD1，之后通过 map 操作，将文件的每一行的数据映射成一个两层嵌套二元组，如图中 B 部分所示，具体方法是，给该文件建立一个标志，比方说从当前文件开始采样，则文件标记为 0，随后的文件依次标记为递增的自然数，以该标记作为 key，每一行原始数据作为 value，构建内层的二元组，然后将该二元组作为 value 值，随机生成的数作为 key 值，构建外层二元组。该随机生成的值与要采样的 SRSP 块数有关，假设要采样 n 个 SRSP 数据块，那么随机生成的值的范围就是从 0 到 n-1。至此，从 A 到 B 的部分处理结束。

当停止读取新文件的时候，执行从 B 到 C 的操作，从 B 到 C 使用的是 union，把 B 中原本独立在内存的各个时间批次的文件合并在了一起，此时多个 RDD 合并成了一个 RDD，其分区数是原本 B 中所有 RDD 的分区数的叠加。接下来，从 C 到 D 是最关键的一步，通过 partition 对数据进行 shuffle，重新分区的数量也就是需要生成的 RSP 数据块的数量，并且这里重新分区的依据是根据 C 部分中每条数据的 key 值，也就是从 A 到 B 随机生成的数值，数值相同的数据会被分到同一个分区，经过这一步，D 中数据的每一个分区实际上已经是原始流数据的一个随机划分样本了，但是，对于流数据而言，时序性是一个重要的特征，但是 Spark 的 shuffle 过程并不能保证分区内的数据是有序的，所以为了保持每隔一个随机划分样本的时序性和原始数据的时序性相似，需要用到内层元组的 Key，即是从 A 到 B 给每个流数据文件分配的标志，在从 D 到 E 的过程中，将分区内的数据根据标志来进行排序，经过排序之后，我们能保证每一个分区内的数据是可以跟原数据保持一定时序性。

至此，流数据的采样已经完成，只需要将元组数据的 key 值去掉把 value 值按分区保存在文件系统上就完成 SRSP 的采样流程。

5 实验结果分析

本节首先对实验平台进行介绍，然后说明实验过程和内容，并对实验结果进行描述分析。

5.1 实验环境与数据集

本文的实验平台为 CDH6.3.2 集群，共有 34 个节点，节点 CPU 为 Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz，集群共 960 个核心，2.50TB 的可用内存。以下的实验基于的软件版本是 Hadoop3.0.0 以及 Spark2.4.0。

实验一共采用了两个真实数据集，如表 1所示，分别是 Covertype 和 HIGGS 数据集，Covertype 是关于植被覆盖的数据集，该数据集包含科罗拉多州罗斯福国家森林四个地区的树木观测结果，样本总数为 581012 ，每个样本有 54 个特征，且有 7 种类型。HIGGS 数据集包含 11000000 个样本，每个对象有 29 列描述，第一列为标签，其他列为特征。

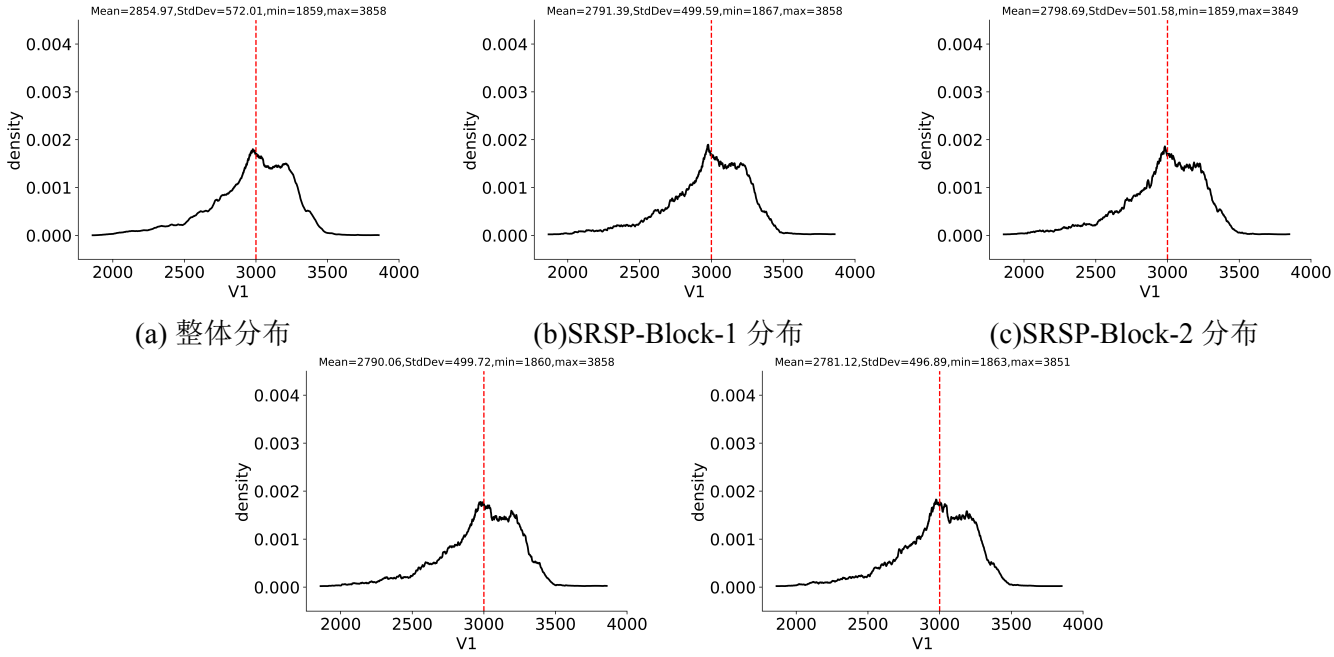
表 1: 数据集介绍

数据集	样本数	块数	每块样本数
Covertype	581012	12	50000
HIGGS	11000000	100	110000

本文将 Covertype 数据集和 HIGGS 数据集分别按次序切分成 12 块和 100 块存储在 HDFS 上，每块的样本数分别约为 50000 和 110000，用于之后的实验。

5.2 随机样本划分

如图 4(a) 是该数据集全量数据维度 V1 的分布，V1 是一个描述森林海拔高度的特征，单位为米。通过本文提出的流式数据随机样本划分方法，将 Covertype 数据集的 12 个块进行 SRSP 转换，得到 12 个 SRSP 数据块。从 12 个块中随机选取 4 个块并画出其 V1 维度分布图如图 4(b)-(e) 所示，可以看到 4 个块在同一维度特征 V1 下的分布是相似的，并且这 4 个块的分布和全量数据的分布也是相似的，所以这些块可以作为整体数据的随机样本使用。



(a) 整体分布　(b)SRSP-Block-1 分布　(c)SRSP-Block-2 分布

(d)SRSP-Block-5 分布　(e)SRSP-Block-9 分布

图 4: SRSP 数据块与大数据整体分布一致性的对比

5.3 近似大数据分析

基于 SRSP 转换方法，我们的近似流式大数据分析实验步骤如下，首先，通过 SRSP 转换将数据集转换成指定数目的的随机样本块，然后随机地取出部分的 SRSP 块用于建模分析，并和使用全量数

据进行计算的结果进行对比。实验分别采用全量数据和随机抽取一个 SRSP 数据块进行分类任务的建模，将所得到的模型对训练数据集进行预测，然后计算预测的准确率。在这个实验中，本文在两个数据集下测试了三个分类任务，分别是决策树、随机森林和逻辑回归。如表 2是全量数据构建的模型和基于单个 SRSP 块构建的模型精度的对比。

表 2: 基于 SRSP 数据块的模型与基于全量数据建模的精度对比

数据集	决策树分类准确率		逻辑回归分类准确率		随机森林分类准确率	
	全量数据	RSP 块	全量数据	RSP 块	全量数据	RSP 块
Covertype(7 分类)	0.773	0.758	0.683	0.683	0.673	0.679
HIGGS(2 分类)	0.705	0.689	0.622	0.621	0.662	0.658

从数据可以看出，基于单个 SRSP 块构建的模型精度和全量数据所构建的模型是非常接近的，但是 SRSP 构建模型所需的数据量远少于全量数据，对于 Covertype 的决策树模型，使用 12 个块中的 1 个块 (8.33%) 可以构建出准确度为 0.758 的模型。

6 总结与展望

本文基于所复现论文提出的 RSP 模型，实现了一种流式数据随机样本划分的方法，得到的 SRSP 数据块具有相似的分布，并且和原始数据的分布也高度相似，用改方法得到的 RSP 块去进行分类任务实验，可以用比原始数据少得多的数据量建立和全量数据准确度相当的预测模型。不足的是由于时间原因，没有完全搭建出流数据处理的框架，而通过多个文件模拟多个时间批次的流数据进行了实现。但是本质是一样的，都是通过将流数据转换成一个时间批次内的文件进行采样处理，故本文后续的工作是配合 Spark Streaming，实现将实时生成的流数据切分成多个小的时间片单元转换成 RDD 数据结构，剩下的工作就是本文已经实现的随机采样过程。

在流数据处理中，效率也是需要考虑的问题，在验证了采样方法的有效性之后，后续工作还需要衡量采样方法的效率。本文还有一些想法尚待实现中，比如面对长时间的采样，应当考虑落盘部分中间结果，减轻集群内存压力，给其他任务留出更多资源。并且流数据的分布可能是不平衡的，有些流数据还存在权重的问题，如何进行更可靠的抽样是未来需要解决的问题。

参考文献

[1] SALLOUM S, HUAN J Z, HE Y. Random Sample Partition: A Distributed Data Model for Big Data Analysis[J]. IEEE transactions on industrial informatics, 2019, 15(11): 5846-5854.

[2] Waksberg, Joseph. Sampling Methods for Random Digit Dialing[J]. Journal of the American Statistical Association, 1978, 73(361): 40-46.

[3] VITTER J S. Random Sampling with a Reservoir[J／OL]. ACM Trans. Math. Softw., 1985, 11(1): 37-57. https://doi.org/10.1145/3147.3165. DOI: 10.1145/3147.3165.

[4] BABCOCK B, DATAR M, MOTWANI R. Sampling from a Moving Window over Streaming Data[C]//SODA '02: Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms. San Francisco, California: Society for Industrial, 2002: 633-634.