1.研究背景

自驱动数据库（Self-Driving Database Management System）是AI for DB领域的一个重要产物，它的一个重要特征是解决人工（如DBA）数据库管理工作的繁杂流程，取而代之构建自调节的一体化过程。这项工作的研究背景是复杂的调优过程带给DBAs的困扰。据统计DBAs花费了将近25%的工作时间在调优活动中，而且在调优活动上的人员配置尝尝占据了DBMS的50%的维护成本。

来自卡耐基梅隆大学的Andrew Pavlo教授是自驱动数据库领域的先行者，它在2017年CIDR会议的一篇论文中，将自驱动数据库要解决的问题分为3个方面：

	Types	Actions
PHYSICAL	Indexes	AddIndex, DropIndex, Rebuild, Convert
	Materialized Views	AddMatView, DropMatView
	Storage Layout	Row→Columnar, Columnar→Row, Compress
DATA	Location	MoveUpTier, MoveDownTier, Migrate
	Partitioning	RepartitionTable, ReplicateTable
RUNTIME	Resources	AddNode, RemoveNode
	Configuration Tuning	IncrementKnob, DecrementKnob, SetKnob
	Query Optimizations	CostModelTune, Compilation, Prefetch

- 物理层：

  第一个方面是数据库的物理设计，包括索引（索引推荐、创建与删除）、物理视图（决定何时添加与删除物理视图）、存储布局（设计哪种存储布局，行存、列存以及混合存储，数据压缩）三个方面。

- 数据层：

  第二个方面是数据层，包括数据存放位置（数据上移、下移，数据迁移）与数据分区（垂直分区、水平分区、垂直与水平混合分区、数据是否复制）。

- 运行层：

  第三个方面包括资源估计、配置调优（Knob的参数配置）、查询优化（查询计划优化、join操作优化、代价模型的调优、预处理操作例如表数据特征的预分析）。

  为此，他以研究小组的一个智能数据库成果peloton为例，总结自驱动DBMS的重要结构：

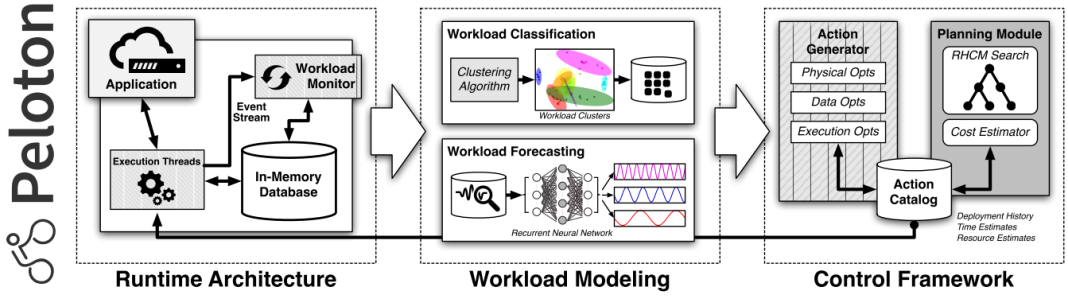


Figure 1: Peloton Self-Driving Architecture – An overview of the forecasting and runtime workflow of Peloton's self-driving components.

Peloton的系统结构包括3个方面运行时体系结构、负载建模、控制框架。这与传统数据库的结构如Postgres、MYSQL相违背，他有很多新颖的特点：首先它使用一种多版本并发控制的变体，在不阻止OLAP查询的条件下交错执行OLTP事务和行动。另外，为了支持HTAP负载，它采用一种具有无锁数据结构和灵活布局的内存存储管理器。总体来说，Peloton的目标是在无人为引导的情况下，保持运行的高性能，也即是提高系统查询的延迟率。具体而言，Peloton包含一个嵌入式的监视器，依据系统内部执行查

询的事件流，记录每个查询条目的资源利用率，构建预测模型，然后识别系统中影响性能的瓶颈问题，如缺少索引、节点过载、分区不合理等等，选择最佳操作，并收集新的监视数据以学习这些操作如何影响其性能。

Peloton是智能数据库的一项重要成果，涉及到数据库的多个组件。而本次大作业主要解决负载建模中的负载预测问题。主要内容为训练预测模型预测不同时间序列下负载的重要信息：如查询特征、查询数目、估计运行时指标等等。

2.问题描述

数据库的工作负载可以简单理解为按时间顺序排列的一系列SQL语句流，SQL从类型上看可划分为3种：SELECT、UPDATE、DELETE，其中UPDATA包括修改数据和插入数据，从表的维度可划分为单表查询和多表查询，从事务的角度看可分为单个查询和多个查询。

用公式可描述为：

$$W = \{Q_{t_1}, Q_{t_2}, \ldots, Q_{t_{n-1}}, Q_{t_n}\} \tag{1}$$

这里，n表示系统运行终止时处理的负载规模，Q_{t_i}表示第i个事务在时间t_i下的基本信息（查询类型、个数、是否跨表等）、运行时信息（运行时长包括开始时间和终止时间、CPU COST、IO COST）。

而本次作业的目标是寻找一个最佳的预测模型f仅预测不同时间点下的负载到达数量，对其他信息则不作考虑，可公式描述：

$$N_{Q_t} = f(E, W_o, t) \tag{2}$$

这里，E表示负载训练数据时的系统环境，W_o表示训练模型所用的负载时序数据，t表示未来任意的时间点，N_{Q_t}代表系统在时间t到达的查询数量。

3.相关工作

关于负载预测的相关工作，我在之前实验室的自然科学基金项目中调研过，并做了详细的总结，下面将展现具体的调研结果：

负载预测技术相关的工作可以分为如下几类：资源估计和弹性伸缩，性能诊断，负载变化检测，查询负载的指标预测以及工作负载特征化等。

资源估计和弹性伸缩是指自动识别工作负载的趋势并扩展资源以进行配置，这里已经有许多研究工作：Gong等人[Gong,2010]提出的PRESS方法通过在应用程序资源需求中提取细粒度的动态模式，自动调整系统的资源分配，通过利用轻量级信号处理和统计学习算法来实现动态应用程序资源需求的在线预测。Roy等人[Roy,2011]通过二阶自回归滑动平均模型即ARMA过滤器实现负载预测用于云平台的资源的弹性伸缩。Das等人提出了一种使用人工构建的层次规则为数据库服务提供弹性伸缩的解决方案[Das,2016]，资源需求估算器从DBMS的内部延迟，资源利用率和等待统计信息中得出信号，以确定对资源的需求是高还是低。他们的工作侧重于短期趋势，并单独估计每种资源的需求。以上的方法都估计在未来短期内需要哪种资源。Higginson等人研究分别适用于短期、中期和长期资源需求预测的方法，使用时间序列分析技术来学习主流数据库工作负载的模型，包括趋势和季节性等模式，以解决DBaaS的容量规划问题[Higginson,2020]。

在DBMS性能建模和诊断方面先前的研究情况如下：文献[Holze,2007]提出利用负载自身的周期性来减少系统工作量的思路,并针对负载变化的大小来对负载监控系统进行控制。DBSeer工具能够在给定工作负载变化的情况下回答“what-if”问题，例如估算磁盘I/O以应对将来的工作负载波动，该模型在脱机状态下，基于事务类型对工作负载进行聚类，并根据事务混合预测系统的资源利用率[Mozafari,2013]。DBSherlock是用于事务数据库的诊断工具[Yoon,2016]，它使用因果模型来识别异常性能行为的潜在原因，并提供可视化效果。文献[Huang,2017]提出Profiler工具，通过识别数据延迟方差的来源，确定标准差，最小化方差代替减少延迟率，完成性能可预测性，以便于最终识别事务延迟的主要差异源。

关于负载变化检测的研究工作如下：文献[Holze,2008] [Pavlo,2011]使用Markov模型根据DBMS当前正在执行的语句来预测用户将执行的下一条SQL语句。Du等人将Markov与Petri-net相结合以预测下一批事务[Du,2009]。文献[Holze,2009][Holze,2010]将这些技术与工作负载分类方法结合在一起，以对工作负载的周期性和重复模式进行建模，具体过程如下：[Holze,2010]结合数据库负载变化的规律来对DBMS未来的工作量进行预测，应用[Holze,2009]中的特征提取方法,从查询语句中提取特征,利用分类算法根据特征来做负载分类,将结果存到负载池中并构建N-gram模型用来对当前的负载进行类型匹配，除此之外，还对负载的周期性进行分析,最后根据当前负载的类型和周期性给出未来负载的预测。Lei等人描述了一种在虚拟化环境中检测负载变化的方法，通过扫描在监视间隔期间收集的硬件利用率度量的时间序列，统计平均资源利用率超过预定义阈值的次数，以识别集群利用率不平衡这一重要的工作负载变化[Lei,2014]。Khanna等人提出一种工作负载"指纹"的概念来描述工作负载，主要使用Adaboost集成算法作为工作负载检测模型[Khanna,2014]。Mozaffari等人提出能够对数据库负载连续监测和轻量化分析的反馈控制循环，此循环描述了自调优特性的设计模式，并在数据库调优过程中对工作负载变化进行检测[Mozaffari,2019]。

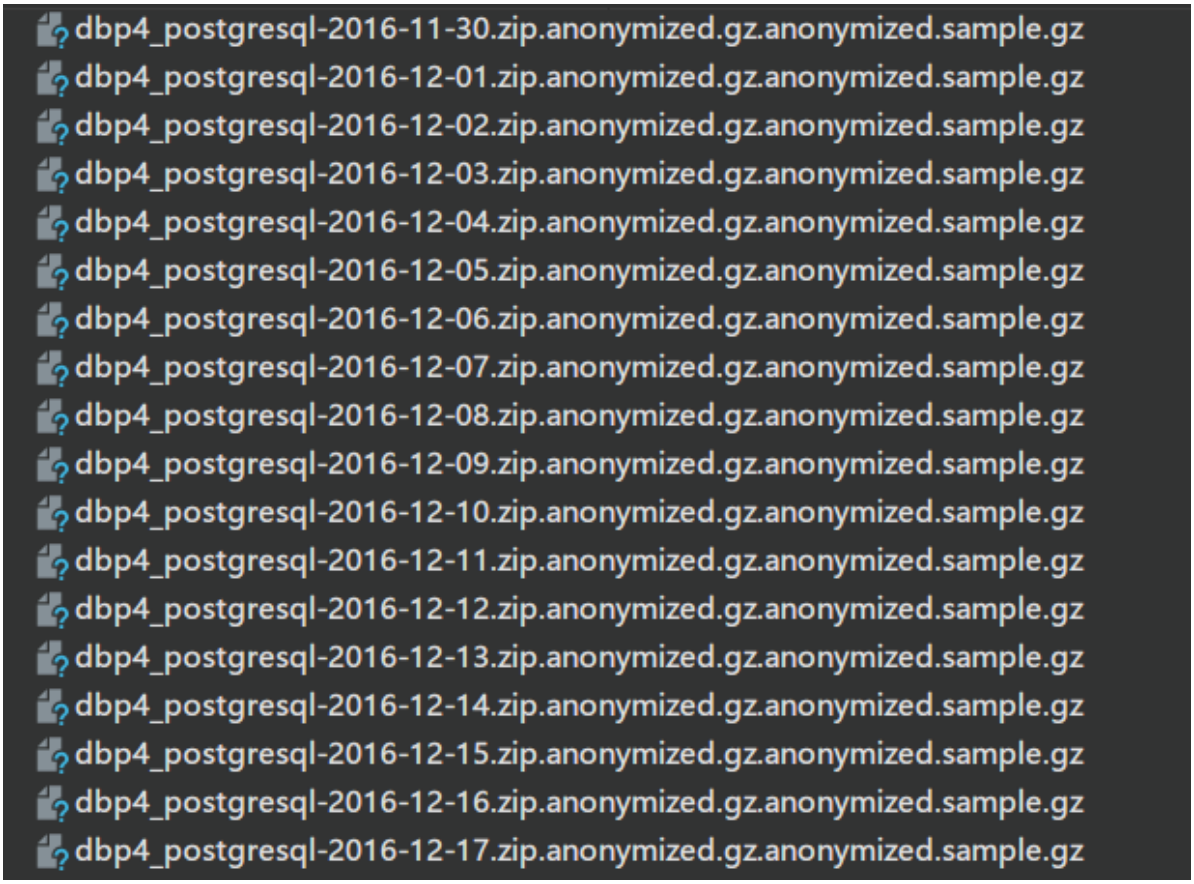
另外一些研究工作预测特定查询的运行时指标：[Gupta,2008]提出的PQR技术使用决策树的变体，使用查询计划和系统负载指标作为输入特征来确定查询等待时间属于哪个存储桶。Ganapathi等人应用核相关分析将查询计划向量和性能指标向量投影到同一子空间中，以估计新查询的指标[Ganapathi,2009]。Popescu等人通过将每个查询分割成几个部分分别使用不同的机器学习模型估计，最后把估计结果插入到全局分析模型中，以预测不同输入数据集上的固定查询集的运行时性能[Popescu,2012]。Zhang利用机器学习技术来预测SPARQL查询的多个性能指标，首先将SPARQL查询转换为向量表示，利用SPARQL查询的语法和结构特征来构建特征向量，提出一个两步预测过程，即考虑冷暖两个阶段的性能，最后，采用SVR和KNN回归模型进行预测[Zhang,2017]。文献[Pei,2019]使用QueryRating来量化查询交互，并引入了一种新的统计模型TMT&TP在不受查询大小的限制下预测查询的响应时间。

对工作负载特征建模进行负载预测的研究情况总结如下：Salza等人假定所有到达系统的事务都属于一组固定的预定义事务类型,使用集合表示法对数据库工作负载进行建模,[Salza,1985] [Salza,1992]。Yu等人将各种收集/汇总的运行时统计信息用于分析SQL语句的结构和复杂性以及工作负载的运行时行为[Yu,1992]。Chaudhuri等人通过压缩一组SQL DML语句，使用任意一对SQL查询q_i和q_j的距离函数$D(q_i, q_j)$,查找要从工作负载中删除的查询，此方法不考虑查询的时间模式[Chaudhuri,2002]。Khan等人使用多时间序列方法对工作负载特征进行建模，首先从集群中得到多个负载信息，记录周期性的负载模式，作为时间序列样本数据，通过聚类技术，识别经常表现出相关负载模式的节点及其处于活动状态的时段，然后通过一种基于隐马尔科夫模型的方法对集群的时间相关性进行表征，预测工作负载模式的变化[Khan,2012]，这种方法不依赖于具体的应用程序，以便于更好地管理其资源，例如，云提供商经常面临资源整合的挑战[Gong,2010] [Sharma,2011]。ARMAs能够捕捉时间序列数据中的线性关系，但往往需要人工决定模型中的差序和项数，此外，由于受到外部因素的影响，线性假设可能对许多数据库工作负载无效，因此，Peloton系统采用RNN的变体LSTM完成负载预测模块，设置多个RNN组，预测不同时间范围和间隔粒度下的工作负载[Pavlo,2017]。

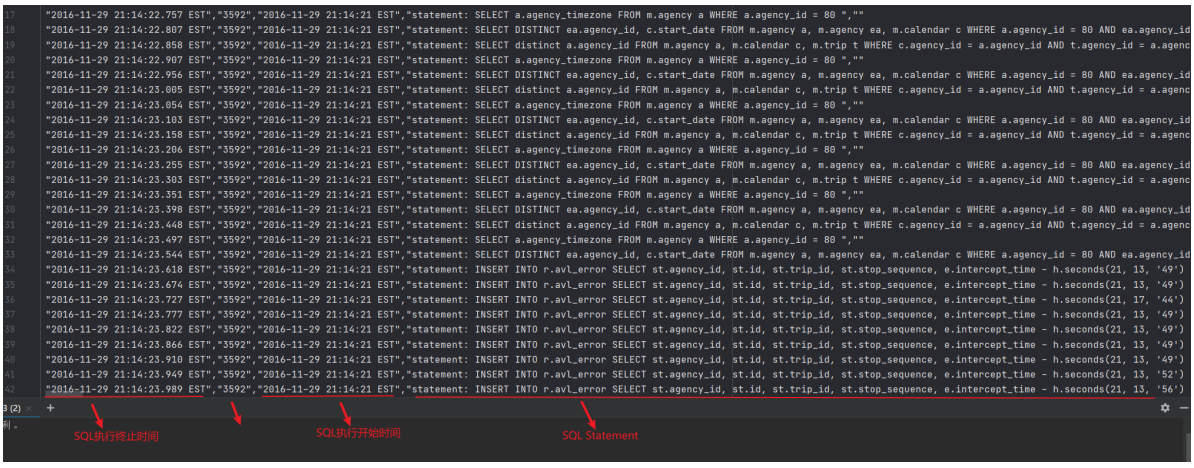
4.数据集

这里用到了一个开放的数据集，来源于一个用于实时跟踪公交系统的app，它会定期从公交系统提取公交车的位置信息，然后引导用户找到附近的公交车站并获取路线信息。

内容包括从2016年11-30日到2017年1月24日共57天的负载数据，每个文件包含8000条SQL数据，负载规模共计48万条数据。



数据集的内容结构比较清晰，一共包括四列，第一列和第三列分别是数据库处理SQL的执行结束时间和开始时间，第四列是具体的SQL语句信息，第二列是事务的ID，同一ID的SQL语句处于同一个事务中。



这里没有给出数据库表的结构信息（字段信息、索引结构、视图等），因为通常这些重要的结构信息对于一个系统来说是绝对机密的。

5.模型构建

通常一个预测问题需要经历数据预处理（数据清洗）、模型验证与选择两个阶段，并辅以一些重要的技术如Hybird方法、Ensemble方法、伪标签技术等等。而本文也将从上述思路展开，并结合数据库的具体场景（SQL特征繁而杂），进行一些细化，下面将给出预测系统的具体结构以及说明为什么要这样设计：

- 预处理模块

该模块主要承担对数据清洗的工作，包括常规的操作如去重、空值填充、异常值检测。首先是去重，由于不同事务中的SQL语句是允许重复的，而事务中SQL是顺序递增执行的，所以同一事务中出现重复的时间段一定是不合理的，属于重复数据。其次空值的处理，如果第四列SQL Statement是空值，则直接删除该条数据，如果第二列事务ID存在空值，要具体判断上一条SQL和下一条SQL的ID是否相同，如果相同，说明该条数据属于该事务中，则对上条事务ID进行复制填充，如果不同，此时无法判断该条数据属于哪个事务中，为了避免对真实数据产生噪音，则直接删除该数据。最后是异常值检测环节，第一列和第三列都是日期格式的字段，第二列事务ID属于整数型数据，而第四列必须是合法的SQL信息，可由语法分析器进行验证。

进行基础的数据清洗工作，一个比较棘手的难题是SQL的结构太复杂了，几乎很难找到重合的SQL，而且SQL的内容和时间点并没有直接的关系，很难找到SQL和时间潜在的关系。但从现实系统的角度分析，对于公交追踪APP系统，有许多实际场景，比如一定有这样一个场景：在上下班时间段，都有很多乘客查询某个地点、去往某地的公交车有多久到达，对于数据库来说，首先要查询行程表中含出发地与目的地的所有的公交车型号，然后查看这些车辆距离乘客有多远的距离，根据平均速度估计时间，如果乘客乘坐了这辆车，就把消费信息和乘坐记录插入对应的表中。对于这种场景，其中涉及到的一组SQL语句（含多个事务）通常是类似的，只是其中SQL中的数值不同罢了，当然任何一个系统肯定有多种场景。为此，第一步要做的是将数值进行剔除，替换成固定的常量符号如"#"，第二步是挖掘潜在的场景，不同系统的应用场景多种多样，而要设计的模型一定要是通用的，为此，需要采取通用的思路找到同一场景的关系。

（注：以上是从功能角度分析的场景，也可以从其他角度，例如：①应用程序新版本或进行了大的更新，涌现了很多新的用户以及老用户的回归，此时系统在版本更新附近的时间段内，负载总量都会得到提高，查询到达率显著提高；②对于公交系统，上下班高峰期是系统负载规模的峰值，其他时间段变化比较缓慢，周一到周五每填24小时的变化规律基本一致。）

- 负载模式提取

  该过程的目的是替换SQL中的变量数值，最好的处理方式是首先结合DBMS's SQL parser和正则表达式进行匹配替换。对不同的SQL语句分析：出现数值的地方通常处于WHERE子句后的键值对中、多表连接时临时为表赋予的别名、UPDATA...SET...语句中的键值对、INSERT子句中VALUES后的元组值等。数值的类型通常是字符型（unicode编码）、日期型、数值型（整形、浮点型）。

- 模式合并

  根据替换变量后的SQL语句，将具有相同语义特征的SQL进行合并，这里并非使用常规的字符串等值匹配，因为不同SQL仍然有很多的结构差异，为此，可采取启发式规则，如果两个模式访问了相同的表、使用完全相同的谓词、投影相同数量和类型的属性，则认为两个SQL具有相同语义特征。

- 聚类模块

  该模块主要解决挖掘应用场景的难题。一种思路是根据SQL执行的物理信息如运行时指标、资源消耗等，但由于数据库（物理环境）的差异，这种方法没有独立于DBMS的配置和硬件，如果发生改变，先前经统计设定的参数将失效。为此，不如从最简单的角度出发：到达率（SQL历史时间段的到达时间序列）。由于每个场景都含有一定数量的模式，且这些模式在过去时间段的周期性一定是相似的，因为它们通常是某个时间点成组出现的，因此计算周期的相似性可以直接通过计算向量的余弦相似度来表示。

  在聚类算法上，这里选择无监督的聚类算法DBSCAN来实现，在此基础上对一些数量级较小且具有一定噪音的聚簇进行修剪。这里对DBSCAN算法进行小的改进，首先在算法的分配规则上选择模板与簇中心节点相似度高的，因为我们仅仅需要将每个模式分配到最近的聚簇中心，而非原始算法中通过计算与聚簇中任意一个对象的最小距离值来判断是否属于该聚簇。第二，在相似度寻找过程中使用kd-tree结构加快查找速度。

- 预测模块

我们使用尽可能多的预测模型和组合方法来挑选最合适的预测方法，将数据集按照3:3:4的比例分为训练集、验证集和测试集，在测试集中均方差越低说明预测效果越好，其中大多数模型已有成熟的库所封装，以下列表是本文所使用的模型及组合：

（1）Autoregressive Moving Average (ARMA)

（2）Predictive State Recurrent Neural Network (PSRNN)

（3）Feed-forward Neural Network (FNN)

（4）Linear Regression (LR)

（5）Recurrent Neural Network (RNN)

（6）Kernel Regression (KR)

（7）Ensemble(LR+RNN)

（8）HYBRID(Ensemble+KR)

补充：Ensemble方法主要有两种策略：

（1）Simple ensemble：

每轮运行过程中，数据被分为两个不重叠的集合，然后分别对模型进行训练，并对两部分中的每一部分进行预测。

（2）Ensemble of specialists：

对于除月度和季度系列之外的所有数据，几个同时训练的模型，分别从系列的不同子集中学习。这时当一个数据集包含大量来自未知来源的序列时，如果对每个组使用一个单独的模型而不是对整个数据集使用一个单独的模型，则总体预测精度将得到提高。该策略的问题是如何执行分组任务。

6.实验

6.1实验环境

操作系统	**linux(18.04.2-Ubuntu)**
CPU	20cores
GPU	Tesla P40*4
内存	264G
IDE	Pycharm
软件环境	python3.6、pytorch1.7.0、cudakit11.0.3

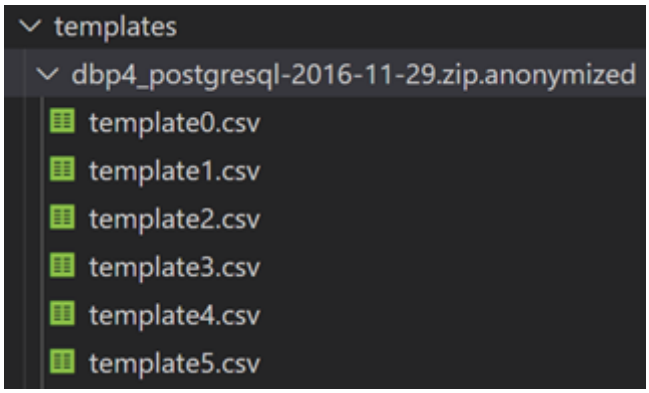
使用每个模式过去1万个时间点的时序数据作为特征向量。

6.2执行过程

（1）预处理

- 数据清洗：处理结果不再展示
- 负载模式提取：将每天的数据集进行模式提取，其中字符串中的每个字符用@代替，数值型数据用#，日期用&&&，提取的每个模式存放到单独的csv文件中，并分类存放到指定的文件夹中，统计模板数量，结果如下：

```
111 tiramisu-sample/dbp5_postgresql-2017-01-24.zip.anonymized.gz.anonymized.sample.gz
Start processing: tiramisu-sample/dbp4_postgresql-2016-12-25.zip.anonymized.gz.anonymized.sample.gz
Generating CSV files...
templates/dbp4_postgresql-2016-12-25.zip.anonymized/
Template count: 31
Start processing: tiramisu-sample/dbp4_postgresql-2016-12-30.zip.anonymized.gz.anonymized.sample.gz
Generating CSV files...
templates/dbp4_postgresql-2016-12-30.zip.anonymized/
Template count: 41
Start processing: tiramisu-sample/dbp4_postgresql-2016-12-24.zip.anonymized.gz.anonymized.sample.gz
Generating CSV files...
templates/dbp4_postgresql-2016-12-24.zip.anonymized/
Template count: 34
Start processing: tiramisu-sample/dbp4_postgresql-2016-12-27.zip.anonymized.gz.anonymized.sample.gz
Generating CSV files...
templates/dbp4_postgresql-2016-12-27.zip.anonymized/
Template count: 34
Start processing: tiramisu-sample/dbp4_postgresql-2016-12-18.zip.anonymized.gz.anonymized.sample.gz
Generating CSV files...
templates/dbp4_postgresql-2016-12-18.zip.anonymized/
Template count: 35
Start processing: tiramisu-sample/dbp4_postgresql-2016-12-21.zip.anonymized.gz.anonymized.sample.gz
```



每个模板的内容如下：（第一行是SQL语句，剩余的是按照每分钟分割的时序数据，第一列表示时间间隔、第二列表示间隔内的模式数量，如果在时间段内，该模式数量为0，则不记录。）

tiramisu-combined-csv > template6.csv

```
1   62434,SELECT COUNT (*) FROM dv.notes_message WHERE user_id = # AND
2   2016-11-29 21:30:00,48
3   2016-11-29 21:31:00,55
4   2016-11-29 21:32:00,46
5   2016-11-29 21:33:00,38
6   2016-11-29 21:34:00,57
7   2016-11-29 21:35:00,66
8   2016-11-29 21:36:00,61
9   2016-11-29 22:00:00,11
10  2016-11-29 23:00:00,5
11  2016-11-29 23:30:00,2
12  2016-11-30 05:00:00,2
```

- 模式和合并

  将不同日期的模式数据按照语义相似度进行合并操作，最后一共得到225个模式，处理日期范围为：2016-11-29 21:13:00到2017-01-25 23:59:00。

```
2016-11-29 21:13:00
2017-01-25 23:59:00
Generating CSV files...
tiramisu-combined-csv/
Template count: 225
template0.csv
template1.csv
template10.csv
template100.csv
template101.csv
template102.csv
template103.csv
template104.csv
template105.csv
template106.csv
template107.csv
template108.csv
template109.csv
template11.csv
template110.csv
template111.csv
template112.csv
template113.csv
template114.csv
```

（2）聚类处理

执行改进的DBScan聚类算法，并构建kdtree结构加速查找过程（模式合并到指定的聚簇中心），同时修剪模式数量少的聚簇。

```
Building kdtree for single point assignment
Finish building kdtree for single point assignment
2016-11-30 21:13:00: template 4 created cluster as 0 with total 94
2016-11-30 21:13:00: template 7 created cluster as 1 with total 117943
2016-11-30 21:13:00: template 8 created cluster as 2 with total 10908
2016-11-30 21:13:00: template 10 created cluster as 3 with total 950
2016-11-30 21:13:00: template 13 created cluster as 4 with total 544
2016-11-30 21:13:00: template 19 created cluster as 5 with total 40
2016-11-30 21:13:00: template 23 created cluster as 6 with total 62434
2016-11-30 21:13:00: template 26 created cluster as 7 with total 45096
2016-11-30 21:13:00: template 27 created cluster as 8 with total 8253
2016-11-30 21:13:00: template 29 joined cluster 1 with total 915189
2016-11-30 21:13:00: template 30 joined cluster 6 with total 62474
2016-11-30 21:13:00: template 33 created cluster as 9 with total 347
2016-11-30 21:13:00: template 41 joined cluster 1 with total 914943
2016-11-30 21:13:00: template 48 created cluster as 10 with total 2766
2016-11-30 21:13:00: template 108 created cluster as 11 with total 369
2016-11-30 21:13:00: template 109 created cluster as 12 with total 269
2016-11-30 21:13:00: template 123 created cluster as 13 with total 188
2016-11-30 21:13:00: template 129 joined cluster 9 with total 28
```

```
Building kdtree for cluster merging
Finish building kdtree for cluster merging
Building kdtree for single point assignment
Finish building kdtree for single point assignment
2016-12-01 21:13:00: template 9 created cluster as 24 with total 139
2016-12-01 21:13:00: template 11 created cluster as 25 with total 135
2016-12-01 21:13:00: template 12 created cluster as 26 with total 31
2016-12-01 21:13:00: template 22 created cluster as 27 with total 84
2016-12-01 21:13:00: template 31 created cluster as 28 with total 1766
2016-12-01 21:13:00: template 40 joined cluster 9 with total 4
2016-12-01 21:13:00: template 42 created cluster as 29 with total 21
2016-12-01 21:13:00: template 46 created cluster as 30 with total 17
2016-12-01 21:13:00: template 60 joined cluster 9 with total 5
2016-12-01 21:13:00: template 61 joined cluster 9 with total 5
2016-12-01 21:13:00: template 64 joined cluster 9 with total 5
2016-12-01 21:13:00: template 66 joined cluster 9 with total 5
2016-12-01 21:13:00: template 95 created cluster as 31 with total 19
2016-12-01 21:13:00: template 102 created cluster as 32 with total 2
2016-12-01 21:13:00: template 107 created cluster as 33 with total 312
```

最终得到3个聚簇，

```
10160   2017-01-18 21:13:00 [(43, 19292820), (1, 2211886), (6, 251491)]
10161   2017-01-19 21:13:00 [(43, 19779424), (1, 2265585), (6, 251491)]
10162   2017-01-20 21:13:00 [(43, 20248313), (1, 2319591), (6, 251493)]
10163   2017-01-21 21:13:00 [(43, 20550987), (1, 2357951), (6, 251494)]
10164   2017-01-22 21:13:00 [(43, 20736924), (1, 2381964), (6, 251495)]
10165   2017-01-23 21:13:00 [(43, 21141911), (1, 2433532), (6, 251496)]
10166   2017-01-24 21:13:00 [(43, 21471633), (1, 2486386), (6, 251497)]
10167   2017-01-25 21:13:00 [(43, 21514883), (1, 2540281), (6, 251497)]
10168   online-clustering-results/None-0.8-assignments.pickle [0.8734053124980079, 0.9817411323138132, 0.9931344075143536]
```

online-clusters > 1.csv

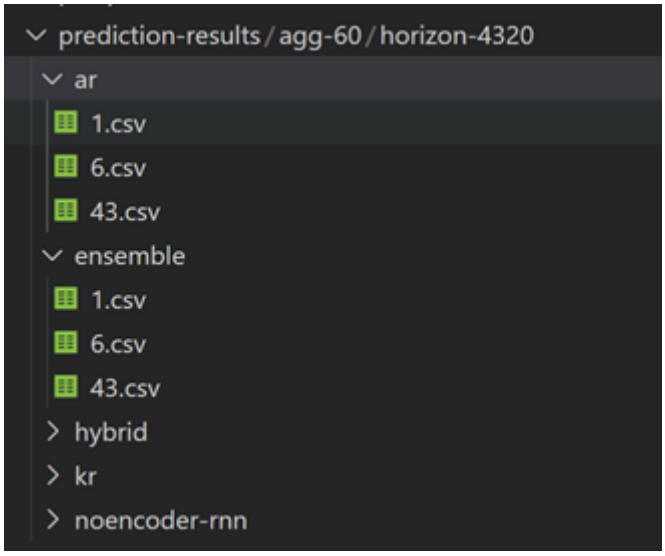
```
1    "2016-11-29 21:20:00","288"
2    "2016-11-29 21:30:00","292"
3    "2016-11-29 21:40:00","368"
4    "2016-11-29 21:50:00","335"
5    "2016-11-29 22:00:00","242"
6    "2016-11-29 22:10:00","316"
7    "2016-11-29 22:20:00","224"
8    "2016-11-29 22:30:00","285"
9    "2016-11-29 22:40:00","272"
10   "2016-11-29 22:50:00","198"
```

```
8219   "2017-01-25 23:00:00","192"
8220   "2017-01-25 23:10:00","249"
8221   "2017-01-25 23:20:00","176"
8222   "2017-01-25 23:30:00","229"
8223   "2017-01-25 23:40:00","222"
8224   "2017-01-25 23:50:00","209"
8225
```

（3）预测算法

选择以下5种预测方法，对聚簇1、6、43的样本数据进行训练集、验证集、测试集的划分，设置epoch=300，分别得到最终的结果：

> **AR**(Autoregressive Moving Average (ARMA)
>
> **Ensemble**(LR+RNN)
>
> **Hybrid** (Ensemble+KR)
>
> **KR**(Kernel Regression)
>
> **Rnn**



例如ensemble方法在聚簇1的结果文件（对到达率y标准化，公式如下）中，第一列数据代表DBMS处理SQL的时间点，第二列、第三列分别表示聚簇1模式实际到达和预测到达的数量。

$$\hat{y} = e^{log(y-y_{min})*y_{std}+\bar{y}} \tag{3}$$

这里，y_{std}代表真实到达率y的标准差。

prediction-results > agg-60 > horizon-4320 > ensemble > 1.csv

```
"2016-12-12 21:13:00","2074.00025","1936.6107278177708"
"2016-12-12 22:13:00","1584.0002","1340.6591577402373"
"2016-12-12 23:13:00","1234.0001000000002","1066.0727613870306"
"2016-12-13 00:13:00","195.00001499999996","160.51830215960354"
"2016-12-13 01:13:00","0.0","0.019024442158200205"
"2016-12-13 02:13:00","0.0","0.0"
"2016-12-13 03:13:00","0.0","0.0"
"2016-12-13 04:13:00","0.0","0.06726780097588803"
"2016-12-13 05:13:00","1012.0001849999999","646.9620309336264"
"2016-12-13 06:13:00","2177.0001","1753.793518186114"
"2016-12-13 07:13:00","3262.9999999999986","2678.968359859906"
"2016-12-13 08:13:00","3719.0008499999994","3336.6162485888262"
"2016-12-13 09:13:00","3041.0011999999997","2966.243977449713"
"2016-12-13 10:13:00","2682.9999000000007","2581.306363037936"
"2016-12-13 11:13:00","2600.0","2478.687500144444"
"2016-12-13 12:13:00","2489.0003500000003","2630.9844864989136"
"2016-12-13 13:13:00","2745.0006000000003","2472.4350084863986"
```

6.3结果分析

将各个模型的MSE结果进行统计，计算不同时区内（从1hour到1week）的平均预测准确率进，其中绿色底色代表同一时区内表现最好的模型。最后对所有时区取均值用黄色底色标注性能最好的方法。

模型 / MSE(log) / 未来时间点	AR	Ensemble	Hybird	KR	RNN
1小时	0.6	0.7	0.9	1	1.7
12小时	1.6	1.7	1.7	1.7	2.5
1天	2.1	1.6	1.7	2.2	1.6
2天	3.1	1.9	1.9	2.7	2
3天	2.9	2.3	2.4	3.2	2.2
5天	3.1	3	3.5	4.3	3.3
1周	2.7	2.9	3	3.3	3.3
Average MSE	2.30	2.01	2.16	2.63	2.37

纵向来看，AR和Ensemble方法都有着较优秀的名次，但Ensemble方法更稳定，无论在哪个时区内，误差都保持较小的数值，排名分别为2、2、1、1、2、1、2，说明复杂的混合模型处理噪音的能力更强。

而AR模型的排名则恰恰相反，为1、1、4、5、4、2、1，时区1天、2天、3天中误差较大，说明时区越小，未来的到达速率与过去接近线性关系，此时AR表现就越好，相反，当时区增大时，到达速率更可能呈非线性关系，RNN的学习效果更好，可以说，AR的准确率在不同时区中是极不稳定的。

此外，运行结果也表明时区的大小对结果误差具有显著的影响，大致成正相关关系，所有方法的MSE都在增大。

7.总结

通过一个学期课程的学习，对CNN、RNN（LSTM、Bert）、GNN、Transformer、Gan等神经网络的原理和实验有了更多的了解和思考，也对Matlab、Pycharm等工具的掌握更加熟练。通过这次实践，对RNN进行预测的原理和代码进行了学习，提高了代码能力，包括pytorch平台的使用、GPU的使用、大规模数据清洗、大规模数据处理（正则表达式、数据合并）等等。

目前，大作业虽然完成了基本的工作负载预测功能，但仍然有许多不足，比如DBScan聚类提高了准确率，但同时降低了预测的精度，类别预测结果是否能够加快数据库的其他组件，还有待探究，目前还有很多可以完善的空间。

参考文献

[Gong,2010]Z. Gong, X. Gu, and J. Wilkes. Press: Predictive elastic resource scaling for cloud systems. In International Conference on Network and Service Management (CNSM), pages 9–16. Ieee, 2010.

[Roy,2011]N. Roy, A. Dubey, and A. Gokhale. Efficient autoscaling in the cloud using predictive models for workload forecasting. In International Conference on Cloud Computing, pages 500–507. IEEE, 2011.

[Das,2016]S. Das, F. Li, V. R. Narasayya, and A. C. König. Automated demand-driven resource scaling in relational database-as-a-service. In Proceedings ofthe 2016 International Conference on Management ofData, pages 1923–1934. ACM, 2016.

[Higginson,2020]Higginson, A.S., Dediu, M., Arsene, O., Paton, N., & Embury, S.M. (2020). Database Workload Capacity Planning using Time Series Analysis and Machine Learning. Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data.

[Holze,2007]Holze M, Ritter N. Towards workload shift detection and prediction for autonomic databases. In: Proc. of the ACM 1st Ph. D. Workshop in Conf. on Information and Knowledge Management. 2007. 109−116.

[Mozafari,2013]B. Mozafari, C. Curino, A. Jindal, and S. Madden. Performance and resource modeling in highly-concurrent oltp workloads. In Proceedings ofthe 2013 International Conference on Management ofdata, pages 301–312. ACM, 2013

[Yoon,2016]D. Y. Yoon, N. Niu, and B. Mozafari. Dbsherlock: A performance diagnostic tool for transactional databases. In Proceedings ofthe 2016 International Conference on Management ofData, pages 1599–1614. ACM, 2016.

[Huang,2017]Huang, J., Mozafari, B., Schoenebeck, G., & Wenisch, T. (2017). A Top-Down Approach to Achieving Performance Predictability in Database Systems. Proceedings of the 2017 ACM International Conference on Management of Data.

[Holze,2008]M. Holze and N. Ritter. Autonomic databases: Detection ofworkload shifts with n-gram-models. In East European Conference on Advances in Databases and Information Systems, pages 127–142. Springer, 2008.

[Pavlo,2011]A. Pavlo, E. P. Jo+nes, and S. Zdonik. On predictive modeling for optimizing transaction execution in parallel OLTP systems. Proc. VLDB Endow., 5:85–96, October 2011.

[Du,2009]N. Du, X. Ye, and J. Wang. Towards workflow-driven database system workload modeling. In Proceedings ofthe Second InternationalWorkshop on Testing Database Systems, page 10. ACM, 2009.

[Holze,2009]M. Holze, C. Gaidies, and N. Ritter. Consistent on-line classification of dbs workload events. In Proceedings ofthe 18th ACM conference on Information and knowledge management, pages 1641–1644. ACM, 2009.

[Holze,2010] M. Holze, A. Haschimi, and N. Ritter. Towards workload-aware self-management: Predicting significant workload shifts. In 26th International Conference on Data EngineeringWorkshops (ICDEW), pages 111–116. IEEE, 2010.

[Lei,2014]Z. Lei, B. Hu, J. Guo, L. Hu, W. Shen, and Y. Lei, “Scalable and efficient workload hotspot detection in virtualized environment,”ClusterComputing, vol. 17, pp. 1253–1264, 2014.

[Khanna,2014]R. Khanna, M. Ganguli, A. Narayan, A. R., and P. Gupta, “Autonomic characterization of workloads using workload fingerprinting,” in In Proceedings of 2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), 2014.

[Mozaffari,2019]Mozaffari, M., Nazemi, E., & Moghadam, A.M. (2019). A New Solution for Workload Change Detection in Self-Tuning NoSQL Database.

[Gupta,2008]C. Gupta, A. Mehta, and U. Dayal. Pqr: Predicting query execution times for autonomous workload management. In International Conference on Autonomic Computing, pages 13–22. IEEE, 2008.

[Ganapathi,2009]A. Ganapathi, H. Kuno, U. Dayal, J. L. Wiener, A. Fox, M. Jordan, and D. Patterson. Predicting multiple metrics for queries: Better decisions enabled by machine learning. In International Conference on Data Engineering, pages 592–603. IEEE, 2009.

[Popescu,2012]Popescu, A., Ercegovac, V., Balmin, A., Branco, M., & Ailamaki, A. (2012). Same Queries, Different Data: Can We Predict Runtime Performance? 2012 IEEE 28th International Conference on Data Engineering Workshops, 275-280.

[Zhang,2017]Zhang, W., Sheng, Q.Z., Qin, Y., Taylor, K., & Yao, L. (2017). Learning-based SPARQL query performance modeling and prediction. World Wide Web, 21, 1015-1035.

[Pei,2019]Pei, Z., Niu, B., Zhang, J., & Amjad, M. (2019). A QueryRating-Based Statistical Model for Predicting Concurrent Query Response Time. WISA.

[Salza,1985] S. Salza and M. Terranova. Workload modeling for relational database systems. In Database Machines, pages 233–255. Springer, 1985.

[Salza,1992] S. Salza and R. Tomasso. A modelling tool for the performance analysis of relational database applications. In Proc. 6th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation, pages 323–337, 1992.

[Yu,1992]P. S. Yu, M.-S. Chen, H.-U. Heiss, and S. Lee. On workload characterization of relational database environments. IEEE Transactions on Software Engineering, 18(4):347–355, 1992.

[Chaudhuri,2002]S. Chaudhuri, A. K. Gupta, and V. Narasayya. Compressing sql workloads. In Proceedings ofthe 2002 International Conference on Management ofData, pages 488–499. ACM, 2002.

[Khan,2012]Khan, A., Yan, X., Tao, S., & Anerousis, N. (2012). Workload characterization and prediction in the cloud: A multiple time series approach. 2012 IEEE Network Operations and Management Symposium, 1287-1294.

[Gong,2010]Gong, Z., & Gu, X. (2010). PAC: Pattern-driven Application Consolidation for Efficient Cloud Computing. 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 24-33.

[Sharma,2011]Sharma, U., Shenoy, P., Sahu, S., & Shaikh, A. (2011). Kingfisher: Cost-aware elasticity in the cloud. 2011 Proceedings IEEE INFOCOM, 206-210.

[Pavlo,2017]Pavlo, A., Angulo, G., Arulraj, J., Lin, H., Lin, J., Ma, L., Menon, P., Mowry, T., Perron, M., Quah, I., Santurkar, S., Tomasic, A., Toor, S., Aken, D.V., Wang, Z., Wu, Y., Xian, R., & Zhang, T. (2017). Self-Driving Database Management Systems. CIDR.