



HAL
open science

Learning from partly observable time- dependent graphs

Jan Ramon

► **To cite this version:**

Jan Ramon. Learning from partly observable time- dependent graphs. ECML/PKDD 2017 - Workshop on Time-Dependent Large-Scale Graphs, Sep 2017, Skopje, Macedonia. hal-01721301

HAL Id: hal-01721301

<https://inria.hal.science/hal-01721301v1>

Submitted on 1 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Learning from partly observable time-dependent graphs

Jan Ramon

TDLSG workshop @ ECML/PKDD 2017

Skopje, September 18th, 2017

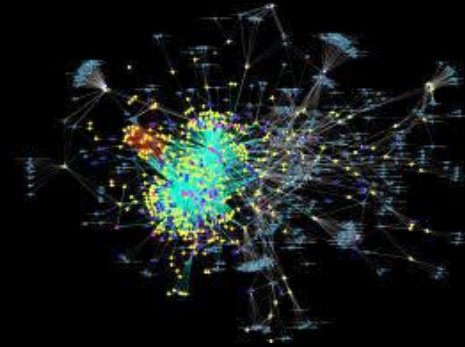


Contents

- Introduction
 - Types of time-dependent graphs
 - Unobservable data
- Case studies
 - Traffic
 - homophilic preferential attachment
 - biological networks
- Discussion & conclusions

Types of time-dependent graphs

- Fixed structure, evolving attributes
 - E.g. Gene regulatory network
 - Node: gene
 - Arc: regulation
 - Attribute: expression level
- Fixed attributes, evolving structure
 - E.g. Citation network
 - Node: article
 - Arc: citation
- Evolving attributes, evolving structure
 - E.g. Social network

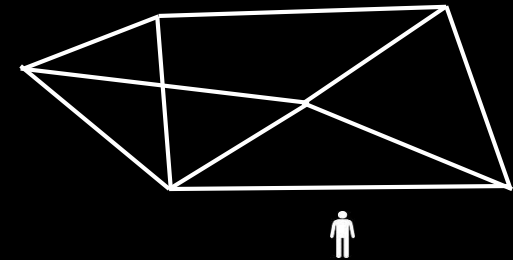
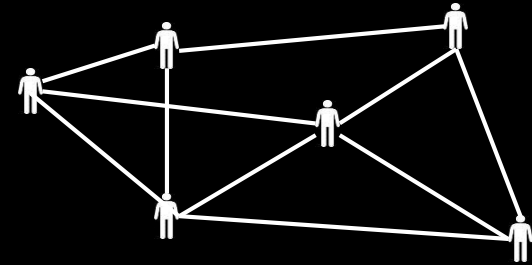


Observability

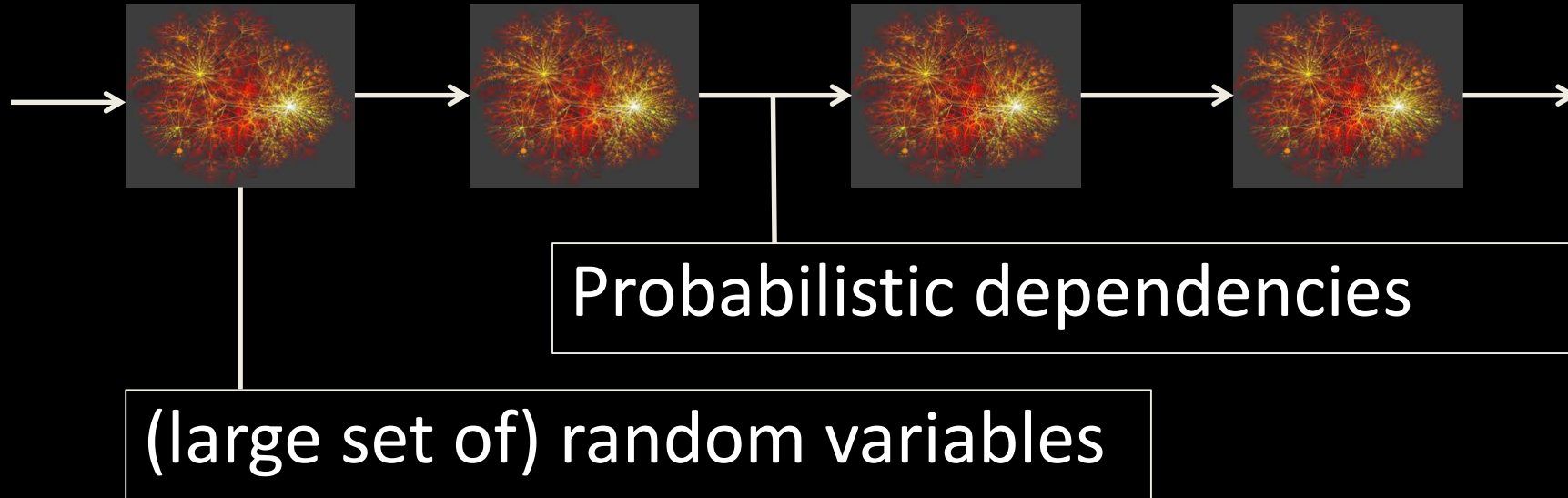
- Collecting actual information
 - is expensive
 - Communication costs
 - Bothering people to record everything
 - Isn't privacy-friendly
 - May be technically infeasible
 - If observations require destructive measurements

Some observation settings

- Nodes as observers
 - Observe/query/inform neighborhood
 - Limited information about distant nodes
 - E.g. Node in communication network
 - E.g. Car in road network
- External observer
 - Overview of graph structure
 - Only limited observation of local details
 - E.g. Service provider in communication network
 - E.g. Government/maintainer monitoring road network



Dynamic probabilistic models



- TD-graph can be modeled with probabilistic relational model (MLN, ProbLog, PRM, RBN, ...)
- But existing inference techniques can't exploit all structure easily

Exploitable structure

- Uniformity of network
 - closeby vertices are/behave similar(ly)
- Continuity
 - Values don't change
- Information flow
 - Also: causality
 - (but not easily conditional independence)

Case studies

- Case study 1: Local observation
- Case study 2: Unobservable history
- Case study 3: Destructive measurement

Case study: Traversal time learning in connected cars

- Networks:

- Traffic

- Static structure: Road network
 - Time-dependent attribute: traffic load



- Communication

- Time-dependent structure: cars communicate if they are close together (sufficiently close to exchange useful traffic experience)

Traffic data

- Car collects data:
 - Car sensors (windows, seats, fuel, ...)
 - Real-time data: acceleration, ...
 - GPS localisation
 - Video data
- Infrastructure collects data:
 - Is parking space used?
 - Speed of traffic
- Databases:
 - Maps, navigation data
 - Ride organization (agenda, ride sharing, ...)

Case study: Traversal time learning in connected cars

- Tasks:
 - Understand behavior of other traffic users
 - Understand & maintain car
 - Determine best route (possibly avoiding congestions)
 - Find free parking slots
 - Learn about unusual traffic situations
 - Update world map
 - ...

Case study: Traversal time learning in connected cars

- Two main problems:
 - Learn regularities in the system
 - Collect real-time information sufficiently quickly
 - Learn to communicate, prioritize, ...
- Here:
 - Learn traversal time of road segments
 - Privacy-friendly

Centralized vs. decentralized

- Privacy
 - Central server can track location of users
- Risk of failure of central service
- Communication & computation cost
 - Amount of applications increases
 - Data volume increases

Constraints

- Each car has a local model of the traffic in its neighborhood
- Cars can only communicate with nearby cars
- Need to respect privacy
- Semi-stream: the car memory is large, but cars hear all broadcasts at most once

Model

- Road segments $e \in E$
 - have average traversal time $\bar{d}_{e,\tau}$ in time-of-days τ .
- Users (cars) $u \in U$
 - prefer to take $w_z^u \bar{d}_{e,\tau}$ time for a unit distance on roads of type z (highway, local road, ...)
- Actual traversal i of $x_i^e \in E$ by $x_i^u \in U$ at time x_i^t takes time x_i^d , assumed close to $w_z^u \bar{d}_{e,\tau} + c_i$, where c_i is the additional time due to congestion

Objective function

$$L(w, c) = \sum_i (x_i^u - w_z^u \bar{d}_{e,\tau} - c_i)^2$$
$$+ \lambda_1 \sum_{i,j} (c_i - c_j)^2 \exp\left(-\left(x_i^t - x_j^t\right)^2 - d(x_i^e, x_j^e)\right)$$
$$+ \lambda_2 \sum_i c_i^2$$

Accurate predictions

Traffic situation is often similar in nearby places and at nearby times

Try to explain things with few congestions

Algorithm

- Keep knowledge alive:
 - Broadcast learned average traversal times and current congestions, newcomers in area will learn it, those leaving can forget.
- Decentralized: Not just any optimization algorithm
- EM strategy:
 - Estimate c_i from difference between predicted and observed traversal time and from neighborhood
 - Estimate w_z^u by averaging difference between broadcasted average and personal average.
 - Improve $\bar{d}_{e,\tau}$ incrementally from new observations

Learnability

- Static: Converges rather quickly
 - Quadratic program
 - Each step decreases loss
 - (Wider broadcast is helpful)
- Time-dependent:
 - Target may drift
 - Can converge (and predict well) if target moves sufficiently slowly

Privacy

- Broadcast only
 - Congestion c_i
 - Road traversal time average $\bar{d}_{e,\tau}$
- But not
 - Personal deviation in speed w_z^u
- Apply protocol to broadcast training example (x_i^e, x_i^t, x_i^d) without revealing identity x_i^u
 - Challenge: ensure trust in broadcasted messages (e.g., making traceable in case of misinformation)

Case study: Homophilic preferential attachment

- Preferential attachment → powerlaw graphs
 - Very simple
 - Very popular
 - No data
- E.g. Social networks
- Can we parameterize the generative process to understand better?
 - Parameterized preferential attachment PA_θ
 - Newly arrived node selects neighbors v with probability $PA_\theta(v)$



Homophilic preferential attachment

Problem statement

- We only see the end-result, nobody kept a log
- Can we still estimate θ ?
- Simple case: homophily
 - Each vertex v has feature vector $x^v \in F$ with distribution $p_f(\cdot)$
 - Probability of selecting neighbor v for u is proportional to $\sum_i \theta_i x_i^u x_i^v$.

Learning homophilic weights

- Known history (order in which nodes arrive):
 - Regression
 - Given newcomer features, existing node features and edge (1) / no-edge (0)
- Undirected graph:
 - Can't reconstruct attachment examples directly
 - Probabilistic approach
 - Assume order of node arrivals is random

Homophilic preferential attachment

Solution by minimizing loss

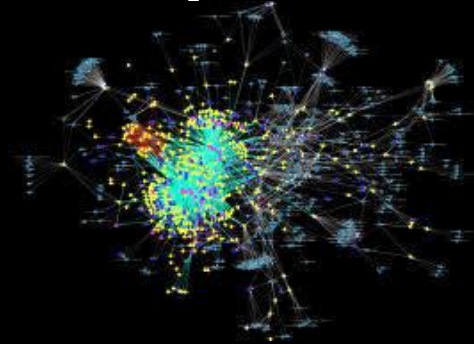
$$L(w) = \sum_{x \in F} \sum_{y \in F} \left(E_{x,y} - \frac{p_f(x)p_f(y) \sum_i w_i x_i y_i}{\sum_{z \in F} p_f(z) \sum_i w_i x_i z_i} \right)$$

Learnability

- Yes, one can learn from a snapshot
 - Need $O(\epsilon^{-2} p_{min}^2)$ examples
- Extensions:
 - Including degree, patterns, ...
 - Becomes quickly more complex

Case study: biological regulatory networks

- Network:
 - Nodes: genes
 - Arcs: regulations
- More activated genes (high expression level) generate more transcription factors that (up or down) regulate other genes.



$$v_{g,t} = f(v_{pa(g,1),t-1}, v_{pa(g,2),t-1}, \dots)$$

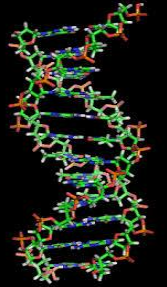
where $pa(g, i)$ is i -th parent of gene g and $v_{g,t}$ is expression level of gene g at time t .

Learning task

- Given:
 - Input (experimental setup), output (measurement) pairs
- Learn:
 - The network
 - structure pa
 - combination functions f
- Active learning operators:
 - Can choose experiment

Biological regulatory networks

- Hardness:
 - In principle turing complete (see also DNA computing)
 - Hence, learning network from (only) input and output is hard.
- Experiments:
 - Destructive
 - May measure output at selected point in time
 - May modify the network (gene knock out)



Biological regulatory networks

- Program induction?
 - Learning programs from input-output pairs is hard
 - But what if we can also get input-output pairs of programs obtained by deleting lines in the original program?
- Approach:
 - Hill climbing through model space
 - In each iteration evaluate minor changes to model
 - Exploit domain knowledge
- Results:
 - Can't expect to learn exactly the same network
 - With sufficient domain knowledge, improvements on prior models can be made

Case studies

| Setting | Observation | Exploit | |
|-------------------------|--|--|-------------------------|
| Traffic | Neighborhood Semi-stream Multi-agent | Continuity Collaboration (info flow) (regularity of network) | Stochastic gradient |
| Preferential attachment | Global Only present Single world | Regularity of network (and Large Scale) | Analysis & optimization |
| Regulatory network | Selected variable Selected time Repeatable | Experimental options (info flow) | Hill climbing |

Discussion – Generic approaches?

- Several types of settings for time-dependent networks.
 - Wide range in learnability
- generic probabilistic model solvers can't (yet) exploit the various structures. How does this map to lifted inference?
- Meta-learning from 3 case studies:
 - Network regularity (i.e. High network entropy) is important in determining learnability

Discussion - Complexity

- What makes learning from time-dependent large-scale partially-observed graphs hard?
 - No network regularity
 - Long sequences without good observations (as in (PO)MDPs) (making information flow harder to exploit)
 - Discrete / non-continuous / combinatorial evolution

Conclusions

- TD graph is often not fully observable
- LS can help to collect sufficient information
- Often domain-specific solutions needed

Future directions

- More in-depth study of learnability
 - Exploit network regularity?
- More generic approach?
 - Can we automatically understand structure, information flow, ...?
- Tagging: only “TDG” may not be sufficient
 - Can we isolate frequently recurring (sub)problems?

Questions?

