## 周报

时间：2021-8-17

汇报人：杨琴琴

1. 本周学习内容
2. 整理推理相关论文

2.论文学习

《**Temporal Knowledge Graph Reasoning Based on EvolutionalRepresentation Learning**》-GCN、GRU、time gate

《**Search from History and Reason for Future: Two-stage Reasoning on Temporal Knowledge Graphs** 》--RL、随机束波搜索、GCN

1. 学习笔记
2. **知识推理相关工作**

内推：

1.《Temporal reasoning overevent knowledge graphs》-Sadeghian et al., 2016

2.《Learning sequence encodersfor temporal knowledge graph completion》-Garc´ıaDur´an et al. , 2018;

3.《Deriving validity time in knowledge graph.》-Leblay and Chekol, 2018

4.《Hyperplane-based temporally aware knowledge graph embedding》-Dasgupta et al., 2018;

5.《Effificiently embedding dynamic knowledge graphs》-Wu et al., 2019

6.《Temporal knowledge graph completion based on time series gaussian embedding-Xu et al., 2020;

7.《Diachronic embedding for temporal knowledge graph completion.》-Goel et al., 2020

8.《Temporal message passing for temporal knowledge graph completion》-Wu et al., 2020;

9.《Temporal knowledge graph forecasting with neural ode.》-Han et al., 2020a;

10.《Learning to walk across time for temporal knowledge graph completion》-Jung et al., 2020)

外推：

1.《Know-evolve: deep temporal reasoning for dynamic knowledge graphs》-Trivedi et al. (2017, 2018)

2.《Graph hawkes neural network for forecasting on temporal knowledge graphs》-Han et al., 2020b;

3.《Dynamic knowledge graph based multievent forecasting》-Deng et al., 2020;

4.《Recurrent event network for reasoning over temporal knowledge graphs.》

《Recurrent event network: Autoregressive structure inference over temporal knowledge graphs.》Jin et al., 2019, 2020;

RE-NET使用子图聚合器和GRU对子图序列一跳进行分析进行建模，对每次茶隼为给定实体提取这些直接参与的历史事实

1. 《Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks.》Zhu et al., 2020

CyGNet 将具有相同实体的历史事实建模，并与实体预测的每次查询相关联，因此主要侧重于用重复模式预测事实

6.《Temporal knowledge graph reasoning based on evolutional representation learning》

《Search from History and Reason for Future: Two-stage Reasoning on

Temporal Knowledge Graphs 》Li et al., 2021

**2.强化学习之八要素**

强化学习之八要素：

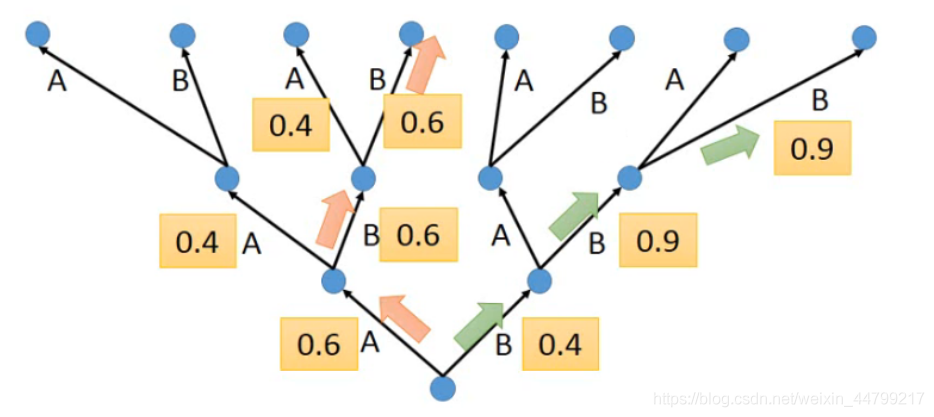
简单模型下的三要素：

环境的状态S、个体的动作A、环境的奖励R

复杂模型下的五要素：个体的策略、个体策略和状态s采取的行动、衰减因子、环境状态转化模型、探索率。

**3.束波搜索**

束波搜索最早是用在翻译当中，每次我们选择概率最大的token id作为输出，那么整个输出的句子的概率就是最大的么？是一种seq2seq中用来优化输出结果的算法(不在训练过程中使用)



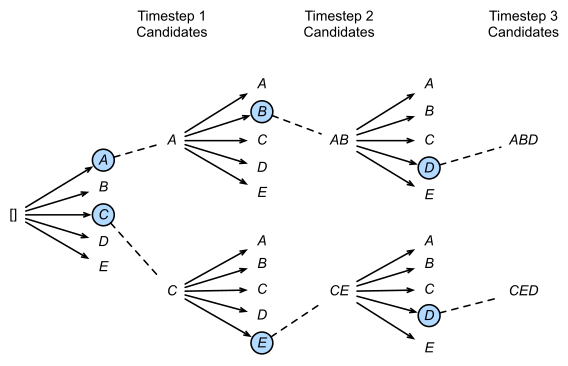
传统的获取解码器输出的过程中，每次只选择概率最大的那个结果，作为当前时间步的输出，等到输出结束，我们会发现，整个句子可能并不通顺。虽然在每一个时间步上的输出确实是概率最大的，但是整体的概率确不一定最大的，我们经常把它叫做greedy search[贪心算法]

为了解决上述的问题，可以考虑计算全部的输出的概率乘积，选择最大的哪一个，但是这样的话，意味着如果句子很长，候选词很多，那么需要保存的数据就会非常大，需要计算的数据量就很大

束波搜索：Beam Search 就是介于上述两种方法的一个这种的方法，假设Beam width=2，表示每次保存的最大的概率的个数，这里每次保存两个，在下一个时间步骤一样，也是保留两个，这样就可以达到约束搜索空间大小的目的，从而提高算法的效率

beam width =1 时，就是贪心算法，beam width=候选词的时候，就是计算全部的概率。beam width 是一个超参数。

下图是一个实际的例子，每个时间步有ABCDE共5种可能的输出，即 ，图中的num\_beams=2，也就是说每个时间步都会保留到当前步为止条件概率最优的2个序列。

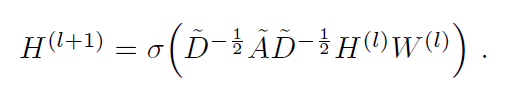


**4.图神经网络之GCN（2016年诞生）**

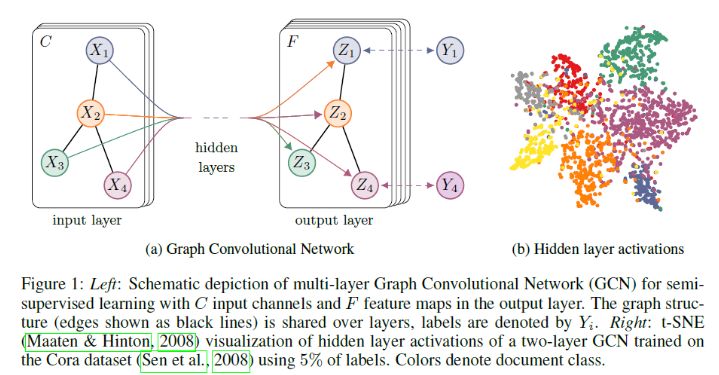
图的结构一般来说是十分不规则的，可以认为是无限维的一种数据，所以它没有平移不变性。每一个节点的周围结构可能都是独一无二的，这种结构的数据，就让传统的CNN、RNN瞬间失效。所以很多学者从上个世纪就开始研究怎么处理这类数据了。这里涌现出了很多方法，例如GNN、DeepWalk、node2vec等等，GCN只是其中一种。

设有一批图数据，其中有N个节点（node），每个节点都有自己的特征，我们设这些节点的特征组成一个N×D维的矩阵X，然后各个节点之间的关系也会形成一个N×N维的矩阵A，也称为邻接矩阵（adjacency matrix）。X和A便是我们模型的输入。

GCN也是一个神经网络层，它的层与层之间的传播方式是：



* A波浪=A+I，I是单位矩阵
* D波浪是A波浪的度矩阵（degree matrix）
* H是每一层的特征，对于输入层的话，H就是X
* σ是非线性激活函数

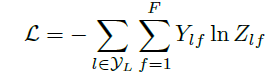


上图中的GCN输入一个图，通过若干层GCN每个node的特征从X变成了Z，但是，无论中间有多少层，node之间的连接关系，即A，都是共享的。

假设我们构造一个两层的GCN，激活函数分别采用ReLU和Softmax，则整体的正向传播的公式为：

IMG_256

最后，针对所有带标签的节点计算cross entropy损失函数：

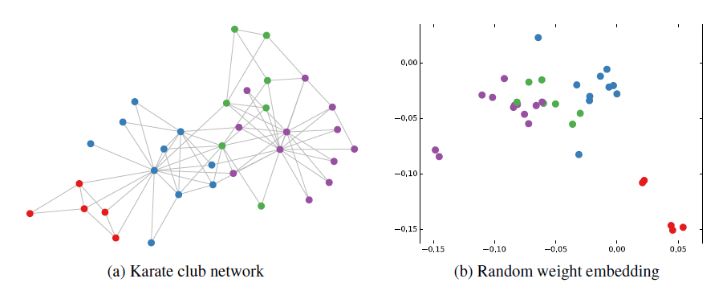


就可以训练一个node classification的模型了。由于即使只有很少的node有标签也能训练，作者称他们的方法为半监督分类。

当然，也可以用这个方法去做graph classification、link prediction，只是把损失函数给变化一下即可

GCN的优点：即使不训练，完全使用随机初始化的参数W，GCN提取出来的特征就以及十分优秀了！这跟CNN不训练是完全不一样的，后者不训练是根本得不到什么有效特征的

作者做了一个实验，使用一个俱乐部会员的关系网络，使用随机初始化的GCN进行特征提取，得到各个node的embedding，然后可视化



可以发现，在原数据中同类别的node，经过GCN的提取出的embedding，已经在空间上自动聚类了。

而这种聚类结果，可以和DeepWalk、node2vec这种经过复杂训练得到的node embedding的效果媲美了。

1. 下周任务：

由于TKG实际上是一个KG**序列**，现有的方法有三个问题：

( 1 )主要关注给定查询的实体和关系，忽略KG中每个时间戳中所有事实之间的结构依赖关系；

( 2 )推理往往有逻辑规则，融合静态图谱和使用RL系统能够使模型具有更好的解释性

( 3 )忽略实体类型等一些静态属性的函数。此外，现有的方法只注重实体预测，而关系预测不能同时用同一模型求解

1. GCN内部原理理解，和相关发展了解
2. 继续阅读事件外推的论文