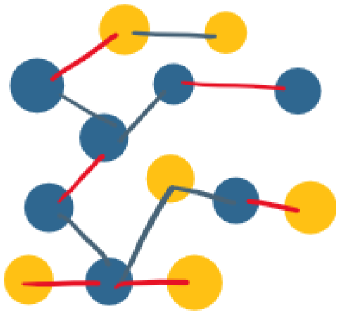
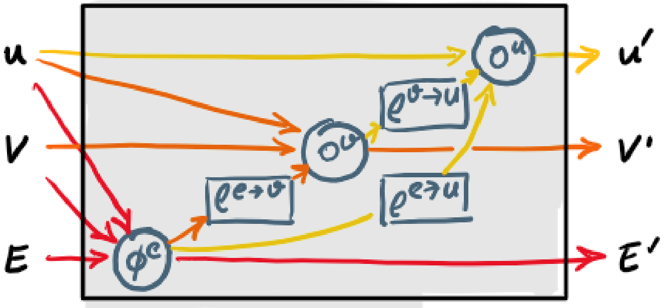
Karlsruhe Institute of Technology

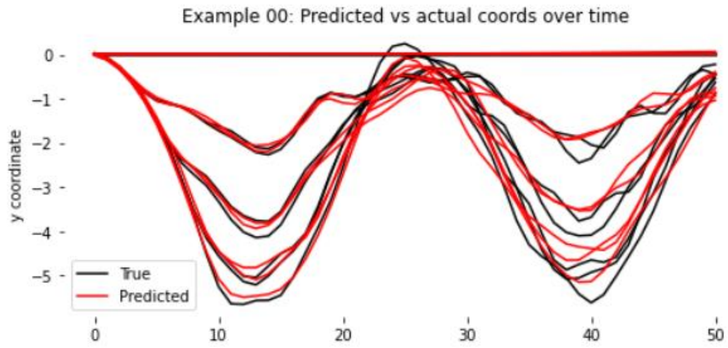
Data Driven Engineering II: Advaced Topics

Graph Neural Networks I

Institute of Thermal Turbomachinery
Prof. Dr.-Ing. Hans-Jörg Bauer



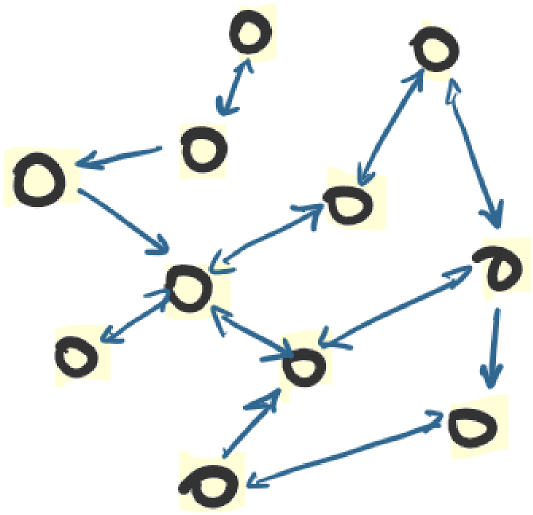




Graph Neural Networks :

1) GNN Basics
2) How GNN works
3) Basic architectures
4) Coding : Graph Nets library

Graph Neural Networks :



Graphs : a way to represent what we know about the system including the relationships btw. entities.

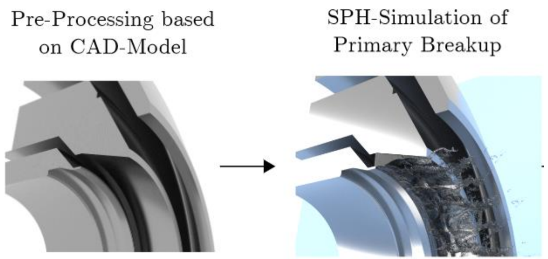
Q How can we exploit relational structure for a better prediction ?

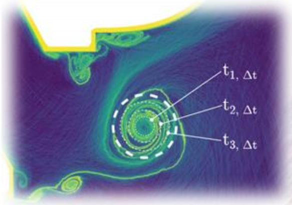
Graph Neural Networks :

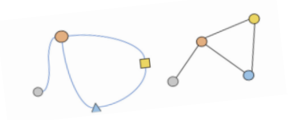
Graphs

* Many data systems are "graphs"

✓ Particle networks ✓ Social networks ✓ Mobility
✓ Disease modeling ✓ Recommend. systems ✓ IoT
✓ Multiphase flows ✓ Graph mining ✓ Codes
✓ Particle Physics ✓ Chemistry → Protein Folding
✓ Robotics → Fingerprint
→ Rxn Models
✓ Image & text analysis → Biomedical eng.







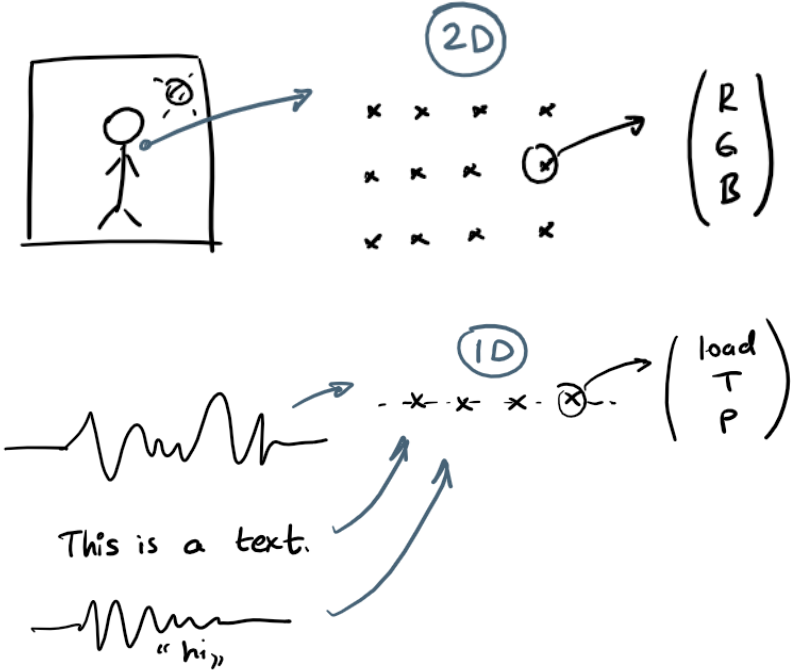
What we know already

* { CNN & RNN }

"regular & structured graphs"

~ deep learning ~

e.g.



Generalizing what we did...

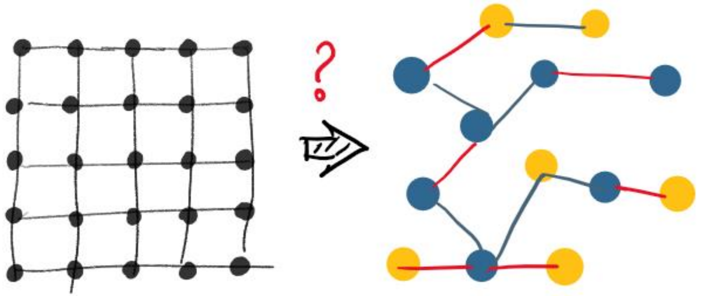
* GNN ⟵ motivated by CNN

local connections

shared weights

hierarchical patterns captured

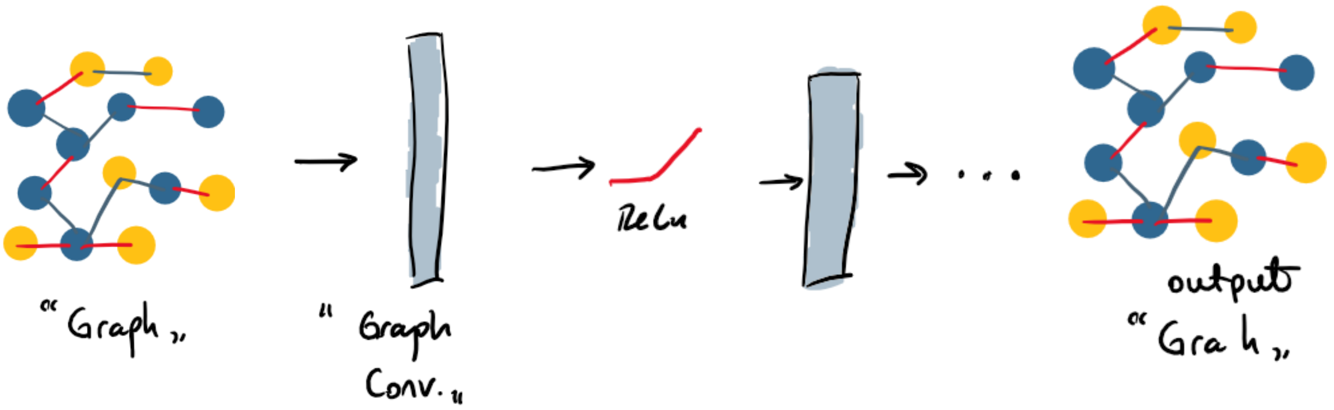
*



How to transform?

Graph Neural Networks :

* What is the aim ?

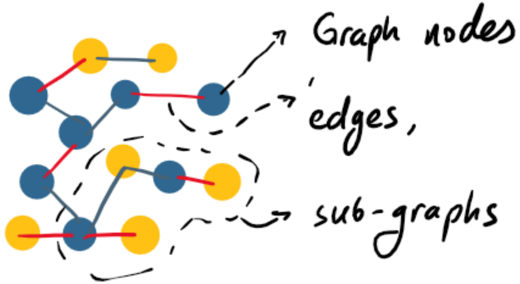


→ . . . → Nodes
→ Connections
→ Graph as a whole
→ a decision

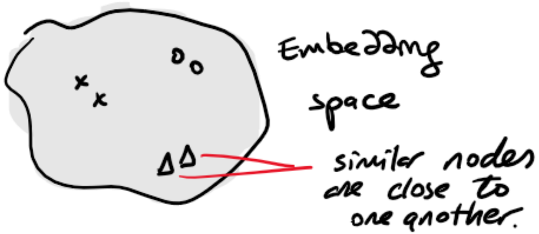
? how we can code it ?

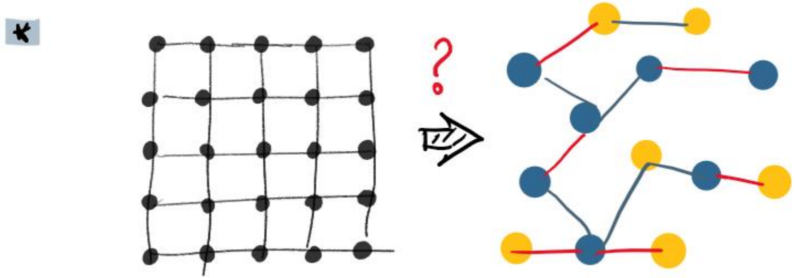
"Representation learning" } automate learning of features (embedding)

Embedding



Embedded on a vector :

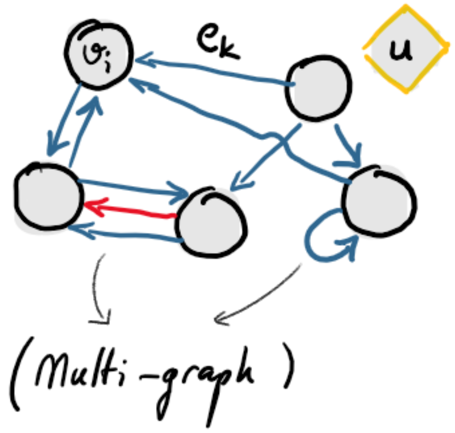


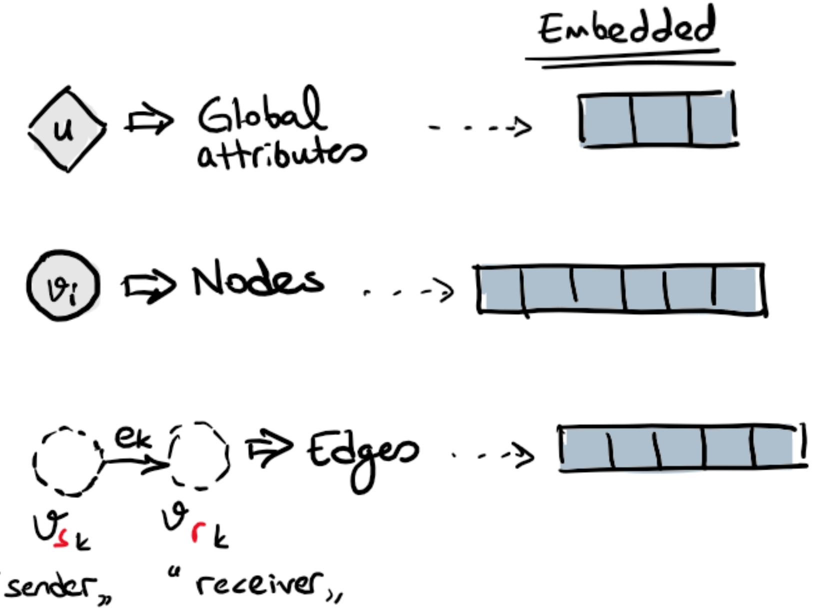


How to transform?

Understanding the Graph :

1





Understanding the Graph :

(2) Graph := 3-tuple ; $G = (u, V, E)$

- u is for the whole graph $\Rightarrow$ label, parameter ($\vec{g}$) ...

- $V = \{ v_i \}_{i=1,N^v}$

  $v_i \Rightarrow$ "particle i" $\Rightarrow [x, y, z, u, v, w, m]$

- $E = \{ e_k, r_k, s_k \}_{k=1,N^e}$

  $e_k \Rightarrow$ Edge attribute

  $r_k \Rightarrow$ receiver index

  $s_k \Rightarrow$ sender index

$e_k = [1, k]$

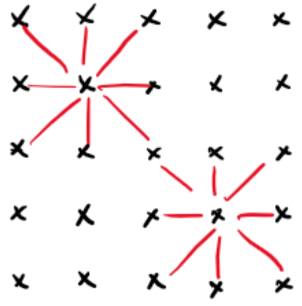
There is a spring

spring constant

Understanding Graph Network :

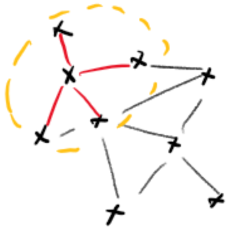
Convolution ⇒ "message passing" layers

node embeddings ⇒ info. about connections !
in a compressed format



Conv.

- Kernel is moved.
  ↓
  neigh. info.



graph conv.

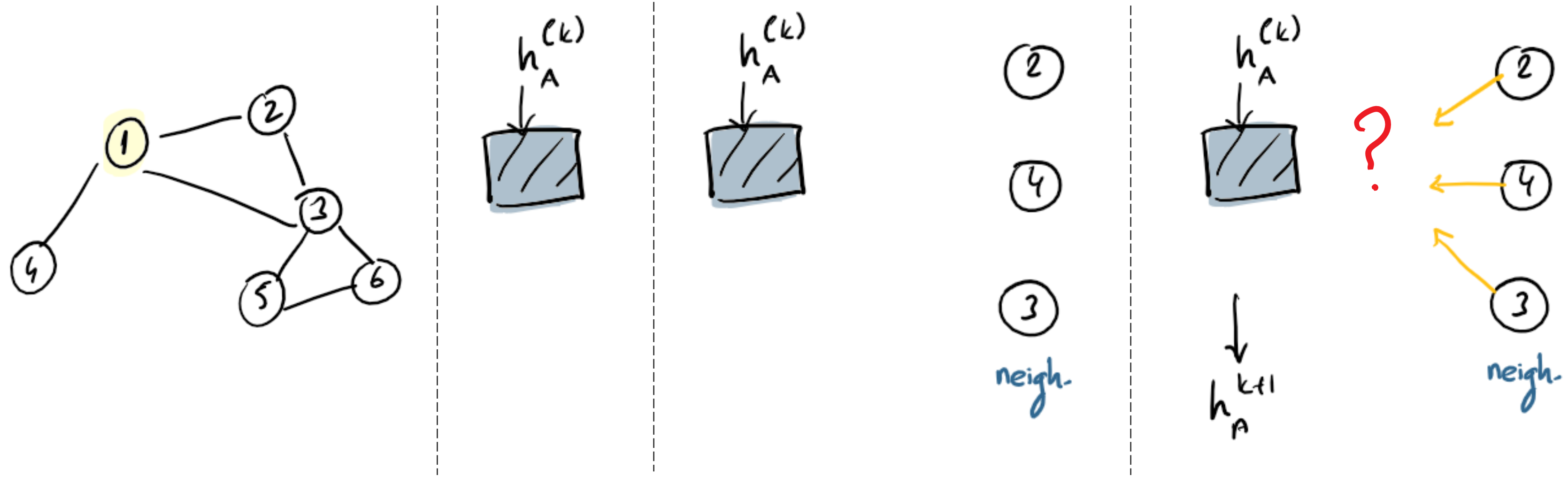
⇒ Connections are dynamic ↔ change

⇒ hidden state ↻ updated

hidden embedding

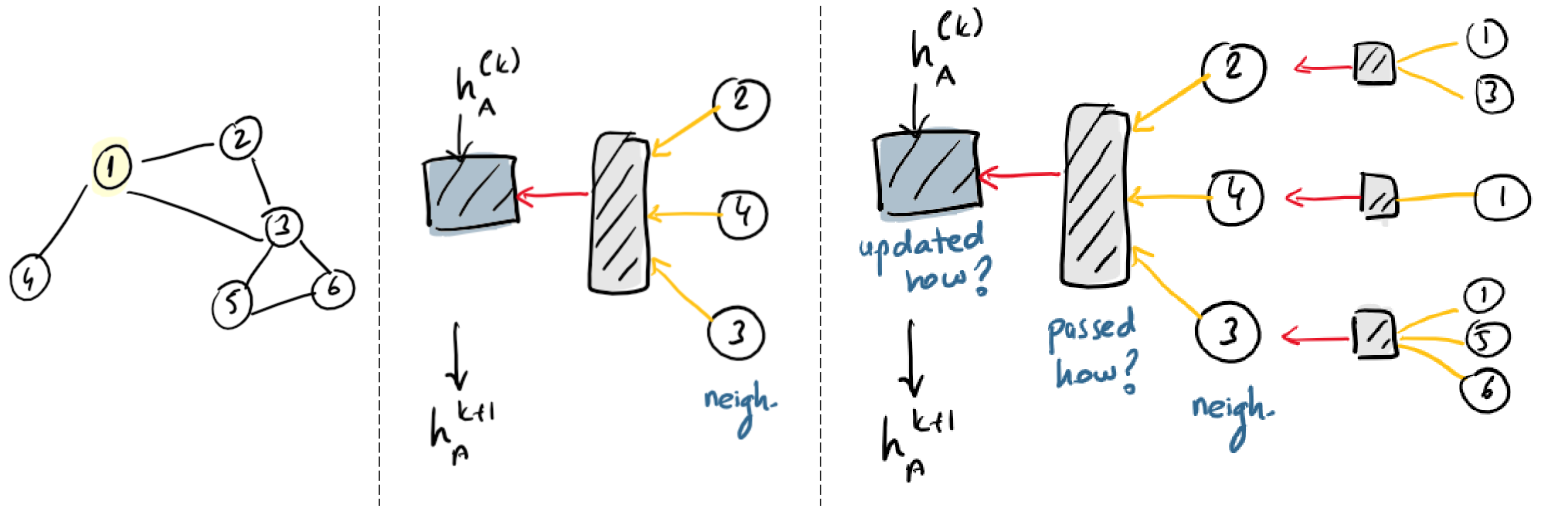
Understanding Graph Network :

idea ⇒ hidden states of nodes v_i updated (I) according to the info. passed (II) from neighbours (III)



Understanding Graph Network :

idea ⇒ hidden states of nodes v_i updated (I) according to the info. passed (II) from neighbours (III)



Understanding Graph Network :

idea ⇒ hidden states of nodes v_i updated (I) according to the info. passed (II) from neighbours (III)

$$* \quad h_i^{(k+1)} = \phi_{update}\left(h_A^{(k)}, \phi_{aggregate}^{(k)}\left(\{h_j^{(k)}, \forall k \in \mathcal{N}(i)\}\right)\right)$$

arbitrary differentiable functions

message from neighbour

Understanding Graph Network :

* Algorithm of a graph network

edges, nodes, global } (i) 3 update functions / (ii) 3 aggregate functions → edge, nodes, globals

1) Update edge attributes $e'_k = \phi^e(e_k, v_{r_k}, v_{s_k}, u)$

2) Aggregate edge att. per node $\bar{e}'_i = \rho^{e\to v}\{(e'_k, r_k, s_k)\}_{r_k=i,\, k=1:N^e}$

Understanding Graph Network :

* Algorithm of a graph network

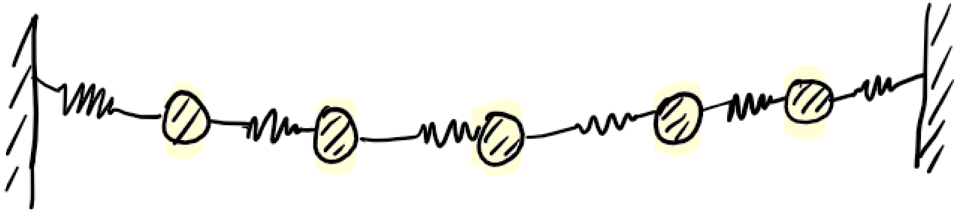
3) Update node attributes $v_i' = \phi\left(\bar{e}_i', v_i, u\right)$

4) Aggregate edge att. globally $\bar{e}' = \rho^{e\to u}\{(e_k', r_k, s_k)\}_{k=1:N^e}$

5) Aggregate node att. globally $\bar{v}' = \rho^{v\to u}\{(v_i')\}_{i=1,N^v}$

6) Update global attributes $u' = \phi^u\left(\bar{e}', \bar{v}', u\right)$

Eg



① Apply $\phi^e \rightarrow e'_k$ $\quad$ $e_k \Rightarrow$ forces btw. two connected balls.

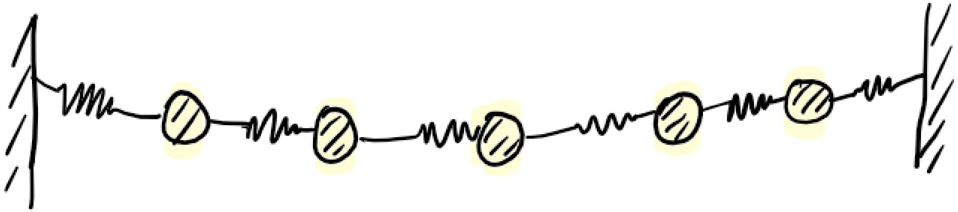
$\Downarrow$ get forces updated for each ball for each connection.

② $\rho^{e\rightarrow v} \Rightarrow \bar{e}'_i$ $\quad$ $\bar{e}_i \Rightarrow \sum$ force acting on i^{th} ball.

③ $\phi^v \Rightarrow v'_i$ $\quad$ $v_i \Rightarrow$ position, velocity, KE of ball i;

$\phi^v \hookrightarrow$ updates v_i as a func. $(\bar{e}'_i, v_i, u)$.

Eg

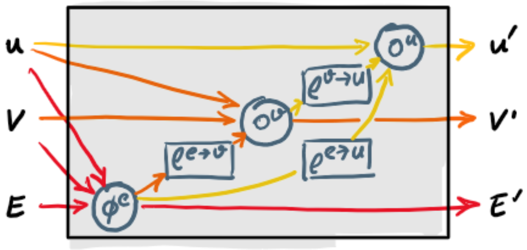


(4) $e^{e \to u} \Rightarrow \bar{e}'$ $\quad\quad$ $\bar{e} \Rightarrow \underbrace{\sum \text{forces}}_{=\emptyset}$ } Global info.

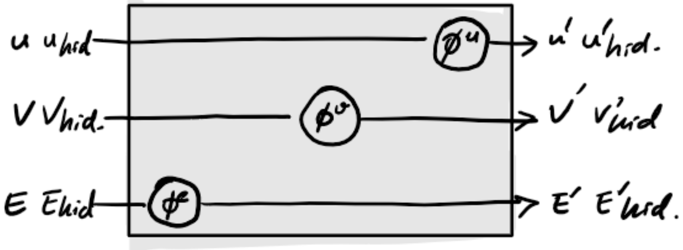
(5) $e^{v \to u} \Rightarrow \bar{v}'$ $\quad\quad$ $\bar{v} \Rightarrow \sum KE$ } Global info.

(6) $\phi^{u} \Rightarrow u'$ $\quad\quad$ $u' \Rightarrow \sum \text{energy}$ } Global info.

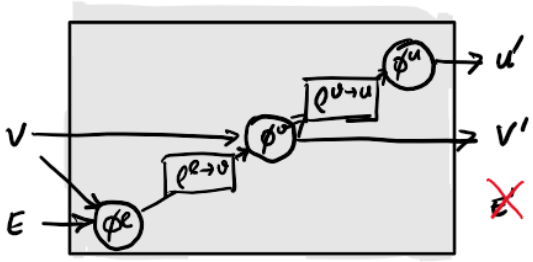
Graph Neural Networks



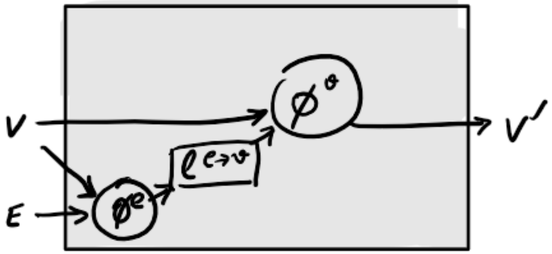
Full GN block



Independent recurrent block



Message-passing NN.



Non-local N.N.

GNN 101 : $\phi = ?\ ;\ \ell = ?$

$*$ $$h_i^{(k)} = \underbrace{\sigma}_{\text{ReLu, tanh, ...}}\Big(\underbrace{W_{self}^{(k)}}_{\text{Trainable}}\ \underbrace{h_i^{(k-1)}}_{\text{self state}} + \underbrace{W_{neigh.}^{(k)}}_{\text{Trainable}} \sum_{j\in\mathcal{N}} \underbrace{h_j^{(k-1)}}_{\text{neigh. steps}} + \underbrace{b^{(k)}}_{\text{bias term}}\Big)$$

Eg

$$G_0 \rightarrow G_1 \rightarrow G_2 \rightarrow G_3 \cdots \rightarrow G_T \Rightarrow h_i^T$$

Output; $\quad x_i = \sigma\left(w^T h_i^T + b\right)$

Loss; $\quad \mathcal{L} = \left. y_i \cdot \log x_i + (1-y_i)\log(1-x_i) \right\}$ binary cross entropy.

Improvements over basic GNN :

Karlsruhe Institute of Technology

Neighbourhood normalization

* Aggregation $\Rightarrow$ Σ operation } not very stable & sensitive to node degrees.

* Normalize the agg. operation by degree ;

$$\text{message} = m_{\mathcal{N}(i)} = \sum_{j \in \mathcal{N}(i)} h_j \Big/ |\mathcal{N}(i)|$$

* Symmetric normalization ;

$$m_{\mathcal{N}(i)} = \sum_{j \in \mathcal{N}(i)} \left(h_j \Big/ \sqrt{|\mathcal{N}(i)||\mathcal{N}(j)|} \right)$$

Graph Convolutional Networks

* GCN ⇒ Popular Baseline model

* GCN := Symm.-normalized aggregation + Self-loop update

$$h_i = \sigma\left(W \underbrace{\sum_{j \in \mathcal{N}(i) \cup \{i\}}} \frac{h_j}{\sqrt{|\mathcal{N}(i)|\,|\mathcal{N}(j)|}} \right)$$

agg. is taken over the neigh. & the node itself.

! No need to define update function

! Info. coming from the nodes // neighbors ?

Aggregation: A key step for success

* Σ maybe not the best option. → normalization
→ something better?

idea 1

$$m_{\mathcal{N}(i)} = MLP_\theta\left(\underbrace{\sum_{j\in\mathcal{N}(i)}}_{\text{min / max}} MLP_\phi(h_j)\right)$$

⇒ permutation invariant

idea 2

$$m_{\mathcal{N}(i)} = MLP_\theta\left(\frac{1}{|\Pi|}\sum_{\pi\in\Pi} LSTM\left(h_{j_1}, h_{j_2}, \dots h_{j_{\mathcal{N}(i)}}\right)_{\pi_i}\right)$$

permutation sensitive

set of permutations

Aggregation: A key step for success

idea 3

Not all neigh. are equally important $\Rightarrow$ Attention

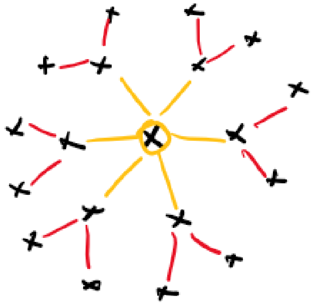
* GAT := Graph Attention Network $\Rightarrow$ $m_{\mathcal{N}(i)} = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} h_j$

* Attention models $\Rightarrow$ $\alpha_{ij} = \dfrac{\exp(h_i^T W h_j)}{\sum_{j' \in \mathcal{N}(i)} \exp(h_i^T W h_{j'})}$

Update Methods

* $$h_i^{(k+1)} = \phi_{update}\left(h_A^{(k)}, \phi_{aggregate}^{(k)}\left(\{h_j^{(k)}, \forall k \in \mathcal{N}(i)\}\right)\right)$$

Issue: Over-smoothing ⇨ node info. "washed out"



@ each rolling; h_i will changed by neigh.

& their neigh;

& their neigh;

...

Update Methods

Solution $\Rightarrow$ CNN $\rightarrow$ vector concatenations

$\rightarrow$ skip connections

(i) $\text{update}^{*}(h_i, m_{\mathcal{N}(i)}) = [\text{update}(h_i, m_{\mathcal{N}(i)}) \oplus h_i]$

$\rightarrow$ message from neighbours

$\rightarrow$ current representation of the node

(ii) $\text{update}^{**}(h_i\ m\ , \alpha) = \alpha\ \text{update}(h_i, m_{\mathcal{N}(i)}) + (1-\alpha)\, h_i$

$\Rightarrow$ linear interpolation $\Rightarrow$ learnable

Update Methods

Solution $\Rightarrow$ RNN $\rightarrow$ Gated information prop.

Agg. Function := Receive observation from neighbours &
update hidden states

$$h_i^k = \begin{matrix} GRU \\ LSTM \end{matrix} \left(h_i^{k-1}, m_{\mathcal{N}(i)}^k \right)$$

hidden state $\Rightarrow$ hidden embedding

observation x^t $\Rightarrow$ $m_{\mathcal{N}(i)}$ (agg. info. from neigh.)

Graph Pooling :

* So far ⇒ Node embeddings } Graph level information

$$z_G = \frac{\sum_{i \in V} h_i^T}{fn(|V|)}$$

last hidden emb.

normalization

sum ⇓ mean

* Use LSTM update + Attention mechanisms

* Using clustering on graph (~CNN) ↘ fclustering must be differentible

Karlsruhe Institute of Technology

colab