AI ProCon 开发者大会

CSDN

Euler：超大规模图深度学习开源框架

林伟

阿里巴巴Euler开源项目团队

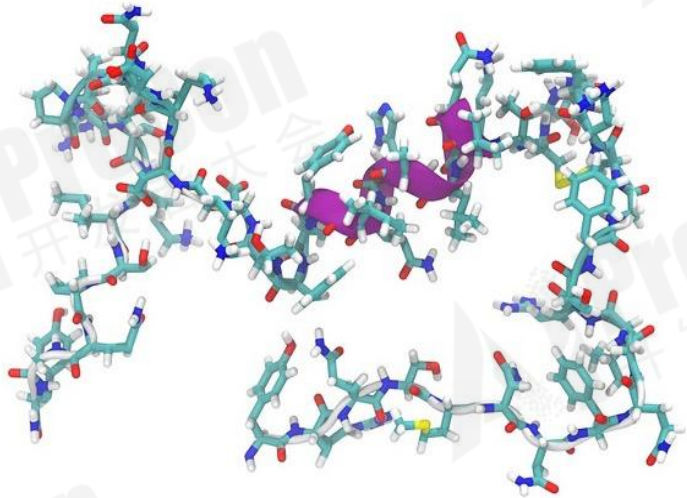
大纲

- 图深度学习的背景与技术挑战
- 从算法到框架的联合设计
- 阿里巴巴的应用案例介绍
- 使用Euler快速构建图神经网络
- Euler开源项目

Graph Deep Learning经典应用场景

电商场景的人货场匹配

社交网络的人群发现

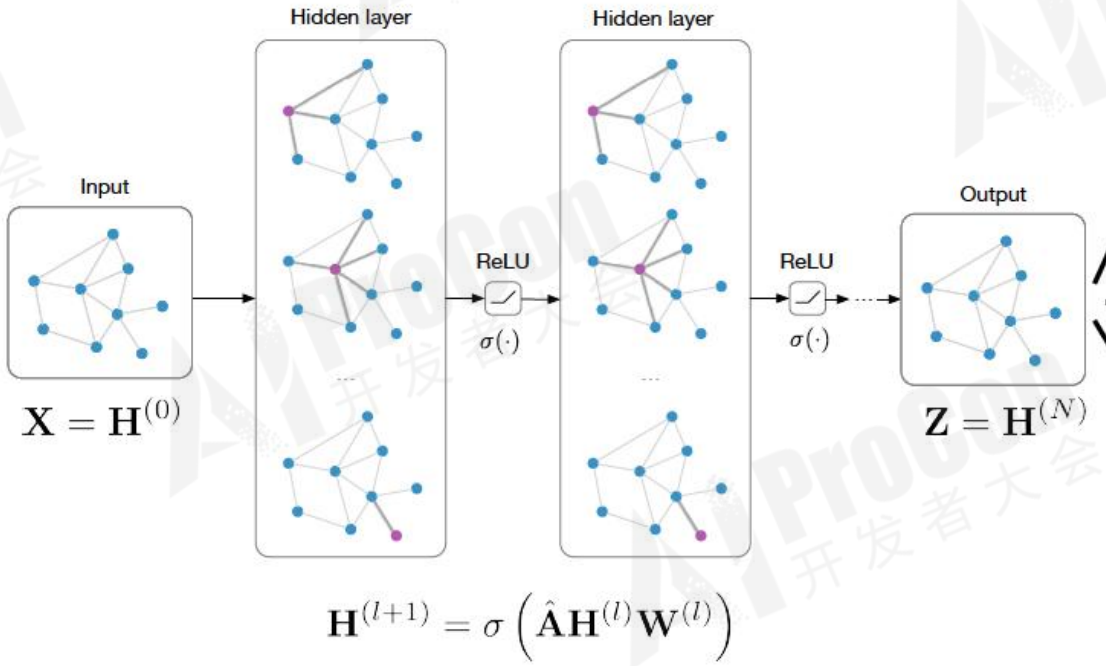


蛋白质分类

达摩院和DeepMind的观点：
“图神经网络将端到端学习与归纳推理相结合，有望解决深度学习无法处理的关系推理、可解释性等一系列问题，有望形成**下一代AI技术**”

Graph Deep Learning的价值

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



Node classification:

$\mathrm{softmax}(\mathbf{z_n})$

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

$\mathrm{softmax}(\sum_n \mathbf{z_n})$

e.g. Duvenaud et al. (NIPS 2015)

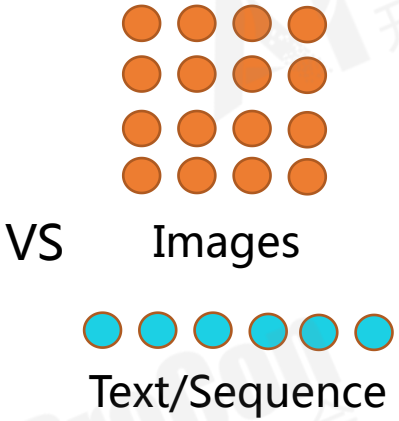
Link prediction:

$p(A_{ij}) = \sigma(\mathbf{z_i^T z_j})$

Kipf & Welling (NIPS BDL 2016)
"Graph Auto-Encoders"

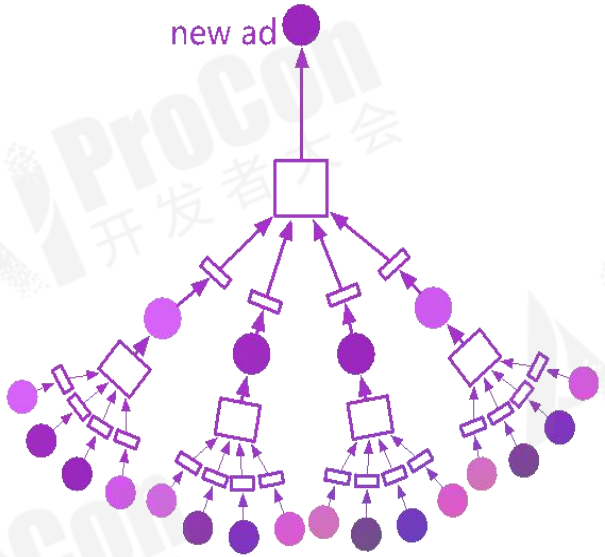
- 结构化的图与深度学习的有机结合
- 深层复杂关系的动态刻画
- 可解释能和推理能力

Graph Deep Learning的主要挑战



超大规模异构图

1. 数十亿点，数百亿边
2. 异构图建模的能力
3. 动态更新



计算模式的爆炸

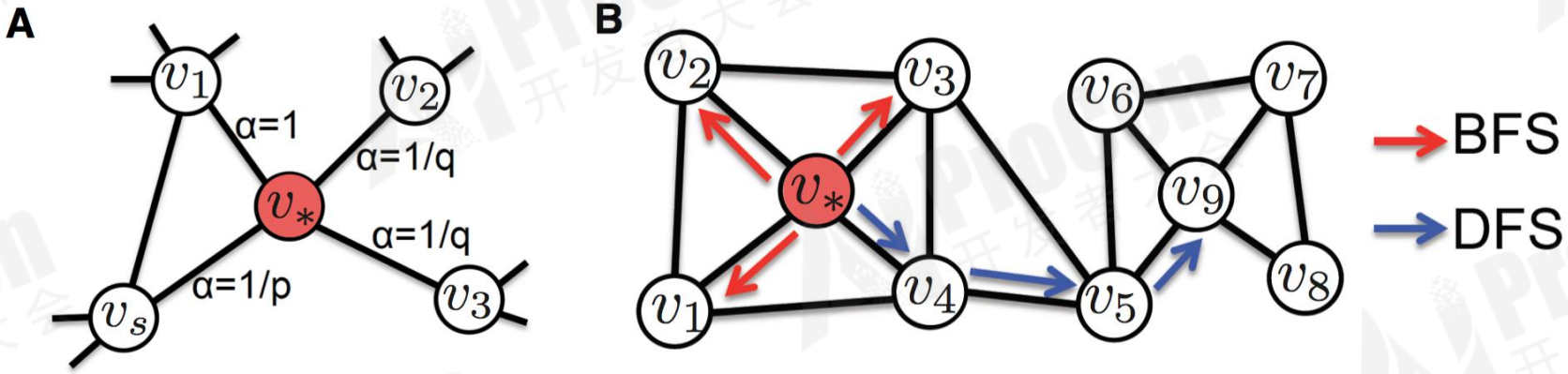
1. 图深度学习算法带来的指数爆炸
2. 灵活的算法组合与创新能力
3. 图学习与深度学习的端到端结合

大纲

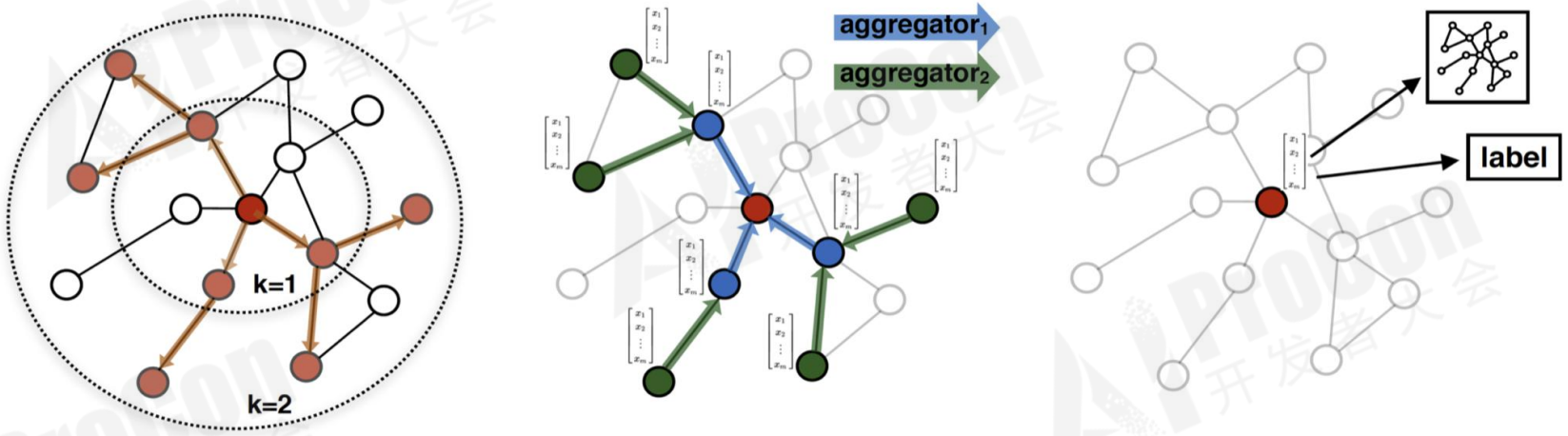
- 图深度学习的背景与技术挑战
- 从算法到框架的联合设计
- 阿里巴巴的应用案例介绍
- 使用Euler快速构建图神经网络
- Euler开源项目

图深度学习的两种主要Pattern

Random Walk based



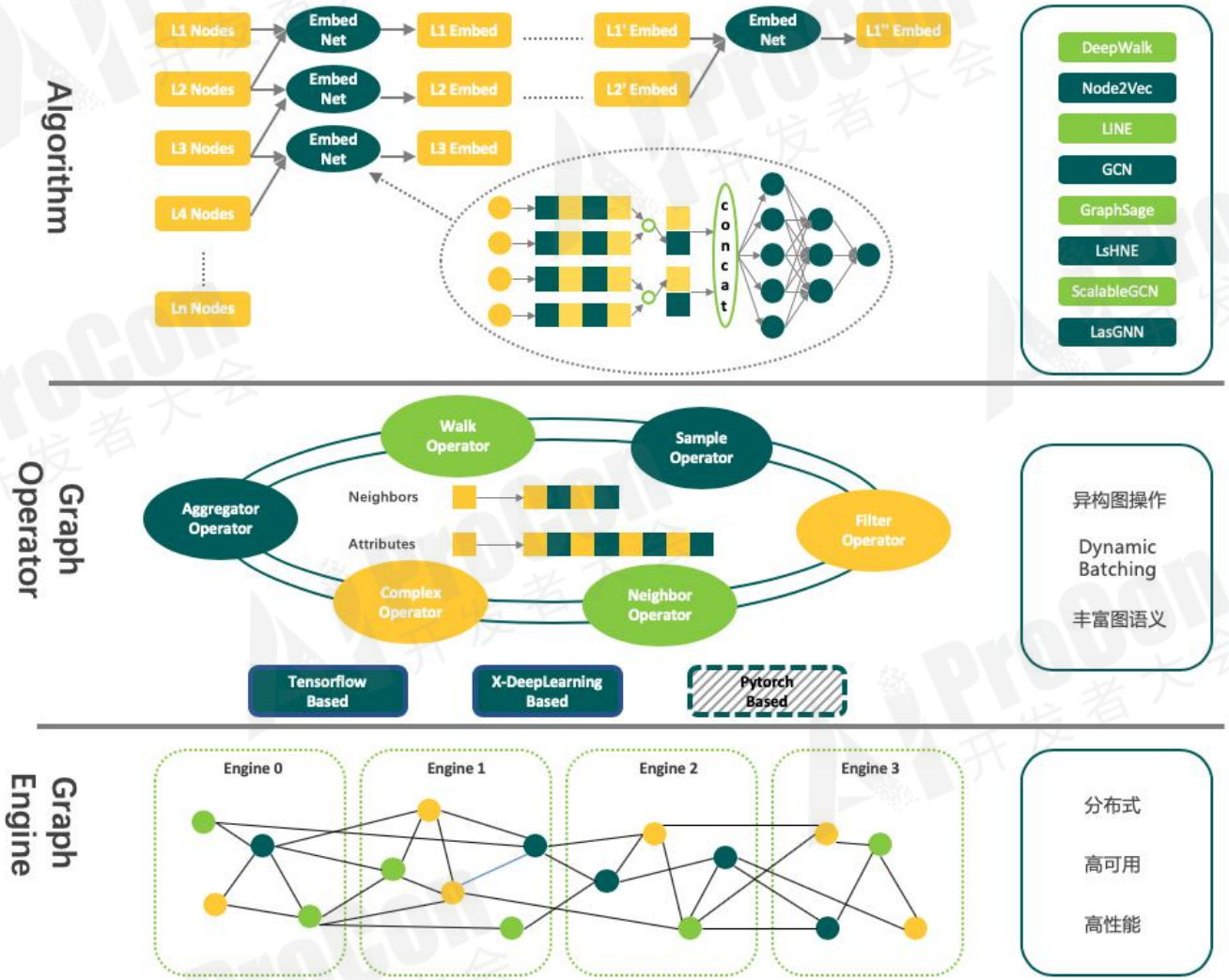
Neighbor Aggregation based



Euler的设计理念

- 分层灵活可扩展设计
- 大规模高性能异构图学习
- 灵活多样的图算法支持
- 通用GNN训练加速

分层灵活可扩展设计


Algorithm
L1 Nodes
L2 Nodes
L3 Nodes
L4 Nodes
Ln Nodes
Embed Net
Embed Net
Embed Net
L1 Embed
L2 Embed
L3 Embed
L1' Embed
L2' Embed
Embed Net
L1'' Embed
concat
DeepWalk
Node2Vec
LINE
GCN
GraphSage
LsHNE
ScalableGCN
LasGNN
Graph Operator
Walk Operator
Sample Operator
Aggregator Operator
Neighbors
Attributes
Filter Operator
Complex Operator
Neighbor Operator
Tensorflow Based
X-DeepLearning Based
Pytorch Based
异构图操作
Dynamic Batching
丰富图语义
Graph Engine
Engine 0
Engine 1
Engine 2
Engine 3
分布式
高可用
高性能

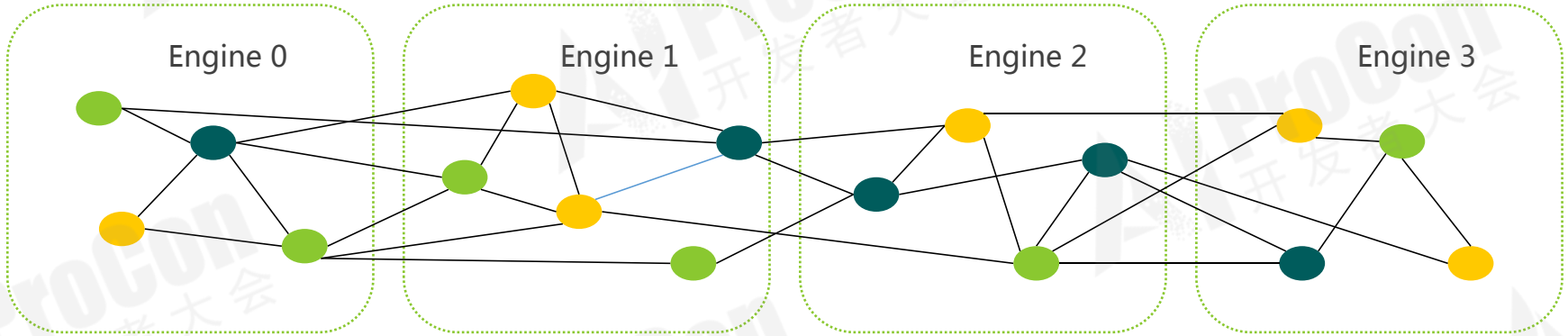
大规模分布式异构图

水平可扩展

- Random打散的子图划分
- 图分片可指定partition count，支持十亿甚至百亿图数据的存储
- 支持多种类型边点数据存储

高吞吐

- 多Replica设计
- 图操作执行优化
- 子图计算逻辑内聚



为图学习优化的存储结构

高效全局索引结构

- O(1)时间复杂度全图点边采样

可配邻居索引结构

- AliasTable / PartialSum平衡空间与时间

异构点边索引支持

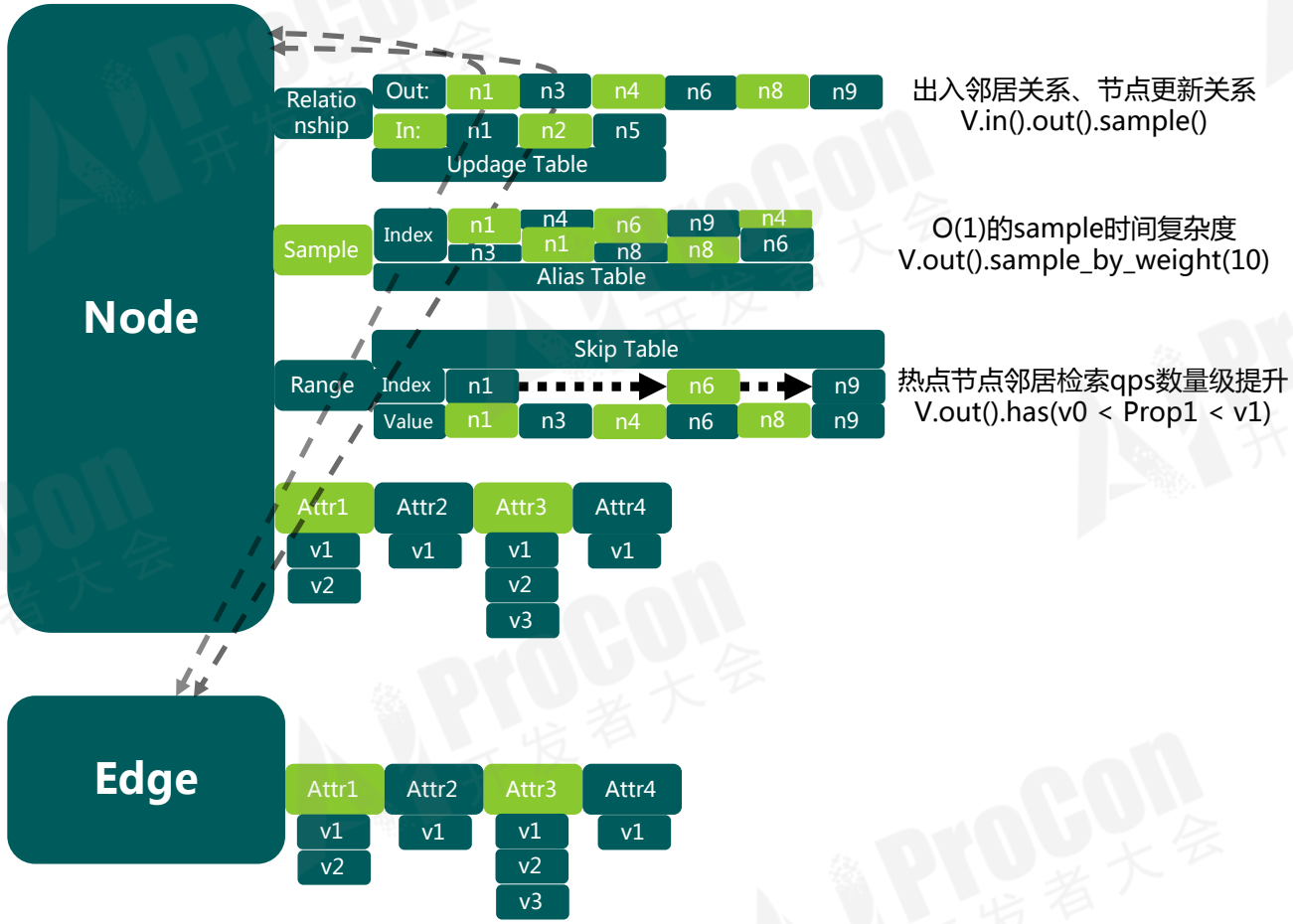
- 按标签（type）图遍历筛选

丰富邻居访问接口

- Sample / Full / Sorted / TopK / Layer-wise

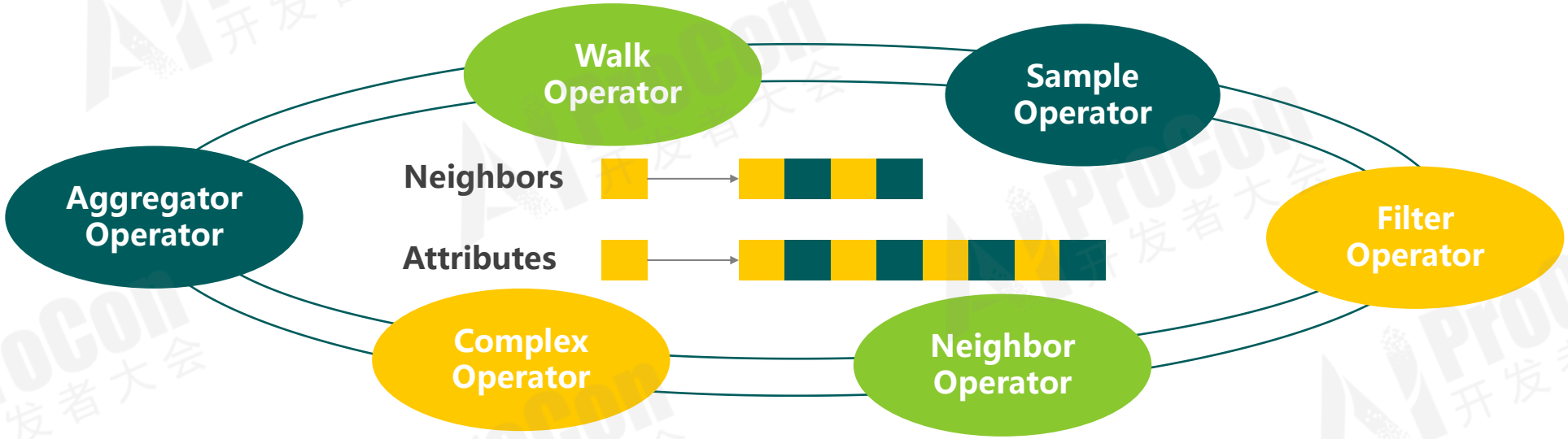
异构点边属性存储

- 多种深度学习特征存取



图学习的语义

- Global Operator
  - Node Sample
  - Edge Sample
- Neighbor Operator
  - Random Walk
  - Multi-Hop Neighbor
  - Neighbor Sample
- Attribute Operator
  - Join
  - Merge
  - Aggregator
- Logical Operator
  - Filter
  - And/Or



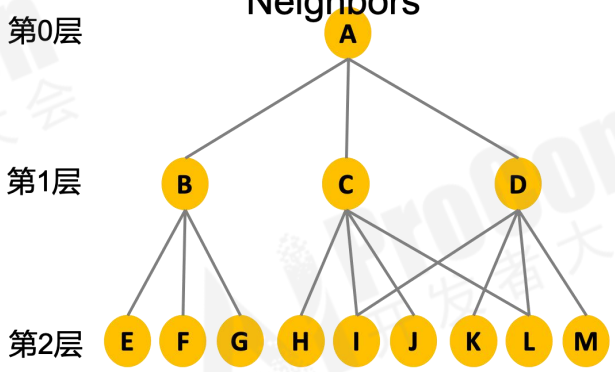
灵活多样的图算法支持

	算法类型	是否自研	特点
DeepWalk	Random Walk	否	经典的无偏随机游走无监督算法
Node2Vec	Random Walk	否	利用可配置参数在游走时可倾向BFS或DFS
LINE	Random Walk	否	灵活利用1阶，2阶邻居信息的无监督算法
GCN	GNN	否	CNN操作类似推广到非欧空间的算法
GraphSAGE	GNN	否	GCN改进，提出邻居采样，多种汇聚函数等
GAT	GNN	否	将Attention技术用于邻居汇聚
Scalable-GCN	GNN	是	加速GCN训练的一种方法
LsHNE	Random Walk	是	异构图中随机游走，利用深度网络编码
LasGNN	GNN	是	半监督大规模异构图卷积网络学习方法

训练加速-邻居选择加速算法

邻居选择是GNN类算法中的重要环节，对计算性能和精度有很大的影响，下图对比了三种邻居选择的策略。

Full Neighbors

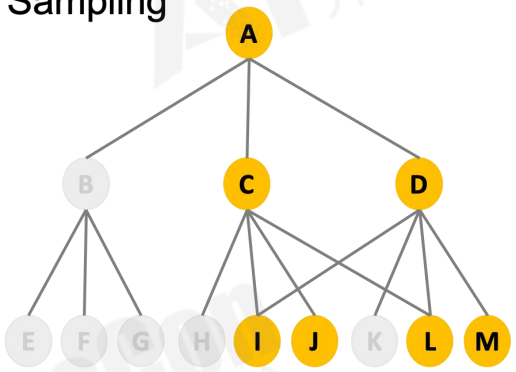


假设每个节点有N个邻居，展开K阶邻居，总共选择的邻居节点数量为：

$$\text{Neighbours} = \sum_{i=0}^{k} N^i$$

计算量指数增长，规模不可控。

Node-wise Neighbors Sampling

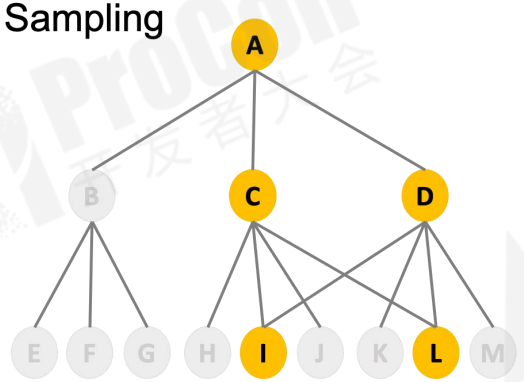


假设每个节点采样M(M<N)个邻居，展开K阶邻居，总共选择的邻居节点数量为：

$$\text{Neighbours} = \sum_{i=0}^{k} M^i$$

计算量仍然指数增长，规模不可控。

Layer-wise Neighbors Sampling



假设层节点最多允许采样L个，展开K阶邻居，总共选择的邻居节点数量为：

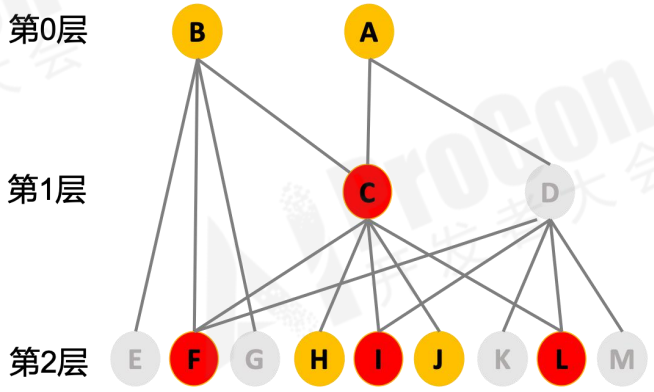
$$\text{Neighbours} = K * L$$

计算量线性增长，规模可控。

Euler2.0已经内置了对分布式Layer-wise Neighbors Sampling算法的支持

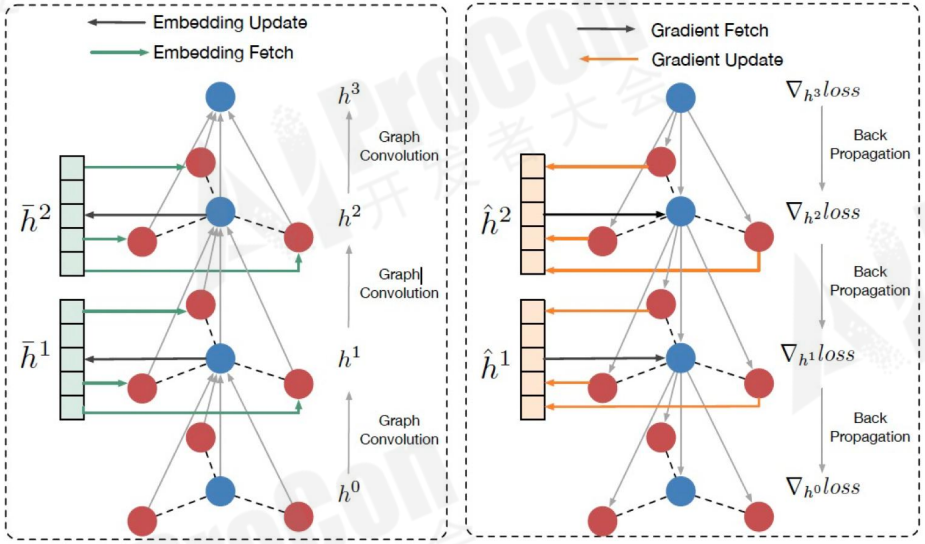
训练加速-邻居汇聚加速算法

在图结构中存在一个节点被多个节点作为**公共的一阶或者多阶邻居**，计算图卷积时，这些点会被**大量重复计算**。



引入缓存机制，**用空间换时间**
缓存节点的1～K阶Embedding

引入梯度补偿机制，保证收敛性



例如：
C为A,B的一阶公共邻居
F,I,L为C,D的一阶公共邻居
F,H,I,J,L为A,B的二阶公共邻居

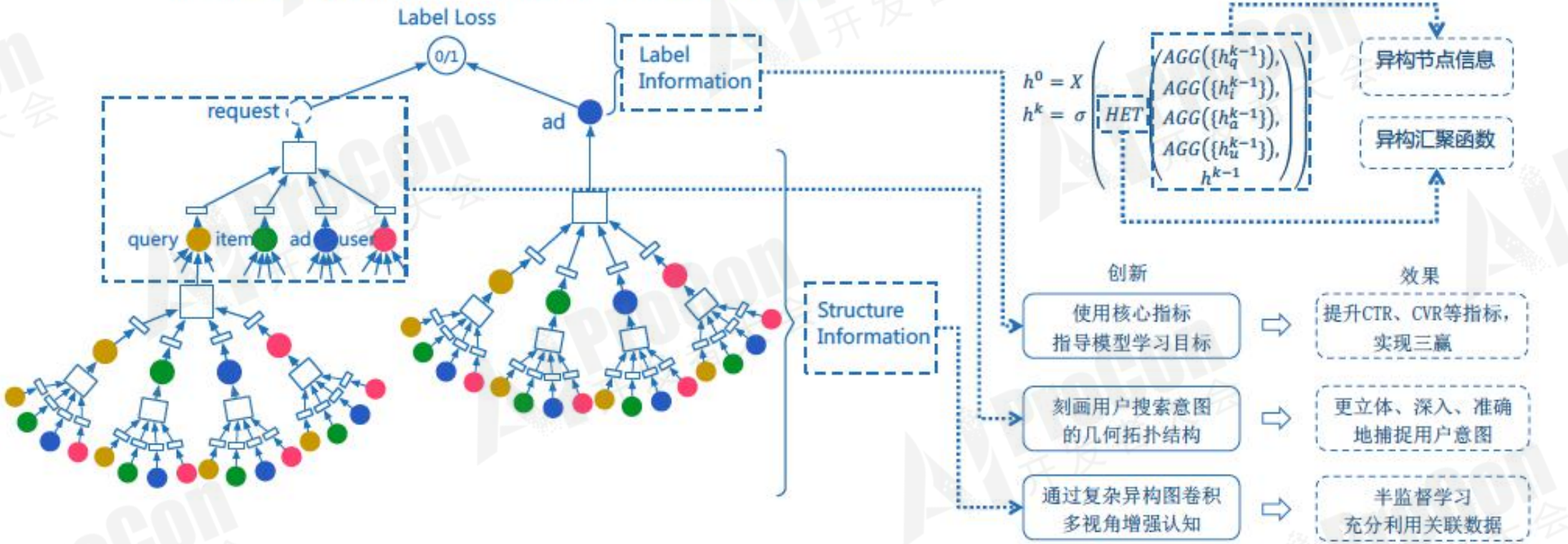
一个三阶卷积的例子：
左侧前向计算时通过获取节点缓存embedding进行加速，右侧反响传播时累积embedding向量的梯度去更新卷积核参数

大纲

- 图深度学习的背景与技术挑战
- 从算法到框架的联合设计
- 阿里巴巴的应用案例介绍
- 使用Euler快速构建图神经网络
- Euler开源项目

场景一-向量召回LasGNN模型(1)

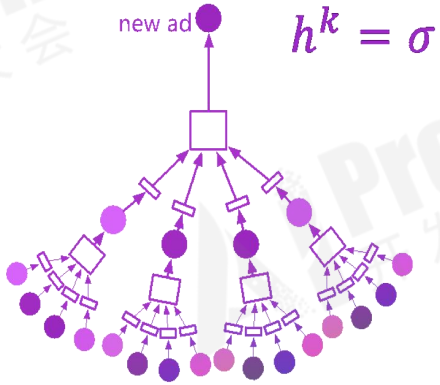
通过图卷积同时刻画标注信息和图结构信息



场景一-向量召回LasGNN模型(2)

用户行为序列刻画示意图
搜索序列->搜索子图：更立体全面的意图捕捉

广告刻画示意图图
图卷积的传播机制：有很强的推理泛化能力，对新广告友好



$$h^k = \sigma([W^k AGG(\{h_N^{k-1}\}), B^k h^{k-1}])$$

通过异构图卷积，高维
Embedding信息通过多种视角，
从有标注节点（热门广告）传播
至无标注节点（新广告）

场景一-向量召回LasGNN模型(3)

基于Metapath的方式，是比较实用有效的异构图GNN方法：

目前用到的Metapath的类型

Table 3: Meta-paths used in hierarchical network

Number	Metapaths
1	$Query \xrightarrow{e_{click}} Item\backslash Ad \xrightarrow{e_{session}} Item\backslash Ad$
2	$Query \xrightarrow{e_{CF}} Query \xrightarrow{e_{CF}} Item\backslash Ad$
3	$Query \xrightarrow{e_{CF}} Query \xrightarrow{e_{semantic}} Query$
4	$Query \xrightarrow{e_{click}} Item\backslash Ad \xrightarrow{e_{CF}} Query$
5	$Item\backslash Ad \xrightarrow{e_{session}} Item\backslash Ad \xrightarrow{e_{domain}} Item\backslash Ad$
6	$Item\backslash Ad \xrightarrow{e_{session}} Query \xrightarrow{e_{click}} Item\backslash Ad$

基于Metapath选邻居的示例

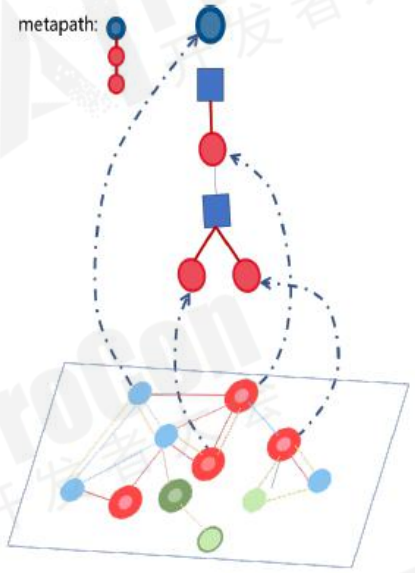


Figure 6: Use metapath to guide neighborhood selecting

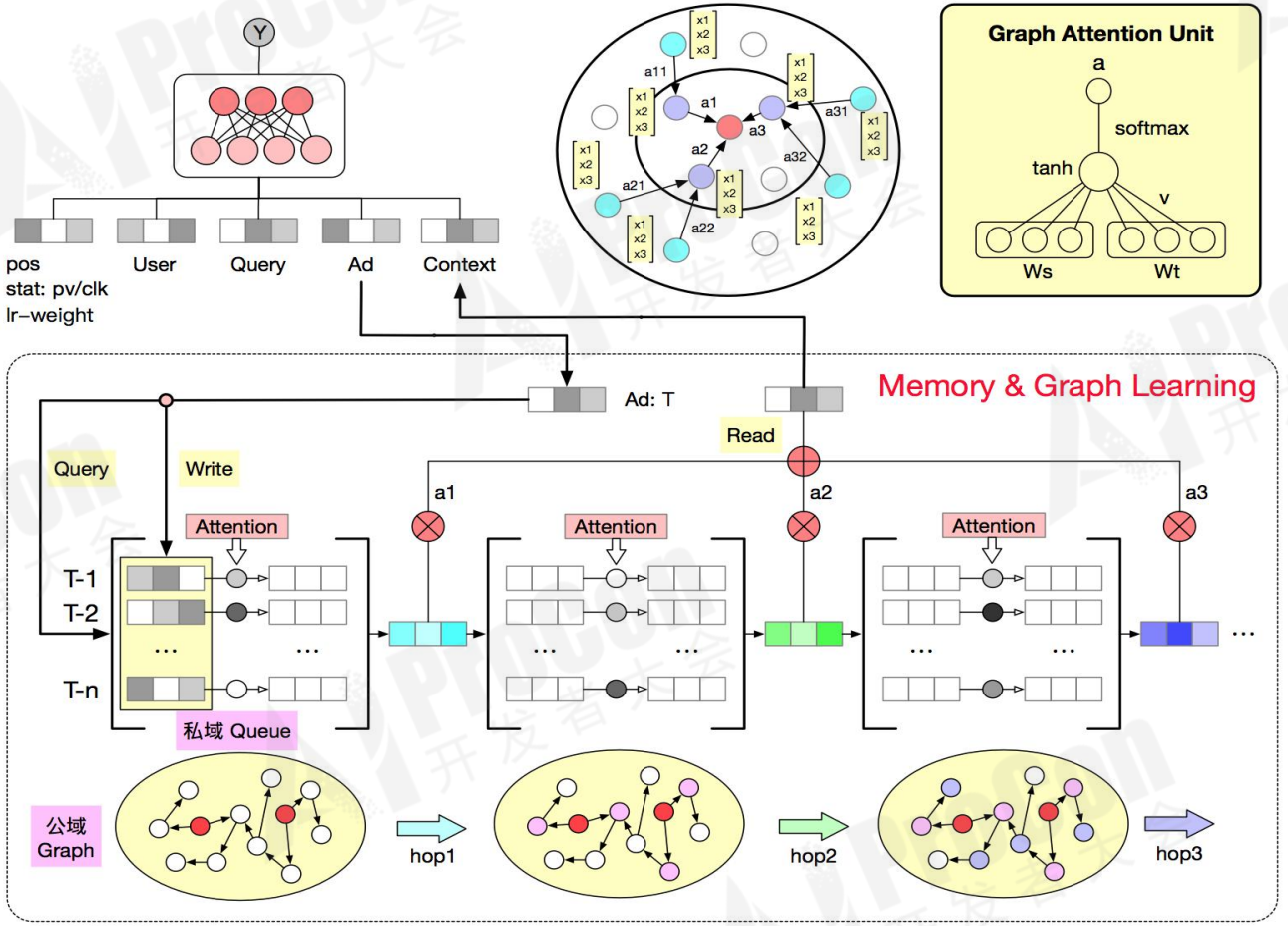
场景二-点击率预估GIN模型(1)

1、核心问题：

- 完整消费需求= 显性意图Query+ 隐性意图+ 潜在隐性意图
- 利用图结构的空间延展能力，进一步挖掘用户意图信息

2、关键解法：Memory+Graph

- 存储结构：
  - 私域-时间维度：用户实时前置行为队列
  - 公域-空间维度：协同过滤公共行为大图
- 逻辑算子：
  - 潜在意图扩散：共现 TopK 邻居节点采样
  - 潜在意图收敛：Attention机制汇聚



场景二-点击率预估GIN模型(2)

用户行为序列的图拓扑展开示例

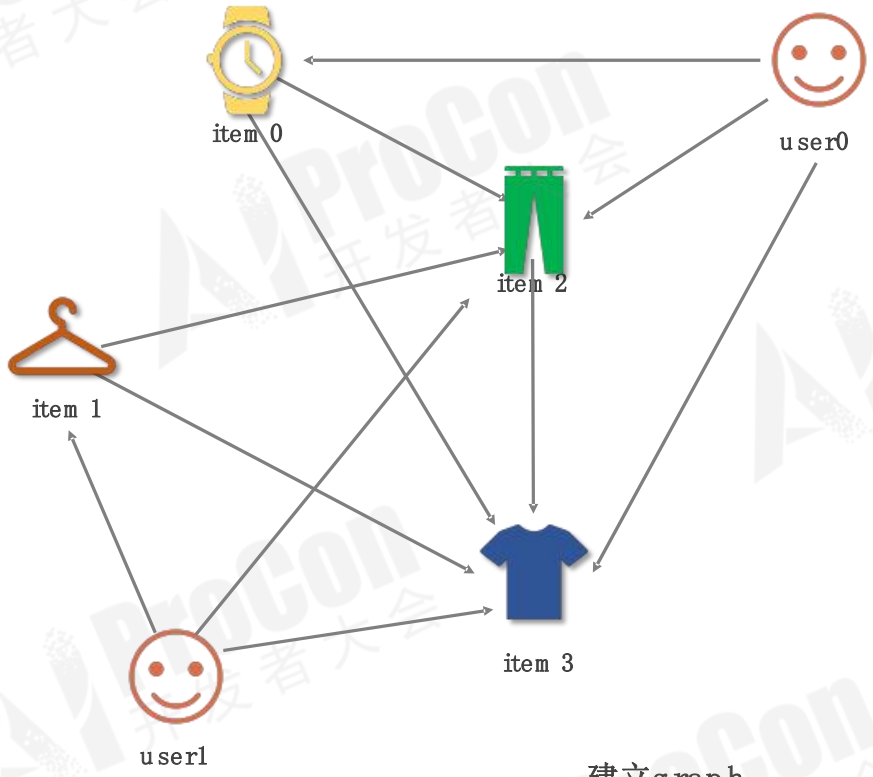
user_pv_bin	auc_gap
1	0.0030
2	0.0047
3	0.0049
4	0.0055
5	0.0058
6	0.0061
7	0.0062
8	0.0071
9	0.0072
10	0.0058

大纲

- 图深度学习的背景与技术挑战
- 从算法到框架的联合设计
- 阿里巴巴的应用案例介绍
- 使用Euler快速构建图神经网络
- Euler开源项目

商品推荐场景-建立图数据

历史访问数据



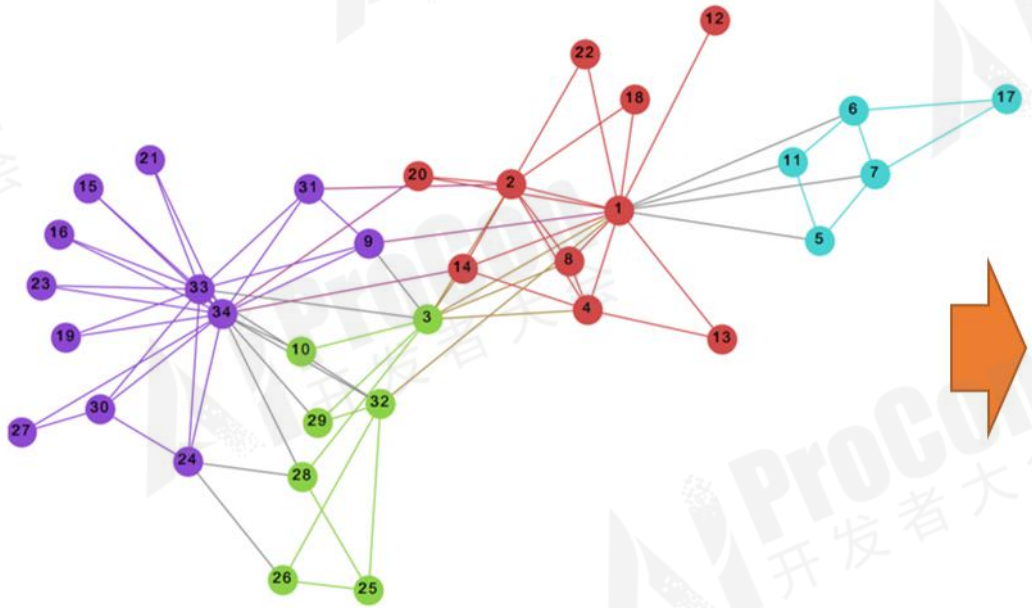
建立graph

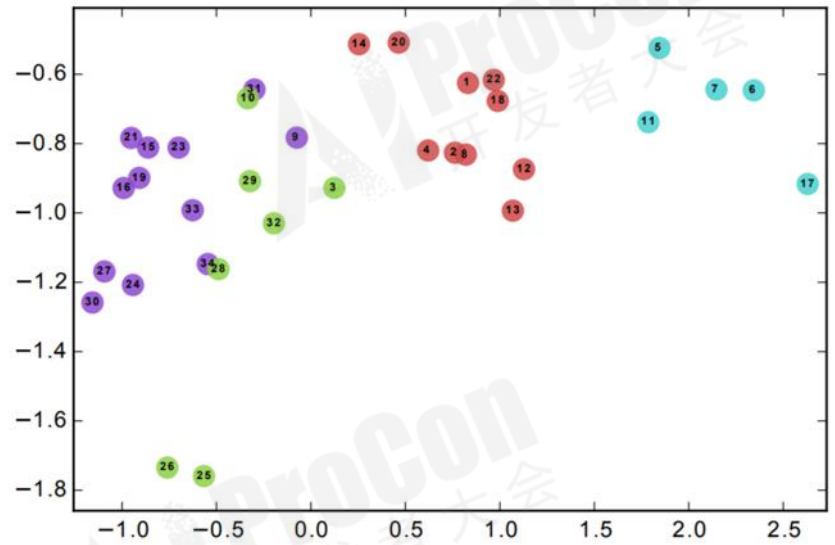
商品推荐场景-生成向量化表征

```python
class DeepWalk(tf_euler.layers.Layer):
  def __init__(self, node_type, edge_type, max_id, dim,
               num_negs=8, walk_len=3, left_win_size=1, right_win_size=1):...

  def call(self, inputs):
    src, pos, negs = self.sampler(inputs)
    embedding = self.target_encoder(src)
    embedding_pos = self.context_encoder(pos)
    embedding_negs = self.context_encoder(negs)
    loss, mrr = self.decoder(embedding, embedding_pos, embedding_negs)
    embedding = self.target_encoder(inputs)
    return embedding, loss, 'mrr', mrr

  def sampler(self, inputs):
    batch_size = tf.size(inputs)
    path = tf_euler.random_walk(
        inputs, [self.edge_type] * self.walk_len,
        default_node=self.max_id + 1)
    pair = tf_euler.gen_pair(path, self.left_win_size, self.right_win_size)
    num_pairs = pair.shape[1]
    src, pos = tf.split(pair, [1, 1], axis=-1)
    negs = tf_euler.sample_node(batch_size * num_pairs * self.num_negs,
                                self.node_type)
    src = tf.reshape(src, [batch_size * num_pairs, 1])
    pos = tf.reshape(pos, [batch_size * num_pairs, 1])
    negs = tf.reshape(negs, [batch_size * num_pairs, self.num_negs])
    return src, pos, negs
```

- python -m tf_euler --data_dir item_item_graph --model random_walk --mode train
- python -m tf_euler --data_dir item_item_graph --model random_walk --mode save_embedding

商品推荐场景-商品推荐

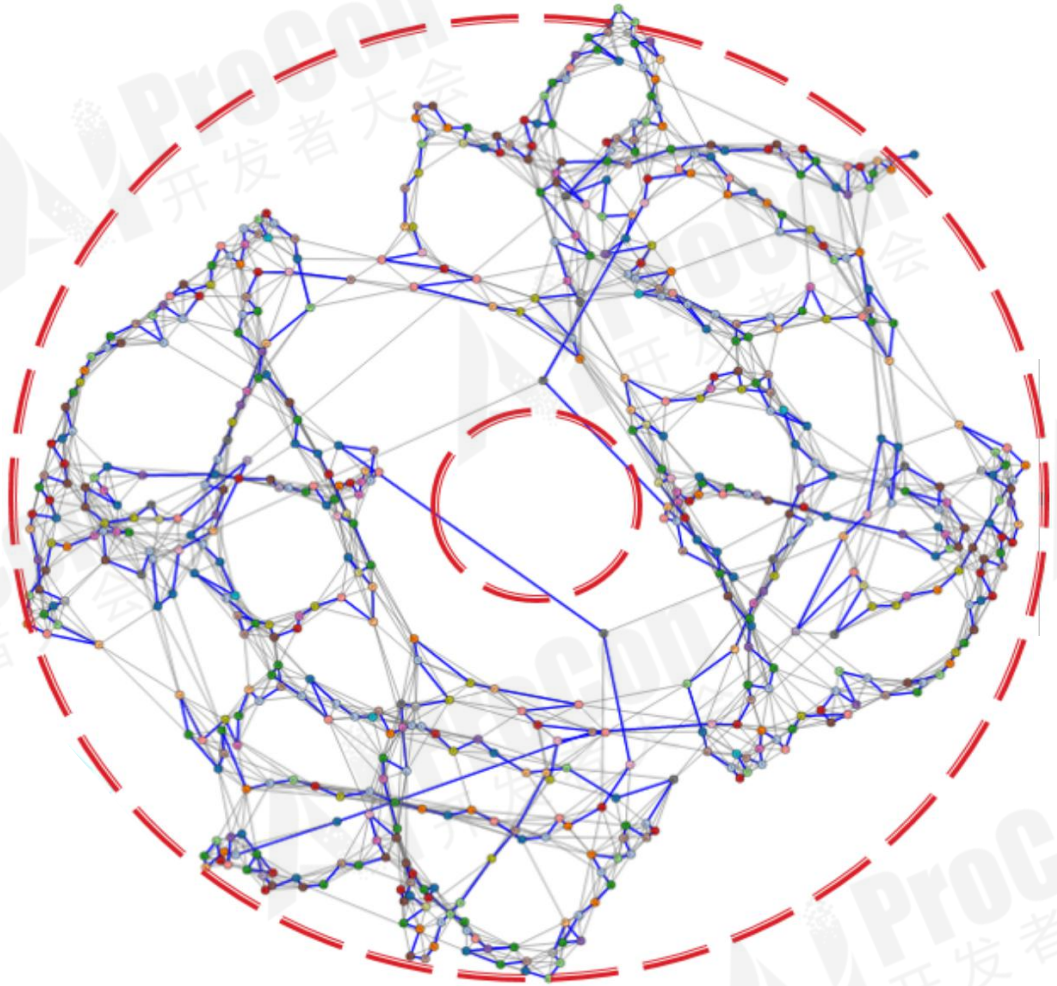




- 使用user最近访问的item，在向量化空间召回相似的item

蛋白质分类场景-建立图数据

Protein Protein Interaction



蛋白质分类场景-邻居信息聚合

```
class SageEncoder(tf_euler.layers.Layer):

  def call(self, inputs):
    samples = tf_euler.sample_fanout(inputs, self.metapath, self.fanouts)[0]  # 多跳邻居采样
    hidden = [
        tf_euler.get_dense_feature(sample,
                                   [self.feature_idx], [self.feature_dim])[0]
        for sample in samples]
    for layer in range(self.num_layers):
      aggregator = self.aggregators[layer]
      next_hidden = []
      for hop in range(self.num_layers - layer):                    # 多层邻居聚合，层层递进
        neigh_shape = [-1, self.fanouts[hop], self.dims[layer]]
        h = aggregator((hidden[hop], tf.reshape(hidden[hop + 1], neigh_shape)))
        next_hidden.append(h)
      hidden = next_hidden
    return hidden[0]
```

- python -m tf_euler --data_dir protein_graph --model graphsage_supervised --mode train

蛋白质分类场景-多分类预测

```
class GraphSage(tf_euler.layers.Layer):

  def call(self, inputs):
    nodes, labels = self.sampler(inputs)
    embedding = self.encoder(nodes)
    predictions, loss, f1 = self.decoder(embedding, labels)
    return (prediction, loss, 'f1', f1)

  def sampler(self, inputs):
    labels = tf_euler.get_dense_feature(inputs, [self.label_idx],
                                        [self.label_dim])[0]
    return inputs, labels

  def decoder(self, embedding, labels):
    logits = self.predict_layer(embedding)
    loss = tf.nn.sigmoid_cross_entropy_with_logits(labels=labels, logits=logits)  # 使用Em bedding层预测结果
    predictions = tf.floor(tf.nn.sigmoid(logits) + 0.5)
    f1 = tf_euler.metrics.f1_score(labels, predictions)
    return predictions, tf.reduce_mean(loss), f1
```

- python -m tf_euler --data_dir protein_graph --model graphsage_supervised --mode predict

大纲

- 图深度学习的背景与技术挑战
- 从算法到框架的联合设计
- 阿里巴巴的应用案例介绍
- 使用Euler快速构建图神经网络
- Euler开源项目

https://github.com/alibaba/euler

alibaba / euler

Watch 129 | Star 1,678 | Fork 356

Code | Issues 82 | Pull requests 0 | Projects 0 | Wiki | Security | Insights

Dismiss

Join GitHub today

GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

Sign up

A distributed graph deep learning framework.

graph graph-learning network-embedding deep-learning graph-convolutional-networks graph-neural-networks graphsage random-walk node2vec graph-embedding gcn ggnn

30 commits | 3 branches | 2 releases | 5 contributors | Apache-2.0

Branch: master | New pull request | Find File | Clone or download

yangsiran Update README.md — Latest commit ff40594 on 4 Jun

cmake	Patch 20190208.	5 months ago
euler	index fix	3 months ago
examples	Initial import.	8 months ago
tf_euler	Fix the usage of SyncExitHook.	3 months ago

CSDN

AI ProCon

开发者大会