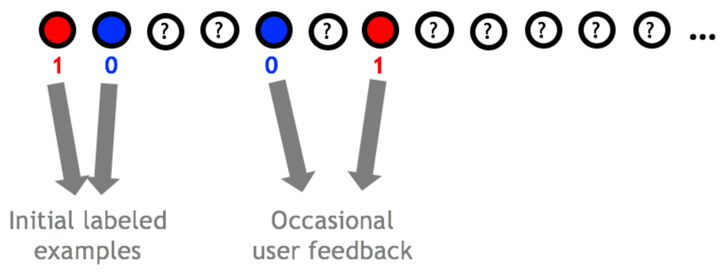
Semi-Supervised Learning on Data Streams via Temporal Label Propagation

Tal Wagner — MIT
Sudipto Guha — Amazon
Shiva Kasiviswanathan — Amazon
Nina Mishra — Amazon

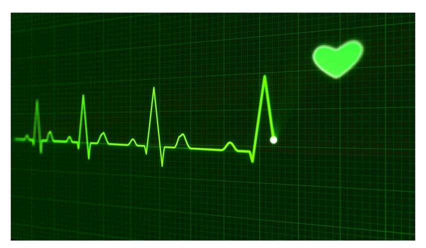
Introduction

Problem:

- Data points arrive on stream
- Few are labeled (0, 1)
- Most are unlabeled
- **Task:** Label points on-the-fly



Examples:



medical signal monitoring

network monitoring

object detection on video feed

Challenges:

- **Time:** Label points quickly
- **Memory:** Stream too large to fully store
- **Semi-supervised learning:** Learn from labeled and unlabeled data

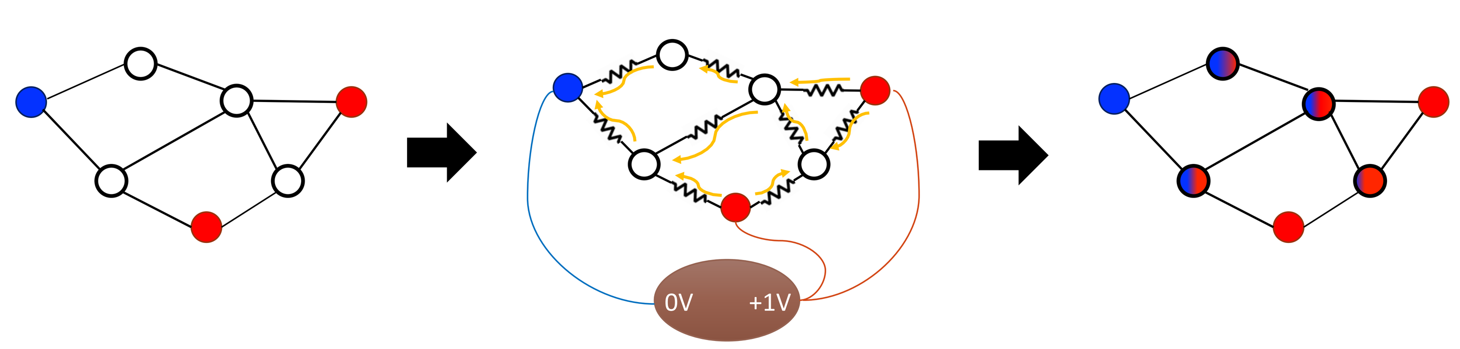
Background

Label Propagation:

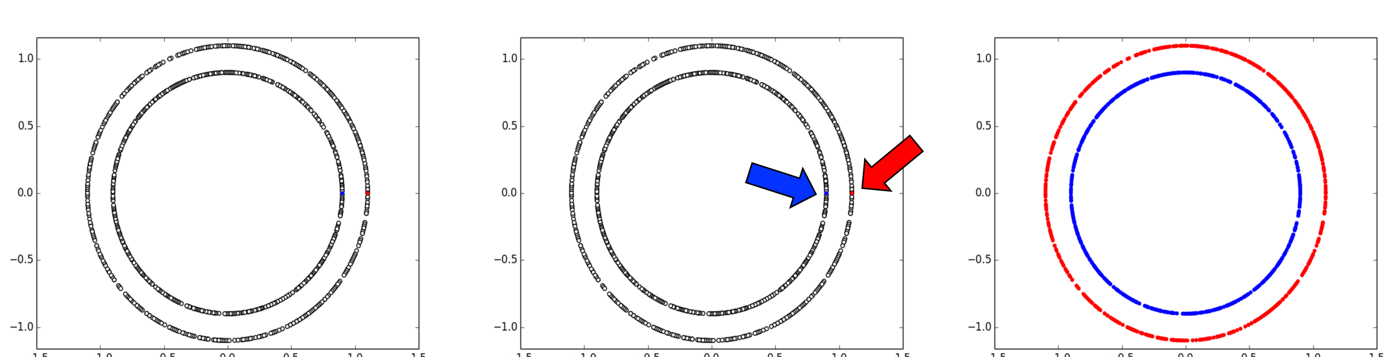
- Offline algorithm for semi-supervised learning
- *(Zhu, Ghahramani, Lafferty, ICML 2003)*
- *ICML 10-Year Classic Paper Award, 2013*

Algorithm:

- Build graph on data points
- Propagate labels by minimizing electric energy



Example 1: Separate two rings



input   *labeled examples*   *Label Propagation*

Example 2: Coloring grayscale image

(Levin, Lischinski, Weiss, SIGGRAPH 2004)

input   *labeled examples*   *Label Propagation*

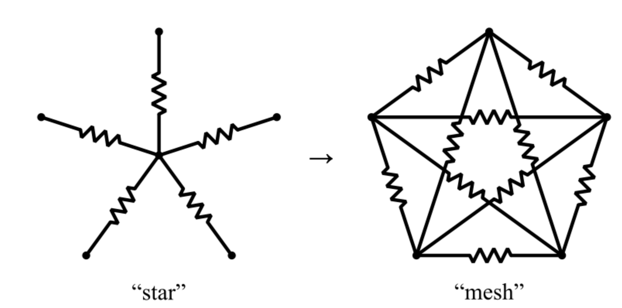
Graph Compression

Intermediate goal: Compress irrelevant nodes in graph

- "Irrelevant" means we do not need to classify them anymore
- E.g., old points on the stream
- **Problem:** Relevant labels still depend globally on irrelevant nodes
- **Solution:** Encode irrelevant nodes into edge weights

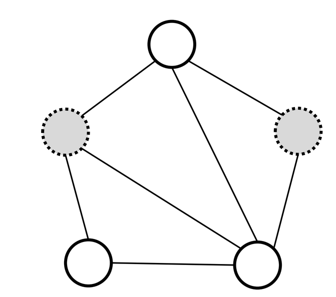
Star-Mesh Transform

- Known in electric network theory *(Campbell 1922, Rosen 1924)*
- Removes one irrelevant node
- Updates edge weights to encode it



Sequential Star-Mesh Transforms

- **Input:** Large graph with n nodes, only $\tau < n$ are relevant

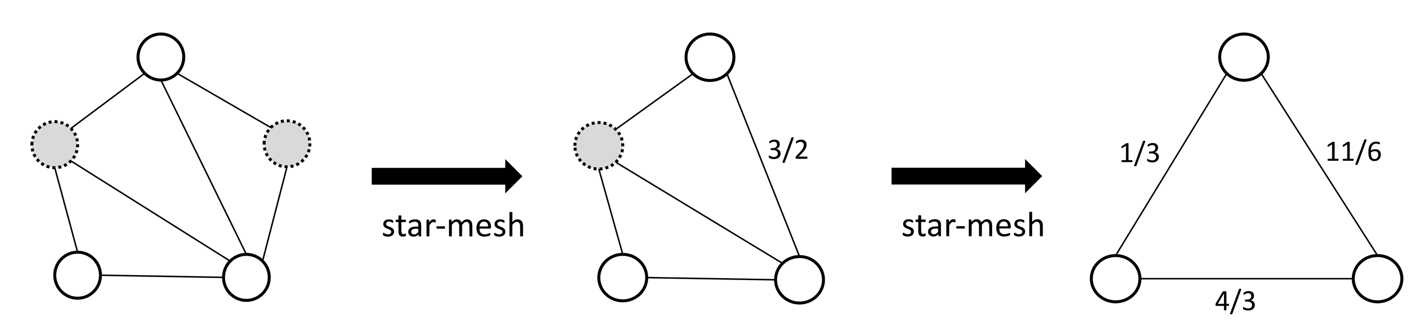


Relevant node

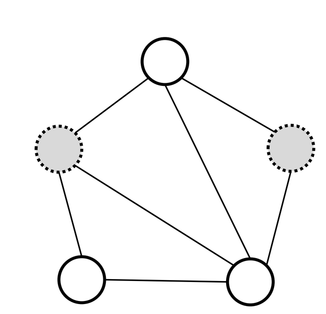
Irrelevant node

Assume edge weights are unit for demonstration (this also works with arbitrary edge weights)

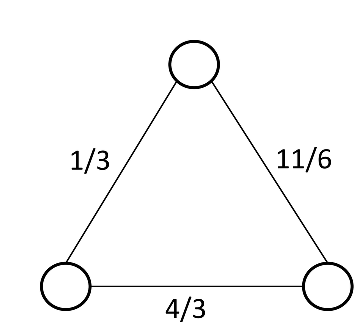
- Compress irrelevant nodes one by one by star-mesh transforms:



- **Compression:**

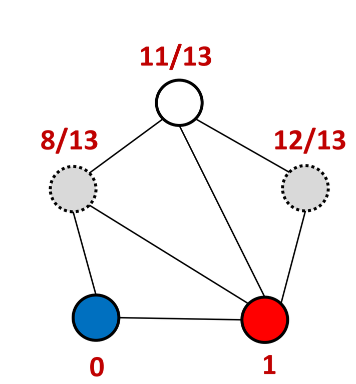


Size before: $\Omega(n \log n)$ bits

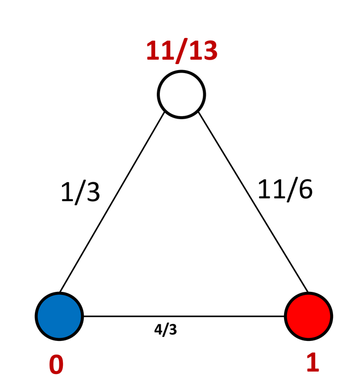


Size after: $O(\tau^2 \log n)$ bits

- **Theorem:** Label Propagation is preserved on the relevant nodes



Label Propagation before



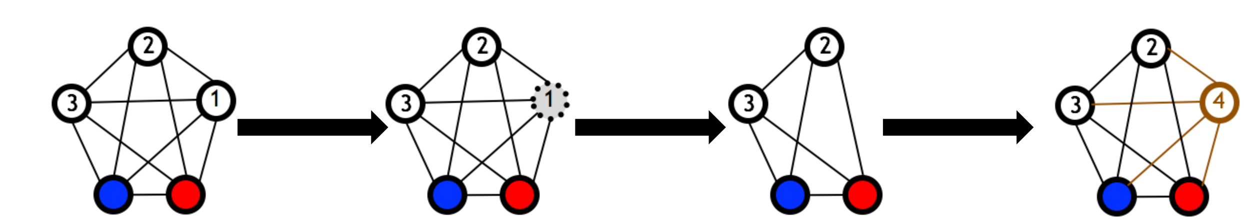
Label Propagation after

- This works for any graph.
- However, sequential star-mesh transforms are as slow as Label Propagation on the entire original graph, so we get no improvement for offline data.
- **We will use it for streaming data.**

Streaming Algorithm

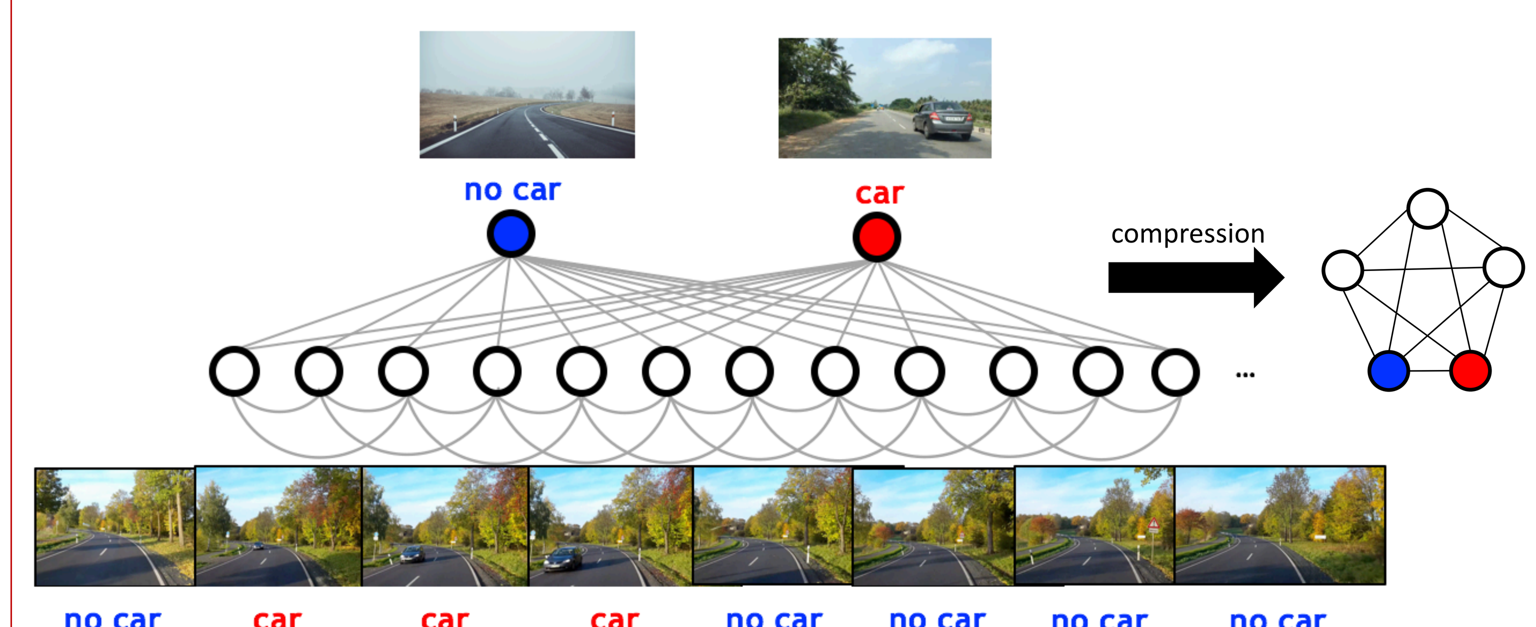
Temporal Label Propagation:

- Pick small integer parameter τ
- Store the most recent τ unlabeled points on the stream
- **Upon new unlabeled point arrival:**
  - Compress oldest unlabeled point by star-mesh transform
  - Insert new point
  - Propagate labels



Guarantees for step n:

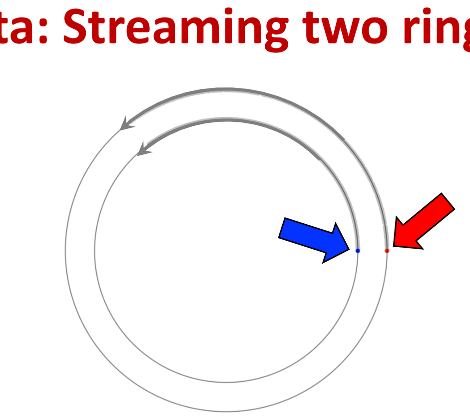
- **Running time:** $\tilde{O}(\tau^3)$
- **Memory:** $\tilde{O}(\tau^2 \log n)$
- **Output:** Label Propagation on a *"temporal vicinity graph"* that contains the entire stream seen so far:



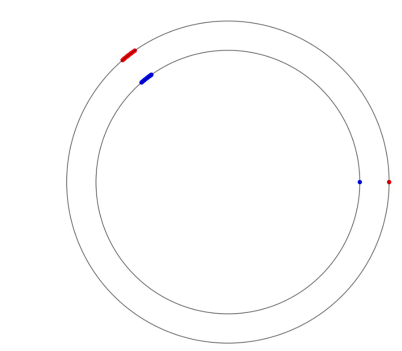
"Temporal vicinity graph": Suitable for temporally-ordered streams (e.g. time-series, video, …)

Experiments

Toy data: Streaming two rings



Input stream runs counter-clockwise from the labeled points at x-axis

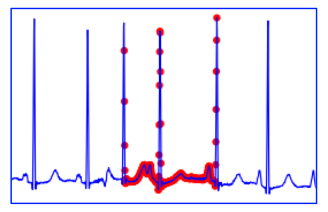


Temporal Label Propagation classifies the entire stream correctly

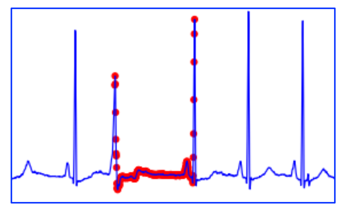
Real data: Classifying irregular heartbeats on EKG signal

- **Task:** Classify *atrial* vs. *ventricular* premature contractions

atrial premature contraction



ventricular premature contraction



- Accuracy: **94.9%** with $\tau = 5$ and **2** labeled examples per class.
- *Data source: physionet.org. Normal heartbeats were ignored.*