Stanford CS224W: GNNs and Algorithmic Reasoning

CS224W: Machine Learning with Graphs

Joshua Robinson, Stanford University

http://cs224w.stanford.edu

Announcements

- Colab 5 due **EOD Tuesday**

Stanford CS224W: GNNs and Algorithmic Reasoning

CS224W: Machine Learning with Graphs

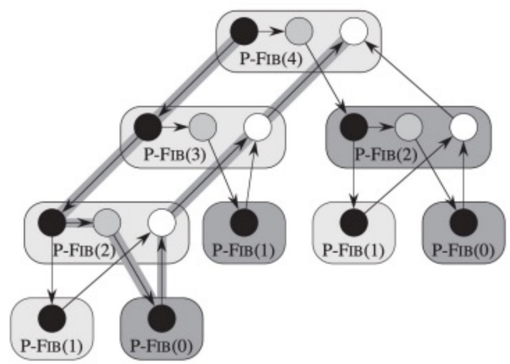
Joshua Robinson, Stanford University

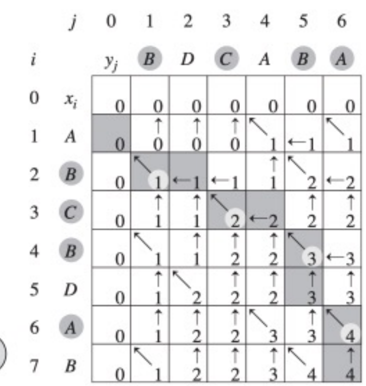
http://cs224w.stanford.edu

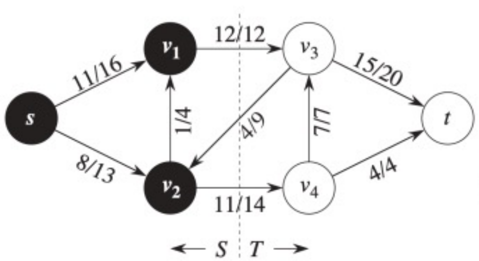
Graphs and Computer Science

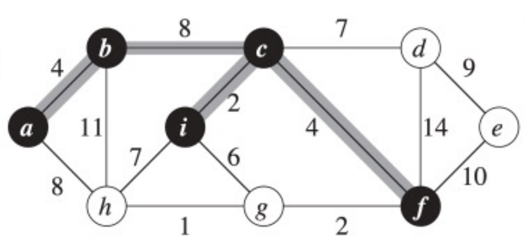
- 20th century saw unprecedented development of **algorithms**
  - **Sorting, shortest paths, graph search, routing**
  - **Algorithmic paradigms such as greedy, divide-and-conquer, parallelism, recursion, deterministic vs non-deterministic**

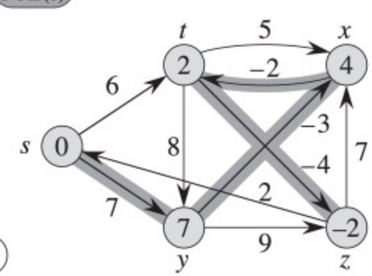
```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```









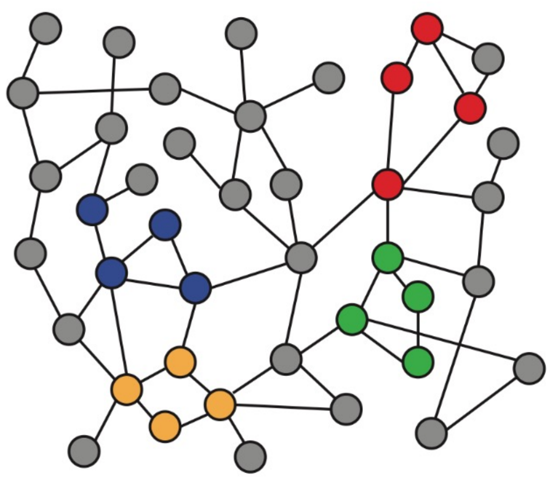
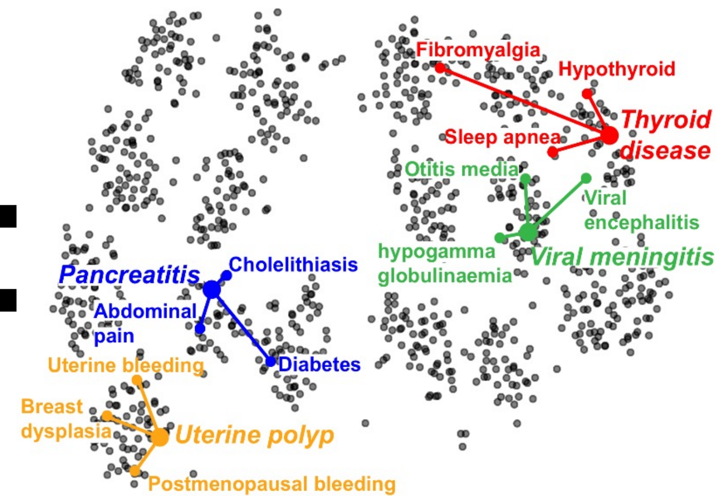


Graphs and Computer Science

- The study of algorithms and data structures are one of the **most coveted areas of computer science**
- All of computing is built on top of these fundamental algorithms
  - 100% including ML!
- But so far this class has (mostly) treated GNNs as a "new" type of graph algorithm

Graph Machine Learning

- This class:

f() = 

Input graph node embeddings

How to <u>learn</u> mapping function *f*?

Connection to classical graph algorithms unclear

Graph ML and Graph Algorithms

- So far treated GNNs as a “new” type of graph algorithm.
- **But in reality, graph ML has deep connections to the theory of computer science**
- **Today:**
  - **Ground development of GNNs in context of prior graph algorithms**
    - **Deep connections between “classical” algorithms and GNNs**
  - **Use to inform neural networks architecture design**

Plan for Today

- **Part 1**
  - An algorithm GNNs can run
- **Part 2**
  - Algorithmic structure of neural network architectures
- **Part 3**
  - What class of graph algorithms can GNNs simulate?
- **Part 4**
  - Algorithmic alignment: a principle for neural net design

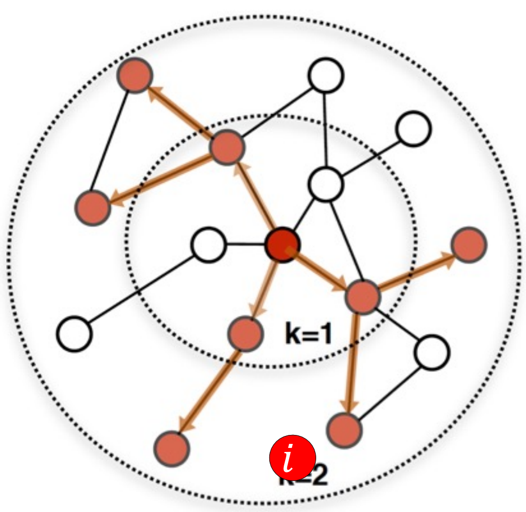
Other Reading

- **The work of Petar Veličković**
  - Lectures at Cambridge, expository papers, tutorials etc.
  - Some of today’s material drawn from Petar’s lectures

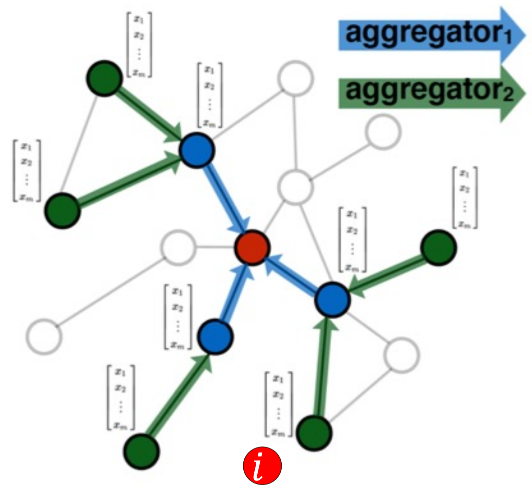
Stanford CS224W: GNNs and Classical Algorithms

CS224W: Machine Learning with Graphs
Joshua Robinson, Stanford University
http://cs224w.stanford.edu

Graph Neural Networks



Determine node computation graph



Propagate and transform information

- **GNNs defined by computation process**
- **I.e., how information is propagated across the graph to compute node embeddings**

GNNs as graph algorithms

- We define "message passing" a computational process
- Message passing defines a **class of algorithms on graphs**
- But it is not clear what algorithm(s)
- **A clue to get started:** we have already seen one algorithm GNNs can express…

GNNs can express 1-WL algorithm

- **GNNs can execute the 1-WL isomorphism test**
  - Recall lecture 6: GNNs at most as expressive as the 1-WL isomorphism test
  - GIN is exactly as expressive as 1-WL
  - Argument: show that GIN is a neural version of 1-WL

- Let's recall the test…

Stanford CS224W: GNNs and the Weisfeiler-Lehman Isomorphism Test

CS224W: Machine Learning with Graphs

Joshua Robinson, Stanford University

http://cs224w.stanford.edu

GNNs and the 1-WL isomorphism test

- Simple test for testing if two graphs are the same:
  - Assign each node a “color”
  - Randomly hash neighbor colors until stable coloring obtained
  - Read out the final color histogram
- Declare two graphs:
  - Non-isomorphic if final color histograms differ
  - Test inconclusive otherwise *(i.e., we do not know for sure that two graphs are isomorphic if the counts are the same)*


УДК 519.1

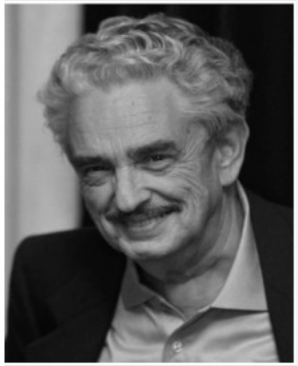
ПРИВЕДЕНИЕ ГРАФА К КАНОНИЧЕСКОМУ ВИДУ И ВОЗНИКАЮЩАЯ ПРИ ЭТОМ АЛГЕБРА

Б. Ю. ВЕЙСФЕЙЛЕР, А. А. ЛЕМАН

Рассматривается алгоритм приведения заданного конечного мультиграфа Г к каноническому виду. В процессе такого приведения возникает новый инвариант графа — алгебра 𝔄 (Г). Изучение свойств алгебры 𝔄 (Г) оказывается полезным при решении некоторых задач теории графов.

Выдвигаются и обсуждаются некоторые предположения относительно связи между свойствами алгебры 𝔄 (Г) и группой автоморфизмов графа Aut (Г). Построен пример неориентированного графа Г, алгебра 𝔄 (Г) которого совпадает с групповой алгеброй некоторой некоммутативной группы.

An algorithm is considered, reducing the specified finite multigraph Г to canonical form. In the course of this reduction, a new invariant of the graph is generated — algebra 𝔄 (Г). Study of the properties of the algebra 𝔄 (Г) proves helpful in solving a number of graph-theoretic problems. Some propositions concerning the relationships between the properties of the algebra 𝔄 (Г) and the graph's automorphism group Aut (Г) are discussed. An example of non—oriented graph Г is constructed whose algebra 𝔄 (Г) coincides with the group algebra of a non—commutative group.



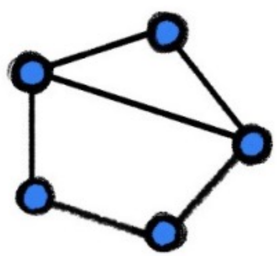
Lehman

Weisfeiler

GNNs and the 1-WL isomorphism test

- Running the test…

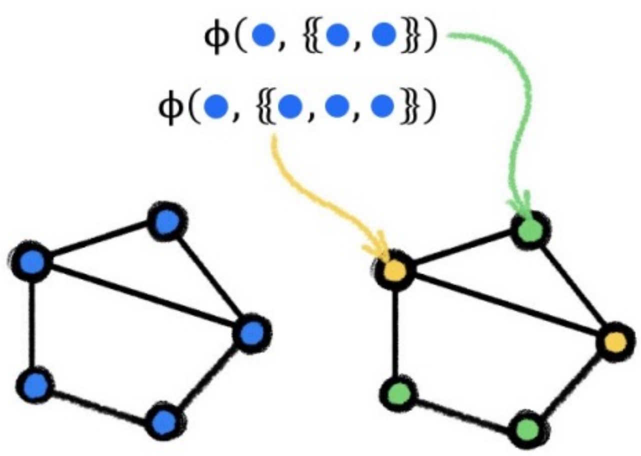
$\phi = HASH$ function
(i.e., injective function)



(diagrams thanks to Petar Veličković)

GNNs and the 1-WL isomorphism test

- Running the test…



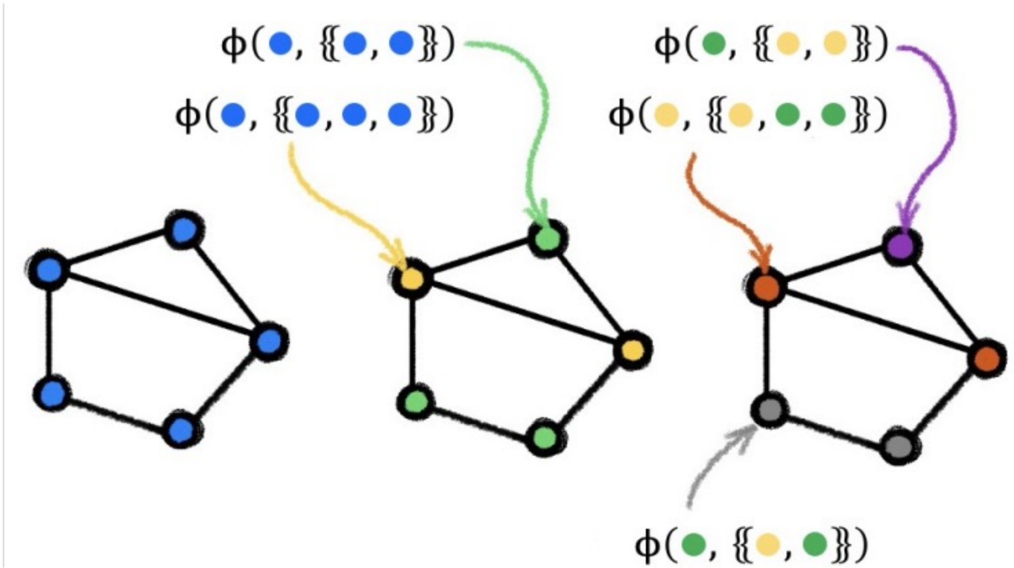
$\phi = HASH$ function
(i.e., injective function)

(diagrams thanks to Petar Veličković)

GNNs and the 1-WL isomorphism test

- Running the test…

$\phi = HASH$ function
(i.e., injective function)

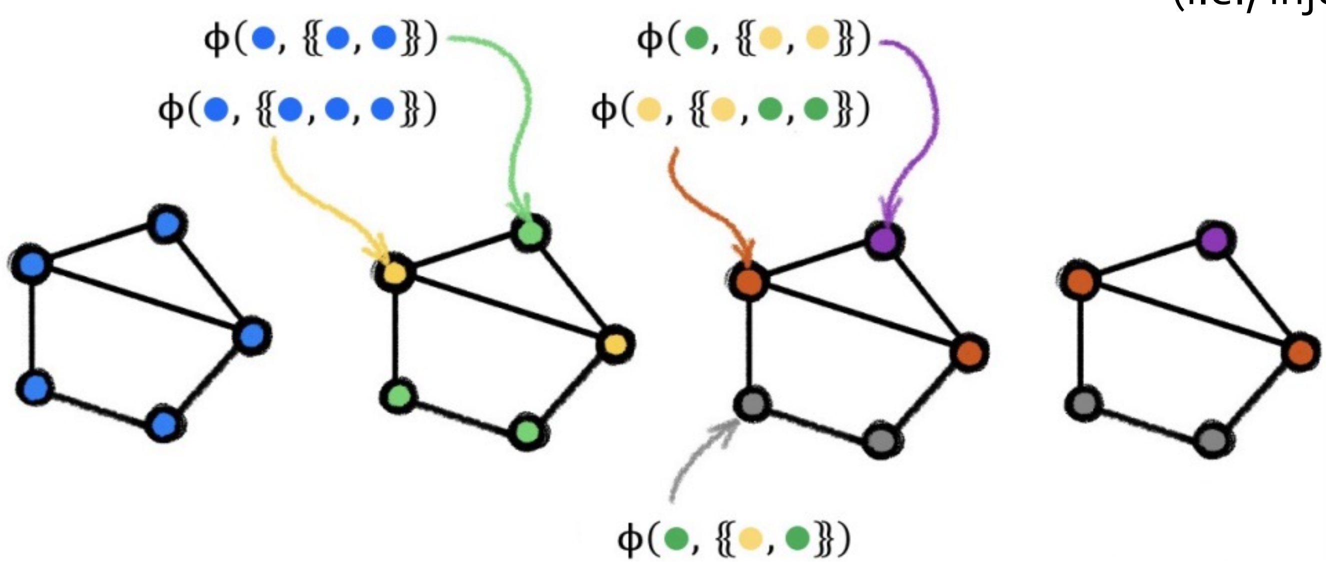


(diagrams thanks to Petar Veličković)

GNNs and the 1-WL isomorphism test

- Running the test…

$\phi = HASH$ function
(i.e., injective function)

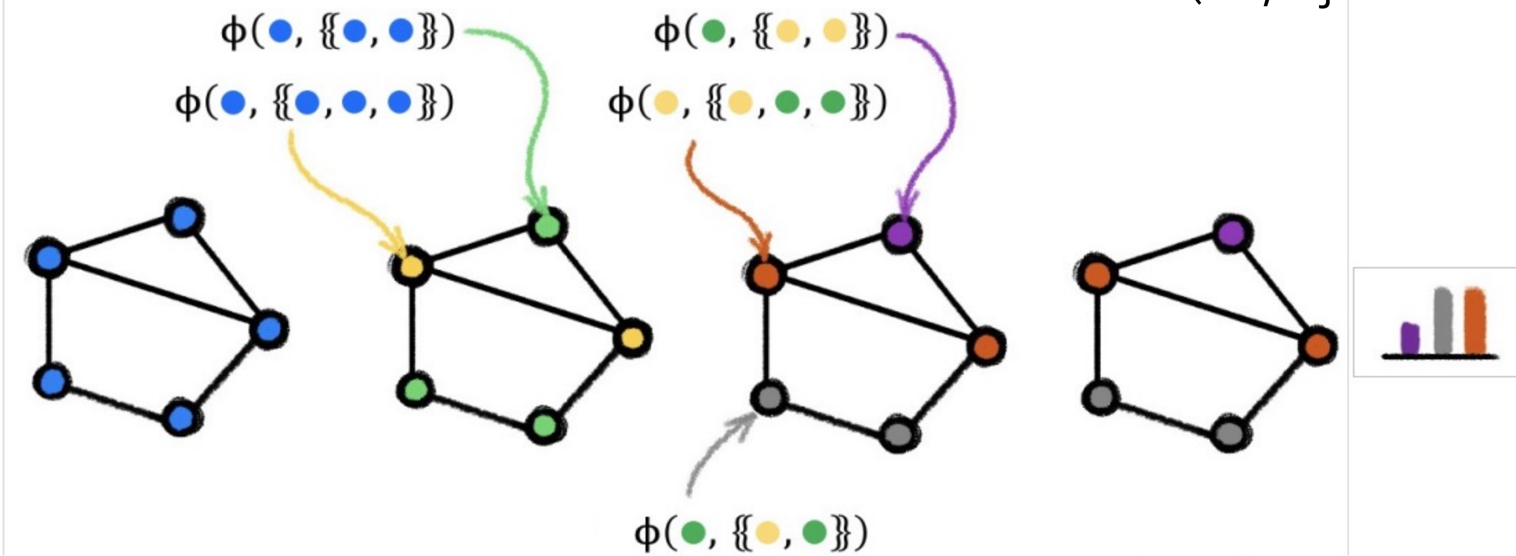


(diagrams thanks to Petar Veličković)

GNNs and the 1-WL isomorphism test

- Running the test…

$\phi = HASH$ function
(i.e., injective function)

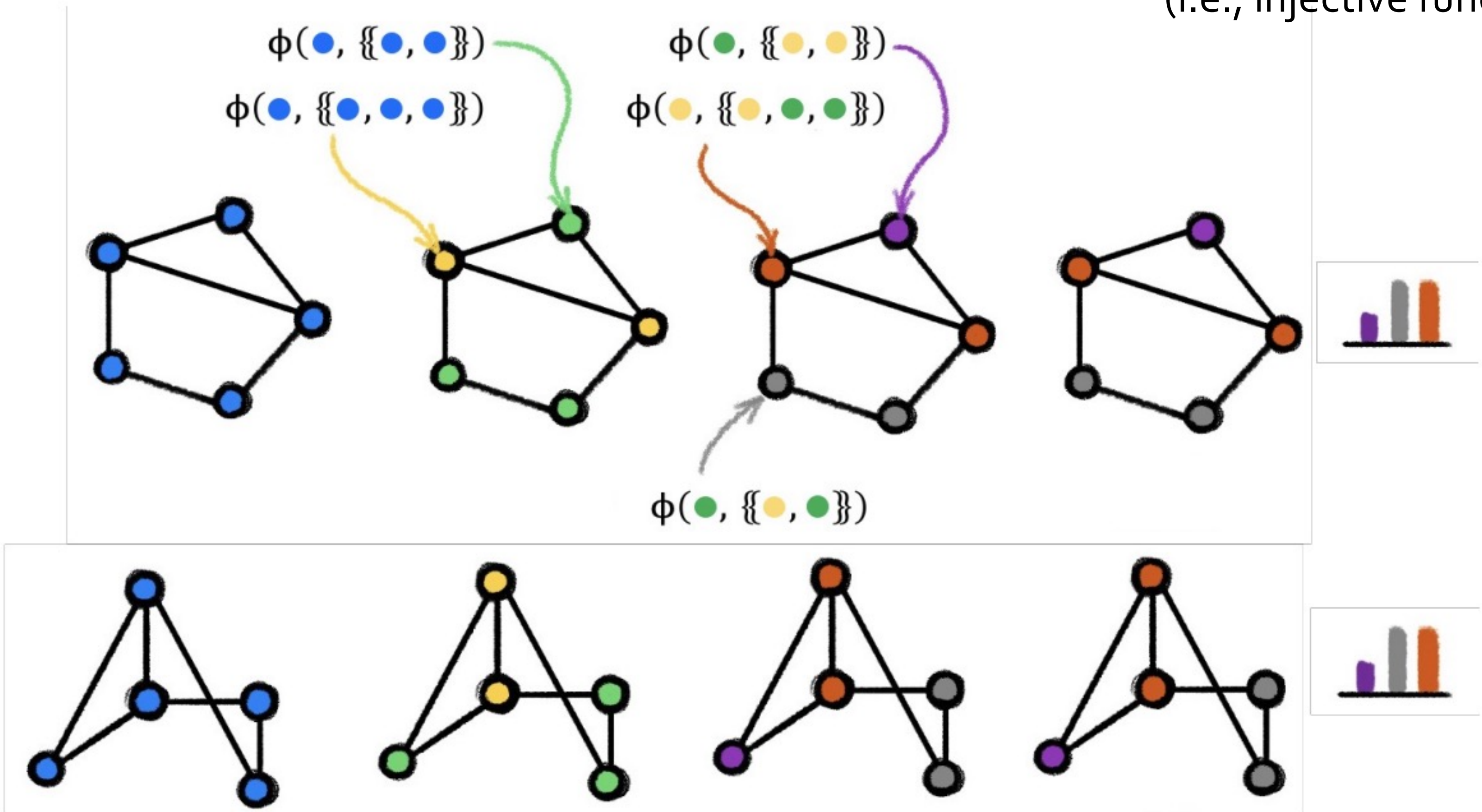


(diagrams thanks to Petar Veličković)

GNNs and the 1-WL isomorphism test

- Running the test…

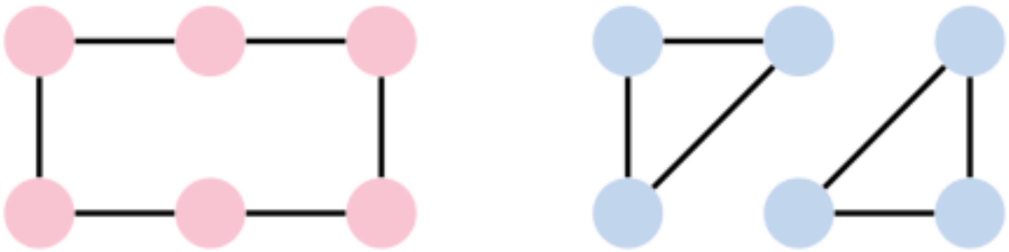
$\phi = HASH$ function
(i.e., injective function)



(diagrams thanks to Petar Veličković)

GNNs and the 1-WL isomorphism test

- Test does fail to distinguish some graphs, e.g.,



GNNs and the 1-WL isomorphism test

- We have seen GIN is as expressive as the 1-WL test
  - i.e., Given G_1, G_2, the following are equivalent:
    - there exist parameters s.t. $\mathrm{GIN}(G_1) \neq \mathrm{GIN}(G_2)$
    - 1-WL distinguishes G_1, G_2
- GIN is a "neural version" of the 1-WL algorithm
  - Replaces HASH function with learnable MLP

GNNs and the 1-WL isomorphism test

- We have seen GIN is as expressive as the 1-WL test
  - i.e., Given G_1, G_2, the following are equivalent:
    - there exist parameters s.t. $\mathrm{GIN}(G_1) \neq \mathrm{GIN}(G_2)$
    - 1-WL distinguishes G_1, G_2
- GIN is a "neural version" of the 1-WL algorithm
- **But this does not mean that 1-WL is the only graph algorithm GNNs can simulate**
- An **untrained GNN** (random MLP = random hash) is close to the 1-WL test

GNNs and the 1-WL isomorphism test

- We have seen GIN is as expressive as the 1-WL test
  - i.e., Given G_1, G_2, the following are equivalent:
    - there exist parameters s.t. $\mathrm{GIN}(G_1) \neq \mathrm{GIN}(G_2)$
    - 1-WL distinguishes G_1, G_2
- GIN is a "neural version" of the 1-WL algorithm
- **But this does not mean that 1-WL is the only graph algorithm GNNs can simulate**
- An **untrained GNN** (random MLP = random hash) is close to the 1-WL test
- **Today's question:** what other algorithms can (trained) GNNs simulate?

Plan for Today

- **Part 1**
  - An algorithm GNNs can run
- **Part 2**
  - Algorithmic structure of neural network architectures
- **Part 3**
  - What class of graph algorithms can GNNs simulate?
- **Part 4**
  - Algorithmic alignment: a principle for neural net design

Stanford CS224W: Algorithmic structure of neural networks

CS224W: Machine Learning with Graphs

Joshua Robinson, Stanford University

http://cs224w.stanford.edu

Neural Networks as Algorithms

- A neural network architecture defines a learnable computer program

- **Eventual Aim:** identify a broad class of "classical" (graph) algorithms that GNNs can **easily learn**
  - **This is different from our previous study of expressive power**

Neural Networks as Algorithms

- **Key perspective switch:**
  - In this lecture, we are **not focusing on expressive power** (as in lecture 6).
  - Instead we are focused on what tasks an architecture can **easily learn to solve**
    - For today: **easily = sample efficient** (not too much training data)
- **Key intuition:**
  - **MLPs easily learn smooth functions (e.g., linear, log, exp)**
  - **MLPs bad at learning complex function (e.g., sums of smooth functions - i.e., for-loops)**

Neural Networks as Algorithms

- **Approach:** **define progressively more complex algorithmic problems, and corresponding neural net architectures capable of solving each**

What Can Neural Networks Reason about? Xu et al. ICLR 2020

Neural Nets and Algorithm Structure

- **Problem 1 (feature extraction):**
  - **Input:** "flat" features $\mathbf{x} \in \mathbb{R}^n$ (e.g., color, size, position)
  - **Output:** scalar value $\boldsymbol{y}$ (e.g., is it round and yellow?)

Neural Nets and Algorithm Structure

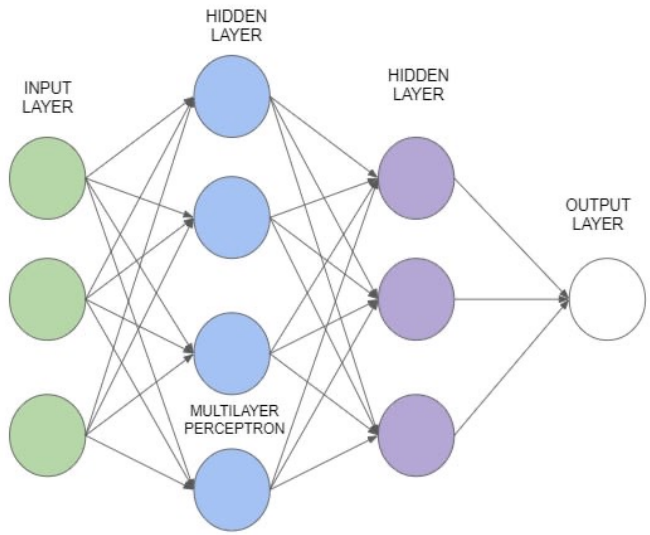
- **Problem 1 (feature extraction):**
  - **Input:** "flat" features $\mathbf{x} \in \mathbb{R}^n$ (e.g., color, size, position)
  - **Output:** scalar value y (e.g., is it round and yellow?)
- **No other prior knowledge (minimal assumptions)**

What Can Neural Networks Reason about? Xu et al. ICLR 2020

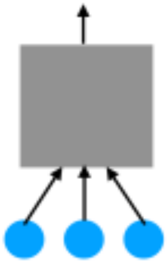
Problem 1: feature extraction

- **Problem 1 (task on one object):**
  - **Input:** "flat" features $\mathbf{x} \in \mathbb{R}^n$ (e.g., color, size, position)
  - **Output:** scalar value y (e.g., is it round and yellow?)
- **No other prior knowledge (minimal assumptions)**

- **Q:** What neural network choice suits this problem?
- **A: MLPs (multilayer perceptrons)**
  - Universal approximator
  - Makes no assumptions on input/output structure



Architectures and Problem Type



- **MLP**
  - task on one object
  - ~ feature extraction

Lets consider tasks on many objects…

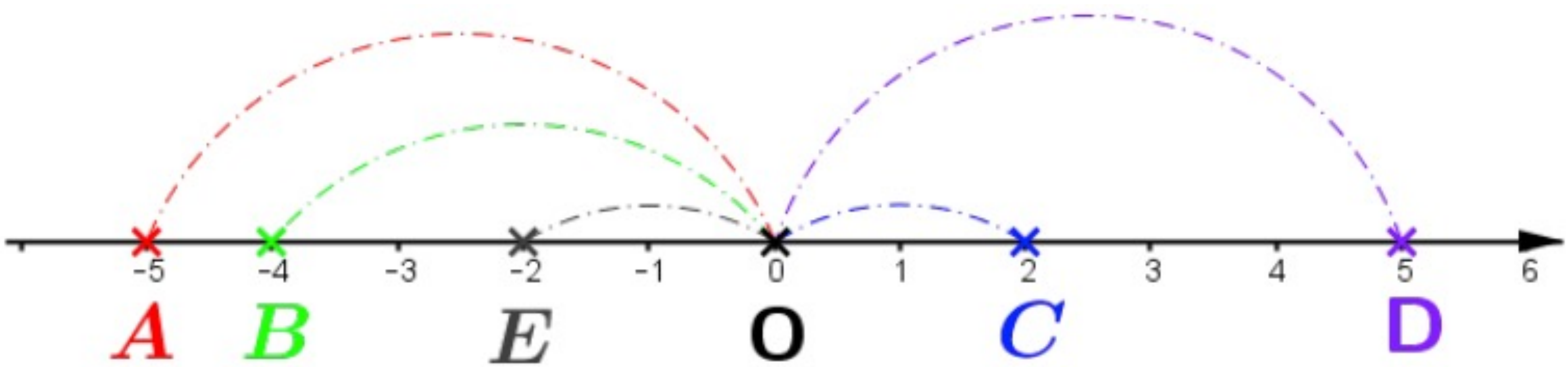
What Can Neural Networks Reason about? Xu et al. ICLR 2020

Problem 2: Summary statistics

- **Problem 2 (summary statistics):**
  - **Input:** a set of objects $\{\mathbf{x_i}\}$, each with features containing their coordinate and color $\mathbf{x_i} = [\mathbf{x_i^{color}}, \mathbf{x_i^{coordinate}}]$

Problem 2: Summary statistics

- **Problem 2 (summary statistics):**
  - **Input:** a set of objects $\{\mathbf{x_i}\}$, each with features containing their coordinate and color $\mathbf{x_i} = [\mathbf{x_i^{color}}, \mathbf{x_i^{coordinate}}]$
  - **Task Output:** some aggregate property of the set (e.g., largest x-coordinate)

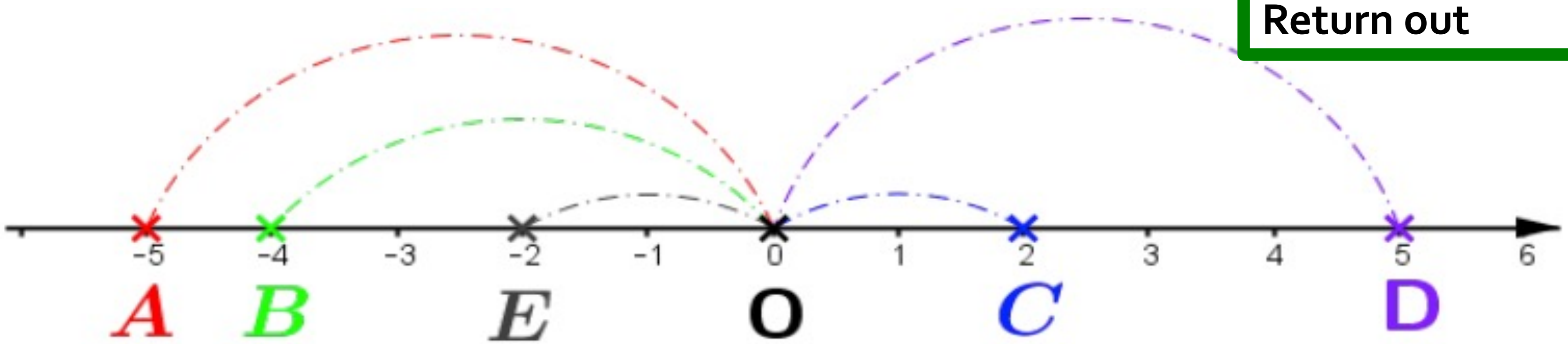


(Answer: 5)

Problem 2: Summary statistics

- **Problem 2 (summary statistics):**
  - **Input:** a set of objects $\{\mathbf{x_i}\}$, each with features containing their coordinate and color $\mathbf{x_i} = [\mathbf{x_i^{color}}, \mathbf{x_i^{coordinate}}]$
  - **Task Output:** some aggregate property of the set (e.g., largest x-coordinate
  - $\mathbf{y(\{x_i\}) = max_i(x_i^{coordinate})}$

out = -inf
For i $= 1, \ldots$
if $\mathbf{x_i^{coordinate}} > \mathbf{out}$:
out = $\mathbf{x_i^{coordinate}}$
Return out

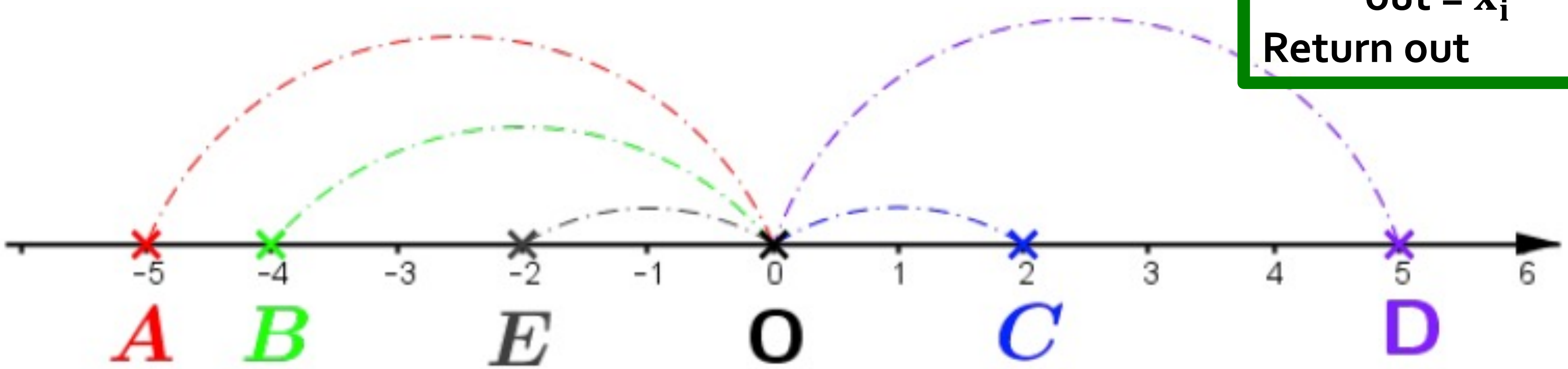


(Answer: 5)

Problem 2: Summary statistics

- MLP model: $\mathrm{MLP}(\mathbf{x}_1, \dots, \mathbf{x}_n)$
- **Not** well suited to this task
- To learn max (and min) MLP **has to learn to execute a for-loop**
- This is a complex operation, MLP needs lots of data to learn

```
out = -inf
For i = 1, ...
  if x_i^coordinate > out:
     out = x_i^coordinate
Return out
```



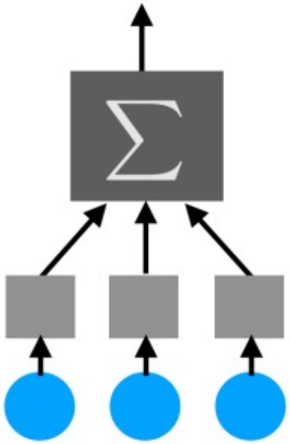
$$\mathbf{y}(\{\mathbf{x}_\mathrm{i}\}) = \mathbf{max}_\mathrm{i}(\mathbf{x}_\mathrm{i}^{\mathbf{coordinate}})$$

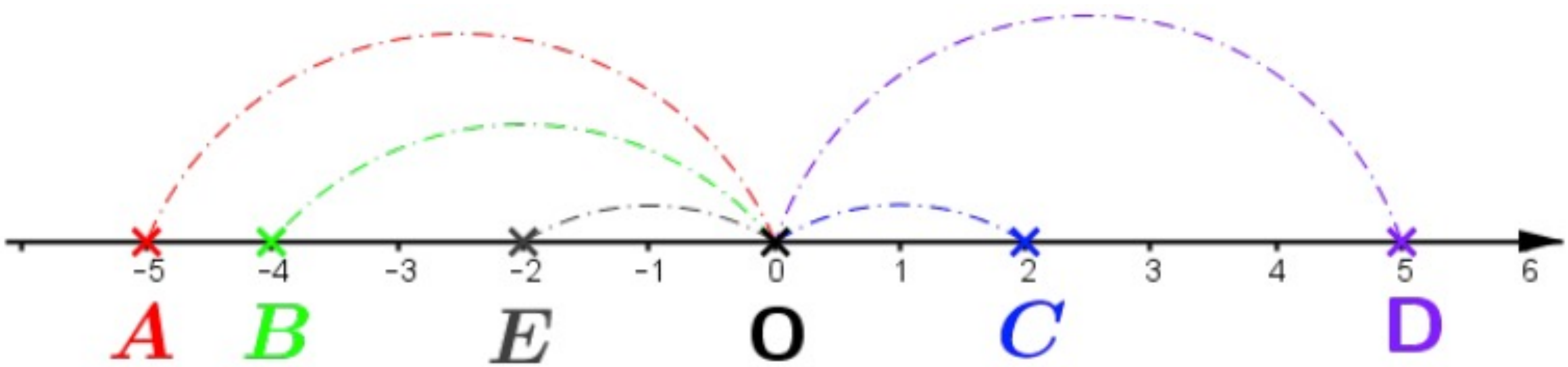
(Answer: 5)

What Can Neural Networks Reason about? Xu et al. ICLR 2020

Problem 2: Summary statistics

- New DeepSet model:
  - $\text{DeepSet}(\{\mathbf{x_i}\}) = \text{MLP}_1(\sum_i \text{MLP}_2(\mathbf{x}_i))$
- **Well suited to this task**
- **Why?**



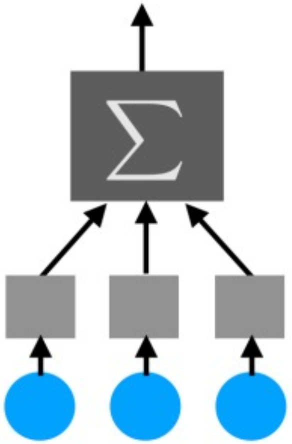


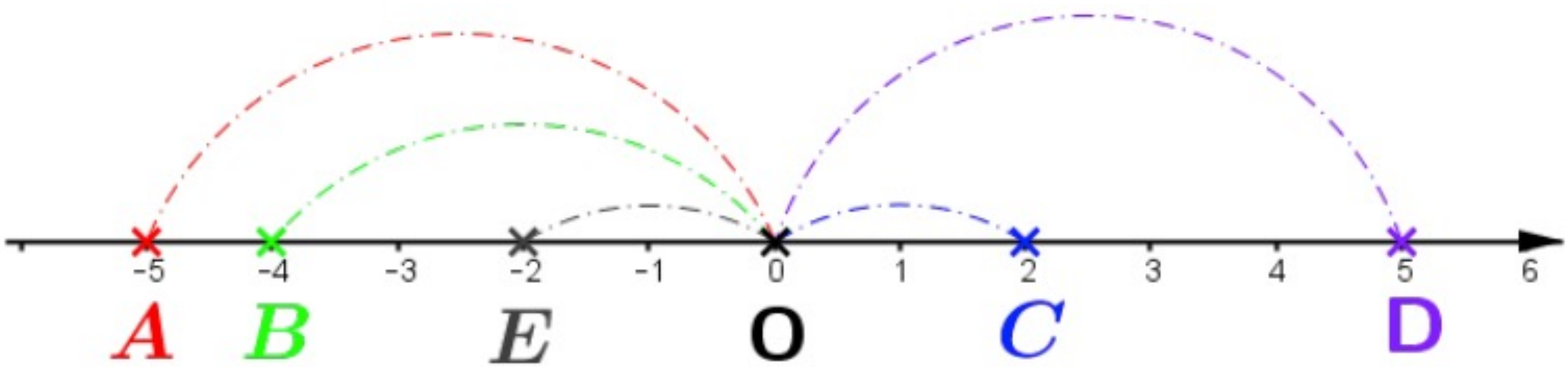
$\mathbf{y}(\{\mathbf{x_i}\}) = \mathbf{max_i}(\mathbf{x_i^{coordinate}})$

(Answer: 5)

Problem 2: Summary statistics

- New DeepSet model:
  - $\text{DeepSet}(\{\mathbf{x_i}\}) = \text{MLP}_1(\sum_i \text{MLP}_2(\mathbf{x_i}))$
- **Well suited to this task**
- **Why? Can approx. softmax**, a simple approx. to max
  - $\max_i(x_i^{\text{coordinate}}) \approx \mathbf{log}\left(\sum_i e^{\mathbf{x_i^{coordinate}}}\right)$ (MLP_1 learns log, MLP_2 learns exp)

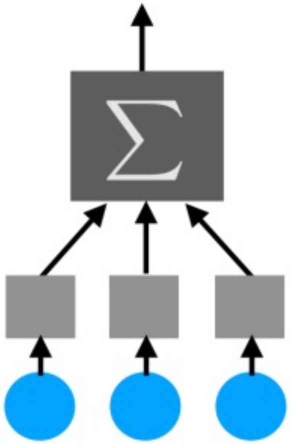




$$\mathbf{y(\{x_i\}) = max_i(x_i^{coordinate})}$$

(Answer: 5)

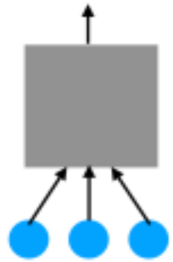
Problem 2: Summary statistics



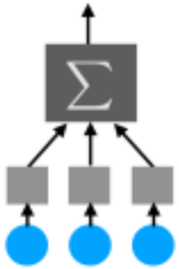
- New DeepSet model:
  - $\text{DeepSet}(\{\mathbf{x_i}\}) = \text{MLP}_1(\sum_i \text{MLP}_2(\mathbf{x}_i))$
- **Well suited to this task**
- **Why? Can approx. softmax**, a simple approx. to min/max
  - $\max_i(x_i^{\text{coordinate}}) \approx \mathbf{log}\left(\sum_i e^{\mathbf{x_i^{coordinate}}}\right)$ (MLP_1 learns log, MLP_2 learns exp)
- Key point:
  - Consequence: MLPs only must learn **simple functions** (log / exp)
  - This can be done easily, without needing much data
- MLP can provably also learn this. But must learn complex for-loop, which requires lots of training data

What Can Neural Networks Reason about? Xu et al. ICLR 2020

Architectures and Problem Type



- **MLP**
  - Task on one object
  - ~ feature extraction



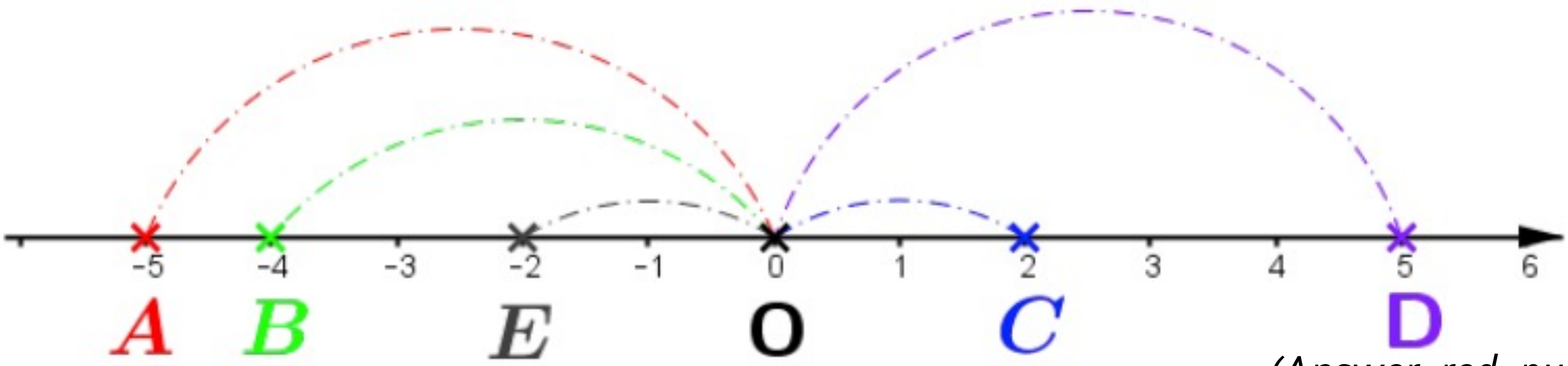
- **DeepSet**
  - Task on many objects
  - ~ summary statistics
  - $\mathbf{y(\{x_i\}) = max_i(x_i^{coordinate})}$

Lets consider a harder task on many objects…

What Can Neural Networks Reason about? Xu et al. ICLR 2020

Problem 3: Relational argmax

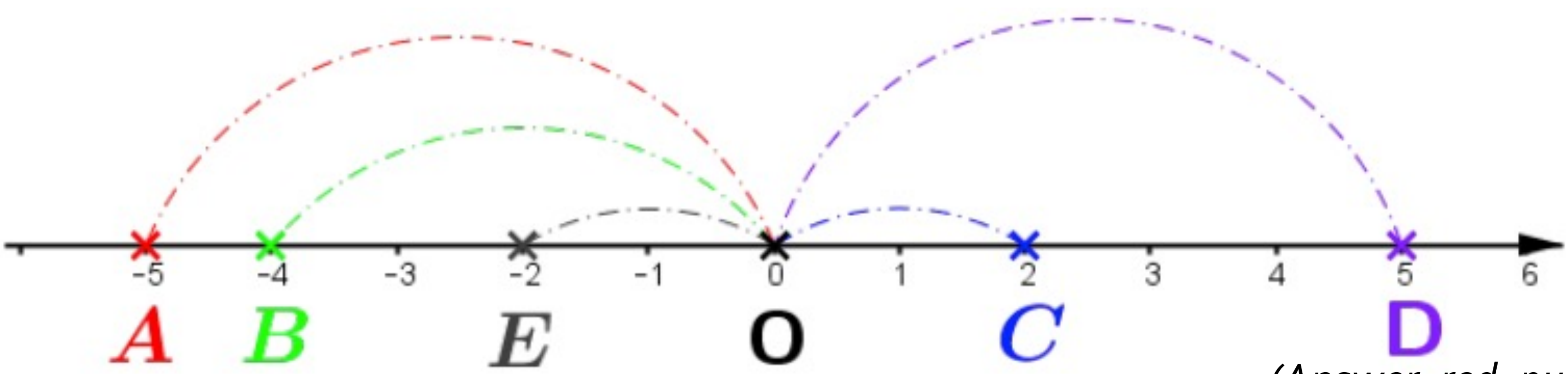
- **Problem 3 (relational argmax):**
  - **Input:** a set of objects $\{\mathbf{x_i}\}$, each with features containing their coordinate and color $\mathbf{x_i} = [\mathbf{x_i^{color}}, \mathbf{x_i^{coordinate}}]$



(Answer: red, purple)

Problem 3: Relational argmax

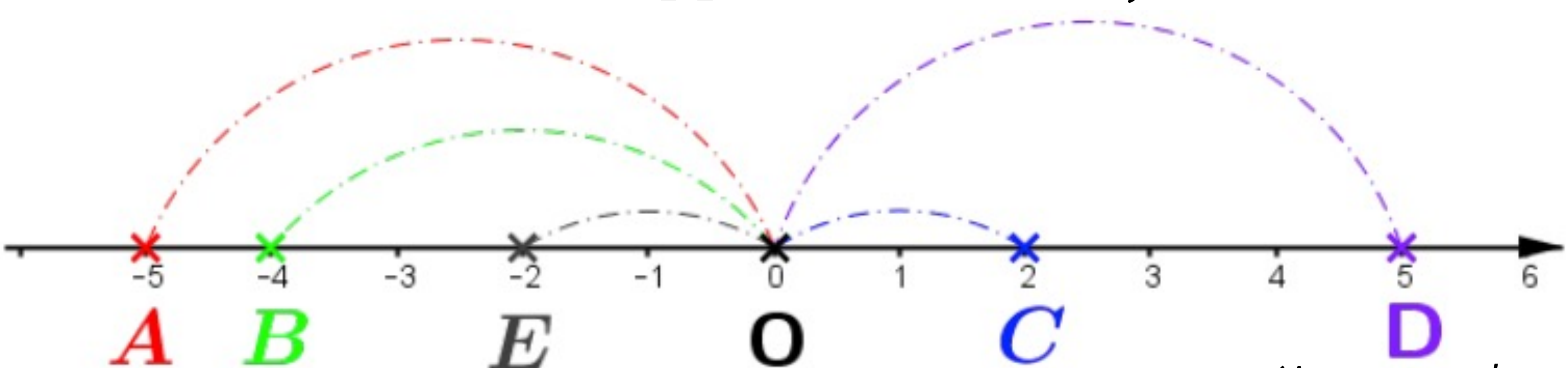
- **Problem 3 (relational argmax):**
  - **Input:** a set of objects $\{\mathbf{x_i}\}$, each with features containing their coordinate and color $\mathbf{x_i} = [\mathbf{x_i^{color}}, \mathbf{x_i^{coordinate}}]$
  - **Task Output:** property of pairwise relation (e.g., what are the colors of the two furthest away objects?)



(Answer: red, purple)

Problem 3: Relational argmax

- **Problem 3 (relational argmax):**
  - **Input:** a set of objects $\{\mathbf{x_i}\}$, each with features containing their coordinate and color $\mathbf{x_i} = [\mathbf{x_i^{color}}, \mathbf{x_i^{coordinate}}]$
  - **Task Output:** property of pairwise relation (e.g., what are the colors of the two furthest away objects?)
  - $\mathbf{y(\{x_i\})} = (\mathbf{x_{i_1}^{color}}, \mathbf{x_{i_2}^{color}})$
    $\mathbf{s.t.}\ \ \mathbf{i_1, i_2} = \mathbf{argmax_{i_1 i_2}} ||\mathbf{x_i^{coordinate}} - \mathbf{x_j^{coordinate}}||$

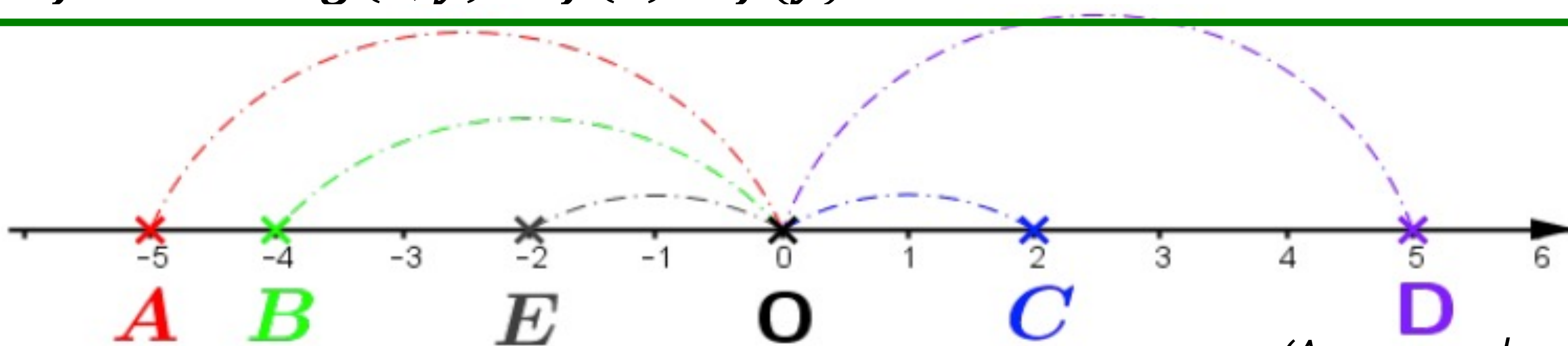


(Answer: red, purple)

Problem 3: Relational argmax

- **DeepSet poorly suited to modelling pairwise relations**
  - **Recall:** $\text{DeepSet}(\{\mathbf{x_i}\}) = \text{MLP}_2(\sum_i \text{MLP}_1(\mathbf{x_i}))$
- **Reason:**
  - task requires comparing pairs of objects – i.e., a for-loop
  - each object processed independently by MLP_1
  - Consequence: MLP_2 has to learn complex for-loop **(hard)**
- $\sum_i \text{MLP}_1(\mathbf{x_i})$ **provably** cannot learn pairwise relations

  Theorem: Suppose $g(x, y) = 0$ if and only if $x = y$. Then there is no f such that $g(x, y) = f(x) + f(y)$

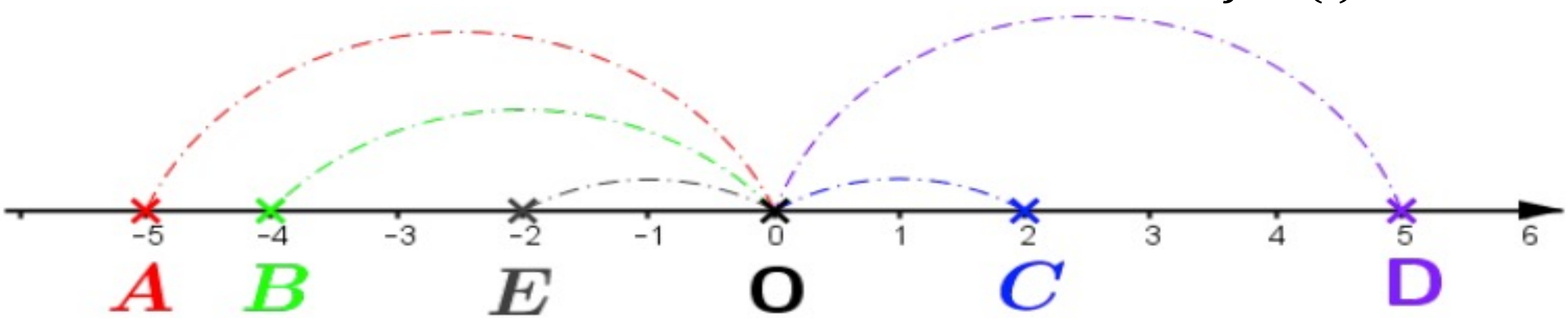


(Answer: red, purple)

$$\mathbf{y}(\{\mathbf{x_i}\}) = \left(\mathbf{x}_{\mathbf{i_1}}^{\mathbf{color}}, \mathbf{x}_{\mathbf{i_2}}^{\mathbf{color}}\right) \text{ s.t. } \mathbf{i_1}, \mathbf{i_2} = \mathbf{argmax}_{\mathbf{i_1 i_2}} ||\mathbf{x}_\mathbf{i}^{\mathbf{coordinate}} - \mathbf{x}_\mathbf{j}^{\mathbf{coordinate}}||$$

Problem 3: Relational argmax

- GNN well suited to this task: for-loop is built in!
  - E.g., recall GIN update
  - **For $i = 1, \dots, n$:**
    - $h_i^{l+1} = \mathbf{MLP_2}(\mathbf{MLP_1}(h_i^l) + \sum_{j \in N(i)} \mathbf{MLP_1}(h_j^l))$
  - Update of node embedding depends on other nodes
    - $\mathbf{MLP_1}$ computes distance from i to j
    - $\mathbf{MLP_2}$ identifies **which pair is best in** $\{(i, j)\}_{j \in N(i)}$



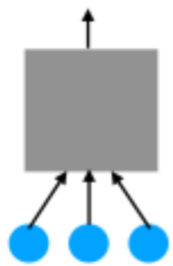
ple)

$$y(\{x_i\}) = (x_{i_1}^{color}, x_{i_2}^{color}) \text{ s.t. } i_1, i_2 = \text{argmax}_{i_1 i_2} ||x_i^{coordinate} - x_j^{coordinate}||$$

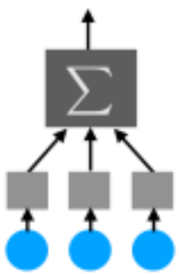
[What Can Neural Networks Reason about? Xu et al. ICLR 2020]

Architectures and Problem Type

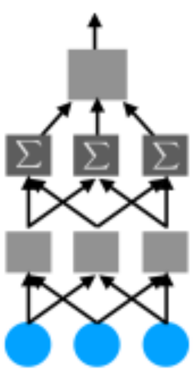
In each case, the neural net architecture "fits" the computations needed to compute the target… we will come back to this



- **MLP**
  - Task on one object
  - ~ feature extraction



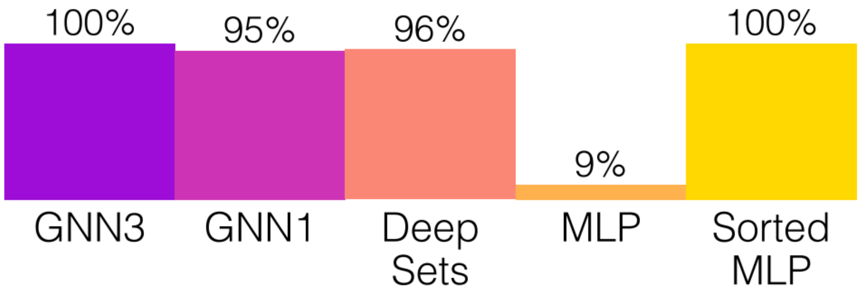
- **DeepSet**
  - Task on many objects
  - ~ summary statistics (max value difference)
  - $\mathbf{y(\{x_i\}) = max_i(x_i^{coordinate})}$



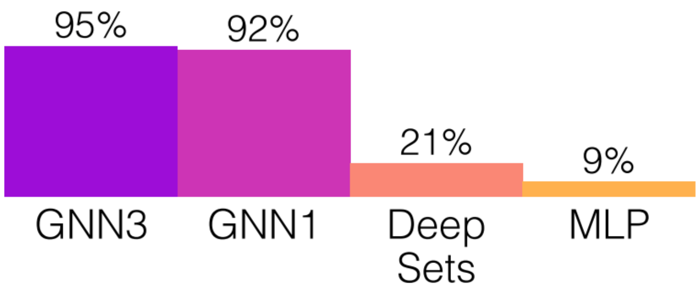
- **GNN**
  - Task on many objects
  - ~ pairwise relations (relational argmax)
  - $\mathbf{y(\{x_i\}) = (x_{i_1}^{color}, x_{i_2}^{color})\ s.t.\ \ i_1, i_2 = argmax_{i_1 i_2} ||x_i^{coordinate} - x_j^{coordinate}||}$

Results in practice

- Task 2: maximum value MLP fails due to inability to compute max



- Task 3: relational argmax
  - Both DeepSet and MLP fail



General algorithm class for GNN?

- **GNNs are good at solving tasks that require relating pairs of objects (nodes)**
  - MLPs/DeepSets cannot do this easily since they have to learn for-loop

- “Relational argmax” is just one problem that GNN can solve…

- What is the **general class** of algorithms GNNs can run?

Plan for Today

- **Part 1**
  - An algorithm GNNs can run
- **Part 2**
  - Algorithmic structure of neural network architectures
- **Part 3**
  - What class of graph algorithms can GNNs simulate?
- **Part 4**
  - Algorithmic alignment: a principle for neural net design

Stanford CS224W: Algorithmic Class of GNNs

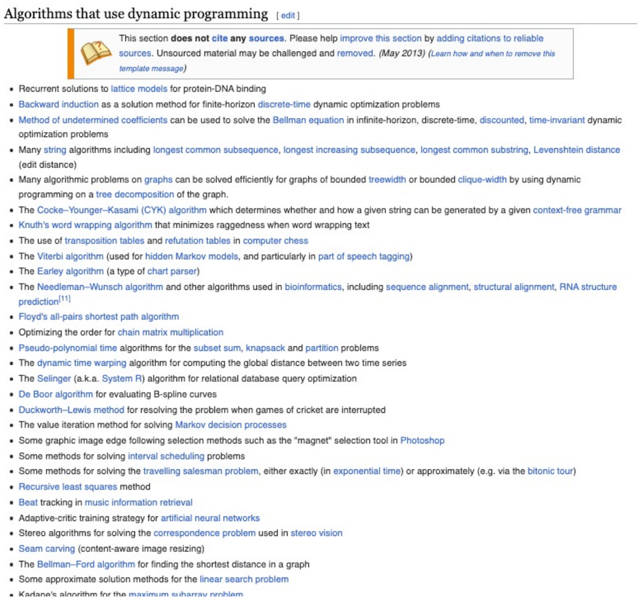
CS224W: Machine Learning with Graphs

Joshua Robinson, Stanford University

http://cs224w.stanford.edu

Dynamic Programming

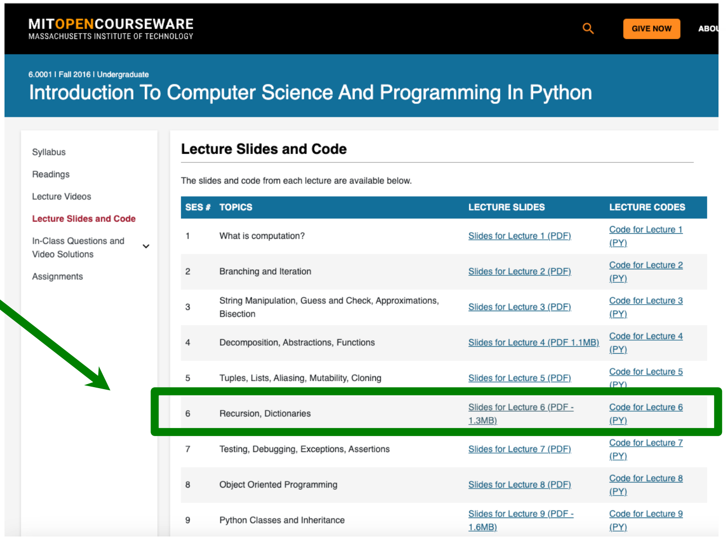
- Fundamental algorithmic paradigm
- One of the most influential algorithm classes in computer science (lecture 6 in MIT's intro to Comp Sci)
- **Works by recursively breaking a problem into smaller instances of the same problem type**



Algorithms that use dynamic programming [edit]

This section **does not** cite **any** sources. Please help improve this section by adding citations to reliable sources. Unsourced material may be challenged and removed. *(May 2013) (Learn how and when to remove this template message)*

- Recurrent solutions to lattice models for protein-DNA binding
- Backward induction as a solution method for finite-horizon discrete-time dynamic optimization problems
- Method of undetermined coefficients can be used to solve the Bellman equation in infinite-horizon, discrete-time, discounted, time-invariant dynamic optimization problems
- Many string algorithms including longest common subsequence, longest increasing subsequence, longest common substring, Levenshtein distance (edit distance)
- Many algorithmic problems on graphs can be solved efficiently for graphs of bounded treewidth or bounded clique-width by using dynamic programming on a tree decomposition of the graph.
- The Cocke–Younger–Kasami (CYK) algorithm which determines whether and how a given string can be generated by a given context-free grammar
- Knuth's word wrapping algorithm that minimizes raggedness when word wrapping text
- The use of transposition tables and refutation tables in computer chess
- The Viterbi algorithm (used for hidden Markov models, and particularly in part of speech tagging)
- The Earley algorithm (a type of chart parser)
- The Needleman–Wunsch algorithm and other algorithms used in bioinformatics, including sequence alignment, structural alignment, RNA structure prediction[11]
- Floyd's all-pairs shortest path algorithm
- Optimizing the order for chain matrix multiplication
- Pseudo-polynomial time algorithms for the subset sum, knapsack and partition problems
- The dynamic time warping algorithm for computing the global distance between two time series
- The Selinger (a.k.a. System R) algorithm for relational database query optimization
- De Boor algorithm for evaluating B-spline curves
- Duckworth–Lewis method for resolving the problem when games of cricket are interrupted
- The value iteration method for solving Markov decision processes
- Some graphic image edge following selection methods such as the "magnet" selection tool in Photoshop
- Some methods for solving interval scheduling problems
- Some methods for solving the travelling salesman problem, either exactly (in exponential time) or approximately (e.g. via the bitonic tour)
- Recursive least squares method
- Beat tracking in music information retrieval
- Adaptive-critic training strategy for artificial neural networks
- Stereo algorithms for solving the correspondence problem used in stereo vision
- Seam carving (content-aware image resizing)
- The Bellman–Ford algorithm for finding the shortest distance in a graph
- Some approximate solution methods for the linear search problem
- Kadane's algorithm for the maximum subarray problem



MITOPENCOURSEWARE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
GIVE NOW ABOU

6.0001 | Fall 2016 | Undergraduate

Introduction To Computer Science And Programming In Python

Syllabus
Readings
Lecture Videos
Lecture Slides and Code
In-Class Questions and Video Solutions
Assignments

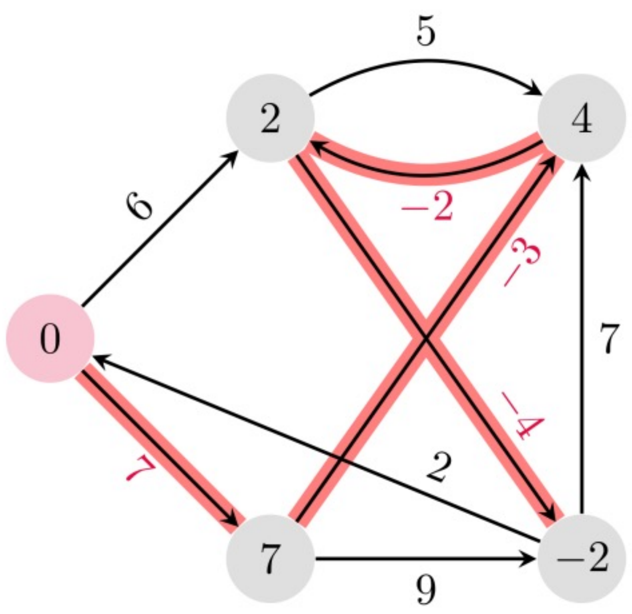
Lecture Slides and Code

The slides and code from each lecture are available below.

SES #	TOPICS	LECTURE SLIDES	LECTURE CODES
1	What is computation?	Slides for Lecture 1 (PDF)	Code for Lecture 1 (PY)
2	Branching and Iteration	Slides for Lecture 2 (PDF)	Code for Lecture 2 (PY)
3	String Manipulation, Guess and Check, Approximations, Bisection	Slides for Lecture 3 (PDF)	Code for Lecture 3 (PY)
4	Decomposition, Abstractions, Functions	Slides for Lecture 4 (PDF 1.1MB)	Code for Lecture 4 (PY)
5	Tuples, Lists, Aliasing, Mutability, Cloning	Slides for Lecture 5 (PDF)	Code for Lecture 5 (PY)
6	Recursion, Dictionaries	Slides for Lecture 6 (PDF - 1.3MB)	Code for Lecture 6 (PY)
7	Testing, Debugging, Exceptions, Assertions	Slides for Lecture 7 (PDF)	Code for Lecture 7 (PY)
8	Object Oriented Programming	Slides for Lecture 8 (PDF)	Code for Lecture 8 (PY)
9	Python Classes and Inheritance	Slides for Lecture 9 (PDF - 1.6MB)	Code for Lecture 9 (PY)

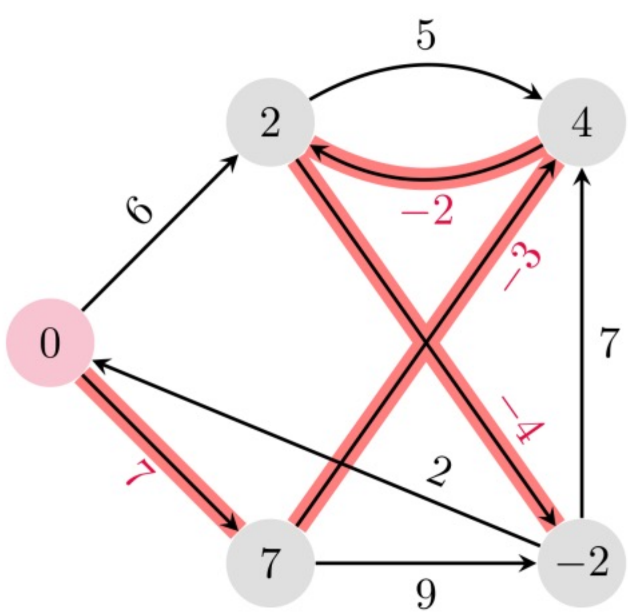
Dynamic Programming

- Task 4 (shortest path):
  - **Input:** a weighted graph and a chosen source node
  - **Output:** all shortest paths out of source node (shortest path tree)



Dynamic Programming

- Task 4 (shortest path):
  - **Input:** a weighted graph and a chosen source node
  - **Output:** all shortest paths out of source node (shortest path tree)
- Algorithmic solution: Bellman-Ford



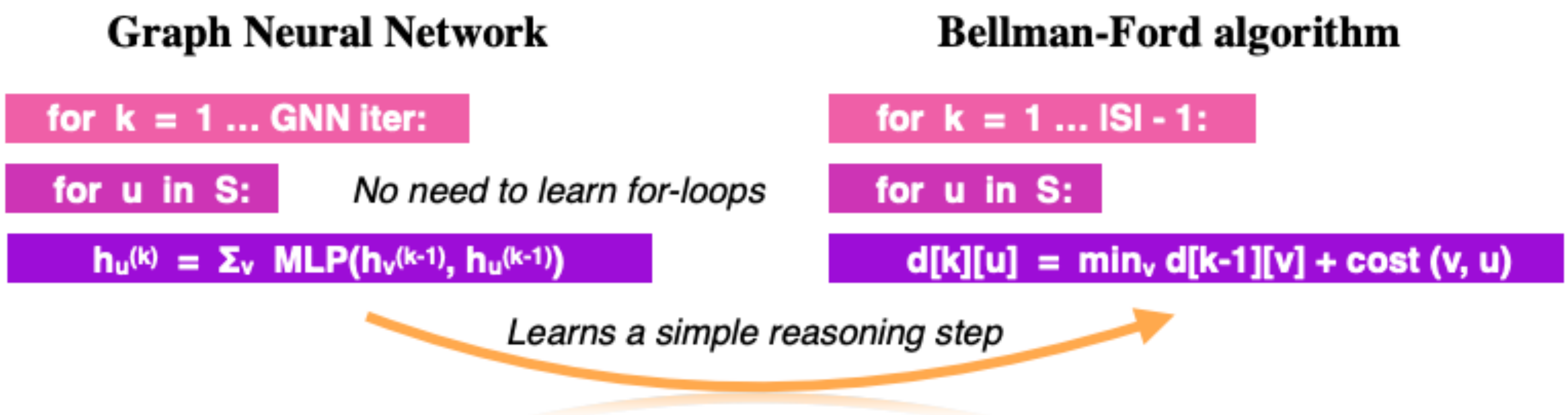
Bellman-Ford algorithm

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

[What Can Neural Networks Reason about? Xu et al. ICLR 2020](#)

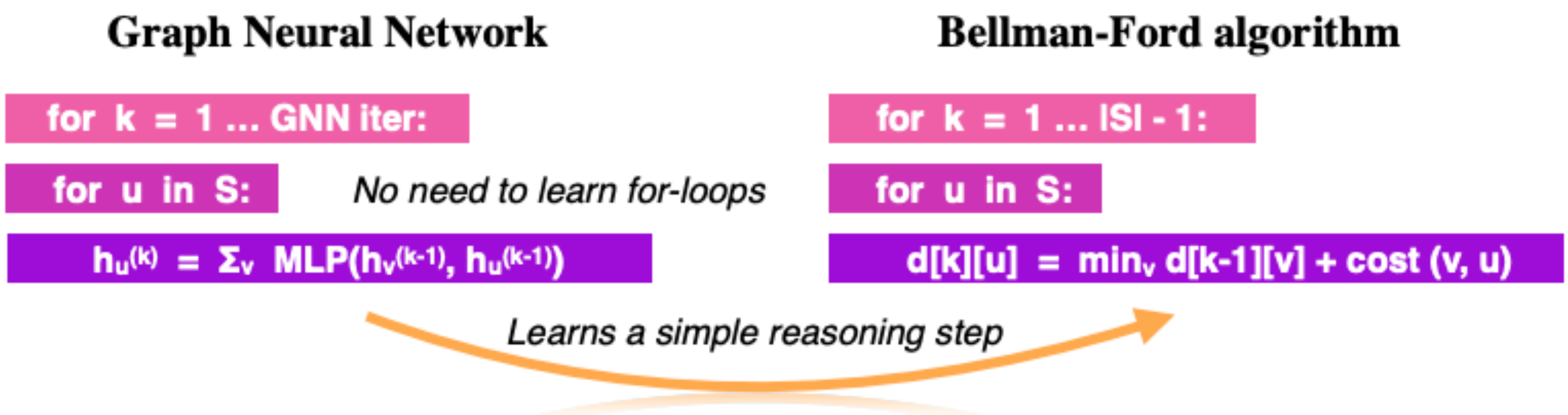
GNNs are Dynamic Programs

- Dynamic programming has very similar form to GNN



GNNs are Dynamic Programs

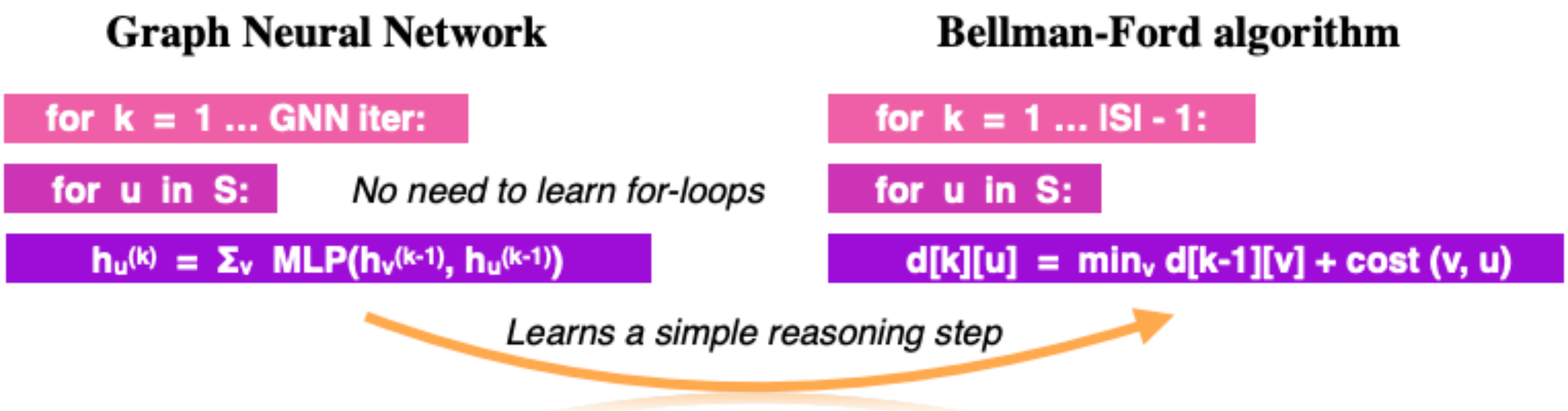
- Dynamic programming has very similar form to GNN
- Both have nested for-loops over:
  - Number of GNN layers / iterations of BF
  - Each node in graph



What Can Neural Networks Reason about? Xu et al. ICLR 2020

GNNs are Dynamic Programs

- Dynamic programming has very similar form to GNN
- Both have nested for-loops over:
  - Number of GNN layers / iterations of BF
  - Each node in graph
- GNN aggregation + **MLP** only needs to learn **sum + min**
- **An MLP trying to learn a DP has to learn double-nested for loop – really hard to do!**



GNNs are Dynamic Programs

- There is an even better choice of GNN…
  - Choose **min activation** to match DP
  - Then MLP only needs to learn **linear function!**

GNN Architectures

$$h_u^{(k)} = \Sigma_v \ \text{MLP}^{(k)}\left(h_v^{(k-1)}, h_u^{(k-1)}, w(v,u)\right)$$

✗ *MLP has to learn non-linear steps*

$$h_u^{(k)} = \min_v \ \text{MLP}^{(k)}\left(h_v^{(k-1)}, h_u^{(k-1)}, w(v,u)\right)$$

✓ *MLP learns linear steps*

DP Algorithm (Target Function)

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

GNNs are Dynamic Programs

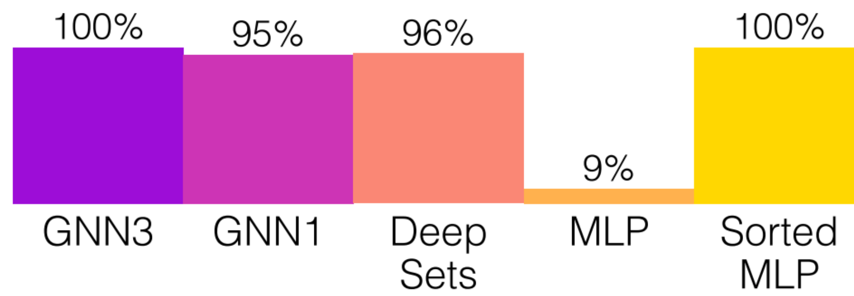
- We expect GNNs to be good at solving tasks that can be solved with DP
  - E.g., shortest paths

- **Does this actually happen?**

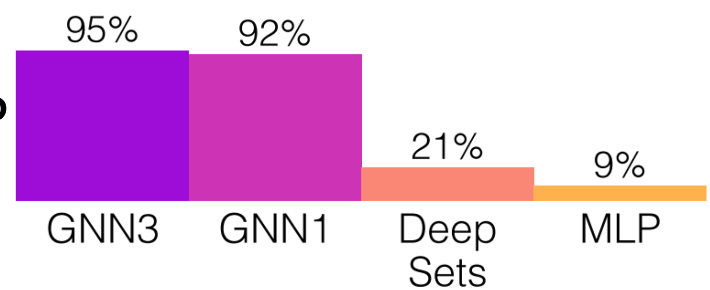
What Can Neural Networks Reason about? Xu et al. ICLR 2020

Results in practice

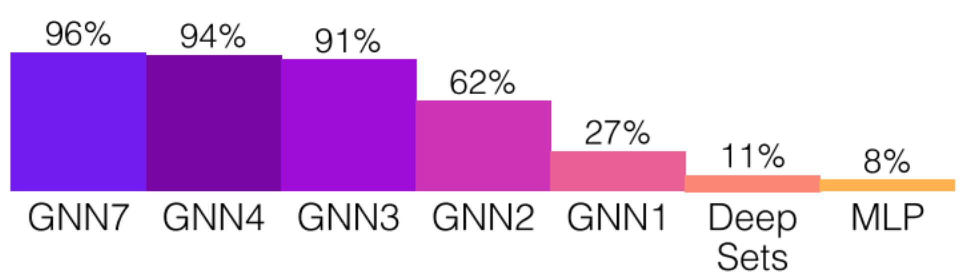
- Task 2: maximum value MLP fails due to inability to compute max



- Task 3: relational argmax
  - Both DeepSet and MLP fail



- Task 4: shortest path (dynamic programming)
  - Task shortest path length up to 7
  - 7 layer GNN gets best performance



Conclusion

- **Goal:** understand what tasks GNNs are good at solving
  - We are **not** focusing on expressivity
  - Instead we are interested in how **easy** it is to learn the solution (e.g., how much data the model needs to see)
- **GNN message passing is a dynamic programing algorithm**
- Consequence: GNNs are a good choice of architecture for tasks that can be solved by a DP (e.g., finding shortest paths)

Plan for Today

- **Part 1**
  - An algorithm GNNs can run
- **Part 2**
  - Algorithmic structure of neural network architectures
- **Part 3**
  - What class of graph algorithms can GNNs simulate?
- **Part 4**
  - Algorithmic alignment: a principle for neural net design

Stanford CS224W: Algorithmic Alignment

CS224W: Machine Learning with Graphs

Joshua Robinson, Stanford University

http://cs224w.stanford.edu

Algorithmic-Centric Principle For Neural Network Design

- In the previous section we studied what type of tasks GNNs excel at solving
  - **Key idea:** focus on the algorithm that solves the task
  - If the neural net can express the algorithm easily, then it's a good choice of architecture

- **How to formulate a general principle?**

Algorithmic Alignment

Algorithmic Alignment

Given a target algorithm $g = g_m \circ \cdots \circ g_1$, a neural network architecture $f = f_m \circ \cdots \circ f_1$ if:

- g_i a simple function
- f_i can express g_i
- Each f_i has few learnable parameters (so can learn g_i easily)

- If you remember any phrase from today, let it be **algorithmic alignment** – all of todays lecture can be understood with this idea
- About **how** a model expresses a target function, **not if** (i.e., expressive power). Recall that an MLP is a universal approximator
- **Intuition:** overall algorithm can be learned more easily by learning individual simple steps

Designing New Neural Nets with Algorithmic Alignment

- GNN is **algorithmically aligned** to dynamic programming (DP)
- But algorithmic alignment is a **general principle** for designing neural network architectures
- So we should be able to use it to design entirely new neural networks given a particular problem

Designing New Neural Nets with Algorithmic Alignment

- Many successful example of this in the literature
  - Neural Shuffle-Exchange Networks (Freivalds et al., NeurIPS'19)
    - Linearithmic algorithms
  - Neural Execution of Graph Algorithms (Veličković et al., ICLR'20)
    - Improved dynamic programming
  - PrediNet (Shanahan et al., ICML'20)
    - Predicate Logic
  - IterGNNs (Tang et al., NeurIPS'20)
    - Iterative algorithms
  - Pointer Graph Networks (Veličković et al., NeurIPS'20)
    - Pointer-based data structures
  - Persistent Message Passing (Strathmann et al., ICLR'21 SimDL)
    - Persistent data structures

Stanford CS224W: Applications of Algorithmic Alignment

CS224W: Machine Learning with Graphs

Joshua Robinson, Stanford University

http://cs224w.stanford.edu

Designing New Neural Nets with Algorithmic Alignment

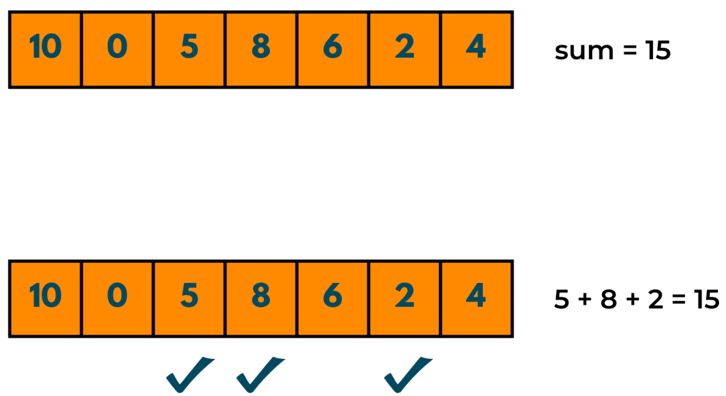
- **Application 1:** building a network to solve a new task
  - The subset-sum problem (NP-hard)
- **Application 2:** building neural networks that can generalize out-of-distribution
  - The linear algorithmic alignment hypothesis

Solving an NP-hard Task: Subset Sum

- **Task:** given a set of numbers S, decide if there exists a subset that sums to k

Solving an NP-hard Task: Subset Sum

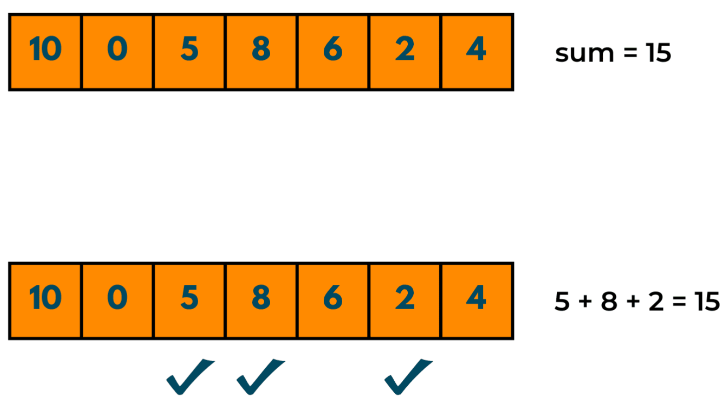
- **Task:** given a set of numbers S, decide if there exists a subset that sums to k



- Known to be NP-hard, no DP algorithm can solve this (so GNN not suitable)

Solving an NP-hard Task: Subset Sum

- **Exhaustive Search Algorithm for solving subset sum:**
  - Loop over all subsets $\tau \in S$ and check if sum is k
- **Clearly not polynomial time… but can it inspire a neural net architecture?**

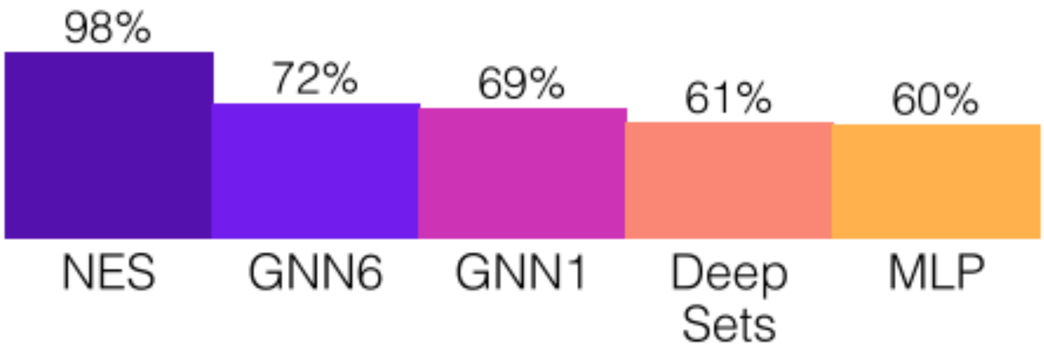


Solving an NP-hard Task: Subset Sum

- **Exhaustive Search Algorithm for solving subset sum:**
  - Loop over all subsets $\tau \in S$ and check if sum is k
- **Clearly not polynomial time… but can it inspire a neural net architecture?**
- **Neural Exhaustive Search:**
  - **Given** $S = \{X_1, \dots, X_n\}$,
  - $\text{NES}(S) = \text{MLP}\left(\max_{\tau \subseteq S} \text{LSTM}(X_1, \dots, X_{|\tau|} : X_1, \dots, X_{|\tau|} \in \tau\right)$
    - **Algorithmically aligned to exhaustive search:**
      - **LSTM learns if the sum** $X_1 + \dots + X_{|\tau|} = k$ **(simple function)**
      - **Max aggregation identifies best subset**
      - **MLP maps to true/false value**

Solving an NP-hard Task: Subset Sum

- **Result in practice**
- **Random guessing gets 50% accuracy**



- **Neural Exhaustive Search:**
  - **Given** $S = \{X_1, \ldots, X_n\}$,
  - $\text{NES}(S) = \text{MLP}\left(\max_{\tau \subseteq S} \text{LSTM}(X_1, \ldots, X_{|\tau|} : X_1, \ldots, X_{|\tau|} \in \tau\right)$
    - **Algorithmically aligned to exhaustive search:**
      - **LSTM learns if the sum** $X_1 + \ldots + X_{|\tau|} = k$ **(simple function)**
      - **Max aggregation identifies best subset**
      - **MLP maps to true/false value**

Designing New Neural Nets with Algorithmic Alignment

- **Application 1:** building a network to solve a new task
  - The subset-sum problem (NP-hard)
- **Application 2:** building neural networks that can generalize out-of-distribution
  - The linear algorithmic alignment hypothesis

How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks, Xu et al. ICLR 2021

Algorithmic Alignment and Extrapolation

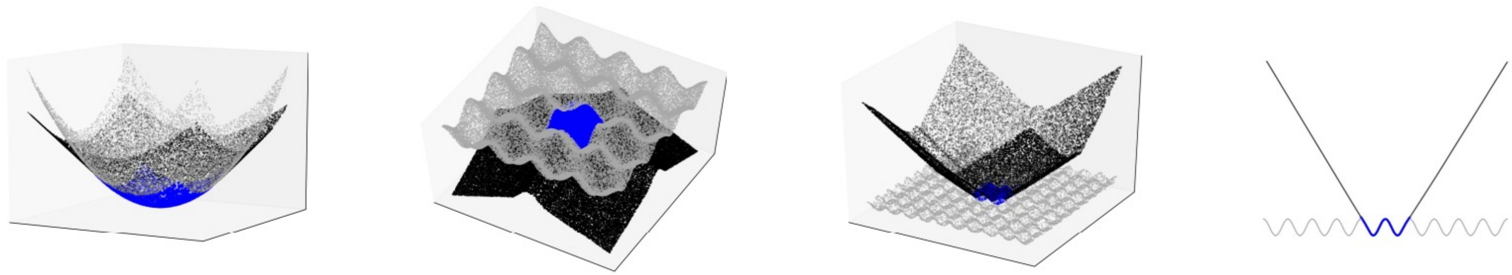
- We have argued that **algorithmic alignment** can help inspire architectures well suited to particular tasks
  - By well suited, we **mean generalizes well using little training data**
- **But true AI requires something stronger than this…**
  - Also needs to **"extrapolate"** to instances that look very different from the training data

Algorithmic Alignment and Extrapolation

- Extrapolation is also called out-of-distribution generalization
- Extrapolation is **a holy grail of AI**, necessary for systems to **behave reliably** in unforeseen future situations
- **Can algorithmic alignment help with extrapolation?**
  - Let's start with a simple but important observation

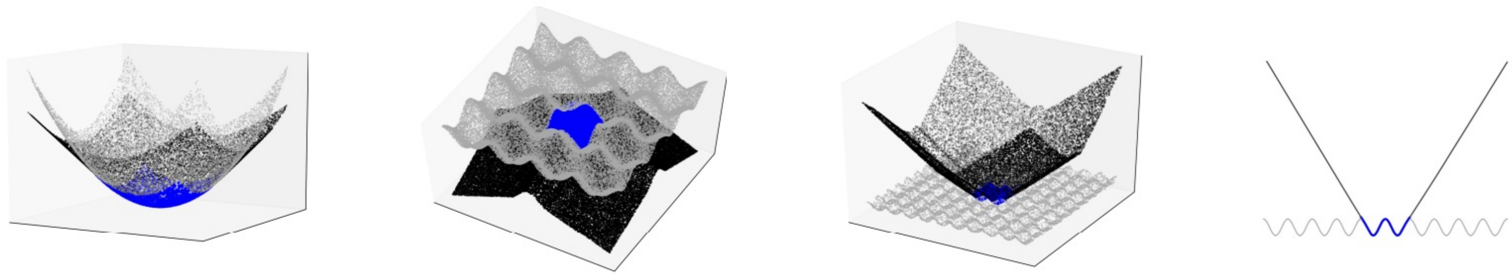
How MLPs extrapolate

- **Observation:** ReLU MLPs extrapolate **linearly**



How MLPs extrapolate

- **Observation:** ReLU MLPs extrapolate **linearly**



- Can be proved that extrapolation is perfect for linear target functions
- But ReLU MLPs cannot generalize for non-linear target functions…
- **The need for linearity for MLP extrapolation suggests a hypothesis for GNN extrapolation…**

The Linear Algorithmic Alignment Hypothesis

Linear Algorithmic Alignment Hypothesis

Linear algorithmic alignment implies a neural network can extrapolate to unseen data

How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks, Xu et al. ICLR 2021

The Linear Algorithmic Alignment Hypothesis

Linear Algorithmic Alignment Hypothesis

Linear algorithmic alignment implies a neural network can extrapolate to unseen data

Linear Algorithmic Alignment

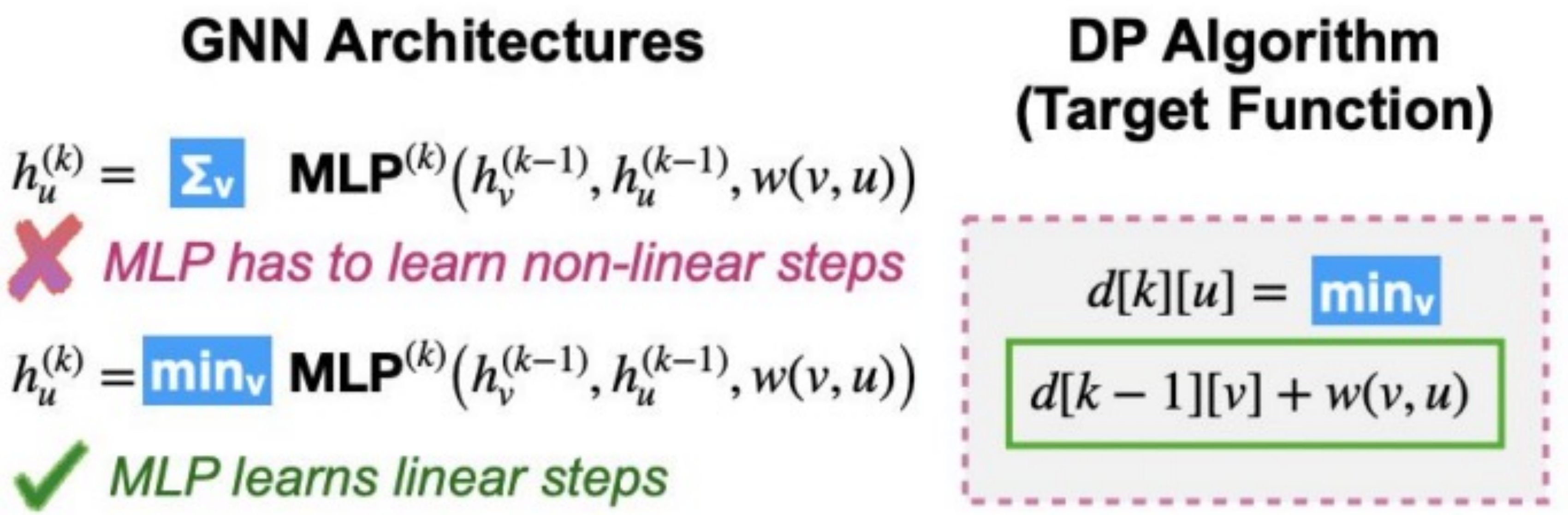
Given a target algorithm $\boldsymbol{g} = \boldsymbol{g_m} \circ \cdots \circ \boldsymbol{g_1}$, a neural network architecture $\boldsymbol{f} = \boldsymbol{f_m} \circ \cdots \circ \boldsymbol{f_1}$ linearly aligns if:

- f_i can express g_i
- f_i contains a combination of non-linearities and MLPs
- Each MLP in f_i only has to learn a linear map to perfectly fit g_i

How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks, Xu et al. ICLR 2021

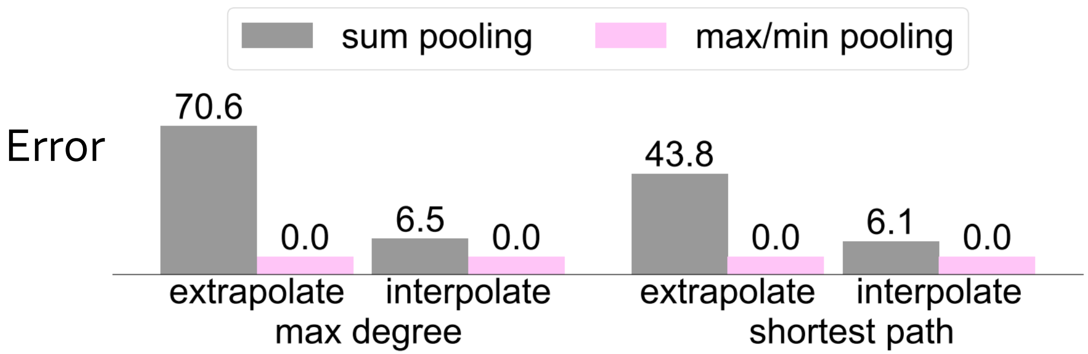
How GNNs extrapolate

- Recall GNN for learning dynamic programs
- GNN aggregation function is key
  - **Min aggregation is linearly algorithmically aligned**
  - Sum aggregation is not
- Does linear algorithmic alignment lead to extrapolation?



[How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks, Xu et al. ICLR 2021](#)

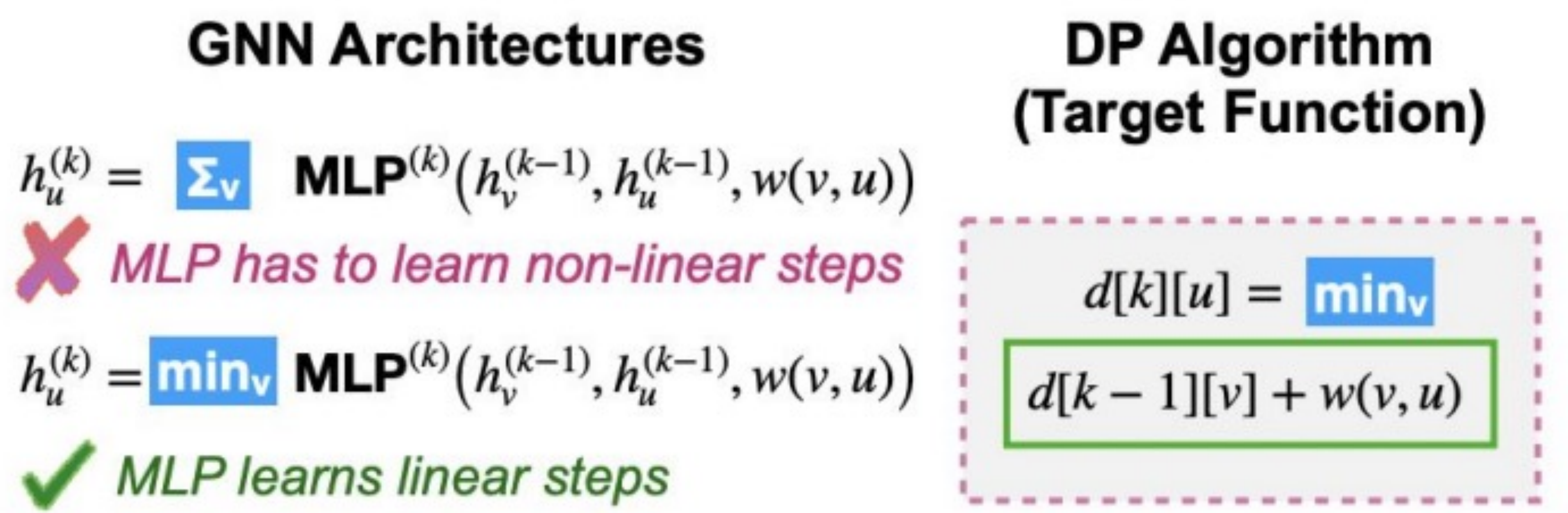
How GNNs extrapolate



Max degree and shortest paths are DP tasks

Yes!

- Does linear algorithmic alignment lead to extrapolation?



Concusion

- Neural networks can be viewed as programs, or algorithms
- Different neural network architectures are better suited to learning different algorithms
- **Graph neural networks are dynamic programs**
- **Algorithmic alignment:** make the computations steps of the neural net closely match the computational steps of the target algorithm
  - **Learn quicker, extrapolate better**