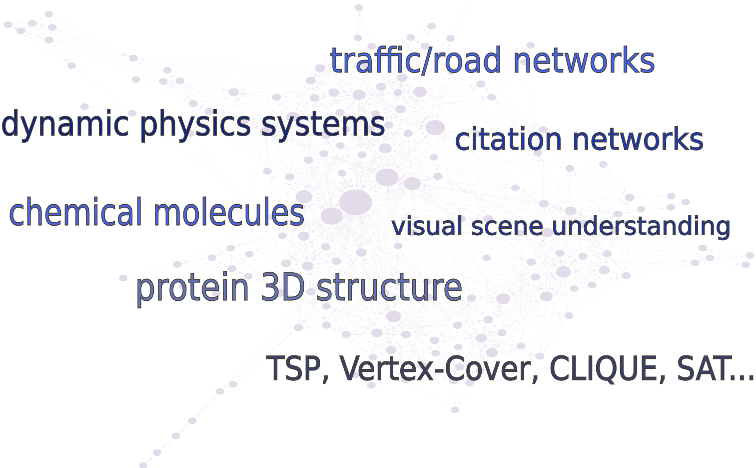
Graph Neural Networks

Felix Becker

University of Greifswald

June 14, 2021

Graphs are everywhere



traffic/road networks

dynamic physics systems

citation networks

chemical molecules

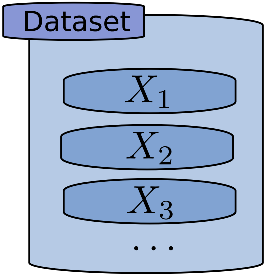
visual scene understanding

protein 3D structure

TSP, Vertex-Cover, CLIQUE, SAT...

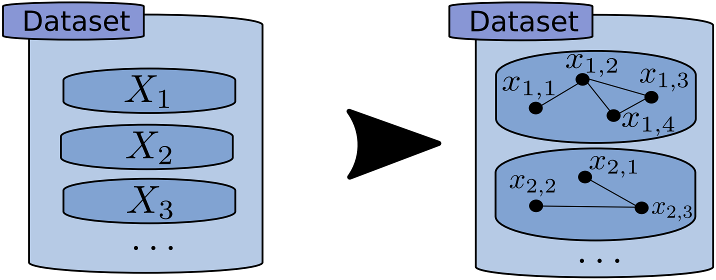
Inductive bias

- Often, data does not come as individual objects...



Inductive bias

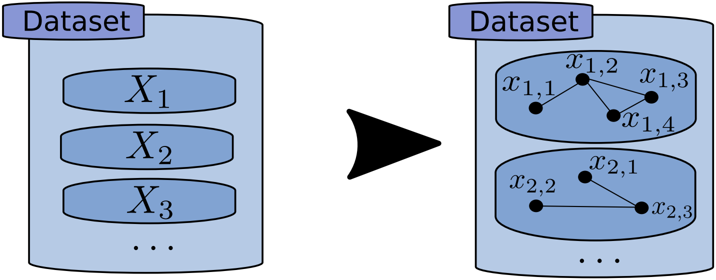
- Often, data does not come as individual objects...



- ... but rather in sets of objects and rules on how they interact.

Inductive bias

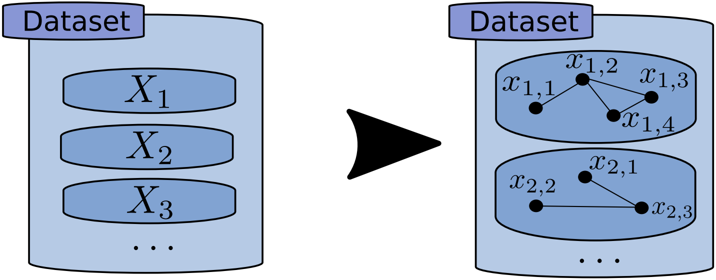
- Often, data does not come as individual objects...



- ... but rather in sets of objects and rules on how they interact.
- Inductive bias: constraints imposed on the set of possible pairwise interactions (represented as a graph).

Inductive bias

- Often, data does not come as individual objects...



- ... but rather in sets of objects and rules on how they interact.
- Inductive bias: constraints imposed on the set of possible pairwise interactions (represented as a graph).
- Making predictions requires 'relational reasoning' based on the graph structure.

Inductive bias

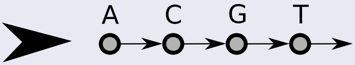
Inductive biases can be well defined independent of the data examples:

Inductive bias

Inductive biases can be well defined independent of the data examples:

Sequences

>1j46_A
ACGTAAGTGTAAG (...)

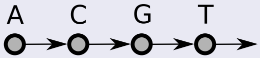


Inductive bias

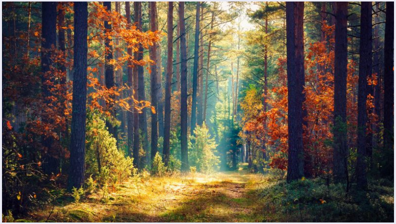
Inductive biases can be well defined independent of the data examples:

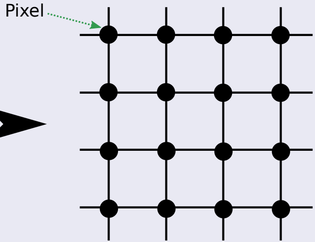
Sequences

>1j46_A
ACGTAAGTGTAAG (...)



Images





Inductive bias

- We have dedicated models for some cases:

Inductive bias

- We have dedicated models for some cases:
  - Recurrent architectures for sequences.

Inductive bias

- We have dedicated models for some cases:
  - Recurrent architectures for sequences.
  - Convolutional neural networks for images.

Inductive bias

- We have dedicated models for some cases:
  - Recurrent architectures for sequences.
  - Convolutional neural networks for images.
- But how to handle data with less well defined inductive biases? (e.g. chemical molecules, road networks, citation networks...)

Graph Neural Networks (GNNs)

Definition: Feature graph

A (directed) feature graph is a 3-tuple $G = (u, V, E)$ with a global attribute u, nodes $V = \{ v_i \}_{i=1,\dots,n}$ where v_i are the attributes of the node at index i and edges $E = \{ (e_j, s_j, r_j) \}_{j=1,\dots,m}$ with edge attributes e_j, a sender node index s_j and a receiver node index r_j.

Graph Neural Networks (GNNs)

Definition: Feature graph

A (directed) feature graph is a 3-tuple $G = (u, V, E)$ with a global attribute u, nodes $V = \{ v_i \}_{i=1,\dots,n}$ where v_i are the attributes of the node at index i and edges $E = \{ (e_j, s_j, r_j) \}_{j=1,\dots,m}$ with edge attributes e_j, a sender node index s_j and a receiver node index r_j.

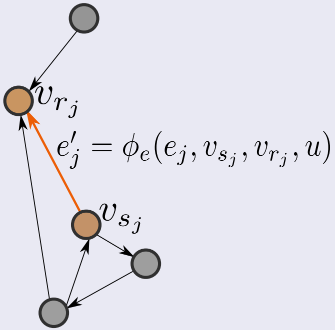
Definition: Graph neural network

A graph neural network (GNN) is a mapping $\omega : G \mapsto G'$ that maps a feature graph $G = (u, V, E)$ to another feature graph $G' = (u', V', E')$ with $V' = \{ v_i' \}_{i=1,\dots,n}$ and $E' = \{ (e_j', s_j, r_j) \}_{j=1,\dots,m}$.

An implementation of ω

Let ϕ_v, ϕ_e, ϕ_u be learnable non-linear functions (e.g. multilayer perceptrons).

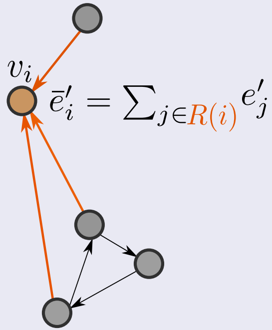
1. Update all edges



An implementation of ω

Let ϕ_v, ϕ_e, ϕ_u be learnable non-linear functions (e.g. multilayer perceptrons).

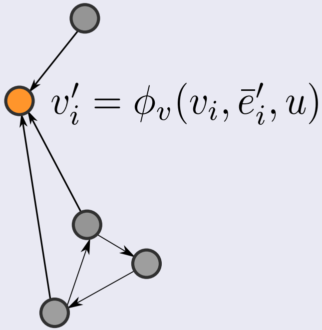
1. Update all edges
2. Aggregate neighborhoods



An implementation of ω

Let ϕ_v, ϕ_e, ϕ_u be learnable non-linear functions (e.g. multilayer perceptrons).

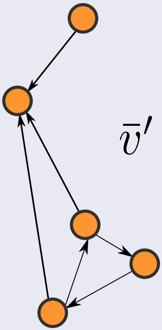
1. Update all edges
2. Aggregate neighborhoods
3. Update all nodes



An implementation of ω

Let ϕ_v, ϕ_e, ϕ_u be learnable non-linear functions (e.g. multilayer perceptrons).

1. Update all edges
2. Aggregate neighborhoods
3. Update all nodes
4. Aggregate nodes

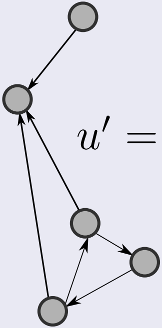


$$\bar{v}' = \sum_i v_i'$$

An implementation of ω

Let ϕ_v, ϕ_e, ϕ_u be learnable non-linear functions (e.g. multilayer perceptrons).

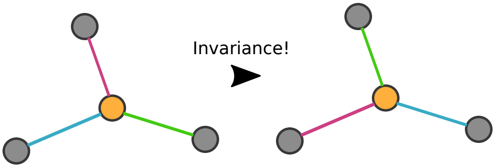
1. Update all edges
2. Aggregate neighborhoods
3. Update all nodes
4. Aggregate nodes
5. Update global attribute



$$u' = \phi_u(u, \bar{v}')$$

Remarks

A GNN is invariant to graph isomorphism, if the aggregation operations are symmetric functions (sum, average...)



Remarks

- A GNN is differentiable, if ϕ_v, ϕ_e, ϕ_u are (w.r.t. their weights θ).

Remarks

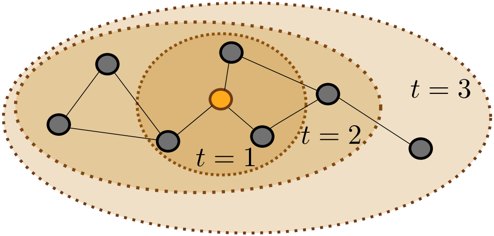
- A GNN is differentiable, if ϕ_v, ϕ_e, ϕ_u are (w.r.t. their weights θ).
- Therefore, we can backpropagate a loss signal in order to update θ.

Remarks

- A GNN is differentiable, if ϕ_v, ϕ_e, ϕ_u are (w.r.t. their weights θ).
- Therefore, we can backpropagate a loss signal in order to update θ.
- The loss may depend on u' (graph focused), V' (node focused) or E' (edge focused) or all of these.

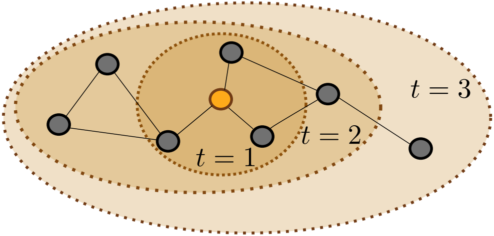
Message passing

Message passing: a composition of a GNN ω with itself for a fixed number of iterations: $\omega(\omega(\ldots\omega(G)))$



Message passing

Message passing: a composition of a GNN ω with itself for a fixed number of iterations: $\omega(\omega(\ldots\omega(G)))$



- If the global attribute u is excluded, the output of a node v after N iterations is conditioned on all nodes with a distance of at most N to v.

A brief history of GNNs

The Graph Neural Network Model
– Scarselli et al., 2009

Neural Message Passing
– Gilmer et al., 2017

Spectral Networks
– Bruna et al., 2014

Relational inductive biases, deep learning, and graph networks
– Battaglia et al., 2018

Graph Convolutional Networks
– Kipf et al., 2017

Representation learning

What do the learned representations v_i, e_j and u mean?

Representation learning

What do the learned representations v_i, e_j and u mean?
Short answer: We do not know.

Representation learning

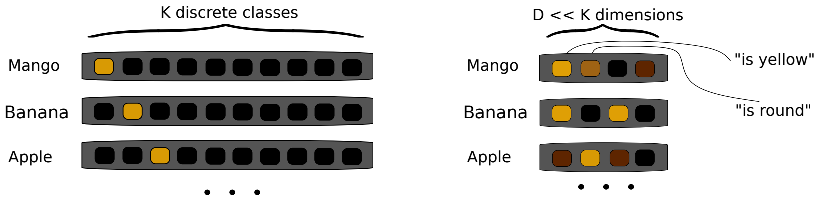
What do the learned representations v_i, e_j and u mean?

Short answer: We do not know.

But we can have an idea:

Representation learning

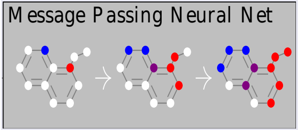
What do the learned representations v_i, e_j and u mean?
Short answer: We do not know.
But we can have an idea:



We use a D dimensional latent space where each neuron could represent a rather simple, independent property. Some of these properties, we can try to interpret as a human.
$\Longrightarrow$ Empirical: A shallow model on top (e.g. a linear combination) could make accurate predictions.

Some GNN applications

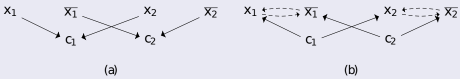
Predict properties of chemical molecules



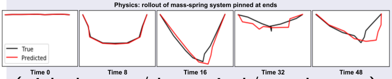
Neural Message Passing for Quantum Chemistry
– Gilmer et al., 2017

Solve SAT

Learning a SAT solver from single-bit supervision
– Selsam et al., 2019



Timesteps in a dynamic physics system



(github.com/deepmind/graph_nets)

Relational inductive biases, deep learning, and graph networks
– Battaglia et al., 2018

A practical example: Sorting numbers

Input

A list of pairwise distinct numbers $x_1, \ldots, x_k$

- WLOG $x_i \in [0, 1]$
- k is variable

Output

A sorted list of the same numbers

Let π be a sorting permutation of the indices, i.e. $\pi(i) = j \Leftrightarrow x_i$ is the jth smallest element

A naive approach without GNNs

We could try to solve the problem by using a simple neural network with an input vector of fixed length L:

$$(x_1, x_2, \dots, x_k, P, P, \dots, P)$$

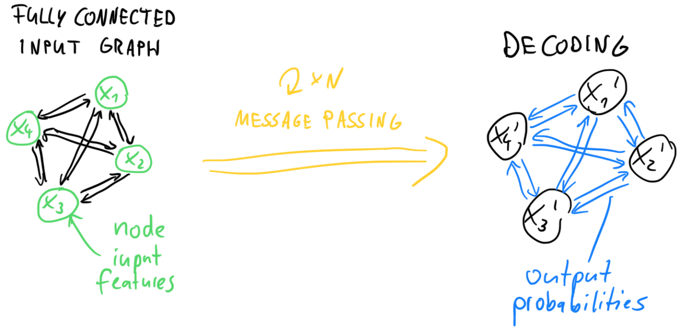
where L is large enough for all reasonable inputs and P is some *padding* added to input lists shorter than L.

Why **NOT** do it this way.

- maximum list length L is not intuitive
- overhead for $k << L$
- no generalization to longer lists than seen during training

Idea

1. Construct a fully connected graph with numbers as nodes
2. Refine nodes/edge embeddings with message passing
3. Decode $P(\pi(j) = \pi(i) + 1 | x_1, x_2, \ldots, x_k)$ for all $i, j = 1, \ldots, k$



Loss and Training

For details see notebook "Sort.ipynb".