End to End Learning and Optimization on Graphs

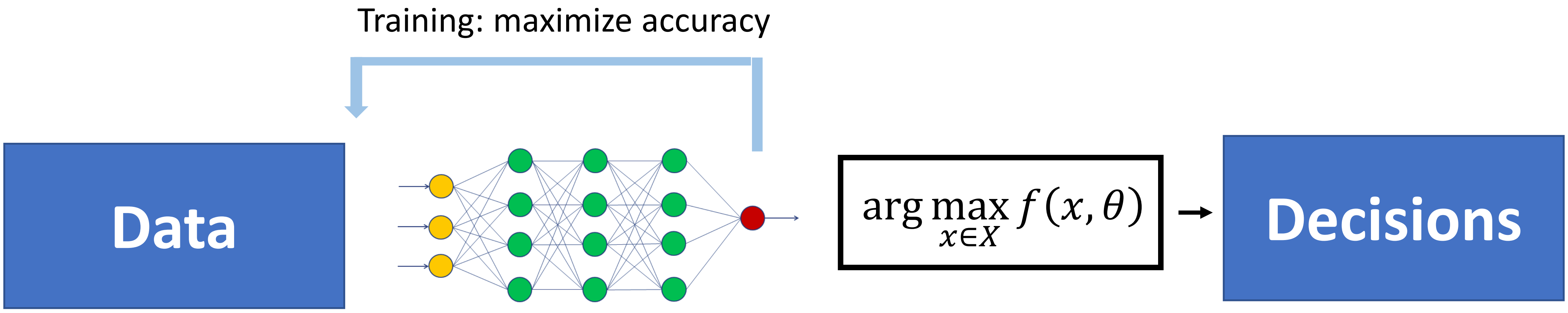
Bryan Wilder[1], Eric Ewing[2], Bistra Dilkina[2], Milind Tambe[1]
[1]Harvard University
[2]University of Southern California

To appear at NeurIPS 2019

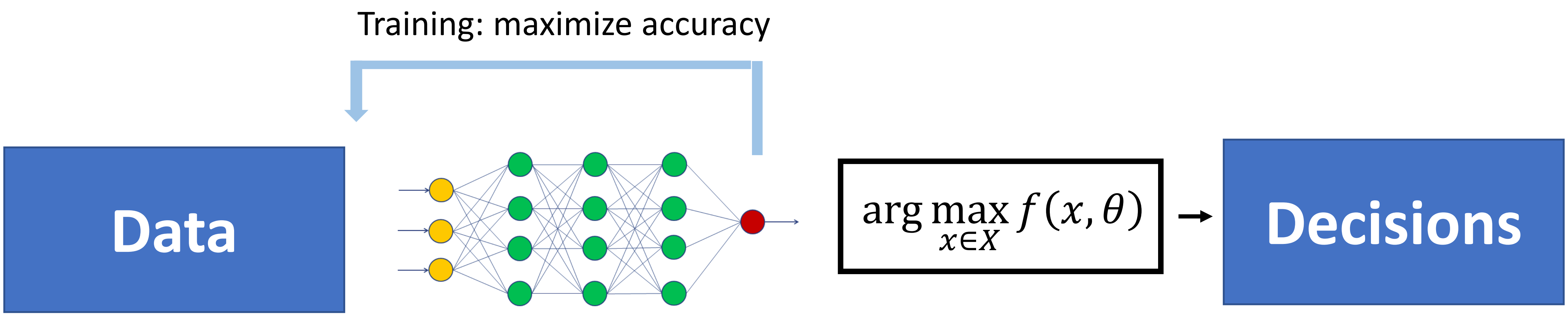
Data

???

Decisions

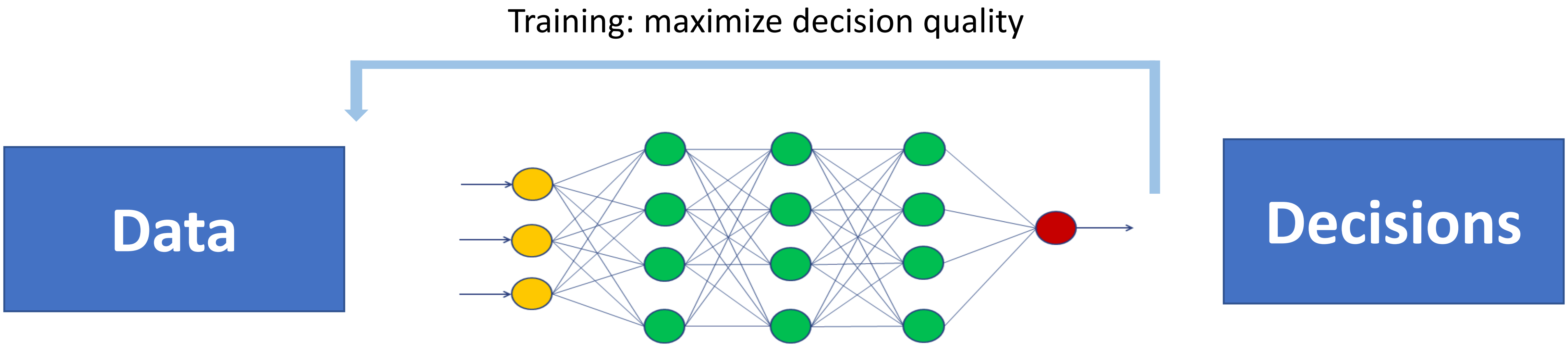


Standard two stage: predict then optimize

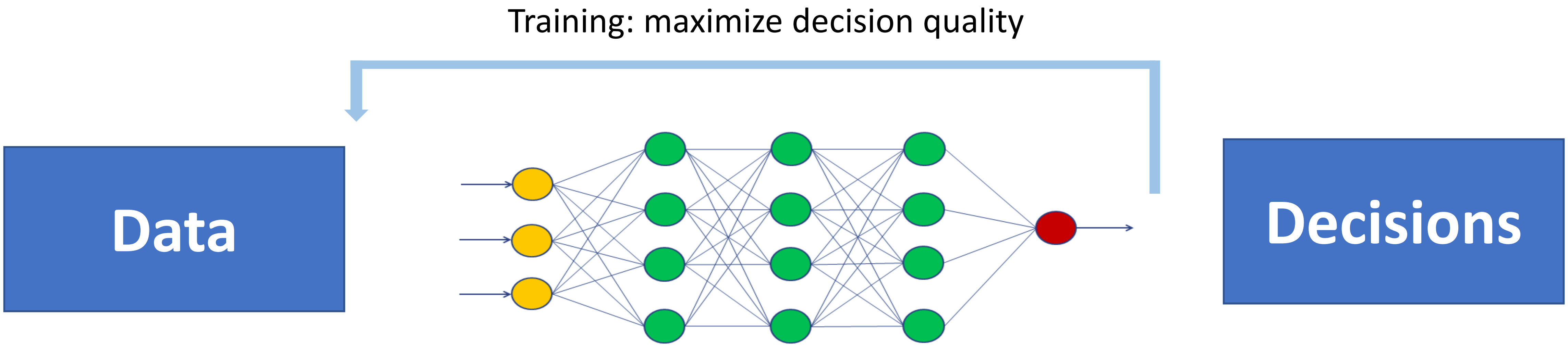


Standard two stage: predict then optimize

Challenge: misalignment between "accuracy" and decision quality

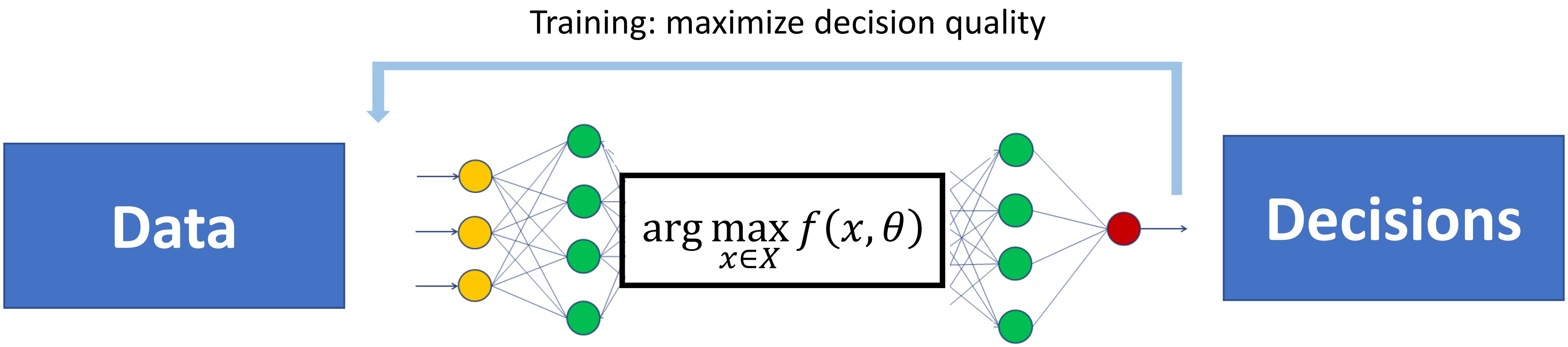


Pure end to end

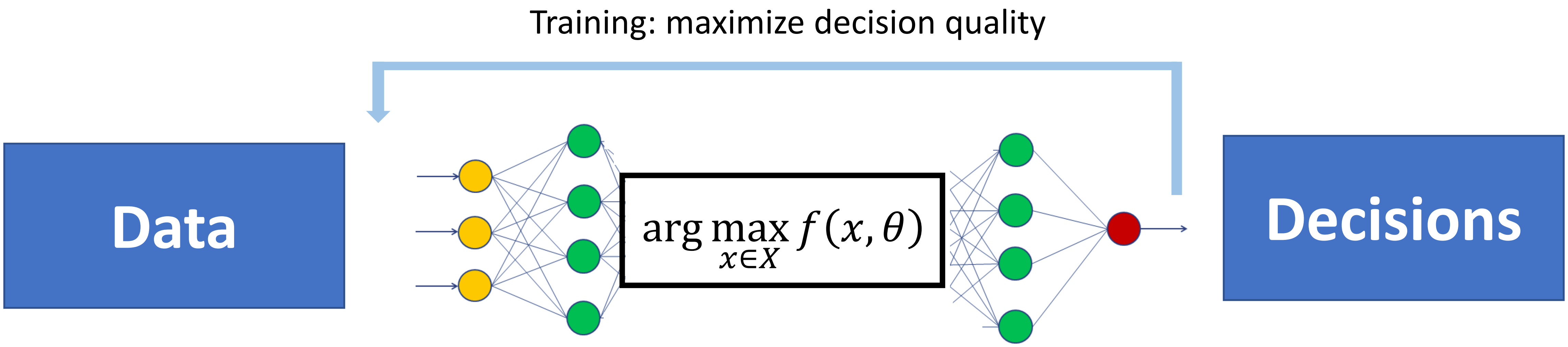


Pure end to end

Challenge: optimization is hard



Decision-focused learning: differential optimization during training

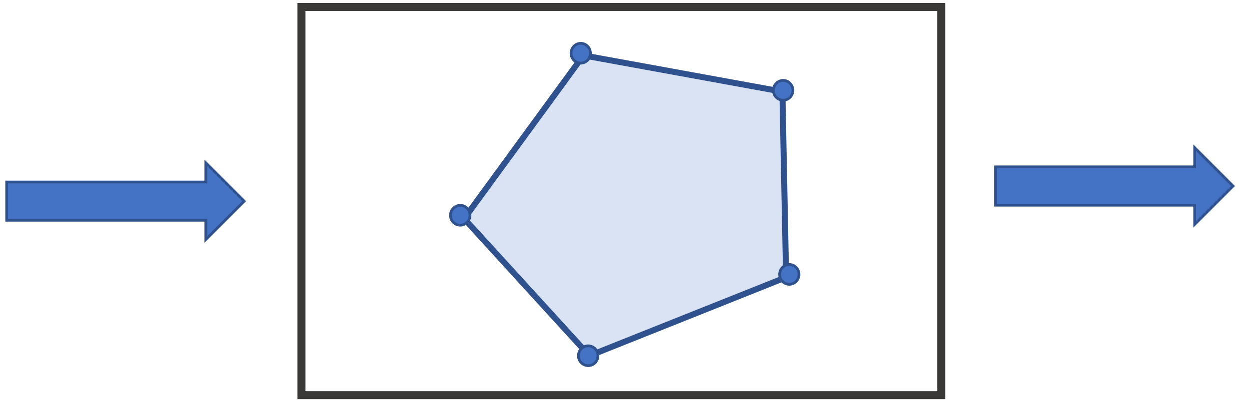


Decision-focused learning: differential optimization during training

Challenge: how to make optimization differentiable?

Relax + differentiate

Forward pass: run a solver



Backward pass: sensitivity analysis via KKT conditions

Relax + differentiate

- Convex QPs *[Amos and Kolter 2018, Donti et al 2018]*
- Linear and submodular programs *[Wilder, Dilkina, Tambe 2019]*
- MAXSAT (via SDP relaxation) *[Wang, Donti, Wilder, Kolter 2019]*
- MIPs *[Ferber, Wilder, Dilkina, Tambe 2019]*
  - Monday @ 11am, Room 612

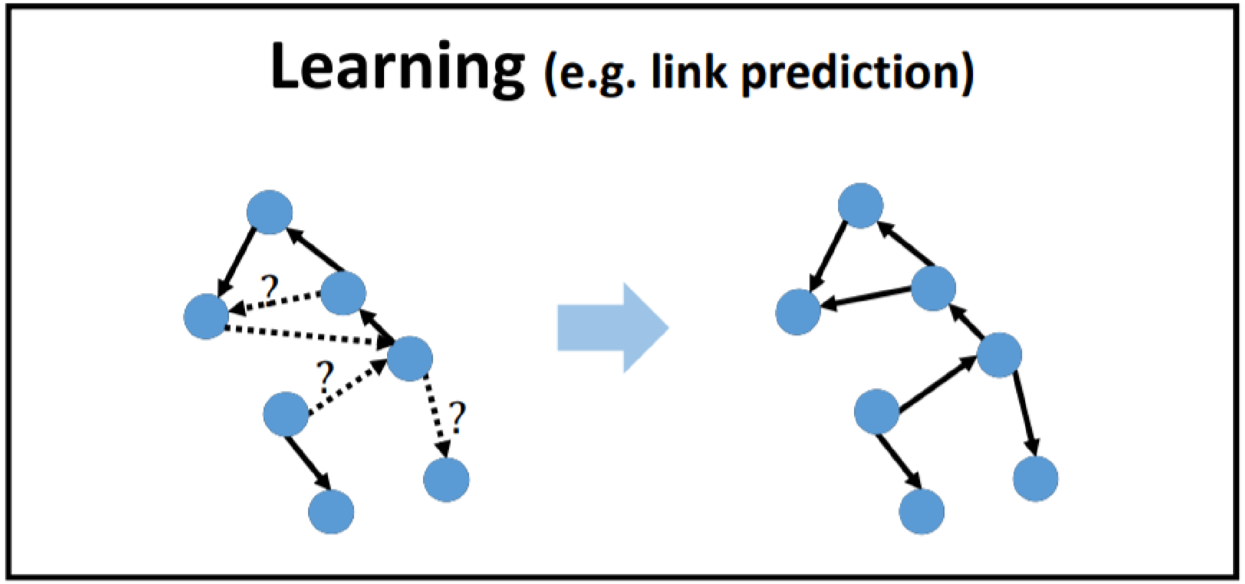
What’s wrong with relaxations?

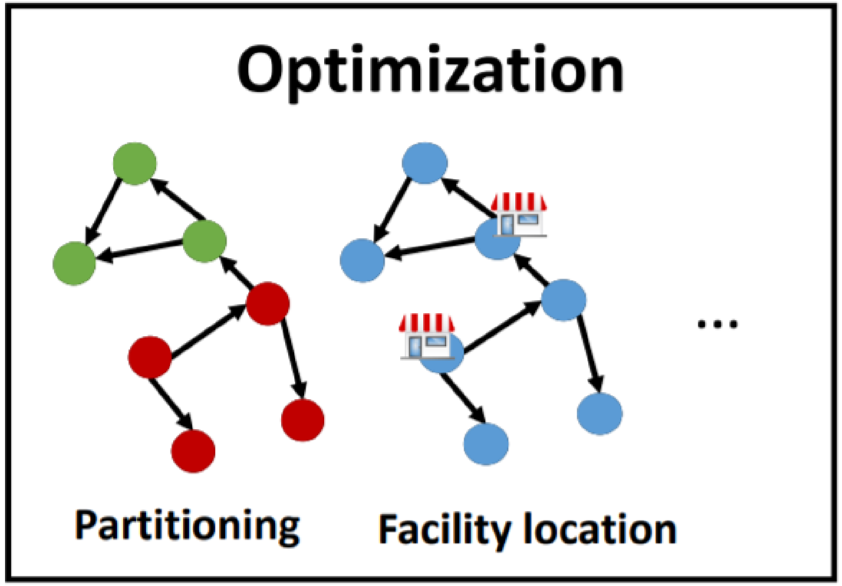
- Some problems don’t have good ones
- Slow to solve continuous optimization problem
- Slower to backprop through – $O(n^3)$

This work

- Alternative: include solver for a **simpler** proxy problem
- Learn a **representation** that maps hard problem to simple one
- Instantiate this idea for a class of graph optimization problems

Graph learning + optimization





Problem classes

- Partition the nodes into K disjoint groups
  - Community detection, maxcut, …
- Select a subset of K nodes
  - Facility location, influence maximization, vaccination, …
- Methods of choice are often combinatorial/discrete

Approach

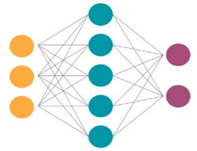
- Observation: grouping nodes into communities is a good heuristic
  - Partitioning: correspond to well-connected subgroups
  - Facility location: put one facility in each community
- Observation: graph learning approaches already embed into R^n

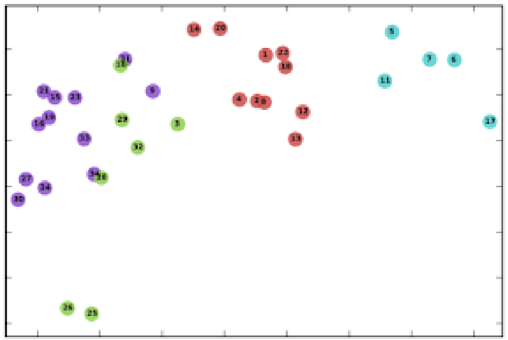
Approach

1. Start with clustering algorithm (in R^n)
   - Can (approximately) differentiate very quickly
2. Train embeddings (representation) to solve the particular problem
   - Automatically learning a good continuous relaxation!

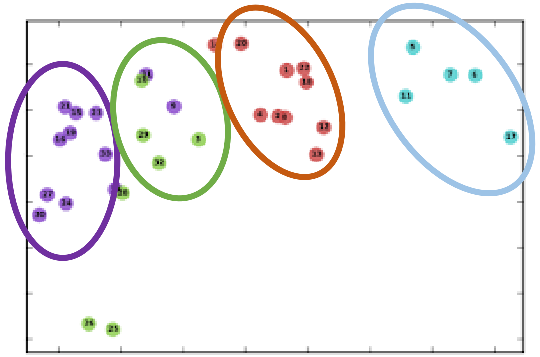
ClusterNet Approach

Node embedding (GCN)





K-means clustering



Locate 1 facility in each community

Differentiable K-means

Forward pass

$$\mu_k = \frac{\sum_j r_{jk} x_j}{\sum_j r_{jk}}$$

Update cluster centers

$$r_{jk} = \frac{\exp(-\beta||x_j - \mu_k||)}{\sum_\ell \exp(-\beta||x_j - \mu_\ell||)}$$

Softmax update to node assignments

Differentiable K-means

Backward pass

- Option 1: differentiate through the fixed-point condition

$$\mu^t = \mu^{t+1}$$

  - Prohibitively slow, memory-intensive

Differentiable K-means

Backward pass

- Option 1: differentiate through the fixed-point condition

$$\mu^t = \mu^{t+1}$$

  - Prohibitively slow, memory-intensive
- Option 2: unroll the entire series of updates
  - Cost scales with # iterations
  - Have to stick to differentiable operations

Differentiable K-means

Backward pass

- Option 1: differentiate through the fixed-point condition

$$\mu^t = \mu^{t+1}$$

  - Prohibitively slow, memory-intensive
- Option 2: unroll the entire series of updates
  - Cost scales with # iterations
  - Have to stick to differentiable operations
- **Option 3: get the solution, then unroll one update**
  - Do anything to solve the forward pass
  - Linear time/memory, implemented in vanilla pytorch

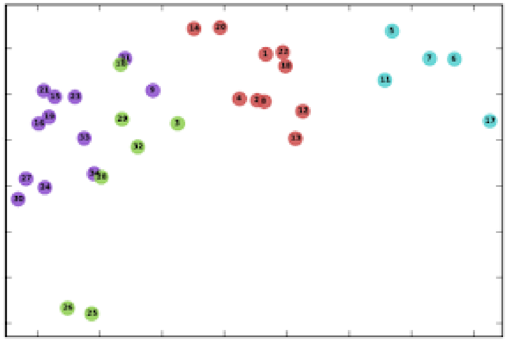
Differentiable K-means

Theorem [informal]: provided the clusters are sufficiently balanced and well-separated, the Option 3 approximate gradients converge exponentially quickly to the true ones.

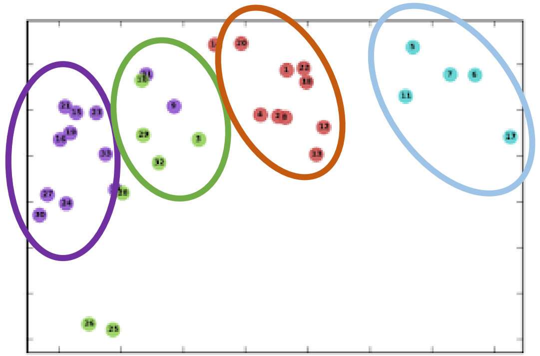
Idea: show that this corresponds to approximating a particular term in the analytical fixed-point gradients.

ClusterNet Approach

GCN node embeddings

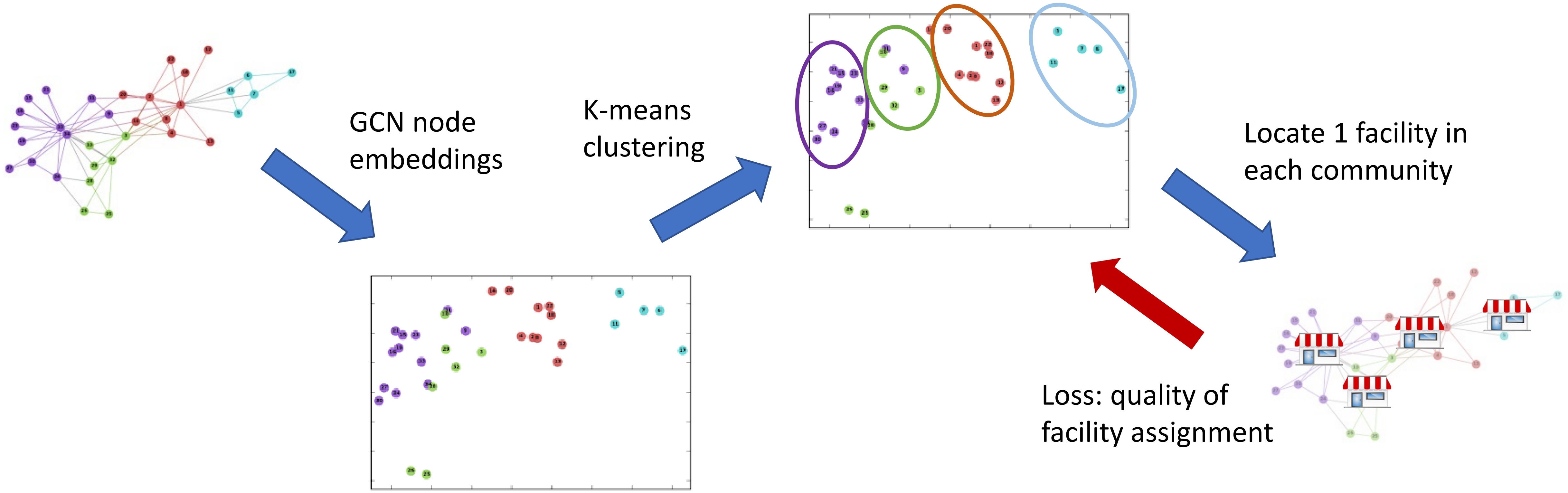


K-means clustering



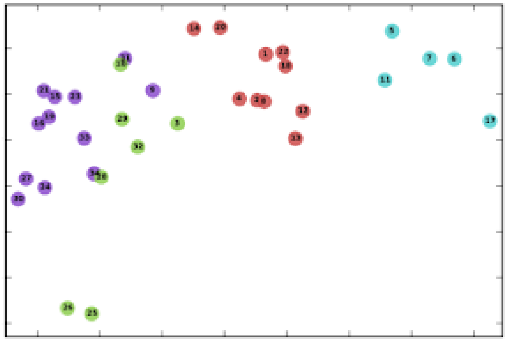
Locate 1 facility in each community

ClusterNet Approach

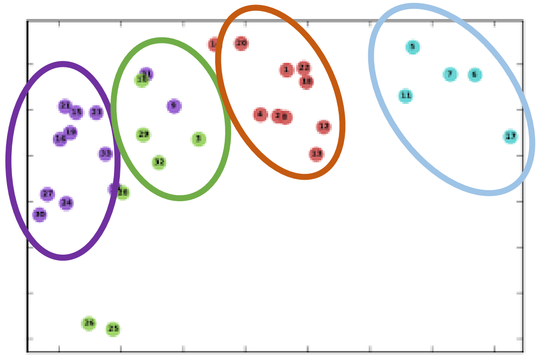


ClusterNet Approach

GCN node
embeddings



K-means
clustering

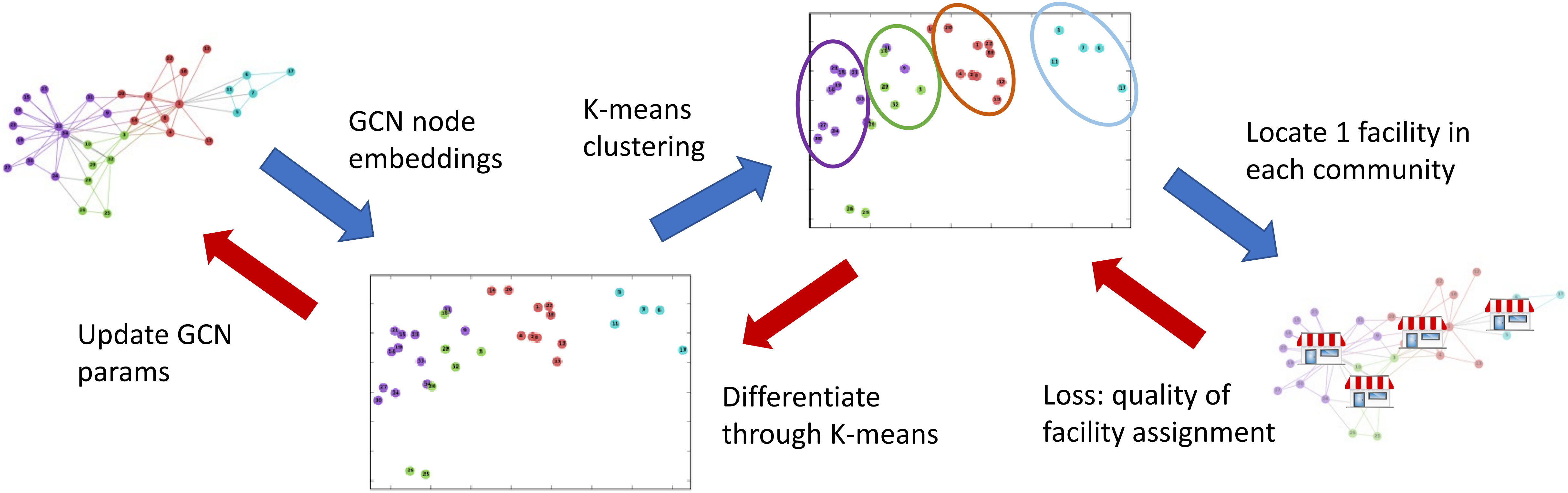


Locate 1 facility in
each community

Differentiate
through K-means

Loss: quality of
facility assignment

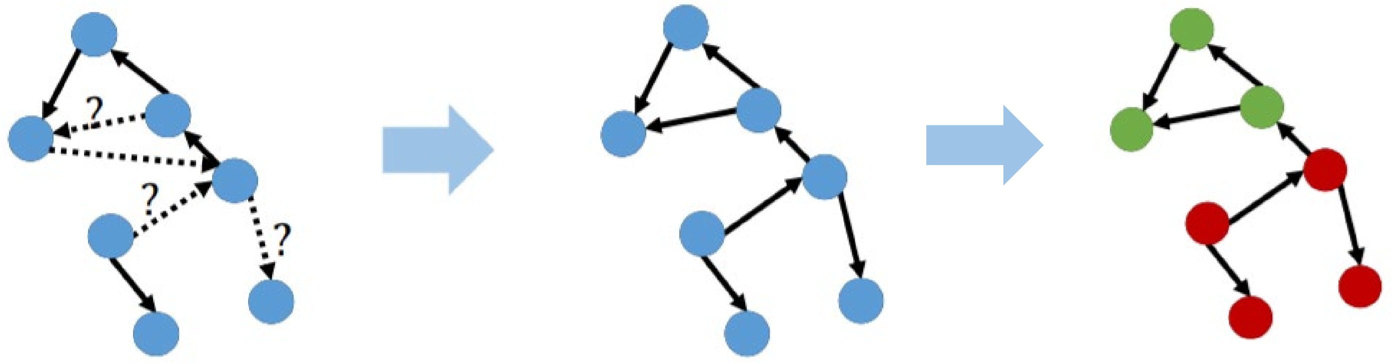
ClusterNet Approach



Experiments

- **Learning problem**: link prediction
- **Optimization:** community detection and facility location problems
- Train **GCNs** as predictive component

Example: community detection



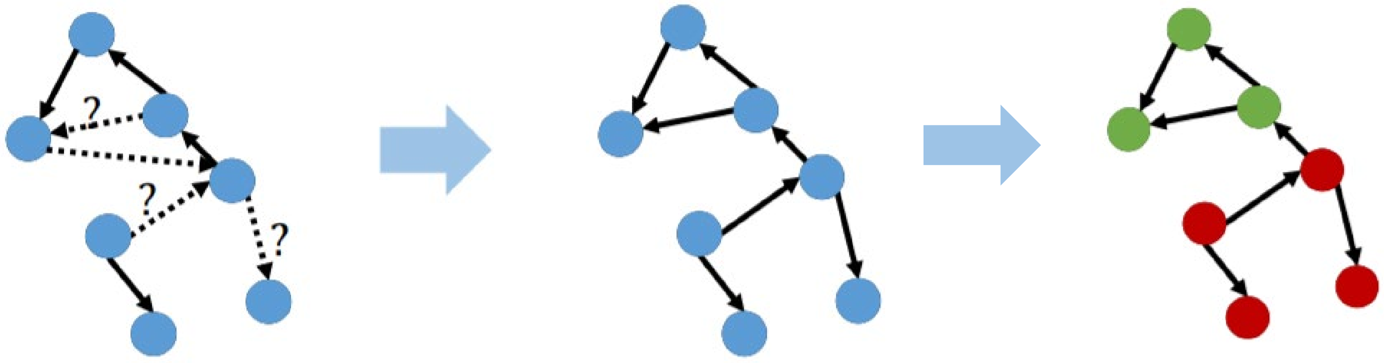
Observe partial graph | Predict unseen edges | Find communities

$$\max_r \frac{1}{2m} \sum_{u,v \in V} \sum_{k=1}^{K} \left[A_{u,v} - \frac{d_u d_v}{2m} \right] r_{uk} r_{vk}$$

$$r_{uk} \in \{0,1\} \quad \forall u \in V, k = 1 \ldots K$$

$$\sum_{k=1}^{K} r_{uk} = 1 \quad \forall u \in V$$

Example: community detection



Observe partial graph | Predict unseen edges | Find communities

$$\max_{r} \frac{1}{2m} \sum_{u,v \in V} \sum_{k=1}^{K} \left[A_{u,v} - \frac{d_u d_v}{2m} \right] r_{uk} r_{vk}$$

$$r_{uk} \in \{0,1\} \quad \forall u \in V, k = 1 \ldots K$$

$$\sum_{k=1}^{K} r_{uk} = 1 \quad \forall u \in V$$

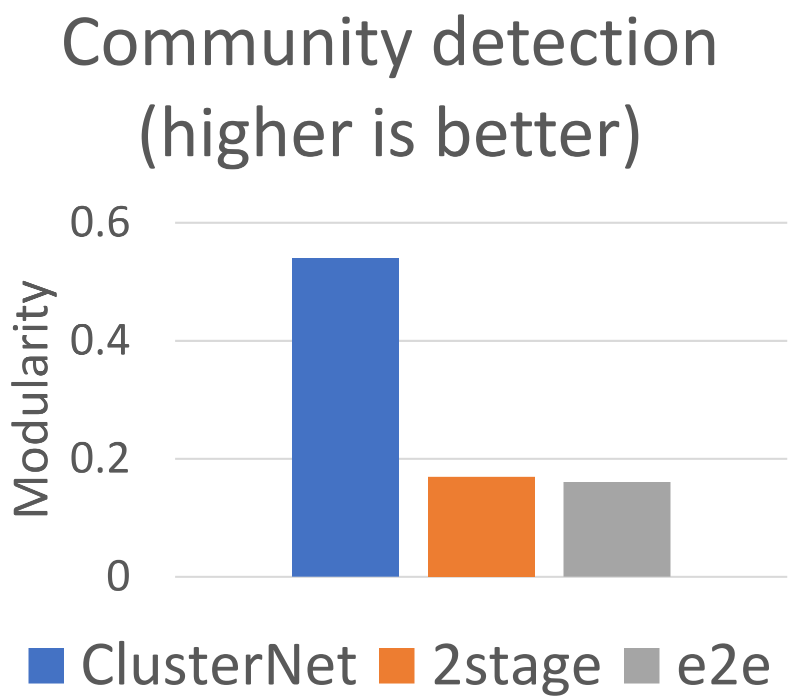
- Useful in scientific discovery (social groups, functional modules in biological networks)
- In applications, two-stage approach is common

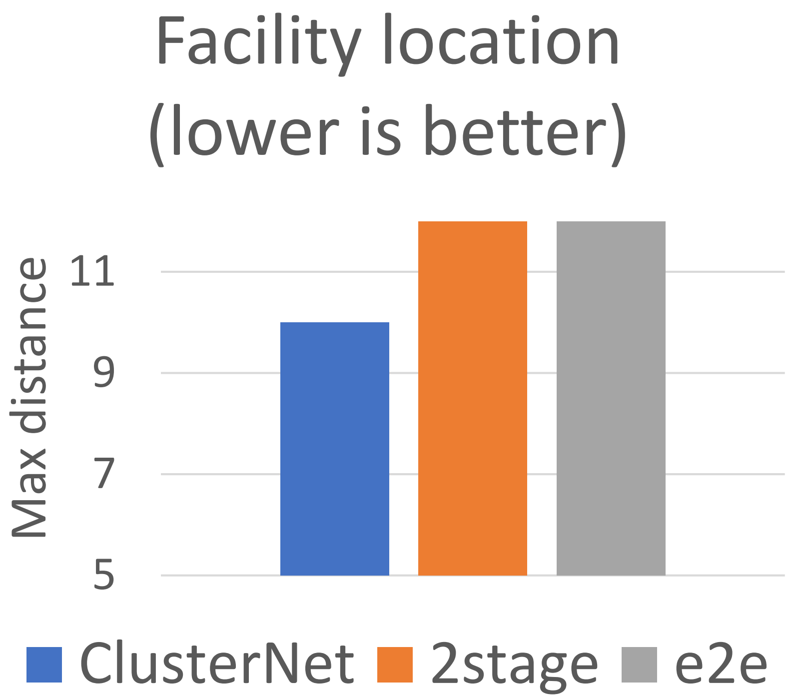
  [Yan & Gegory ’12, Burgess et al ‘16, Berlusconi et al ‘16, Tan et al ‘16, Bahulker et al ’18…]

Experiments

- **Learning problem**: link prediction
- **Optimization:** community detection and facility location problems
- Train **GCNs** as predictive component
- **Comparison**
  - Two stage: GCN + expert-designed algorithm (2Stage)
  - Pure end to end: Deep GCN to predict optimal solution (e2e)

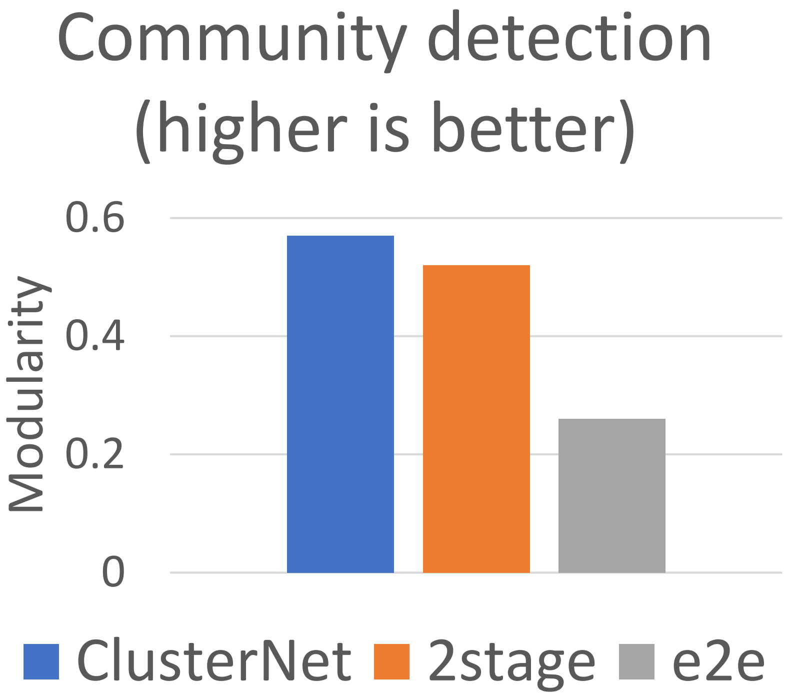
Results: single-graph link prediction

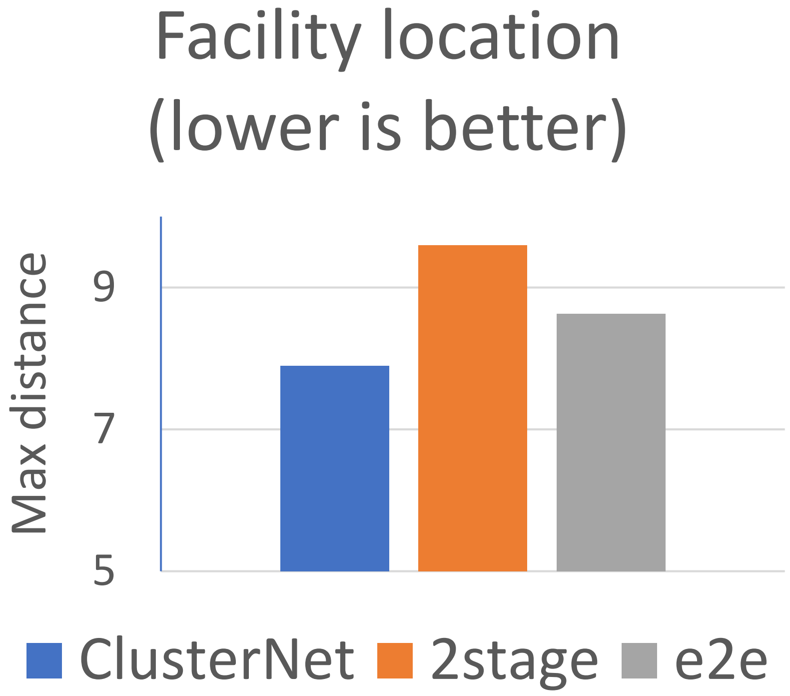




Representative example from **cora**, citeseer, protein interaction, facebook, adolescent health networks

Results: generalization across graphs





ClusterNet learns generalizable strategies for optimization!

Takeaways

- Good decisions require integrating learning and optimization
- Pure end-to-end methods miss out on useful structure
- Even simple optimization primitives provide good inductive bias

NeurIPS'19 paper, see bryanwilder.github.io

Code available at https://github.com/bwilder0/clusternet