Limits of Learning in Incomplete Networks

Timothy LaRock
larock.t@husky.neu.edu

In collaboration with


Timothy Sakharov


Sahely Bhadra


Tina Eliassi-Rad

Northeastern University
Network Science Institute

Supported by NSF CNS-1314603 & NSF IIS-1741197

Background: Incomplete Networks

- Network data is often incomplete
- Acquiring more data is often expensive and/or hard
- Research question: Given a networked dataset and limited resources to collect more data, how can you get the most bang for your buck?

Two general approaches to network completion

Don’t collect more data

Collect more data

Two general approaches to network completion

Don’t collect more data

Assume a network model

Combine network model with incomplete data to get a model of the network structure

Infer missing data from this model

- [Kim et al. 2011]
- [Chen et al. 2018]

Collect more data

Two general approaches to network completion

Don’t collect more data

Assume a network model

Combine network model with incomplete data to get a model of the network structure

Infer missing data from this model

- [Kim et al. 2011]
- [Chen et al. 2018]

Collect more data

Estimate Statistics from partially observed network

- [Soundarajan et al. 2015]
- [Soundarajan et al. 2016]

Utilize an explore-exploit approach

- [Pfeiffer III et al. 2014]
- [Soundarajan et al. 2017]
- [Murai et al. 2018]
- [Madhawa et al. 2018]
- This work!

Our solution: Network Online Learning (NOL)

- Research Question: Given a networked dataset and limited resources to collect more data, how can you get the most bang for your buck?

- Learn to grow an incomplete network through sequential, optimal queries (to some API)
- Agnostic to both *data distributions* and *sampling method*
- Interpretable features that are computable online

Assumptions

We assume...

We know the API access model (complete vs incomplete queries).

The underlying network is static (probing the same node twice gives no new information).

We do not assume...

A model of the underlying graph.

How the initial sample was collected.

Network Online Learning (NOL)

Inputs:

- $\hat{G}_0$: Incomplete network
- *b* : probing budget
- *r* : reward function

Probe $\hat{G}_0$
b times to maximize cumulative reward

Output:

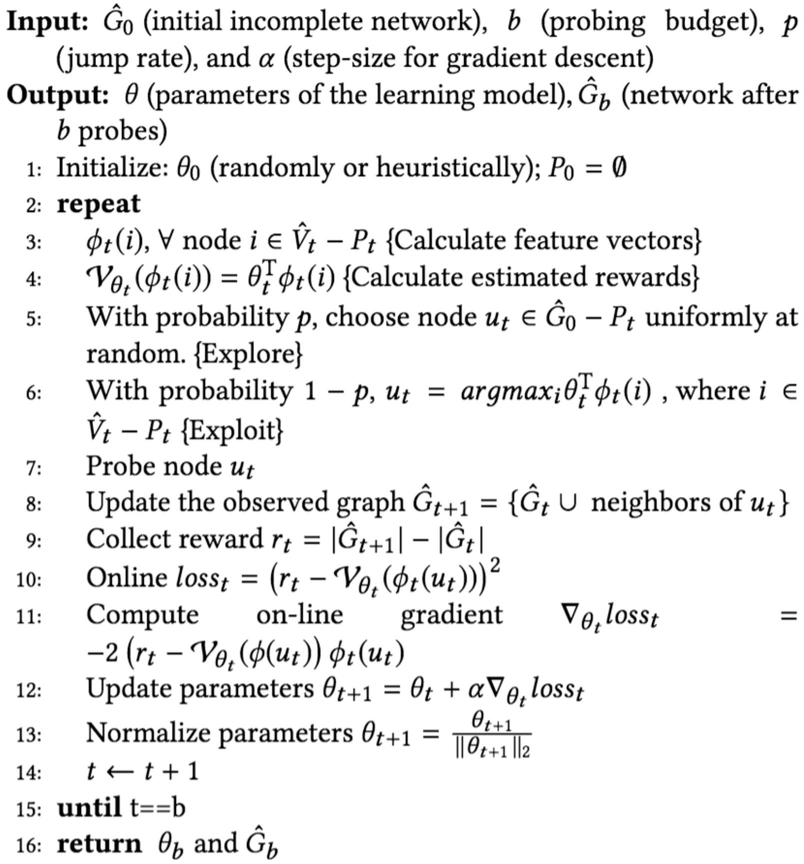
Network after *b* probes

$$\hat{G}_b \approx G$$

Example reward function:

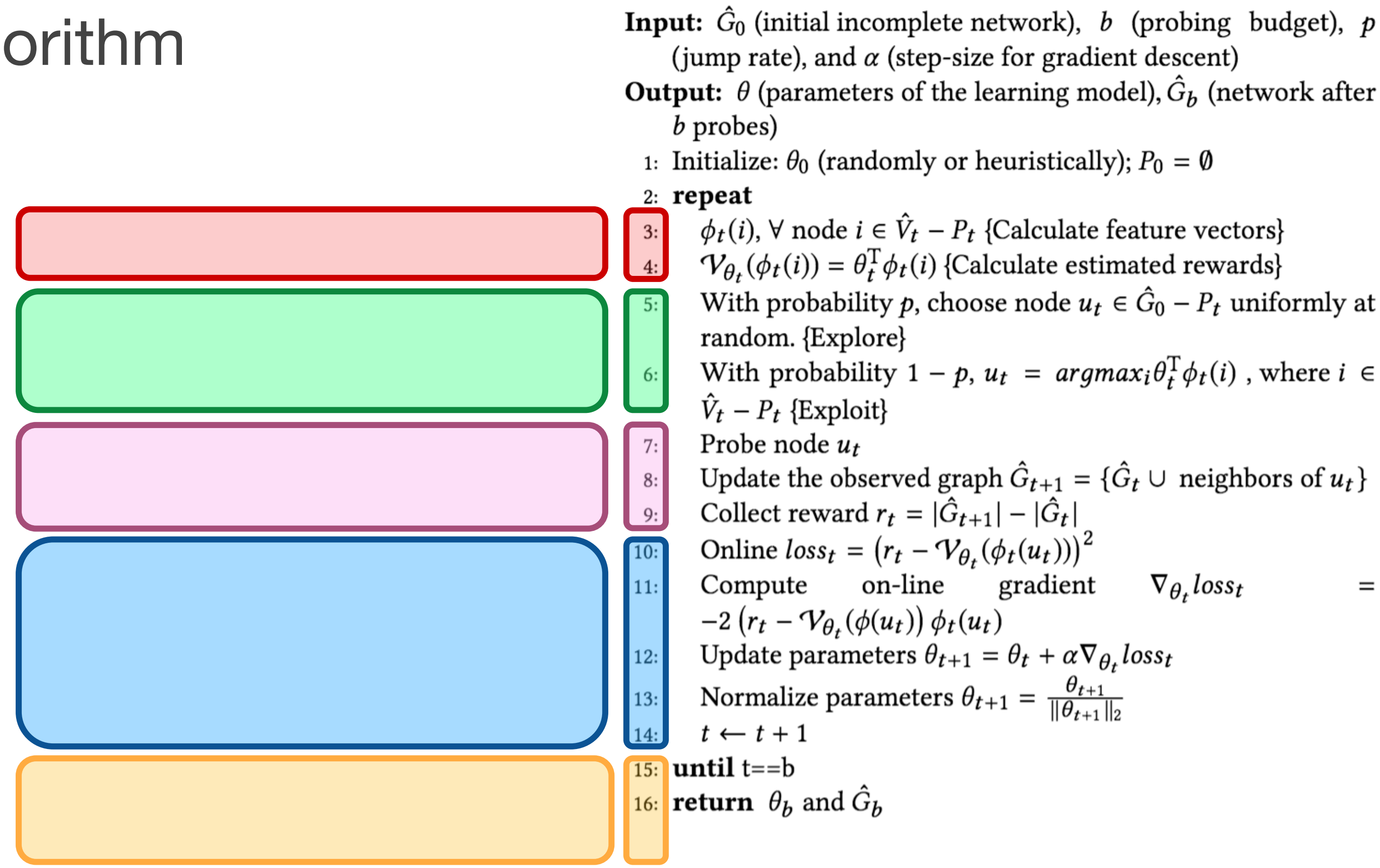
- *number of new nodes observed*

NOL algorithm



```
Input: $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$ (jump rate), and $\alpha$ (step-size for gradient descent)
Output: $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after $b$ probes)
Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
repeat
    $\phi_t(i), \forall$ node $i \in \hat{V}_t - P_t$ {Calculate feature vectors}
    $\mathcal{V}_{\theta_t}(\phi_t(i)) = \theta_t^{\mathrm{T}} \phi_t(i)$ {Calculate estimated rewards}
    With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random. {Explore}
    With probability $1 - p$, $u_t = argmax_i \theta_t^{\mathrm{T}} \phi_t(i)$ , where $i \in \hat{V}_t - P_t$ {Exploit}
    Probe node $u_t$
    Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup$ neighbors of $u_t\}$
    Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
    Online $loss_t = \left(r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t))\right)^2$
    Compute on-line gradient $\nabla_{\theta_t} loss_t = -2\left(r_t - \mathcal{V}_{\theta_t}(\phi(u_t)\right) \phi_t(u_t)\right)$
    Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} loss_t$
    Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{\|\theta_{t+1}\|_2}$
    $t \leftarrow t + 1$
until t==b
return $\theta_b$ and $\hat{G}_b$
```

NOL algorithm



```
**Input:** $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$ (jump rate), and $\alpha$ (step-size for gradient descent)
**Output:** $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after $b$ probes)
Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
**repeat**
    $\phi_t(i), \forall$ node $i \in \hat{V}_t - P_t$ {Calculate feature vectors}
    $\mathcal{V}_{\theta_t}(\phi_t(i)) = \theta_t^{\mathrm{T}} \phi_t(i)$ {Calculate estimated rewards}
    With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random. {Explore}
    With probability $1 - p$, $u_t = argmax_i \theta_t^{\mathrm{T}} \phi_t(i)$, where $i \in \hat{V}_t - P_t$ {Exploit}
    Probe node $u_t$
    Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup$ neighbors of $u_t\}$
    Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
    Online $loss_t = \left(r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t))\right)^2$
    Compute on-line gradient $\nabla_{\theta_t} loss_t = -2\left(r_t - \mathcal{V}_{\theta_t}(\phi(u_t)\right)\phi_t(u_t)$
    Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} loss_t$
    Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{\|\theta_{t+1}\|_2}$
    $t \leftarrow t + 1$
**until** t==b
**return** $\theta_b$ and $\hat{G}_b$
```

Northeastern University
Network Science Institute

NOL algorithm

Observe the current state

```
Input: $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$ (jump rate), and $\alpha$ (step-size for gradient descent)
Output: $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after $b$ probes)
Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
repeat
    $\phi_t(i), \forall$ node $i \in \hat{V}_t - P_t$ {Calculate feature vectors}
    $\mathcal{V}_{\theta_t}(\phi_t(i)) = \theta_t^{\mathrm{T}} \phi_t(i)$ {Calculate estimated rewards}
    With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random. {Explore}
    With probability $1 - p$, $u_t = argmax_i \theta_t^{\mathrm{T}} \phi_t(i)$ , where $i \in \hat{V}_t - P_t$ {Exploit}
    Probe node $u_t$
    Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup$ neighbors of $u_t\}$
    Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
    Online $loss_t = (r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t)))^2$
    Compute on-line gradient $\nabla_{\theta_t} loss_t = -2 \left(r_t - \mathcal{V}_{\theta_t}(\phi(u_t))\right) \phi_t(u_t)$
    Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} loss_t$
    Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{||\theta_{t+1}||_2}$
    $t \leftarrow t + 1$
until t==b
return $\theta_b$ and $\hat{G}_b$
```

NOL algorithm

Input: $\hat{G}_0$ (initial incomplete network), b (probing budget), p (jump rate), and α (step-size for gradient descent)

Output: θ (parameters of the learning model), $\hat{G}_b$ (network after b probes)

```
Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
repeat
    $\phi_t(i), \forall$ node $i \in \hat{V}_t - P_t$ {Calculate feature vectors}
    $\mathcal{V}_{\theta_t}(\phi_t(i)) = \theta_t^T \phi_t(i)$ {Calculate estimated rewards}
    With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random. {Explore}
    With probability $1 - p$, $u_t = argmax_i \theta_t^T \phi_t(i)$ , where $i \in \hat{V}_t - P_t$ {Exploit}
    ode $u_t$
    Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup$ neighbors of $u_t\}$
    Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
    Online $loss_t = (r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t)))^2$
    Compute on-line gradient $\nabla_{\theta_t} loss_t = -2 (r_t - \mathcal{V}_{\theta_t}(\phi(u_t)) \phi_t(u_t)$
    Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} loss_t$
    Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{||\theta_{t+1}||_2}$
    $t \leftarrow t + 1$
until t==b
return $\theta_b$ and $\hat{G}_b$
```

Observe the current state

Regression weights

Feature vector of node i

Northeastern University
Network Science Institute

netsi

NOL algorithm

Observe the current state

```
Input: $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$ (jump rate), and $\alpha$ (step-size for gradient descent)
Output: $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after $b$ probes)
Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
repeat
    $\phi_t(i), \forall$ node $i \in \hat{V}_t - P_t$ {Calculate feature vectors}
    $\mathcal{V}_{\theta_t}(\phi_t(i)) = \theta_t^{\mathrm{T}} \phi_t(i)$ {Calculate estimated rewards}
    With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random. {Explore}
    With probability $1 - p$, $u_t = argmax_i \theta_t^{\mathrm{T}} \phi_t(i)$ , where $i \in \hat{V}_t - P_t$ {Exploit}
    Probe node $u_t$
    Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup$ neighbors of $u_t\}$
    Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
    Online $loss_t = (r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t)))^2$
    Compute on-line gradient $\nabla_{\theta_t} loss_t = -2\,(r_t - \mathcal{V}_{\theta_t}(\phi(u_t)))\,\phi_t(u_t)$
    Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} loss_t$
    Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{||\theta_{t+1}||_2}$
    $t \leftarrow t + 1$
until t==b
return $\theta_b$ and $\hat{G}_b$
```

NOL algorithm

Observe the current state

Choose the next action

```
Input: $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$ (jump rate), and $\alpha$ (step-size for gradient descent)
Output: $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after $b$ probes)
Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
repeat
    $\phi_t(i), \forall$ node $i \in \hat{V}_t - P_t$ {Calculate feature vectors}
    $\mathcal{V}_{\theta_t}(\phi_t(i)) = \theta_t^{\mathrm{T}} \phi_t(i)$ {Calculate estimated rewards}
    With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random. {Explore}
    With probability $1 - p$, $u_t = argmax_i \theta_t^{\mathrm{T}} \phi_t(i)$ , where $i \in \hat{V}_t - P_t$ {Exploit}
    Probe node $u_t$
    Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup$ neighbors of $u_t\}$
    Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
    Online $loss_t = \left(r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t))\right)^2$
    Compute on-line gradient $\nabla_{\theta_t} loss_t = -2\left(r_t - \mathcal{V}_{\theta_t}(\phi(u_t))\right)\phi_t(u_t)$
    Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} loss_t$
    Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{\|\theta_{t+1}\|_2}$
    $t \leftarrow t + 1$
until t==b
return $\theta_b$ and $\hat{G}_b$
```

NOL algorithm

Observe the current state

Choose the next action

Take action, update network, collect reward

```
Input: $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$ (jump rate), and $\alpha$ (step-size for gradient descent)
Output: $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after $b$ probes)
Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
repeat
    $\phi_t(i), \forall$ node $i \in \hat{V}_t - P_t$ {Calculate feature vectors}
    $\mathcal{V}_{\theta_t}(\phi_t(i)) = \theta_t^{\mathrm{T}} \phi_t(i)$ {Calculate estimated rewards}
    With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random. {Explore}
    With probability $1 - p$, $u_t = argmax_i \theta_t^{\mathrm{T}} \phi_t(i)$ , where $i \in \hat{V}_t - P_t$ {Exploit}
    Probe node $u_t$
    Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup$ neighbors of $u_t\}$
    Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
    Online $loss_t = \left(r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t))\right)^2$
    Compute on-line gradient $\nabla_{\theta_t} loss_t = -2\left(r_t - \mathcal{V}_{\theta_t}(\phi(u_t))\right) \phi_t(u_t)$
    Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} loss_t$
    Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{\|\theta_{t+1}\|_2}$
    $t \leftarrow t + 1$
until t==b
return $\theta_b$ and $\hat{G}_b$
```

Northeastern University
Network Science Institute

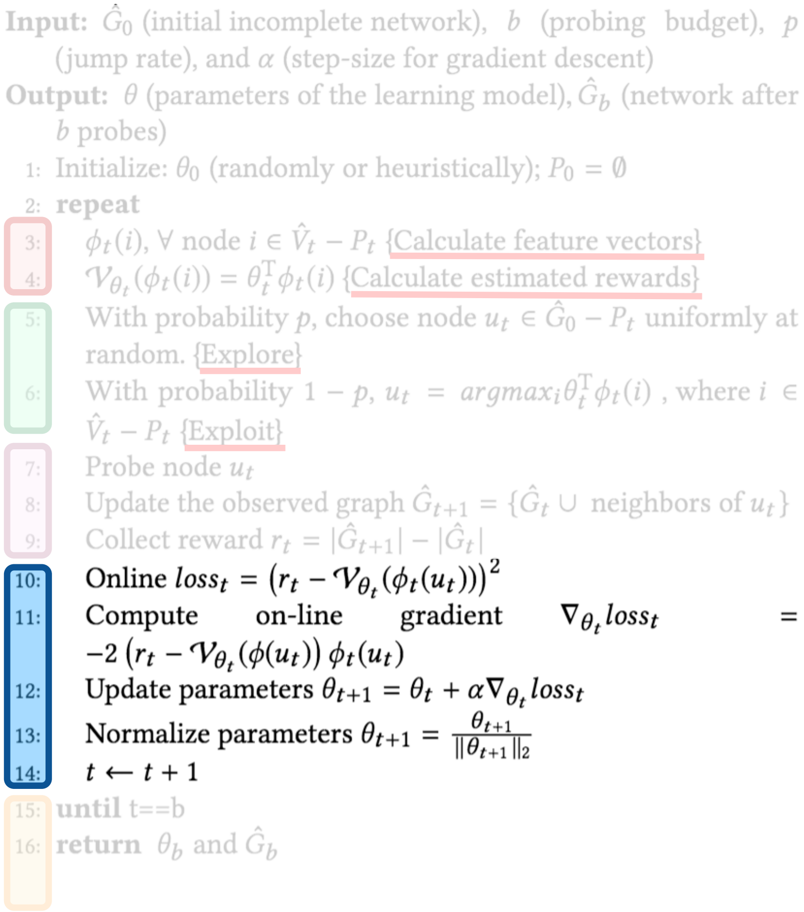
NOL algorithm

Observe the current state

Choose the next action

Take action, update network, collect reward

Update parameters



```
Input: $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$ (jump rate), and $\alpha$ (step-size for gradient descent)
Output: $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after $b$ probes)
Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
repeat
    $\phi_t(i), \forall$ node $i \in \hat{V}_t - P_t$ {Calculate feature vectors}
    $\mathcal{V}_{\theta_t}(\phi_t(i)) = \theta_t^{\mathrm{T}} \phi_t(i)$ {Calculate estimated rewards}
    With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random. {Explore}
    With probability $1 - p$, $u_t = argmax_i \theta_t^{\mathrm{T}} \phi_t(i)$ , where $i \in \hat{V}_t - P_t$ {Exploit}
    Probe node $u_t$
    Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup$ neighbors of $u_t\}$
    Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
    Online $loss_t = (r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t)))^2$
    Compute on-line gradient $\nabla_{\theta_t} loss_t = -2\left(r_t - \mathcal{V}_{\theta_t}(\phi(u_t))\right) \phi_t(u_t)$
    Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} loss_t$
    Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{\|\theta_{t+1}\|_2}$
    $t \leftarrow t + 1$
until t==b
return $\theta_b$ and $\hat{G}_b$
```

NOL algorithm

Observe the current state

Choose the next action

Take action, update network, collect reward

Update parameters

```
Input: $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$
    (jump rate), and $\alpha$ (step-size for gradient descent)
Output: $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after
    $b$ probes)
Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
repeat
    $\phi_t(i), \forall$ node $i \in \hat{V}_t - P_t$ {Calculate feature vectors}
    $\mathcal{V}_{\theta_t}(\phi_t(i)) = \theta_t^{\mathrm{T}}\phi_t(i)$ {Calculate estimated rewards}
    With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at
        random. {Explore}
    With probability $1 - p$, $u_t = argmax_i \theta_t^{\mathrm{T}}\phi_t(i)$ , where $i \in$
        $\hat{V}_t - P_t$ {Exploit}
    Probe node $u_t$
    Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup$ neighbors of $u_t\}$
    Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
    Online $loss_t = (r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t)))^2$
    Compute on-line gradient $\nabla_{\theta_t} loss_t$ =
        $-2(r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t)))\phi_t(u_t)$
    Update parameters $\theta_{t+1} = \theta_t - \alpha\nabla_{\theta_t} loss_t$
    Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{\|\theta_{t+1}\|_2}$
    $t \leftarrow t + 1$
until t==b
return $\theta_b$ and $\hat{G}_b$
```

Online linear regression following Strehl et al., NIPS 2008.

NOL algorithm

Observe the current state

Choose the next action

Take action, update network, collect reward

Update parameters

Repeat until budget depleted

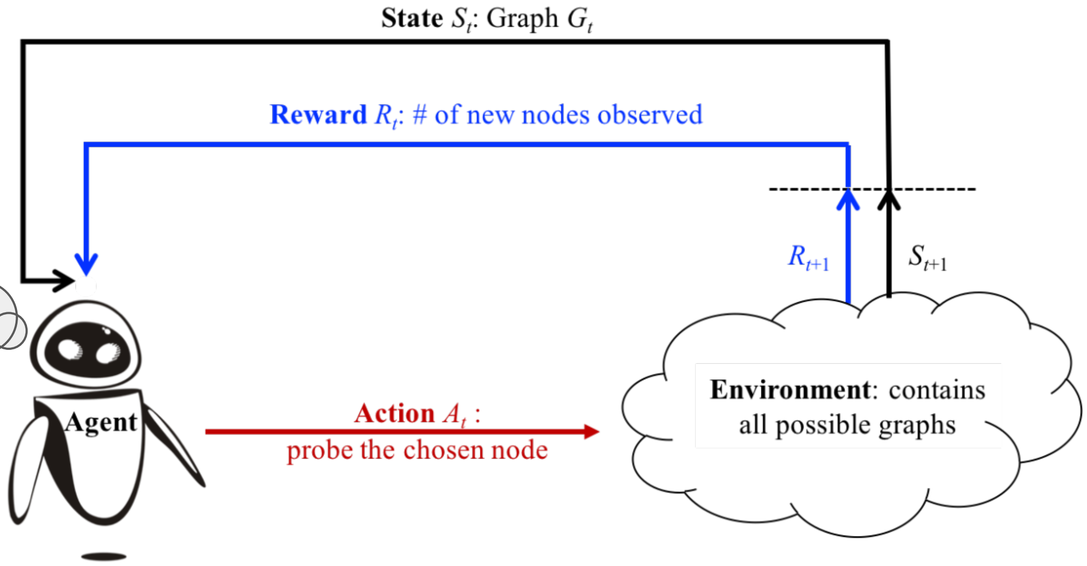
```
**Input:** $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$ (jump rate), and $\alpha$ (step-size for gradient descent)
**Output:** $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after $b$ probes)
Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
**repeat**
    $\phi_t(i), \forall$ node $i \in \hat{V}_t - P_t$ {Calculate feature vectors}
    $\mathcal{V}_{\theta_t}(\phi_t(i)) = \theta_t^T \phi_t(i)$ {Calculate estimated rewards}
    With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random. {Explore}
    With probability $1 - p$, $u_t = argmax_i \theta_t^T \phi_t(i)$ , where $i \in \hat{V}_t - P_t$ {Exploit}
    Probe node $u_t$
    Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup$ neighbors of $u_t\}$
    Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
    Online $loss_t = \left(r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t))\right)^2$
    Compute on-line gradient $\nabla_{\theta_t} loss_t = -2\left(r_t - \mathcal{V}_{\theta_t}(\phi(u_t))\right)\phi_t(u_t)$
    Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} loss_t$
    Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{\|\theta_{t+1}\|_2}$
    $t \leftarrow t + 1$
**until** t==b
**return** $\theta_b$ and $\hat{G}_b$
```

netsi

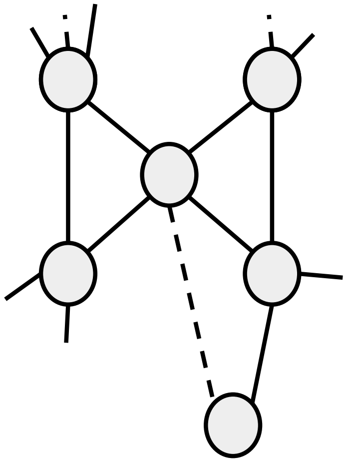
Northeastern University
Network Science Institute

NOL algorithm

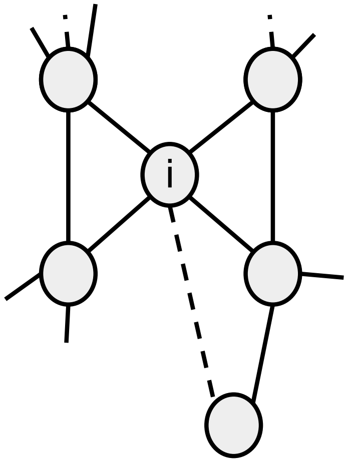
1. Observe the current state
2. Choose the next action
3. Take action, update network, collect reward
4. Update parameters
5. Repeat until budget depleted



Features

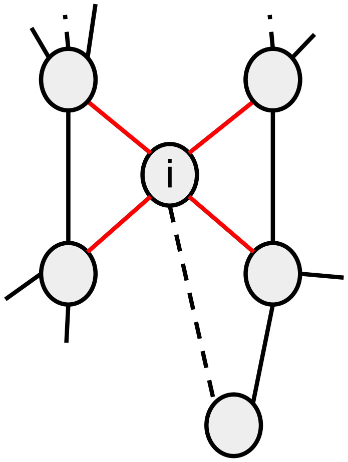


Features



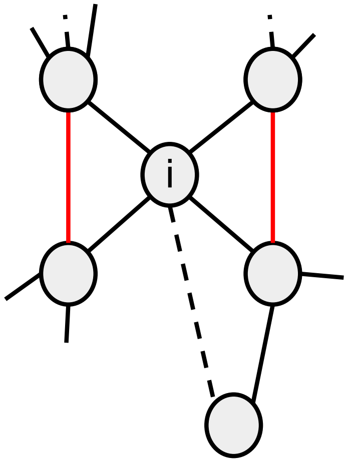
Feature: In-sample degree

$$\phi_{\mathbf{i,0}} = \hat{d}_i$$



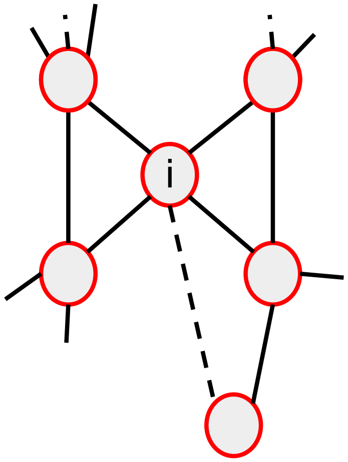
Feature: In-sample clustering coefficient

$$\phi_{\mathbf{i,1}} = \hat{cc}_i$$



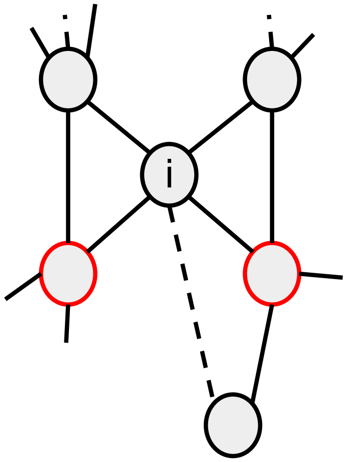
Feature: Normalized size of connected component

$$\phi_{\mathbf{i,2}} = CompSize_i$$



Feature: Fraction of probed neighbors

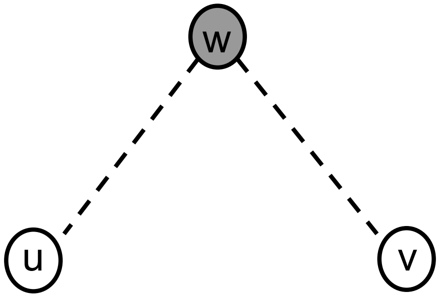
$$\phi_{\mathbf{i,3}} = ProbedNeighbors_i$$



Feature: Lost Reward

$$\phi_{\mathbf{i,4}} = LostReward_i$$

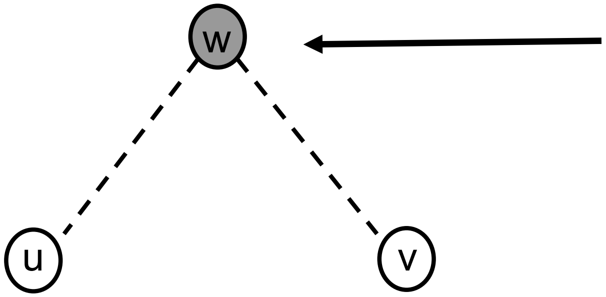
Key idea: The order in which we probe nodes can impact the reward they yield.



Feature: Lost Reward

$$\phi_{\mathbf{i,4}} = LostReward_i$$

Key idea: The order in which we probe nodes can impact the reward they yield.



Unobserved node (potential reward for u **or** v)

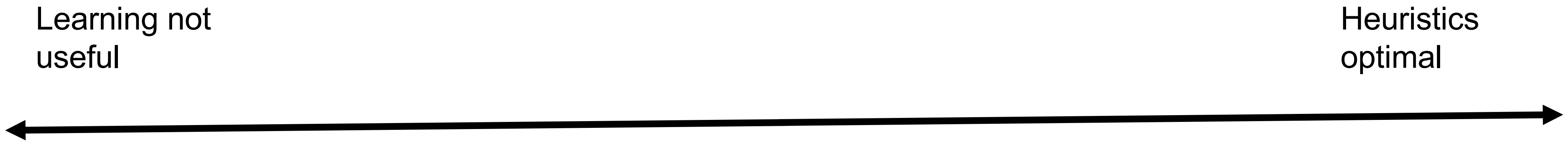
Partially observed nodes

Research Question: Given a networked dataset, and limited resources to collect more data, **how can you get the most bang for your buck?**

- Potential bang for your buck **depends on network structure!**

Heterogeneity in network properties creates a learning “spectrum”

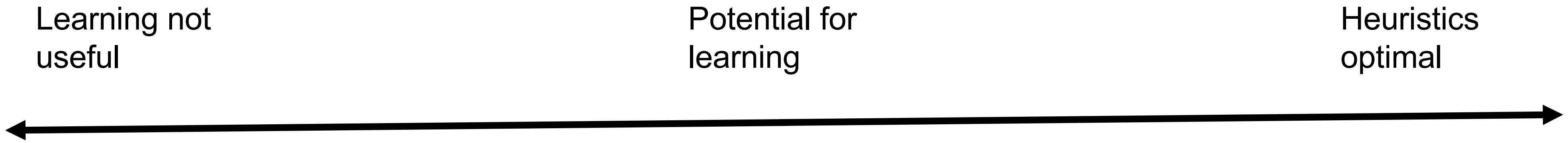
Heterogeneity in network properties creates a learning “spectrum”



Learning not
useful

Heuristics
optimal

Heterogeneity in network properties creates a learning “spectrum”

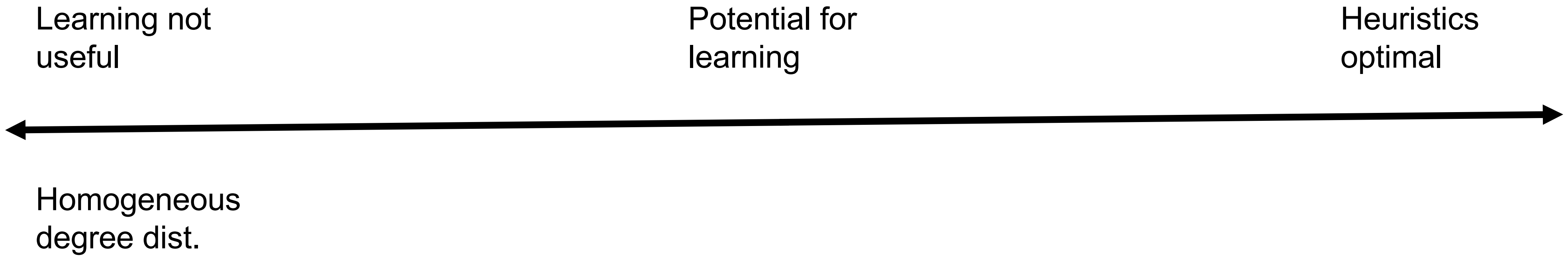


Learning not useful

Potential for learning

Heuristics optimal

Heterogeneity in network properties creates a learning “spectrum”



Learning not
useful

Potential for
learning

Heuristics
optimal

Homogeneous
degree dist.

Heterogeneity in network properties creates a learning “spectrum”

Learning not useful

Potential for learning

Heuristics optimal[1]

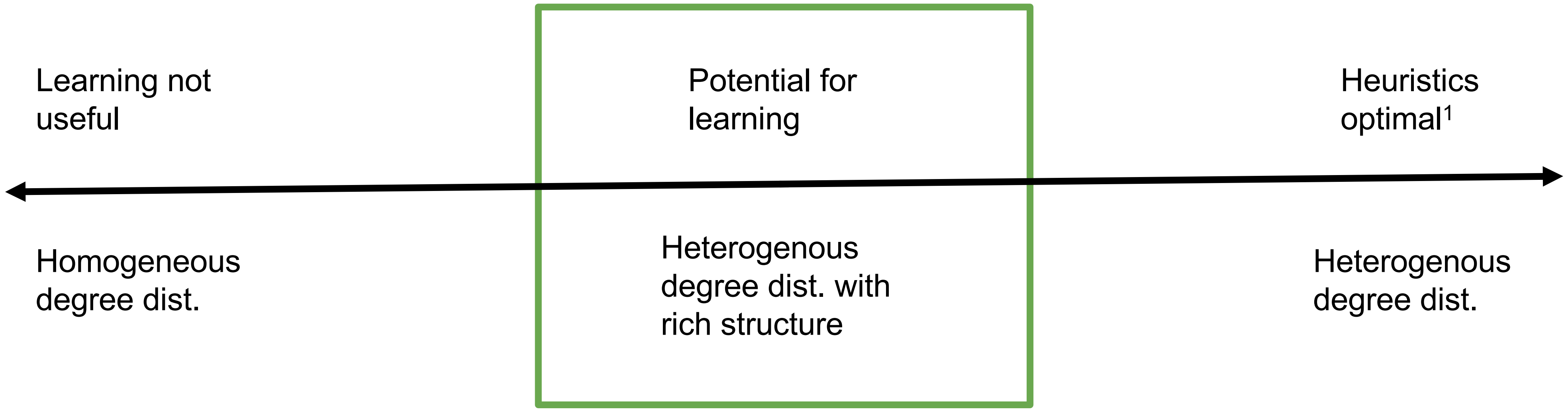
Homogeneous degree dist.

Heterogenous degree dist.

---

[1] Avrachenkov et al. *INFOCOM WKSHPS* (2014)

Heterogeneity in network properties creates a learning “spectrum”



Learning not useful

Potential for learning

Heuristics optimal[1]

Homogeneous degree dist.

Heterogenous degree dist. with rich structure

Heterogenous degree dist.

[1] Avrachenkov et al. *INFOCOM WKSHPS* (2014)

Experiments

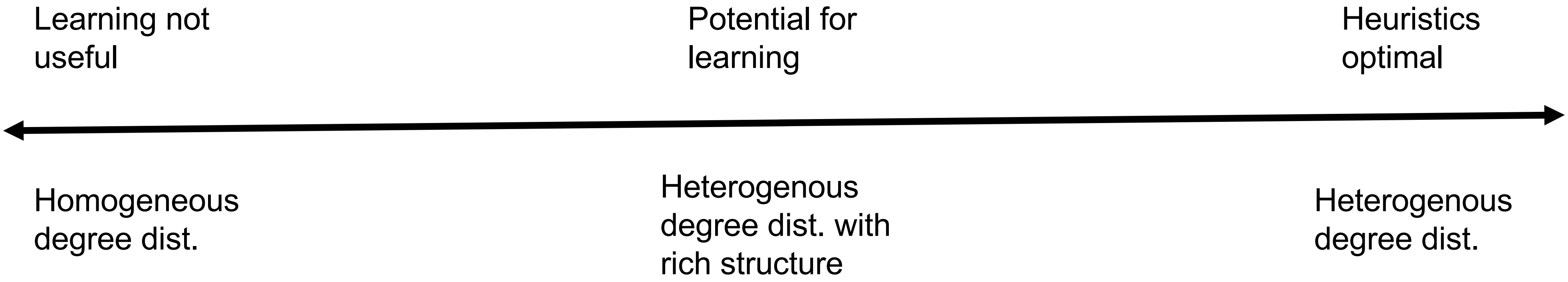
Heuristic Baselines

- High degree
  - Probe the unprobed node with maximum degree
- High degree w/ jump
  - Probe the unprobed node with maximum degree, randomly jump with probability *p*
- Low degree
  - Probe the unprobed node with minimum degree
- Random
  - Probe a node chosen uniformly at random from the unprobed nodes

iKNN-UCB [Madhawa+ ArXiv preprint, 2018]

- K-Nearest Neighbors Upper Confidence Bound
  - Nonparametric multi-armed bandit approach
- Choose node to probe by combining nearest neighbor reward information (based on Euclidean distance between feature vectors) + extent of previous exploration of similar actions

Heterogeneity in network properties creates a learning “spectrum”



Learning not useful

Homogeneous degree dist.

Potential for learning

Heterogenous degree dist. with rich structure

Heuristics optimal

Heterogenous degree dist.

Heterogeneity in network properties creates a learning “spectrum”

Learning not useful

Homogenous degree dist.

Potential for learning

Heterogenous degree dist. with rich structure

Heuristics optimal

Heterogenous degree dist.

Heterogeneity in network properties creates a learning "spectrum"

Learning not useful

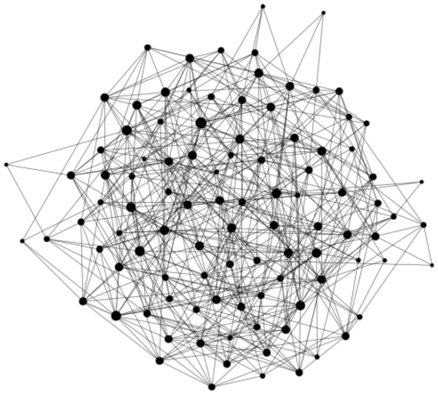
Potential for learning

Heuristics optimal

Erdos-Renyi Model

Heterogenous degree dist. with rich structure

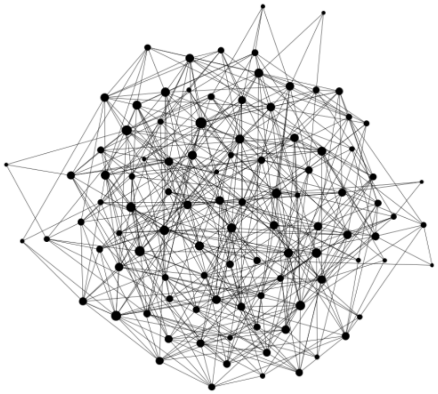
Heterogenous degree dist.



Heterogeneity in network properties creates a learning “spectrum”

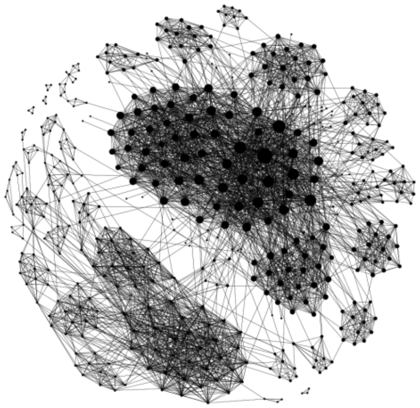
Learning not useful

Erdos-Renyi Model



Potential for learning

BTER Model[1]



Heuristics Optimal

Heterogenous degree dist.

---

[1] C. Seshadhri et al. *Physical Review E* (2012)

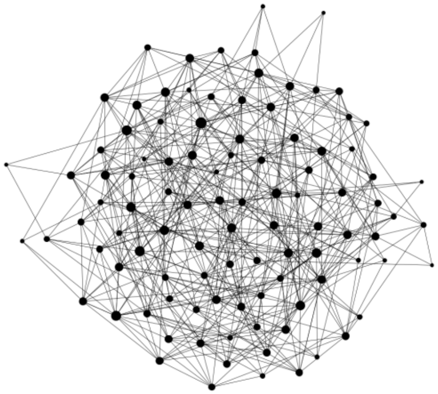
Heterogeneity in network properties creates a learning “spectrum”

Learning not useful

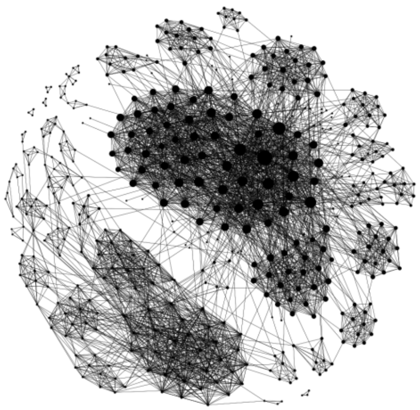
Potential for learning

Heuristics Optimal

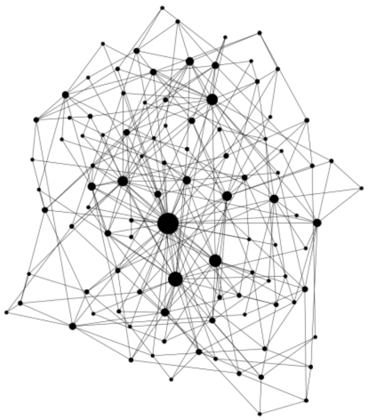
Erdos-Renyi Model



BTER Model[1]



Barabasi-Albert Model



Northeastern University
Network Science Institute

---

[1] C. Seshadhri et al. *Physical Review E* (2012)

Results - Learning Spectrum

Learning not useful

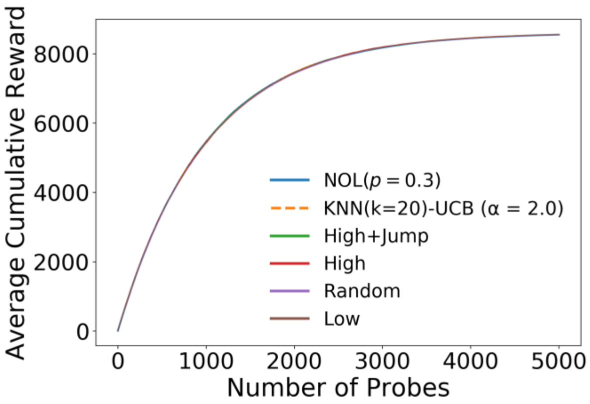
Potential for learning

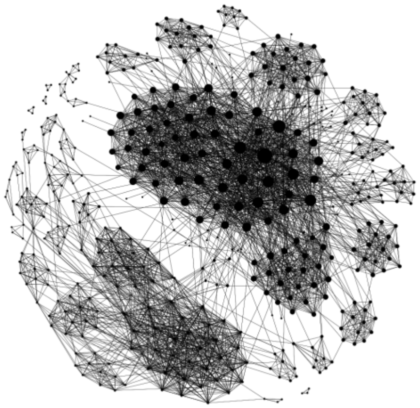
Heuristics Optimal

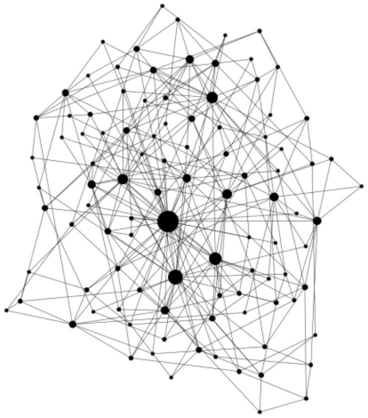
Erdos-Renyi Model

BTER Model

Barabasi-Albert Model



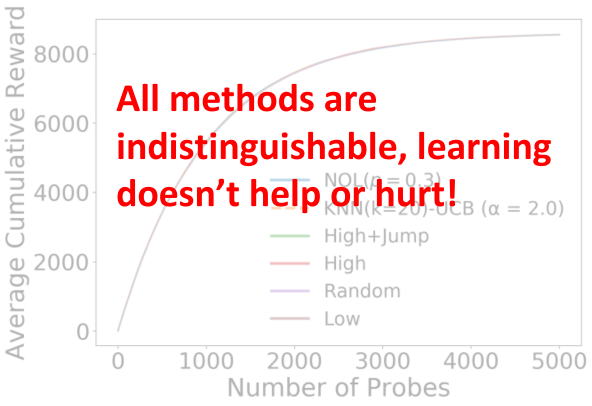




Results - Learning Spectrum

Learning not useful

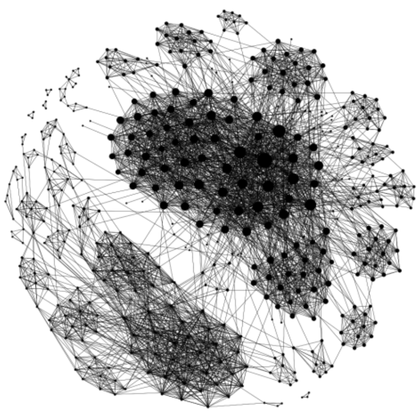
Erdos-Renyi Model



All methods are indistinguishable, learning doesn't help or hurt!

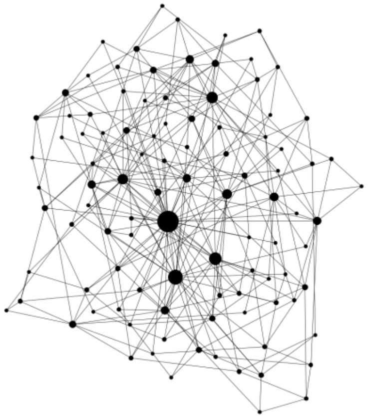
Potential for learning

BTER Model



Heuristics Optimal

Barabasi-Albert Model



Results - Learning Spectrum

Learning not useful

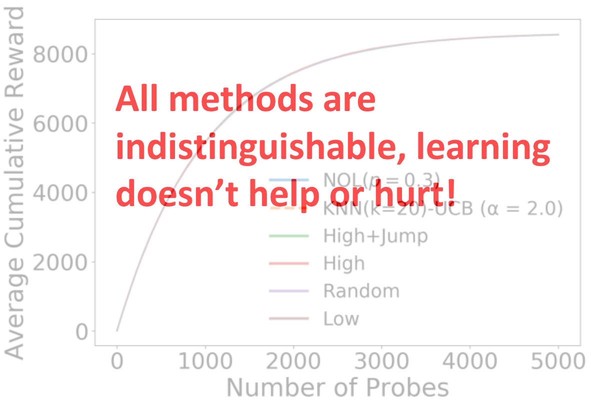
Potential for learning

Heuristics Optimal

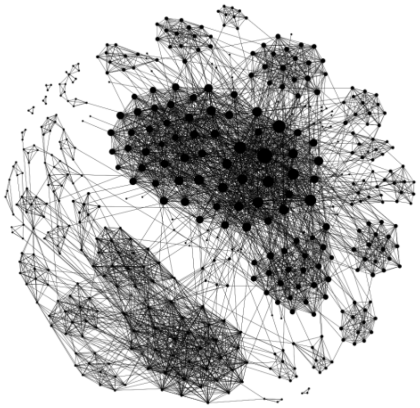
Erdos-Renyi Model

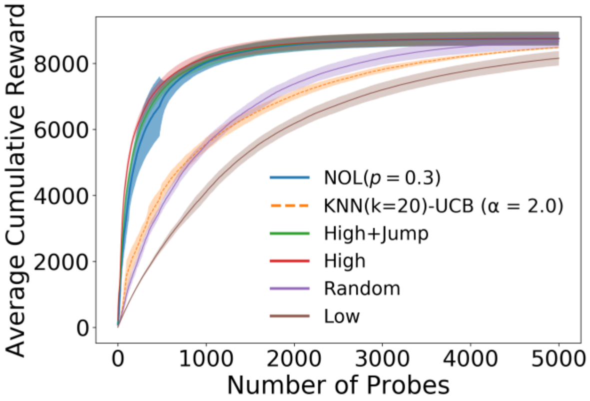
BTER Model

Barabasi-Albert Model



All methods are indistinguishable, learning doesn't help or hurt!





Results - Learning Spectrum

Learning not useful

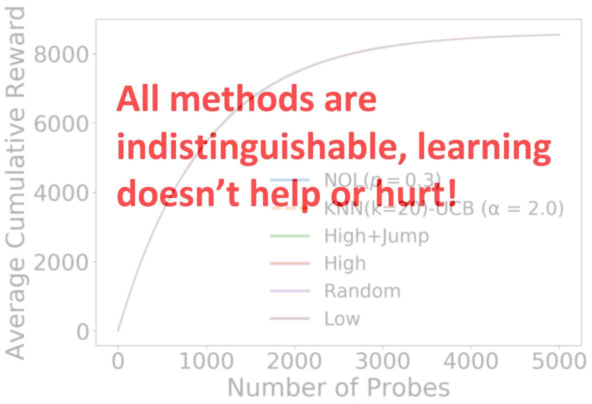
Potential for learning

Heuristics Optimal

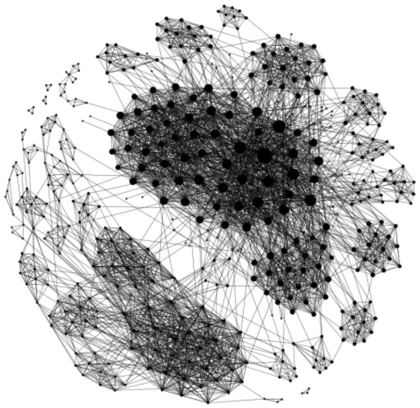
Erdos-Renyi Model

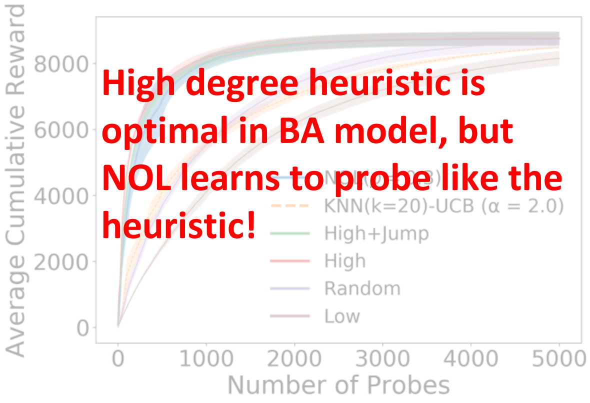
BTER Model

Barabasi-Albert Model



All methods are indistinguishable, learning doesn't help or hurt!





High degree heuristic is optimal in BA model, but NOL learns to probe like the heuristic!

Northeastern University
Network Science Institute

Results - Learning Spectrum

Learning not useful

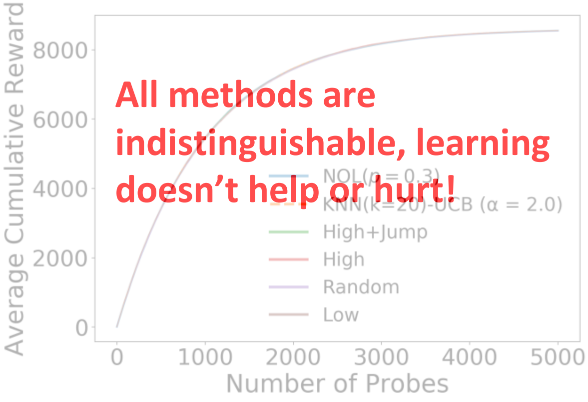
Potential for learning

Heuristics Optimal

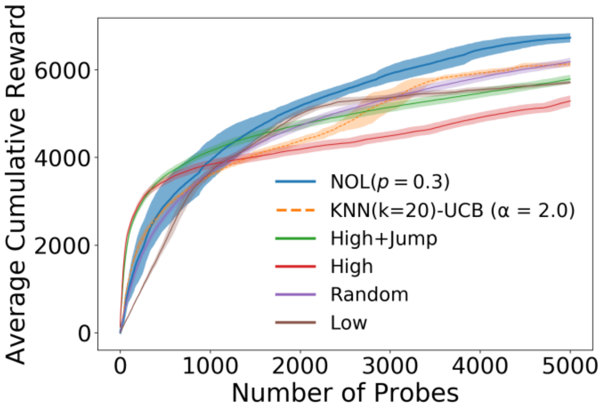
Erdos-Renyi Model

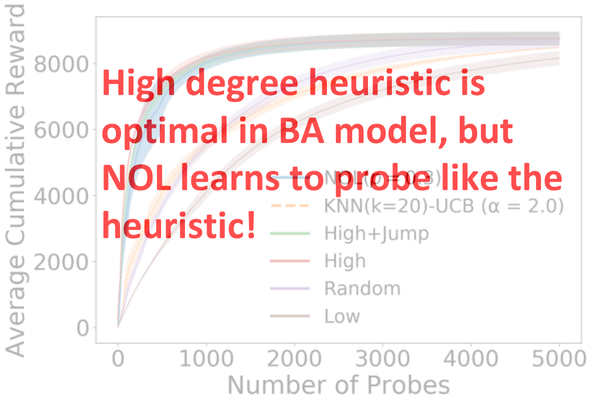
BTER Model

Barabasi-Albert Model



All methods are indistinguishable, learning doesn't help or hurt!





High degree heuristic is optimal in BA model, but NOL learns to probe like the heuristic!

Heterogeneity in network properties creates a learning “spectrum”

Learning Not Useful

Potential for learning

Heuristics Optimal

DBLP

Coauthorship network

N = 6.7k
E = 17k
△s = 21.6k

Cora

Citation network

N = 23k
E = 89k
△s = 78.7k

Enron

Email Communication Network

N = 36.7k
E = 184k
△s = 727k

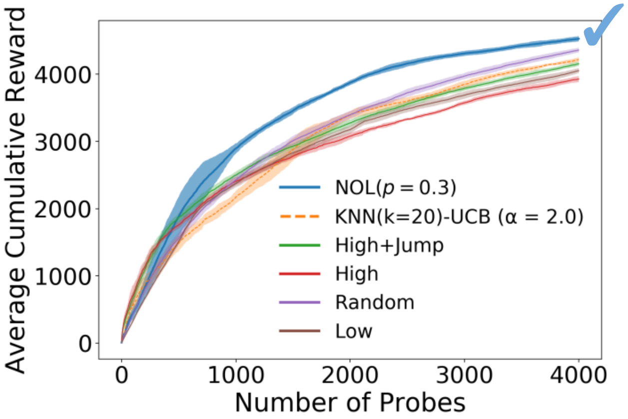
Heterogeneity in network properties creates a learning “spectrum”

Learning Not
Useful

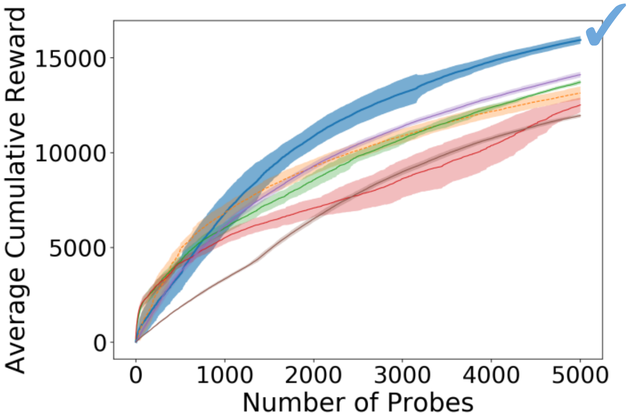
Potential for
learning

Heuristics
Optimal

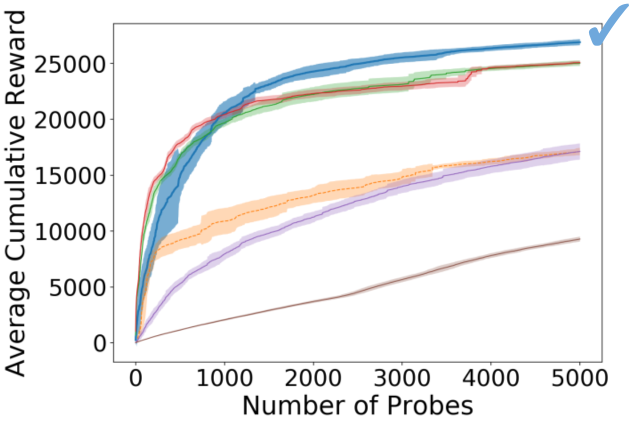
DBLP



Cora



Enron



Summary

- **Network Online Learning** can learn to probe online with minimal assumptions
- Success is tied to properties of the underlying network:
  - Spectrum based on objective function being maximized (degree distribution in these experiments)
  - NOL can learn to behave like the optimal heuristic
- Preliminary experiments suggest some real world complex networks fall in the “learnable” category

Thanks!

Tim LaRock
larock.t@husky.neu.edu

References

Kim et al. (2011). The Network Completion Problem: Inferring Missing Nodes and Edges in Networks. *SIAM International Conference on Data Mining*.

Seshadhri et al. (2012). Community structure and scale-free collections of Erdos-Renyi graphs, *Physical Review E*.

Avrachenkov et al. (2014). Pay few, influence most: Online myopic network covering. *Proceedings - IEEE INFOCOM*.

Pfeiffer III et al. (2014). Active Exploration in Networks: Using Probabilistic Relationships for Learning and Inference. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*.

Soundarajan et al. (2015). MaxOutProbe: An Algorithm for Increasing the Size of Partially Observed Networks. *The 2015 NIPS Workshop on Networks in the Social and Information Sciences*.

Soundarajan et al. (2016). MaxReach: Reducing network incompleteness through node probes. *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*.

Soundarajan et al. (2017). ε-WGX: Adaptive Edge Probing for Enhancing Incomplete Networks. In *Proceedings of the 2017 ACM on Web Science Conference*.

Murai et al. (2018). Selective Harvesting Over Networks. *Data Mining and Knowledge Discovery.*

Madhawa et al. (2018). Exploring Partially Observed Networks with Nonparametric Bandits. arXiv:1804.07059.

Chen et al. (2018). Flexible Model Selection for Mechanistic Network Models via Super Learner. arXiv:1804.00237.

Slides at http://eliassi.org/larock_netsci18.pdf