Parameter Tuning for Influence Maximization

Manqing Ma
Last Updated: 11/19/2018
(CSCI6250 FNS Presentation)
Outline


• **Dataset Preparation**
  • sample graphs and graph metrics

• **Graph Metrics vs BI parameters:**
  • using machine learning; primary result

• **Graph Metrics and Hyperparameter Tuning**
  • ideas
  • Pending work
Objective:

• Param Tuning for BI

\[ BI_i = ar_i + bk_i^{\text{out}} + c \sum_{j \in \partial i \mid r_j = 1} (k_j^{\text{out}} - 1), \]

Params:

a: node resistance (node degree * some distribution within (0, 1))
b: node out-degree (1\textsuperscript{st} level spread)
c: 2\textsuperscript{nd} level spread (no. of nodes able to be activated in the “neighbors of neighbors”)
Objective:

- Param Tuning for GPI

A sketch of parameters in the GPI algorithm:

```
While \( S_Y < S_{\text{goal}} \):
    for i in 1:v: //v: trials of simulations
        ...
        updated node ranks:{GPI_i}
        \( Y = Y \cup Q, |Q|=\lfloor sN \rfloor, s \in (0, 1) \)
        //add top \( \lfloor sN \rfloor \) nodes as Q into Y
        //\( s \): ratio of nodes added into \( Y \) each round
```
Param. Tuning in a nutshell

params: BI – (a, b), GPI – (v, s)

• **without information** – “hyperparameter optimization” **COSTLY**
  • performance: grid search (worse), random search (better), **Bayesian optimization (sequential model-based optimization (SMBO))** (best)

• **with information** – add graph insight to hyperparameter optimization
  • graph insight -> more information

Comparision between grid search and random search, (Bergstra, 2012)

Bayesian Optimization*

* https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f
Question:

• Are the best performance parameters related to certain graph metrics?

-- Let’s find out using machine learning!
Dataset preparation

Use edge swapping method to get graphs with high/low assortativity

Source code: By Panos
Spearman assort. ~(-0.9, 0.9)

Use graph sampling to get sample graphs
Sampling methods:
- Edge sampling
- Random Walk sampling
- 1080 graph samples

Select/compute graph metrics on sampled graphs
~ 20-30 features
Dataset Preparation: (Graph dataset overview)

* summarized from 1080 graph samples
* Asserted “connected” for every graph in the dataset
Dataset Preparation: Graph Metric Overview

(for now) 22 selected from metrics used in:

Overview (BI)

- Parameters we need to tune in BI:
  - \(a\), for resistance \(r\)
  - \(b\), out-degree \(d\)

- Steps:
  1. A grid search of \((a, b)\)s on a collection of sample graphs and get indicator values:

      indicator: “resistance_drop” & “coverage” after 10 rounds of initiator selection

  2. Use machine learning to find the best \((a^*, b^*)\) for given graph metric \((m_1, m_2, ...)\)
  3. Utilize the graph metric information to develop a hyperparameter optimization framework.
Graph Metrics vs BI Params

• Basic machine learning models:

Random Forest
  • (primary result) Achieves ~ .8 accuracy on bipartite classification (e.g. best performance $a \leq 0.5? a > 0.5$)
  • meaningful feature
    • sigma (node threshold distribution)
    • degree variance
    • variance of neighbors’ degrees
    • ...
Graph Metric and hyperparam. tuning

Several Methods --

(1) Pre-train a classification model (e.g. a RandomForest model) using a large quantity of sample graphs. Feed the graph metric values of incoming graph and get the a, b value directly. *Monitor the graph change during spreading process if needed.*

a. Strength: separate param. tuning and deployment;
b. Weakness:
   i. should need a lot of sample graphs;
   ii. might only achieve good prediction regarding intervals (e.g. [0, 0.2) [0.2, 0.6) [0.6, 1]...*)
Graph Metric and hyperparam. tuning

Several Methods:

(2) Use the graph metric prior information to specify how to search the param. space regarding the dataset in “hyperparameter tuning”

a. **Strength**: could always achieve better performance than (1)
b. **Weakness**: might be costly
Graph Metric and hyperparam. tuning

General Hyperparam. Optimization Framework:

Example: “Hyperopt”, Python

Input: objective function; search space; search algorithm (2 implemented so far)

```python
import hyperopt as hp

# define search space

space = hp.uniform('x', -10, 10)
```
Graph Metric and hyperparam. tuning

“Hyperopt” Python

input: objective function; search space; search algorithm (2 choices)

#other search spaces implementations

hp.choice(label, options)
hp.randint(label, upper)
hp.uniform(label, low, high)
hp.quniform(label, low, high, q)
hp.loguniform(label, low, high)
hp.normal(label, mu, sigma)
hp.qnormal(label, mu, sigma, q)
hp.lognormal(label, mu, sigma)
hp.qlognormal(label, mu, sigma, q)
Graph Metric and hyperparam. tuning

“Hyperopt” Python

input: objective function; search space; search algorithm (2 choices)

----

The idea is....

To specify the search space with the graph metric information we have.
Graph Metric and hyperparam. tuning

Inspect the a, b distribution in our dataset:
Graph Metric and hyperparam. tuning

param. distribution could be different given graph metrics:

e.g. “a” distribution given “sigma” (resistence threshold distribution scale)
Graph Metric and hyperparam. tuning

Pending work...

1. How to find the most efficient param. distribution for search space?
   a. Define “efficiency” - **cost and accuracy trade-off**
   b. Derive cost for searching
   c. How well can we predict the accuracy ahead of searching?

2. How to reduce the cost of re-computing graph metrics values in the process of influence spreading?
   a. To derive methods for doing it incrementally
   b. Choose the granularity from experience or current data information
> Thank you!