Aligraph

1.Abstract：

Motivation：

> 图大，且隐含丰富信息

Challenge：

> 高效图信息存储，以及大量计算能力需求

Result：

> 1.建图时间对标powergraph 5min vs hours
> 2.采用新的缓存技术提升40%-50%速度，demo 大约有12倍的性能提升（目测跟自己比较）
> 3.F1值得到统计意义上的显著提升（4.12%-17.19%）

2.Introdution

采用技术：

> Graph embedding(图嵌入技术)，将图信息映射到低维向量空间中
> 然后对低维向量进行投入到CNN，通过共享权重以及多层结构提高学习力

Challenge：

> 1.真实世界图大，属性多，随时间快速演变
> 2.现有图算法对于真实图很难拓展，且需要考虑如何提高GNN对大规模图的时间以及空间效率
> 3.怎么优雅的整合异构的数据为统一的embedding
> 4.统一定义以及保留信息
> 5.动态图上采取增量方法

Contributions

1. 对常见的GE(图嵌入)，GNN算法,根据使用情况分类，其中深色底的为阿里实际使用的算法（在处理显示生活中的问题相对灵活且更实用）

Table 1: The property of different methods.

Category	Method	Heterogeneous		Attributed	Dynamic	Large-Scale
		Node	Edge			
Classic Graph Embedding	DeepWalk	✗	✗	✗	✗	✗
	Node2Vec	✗	✗	✗	✗	✗
	LINE	✗	✗	✗	✗	✗
	NetMF	✗	✗	✗	✗	✗
	TADW	✗	✗	✓	✗	✗
	LANE	✗	✗	✓	✗	✗
	ASNE	✗	✗	✓	✗	✗
	DANE	✗	✗	✓	✗	✗
	ANRL	✗	✗	✓	✗	✗
	PTE	✗	✓	✗	✗	✗
	Methpath2Vec	✗	✓	✗	✗	✗
	HERec	✗	✓	✗	✗	✗
	HNE	✗	✗	✗	✗	✗
	PMNE	✗	✓	✓	✗	✗
	MVE	✗	✓	✓	✗	✗
	MNE	✗	✓	✓	✗	✗
	Mvn2Vec	✗	✓	✓	✗	✗
GNN	Structural2Vec	✗	✗	✓	✗	✗
	GCN	✗	✗	✓	✗	✗
	FastGCN	✗	✗	✓	✗	✗
	AS-GCN	✗	✗	✓	✗	✗
	GraphSAGE	✗	✗	✓	✗	✗
	HEP	✓	✓	✓	✗	✗
	AHEP	✓	✓	✓	✗	✓
	GATNE	✓	✓	✓	✗	✓
	Mixture GNN	✓	✓	✓	✗	✗
	Hierarchical GNN	✓	✓	✓	✗	✗
	Bayesian GNN	✗	✓	✓	✗	✗
	Evolving GNN	✗	✓	✓	✓	✗

2. 对于整个系统抽象为三层，存储层，采样层，操作层
3. 对于存储层主要采用了三种技术structural and attributed specific storage（结构属性分离存储）, graph partition（图分区）,caching neighbors of important vertices（cache存储重要节点）
4. 对于sampling（取样操作）分成三类（1.TRAVERSE（顺序采样）2.NEIGHBORHOOD（临近采样）3.NEGATIVE（负采样？））
5. 对于GNN具体操作，特化成一下两种操作AGGREGATE（聚合），COMBINE（结合）
6. 对于cache层面对中间结果进行缓存加速计算过程

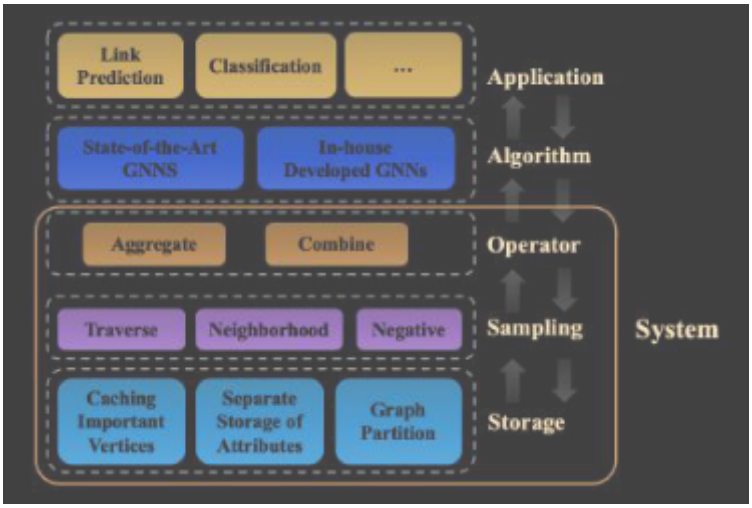
3.PRELIMINARIES

问题定义：

> Embedding问题:将图G转化为d维低维向量空间并尽可能保留图的原本信息（原文指出此处的GNN是一种进行graph embedding的方式，事实上广义上的GNN包括但不限于embedding）

GNN算法架构

自上到下主要分成五层，其中上面两层为算法以及应用层，本文所涉及内容主要针对下三层，下三层为上两层提供接口以及对应服务



存储层：存储不同种类原生数据，并为上层操作以及算法提供高速数据传输
采样层：是操作层的基础，直接对所需要的数据进行操作，并为上层提供操作的训练样本
操作层：对GNN进行具体的操作

存储层

1.采用图分区减少分布式存储时跨服务器的边，本文主要采用了以下4种不同的分区策略
（1）METIS；
（2）Vertex cut and edge cut partition；
（3）2-D partition ；
（4）Streaming-style partition strategy
2.对重要顶点的临近节点cashe缓存，具体内同以及代价运算因为感觉目前考虑过来的优先级比较低，先行跳过

采样层

问题：

现实中的图通常是不对称的（即数据不是规整的，例如点的出度/入度通常是不一样的），为了让卷积操作更容易嵌套进当前主流的模型，所以要对数据进行采样训练

概述：

为了满足训练时不同需求，大致定义了三种采样方式
1.TRAVERSE（顺序采样）：对子图的小部分点或者边进行整个采样
2.NEIGHBORHOOD（局部采样）：对某个点临近（曼哈顿距离为1跳，或者若干跳）的节点进行

> 采样，用以encode该节点
> 3.NEGATIVE（负采样）：生成负样本用以加速收敛（融合？）

具体实现：

> 采样时采用并行无锁队列进行采样训练数据生成，进一步提高系统的效率

操作层

> 经过采样层采样之后，生成的训练数据可以保证是同一规格的了，因此对于接下来的操作提供了很大的便利，对于相关之后数据的处理，本文抽象为两种基本操作AGGRAGATE，COMBINE

AGGREGATE

> 卷积操作，从周围的邻居节点搜集信息，对于不同的GNN方法有不同的具体操作方式如：
> elementwise mean，max-pooling neural network，long short-term memory等

COMBINE

> 将前一跳信息以及邻接节点信息整合到同一空间,相当于迭代操作，通过本次训练的结果调整生成的embedding结果