Обзор статьи: Relational inductive biases, deep learning, and graph networks

Малиновский Григорий Станиславович

Московский физико-технический институт

ВЦ РАН, Москва, 2019

Постановка задачи

Проблема

Противостояние двух подходов в искусственном интеллекте: “hand-engineering” vs “end-to-end”

Задача

Обобщение комбинаторных структур является главным приоритетом в развитии искусственного интеллекта. Требуется найти способ совместить два подхода для обобщения классических нейросетевых архетектур, а также расширения возможных комбинаторных структур для более гибкого и качественного обучения.

Методы

Предложена новая структура graph network, которая обобщает в широком смысле нынешние распространенные архитектуры. Также авторами разработана библиотека graph nets.

Необходимые термины

We define *structure* as the product of composing a set of known building blocks. "Structured representations" capture this composition (i.e., the arrangement of the elements) and "structured computations" operate over the elements and their composition as a whole. Relational reasoning, then, involves manipulating structured representations of *entities* and *relations*, using *rules* for how they can be composed. We use these terms to capture notions from cognitive science, theoretical computer science, and AI, as follows:

- An *entity* is an element with attributes, such as a physical object with a size and mass.
- A *relation* is a property between entities. Relations between two objects might include SAME SIZE AS, HEAVIER THAN, and DISTANCE FROM. Relations can have attributes as well. The relation MORE THAN X TIMES HEAVIER THAN takes an attribute, X, which determines the relative weight threshold for the relation to be TRUE vs. FALSE. Relations can also be sensitive to the global context. For a stone and a feather, the relation FALLS WITH GREATER ACCELERATION THAN depends on whether the context is IN AIR vs. IN A VACUUM. Here we focus on pairwise relations between entities.
- A *rule* is a function (like a non-binary logical predicate) that maps entities and relations to other entities and relations, such as a scale comparison like IS ENTITY X LARGE? and IS ENTITY X HEAVIER THAN ENTITY Y?. Here we consider rules which take one or two arguments (unary and binary), and return a unary property value.

Индуктивные отклонения

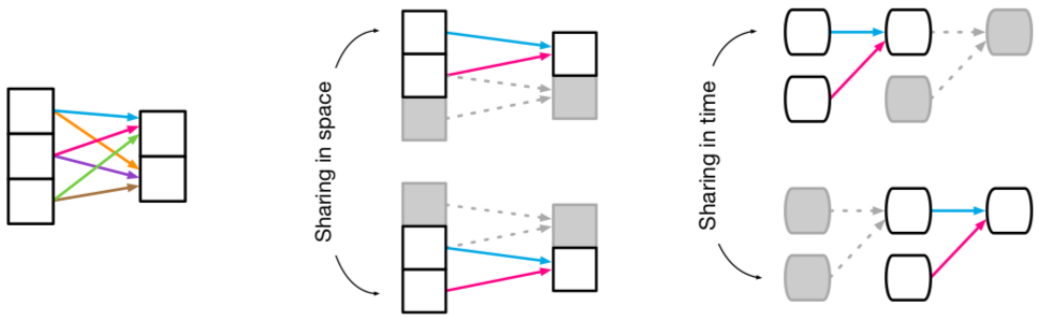
Box 2: Inductive biases

Learning is the process of apprehending useful knowledge by observing and interacting with the world. It involves searching a space of solutions for one expected to provide a better explanation of the data or to achieve higher rewards. But in many cases, there are multiple solutions which are equally good (Goodman, 1955). An *inductive bias* allows a learning algorithm to prioritize one solution (or interpretation) over another, independent of the observed data (Mitchell, 1980). In a Bayesian model, inductive biases are typically expressed through the choice and parameterization of the prior distribution (Griffiths et al., 2010). In other contexts, an inductive bias might be a regularization term (McClelland, 1994) added to avoid overfitting, or it might be encoded in the architecture of the algorithm itself. Inductive biases often trade flexibility for improved sample complexity and can be understood in terms of the bias-variance tradeoff (Geman et al., 1992). Ideally, inductive biases both improve the search for solutions without substantially diminishing performance, as well as help find solutions which generalize in a desirable way; however, mismatched inductive biases can also lead to suboptimal performance by introducing constraints that are too strong.

Примеры архетектур

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.



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

Figure 1: Reuse and sharing in common deep learning building blocks. (a) Fully connected layer, in which all weights are independent, and there is no sharing. (b) Convolutional layer, in which a local kernel function is reused multiple times across the input. Shared weights are indicated by arrows with the same color. (c) Recurrent layer, in which the same function is reused across different processing steps.

Примеры представлений

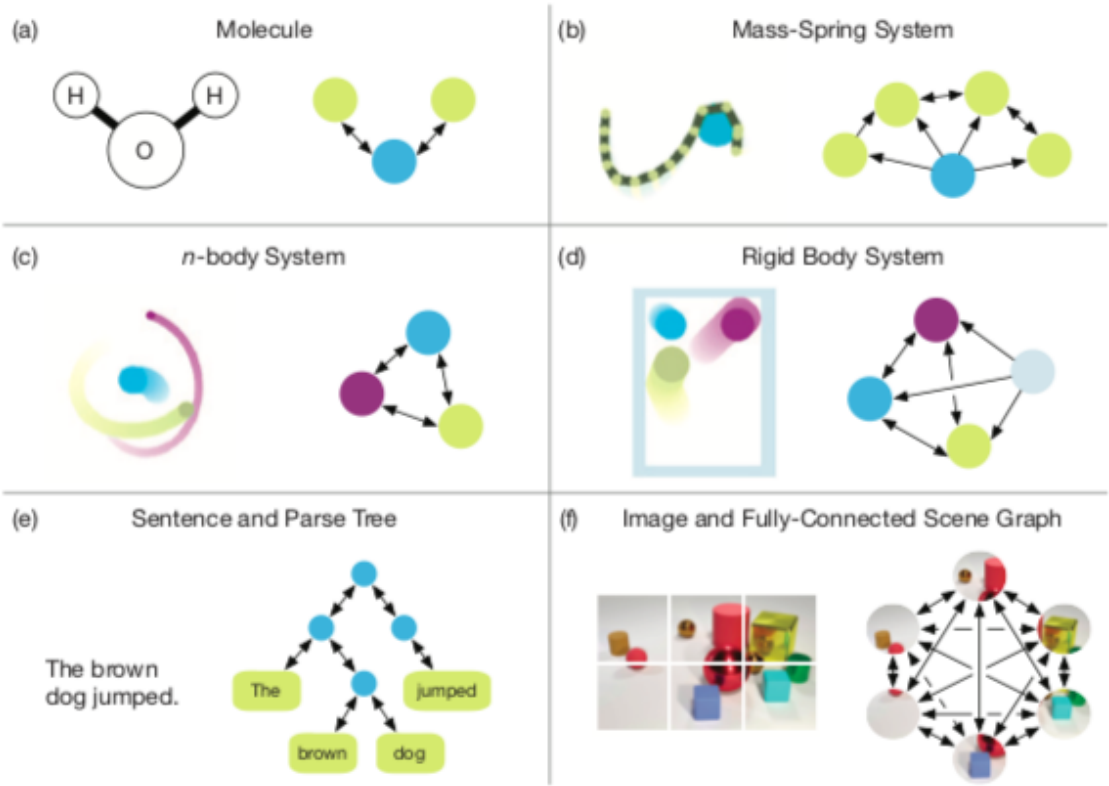
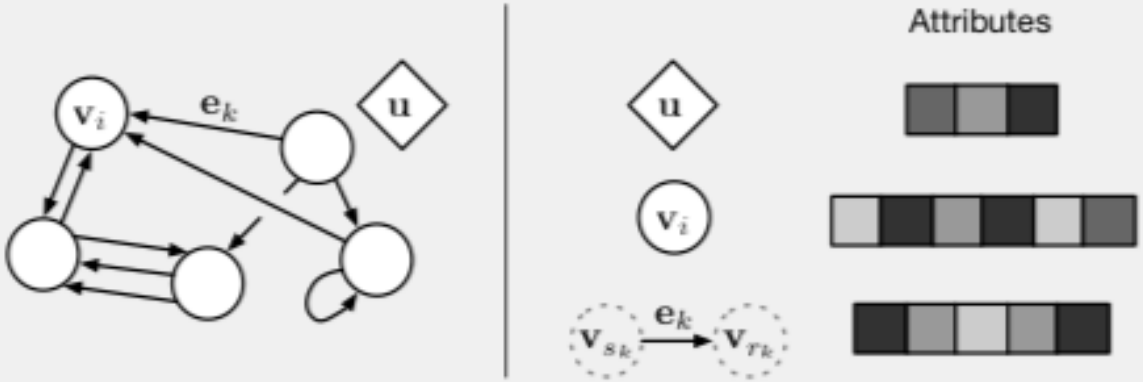


Figure 2: Different graph representations. (a) A molecule, in which each atom is represented as a node and edges correspond to bonds (e.g. Duvenaud et al., 2015). (b) A mass-spring system, in which the rope is defined by a sequence of masses which are represented as nodes in the graph (e.g. Battaglia et al., 2016; Chang et al., 2017). (c) A n-body system, in which the bodies are nodes and the underlying graph is fully connected (e.g. Battaglia et al., 2016; Chang et al., 2017). (d) A rigid body system, in which the balls and walls are nodes, and the underlying graph defines interactions between the balls and between the balls and the walls (e.g. Battaglia et al., 2016; Chang et al., 2017). (e) A sentence, in which the words correspond to leaves in a tree, and the other nodes and edges could be provided by a parser (e.g. Socher et al., 2013). Alternately, a fully connected graph could be used (e.g. Vaswani et al., 2017). (f) An image, which can be decomposed into image patches corresponding to nodes in a fully connected graph (e.g. Santoro et al., 2017; Wang et al., 2018c).

Определение графа

Box 3: Our definition of "graph"



Here we use "graph" to mean a directed, attributed multi-graph with a global attribute. In our terminology, a node is denoted as $\mathbf{v}_i$, an edge as $\mathbf{e}_k$, and the global attributes as $\mathbf{u}$. We also use s_k and r_k to indicate the indices of the sender and receiver nodes (see below), respectively, for edge k. To be more precise, we define these terms as:

Directed : one-way edges, from a "sender" node to a "receiver" node.
Attribute : properties that can be encoded as a vector, set, or even another graph.
Attributed : edges and vertices have attributes associated with them.
Global attribute : a graph-level attribute.
Multi-graph : there can be more than one edge between vertices, including self-edges.

Figure 2 shows a variety of different types of graphs corresponding to real data that we may be interested in modeling, including physical systems, molecules, images, and text.

Алгоритм

Algorithm 1 Steps of computation in a full GN block.

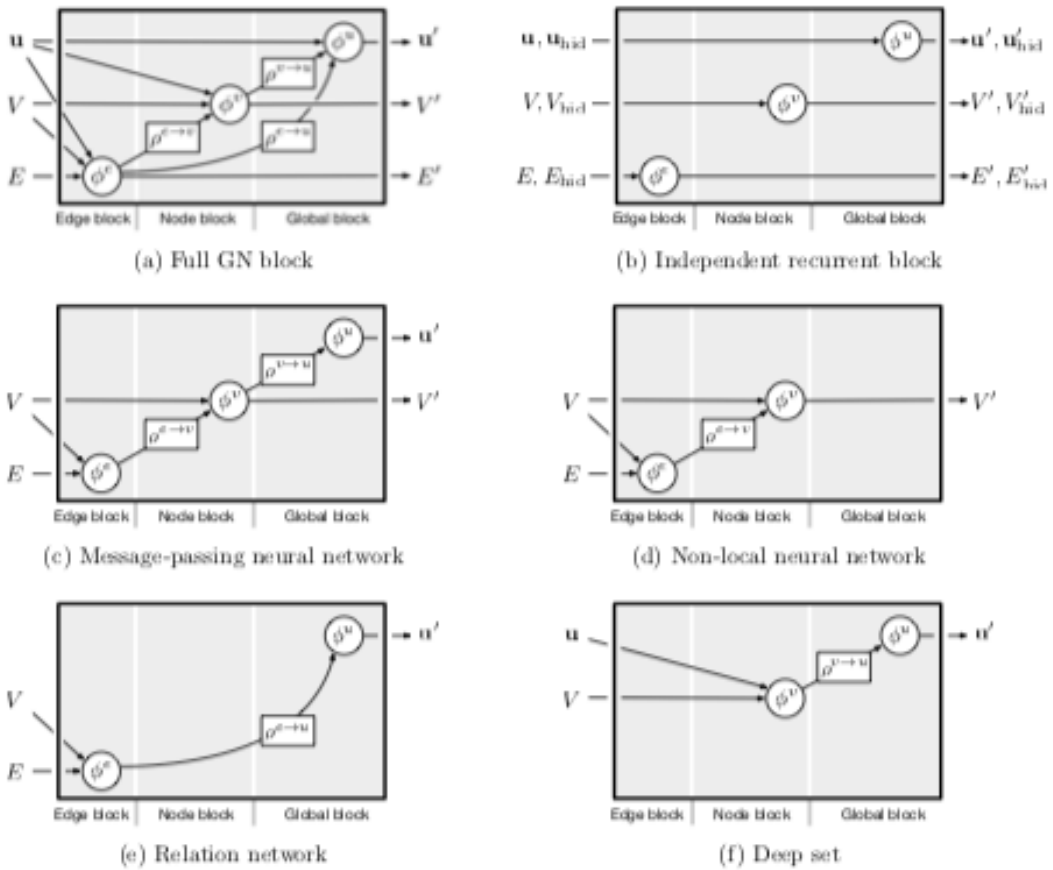
function GRAPHNETWORK(E, V, $\mathbf{u}$)
  for $k \in \{1 \dots N^e\}$ **do**
    $\mathbf{e}'_k \leftarrow \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u})$ ▷ 1. Compute updated edge attributes
  end for
  for $i \in \{1 \dots N^n\}$ **do**
    let $E'_i = \{(\mathbf{e}'_k, r_k,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

Схема выполнения

(a) Edge update (b) Node update (c) Global update

Figure 3: Updates in a GN block. Blue indicates the element that is being updated, and black indicates other elements which are involved in the update (note that the pre-update value of the blue element is also used in the update). See Equation 1 for details on the notation.

Примеры архитектур



(a) Full GN block

(b) Independent recurrent block

(c) Message-passing neural network

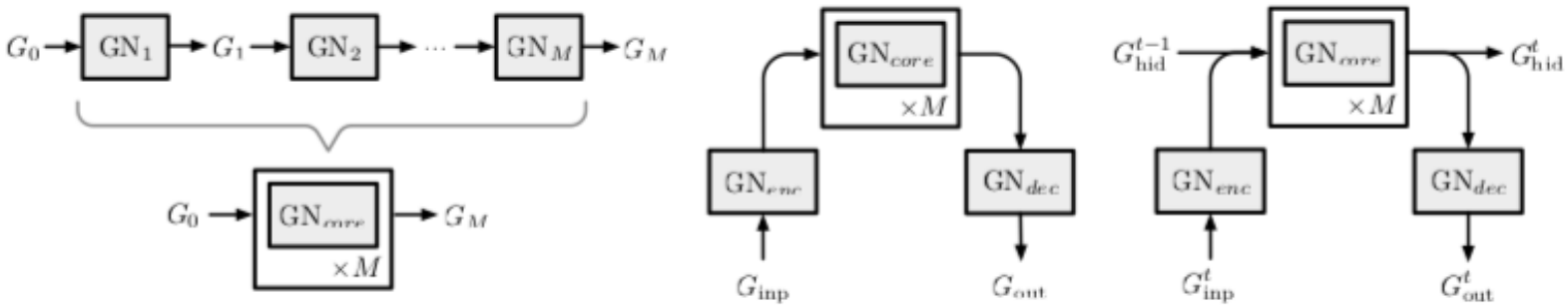
(d) Non-local neural network

(e) Relation network

(f) Deep set

Figure 4: Different internal GN block configurations. See Section 3.2 for details on the notation, and Section 4 for details about each variant. (a) A full GN predicts node, edge, and global output attributes based on incoming node, edge, and global attributes. (b) An independent, recurrent update block takes input and hidden graphs, and the ϕ functions are RNNs (Sanchez-Gonzalez et al., 2018). (c) An MPNN (Gilmer et al., 2017) predicts node, edge, and global output attributes based on incoming node, edge, and global attributes. Note that the global prediction does not include aggregated edges. (d) A NLNN (Wang et al., 2018c) only predicts node output attributes. (e) A relation network (Raposo et al., 2017; Santoro et al., 2017) only uses the edge predictions to predict global attributes. (f) A Deep Set (Zaheer et al., 2017) bypasses the edge update and predicts updated global attributes.

Композиция блоков



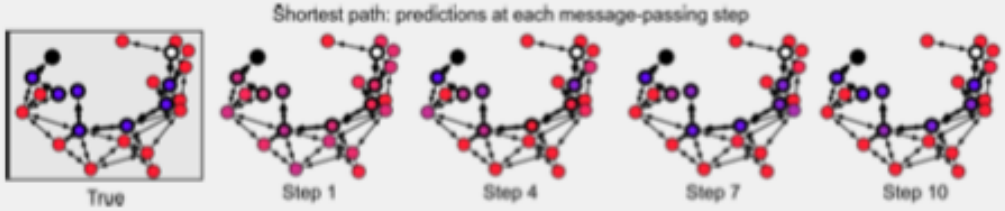
(a) Composition of GN blocks (b) Encode-process-decode (c) Recurrent GN architecture

Figure 6: (a) An example composing multiple GN blocks in sequence to form a GN "core". Here, the GN blocks can use shared weights, or they could be independent. (b) The *encode-process-decode* architecture, which is a common choice for composing GN blocks (see Section 4.3). Here, a GN encodes an input graph, which is then processed by a GN core. The output of the core is decoded by a third GN block into an output graph, whose nodes, edges, and/or global attributes would be used for task-specific purposes. (c) The encode-process-decode architecture applied in a sequential setting in which the core is also unrolled over time (potentially using a GRU or LSTM architecture), in addition to being repeated within each time step. Here, merged lines indicate concatenation, and split lines indicate copying.

Композиция блоков

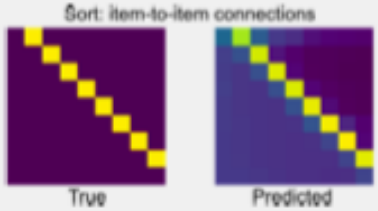
Shortest path demo: tinyurl.com/gn-shortest-path-demo

This demo creates random graphs, and trains a GN to label the nodes and edges on the shortest path between any two nodes. Over a sequence of message-passing steps (as depicted by each step's plot), the model refines its prediction of the shortest path.



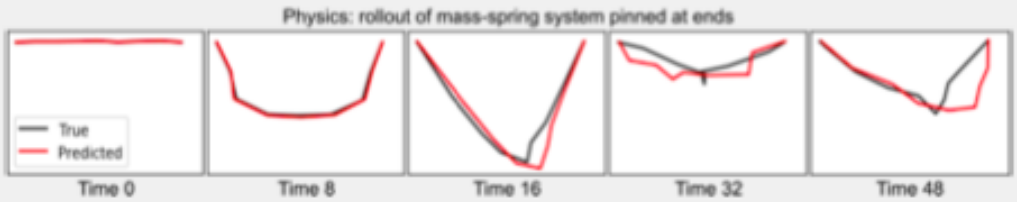
Sort demo: tinyurl.com/gn-sort-demo

This demo creates lists of random numbers, and trains a GN to sort the list. After a sequence of message-passing steps, the model makes an accurate prediction of which elements (columns in the figure) come next after each other (rows).



Physics demo: tinyurl.com/gn-physics-demo

This demo creates random mass-spring physical systems, and trains a GN to predict the state of the system on the next timestep. The model's next-step predictions can be fed back in as input to create a rollout of a future trajectory. Each subplot below shows the true and predicted mass-spring system states over 50 timesteps. This is similar to the model and experiments in (Battaglia et al., 2016)'s "interaction networks".



Заключение

Открытые вопросы

- Проблема изоморфных графов
- Проблема оптимизации и прореживания
- Поиск других комбинаторных структур

Выводы

Предложенный метод существенно обобществляет существующие архитектуры, но это является лишь первым шагом. Обобщения комбинаторных структур продвинут нас в deep learning, meta learning и AI в целом.