Relational inductive biases, deep learning, and graph network

0. Abstract

- **核心思想**

  反对单纯使用人工特征或者单纯使用端到端系统的方法.AI中首要的问题是要解决组合泛化的问题, 而解决这个问题的方法是要解决**结构化表征方法**和**结构化计算**的问题.

  > 个人理解 :
  >
  > - 这里的**结构化表征方法**应该针对的是如今的单纯数值类型的词向量输入, 虽然词向量有很多维度, 但是知识之间没有组织结构(当然, 依存关系属于结构化表征的一种.)
  > - 这里的**结构化计算**应该针对的是如今单纯的数值运算法, 虽然存在着 pooling 等等的非连续性计算单元, 但是缺乏逻辑性的本质还是没有改变, 现1在使用的神经网络虽说是利用数值运算去模拟逻辑运算, 但是由于数值运算的连续性和逻辑运算的离散性的差异(还有什么其他的原因嘛?), 使得两者之间存在着一定的距离.
> - **组合泛化**
>
>   这里我的理解就是 :
>
>   组合泛化指的是, 将不同的关系和实体进行组合可以产生无限的生成能力, 参照自然语言.

- **任务描述**

  如何利用**关系归纳偏置**去 **帮助深度模型** 去学习实体,关系以及组织他们的方法.

> - 注意 : 核心模型还是深度模型.
> - 关系归纳偏差是:
>   - 归纳偏差:详见2.0.2节
>   - **关系归纳偏差**
>
>   详见2.0.3节

- **graph network**

  这里去实现该归纳偏差(假设模型)的架构是 graph network.

  - 这是一个基于图操作的神经网络, 再次之上可以添加各种各样的方法

    > 难道是在神经网络中嵌入特征嘛?如果是的话, 那和增加特征最为输入有什么区别呢?如果不是的话当我没说.

  - 可以操作结构化知识和产生结构化行为
  - 接下来会讨论, graph network如何支持关系推理和组合泛化. 这里的推理系统是复杂的, 可解释的, 以及灵活的.

    > 可解释的应该是针对于网络模型的缺点. 灵活的对应的是纯逻辑推理, 例如semantic parse的缺点. 复杂的? 大家都很复杂啊

1. Introduction

- **定义**

  Combinatorial generalization

  组合泛化对应的是语言的可以无限生成的能力, 从有限的词汇和规则中,通过组合可以生成无限的序列和意义. 准确定义为:

  Construct new inferences, predictions, and behaviors from known building blocks.

> 现在NLP的很多任务都是在做这样的工作, 例如神经翻译系统.神经翻译系统的known building blocks中, 词汇是已知的, 还会加入依存序列信息, 但是其中的最重要的inference(翻译)的信息是隐藏在数据中, 然后希望可以通过表现力强大的神经网络去学得这个信息.
>
> 个人认为还是要有一些显式的block, 也就是规则或者逻辑(一阶逻辑等等)作为已知信息.

- **认知机制**

  对**关系**的**复杂心理结构表征**, 以及**基于关系的推理**, 构成了人的认知系统, 而认知系统衍生了人类强大的组合泛化能力. 相关论文见论文列表1[1].

  人可以利用层级结构去高度抽象细粒度的差异, 并且可以找到表征和行为的一般性行为!!!!

  > 这个应该是很多做NLP都想去做的事情, 每个人或多或少都想过一些, 下面是我的个人理解 :
  >
  > - **概念阐述**
  >
  >   一个实体或者关系的表征毫无疑问是具有层级性或者说结构性的, 暂且不论这个结构是怎么样的(个人认为是树状, wordnet也是树状). 这个结构中肯定在较高层次(个人认为是最高层次)的抽象中, 拥有两个向下分的凭据 - **类型属性**以及**认知属性.**
  >
  >   - **认知属性**决定的是这个事物是如何被人感知的. 是一种可以独立于其他语义而存在的结构. 它的习得是凭借人类的原始感知能力, 我感觉大致分为三种, 对**时间**的感受(时序性), 对**空间**的感受(空间模型), 以及自身的**生理结构的内部感受**(体感以及情感等等). 认知属性的特点是结构复杂, 但是基本要素不多.
  >   - **类型属性**决定的是, 这个食物在现实世界中由于经常的共同出现而被联想起来的结构.是完全依存于语义之间的关系的. 结构简单但是基本要素很多.
  >
  > - **举例阐述**
  >
  >   比如说, "泄漏"这个词, 会想到两个用法.

    - 泄漏情报
    - 液体泄漏
  - **例子分析**

    这同一个词汇在两个意思下描述的是不同的事情, 但是两种存在着一种共性. 这种共性就是认知属性, 下面是自己的描述:

    - **泄漏的认知意义描述** : **实体E**在**被动**的情况下从**空间S1**以某种形式**移动到了空间S2**.

      其中, **实体E, 空间S1, 空间S2** 都是人可以通过对**空间**进行感知感受到的东西. 而**被动**是主格本体和其他本体之间存在的一种关系, 是属于**生理结构的内部感受**

    - **泄漏的类型意义描述** : 这里就可以用于简单的分类. 信息类别和液体的类别, 这个也许通过wordnet就可以办到.
  - **实际研究领域相关**

    大致的分类的话, 个人感觉图像能够得出认知属性中的信息(二维空间以及时间信息(基于视频)).

    plain text 的话, 虽然其中包含的最表层的信息是只基于类型的, 但是由于语言中存在着实体以及关系之外的描述词汇. 因此可以从中习得认知属性, 但是相比与直接的感知而言, 挖掘出认知属性会是相当复杂的过程.
  - **强调**

    由于这个只是自己在没有相关认知学基础上的胡思乱想, 所以还是想给自己的描述加上一些条件.

    - 这里只是举出了两个抽象等级较高的分类标准，分类的层次不一定是最高, 但是这一个层次内应该只有这两个分类标准.
    - 第二, 其下面还有很多细粒度的分类标准.

- **利用知识的层级结构可以进行泛化.**

  例如上面说的例子, 在得知**泄漏**这个词汇可以和**情报**一起使用后, 我们怎么对**泄漏**的使用进行泛化呢?

- 已知条件: 一些实体和关系的一部分认知属性和一部分类型属性, 实例->"泄漏情报"
- 求解问题: "泄漏"可以搭配的词汇
- 方法: 假设已经知道了泄漏拥有一个认知属性(即上面那个), 但是不知道其相应的搭配, 通过实例之后, 我们不仅知道了 泄漏可以和情报搭配使用, 我们还知道了这种认知属性可以和那种类型属性一起搭配使用, 因此我们就可以将"泄漏"搭配上其他同属于"信息"类型的词汇, 例如-> "泄漏数据"

这种泛化其实是基于一个很强的假设:

- 实例类的可组合性 是 抽象类可组合性的充分必要条件.

这个例子里面, 具体的单词的某个词义就是实例, 这个词义的认知属性以及类型属性就是抽象类

其实像很多analogy之类的认为其实都可以认为是这个本质.

- **The Nature of Explanation**

  接着论文引用了1943年的一段文字, 用来解释 "解释的本质". 是学术界对人去理解解释世界的模型.

  简单来说, 上面我个人的理解是将抽象和类型完全进行了分离. 这个模型,使用的是范畴中的原型概念(认知语言学相关概念). 一个范畴中的一些原型词汇的使用方法, 决定了同一个范畴内的其他词汇和概念的使用方法. 这就使得, 有限的结构化的知识在接受有限的实例后可以拥有了无限表示的能力, 也就是说他将范畴内的原型的组合方法拓展到了其他的同范畴内的其他词汇.

  我们在学习的时候(接受一个正确实例的时候), 可以做两个事情

  - 将新的知识放入已有的结构化知识框架.
  - 调整框架去适应新的知识.

- **现在的实现组合泛化的理论**

  logic, grammars, classic planning, graphical models, causal reasoning, Bayesian nonparametrics, and probabilistic programming

其下面的子领域都明确的以实体和关系为中心进行学习

- **为什么结构化的方法如此重要?**

  这也是本文强调的一个重点, 结构化对应的是这篇论文的主题, 也就是关系归纳偏置. 其对应的是神经网络模型,深度网络模型中的弱归纳偏置, 深度网络的端到端的设计哲学通过最小先验表示来计算优化和计算的, 并且使用的是简单结构, 以及尽可能的避免人工特征, 但是这样就需要大量的数据和大量的计算时间. 并且其无法解决一下几种问题:

  - 复杂的语言以及场景的理解
  - 结构化数据的推理
  - 不同训练条件下的迁移学习
  - 小数据量的任务

  > 关于批评深度模型的论文见论文列表2. [2]

  弱归纳偏执对应的是数据的低学习效率, 也就意味着需要大量的数据.

  但是数据是非常昂贵的(在特殊领域确实是这样, 并且需要应用的任务大部分都是特殊领域内的问题, 比如说上面提到的结构化数据的推理). 因此通过增强归纳偏置来提供一个结构化的方法是非常重要的.

- **结构化方法和现在深度方法一定是对立的嘛?**

  - **早期的结构化方法**

    这些是早期的连接主义学家面对结构化数据和问题时提出的一些解决方法:

    - 类比决策?(analogy-making)
    - 语言学分析
    - 符号理论
    - 其他见**论文列表3**[3]

  - **近期与连接主义的结合**

    利用这些理论, 出现了一些使用深度模型去利用分布式的连续的向量表示文本, 图, 几何和逻辑甚至编程的方法. 详细见**论文列表4**[4]

这些论文提出, **结构化和灵活性并不是对立的!!!!!**. 我们应该将两者的组合泛化作为今后的目标!!!!!并且提出了一些相关的研究. 这些研究都表现出了很好的效果. 详细见**论文列表5**[5].

这些论文中的方法都有一个很大的特点就是:

performing computation over discrete entities and the relations between them.

> 这个真的很激动人心!!!!!!!!!!!!
>
> 在之前看有关认知学的书的时候, 就得知人的思维既拥有连续特性, 又拥有离散特性, 这个论文将两者结合了!!!!!

- **符号主义(离散)和连接主义(连续)结合的思想**

> 具体可以看论文 : Mitchell, T. M. (1980). The need for biases in learning generalizations

然后, 我们其实很容易就可以想到这一点, 就是学习知识需要离散和连续表征, 但是到目前的经典方法无法实现这个的原因是, **不知道如何去学习实体和关系的结构表征, 以及对应的计算的方法表征(即将其转换为计算机可理解的形式),而要想解决这些问题, 继续要事先指定他们**

> 解释一下上面加黑的一句话:
>
> 最后的他们指的是, 实体关系的抽象结构的表示方法, 不是说制定具体的关系和实体有什么关系, 知道了这些也就没必要学习直接应用就好了. 这里需要提前指定的是这些实体, 关系以及方法的抽象表示法.比如说:
>
> 我们要想去学习DNA的结构, 就需要提前指出, DNA之间的组成结构是双螺旋结构.
>
> 要想学习化学分子和其间关系的, 就需要提前指出, 化学分子其转化应满足的基本原则, 比如物质守恒, 能量守恒, 价守恒等等(化学忘完了, 胡绉的,大概这个意思)

而这种解决方法就需要**强关系归纳偏置**去约束, 约束的具体形式就是设计一些特殊的结构, 这些结构可以引导对实体表征和关系表征的学习.

更多具体方法见**论文列表6**[6]

- **接下来的文章构造介绍**
  - 介绍目前的已经出现的一些强关系归纳偏置模型

    > 其实这种思想一直有人在用, 但是都没有意识到其本质.这里想到于用例子去解释.
    >
    > 其实, 笔者认为, CNN, RNN, attention , 甚至DNN等等划时代模型的背后都有着对应的归纳偏置.(个人看法不一定正确)
    >
    > CNN是利用到了临近单元具有某种特殊关系的假设; RNN用到了序列的时序性假设;attention利用的是信息量的信息表达形式的不平衡假设;DNN用到的是知识的多层级性假设.
    >
    > 这些就是一种对知识的结构性描述.

  - 提出一种通用框架去描述基于实体和基于关系的推理, 也就是graph network. 统一并扩展了现在基于图的处理方法, 并描述了使用graph network作为基本单元处理时的核心设计思想.

    > 其实看到这里, 大概知道文章想做什么了.
    >
    > 这篇文章想要把所有的神经网络框架做一个抽象化, 即用一个抽象的方法去表示所有的模型.
    >
    > 比如, 把CNN抽象成一个图. 或者把RNN抽象出一个图. 等等.
    >
    > 从中去发掘设计结构化知识思想的本质.

2. Relational inductive bises

2.0 基本概念解释

2.0.1 relational reasoning

详见笔记 [知识点笔记]:结构化表征和结构化计算.

图模型是机器学习中用来处理关系推理的方法之一. 图模型可以通过明确随机变量之间的随机条件独立性来表征复杂的联合分布. 该模型可以捕捉构成真实世界的生成过程的稀疏结构.

文章还讲了几种图模型的具体算法, 自己看吧.

感觉这篇文章应该是用图模型去抽象化深度模型

2.0.2 Inductive biases

归纳偏差解释详见笔记 [知识点笔记]:归纳偏差

2.0.3 关系归纳偏置

深度学习中的各种模型其实就可以看作是由不同的初级building block组成的复杂的深层的层级结构或者图结构.

> 注释 : 下面说的 block 就是在2.0.1中提到的 "个体".

初级building block包括, “fully connected” , "convolutional layers"等等等等很多.

复杂的结构, 例如 MLP就是由多个 “fully connected” 组成的. 这个时候, "MLP" 也可以作为一个block.

而CNNs就是由 "convolutional layers" 和 "MLP" 组成的一个新的block.

最重要的是, **不同的block都包含着不同的 "relational inductive biase"**, 也就是不同的设计指导思想.

下面是几个例子:

Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

Table 1: Various relational inductive biases in standard deep learning components. See also Section 2

除此之外, 还有还有很多 **no-relational inductive biase**.

- activation non-linearities
- weight decay,
- dropout ,
- batch and layer normalization ,
- data augmentation, training curricula
- optimization algorithms all impose constraints on the trajectory and outcome of learning.

这些称为非关系归纳偏置, 与上面的关系归纳偏置进行对比, 我们就可以大概知道:

关系归纳偏置 是 应用了关系推理的归纳偏置. 是具有生成能力的.

2.0.4 推广到深度模型

将 relational reasoning 推广到深度模型的话就是:

entities和relaitions是分布表征. 也就是不同block的输出.

rules是 neural network function approximators(神经网络函数逼近器)、也就是各层之间的激活函数.

> 事实上, 这里对三者的定义没有和前面举的例子完全吻合. 我也搞不清三者的抽象定义到底是什么.
>
> 现在只考虑深度模型中的 entities, relations和rules.
>
> entitis就是模型中的unit, 一般都是框架中的variable变量. 就是模型图中的小圈圈.
>
> relations是制定那些unit和哪些unit之间有联系, 就是模型图中的连线. 包含weight, bias这些变量, 以及sigmoid函数或者ReLU这些个激活函数.
>
> rules是制定连线之间进行的操作.

每个深度模型的entities, relations 和 rules都是不一样的, 为了更好的理解他们, 这里定义了几个术语.

- arguments : 是 rule functions的输入, 就是entities和relations
- reused, shared : 指的是 rule functions 是否被reused或者shared(也就是说不同relation(连线)之间是否使用相同的rules(激活函数,不同权重算不同函数)).
- interactions, isolation : 得出的结果是全部entities之间协作的结果的话就是interactions, 否则就是isolation.

2.1 深度模型中的关系归纳偏置

2.1.1 Fully connected

- entities : the units in the network
- relations : all-to-all
- rules : specified by the weights and biase
- The argument to the rule is the full input signall, there is no reuse, and there is no isolation of information.
- relational inductive bias : very weak, all input units can interact to determine any output unit's value, independently across output

2.1.2 Convolutional layers

- entities : 还是独立的unit
- relations : 是稀疏的, 不再是全部连接.
- 关系归纳偏置 : locality and translation invariance
  - locality : 意思是空间相近的实体有联系远的没有.
  - translation invariance : 局部单元对于rules(卷积子)的复用,Spatial translation
- 远距离单元之间是isolated 的

2.1.3 Recurrent layers

- rule : 接受 a step’s inputs 和 hidden state作为arguments.
- reused : The rule is reused over each step
- relational inductive bias : temporal invariance
- 全部是Interaction.

2.2 Computations over sets and graphs

2.2.1 具有处理结构性知识的模型是什么样的?

- **一句话总结**

  这个论文想做的一个是实体和关系的明确表示和抽象化数学定义, 以及利用这种数学定义去表达知识中的结构特性.

  我们到目前的关于模型的定义通常采用图示,以及一些特殊的定义, 并没有赋予他们在一个广义框架下的明确的定义方式.

  并且我们还有设计一个学习算法去找到他们的方式, 也就是rules. 也就是合适的参数.

  真实世界的物体之间通常是没有order的, 但是, 他们之间的order是可以通过他们的属性的关系去决定的.

  > 例如, 词向量的顺序.

  顺序的不变性, 应该是需要通过深度模型的结构或者组件进行规范和反映的, 这样的模型结构就具有了 (一定程度的)relational reasoning 的能力.

  > 这里强调了结构性知识的顺序性, 但是结构性知识应该还具有其他特性, 例如两个知识的离散可操作性?(个人定义), 或者下面提到的知识的相互无关性和成对交互性等等.

- **相互无关性和成对交互性的例子**

  这里以**相互无关性**为例去给了一个具体的结构性知识的属性去展示如何通过改变模型结构去"支持"这种属性.

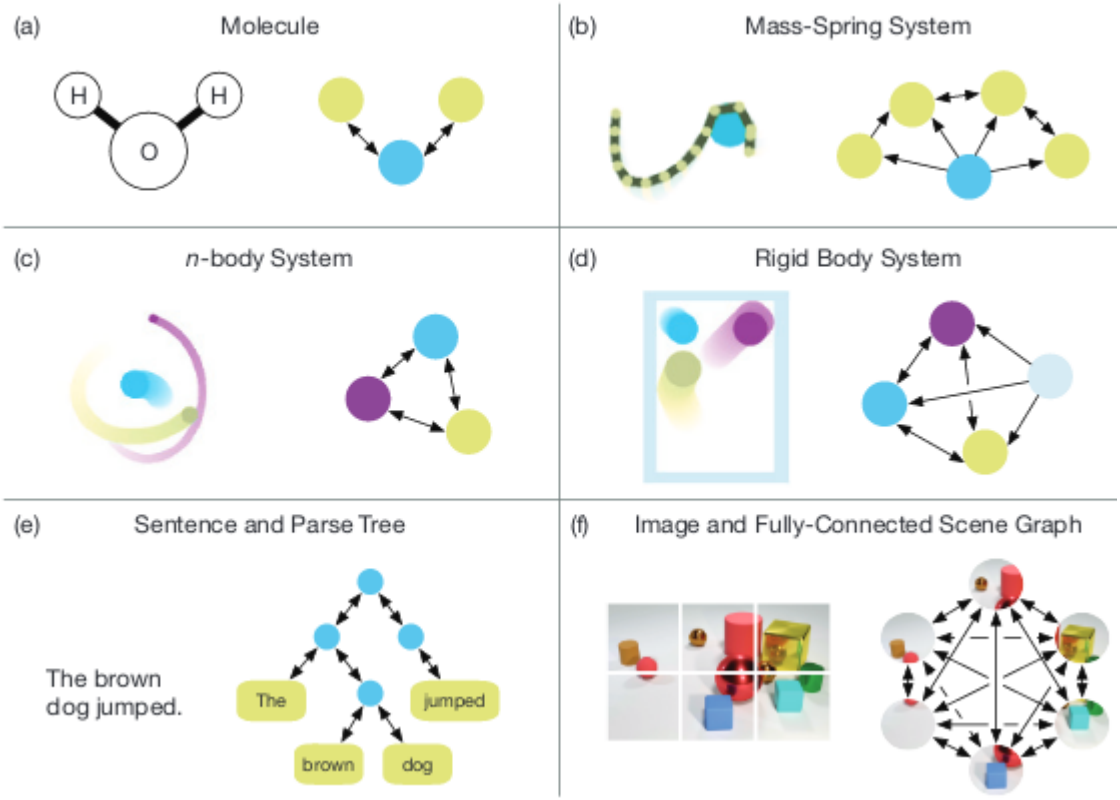
> 这里我对"支持"用了引号. 是因为, 一个结构属性对于一个模型而言, 不应该是存在不存在的问题(*relational inductive bias does not come from the presence*)
>
> 而应该是缺少不缺少的问题. (*but rather from the absence*)
>
> 存在 和 缺少 是有本质差距的, 下面是我对其的理解.
>
> 这里存在四个实体(认知概念上的实体) - 用灭霸的手套来举例
>
> - 一个模型(大框架) - 对应着 灭霸的无限手套
> - 一个待处理的任务 - 对应者 - 要消灭一般的生物
> - 多个模型中处理结构性知识的能力 - 对应者 - 需要控制时间, 力量, 心灵, 吧啦吧啦
> - 任务(知识)所拥有的结构性属性 - 对应着- 时间宝石, 力量宝石, 心灵宝石
>
> **这样我们就大概明白了, 当一个模型没有处理一种结构知识属性的话, 他就像手套上少一块宝石, 有一个空洞!!!!**

上一个黑点我们说了知识的顺序性, 但是这个也是相对于具体任务的, 对于某些任务是不需要使用顺序性的.例如利用已经知道的星系中的行星位置重量体积等信息去计算这个星系的中心位置. 这个问题中, 我们就不需要使用知识的顺序性, 反之这个里面拥有知识的相互无关性, 也可以理解为对称性. 其相应的结构叫做 Deep Sets model .

> 文章中还举出了当求解的是各个行星在一段时间之后的位置的时候, 需要利用到新的特性 - **成对交互性(全部成对交互)**, 这里就不细说了.

- **一些其他的结构**

  这里以上面的 **相互无关性** 和 **成对交互性(全部)** 作为两个极端, 提出了一些中间结构 **成对交互性(部分)**:



> 我的思考:
>
> 这个里面其实只用到了一个二元关系去表达 实体(unit) 之间的 关系(relation), 即 关系是 0或者1.
>
> 如果将 rules 考虑在内的话, 就可以表示多元关系. 上面说, 将会设计学习算法去学习 rules. 拭目以待吧.

3. Graph networks

本章组成, 首先简介过往有关 'graph nerual network' 任务. 其次, 提出自己的 'graph network framwork'.

3.1 Background

第一段, 首先介绍了很多利用网络模型做的任务, 很多很多, 里面提到了几个感兴趣的研究,记在论文列表6.

第二段, 介绍了一些利用个网络模型去解决需要对离散的实体和关系进行推理的任务, 例如组合优化问题, boolean satisfiability(布尔可满足性问题), 以及performing inference in graphical models等等,还有一个很有趣的方向是building generative models of graph, 对这四个方向感兴趣的论文列表7.

最后一段, 更多的有关graph neural networks的论文, 见论文列表8.

这一节非常非常重要, 介绍了很多很有趣的方向.

3.2 Graph network (GN) block

3.2.0 Introduction

- **一句话总结**

  GN框架基于图结构表征(表示), 定义了一组用来进行关系推理的函数.

  > *defines a class of functions for relational reasoning over graph-structured representations*

- **功能**

  最重要的功能是,

  supports constructing complex architectures from simple building blocks

  就是从一些基本的 building blocks 去构建复杂的结构.(并且后续会发布新的框架.)

- **组成**

  - GN框架的基本单元是 **GN block**.
  - 输入是graph. (表现在网络中就是unit, 结合任务比如说神经翻译,就可以是词向量, 字符向量等等)
  - 输出是graph
  - 执行的是基于结构的计算, 也就是最最开始说的 structural computation.
  - graph 的 node 是实体(entities), edges 是 关系(relations)

  具体的见 3.2.1 节.

3.2.1 Definition of “graph”

graph 的定义

- node v_i : 拥有属性 attribute, 抽象含义是一个node, 具体值是 attribute v_i .

  $V = \{v_i\}_{i=1:N_v}$, 是所有nodes的集合

- edge e_k : 拥有属性, 抽象含义是一个edge, 具体值是 attribute e_k

  $E = \{(e_k, r_k, s_k)\}_{k=1:N_e}$, 是所有edges的集合. 其中 r_k,e_k是输入的子图或者node的indice

- 全局属性 u : 图的全局属性.

> 上面是三个基本组成元素.

- "sender" node v_{s_k} : edge e_k 的发射端node
- "receiver" node v_{r_k} : edge e_k 的接受端node

3.2.2 Internal structure of a GN block

GN Block 的构造

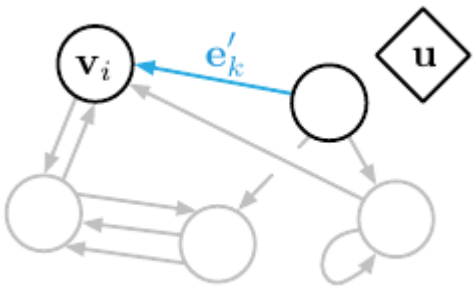
- **三个更新函数**

  三个更新函数中含有三个聚合函数

  - **对每一个边的状态进行的更新:**

    $$\mathbf{e}'_k = \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right)$$

    图示(蓝色为输出, 黑色为输入):



(a) Edge update

- **对每一个节点的状态进行的更新:**

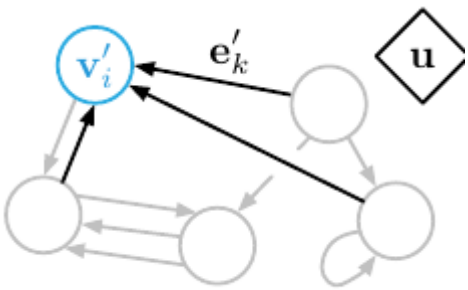
$$\mathbf{v}'_i = \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right)$$

其中,

$$\bar{\mathbf{e}}'_i = \rho^{e\to v}\left(E'_i\right)$$

他是一个聚合函数, 其接受参数为一个边的集合. 然后利用集合中所有边的信息去调整一个node的状态.

图示:



(b) Node update

- **对全局状态进行跟新.**

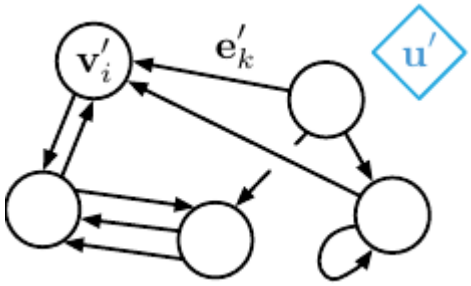
$$\mathbf{u}' = \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right)$$

其中包含两个聚合函数,

$$\bar{\mathbf{e}}' = \rho^{e\to u}\left(E'\right)$$
$$\bar{\mathbf{v}}' = \rho^{v\to u}\left(V'\right)$$

  - 第一个, 其接受参数为一个边的集合. 然后利用集合中所有边的信息去调整全局状态.
  - 第二个, 其接受参数为一个点的集合. 然后利用集合中所有点的信息去调整全局状态.

图示:



- **聚合函数需要具有的特点**
  - 对于参数的顺序不敏感
  - 可以接受不定数的变量
  - 举例 : elementwise summation, mean, maximum
- **个人总结**

  可以注意到, 这个GN Block并没有指定具体的node数和edge数, 其定义的只是点之间, 边之间, 点和边之间, 以及点, 边, 图之间的关系. 相当于生成语法中定义了 终端子, 非终端子, 文法等等这样的感觉, 剩下的就只是如何去组合.

3.2.3 Computational steps within a GN block

算法:

Algorithm 1 Steps of computation in a full GN block.

function GRAPHNETWORK(E, V, $\mathbf{u}$)
  for $k \in \{1 \dots N^e\}$ **do**
    $\mathbf{e}'_k \leftarrow \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right)$ ▷ 1. Compute updated edge attributes
  end for
  for $i \in \{1 \dots N^n\}$ **do**
    let $E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i,\, k=1:N^e}$
    $\bar{\mathbf{e}}'_i \leftarrow \rho^{e\to v}\left(E'_i\right)$ ▷ 2. Aggregate edge attributes per node
    $\mathbf{v}'_i \leftarrow \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right)$ ▷ 3. Compute updated node attributes
  end for
  let $V' = \{\mathbf{v}'\}_{i=1:N^v}$
  let $E' = \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:N^e}$
  $\bar{\mathbf{e}}' \leftarrow \rho^{e\to u}\left(E'\right)$ ▷ 4. Aggregate edge attributes globally
  $\bar{\mathbf{v}}' \leftarrow \rho^{v\to u}\left(V'\right)$ ▷ 5. Aggregate node attributes globally
  $\mathbf{u}' \leftarrow \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right)$ ▷ 6. Compute updated global attribute
  return $(E', V', \mathbf{u}')$
end function

这里的先对哪部分函数进行更新并不是一个定值, 而是可以选择的.

3.2.4 Relational inductive biases in graph networks

这个地方又重新开始精彩了!!!!!

在第三章之前我们一直在强归纳偏置的重要性, 那么在这里将要介绍这个架构如何去实现强归纳偏置!!!1

1. graph本身可以根据实体和关系去设计!!!!

   在之前的网络结构中, 我们通常是对node没有进行关于输入的分别, 而是一视同仁, 即任何词汇都可以instantiated到每个node. 但是, 其实(为了准确, 放上原文):

   graphs can express arbitrary relationships among entities, which means the GN's input determines how representations interact and are isolated, rather than those choices being determined by the fixed architecture.

   个人理解是, graph的node之间的关系是可以根据对node的输入来定的, 这样的话, 网络架构就会变成一个动态的架构!!!

2. 图关于entities和他们之间的relations是的表征是基于set的. 也就是说, 对于参数位置的交换是没有反应的. GN block对于参数的顺序不敏感, 这个就支持了对于很多对顺序不敏感的模型的支持.
3. 每个node和每个edge都是可以被复用的, 这个就支持了组合泛化. 这个是相当重要的. 在决定了GN block后可以根据实例改变size以及shape.

4. Design principles for graph network architectures

上面说的是最广义的GN block. 下面针对的是对于深度模型的 graph network 设计规则. 使得GN block 成为一个**可学习的 graph-to-graph function approximator.**

4.1 Flexible representations

灵活性体现在两点:

1. 属性值表征的灵活性
2. 图的结构本身的灵活性

4.1.1 Attributes

这里的属性值代表的是 edges, nodes, globel 的值.

- **属性值的形式**

  属性的具体数值形式是依据任务的不同而不同的. 在深度模型中, 通常是采用real value vector or tensor 的形式.

- **输出的单元**

  根据任务的不同, 输出的东西也是可以不同的. 有三种基本类型的输出.

  - edge-focused GN : 采取边作为输出, 例如下面的研究:
    - **Neural relational inference for interacting systems (必读)**

      这个研究是利用变分自动编码器去模拟物理动力学系统的研究, 也就是说研究的是物体和物体的之间的联合计算函数. 这个就是很典型的, edge-focused GN 的应用.

    - Relational inductive bias for physical construction in humans and machine
  - node-focused GN : 采取点作为输出. 这个应该就是我们一般的研究.
  - graph-focused GN : 采取global作为输出. 例如 predict the potential energy of a physical system.

> 还可以有三种的混合形式:
>
> Relational inductive bias for physical construction in humans and machine:

> 这个研究同时利用edge和global去预测一个(人和机器)基于动作的策略.

4.1.2 Graph structure

图的结构有两个极端, 一个是实体显式地决定其之间的关系结构. 一个是, 关系结构必须 被推断或假定. 也可以采取两者的混合.

- 第一个方式的例子:

  知识图谱以及句法树以及社会网络问题, 这样的问题中, edge是根据其相连实体来确定的.也就是说, $\rho^{v \to w}$ 是基于一个知识图谱或者句法树等等.

  > 这里的很典型的例子是 graph LSTM (用于远距离关系抽取). 在这个研究中, 词汇和词汇之间不仅具有临接关系还具有依存句法关系, 用不同的LSTM关联起拥有不同依存关系的两个词汇, 使用方法是,在LSTM中根据依存标签来设计参数. 这里就很能说明这里的方法了, 因为这里的 edge 同时满足了
  >
  > - relation(认知上的定义) - 是两个实体之间存在的关系.
  > - edge(graph上的定义) - 是连接两个点之间的线
  > - rules(逻辑上的定义) - 是用来处理两个entities之间的函数.
  >
  > graph LSTM与一般LSTM的计算不同如下:

$$
\begin{aligned}
f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right) \\
i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right) \\
h_t &= o_t * \tanh\left(C_t\right) \\
i_t &= \sigma(W_i x_t + \sum_{j \in P(t)} U_i^{m(t,j)} h_j + b_i) \\
o_t &= \sigma(W_o x_t + \sum_{j \in P(t)} U_o^{m(t,j)} h_j + b_o) \\
\tilde{c_t} &= \tanh(W_c x_t + \sum_{j \in P(t)} U_c^{m(t,j)} h_j + b_c) \\
f_{tj} &= \sigma(W_f x_t + U_f^{m(t,j)} h_j + b_f) \\
c_t &= i_t \odot \tilde{c_t} + \sum_{j \in P(t)} f_{tj} \odot c_j \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

- 第二个方式的例子:

  例如, visual scenes, 自然语料库等等. 这样的初始的情况下我们不知道实体之间的任何关系.

  具体的例子就是, 将句子中的每一个词汇视为一个node.

  有很多方法去通过非结构性数据去进行推断.比如说利用attention的机器翻译.

  对于这种没有关系的数据, 我们可以针对每一个数据实例化一个关系.

  > 或是全部使用同一归纳偏置极其弱的关系? 例如 LSTM.?

  但是现在也出来了一些**利用非结构性数据推断出稀疏结构的方法**, 如下面的论文:

  Neural relational inference for interacting systems (必读)

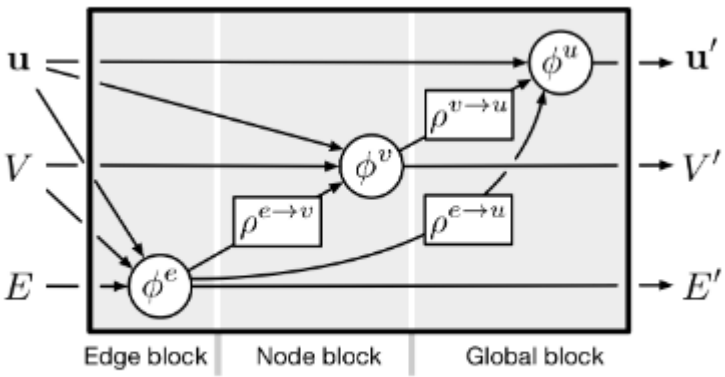
  > 发现和上面提到的必读文章是一致的, 果然这个文章很重要!!!!!!!

4.2 Configurable within-block structure

先放上几个重要论文, 见**论文列表9**[9]

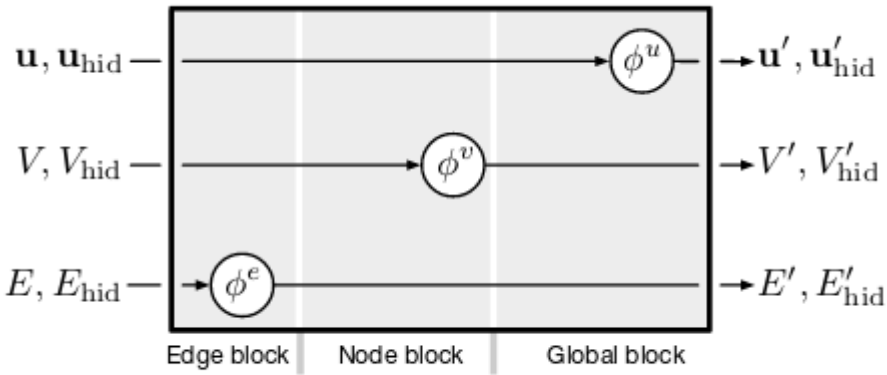
这一节讲述了通过定制 GN block 中的函数改变模型功能的案例. 体现了 GN 的灵活性. 在上面的3.2.2 节, 我们已经介绍了更新各个变量的基本公式. 里面介绍的是下面几个网络模型的 GN block结构.

- **Full GN Block**



(a) Full GN block

- **Independent recurrent block**

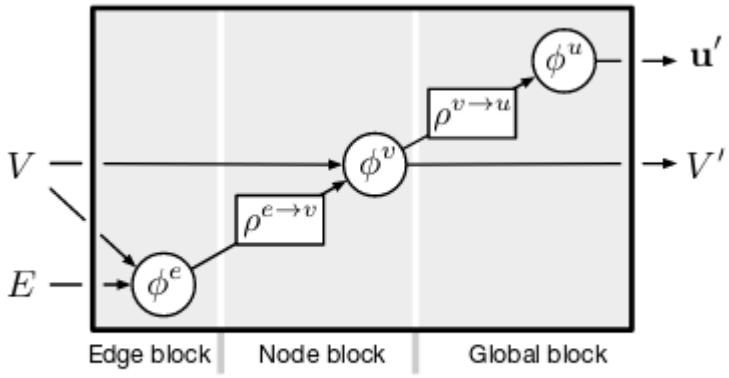


(b) Independent recurrent block

这里使用的 ϕ 是RNN

例如 tree Lstms, 但是这里有关问题是, 不论是什么模型, 边和点肯定要进行处理的啊, 这里的边和点却是相互独立的, 不知道是怎么更新的.

- **Message-passing neural network**



(c) Message-passing neural network

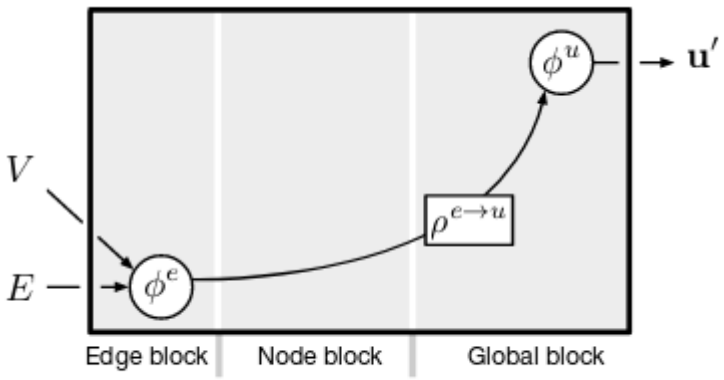
这个可以用物理动力学系统模拟的那篇文章去理解, 用边去更新点, 点去更新边.

- **Non-local neural network**

  见4.2.2

  只预测节点

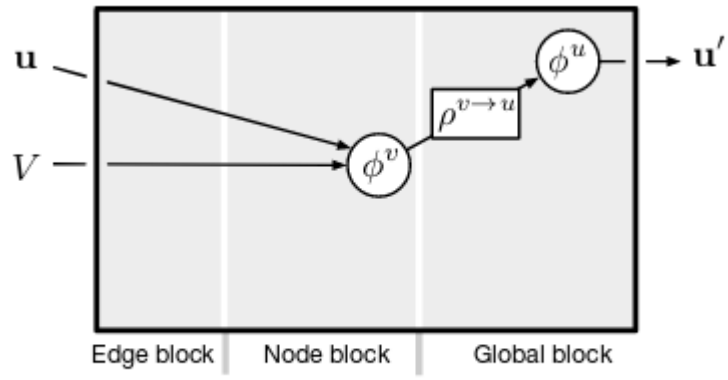
- **Relation network**



(e) Relation network

只预测全局属性的.

- **Deep set**



(f) Deep set

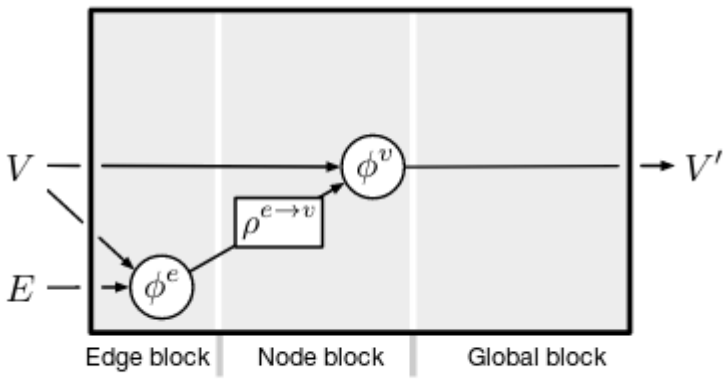
没有边的信息, 只有点和全局属性, 这是一个无序,无结构的模型. 比如最开始提到的, 星系中心点预测系统.

4.2.1 Message-passing neural network (MPNN)

不说

4.2.2 Non-local neural networks (NLNN)

这个模型很重要, 这个是基于加权图模型.



(d) Non-local neural network

- NLNN 使用了various “intra-/self-/vertex-/graph-attention” 方法
- attention 是 更新node的方法

  每个节点的更新是依靠其邻接点的加权和. 节点i和节点j之间的权重是通过两者属性值的a scalar pairwise function 来进行计算的.

NLNN中没有explictly使用edge embedding的方法, 因为, 这里只是利用了两者之间是否互相连接的信息.

这个模型有很多变种, 包括 vertex attention interaction network, 还有 graph attention network.

整个的计算过程如下:

$$\begin{aligned}\phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) &:= f^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) &&= \left(\alpha^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right),\ \beta^e\left(\mathbf{v}_{s_k}\right)\right) = \left(a'_k, \mathbf{b}'_k\right) = \mathbf{e}'_k \\ \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) &:= f^v(\bar{\mathbf{e}}'_i) \\ \rho^{e\to v}\left(E'_i\right) &:= \frac{1}{\sum_{\{k:\, r_k=i\}} a'_k} \sum_{\{k:\, r_k=i\}} a'_k \mathbf{b}'_k\end{aligned}$$

> 其中这个就是attenton模型的最抽象的结构
>
> 我们一开始可以假设输入(例如句子)中的词汇之间构成了一个全连接图. 每个单词是一个key
>
> 然后一整個句子的词向量是一个query, 这个query的获取就是 $\phi^{e\to v}$,
>
> 那么这个全连接图里面, 一个key与query 之间的attention的计算就包含在
>
> $$\phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) := f^v(\bar{\mathbf{e}}'_i)$$
>
> 之中.

4.2.3 Other graph network variants

介绍了很多其他的模型.

反正就是很多很多种模型了...

- Interaction Networks ,
- Neural Physics Engine
- a full GN but for the absence of the global
- CommNet

- structure2vec
- ated Graph Sequence Neural Networks
- Relation Networks
- Deep Sets

其实看到这里, 之前的热情已经消退, 本身以为这篇论文可以教给我们怎么去挖掘信息中的特殊结构, 但是他只是给了我们一个抽象的理解深度模型方法的框架,但是还是很重要, 感觉看完之后加深了对很多模型的理解

4.3 Composable multi-block architectures

可组合的多block结构.

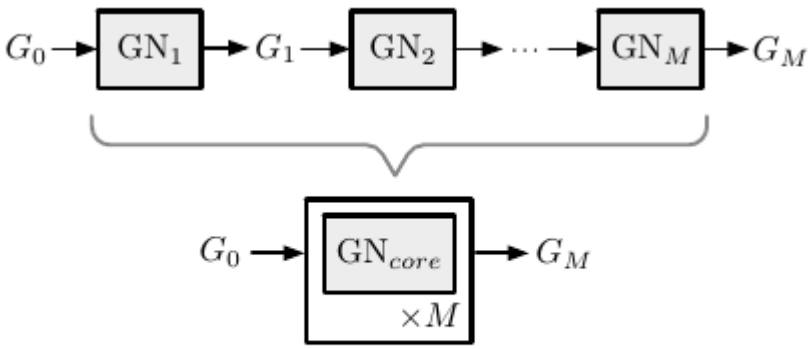
- **一句话总结**

  一个GN block的输入和输出都是 graph, 那么一个Block的输出就可以当成另外一個BLock的输入.即:

  $$G' = \mathrm{GN}_2(\mathrm{GN}_1(G)).$$

  下面介绍三个组合方式.

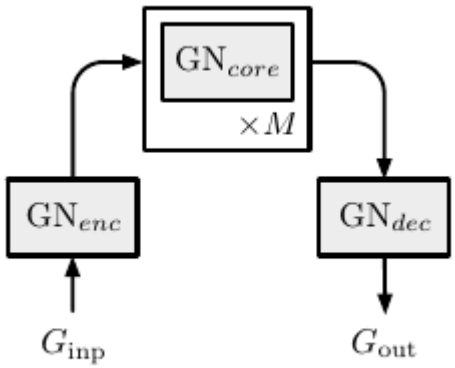
- **Composition of GN blocks**



(a) Composition of GN blocks

这个里面, 一个很直观的例子就是另外一篇论文笔记中提到的物理动力学系统, 这个系统中每个 GN block 可以视为对一个 $\triangle t$ 后系统中 object 的预测, 组合 M 个这样的 Block就可以视为是预测 $M* \triangle t$ 后object的状态.

- **Encode-process-decode**

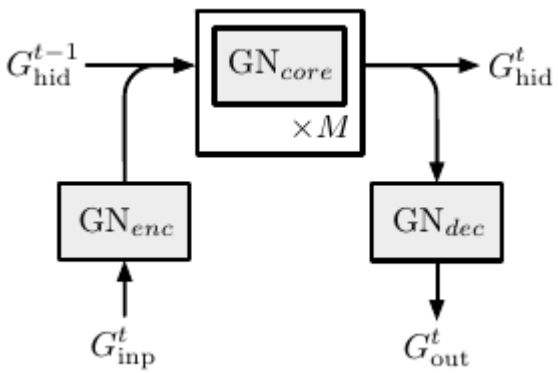


(b) Encode-process-decode

即encoder使用了一个 Block, decoder使用了一个Block, 还有一個处理中间状态的 Block.

还是物理动力学系统模型的例子, 在这个例子中一整個模型就是一个 Encode-process-decode.

- **Recurrent GN arch tecture**



这个可以用来预测时序的图结构数据.例如物理动力学系统中随着时间而不停变化的图的状态.

这里也可以使用LSTM或者GRU.

4.4 Implementing graph networks in code

可设计为并行运行的结构等等.

4.5 Summary

从三个角度讲了GNN的设计原则. 是从底到上的顺序

4.1 最底层, 从 GN 的属性设置和graph 边的设计方法阐释了 GNN 的灵活性

4.2 从GN 的多种变种阐述了 GNN 的灵活性

4.3 从GN 的组合来阐述灵活性.

5. Discussion

5.0 概述

在上面讲了graph network的基本结构, 灵活的使用方法, 这一节讲最核心的, 如何使用graph network 去设计复杂的结构去包含强归纳偏执到模型中.

5.1 Combinatorial generalization in graph networks

这一节介绍了证明 graph network 具有组合泛化能力的论文. 见**论文列表10** [10]

- 被设计用来预测One-step的模型可以用来预测上千步.
- 在object的数量发生变化时也可以保持高正确率.
- GN-based决策模型, 在agents数发生变化時也能很好的应对.
- 还可以生成一些之前没有的useful node embedding.
- 等等特性.

这些特性都证明了 graph network 具有 "从有限中生成无限" 的能力, 即组合泛化能力.

5.2 Limitations of graph network

不能解决的一些问题:

- 非同构图的判别的问题(不懂)
- 一些无法用图表示的结构的问题, recursion(递归), control flow(控制流), and conditional iteration(if-then-这些). 这些 “computer-like” processing 是图无法处理的结构, 但是这些对于人的认知判断是极为重要的, 也就是说, graph network 虽然很强大, 但是想相比于人的认知能力还是差很多必要因素.

  (什么样的模型可以整合进这些结构呢?见**论文列表11**[11])

5.3 Open questions

这一节的中心点在于, 认为在相比于某一个框架下进行调整(即在某一个模型上添添补补), 更加重要的是认识到 graph network 的全部潜力.

这里的问题就是围绕这"潜力"来的.

- **graph network 操作的 graph 是怎么得来的?**

  这个其实是一个最根本的问题. 也就是如何把原始感官数据转化为有结构的数据, 如今的wordnet, dependency parsing 其实都是在做这个工作.

  一个方法就是使用全连接图, 就是上面提到的attention思想的 Non-local neural networks.

  但是实际的结构应该是更为稀疏的. 有一些研究是做推导出稀疏结构的, 因为这部分很重要因此列在这里.**(重要!!)**

  - Visual interaction networks: Learning a physics simulator from video.
  - Relational neural expectation maximization: Unsupervised discovery of objects and their interactions
  - Learning deep generative models of graphs.
  - Neural relational inference for interacting systems.(见论文笔记)
- **如何在计算的过程中动态的改变 graph 的结构?**

比如, 在一个物体分成几个部分之后, 这个entity的node也应该分为几个node.

算法应该具有在计算过程中对edge进行增加和取消的能力.

有两个研究是基于这个的.**(重要!!)**

- **Learning deep generative models of graphs.**
- **Neural relational inference for interacting systems.**

5.4 Integrative approaches for learning and structure

强调本文主要在graph network计算部分, 而在设计思路上只给出了以下的几个方向

(重要!!)

- linguistic trees
- partial tree traversals in a state-action graph
- hierarchical action policies
- capsules
- programs
- 模仿计算机的软硬件结构的方法(难道是可以解决上面提到的graph无法操作的内容?)

详细论文, 论文中有名称(懒得写了)

5.5 Conclusion

强调了**组合泛化**的重要性.

作者在肯定 graph network的同时, 也承认了他的不足, 并且提供了几个作者认为有意思的有前途的但是不是很热门的方向.比较重要, 列在这里:

- Ritchie, D., Horsfall, P., and Goodman, N. D. (2016). Deep amortized inference for probabilistic programs

- Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. (2016). Neural module networks.
- Gaunt, A. L., Brockschmidt, M., Kushman, N., and Tarlow, D. (2016). Differentiable programs with neural libraries.
- Evans, R. and Grefenstette, E. (2018). Learning explanatory rules from noisy data.
- Evans, R., Saxton, D., Amos, D., Kohli, P., and Grefenstette, E. (2018). Can neural networks understand logical entailment?

还有一些强调抽象模型方法的论文: (重要!!)

- Schema networks: Zero-shot transfer with a generative causal model of intuitive physics.
- From skills to symbols: Learning symbolic representations for abstract high-level planning.
- Composable planning with attributes.
- Behavior is everything–towards representing concepts with sensorimotor contingencies.

元学习相关研究: (重要!!)

- Learning to reinforcement learn.
- Prefrontal cortex as a meta-reinforcement learning system.
- Model-agnostic meta-learning for fast adaptation of deep networks.