A journey between graphs and decision making

Philippe Ciblat

Outline

- Attributed graphs
  - *A very short introduction to Graph Theory*
  - *Node classification*
  - *Graph comparison*
  - *Node embedding*

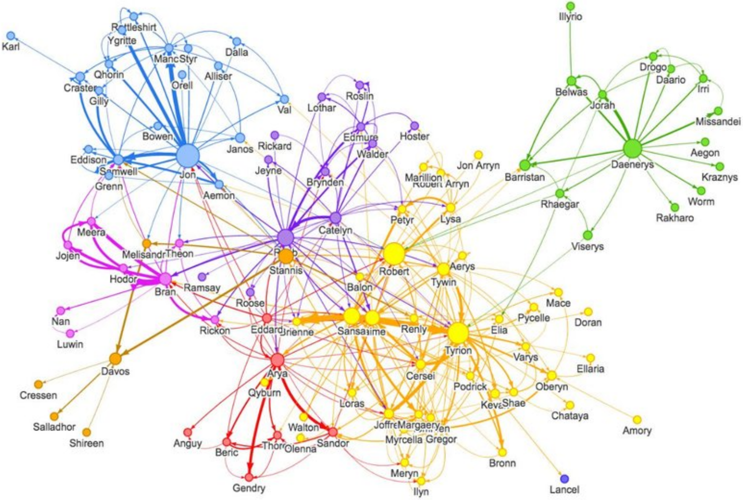
- Index policy: example with Age of Information
  - *Square-root policy*
  - *Whittle's index*
  - *Extension to multivariate states ?*

- Future work directions
  - *Distributed estimation*
  - *Sustainable system*

Part 1 : Attributed graphs

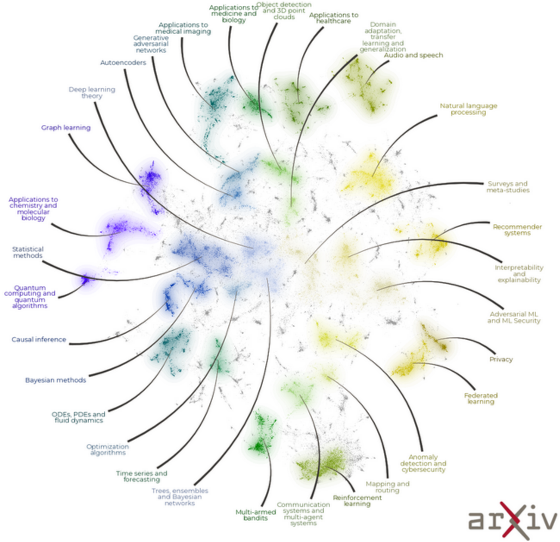
Where do you find graphs?

Social Networks: community detection?



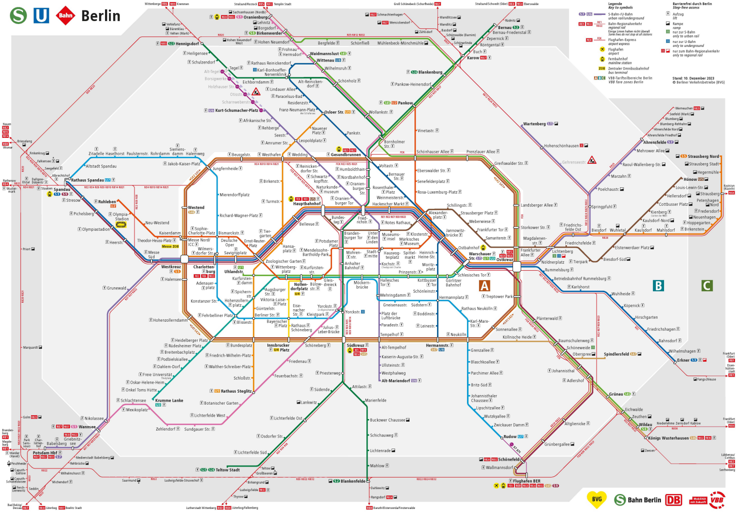
Where do you find graphs?

Papers' database: node classification?



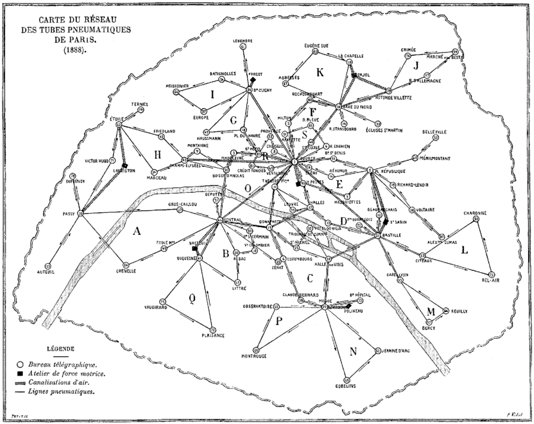
Where do you find graphs?

Public transportation map: graph connectivity level?



Where do you find graphs ?

Communication Networks: information propagation?



Attributed graph

- N nodes (or *vertices*)
- Edges (or *links*) between some nodes
- Edges may be directed/non-directed, weighted/non-weighted
- Each node i may also have a feature (or *value*) $\mathbf{x}_i \in \mathbb{R}^F$

Mathematical representations

We consider non-directed and non-weighted graphs

- Let i be a node and $\mathcal{N}_i$ be the set of its neighbors
- Node degree: $d_i = |\mathcal{N}_i|$ (number of neighbors)
- Degrees matrix: $\mathbf{D} = \text{diag}(d_1, \cdots, d_N)$
- Adjacency matrix: $\mathbf{A}$

$$a_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}$$

  Be careful: $a_{ii} = 0$
- Laplacian matrix: $\mathbf{L} = \mathbf{D} - \mathbf{A}$

Some results

- $\mathbf{L}.\mathbf{1} = \mathbf{0}$
- The second smallest eigenvalue $\lambda_2 \neq 0$ iff graph is connected

Example 1: Heat diffusion

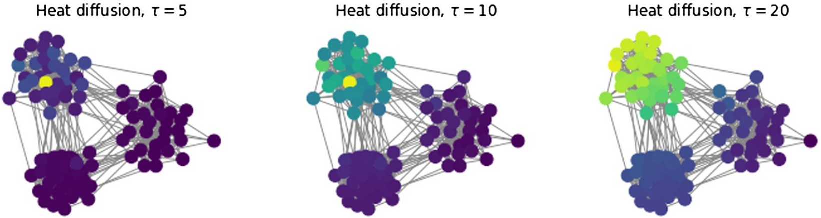
- At time t, temperature of the node ℓ is denoted by $x_\ell(t)$
- The update law comes from Heat diffusion equation

$$\frac{dx_\ell}{dt} = -\sum_{m\in\mathcal{N}_\ell}(x_\ell - x_m) \Leftrightarrow \frac{d\mathbf{x}}{dt} = -\mathbf{L}\mathbf{x} \Leftrightarrow \mathbf{x}(t) = e^{-t\mathbf{L}}\mathbf{x}(0)$$

- Let $\mathbf{L} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{\mathrm{T}}$ with $\lambda_1 = 0$ and $\mathbf{v}_1 = \mathbf{1}/\sqrt{N}$. Then

$$e^{-t\mathbf{L}} = \mathbf{V}e^{-t\boldsymbol{\Lambda}}\mathbf{V}^{\mathrm{T}} \overset{t\to\infty}{\longrightarrow} \mathbf{v}_1\mathbf{v}_1^{\mathrm{T}} = \frac{1}{N}\mathbf{1}\mathbf{1}^{\mathrm{T}} \Rightarrow \lim_{t\to\infty}\mathbf{x}(t) = \overline{x}\mathbf{1}$$

with $\overline{x}$ the average of initial temperatures



source: B. Ricaud et al., "Fourier could be a data scientist: from Graph Fourier transform to signal processing on graphs", Aug. 2019

Example 2: Consensus algorithm

We start with an initial value $\mathbf{x}(0)$
At time t, one node ℓ wakes up and selects $\ell' \in \mathcal{N}_\ell$. Then

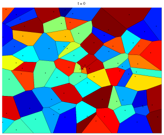
$$x_\ell(t+1) = x_{\ell'}(t+1) = \frac{x_\ell(t) + x_{\ell'}(t)}{2}$$

Finally

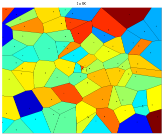
$$\mathbf{x}(t) = \prod_{k=1}^{t} \mathbf{W}_k \mathbf{x}(0)$$

Result

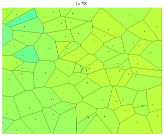
$\lim_{t\to\infty} \mathbf{x}(t) = \overline{x}\mathbf{1}$ as each $\mathbf{W}_k$ doubly-stochastic matrix



Initial Graph



$t = 10$



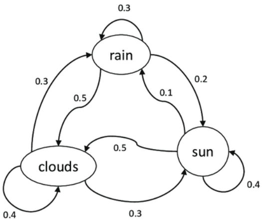
$t = 75$

Link with Markov chain

Finite-state Markov Chain: state $s \in \mathcal{S} = \{s^1, \cdots, s^N\}$

$$\Pr\left(s_{t+1} = s^\ell | s_t = s^k\right) = T_{k,\ell} \geq 0$$

with $\sum_\ell T_{k,\ell} = 1$, so $\mathbf{T}$ is row-stochastic matrix



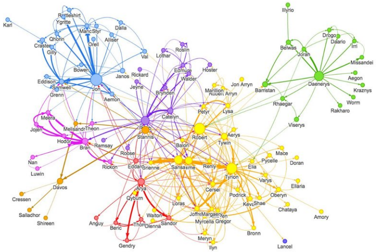
	clouds	rain	sun
clouds	0.4	0.3	0.3
rain	0.5	0.3	0.2
sun	0.5	0.1	0.4

Analyzing Markov chain is equivalent to analyzing Graph

Stationary distribution: μ s.t. $\mu = \mu\mathbf{T}$

source: H. Seyr and M. Muskulus, "Decision Support Models for Operations and Maintenance for Offshore Wind Farms: A Review", 2019

Problem 1: Node classification



Idea: homophily principle

Predict class of each unlabeled node in the graph by relying

- on nodes' features and
- on nodes' graph connections

Main idea: weighted averaging of the current features of adjacent nodes (sometimes followed by a nonlinear function)

- Graph neural networks (GNN): Neural Networks adapted to the attributed graphs. Training done with labeled nodes
- **Our contribution:** we derive in *closed-form* a classifier
  - interpretable algorithm (no black box)
  - less complex since no training
  - no embedding (as done by GNN)

Problem statement

Classifier based on Bayesian decision theory: Maximum A Posteriori

- $\mathcal{V}_u$: set of nodes involved in the classification of node u
- $\mathcal{X}_u = \{\mathbf{x}_u\} \cup \{\mathbf{x}_v, v \in \mathcal{V}_u\}$: set of features of node u and its "helping" nodes
- y_u: class of node u (what we are looking for!)
- D_k: probability density function of features belonging to class k. For any u,

$$D_k(\mathbf{x}_u) = p(\mathbf{x}_u | y_u = k)$$

Graph-Assisted Bayesian (GAB) Classifier

$$\hat{k}_u = \arg\max_k P_u(k)$$

with $P_u(k) = \Pr(y_u = k | \mathcal{X}_u, \mathcal{I}_{\mathcal{G}})$

Problem solution

Derivations of $P_u(k)$. Bayes' rule

$$P_u(k) = \frac{p(\mathcal{X}_u|y_u = k, \mathcal{I}_\mathcal{G})\Pr(y_u = k|\mathcal{I}_\mathcal{G})}{p(\mathcal{X}_u|\mathcal{I}_\mathcal{G})} \propto Q_u(k)\pi_k$$

with $\pi_k = \Pr(y_u = k|\mathcal{I}_\mathcal{G})$ a priori classes' probability
Let Δ_u be the diameter of the set $\mathcal{V}_u$.

$$Q_u(k) = D_k(\mathbf{x}_u) \prod_{d=1}^{\Delta_u} \prod_{v \in \mathcal{N}_u(d)} \left(\sum_{k'=1}^{K} r_{u,v}(k,k') D_{k'}(\mathbf{x}_v) \right)$$

with $r_{u,v}(k,k') = \Pr(y_v = k'|y_u = k, \mathcal{I}_\mathcal{G})$ the probability to be on class k' for node v given the fact that we are in class k for node u

Example

$\mathcal{V}_u = \{v\}$, known $k_v = 1$, $\pi_1 = \pi_2 = 1/2$, and $\Delta_u = 1$:

$$Q_u(1) = D_1(\mathbf{x}_u)\frac{p}{p+q} \text{ and } Q_u(2) = D_2(\mathbf{x}_u)\frac{q}{p+q}$$

with p (resp. q) probability of intra (resp. inter)-class connection

Main result

Assumptions

- 2 equilikely classes
  - $p(k)$ probability that two nodes from class k are connected
    $\overline{p}_{\text{arithmetic}}$ arithmetic average of $\{p(k)\}_k$
  - q probability that two nodes from different classes are connected
- Information on graph is 1-hop

We get

$r(1,2) = \frac{q}{p(1)+q}$	$r(2,2) = \frac{p(2)}{q+p(2)}$
$r(1,1) = \frac{p(1)}{p(1)+q}$	$r(2,1) = \frac{q}{q+p(2)}$

Graph-agnostic iff $r(1,2) = r(2,2)$ and $r(1,1) = r(2,1)$

Main result

Graph-agnostic iff

- $q = \sqrt{p(1)p(2)} = \overline{p}_{\text{geometric}}$, or
- Degree of Impurity $= \frac{q}{\overline{p}_{\text{arithmetic}}} = \frac{\overline{p}_{\text{geometric}}}{\overline{p}_{\text{arithmetic}}} \leq 1$

Graph Neural Network (1/2)

- Use graph structure in addition to node and edge features to generate node representation vectors (i.e., embedding)
- Aggregate the features of neighboring nodes and edges
- Output of the ℓ-th layer of GNN is

$$\mathbf{h}_u^{(\ell)} = \sigma^{(\ell)}(\phi^{(\ell)}(\mathbf{h}_u^{(\ell-1)}, \{\mathbf{h}_v^{(\ell-1)} : v \in \mathcal{N}_u\}))$$

where

- $\mathbf{h}_u^{(\ell)}$ representation vector of node u at ℓ-th layer ($\mathbf{h}_u^{(0)} = \mathbf{x}_u$)
- $\sigma^{(\ell)}$ activation function
- $\phi^{(\ell)}$ linear function associated with weights' matrix $\mathbf{W}^{(\ell)}$

Algorithm

Given $\mathbf{h}_u^{(L)}$, node u is attributed to the class with the highest probability

Graph Neural Network (2/2)

Graph Convolutional Neural Network (GCN):

$$\phi_u^{(\ell)} = \mathbf{W}^{(\ell)} \left(\frac{\mathbf{h}_u^{(\ell-1)}}{d_u + 1} + \sum_{v \in \mathcal{N}_u} \frac{\mathbf{h}_v^{(\ell-1)}}{\sqrt{(d_u+1)(d_v+1)}} \right)$$

Graph convolution Operator Network (GON):

$$\phi_u^{(\ell)} = \mathbf{W}_1^{(\ell)} \mathbf{h}_u^{(\ell-1)} + \mathbf{W}_2^{(\ell)} \left(\sum_{v \in \mathcal{N}_u} \mathbf{h}_v^{(\ell-1)} \right)$$

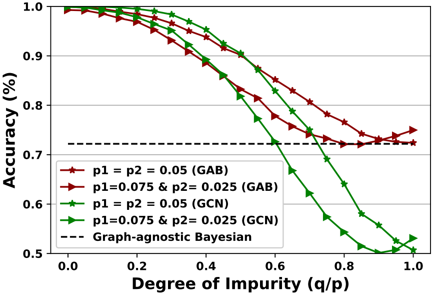
Graph Attention Network (GAT):

$$\phi_u^{(\ell)} = \mathbf{W}^{(\ell)} \left(\alpha_{u,u}^{(\ell)} \mathbf{h}_u^{(\ell-1)} + \sum_{v \in \mathcal{N}_u} \alpha_{u,v}^{(\ell)} \mathbf{h}_v^{(\ell-1)} \right)$$

with $\alpha_{u,v}^{(\ell)}$ the so-called normalized attention coefficients

Numerical illustrations

- 2 classes with different Gaussian distributions
- $N = 5,000$ and $F = 500$
- 500 (already-labeled) nodes



- GAB more robust to DoI than GCN
- GCN becomes worse than graph-agnostic (too confident)

Problem 2: Graph comparison

Question

- Are two graphs close to each other?
- Useful in many applications: link prediction, time-varying analysis, etc

Main issues:

- Balance between features and edges?
- Even if no features, what does it mean two close graphs?
    - counter-example: by cutting a few edges, new graph is not connected : is it far or not from the original one?
    - so just comparing $\mathbf{A}$ is not enough: induced properties are crucial

Detour by the optimal transport

Original problem [Monge1781]

How moving a sand pile with shape 1 into a shape 2 by minimizing the energy consumption?

Shape : f where $f(x)$ provides the level of sand at x

- ◦ $f(x) \geq 0$, and $\int f(x)dx = 1$: probability density function (pdf)

Transport problem

- Transport map: $y = T(x)$
- Transport cost: $c(x, T(x))$, and $C(T) = \int c(x, T(x))f_1(x)dx$
- Transport application: $T_\#$

$$T^\star = \arg \min_{T, T_\# f_1 = f_2} C(T)$$

In general, too hard to solve

Relaxation [Kantorovitch1942]

Transport Map T is not a function anymore but a probability function: given the sand at x, it can be spread at several new positions: $x \mapsto T_x$

$$C(T) = \int \left(\int c(x,y) T_x(y) dy \right) f_1(x) dx$$

s.t.

- Accurate final shape: $f_2(\Omega) = \int (\int_\Omega T_x(y) dy) f_1(x) dx$
- Take only the original shape: $f_1(\Omega) = \int (\int_\Omega T_x(y) f_1(x) dx) dy$

Then consider $T_x(y) f_1(x) = \pi(x,y)$

$$\pi^\star = \arg\min_\pi \iint c(x,y) \pi(x,y) dx dy$$

s.t. $f_2(\Omega) = \int_{y \in \Omega} \left(\int \pi(x,y) dx \right) dy$ and $f_1(\Omega) = \int_{x \in \Omega} \left(\int \pi(x,y) dy \right) dx$

Much easier : Linear programming

Wasserstein distance

Consider two probability mass function (pmf) : discrete version of pdf

- f_1: $\sum_{i=1}^{m} a_i \delta(\bullet - x_i)$ (**a** non-negative vector summing to 1)
- f_2: $\sum_{i=1}^{n} b_i \delta(\bullet - y_i)$ (**b** non-negative vector summing to 1)

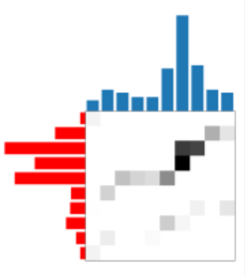
Wasserstein distance (or Earth mover's distance)

Let $\gamma_{i,j}$ be the quantity going from x_i to y_j

$$\mathrm{W}(f_1, f_2) = \min_{\gamma} \sum_{i=1}^{m} \sum_{j=1}^{n} |x_i - y_j|^2 \gamma_{i,j}$$

s.t.

- $b_j = \sum_{i=1}^{m} \gamma_{i,j}, \ \forall j$
- $a_i = \sum_{j=1}^{n} \gamma_{i,j}, \ \forall i$



Graph Diffusion Distance

- Non-attributed graph
- related to Heat diffusion
- Idea: similar graph will diffuse in the same way the heat

[Hammond2013]

$$\mathrm{GDD} = \max_{\tau \geq 0} \left\| e^{-\tau \mathbf{L}_1} - e^{-\tau \mathbf{L}_2} \right\|_2^2$$

Gromov-Wasserstein distance

- Non-attributed graph
- Adaptation of Wasserstein distance to Graph
- Matrices $C^s \in \mathbb{R}^{m\times m}$ and $C^t \in \mathbb{R}^{n\times n}$

[Peyré2016]

$$\text{GW} = \min_{\gamma} \sum_{i,i',j,j'} d(C^s_{i,i'}, C^t_{j,j'})\gamma_{i,j}\gamma_{i',j'}$$

s.t.

- $a_i = \sum_{j=1}^{n} \gamma_{i,j}, \ \forall i$
- $b_j = \sum_{i=1}^{m} \gamma_{i,j}, \ \forall j$

Application to Graph:

- **C** may be the adjacency matrix **A**
- **C** may be a similarity matrix between nodes
- Hyperparameters **a** and **b** to be tuned

Fused Gromov-Wasserstein distance

- Adaptation to attributed graph

[Flamary2020]

$$\text{FGW} = \min_{\gamma} \sum_{i,i',j,j'} \left[\alpha d_1(C^s_{i,i'}, C^t_{j,j'})\gamma_{i,j}\gamma_{i',j'} + (1-\alpha) d_2(\mathbf{x}_i, \mathbf{x}_j)\gamma_{i,j}\right]$$

s.t.

- $a_i = \sum_{j=1}^{n} \gamma_{i,j}, \ \forall i$
- $b_j = \sum_{i=1}^{m} \gamma_{i,j}, \ \forall j$

Diffusion-Wasserstein distance

- Attributed graph
- $\mathbf{y}^s = e^{-\tau^s \mathbf{L}_s} \mathbf{x}^s$ heat diffusion with initial values $\mathbf{x}^s$
- $\mathbf{y}^t = e^{-\tau^t \mathbf{L}_t} \mathbf{x}^t$ heat diffusion with initial values $\mathbf{x}^t$

[Borgnat2021]

$$\mathrm{DW}_{\tau^s,\tau^t} = \min_{\gamma} \sum_{i,j} d(\mathbf{y}_i^s, \mathbf{y}_j^t)\gamma_{i,j}$$

s.t.

- $a_i = \sum_{j=1}^{n} \gamma_{i,j}, \ \forall i$
- $b_j = \sum_{i=1}^{m} \gamma_{i,j}, \ \forall j$

Extreme cases:

- $\tau^s = \tau^t = 0$: Wasserstein distance
- $\tau^s = \tau^t = \infty$: average comparison of features

Application : Image color adaptation

- Histogram on RGB
- Underlying graph for preserving neighborhood.
- Weighted average for color adaptation with path γ

$+$

$=$

Original image *Target color* *Final image*

source: R. Flamary and N. Courty, "https://pythonot.github.io/", 2019

Problem 3: Graph embedding

Main idea

- Vector : nice representation for signals
- Why? many algorithms adapted to vectors
  - in classification (k-means, NN with vector as input)
  - in regression (linear, NN with vector as input)

Embedding: representing any type of signal as a vector

Examples:

- Text: word2vec
  - close vector = synonym
  - semantic vector space: $v_{queen} + v_{man} = v_{king}$
- Graph:
  - graph representation learning (one graph becomes one vector)
  - node representation learning (each node becomes one vector)
    - ↪ "close" points in graph are close points in vector space
    - ↪ meaning of "close" when trade-off between edges and features

Representation self-learning

- $\mathbf{x}_u \in \mathbb{R}^F$ feature vector at node u
- $\mathbf{X} \in \mathbb{R}^{N \times F}$: matrix stacking initial feature vectors of all nodes
- $\mathbf{A}$: adjacency matrix of the graph

Goal

- Self-learning node representation (without human annotation/tag)
- i.e., learning a graph neural network (with L layer)

$$\mathbf{H}^{(L)} := f(\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{N \times F'}$$

  with
  - $F' \leq F$ the embedding size
  - u-th row of $\mathbf{H}^{(L)}$ the embedding/representation vector $\mathbf{h}_u^{(L)}$ of node u
- Finding an appropriate *criterion* to exhibit f

Contrastive learning

Given a feature $\mathbf{h}_u$ of node u, we generate

- a *positive* example $\mathbf{h}_u^+$ (close to $\mathbf{h}_u$)
- a set of *negative* examples Q_u

We define a loss $\mathcal{L}$ offering low value when $\mathbf{h}_u$

- similar to $\mathbf{h}_u^+$
- dissimilar to all elements $\mathbf{h}^-$ of Q_u

A standard loss

$$\mathcal{L} = -\sum_{u \in \mathcal{G}} \mathbf{h}_u^{\mathrm{T}} \mathbf{h}_u^+ + \log \sum_{u \in \mathcal{G}} \left(e^{\mathbf{h}_u^{\mathrm{T}} \mathbf{h}_u^+} + \sum_{\mathbf{h}^- \in Q_u} e^{\mathbf{h}_u^{\mathrm{T}} \mathbf{h}^-} \right)$$

How generating negative examples?

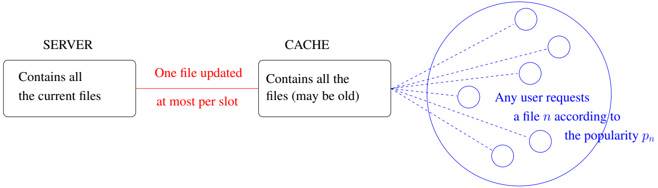
- Consider two (small) perturbations on edges and features
  - $(\mathbf{X}_1, \mathbf{A}_1) \sim t_1(\mathbf{X}, \mathbf{A})$
  - $(\mathbf{X}_2, \mathbf{A}_2) \sim t_2(\mathbf{X}, \mathbf{A})$
- Apply the current node representations
  - the baseline representation $\mathbf{H}^L = f(\mathbf{X}_1, \mathbf{A}_1)$
  - the positive example $\mathbf{H}_+^L = f(\mathbf{X}_2, \mathbf{A}_2)$
- Select negative examples: for node u, all nodes at its ℓ-hop
- Update weights of f using the loss function $\mathcal{L}$

Numerical illustrations: classification based on our node embedding

	Cora	Citeseer	Pubmed	Arxiv
Raw features	47.9	49.3	69.1	55.5
SoTA	82.3	71.8	76.8	**70.2**
Proposed Method	**83.6**	**72.5**	**79.8**	**70.2**
GCN (supervised)	81.5	70.3	79.0	71.7

Part 2 : Index policy

Application: caching with Age of Information



- Content n is time-sensitive ($X_n(t)$: age in caching)
- Content n has its own popularity (p_n: probability to be requested)
- Ex: newspaper website, web crawling, video last version, ...

Question

- Given a timeslot t, which item should be downloaded from the server to the cache to be as up-to-date as possible?
- Scheduling problem

$$\{u_t\}_t = f(\text{information on the system})$$

Optimization problem

Optimization problem

$$\arg\min_{u_1,\cdots,u_T} \sum_{n=1}^{N} p_n \int_0^T X_n(t)\,dt$$

s.t. $u_t \in \{1,\cdots,N\}$ for all t, and $\sum_{t=1}^{T} \mathbb{1}\{u_t > 0\} = T$.

Approaches:

- Probabilistic method by re-writing the problem
- As underlying Markov chain, constrained MDP well adapted
  - Optimal random policy exists
  - Suboptimal approach but simple: Whittle's index

Approach 1: concept of per-file update rate

- Consider λ_n the per-file update rate
- Actually, when T large enough,

$$\frac{1}{T}\int_0^T X_n(t)dt \approx \frac{1}{\lambda_n}$$

New optimization problem

$$\min_{\lambda_1,\dots,\lambda_N} \sum_{n=1}^{N} \frac{p_n}{\lambda_n}$$

s.t. $\lambda_n \geq 0$, and $\lambda_1 + \cdots + \lambda_N = 1$.

Main result

Problem is convex and leads to

$$\lambda_n^* = \frac{\sqrt{p_n}}{\sum_{m=1}^{N} \sqrt{p_m}}$$

Update rate of file n follows a square-root law wrt. its popularity

Practical protocol

Let $\tau_n^\star = 1/\lambda_n^\star$ be the optimal inter-update time for file n

$$u_t = \arg \max_{u \in \{1, \cdots, N\}} (X_u(t) - \tau_u^\star)$$

General context: Schedule-ordered by Age-based Priority (SOAP)

- $r(D, X)$: rank function with descriptor D and age X
- Scheduled user

$$u_t = \arg \max_{u \in \{1, \cdots, N\}} r(D_u(t), X_u(t))$$

- Many policies follow this shape
  - Round-Robin (RR), $r(\emptyset, X_u) = X_u$
  - “Weighted Round-Robin”, $r(d_u, X_u) = d_u.X_u$. How choosing d_u ?

Approach 2: index based policy

Find a suboptimal policy based on an index:

$$u_t = \arg \max_{u \in \{1, \cdots, N\}} \mathcal{I}_u(S_u)$$

- $\mathcal{I}$: it is an heuristic
- Whittle's index: methodology for exhibiting a reasonable index in Restless Multi-Arm Bandit problem
  - N bandits/players/agents
  - At each timeslot, select one bandit (let's say u_t)
  - Its state s_{u_t} is modified according to its action, and is rewarded

  but states of other bandits also modified and rewarded in different ways (restless)
- When non-playing bandits are frozen (no state evolution) and not rewarded: Gittins' index is optimal

$$\mathcal{I}_u^{\mathcal{G}}(S) = \sup_{\tau > 0} \frac{\mathbb{E}[\sum_{t=0}^{\tau-1} \gamma^t r_u(S(t)) | S(0) = S]}{\sum_{t=0}^{\tau-1} \gamma^t}$$

Whittle index (1/2)

$$\arg\max_{\{a_u(t)\}_{u,t}} \lim_{T\to\infty} \mathbb{E}\left[\sum_{t=0}^{T-1}\gamma^t \sum_{u=1}^{N} r_u(s_u(t), a_u(t))\right]$$

s.t. $\sum_{u=1}^{N} a_u(t) = 1$ (C1) and $a_u(t) \in \{0,1\}$ (C2)

Two modifications:

- Relaxation: C1 replaced with $\sum_{t=0}^{\infty}\gamma^t \sum_{u=1}^{N} a_u(t) = 1/(1-\gamma)$
- Lagrangian penalty

$$\arg\max_{\{a_u(t)\}_{u,t}} \mathcal{L}(\lambda)$$

s.t. C2 and with

$$\begin{aligned}\mathcal{L}(\lambda) &= \lim_{T\to\infty} \mathbb{E}\left[\sum_{t=0}^{T-1}\gamma^t \sum_{u=1}^{N} r_u(s_u(t), a_u(t))\right] - \lambda\left(\sum_{u=1}^{N} a_u(t) - 1/(1-\gamma)\right)\\ &= \lim_{T\to\infty} \sum_{u=1}^{N}\mathbb{E}\left[\left(\sum_{t=0}^{T-1}\gamma^t r_u(s_u(t), a_u(t)) - \lambda a_u(t)\right)\right]\end{aligned}$$

Whittle index (2/2)

- The problem is now decoupled
- For each bandit u and fixed λ, we maximize

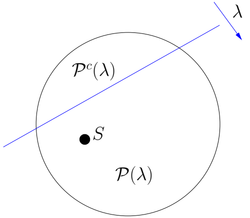
$$\mathcal{L}_u(\lambda) = \lim_{T\to\infty} \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t r_u(s_u(t), a_u(t)) - \lambda a_u(t)\right]$$

- $\mathcal{P}(\lambda)$: set of states leading to $a = 1$ obtained via $\mathcal{L}_u(\lambda)$ (std MDP)
- **Optimal policy:** play ($a = 1$) if $s \in \mathcal{P}(\lambda)$ else idle ($a = 0$)
- **Indexability:** if idle for λ, then still idle for $\lambda' > \lambda$ (if higher penalty for being active, stay idle): if $s \in \mathcal{P}^c(\lambda)$ then $s \in \mathcal{P}^c(\lambda')$

Definition

$$\mathcal{I}^{\mathcal{W}}(S) = \lambda^\star$$

s.t. if $\lambda > \lambda^\star$, $S \in \mathcal{P}^c(\lambda)$, else $S \in \mathcal{P}(\lambda)$

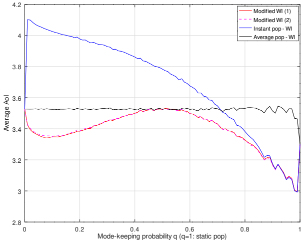


Closed-form expression [unpublished]

$$\mathcal{I}_u^{\mathcal{W}}(X_u) = \sqrt{p_u} X_u \quad (X_u = \text{age})$$

- Is it close to square-root law ?
  - checked by simulation but not theoretically
- Extension to time-varying popularity: 2-D state (age, popularity)
  - Two modes : $R^{(1)} = \{p_n^{(1)}\}$ and $R^{(2)} = \{p_n^{(2)}\}$
  - Whittle's index in 2-D state: unfeasible except special cases (here, $q = 0.5$ where q probability to stay in its mode for each user)

$$\mathcal{I}_u(X_u, R_u) = \sqrt{q p_u^{(R_u)} + (1-q) p_u^{(R_u^c)}} X_u$$



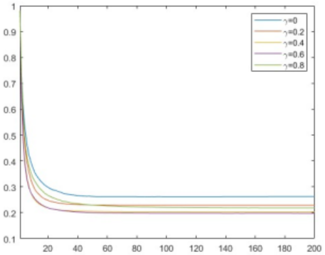
Future works

Perspective 1: Distributed estimation

$$\text{At sensor } k,\ x_k(0) = \theta + w_k(0)$$

Estimation of θ by sharing $x_k(0)$ with neighborhood iteratively

- Gossip algorithm
- Here, consider
  - ↪ flooding error at receiver side related to the size of in-going neighborhood
  - ↪ solution : censorship rate γ at transmitter side



Goal

Closed-form expression for Cramer-Rao Bound to design γ

Perspective 2: Efficiency

$$\textbf{Efficiency} = \frac{\textbf{metric of performance}}{\textbf{consumed energy}}$$

Two kinds of consumed energy: Life-Cycle Assessment

- OPEX-like energy: operational one
- CAPEX-like energy: embodied one

Example 1: Communication network

- OPEX: transmit energy (load-dep.), no-idle hardware (load-ind.)
- CAPEX: Sleeping energy, Mining, Manufacturing
- Concerns: open-data/models missing, depreciation duration, ...

Example 2: Machine Learning

- OPEX: computation energy during usage phase
- CAPEX: Training, Computer's manufacturing, Cooling
- Concerns: open-data/models missing, depreciation duration, ...

Perspective 2: or Sustainability?

Sustainable system "*meets the needs of present generations without compromizing the ability of future generations to meet their own needs*" [Brundtland1987]

Implementation: given a large area, level of power is fixed

Why is it different from energy efficient system?

- rebound effect is avoided
- if gain in energy consumption comes from enablement effect, customer behavior has to be predicted

Main concerns:

- Does not depend only on engineers' answers
- Concept on priority usages, net neutrality
- Required Science and Technology Studies (STS)