Graph Neural Networks

Algorithmic Alignment & Necessity
Halicin
MLP is all you need?

**Theorem 4.1.1 (universal approximation theorem):**
An arbitrary continuous function, defined on $[0,1]$ can be arbitrary well uniformly approximated by a multilayer feed-forward neural network with one hidden layer (that contains only finite number of neurons) using neurons with arbitrary activation functions in the hidden layer and a linear neuron in the output layer. Formally:

Let $\varphi(.)$ be the arbitrary activation function. Then $\forall f \in C([0,1]), \forall \varepsilon > 0: \exists n \in \mathbb{N}, w_i, a_i, b_i \in \mathbb{R}, i \in \{0..n\}$:

$$ (A_n f)(x) = \sum_{i=1}^{n} w_i \varphi(a_i x + b_i) $$

as an approximation of the function $f(.)$; that is

$$ \sup_{x \in [0,1]} |(A_n f)(x) - f(x)| < \varepsilon $$
MLP is all you need? No!

- Inductive bias for Images: **Convolutions**

- Inductive bias for Time Series: **Hidden states**

---

Battaglia et al., 2018
MLP is all you need? No!

- Prior distribution

\[ p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)} \]

- Ridge / Lasso regularization

\[ \mathcal{L} = \mathcal{L}(Y, f_\theta(X)) + \lambda \| \theta \|_p \]
Example: Particle Physics - Predict particle movement

\{ \vec{x}_1, \vec{v}_1, m_1 \} \quad \vec{F}_{1,2} = -\vec{F}_{2,1} \quad \{ \vec{x}_2, \vec{v}_2, m_2 \}

\{ \vec{x}_3, \vec{v}_3, m_3 \} \quad \vec{F}_{2,3} = -\vec{F}_{3,2} \quad \{ \vec{x}_2, \vec{v}_2, m_2 \}

Step 1: Compute forces
Step 2: Compute acceleration
Step 3: Evolve system

\vec{a}_i = \frac{1}{m_i} \vec{F}_{\text{net},i}

\vec{F}_{\text{net},i} = \sum_j \vec{F}_{i,j} = C \sum_j (1-r_{i,j}) \hat{r}_{i,j}

Cranmer et al., 2020
Example: Particle Physics – Predict particle movement

Step 1: Compute messages
Step 2: Pass messages around
Step 3: Update hidden rep.

\[ \tilde{h}_i^{(k)} = \phi^v \left( \tilde{h}_i^{(k)}, \sum_{j \in \mathcal{N}_i} \tilde{e}_{i,j}^{(k)} \right) \]

\[ \tilde{e}_{i,j}^{(k)} = \phi^e \left( \tilde{h}_i^{(k)}, \tilde{h}_j^{(k)} \right) \]
Example: Particle Physics – Predict particle movement

<table>
<thead>
<tr>
<th>Physics</th>
<th>GNNs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Particle</td>
<td>Node</td>
</tr>
<tr>
<td>${\vec{x}_1, \vec{v}_1, m_1}$</td>
<td>$\vec{h}_i^{(k)}$</td>
</tr>
<tr>
<td>Force</td>
<td>Edge / message</td>
</tr>
<tr>
<td>$\overrightarrow{F}_{i,j}$</td>
<td>$\vec{e}_{i,j}^{(k)}$</td>
</tr>
<tr>
<td>Gravitation</td>
<td>Edge model</td>
</tr>
<tr>
<td>$(1-r_{i,j})\vec{r}_{i,j}$</td>
<td>$\phi_c^{(k)}$</td>
</tr>
<tr>
<td>Net force</td>
<td>Aggregation</td>
</tr>
<tr>
<td>$\sum_{j} \overrightarrow{F}_{i,j}$</td>
<td>$\sum_{i,j} \phi_{i,j}^{(k)}$</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Node model</td>
</tr>
<tr>
<td>$\vec{a}_i$</td>
<td>$\phi_v^{(k)}$</td>
</tr>
</tbody>
</table>

Message passing framework: Algorithmic alignment with physical task

Cranmer et al., 2020
Example: Graph Algorithms

Bellman-Ford algorithm

```plaintext
for k = 1 ... |S| - 1:
    for u in S:
        d[k][u] = min_v d[k-1][v] + cost(v, u)
```

GNN

```plaintext
for k = 1 ... GNN iter:
    for u in S:
        h_u(k) = Σ_v MLP(h_v(k-1), h_u(k-1))
```

Xu et al., 2019
Example: Graph Algorithms

GNN

for \( k = 1 \) ... GNN iter:

for \( u \) in \( S \):

\[
\mathbf{h}_u^{(k)} = \Sigma_v \text{MLP}(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)})
\]

Xu et al., 2019
Can we mathematically define Algorithmic Alignment?

Definition 1.1 (PAC learning and sample complexity). Fix an error parameter $\epsilon > 0$ and failure probability $\delta \in (0,1)$. Suppose $\{x_i, y_i\}_{i=1}^M$ are i.i.d. samples from some distribution $\mathcal{D}$, and the data satisfies $y_i = g(x_i)$ for some underlying function $g$. Let $f = \mathcal{A}(\{x_i, y_i\}_{i=1}^M)$ be the function generated by a learning algorithm $\mathcal{A}$. Then $g$ is $(M, \epsilon, \delta)$-learnable with $\mathcal{A}$ if

$$\mathbb{P}_{x \sim \mathcal{D}} [\|f(x) - g(x)\| \leq \epsilon] \geq 1 - \delta.$$ 

The sample complexity $C_{\mathcal{A}}(g, \epsilon, \delta)$ is the minimum $M$ so that $g$ is $(M, \epsilon, \delta)$-learnable with $\mathcal{A}$.

Xu et al., 2019
Can we mathematically define Algorithmic Alignment?

**Definition 1.2 (Algorithmic alignment).** Let $g$ be a reasoning function and $\mathcal{N}$ a neural network with $n$ modules $\mathcal{N}_i$. The module functions $f_1, \ldots, f_n$ generate $g$ for $\mathcal{N}$ if, by replacing $\mathcal{N}_i$ with $f_i$, the network $\mathcal{N}$ simulates $g$. Suppose $\{x_i, y_i\}_{i=1}^M$ are i.i.d. samples from some distribution $\mathcal{D}$, and the data satisfies $y_i = g(x_i)$. Then $\mathcal{N}$ $(M, \epsilon, \delta)$-algorithmically aligns with $g$ if (1) $f_1, \ldots, f_n$ generate $g$ and (2) there are learning algorithms $A_i$ for the $\mathcal{N}_i$’s such that

$$n \cdot \max_i C_{A_i}(f_i, \epsilon, \delta) \leq M.$$
Can we mathematically define Algorithmic Alignment?

**Definition 1.2 (Algorithmic alignment).** Let $g$ be a reasoning function and $\mathcal{N}$ a neural network with $n$ modules $\mathcal{N}_i$. The module functions $f_1, ..., f_n$ generate $g$ for $\mathcal{N}$ if, by replacing $\mathcal{N}_i$ with $f_i$, the network $\mathcal{N}$ simulates $g$. Suppose $\{x_i, y_i\}_{i=1}^M$ are i.i.d. samples from some distribution $\mathcal{D}$, and the data satisfies $y_i = g(x_i)$. Then $\mathcal{N}$ $(M, \epsilon, \delta)$-algorithmically aligns with $g$ if (1) $f_1, ..., f_n$ generate $g$ and (2) there are learning algorithms $\mathcal{A}_i$ for the $\mathcal{N}_i$’s such that

$$n \cdot \max_i \mathcal{C}_{\mathcal{A}_i}(f_i, \epsilon, \delta) \leq M.$$
When can GNNs extrapolate?

Xu et al., 2020 & Trask et al., 2018
When can GNNs extrapolate?

**Definition 1.2 (Algorithmic alignment).** Let \( g \) be a reasoning function and \( \mathcal{N} \) a neural network with \( n \) modules \( \mathcal{N}_i \). The module functions \( f_1, ..., f_n \) generate \( g \) for \( \mathcal{N} \) if, by replacing \( \mathcal{N}_i \) with \( f_i \), the network \( \mathcal{N} \) simulates \( g \). Suppose \( \{x_i, y_i\}_{i=1}^M \) are i.i.d. samples from some distribution \( \mathcal{D} \), and the data satisfies \( y_i = g(x_i) \). Then \( \mathcal{N} \) \((M, \epsilon, \delta)\)-algorithmically aligns with \( g \) if (1) \( f_1, ..., f_n \) generate \( g \) and (2) there are learning algorithms \( A_i \) for the \( \mathcal{N}_i \)'s such that

\[
n \cdot \max_i C_{A_i}(f_i, \epsilon, \delta) \leq M.
\]

"easy to learn" = sample complexity grows polynomial = good interpolation

good extrapolation = algorithm steps can be represented by linear functions via MLP

Xu et al., 2020
When can GNNs extrapolate?

**good extrapolation** = algorithm steps can be represented by **linear** functions via MLP

**GNN Architectures**

- $h_u^{(k)} = \sum_{v} \text{MLP}^{(k)}(h_u^{(k-1)}, h_v^{(k-1)}, w(v, u))$
- **X** MLP has to learn non-linear steps
- $h_u^{(k)} = \min_v \text{MLP}^{(k)}(h_u^{(k-1)}, h_v^{(k-1)}, w(v, u))$
- **✓** MLP learns linear steps

**DP Algorithm (Target Function)**

- $d[k][u] = \min_v$
  
- $d[k-1][v] + w(v, u)$

**Xu et al., 2020**
What about termination?
What about termination?

1 hop
What about termination?

2 hops
What about termination?

3 hops
What about termination?

4 hops
What about termination?

**Idea:** Learn termination with IterGNN

![Flowchart](image-url)

Tang et al., 2020
What about termination?

**Idea:** Learn termination with IterGNN

---

**Algorithm 1:** Iterative module. $g$ is the stopping criterion and $f$ is the iteration body.

- **input:** initial feature $x$; stopping threshold $\epsilon$
- $k \leftarrow 1$
- $h^0 \leftarrow x$
- while $\prod_{i=1}^{k-1}(1 - c^i) > \epsilon$
do
- $h^k \leftarrow f(h^{k-1})$
- $c^k \leftarrow g(h^k)$
- $k \leftarrow k + 1$
end while

**return** $h = \sum_{j=1}^{k-1} \left( \prod_{i=1}^{j-1}(1 - c^i) \right) c^j h^j$

---

Tang et al., 2020
Static graph structure
Static graph structure
Static graph structure
How to overcome static graph structure?

**Idea:** Augment the graph with *dynamic* edges

1. encode entity representations
2. compute new hidden representations
3. decode answer
4. calc pointer mask
5. re-estimate pointer via self-attention

Veličković et al., 2020
How to overcome static graph structure?

Idea: Augment the graph with inferred edges

\[
\tilde{z}_i^{(t)} = f \left( e_i^{(t)}, \tilde{h}_i^{(t-1)} \right)
\]

\[
H^{(t)} = P \left( Z^{(t)}, \Pi^{(t-1)} \right)
\]

\[
\tilde{y}^{(t)} = g \left( \bigoplus_i \tilde{z}_i^{(t)} , \bigoplus_i \tilde{h}_i^{(t)} \right)
\]

\[
\mathbb{P} \left( \mu_i^{(t)} = 1 \right) = \psi \left( \tilde{z}_i^{(t)}, \tilde{h}_i^{(t)} \right)
\]

\[
\bar{q}_i^{(t)} = W_q \tilde{h}_i^{(t)}
\]

\[
\bar{k}_i^{(t)} = W_k \tilde{h}_i^{(t)}
\]

\[
\alpha_{ij}^{(t)} = \text{softmax}_j \left( \left< \bar{q}_i^{(t)}, \bar{k}_j^{(t)} \right> \right)
\]

\[
\tilde{\Pi}_{ij}^{(t)} = \mu_i^{(t)} \tilde{\Pi}_{ij}^{(t-1)} + \left( 1 - \mu_i^{(t)} \right) \mathbb{I}_{j = \text{argmax}_k \left( \alpha_{ik}^{(t)} \right)}
\]

\[
\Pi_{ij}^{(t)} = \tilde{\Pi}_{ij}^{(t)} \lor \tilde{\Pi}_{ji}^{(t)}
\]

Veličković et al., 2020
When *not* to use GNNs?
When *should* we use GNNs?

\[ S = \{ p \in P \mid M \text{ solves } p \} \]

“better than random guessing”

\[
\text{ForE} = \frac{|S(\text{Edges}) \cup S(\text{Features})|}{|P|}
\]

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Features</th>
<th>Edges</th>
<th>E(FandE)</th>
<th>FandE</th>
<th>ForE</th>
<th>GNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>0.586</td>
<td>0.346</td>
<td>0.203</td>
<td>0.192</td>
<td>0.74</td>
<td>0.828</td>
</tr>
<tr>
<td>Citeseer</td>
<td>0.544</td>
<td>0.412</td>
<td>0.224</td>
<td>0.235</td>
<td>0.721</td>
<td>0.699</td>
</tr>
<tr>
<td>Pubmed</td>
<td>0.693</td>
<td>0.407</td>
<td>0.282</td>
<td>0.246</td>
<td>0.854</td>
<td>0.779</td>
</tr>
<tr>
<td>AMZN-Photo</td>
<td>0.777</td>
<td>0.286</td>
<td>0.222</td>
<td>0.172</td>
<td>0.891</td>
<td>0.909</td>
</tr>
<tr>
<td>AMZN-Comp</td>
<td>0.652</td>
<td>0.391</td>
<td>0.255</td>
<td>0.235</td>
<td>0.808</td>
<td>0.809</td>
</tr>
<tr>
<td>MAG-Physics</td>
<td>0.915</td>
<td>0.507</td>
<td>0.464</td>
<td>0.475</td>
<td>0.947</td>
<td>0.949</td>
</tr>
<tr>
<td>MAG-CS</td>
<td>0.924</td>
<td>0.136</td>
<td>0.126</td>
<td>0.129</td>
<td>0.932</td>
<td>0.933</td>
</tr>
<tr>
<td>OGBN-Arxiv</td>
<td>0.658</td>
<td>0.411</td>
<td>0.271</td>
<td>0.281</td>
<td>0.788</td>
<td>0.726</td>
</tr>
<tr>
<td>Mutag</td>
<td>0.45</td>
<td>0.55</td>
<td>0.248</td>
<td>0.45</td>
<td>0.55</td>
<td>0.55</td>
</tr>
<tr>
<td>Enzymes</td>
<td>0.4</td>
<td>0.333</td>
<td>0.133</td>
<td>0.2</td>
<td>0.533</td>
<td>0.65</td>
</tr>
<tr>
<td>Proteins</td>
<td>0.607</td>
<td>0.643</td>
<td>0.39</td>
<td>0.607</td>
<td>0.643</td>
<td>0.616</td>
</tr>
<tr>
<td>IMDB-M</td>
<td>0.26</td>
<td>0.293</td>
<td>0.076</td>
<td>0.24</td>
<td>0.313</td>
<td>0.287</td>
</tr>
<tr>
<td>Reddit-B</td>
<td>0.76</td>
<td>0.775</td>
<td>0.589</td>
<td>0.76</td>
<td>0.775</td>
<td>0.77</td>
</tr>
</tbody>
</table>

Faber et al., 2021
When *should* we use GNNs?

\[
\frac{|S(\text{GNN}) \cap U|}{|U|} \quad U = P \setminus (S(\text{Features}) \cup S(\text{Edges}))
\]
Famous last words

architectural overfitting to characteristics of evaluation data

graph homophily

graph heterophily

Zhu et al., 2020
Thank you for your attention!
Looking forward to the discussion!
References

- Battaglia et al., “Relational inductive biases, deep learning and graph networks”, 2018
- Trask et al., “Neural Arithmetic Logic Units”, 2018
- Xu et al., “What can neural networks reason about?”, 2019
- Veličković et al., “Pointer Graph Networks”, 2020
- Cranmer et al., “Discovering symbolic models from deep learning with Inductive Biases”, 2020
- Veličković et al., “Neural execution of graph algorithms”, 2020
- Zhu et al., “Graph Neural Networks with Heterophily”, 2020
- Faber et al., “Should graph neural networks use features, edges, or both?”, 2021