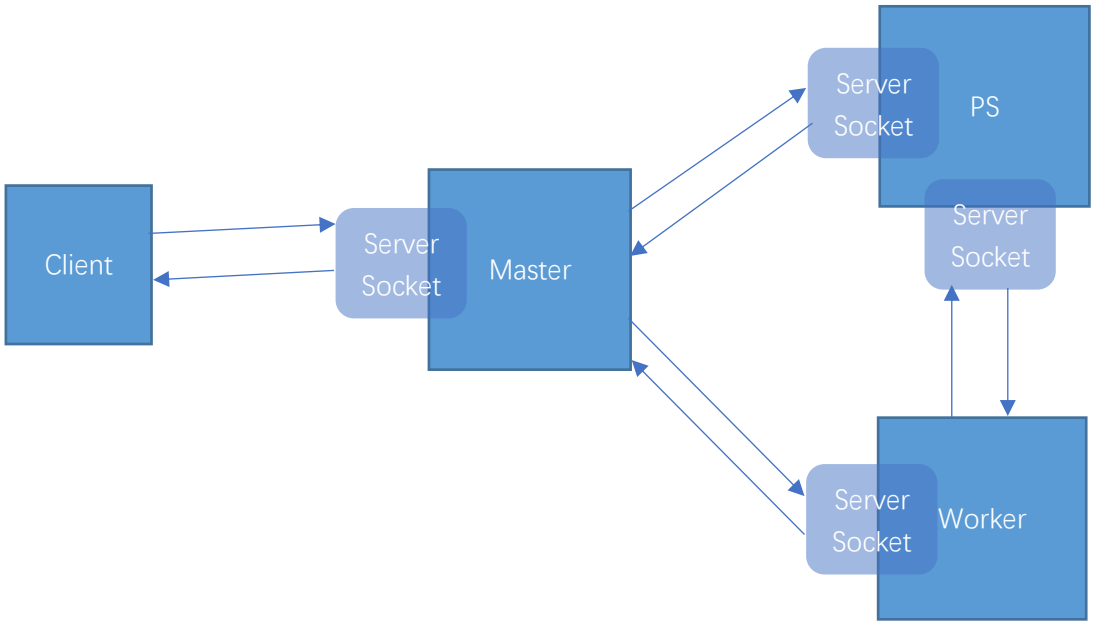
计算机系统工程 Project1 分布式机器学习系统

16302010023 邱轶扬

系统设计架构

本系统主要分为 4 个可执行模块：Client，Master，PS 和 Worker。四个模块可独立运行，通过网络 Socket 接口互相联络交互。



Client 模块负责接收来自用户的请求，生成计算图，通过 Socket 和 Object IO 流传给 Master。最后收取来自 Master 的响应。

Master 模块负责接收来自 Client 的计算图与请求，读取训练集，同时进行调度，将训练集与计算图分配给 PS 与 Worker。本模块通过一个 ServerSocket 监听来自 Client 的请求，需占用一个端口。

PS 模块负责存储计算参数，同时获取 Worker 的计算结果来更新参数。本模块有两个 ServerSocket 套接字，一个负责监听 Master 的连接请求，一个负责监听 Worker 的连接请求，需占用两个端口。

Worker 模块负责从 PS 处获取参数，向计算图注入输入计算结果，训练过程中传给 PS 由 PS 进行参数更新，计算用户请求后将输出传回给 Master。本模块有一个 ServerSocket 负责监听 Master 的连接请求，需占用一个端口。

系统运作流程为 Client 向 Master 发送计算图与输入请求（GRE，GPA 与 rank），Master 读取训练集与调度计算，将计算请求与计算图发给 PS 与 Worker。PS 与 Worker 开始训练，结束之后将输入请求注入计算图得到请求的输出，通过连接传回 Client 输出。

类结构说明

本工程中共有 18 个类，可将其分为四类。

Node 类系列

Node 类系列包括所有抽象类 Node 的子类，他们的作用是描述计算图上的计算节点。每个 Node 都代表了一个变量或一种运算。他是计算图的结构骨架。

每个 Node 都拥有为二维 double 数组类型（矩阵）的 value 属性，value 会在实际运算时被赋值。对于 ValueNode 节点，value 可以直接通过 setter 初始化赋值，而对于运算结点，value 必须通过 compute 方法计算得到。

同时每个 Node 都拥有前向与后向指针，指向数据流动的方向。在计算各节点 value 的时候，会根据前驱 Node 的 value 进行计算。

Framework 类系列

Framework 类系列包括了所有 Framework 类的子类，他们的作用是描述整个计算图。

Framework 中维护着与计算图有关的一切信息。图中的所有节点被分成三个序列维护，Parameters，ServerVariables 和 SendVariables，分别维护着外来接收的参数，自身维护的参数与可传递的参数。在初始化后，调用 computeActualOut 方法可直接计算整张计算图。

FPFramework 与 BPFramework 类分别维护了两种特殊的计算图，神经网络前向计算图与反向传播计算图。FPFramework 维护了整个网络前向的计算流程，而网络的参数值需要外来读取。而 BPFramework 维护了网络反向传播参数更新的计算流程，而用于更新参数的输出误差需要外来读取。

可执行类系列

本工程共四个可执行类，分别代表了工程的四个模块：Client，Master，PS 和 Worker。

Client 中用户会被要求输入网络结构参数与网络输入，然后构建 Framework 计算图，通过 Socket 和 ObjectOutputStream 将计算图和网络输入传给 Master。Master 接收输入之后会与 PS 与 Worker 开始会话，将计算图与输入分配。真正的计算任务交给 PS 与 Worker 执行。

通过对网口参数的修改，这四个程序可分布在不同的计算机上运行，实现分布式计算。

```
E:\java\bin\java.exe "-javaagent:E:\intelliJ\IntelliJ IDEA 2018.1.1\lib\idea_r
Training set has read.
Master waiting for Client...
Client is here!
Master reading finished.
PS got query.
Worker has computed the required output.
Send queryOut to Client.
Master waiting for Client...
Client is here!
Master reading finished.
PS got query.
Worker has computed the required output.
Send queryOut to Client.
```

常量类

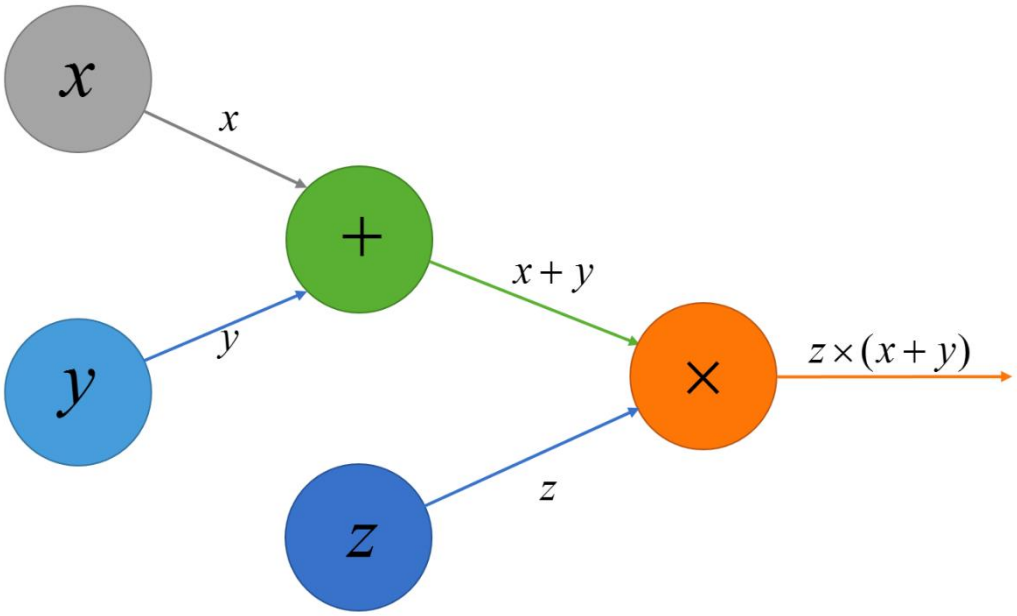
常量类是一个数据资源类，机器学习策略相关参数被记录在其中，如 Batch size，模块间通讯制度参数。

设计说明

Framework 结构设计

在本工程中，我使用了计算图的方式来传递机器学习算法。这一方法能有效地提高算法的复用性与算法结构调整的灵活性。对计算过程建模，可以使系统框架修改余地更加充分，同时能很好适应机器学习训练这类需要大规模重复学习的任务。也使系统对更多元的任务有更好的适应性。

计算图 Framework 的设计体现了普遍计算的复用原则，我将每一个计算与值都抽象为了计算节点。通过初始化值节点的方式开始整个计算流程。



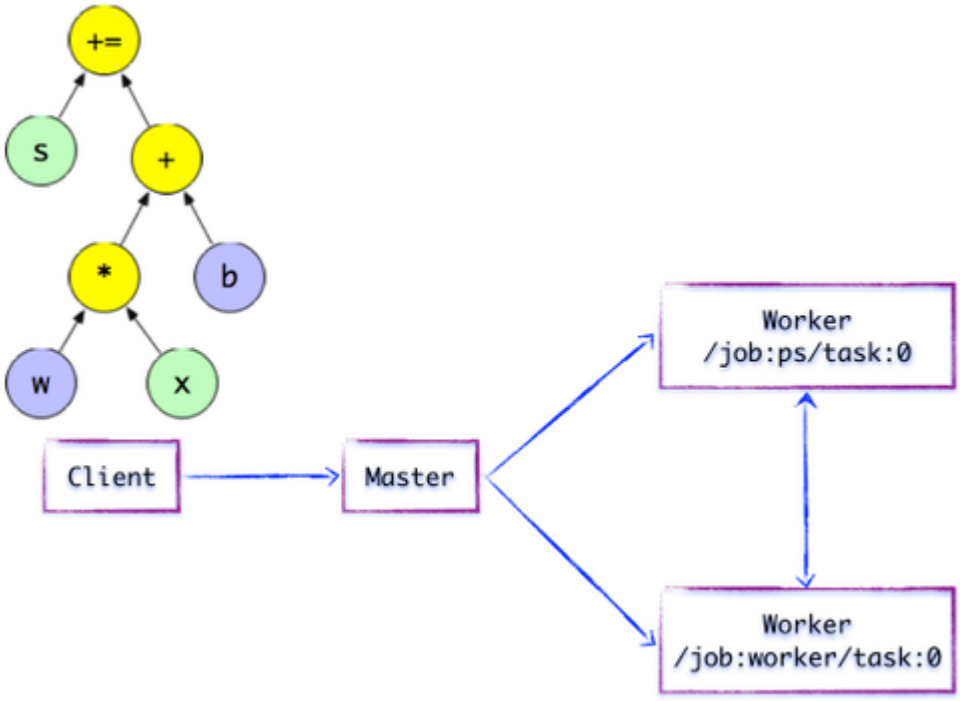
通过节点来抽象计算固然简单，但依然有其问题，譬如说对于一种新的运算必须添加继承 Node 的新类，同时对于这样的图结构很难实现任意结构自动求导自动反向传播优化。但整体上而言，这种设计能胜任神经网的分类任务。

模块通讯结构设计

本工程的各功能模块之间通讯使用了网络套接字，从而使各模块可以独立运行在不同的计算机上。

这样的设计首先很好的分配了算力，使大规模机器学习任务不至于受到计

算机算力瓶颈制约。同时各计算机可以因地制宜，内存空间较大的计算机可作为 PS 存储计算参数，而整体算力较强的计算机可以作为 Worker 负责计算任务。而这样的设计依然有其问题，比如说在多 PS 多 Worker 多 Client 的情况下，Master 依然作为信道瓶颈存在，Master 的并行能力受到极大考验。在如此的 C2S 系统设计下，分布式系统依然会遇到瓶颈。



该工程中共有四个服务器套接字，用于监听请求，分别为 Master 监听 Client 的套接字，Worker 监听 Master 的套接字，PS 监听 Master 的套接字以及 PS 监听 Worker 的套接字，各代表了一条潜在信道。在 PS 与 Worker 之间的通讯方面，我开启了多线程进行交互。

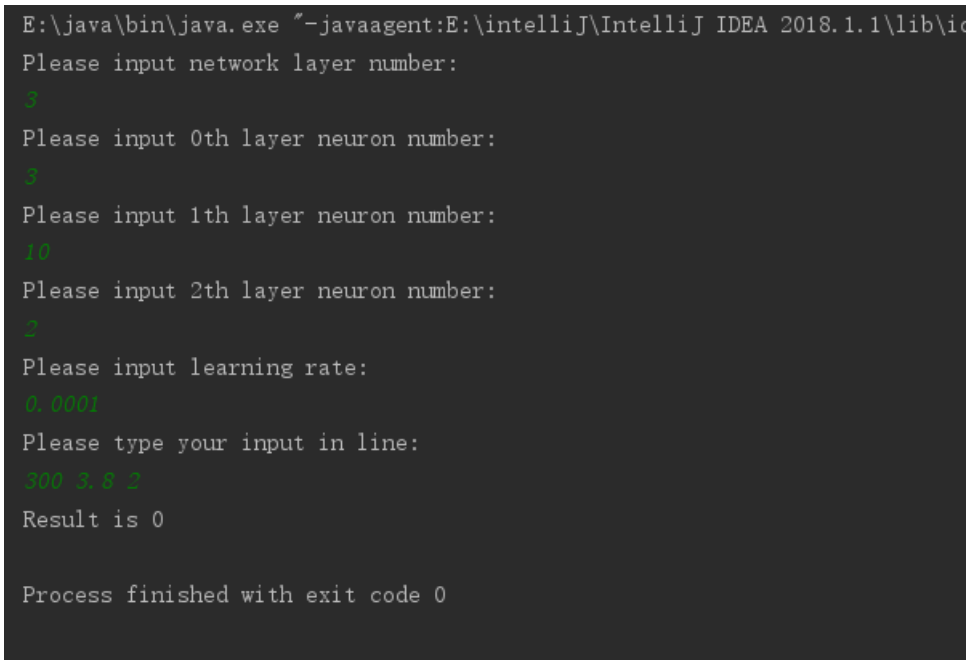
该设计能很好的完成各模块间的合作，可很好分配算力解决大规模学习问题。在此设计中，PS 与 Worker 之间的联系是有主被动之分的，PS 无法主动发起会话而只能被动监听，但由于参数请求本身的被动性，该设计可以说是合理的。

问题反思

计算图建构

在计算图计算流程中，矩阵的流动非常重要。而大部分矩阵运算对于矩阵形状有严苛的要求。这既是计算图设计的困难也是算法流程设计捉虫的关键点。不正确的矩阵形状一般都意味着算法本身流程的问题。

在初次调试时我多次遇上了这样的情况，与此同时 Terminal 会打印出“Failed”。这个 bug 在单进程下调试也较为困难，在网络情况下就难上加难。但通过对计算图的调试追溯，我还是将所有的症结问题解决。



网络套接字编程：端口占用与连接关闭

网络套接字编程中有许多与串行普通编程不同的地方，两方要达成很好的默契，才能达成四次握手。

太快的开启与关闭套接字是有问题的。Linux 系统对端口资源的释放是延时的，在套接字使用了某一端口并关闭之后，系统内核需要 2 分钟才能彻底释放端口并重新加入分配。

在最初，我使用了逐次更新的梯度下降策略，这导致 PS 与 Worker 之间的频繁交互，最终引起系统端口资源紧缺无法分配。于是最终我使用了 mini-batch 策略，批量计算更新参数，大大减少了 PS 与 Worker 的交互频率。

而在之后的尝试中，我需要关闭 PS 方面对 Worker 的监听，对状态进行更新。但由于 PS 作为服务器处于被动态势，监听的关闭还是需要主动方 Worker 来完成。因此我在 IO 流中设定了一个关闭命令码，PS 收到该命令就应该关闭监听，退出循环更新请求状态，进而解决了该问题。

```
E:\java\bin\java.exe "-javaagent:E:\intelliJ\IntelliJ IDEA 20
PS waiting for Master...
PS reading finished.
PS responding finished.
PS waiting for worker...
PS finishes his job.
PS waiting for Master...
PS reading finished.
PS responding finished.
PS waiting for worker...
PS finishes his job.
PS waiting for Master...
PS reading finished.
PS responding finished.
PS waiting for worker...
PS finishes his job.
```