華東師範大學
EAST CHINA NORMAL UNIVERSITY
计算机科学与软件工程学院
School of Computer Science and Software Engineering
多智体人工智能实验室
Multi-Agent Artificial Intelligence Laboratory

Relational inductive biases, Deep learning, Graph Networks

Jarvis

Outline

Introduction

RIB

GN

Design Principles

More

Introduction

Introduction

Hand-engineering:

Sample Efficient

Interpretable

Reasoning Structured Data

End-to-end learning

Minimal priori knowledge

Avoid explicit structure

Relational Biases

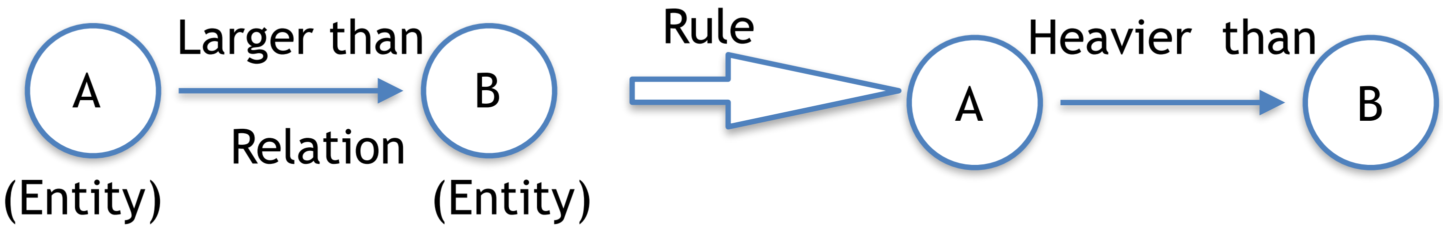
Relational Inductive Biases

Relational reasoning: manipulate entities and relations with rules.

Entity: element with attributes

relation: property between entities

rules: map entities and relations to entities and relations



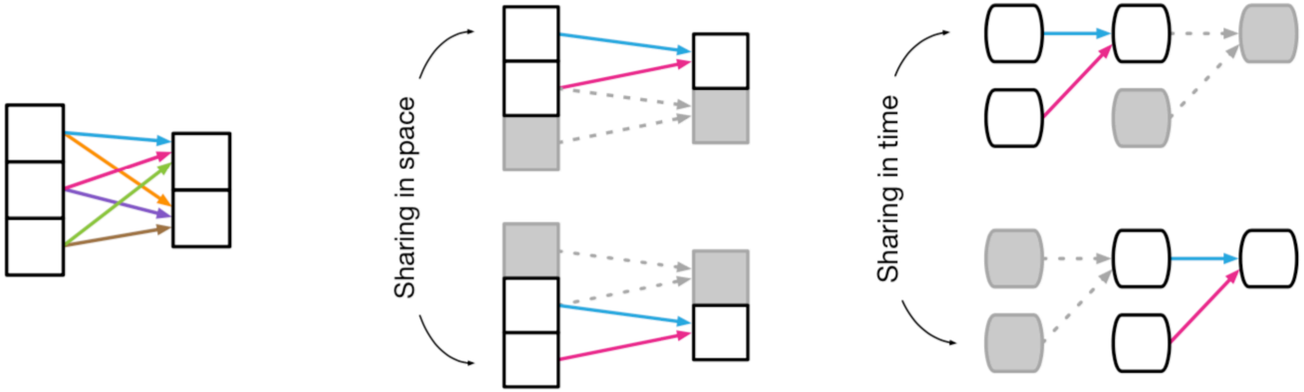
Relational Inductive Biases

Inductive bias: Allows algorithm to prioritize one solution

L2 regularizer: prefer small parameters

L1 regularizer: prefer sparse parameters

Relational Inductive Biases



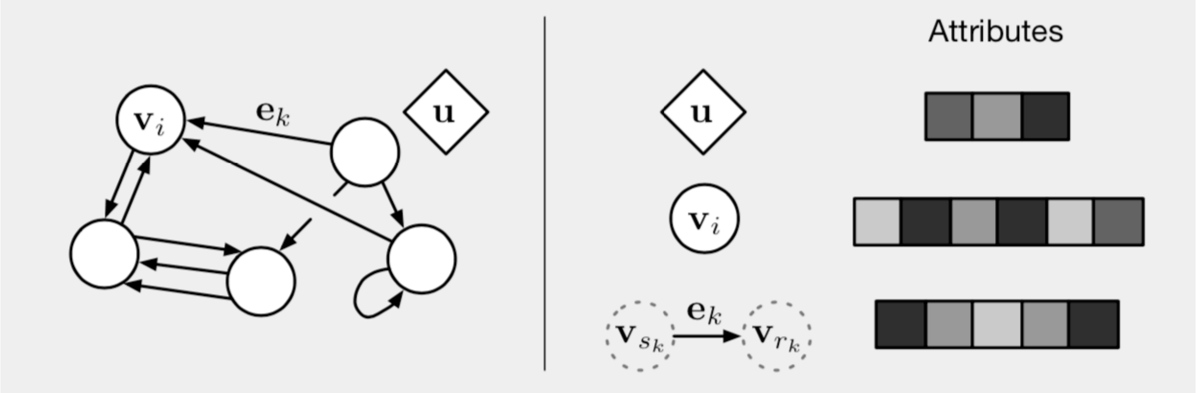
(a) Fully connected (b) Convolutional (c) Recurrent

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

GN

Graph Net

Graph: $G = (\mathbf{u}, V, E)$

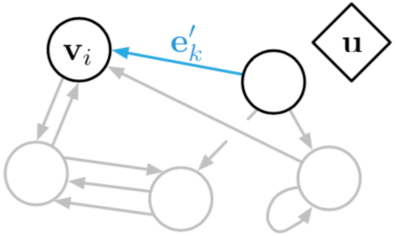


Nodes: $V = \{\mathbf{v}_i\}_{i=1:N^v}$ Edges: $E = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1:N^e}$ Global: $\mathbf{u}$

Graph Net

Update function:

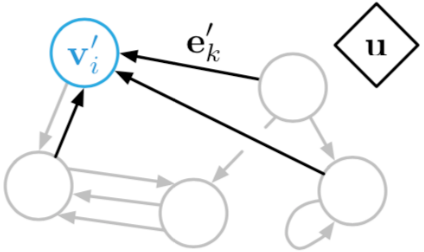
$$\mathbf{e}'_k = \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right)$$
$$\mathbf{v}'_i = \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right)$$
$$\mathbf{u}' = \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right)$$



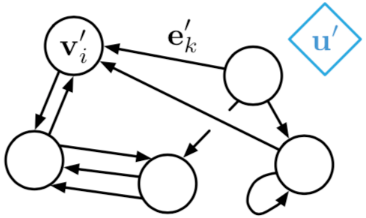
(a) Edge update

Aggregate function:

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



(b) Node update



(c) Global update

$E'_i = \left\{(\mathbf{e}'_k, r_k, s_k)\right\}_{r_k=i,\ k=1:N^e}$, $V' = \left\{\mathbf{v}'_i\right\}_{i=1:N^v}$, and $E' = \bigcup_i E'_i = \left\{(\mathbf{e}'_k, r_k, s_k)\right\}_{k=1:N^e}$.

Graph Net

Algorithm 1 Steps of computation in a full GN block.

function GraphNetwork(E, V, $\mathbf{u}$)
  for $k \in \{1 \ldots N^e\}$ **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 $i \in \{1 \ldots N^n\}$ **do**
    let $E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i,\ k=1:N^e}$
    $\bar{\mathbf{e}}'_i \leftarrow \rho^{e\rightarrow v}\left(E'_i\right)$ ▷ 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 $E' = \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:N^e}$
  $\bar{\mathbf{e}}' \leftarrow \rho^{e\rightarrow u}\left(E'\right)$ ▷ 4. Aggregate edge attributes globally
  $\bar{\mathbf{v}}' \leftarrow \rho^{v\rightarrow u}\left(V'\right)$ ▷ 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

Graph net

Advantages:

1. GN’s input determines how representations interact and are isolated rather than fixed structure

2. Invariant to permutations

3. Combinatorial generalization(reuse update and aggregate func)

Design Principles

Design Principles

Flexible representations:

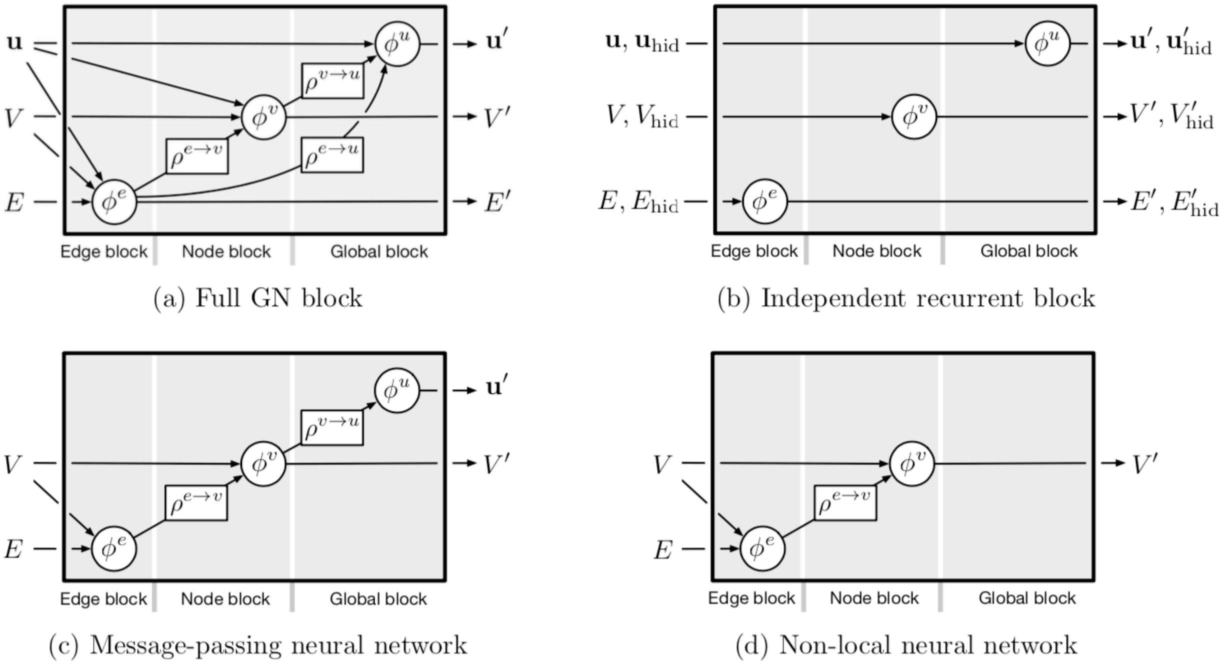
1. Nodes, edges, global features can be arbitrary format.
2. Output block can be edges or nodes or globals
3. The structure of graph can be defined or inferable

Design Principles

Configurable within-block structure

The within-block can be configured in many different way, which offers many invariants

Variants



(a) Full GN block

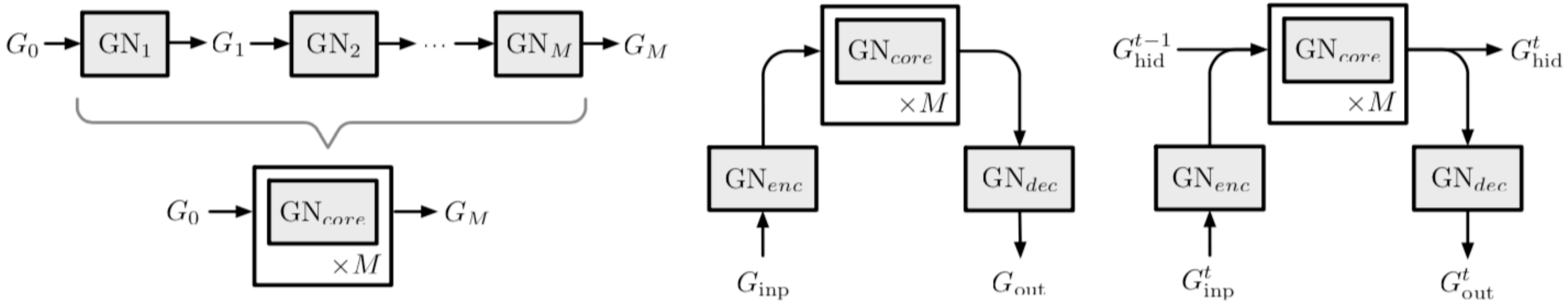
(b) Independent recurrent block

(c) Message-passing neural network

(d) Non-local neural network

Design Principles

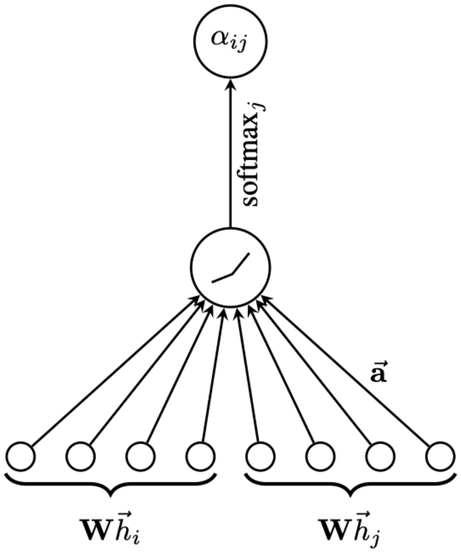
Composable multi-block

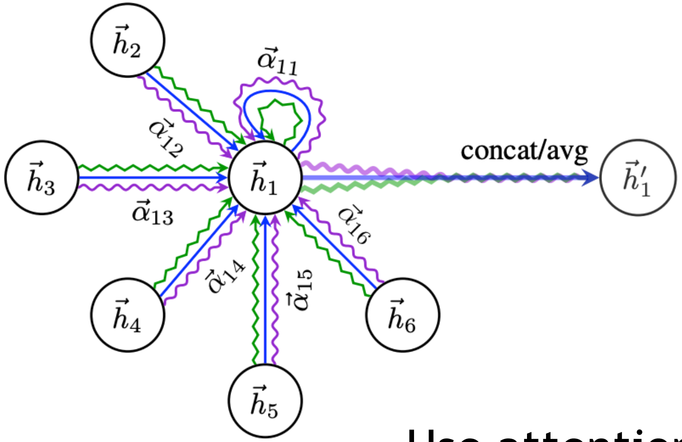


(a) Composition of GN blocks (b) Encode-process-decode (c) Recurrent GN architecture

More

Graph Attention Networks



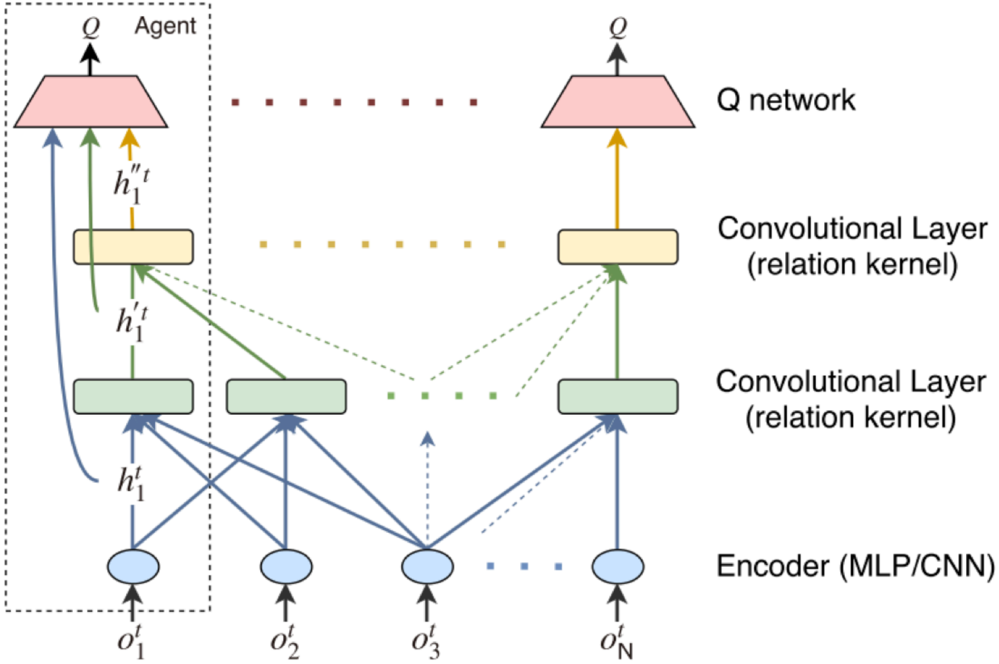


$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \vec{h}_j\right)$$

Use attention network as aggregate func

* GRAPH ATTENTION NETWORKS.ICLR.2018

GCN for MARL



$$\mathcal{L}(\theta) = \frac{1}{S}\sum_{S}\frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q\left(O_i, a_i; \theta\right)\right)^2$$

* Graph Convolutional Reinforcement Learning for Multi-Agent Cooperation. Jiechuan Jiang

GCN for MARL

Remarks:

1. Feed every layer into Q-Net to assemble and reuse features from different receptive fields.

2. Encourage self consistent relation representation by add KL to loss

$$\mathcal{L}(\theta) = \frac{1}{\mathrm{S}} \sum_{\mathrm{S}} \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} ((y_i - Q(O_i, a_i; \theta))^2 + \lambda D_{\mathrm{KL}}(\mathcal{G}^{\kappa}(O_i; \theta) \| z_i))$$

$z_i = \mathcal{G}^{\kappa}\left(O_i'; \theta\right)$ is the attention weight distribution at conv layer k

* Graph Convolutional Reinforcement Learning for Multi-Agent Cooperation. Jiechuan Jiang