Semi-Supervised Learning on Data Streams via Temporal Label Propagation

Tal Wagner [1] **Sudipto Guha** [2] **Shiva Prasad Kasiviswanathan** [2] **Nina Mishra** [2]

Abstract

We consider the problem of labeling points on a fast-moving data stream when only a small number of labeled examples are available. In our setting, incoming points must be processed efficiently and the stream is too large to store in its entirety. We present a semi-supervised learning algorithm for this task. The algorithm maintains a small synopsis of the stream which can be quickly updated as new points arrive, and labels every incoming point by provably learning from the full history of the stream. Experiments on real datasets validate that the algorithm can quickly and accurately classify points on a stream with a small quantity of labeled examples.

1. Introduction

In many real situations, unlabeled data is readily available, whereas labeled data tends to be of a smaller size. This motivates semi-supervised learning (SSL), which aims to make heavy use of a large amount of unlabeled data along with a limited amount of labeled data. For this problem, the celebrated paper due to (Zhu et al., 2003a) provided a principled offline approach with excellent results in practice. Their algorithm casts the problem as label propagation on a graph, where the nodes represent both labeled and unlabeled data points, and the weight of an edge reflects similarity between its endpoints. The labels are spread in the graph by a random walk process that moves through the unlabeled nodes until reaching a labeled node. The labeling computed by this process is known as the *harmonic solution*.

In this paper, we consider the case where the data arrives in a high-throughput stream, such as an electrocardiogram signal or a video feed. The goal is to label each point upon arrival as quickly as possible, ideally by means of semi-supervised learning over all of the data seen so far, both labeled and unlabeled. Example use cases include real-time monitoring of metrics arising from medical patient signals (ECG, EEG, fall detection), data centers (network, I/O and CPU utilization), or a camera mounted on a semi-autonomous car (for road conditions and obstacle detection). In these scenarios, unlabeled data is continuously streaming, but only a small number of manually labeled examples are provided – either at the beginning of the stream or as occasional user feedback. We want algorithms that leverage both inputs and learn how to classify stream elements, such as ECG arrhythmias, network intrusion alerts or driving conditions. Several other applications are given in (Goldberg et al., 2008), who defined a similar model, and in (Krempl et al., 2014).

In practice, this setting requires algorithms that run under severe time and memory constraints, since the labels are expected in real-time and the stream is generally too large to fully store in the memory. This poses a major challenge: How can we leverage the entire stream history to label a new point, when we can only store a tiny fraction of it?

Problem Statement. *Given a stream $x_1, x_2, \ldots$ of interleaved labeled and unlabeled points, and a similarity function between pairs of points, label every incoming point x_n using sublinear time and sublinear space in n.*

Our Solution. Our main contribution is Temporal Label Propagation (TLP), a streaming SSL algorithm which is theoretically sound and also works well in practice. Its processing time for the nth point on the stream is independent of n, and its storage space only scales as $\log n$. At the same time, it provably computes the harmonic solution on a similarity graph that naturally describes the entire stream seen so far, which we call a *temporal vicinity graph*. Thus, it produces labels that utilize all of the labeled and unlabeled points in the past. In comparison, using a batch (offline) label propagation algorithm on the same graph would entail computation and memory requirements that grow at least as a linear function of the stream length n.

Our Techniques. The algorithm is based on graph reduction tools that originate in the theory of electric networks. The short-circuit operator (Campbell, 1922; Anderson, 1971) is a way to compress a large graph $\mathcal{G}$ into a much smaller graph $\mathcal{H}$, that exposes only pre-specified nodes of interest called *terminals*, while preserving some global properties of $\mathcal{G}$. We choose the terminals as the most recent points on the stream, including the incoming point

[1]CSAIL, MIT. Work done during an internship at Amazon. [2]Amazon. Correspondence to: Tal Wagner <talw@mit.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

that we need to classify. Drawing on the electric interpretation of the harmonic solution (Snell & Doyle, 2000), we rigorously show that the labels of the terminals in $\mathcal{G}$ can be computed directly from $\mathcal{H}$. A related graph operation, known as the star-mesh transform (Rosen, 1924), enables us to maintain the compressed graph $\mathcal{H}$ of the temporal vicinity graph over the stream by a sequence of simple local updates.

We evaluate our solution on several real datasets. Our results demonstrate the advantage of TLP over alternative methods.

1.1. Related Work

Semi-supervised learning is a well-established field, and a comprehensive overview is available in (Zhu, 2005; Chapelle et al., 2009). The type of SSL algorithms that we explore are graph-based, which have a long history of work, including (Blum & Chawla, 2001; Szummer & Jaakkola, 2002; Zhu et al., 2003a; Joachims, 2003; Zhou et al., 2004; 2005; Belkin et al., 2004; 2005; Wang et al., 2008).

Graph construction is an important issue. Some prior approaches make deep use of domain knowledge (Levin et al., 2004; Balcan et al., 2005), while others construct general purpose graphs (Zemel & Carreira-Perpiñán, 2005; Wang & Zhang, 2008; Jebara et al., 2009; Ghazvininejad et al., 2011). This topic also plays a role in our paper, as we rely on a graph construction that is suitable for temporal streams.

Graph-based SSL algorithms typically do not scale well with the data size n, often requiring $\Omega(n^2)$ computation time or worse. Some authors suggested representing the graph by a smaller "backbone" graph on which label propagation can be performed much faster (Zhu & Lafferty, 2005; Delalleau et al., 2005; Valko et al., 2010). Our work takes a related approach through the construction of our compressed graph $\mathcal{H}$, but our compression based on the short-circuit operator utilizes completely different ideas from these prior results.

Most of the algorithms mentioned above were designed for the offline setting. Online SSL is a relatively new field that has generated considerable interest (Zhu et al., 2009; Krempl et al., 2014). Online graph-based algorithms were proposed in (Huang et al., 2015) and (Ravi & Diao, 2016). They are applicable to points arriving on a stream, but the processing time and memory for the nth point is still $\Omega(n)$, which is problematic as n grows. Non-graph-based online SSL algorithms were given in (Goldberg et al., 2008; 2011; Dyer et al., 2014).

A related issue is transduction vs. induction. Most graph-based SSL algorithms are transductive, which means the unlabeled data is fully given to them in advance. Inductive algorithms can also label new test points (Zhu et al., 2003b; Sindhwani et al., 2005; Delalleau et al., 2005). However, they do not use the new points to learn how to label future points, which is a desired goal in online/streaming SSL.

Closest in spirit to our work is (Valko et al., 2010) which operates within similar time and memory constraints. Their algorithm quantizes the stream into a small number of k clusters via the online k-center algorithm of (Charikar et al., 1997). A regularized harmonic solution is then computed on the cluster centers. We experimentally compare this algorithm to our approach in Section 6.

2. Preliminaries

Notation. Graphs discussed in this paper are weighted undirected and we denote them by calligraphic letters. Vectors will be written in boldface letters. Let $\mathcal{G} = (V, E, w)$ be a graph with $|V| = n$ and non-negative edge weights $\{w_{x,y} : (x, y) \in E\}$. The (weighted) degree of a node x is $\deg(x) = \sum_{y:(x,y)\in E} w_{x,y}$.

Let $G \in \mathbb{R}^{n\times n}$ be the Laplacian matrix of the graph $\mathcal{G}$. Let

$$G = \begin{bmatrix} G_{aa} & G_{ab} \\ G_{ba} & G_{bb} \end{bmatrix}$$

be a block partition corresponding to a partition $V = V_a \cup V_b$ of the node set. Note that $G_{ba} = G_{ab}^\top$. It is well-known that if $\mathcal{G}$ is connected then G_{aa} is invertible. In that case let G/G_{aa} denote the Schur complement, i.e., $G/G_{aa} = G_{bb} - G_{ba}G_{aa}^{-1}G_{ab}$. Appendix A reviews some additional background on graph matrices.

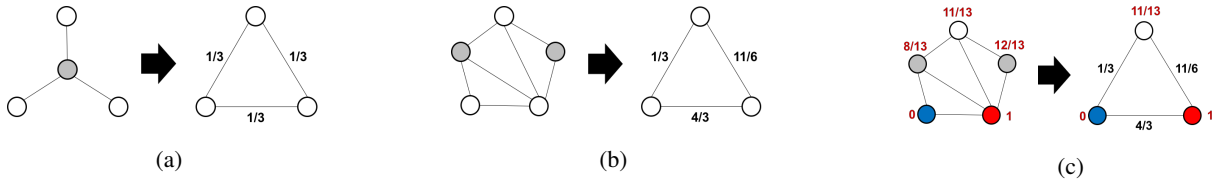
Offline Label Propagation. We review the algorithm of (Zhu et al., 2003a), as it forms the basis of our approach for learning on the stream. For simplicity, we describe the binary classification setting with labels 1 (positive class) and 0 (negative class). The input to the label propagation algorithm is a weighted undirected graph $\mathcal{G} = (V, E, w)$, in which a small subset of nodes $V_l \subset V$ are labeled and the rest $V_u \subset V$ are unlabeled. The weight of an edge (x, y) represents some measure of similarity between its endpoints. The goal is to compute fractional labels in $[0, 1]$ for the unlabeled nodes that would facilitate a good partition into a 0-set and a 1-set.

The vector of fractional labels is denoted by $\mathbf{f} \in \mathbb{R}^V$, with $\mathbf{f}(x)$ representing the fractional label of $x \in V$. We separate $\mathbf{f}$ into two parts $\mathbf{f}_u \in \mathbb{R}^{V_u}$ and $\mathbf{f}_l \in \mathbb{R}^{V_l}$ according to the partition $V = V_u \cup V_l$. The part $\mathbf{f}_l$ is given as input, and $\mathbf{f}_u$ is the part we need to compute. The algorithm computes $\mathbf{f}_u$ by minimizing the following *energy function* of the graph:

$$\min_{\mathbf{f}_u} \ \frac{1}{2} \sum_{(x,y)\in E} w_{x,y}(\mathbf{f}(x) - \mathbf{f}(y))^2. \qquad (1)$$

This is equivalent to minimizing $\frac{1}{2}\mathbf{f}^\top G\mathbf{f}$ under the given part $\mathbf{f}_l$, where G is the Laplacian of $\mathcal{G}$.

Harmonic Solution. The minimizer $\mathbf{f}^{\mathcal{G}}$ of Equation (1) is called the *harmonic solution*. Since the objective is a



(a) (b) (c)

Figure 1. The short-circuit operator on graphs. The left graphs in (a), (b), (c) represent $\mathcal{G}$ in Definition 2.1. They have unit edge weights and their terminals are the non-shaded nodes. The graphs on the right represent $\mathcal{G}\langle V_t\rangle$. Their edge weights appear in black. (a) Star-mesh transform. (b) Short-circuiting two nodes at once. (c) The bottom two nodes are labeled. The harmonic solution (where blue $= 0$ and red $= 1$) is denoted by red numbers on each node. It is the same on the top node before and after short-circuiting, as per Theorem 4.1.

convex quadratic, the gradient at $\mathbf{f}^{\mathcal{G}}$ is zero. This yields the harmonic property of $\mathbf{f}^{\mathcal{G}}$, which is that the value at each unlabeled node equals the average at its adjacent nodes:

$$\mathbf{f}^{\mathcal{G}}(x) = \frac{1}{\deg(x)} \sum_{y:(x,y)\in E} w_{x,y}\mathbf{f}^{\mathcal{G}}(y) \quad \forall\, x \in V_u. \tag{2}$$

By solving this linear system we get a closed-form formula for the unknown part $\mathbf{f}_u^{\mathcal{G}}$:

$$\mathbf{f}_u^{\mathcal{G}} = -G_{uu}^{-1}G_{ul}\mathbf{f}_l. \tag{3}$$

Merging Labeled Nodes. In case there are more than a single labeled node for a class, we can symbolically merge them in $\mathcal{G}$ to just one labeled node per class. It does not affect the harmonic solution on all the remaining unlabeled nodes is (cf. Appendix B.1). We use v_0^* as the node formed by merging the 0-class in V_l, and v_1^* as the node merging the 1-class in V_l. Weights of parallel edges are summed.

2.1. Electric Networks

Our approach draws on the connection between label propagation and the theory of electric networks, which was described in (Zhu et al., 2003a) following (Snell & Doyle, 2000). View the similarity graph $\mathcal{G}$ as an electric network where every edge (x, y) is a resistor with conductance $w_{x,y}$. Connect a +1V voltage source to all nodes in V_l labeled with 1, and a ground source (0V) to all nodes in V_l labeled with 0. The potentials induced at the unlabeled nodes are equal to the harmonic solution.

Short-Circuit Operator. Suppose we have a large network $\mathcal{G}$ with a small subset V_t of distinguished nodes, called *terminals*. The short-circuit operator allows us to encode $\mathcal{G}$ into a smaller network $\mathcal{G}\langle V_t\rangle$ whose only nodes are the terminals.

Definition 2.1. *Let $\mathcal{G} = (V, E, w)$ be a connected graph with a partition $V = V_t \cup V_s$ of its nodes. The short-circuit operator produces a re-weighted graph $\mathcal{G}\langle V_t\rangle = (V_t, E', w')$, defined by the following operation. Let G denote the Laplacian matrix of $\mathcal{G}$. The Schur complement $G/G_{ss} = G_{tt} - G_{ts}G_{ss}^{-1}G_{st} \in \mathbb{R}^{|V_t|\times|V_t|}$ is a Laplacian matrix of a graph on the nodes V_t (see Appendix A), and this graph is the short-circuit graph $\mathcal{G}\langle V_t\rangle$.*

See Figure 1b for illustration. We refer the reader to (Dorfler & Bullo, 2013) for a more comprehensive study of this useful notion. $\mathcal{G}\langle V_t\rangle$ is known to retain certain global electric properties of $\mathcal{G}$; most famously, it preserves the effective resistance between every pair of terminals. The aforementioned connection to the harmonic solution suggests that $\mathcal{G}\langle V_t\rangle$ could be useful for label propagation, and we will prove this is indeed the case.

Star-Mesh Transform. Generally, computing $\mathcal{G}\langle V_t\rangle$ is as expensive as computing the harmonic solution on all of $\mathcal{G}$, since both entail inverting a large Laplacian submatrix. Therefore, it provides no substantial speed-up in the offline setting. However, $\mathcal{G}\langle V_t\rangle$ can also be computed by a sequence of local operations, known as star-mesh transforms. This will be useful for the streaming setting.

Definition 2.2. *The star-mesh transform on a node x_o in a graph $\mathcal{G} = (V, E, w)$ is the following operation:*

1. *"Star": Remove x_o from $\mathcal{G}$ with its incident edges.*
2. *"Mesh": For every pair $x, x' \in V$ such that $(x, x_o) \in E$ and $(x', x_o) \in E$, add the edge (x, x') to E with weight $w_{x_o,x}w_{x_o,x'} / \deg(x_o)$. If (x, x') is already in E then add the new weight to its current weight.*

This is in fact a special case of the short-circuit operator, with a single non-terminal x_o (see Figure 1a). It is known that $\mathcal{G}\langle V_t\rangle$ can be computed by sequential star-mesh transforms on the non-terminals $V_s = V \setminus V_t$ in $\mathcal{G}$ in an arbitrary order. This is a direct consequence of the sequential property of Schur complements (cf. (Zhang, 2005), Theorem 4.10; see also (Dorfler & Bullo, 2013), Lemma III.1).

3. The Streaming Algorithm

Consider a data stream $\{x_i\}_{i=1}^{\infty}$ in which some points are labeled and most of the points are unlabeled. Our streaming algorithm **Temporal Label Propagation** for binary labels[1] is presented in Algorithm TLP. It maintains a graph $\mathcal{H}$ that contains the most recent τ unlabeled points, plus two labeled nodes v_0^* and v_1^* that represent all of the labeled points.

[1]The extension to multiple labels appears in Appendix B.2.

When a new unlabeled point arrives, we add it to $\mathcal{H}$ and evict the oldest unlabeled point by a star-mesh transform, thus always maintaining $\tau+2$ nodes. The harmonic solution for the new point is then computed on $\mathcal{H}$.

Let us give some intuition for the role of the star-mesh transform in the algorithm. The premise of graph-based label propagation is that an unlabeled point x_o provides useful information on the structure of the dataset as encoded by its incident edge weights. The star-mesh transform removes those edges, but meshes their weights with the remaining graph, so that the information provided by x_o remains encoded. As a result, while we lose the ability to compute the harmonic solution for x_o, we retain the ability to compute it for the rest of the nodes as if x_o were still in the graph. In the consumer/provider terminology of Section 4, x_o is removed from the graph as a consumer but remains a provider. This intuition is made rigorous in Theorem 4.1.

The computation time of Algorithm TLP for x_n is independent of n, and the space consumption scales only as $\log n$. At the same time, the fractional label computed for x_n is provably equal to the value of its harmonic solution on a suitable similarity graph associated with the entire stream seen so far (we call it the *temporal vicinity graph* and define it in Section 5.1). Thus, it performs label propagation through all of the data from the past, both labeled and unlabeled. These properties will be stated formally in Theorem 5.3.

Algorithm TLP : Temporal Label Propagation

Initialization

Parameters: integer $\tau > 0$, similarity measure $\mathrm{Sim} : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_{>0}$ where $\mathbb{X}$ is the domain of inputs
$L_0 \leftarrow \emptyset$ *// set of* 0*-labeled points*
$L_1 \leftarrow \emptyset$ *// set of* 1*-labeled points*
$\mathcal{H} = (V_h, E, w) \leftarrow$ graph with $V_h = \{v_0^*, v_1^*\}$, $E = \emptyset$
// v_0^* and v_1^* are nodes labeled with 0 *and* 1 *respectively*

On receiving a point $x_n \in \mathbb{X}$ **labeled** $b \in \{0,1\}$

$L_b \leftarrow L_b \cup \{x_n\}$ *// Merge x_n into v_b^**
for x **in** V_h **:**
  $w_{v_b^*,x} \leftarrow w_{v_b^*,x} + \mathrm{Sim}(x_n, x)$

On receiving an unlabeled point $x_n \in \mathbb{X}$

$V_h \leftarrow V_h \cup \{x_n\}$ *// Add x_n to $\mathcal{H}$*
Add (x_n, v_0^*) to E with weight $\sum_{x \in L_0} \mathrm{Sim}(x_n, x)$
Add (x_n, v_1^*) to E with weight $\sum_{x \in L_1} \mathrm{Sim}(x_n, x)$
for x **in** $V_h \setminus \{v_0^*, v_1^*, x_n\}$ **do**
  Add (x_n, x) to E with weight $\mathrm{Sim}(x_n, x)$
if $|V_h| \geq \tau + 2$ **then** *// Star-mesh transform*
  $x_o \leftarrow$ oldest node in $V \setminus \{v_0^*, v_1^*\}$
  Remove x_o from $\mathcal{H}$
  for all pairs $x \neq x'$ **in** V_h **do**
    $w_{x,x'} \leftarrow w_{x,x'} + \dfrac{w_{x_o,x} w_{x_o,x'}}{\deg(x_o)}$
$\mathbf{f} \leftarrow$ harmonic solution on $\mathcal{H}$ *// Label Propagation*
return $\mathbf{f}(x_n)$

4. Compression by Short-Circuiting

The essence of a streaming algorithm is in maintaining a compressed representation of the stream, from which the desired output can still be computed. In our case, the desired output is the harmonic solution of the incoming point.

The challenge here is two-fold since the algorithm needs to not only compress the data, but also update the compressed representation as new points arrive. We handle the two issues separately: in the current section we present an offline (non-streaming) compression technique, which applies to a more general setting of label propagation on arbitrary graphs. It is useful for saving space, but does not yield faster running time. Section 5 will show how to adapt it to streaming data while achieving fast processing time per point. This will necessitate choosing a specific graph construction which is suitable for data streams.

Consumers and Providers in SSL. The compression scheme we utilize is based on the following reasoning. In graph-based SSL, every unlabeled node plays a dual role: it is both a *consumer* whose own label needs to be computed, and a *provider* that participates in computing the labels of other nodes. Batch algorithms for label propagation do not make this distinction – they compute labels for the entire graph in one computation, rendering each unlabeled node both a consumer and a provider. However, the distinction can be useful when we only need to label one or few nodes, as labeling the entire graph is redundant and potentially wasteful. This will be relevant for the streaming setting in Section 5, since a streaming algorithm only needs to label the incoming point at each time step, so that point is the only consumer. However, its label should ideally depend globally on all of the past points, so they are all desired providers.

Our approach is to refine the representation of the input so as to encode the non-consumers only by their provider role. Ideally this would allow for a substantially more efficient representation. We implement this idea for the harmonic solution in the offline setting by using the short-circuit operator (Definition 2.1), as formalized next.

Let $\mathcal{G} = (V, E, w)$ be an arbitrary connected graph with a partition $V = V_l \cup V_u$ into labeled and unlabeled nodes. Let $V_c \subset V_u$ be the subset of consumer nodes, i.e., those for which we wish to compute the harmonic solution. We are interested in the case where $|V_c| \ll |V|$. The consumer labels depend globally on $\mathcal{G}$, and hence all nodes in V are designated as providers. We define the terminal set as $V_t = V_c \cup V_l$, to include both the consumers and the labeled

nodes. Let $\mathcal{H} = \mathcal{G}\langle V_t \rangle$ be the result of short-circuiting the non-terminals in $\mathcal{G}$. Our main technical result is that the harmonic solution of all consumers in $\mathcal{G}$ is preserved in $\mathcal{H}$.

Theorem 4.1. *Let $\mathbf{f}_l$ be a vector of given labels for V_l. Let $\mathbf{f}^{\mathcal{G}}$ and $\mathbf{f}^{\mathcal{H}}$ be the harmonic solutions on $\mathcal{G}$ and $\mathcal{H}$ respectively. Then $\mathbf{f}^{\mathcal{G}}(x) = \mathbf{f}^{\mathcal{H}}(x)$ for every consumer $x \in V_c$.*

See Figure 1c for an illustration. As a result, the harmonic solution for every consumer in $\mathcal{G}$ can be computed by label propagation on $\mathcal{H}$. At the same time, $\mathcal{H}$ has only $|V_t|$ nodes and is significantly cheaper to store:

Proposition 4.2. *Let ω be the ratio of maximum to minimum edge weights in $\mathcal{G}$. The storage size of $\mathcal{H}$ is $O(|V_t|^2(\log|V| + \log\omega))$ bits.*

In comparison, the storage size of $\mathcal{G}$ is $O(|E|(\log|V| + \log\omega))$ bits, and since $\mathcal{G}$ is connected, $|E| = \Omega(|V|)$. Hence the dependence on $|V|$ is improved exponentially from $|V|$ (in $\mathcal{G}$) to $\log|V|$ (in $\mathcal{H}$).

Full details from this section are collected in Appendix C. Theorem 4.1 and Proposition 4.2 are proven in Sections C.4 and C.5 respectively.

Remark 4.3. Let us put Theorem 4.1 in the context of known results. It is well-known that for every $\mathbf{f}_l$, the energy of the harmonic solutions (cf. eq. (1)) on $\mathcal{G}$ and $\mathcal{H}$ is the same, i.e., $\frac{1}{2}(\mathbf{f}^{\mathcal{G}})^\top G \mathbf{f}^{\mathcal{G}} = \frac{1}{2}(\mathbf{f}^{\mathcal{H}})^\top H \mathbf{f}^{\mathcal{H}}$. However, for the purpose of SSL, it is not enough to just preserve energy since classification relies on the value of $\mathbf{f}^{\mathcal{G}}$ at specific nodes. We therefore require the more general fact that the harmonic solutions are equal at each unlabeled node that appears in both $\mathcal{G}$ and $\mathcal{H}$, namely $\mathbf{f}^{\mathcal{G}}(x) = \mathbf{f}^{\mathcal{H}}(x)$ for every $x \in V_c$.

5. TLP Analysis

In the previous section we presented a compression scheme for label propagation on an arbitrary graph $\mathcal{G}$ in the form of a smaller re-weighted graph $\mathcal{H}$. However, compression by itself does not yield a streaming algorithm, as we also need to efficiently update $\mathcal{H}$ along the stream. To this end, we introduce a suitable similarity graph for streaming data.

5.1. The Temporal Vicinity Graph

Let $\mathbb{X}$ denote the domain of the inputs and let $\mathrm{Sim} : \mathbb{X}\times\mathbb{X} \to \mathbb{R}_{>0}$ be an associated similarity measure.

Definition 5.1. *Let $\tau > 0$ be an integer, and $x_1, x_2, \ldots \in \mathbb{X}$ be a stream of labeled/unlabeled data points. The temporal vicinity graph up to timestep n, denoted by $\mathcal{G}_\tau^{(n)}$, is the following graph:*

- *Nodes: The node set of $\mathcal{G}_\tau^{(n)}$ is $\{x_1, \ldots, x_n\}$.*
- *Edges: Every point (either labeled or unlabeled) is adjacent to the previous τ unlabeled points, and to all the previous labeled points.*
- *Weights: For every adjacent pair x_i, x_j, the edge weight between them is $w_{x_i,x_j} = \mathrm{Sim}(x_i, x_j)$.*

Note that every incoming point x_n defines a new graph $\mathcal{G}_\tau^{(n)}$, which contains the previous graph $\mathcal{G}_\tau^{(n-1)}$ as its subgraph.

Let us explain the effect of this graph on the output labeling. When choosing a graph construction for label propagation, the placing of edges encodes a structure over which the solution will be smooth. This means that adjacent nodes will tend to have similar labels, as seen in Equation (1). One can either opt for global smoothness (say, by placing all possible edges), or incorporate domain knowledge.

Temporal vicinity promotes smoothness over consecutive points in the stream. This is suitable for inherently ordered data, where the context of each point is relevant for its labeling. For example, in an electrocardiogram feed, each arrhythmia lasts several timesteps, and hence most consecutive points have the same groundtruth label. Thus we should favor smoothness across temporally adjacent points.

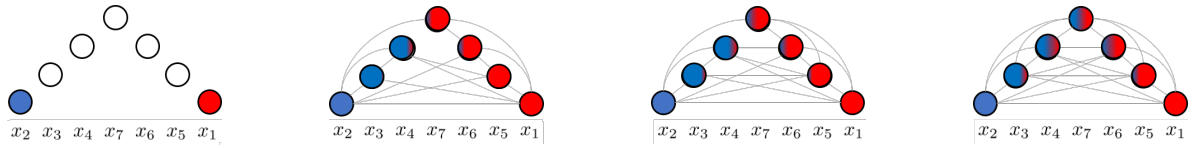
A useful analogy is the *spatial vicinity* graph structure, which was used by (Levin et al., 2004) for propagating colors through pixels in a grayscale image. Their graph contains edges only between neighboring pixels, and the edge weights reflect the similarity in grayscale intensity of the connected pixels. The rationale is that neighboring pixels are expected to have similar colors, in accordance to their grayscale intensities, while distant pixels need not have similar colors even if they have similar grayscale intensities.

The choice of parameter τ is data-dependent. Intuitively, it should capture the context span of each point in the stream – or how much the immediate past matters more than the distant past. This is illustrated in Figure 2. In particular, increasing τ does not monotonically improve the accuracy, although it increases the time and space complexity of TLP.

Our analysis also supports several variations to the graph construction, including connecting each point to the previous τ points (regardless of whether they are labeled or unlabeled) or gradually decaying older edge weights over time. For concreteness we opt for the variant defined above, which connects every unlabeled point to all the previous labeled points. This emphasizes the labeled data without adversely affecting the running time of TLP, which is governed by the number of nodes and edges in $\mathcal{H}$. In particular, $\mathcal{H}$ remains a weighted clique on $\tau + 2$ nodes regardless of how the labeled and unlabeled nodes are connected.

5.2. Theoretical Guarantees of TLP

The key property of the temporal vicinity graph is that on one hand it is a suitable graph construction for data streams, while on the other hand we can maintain a compressed representation of it along the stream by sequential star-mesh



(a) Data stream (b) $\tau = 1$ (c) $\tau = 2$ (d) $\tau = 4$ (clique)

Figure 2. (a) A stream of points $x_1, x_2, \ldots, x_7$ on the plane. x_1 and x_2 are labeled with red and blue. The rest of the points are unlabeled. If the stream order is ignored, then x_7 is equally likely to be blue or red. If the order matters, then x_7 is more likely to be red, since it appears in the context of x_5, x_6 which form a smooth path towards x_1. (b),(c),(d) show the edges of the temporal vicinity graph for $\tau = 1, 2, 4$. The edge weights reflect the Euclidean distances and are omitted. The unlabeled points are colored by their blue/red fractional labels in the harmonic solution. As τ increases, the effect of the distant past on x_7 becomes more expressed, and it is less likely to be red.

transforms. Formally, at every timestep n, the graph $\mathcal{H}$ in Algorithm TLP is the result of the short-circuit operator on $\mathcal{G}_\tau^{(n)}$ with the terminals V_h. Note that V_h is the node set of $\mathcal{H}$ in Algorithm TLP, and is a subset of the nodes in $\mathcal{G}_\tau^{(n)}$.

Claim 5.2. *At every timestep n, $\mathcal{H} = \mathcal{G}_\tau^{(n)}\langle V_h \rangle$.*

In conjunction with Section 4, we obtain the following guarantees for Algorithm TLP. Let $\chi = |\mathbb{X}|$ be the data domain size and let ω be the magnitude of similarities.[2] For every n let n_l denote the number of labeled points up to timestep n. Note that $n_l \ll n$ in the semi-supervised setting.

Theorem 5.3. *For every timestep n in the stream, let x_n be the new received point. Algorithm* TLP *satisfies:*

1. *The fractional label $\mathbf{f}(x_n)$ returned by Algorithm* TLP *is equal to the harmonic solution for x_n on $\mathcal{G}_\tau^{(n)}$.*
2. *The computation time at timestep n is $O(\tau^3 + n_l)$.*
3. *The storage size at timestep n is $O(\tau^2 \log(n\omega) + (n_l + \tau)\log\chi)$ bits.*

The proof appears in Appendix D. To appreciate the effect of compression, note that computing the harmonic solution directly on $\mathcal{G}_\tau^{(n)}$ in timestep n would require $\Omega(n\tau^2)$ time and $\Theta(n\tau\log(n\omega) + n\log\chi)$ storage space. In particular, both are linear in the stream length n.

6. Experimental Evaluation

In this section, we experimentally demonstrate the effectiveness of our temporal label propagation scheme on data streams ranging from medical to computer vision domains.

Compared Methods. We compare Algorithm TLP with the following approaches:

Sliding Window Label Propagation (SWLP): One simple approach to label propagation on a stream is to store in memory only the recent τ unlabeled points in a sliding window fashion, in addition to all the labeled data. This approach ignores the past entirely, while trivially yielding a small memory footprint. This algorithm is identical to TLP except that the star-mesh transform is replaced by usual node deletion. It thus lets us directly evaluate the effectiveness of the short-circuiting operation on the stream.

Quantized Label Propagation (QLP): The streaming algorithm of (Valko et al., 2010), which was described in Section 1.1. Here τ denotes the number of cluster centroids.

Inductive Label Propagation (ILP): The inductive SSL algorithm of (Delalleau et al., 2005). It performs label propagation on a given training set of labeled and unlabeled nodes, and subsequent test points are then labeled by a kernel regression step. However, the new test points are not incorporated into the trained model. To apply this algorithm to a stream, we use the first τ unlabeled points along with the labeled data as the training set.

Note that each of the four algorithms works by choosing τ unlabeled points to fully store in memory and to compute the harmonic solution over. This determines both the memory footprint and the computation time of label propagation, since it is computed by inverting a $\tau \times \tau$ Laplacian submatrix (cf. Equation (3)). The algorithms vary in their choice of the τ points. Therefore, while the overall budget is matched, the algorithms choose their own utilization of the budget, permitting a direct comparison.

In addition, we include the following baseline:

Labeled-only Label Propagation (LOLP): This algorithm labels every incoming point on the stream based only on the labeled examples, without taking any unlabeled points into account. We use this non-SSL baseline to evaluate the advantage of making use of the unlabeled points.

We start with a visual demonstration of the three algorithms on a toy dataset, and proceed to experiments on real data.

6.1. Visual Demonstration

A common demonstration of (offline) label propagation is the two-rings setting, depicted in Figure 3a. In this setting,

[2] $\omega = \max_{x,x' \in \mathbb{X}} \mathrm{Sim}(x, x') / \min_{x,x' \in \mathbb{X}} \mathrm{Sim}(x, x')$. A point takes $\log\chi$ bits and a similarity value takes $\log\omega$ bits to store.

the data points are densely organized on two concentric circles, and each circle has only a single labeled point at its intersection with the x-axis (shown in red on the outer circle and blue on the inner circle). Offline label propagation classifies the circles correctly, as shown in Figure 3b.

To adapt this example to the streaming setting, we turn each circle into a stream by ordering its points counterclockwise, starting from the labeled point on the x-axis. The two streams generated from the two circles are interweaved at random and presented to the algorithms as a single stream of data. The goal is to label the outer circle as red and the inner circle as blue.

Figure 3 shows of the behavior of the methods we compare. For each method we present snapshots from three points of time on the stream (ordered left-to-right from early to late). Each snapshot shows the τ points stored in memory (we use $\tau = 40$) and colored according to their labeling by the algorithm at that moment. The full videos are included in the supplementary material of this paper (see Appendix F).

TLP (Figure 3c) stores the τ most recent points on the stream, forming two "caterpillars" that crawl along the circles as new points arrive and old points get evicted. The history of the stream is encoded in the edge weights between the stored points by star-mesh transforms, so the algorithm "remembers" the paths traversed by the caterpillars so far. In particular, the points on those paths are encoded as providers in the graph maintained by the algorithm. The classification remains correct throughout the whole stream.

SWLP (Figure 3d) stores the τ last points as well, but it does not encode the history of the stream. The classification fails as soon as the caterpillars move away from the labeled points on the x-axis.

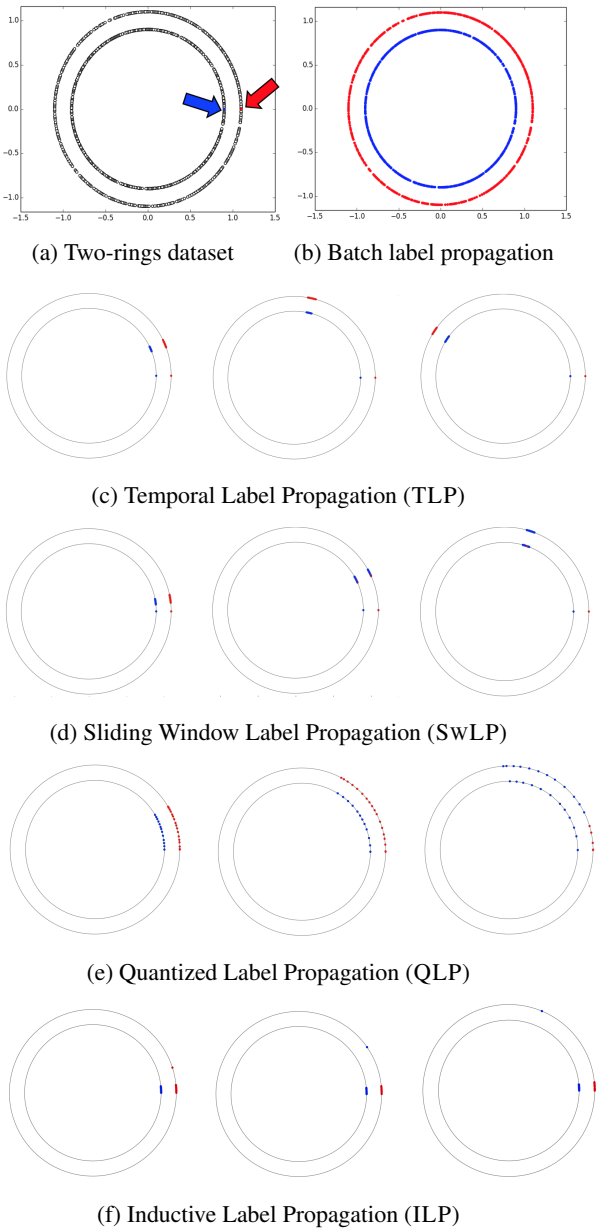
QLP (Figure 3e) takes a different approach: instead of storing the τ recent points, it strategically chooses τ centroids that quantize the stream seen so far. It fails when the stream has become too long to quantize with only τ centroids.

ILP (Figure 3f) trains on the first τ unlabeled points and uses them to label subsequent points, but it does not update the learned model over time. It fails when the model no longer captures the evolution of the stream over time.

LOLP is not depicted. It fails similar to SWLP and ILP, and for the same reason.

6.2. Real Data

Datasets. We use 4 datasets arising from different domains: (a) *Incart-ECG* (Goldberger et al., 2000): Dataset of ECG timeseries from PhysioNet bank, annotated with heartbeat arrhythmias. We use one ECG lead. The task is to classify *atrial* (positive) vs. *ventricular* premature contractions (negative). Both are common arrhythmias that co-occur in patients. Only the timeseries associated with the two arrhythmias is provided to the algorithms for classification; normal heartbeats are ignored. (b) *Daphnet-Gait* (Bachlin et al., 2010): Annotated readings of 9 accelerometer sensors of Parkinson's disease patients that experience freezing of gait during walking tasks. The goal is to detect gait freeze (positive) vs. regular walking (negative). (c) *Caltech*$_{10}$-*101* (Fei-Fei et al., 2006): Caltech-101 dataset consists of images annotated by 101 object classes with about 800 images per class. We restrict ourselves to 10 classes. The images were resized to 100×200 (RGB) pixels. This data



(a) Two-rings dataset (b) Batch label propagation

(c) Temporal Label Propagation (TLP)

(d) Sliding Window Label Propagation (SWLP)

(e) Quantized Label Propagation (QLP)

(f) Inductive Label Propagation (ILP)

Figure 3. Visualization of the compared algorithms. In (c)-(f), the black circles are only depicted as a visual aid.

Dataset	#Labeled examples (per class)	τ	Accuracy (in%)					Avg. Time per point (in msec)				
			TLP (Ours)	SwLP	QLP	ILP	LoLP	TLP (Ours)	SwLP	QLP	ILP	LoLP
Incart-ECG	2	5	94.9	69.6	58.9	52.6	70.0	0.052	0.051	0.071	0.054	0.050
Daphnet-Gait	6	100	75.9	56.3	70.9	44.6	72.0	15.8	15.0	104.3	15.4	14.7
Caltech$_{10}$-101	10	10	80.9	78.1	78.9	80.6	78.9	51.4	51.3	110.6	171.6	49.3
CamVid-Car	3	10	95.9	80.6	54.5	86.9	72.4	426.4	424.2	877.6	623.6	420.8

Table 1. All of the compared algorithms are given the same labeled datapoints and unlabeled data stream. The results for Incart-ECG and Daphnet-Gait are averaged over multiple patients. The results for Caltech$_{10}$-101 are averaged over 5 runs each time with a different ordering of the test stream. The standard deviation for the results across these runs is generally small for all of the approaches.

is non-temporal; we simulate a stream by generating random permutations of the images. (d) *CamVid-Car* (Brostow et al., 2009) (Cambridge-driving Labeled Video Database): CamVid dataset consists of video sequences taken from a moving vehicle in an urban environment with ground truth, provided at a rate of 1Hz, of 32 semantic classes for each pixel of the frames. We restrict ourselves to a binary classification problem of detecting whether there is a car in the frame (positive) or not (negative). We call this the CamVid-Car dataset. For both the Caltech$_{10}$-101 and CamVid datasets, we use the raw RGB features. The properties of the datasets are summarized in Table 2.

Dataset	#Datapoints	#Features	#Classes
Incart-ECG	1850408	1	2
Daphnet-Gait	476813	9	2
Caltech$_{10}$-101	2461	$100 \times 200 \times 3$	10
CamVid-Car	577	$720 \times 960 \times 3$	2

Table 2. Properties of the experimental datasets.

Shingling. A useful technique when dealing with timeseries data is to group consecutive sequences (N-grams) of points into shingles. This lifts the data into a higher dimension N and allows for a richer representation of inputs. For Incart-ECG and Daphnet-Gait we use $N = 50$ and generate the stream by taking the normalized difference between every two consecutive shingles. The normalized shingle difference is useful for capturing local shapes in the signal (such as heartbeat arrhythmias) rather than absolute values.

Experimental Setting. We use the standard RBF similarity, $\mathrm{Sim}(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/\sigma^2)$. We set $\sigma = 0.1$ for Incart-ECG, Daphnet-Gait, and CamVid and $\sigma = 10$ for Caltech$_{10}$-101. For the Incart-ECG, Daphnet-Gait, and CamVid datasets we use the natural ordering of the stream, whereas with Caltech$_{10}$-101 dataset we generate a stream by randomly permuting the images. The labeled examples are given in the beginning of each stream, and we start the labeling process once the mentioned amount of labels from each class arrives. All experiments were performed on a 3.1 GHz Intel Core i7 machine with 16GB RAM.

Results. Table 1 presents our main experimental results. We make the following observations:

(1) Short-circuiting matters: The comparison of TLP to SwLP directly evaluates the effect of summarizing the stream by the star-mesh transform, as they are otherwise identical. As noticed in Table 1, it yields a substantial improvement in the accuracy on the temporally-ordered datasets Incart-ECG, Daphnet-Gait, and CamVid-Car, with almost no effect on the running time. This corroborates the presumption that TLP is well suited for streams that adhere to a temporal vicinity structure as per Section 5.1. However, when there is no natural temporal ordering (such as with Caltech$_{10}$-101 data), we did not observe an advantage over the other methods.

(2) Small amount of labeled data suffice: Notice that we use a very small amount of labeled data in each experiment. For example, on the Incart-ECG dataset, TLP can get to a 95% classification accuracy given only two labeled examples of each type of arrhythmia.

(3) Computational speedup: Notice that on the timeseries datasets, even with shingling, which increases the dimensionality of the data by a factor of shingle size, TLP takes few milliseconds per point. We remark that QLP is slower than the other methods because of the iterative loop in the k-center quantization step.

In Appendix E, we present additional experiments that show how τ and labeled data size effects the performance of TLP. We also present some visualizations of our approach on the tested datasets.

7. Conclusion and Future Work

We presented a principled approach for adapting the label propagation algorithm of (Zhu et al., 2003a) to streaming data. There are many extensions and variants of this fundamental algorithm that address issues like regularization, interpretability, noise in labels, and more. A possible direction of further research is using our methods to adapt these extensions to the streaming setting as well.

Recently, there has been a surge of theoretical work on fast computation of approximate short-circuit graphs and on maintaining them dynamically (Durfee et al., 2017; 2018; Goranci et al., 2017; 2018). It is not directly applicable to label propagation since they approximate the energy but not the values at individual nodes (see Remark 4.3). In light of our work, we are interested if these results could have implications for SSL in dynamic settings.

Acknowledgements

We thank Roger Barga, Kapil Chhabra, Charles Elkan, Praveen Gattu, Lauren Moos, Gourav Roy, Joshua Tokle and the anonymous reviewers for useful comments and suggestions.

References

Anderson, Jr, William N. Shorted operators. *SIAM Journal on Applied Mathematics*, 20(3):520–525, 1971.

Bachlin, Marc, Plotnik, Meir, Roggen, Daniel, Maidan, Inbal, Hausdorff, Jeffrey M, Giladi, Nir, and Troster, Gerhard. Wearable assistant for parkinson?s disease patients with the freezing of gait symptom. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):436–446, 2010.

Balcan, Maria-Florina, Blum, Avrim, Choi, Patrick Pakyan, Lafferty, John D, Pantano, Brian, Rwebangira, Mugizi Robert, and Zhu, Xiaojin. Person identification in webcam images: An application of semi-supervised learning. *CMU Repository*, 2005.

Belkin, Mikhail, Matveeva, Irina, and Niyogi, Partha. Regularization and semi-supervised learning on large graphs. In *International Conference on Computational Learning Theory (COLT)*, pp. 624–638. Springer, 2004.

Belkin, Misha, Niyogi, Partha, and Sindhwani, Vikas. On manifold regularization. In *Artificial Intelligence and Statistics (AISTATS)*, pp. 1, 2005.

Blum, Avrim and Chawla, Shuchi. Learning from labeled and unlabeled data using graph mincuts. In *International Conference on Machine Learning (ICML)*, pp. 19–26, 2001.

Brostow, Gabriel J, Fauqueur, Julien, and Cipolla, Roberto. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30 (2):88–97, 2009.

Campbell, George A. Direct capacity measurement. *Bell Labs Technical Journal*, 1(1):18–38, 1922.

Chapelle, Olivier, Scholkopf, Bernhard, and Zien, Alexander. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

Charikar, Moses, Chekuri, Chandra, Feder, Tomás, and Motwani, Rajeev. Incremental clustering and dynamic information retrieval. In *ACM Symposium on Theory of Computing (STOC)*, pp. 626–635, 1997.

Delalleau, Olivier, Bengio, Yoshua, and Le Roux, Nicolas. Efficient non-parametric function induction in semi-supervised learning. In *Artificial Intelligence and Statistics (AISTATS)*, volume 27, pp. 100, 2005.

Dorfler, Florian and Bullo, Francesco. Kron reduction of graphs with applications to electrical networks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(1):150–163, 2013.

Durfee, David, Kyng, Rasmus, Peebles, John, Rao, Anup B., and Sachdeva, Sushant. Sampling random spanning trees faster than matrix multiplication. In *ACM Symposium on Theory of Computing (STOC)*, pp. 730–742, 2017.

Durfee, David, Gao, Yu, Goranci, Gramoz, and Peng, Richard. Fully dynamic effective resistances. *arXiv preprint arXiv:1804.04038*, 2018.

Dyer, Karl B, Capo, Robert, and Polikar, Robi. Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. *IEEE transactions on neural networks and learning systems*, 25(1):12–26, 2014.

Fei-Fei, Li, Fergus, Rob, and Perona, Pietro. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

Ghazvininejad, Marjan, Mahdieh, Mostafa, Rabiee, Hamid R, Roshan, Parisa Khanipour, and Rohban, Mohammad Hossein. Isograph: Neighbourhood graph construction based on geodesic distance for semi-supervised learning. In *IEEE International Conference on Data Mining (ICDM)*, pp. 191–200. IEEE, 2011.

Goldberg, Andrew B, Li, Ming, and Zhu, Xiaojin. Online manifold regularization: A new learning setting and empirical study. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 393–407. Springer, 2008.

Goldberg, Andrew B, Zhu, Xiaojin, Furger, Alex, and Xu, Jun-Ming. Oasis: Online active semi-supervised learning. In *AAAI Conference on Artificial Intelligence*, 2011.

Goldberger, Ary L, Amaral, Luis AN, Glass, Leon, Hausdorff, Jeffrey M, Ivanov, Plamen Ch, Mark, Roger G, Mietus, Joseph E, Moody, George B, Peng, Chung-Kang, and Stanley, H Eugene. Physiobank, physiotoolkit, and physionet. *Circulation*, 101(23), 2000.

Goranci, Gramoz, Henzinger, Monika, and Peng, Pan. The power of vertex sparsifiers in dynamic graph algorithms. In *European Symposium on Algorithms (ESA)*, pp. 45:1–45:14, 2017.

Goranci, Gramoz, Henzinger, Monika, and Peng, Pan. Fully dynamic effective resistances. *arXiv preprint arXiv:1802.09111*, 2018.

Horn, Roger A and Johnson, Charles R. *Matrix analysis*. Cambridge university press, 1990.

Huang, Lei, Liu, Xianglong, Ma, Binqiang, and Lang, Bo. Online semi-supervised annotation via proxy-based local consistency propagation. *Neurocomputing*, 149:1573–1586, 2015.

Jebara, Tony, Wang, Jun, and Chang, Shih-Fu. Graph construction and b-matching for semi-supervised learning. In *International Conference on Machine Learning (ICML)*, pp. 441–448. ACM, 2009.

Joachims, Thorsten. Transductive learning via spectral graph partitioning. In *International Conference on Machine Learning (ICML)*, pp. 290–297, 2003.

Kirchhoff, Gustav. Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Annalen der Physik*, 148(12):497–508, 1847.

Krempl, Georg, Žliobaite, Indre, Brzeziński, Dariusz, Hüllermeier, Eyke, Last, Mark, Lemaire, Vincent, Noack, Tino, Shaker, Ammar, Sievi, Sonja, Spiliopoulou, Myra, et al. Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter*, 16(1):1–10, 2014.

Le Gall, François. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation*, pp. 296–303. ACM, 2014.

Levin, Anat, Lischinski, Dani, and Weiss, Yair. Colorization using optimization. *ACM Transactions on Graphics (ToG)*, 23(3):689–694, 2004.

Ravi, Sujith and Diao, Qiming. Large scale distributed semi-supervised learning using streaming approximation. In *Artificial Intelligence and Statistics (AISTATS)*, pp. 519–528, 2016.

Rohde, Charles A. Generalized inverses of partitioned matrices. *Journal of the Society for Industrial and Applied Mathematics*, 13(4):1033–1035, 1965.

Rosen, A. A new network theorem. *Journal of the institution of electrical engineers*, 62(335):916–918, 1924.

Sindhwani, Vikas, Niyogi, Partha, and Belkin, Mikhail. Beyond the point cloud: from transductive to semi-supervised learning. In *International Conference on Machine Learning (ICML)*, pp. 824–831. ACM, 2005.

Snell, PGDJL and Doyle, Peter. Random walks and electric networks. *Free Software Foundation*, 2000.

Szummer, Martin and Jaakkola, Tommi. Partially labeled classification with markov random walks. In *Advances in neural information processing systems (NIPS)*, pp. 945–952, 2002.

Valko, Michal, Kveton, Branislav, Ling, Huang, and Daniel, Ting. Online semi-supervised learning on quantized graphs. In *Uncertainty in Artificial Intelligence (UAI)*, 2010.

Wang, Fei and Zhang, Changshui. Label propagation through linear neighborhoods. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):55–67, 2008.

Wang, Jun, Jebara, Tony, and Chang, Shih-Fu. Graph transduction via alternating minimization. In *International Conference on Machine Learning (ICML)*, pp. 1144–1151. ACM, 2008.

Zemel, Richard S and Carreira-Perpiñán, Miguel Á. Proximity graphs for clustering and manifold learning. In *Advances in neural information processing systems (NIPS)*, pp. 225–232, 2005.

Zhang, F. *The Schur Complement and Its Applications*. Numerical Methods and Algorithms. Springer, 2005.

Zhou, Denny, Bousquet, Olivier, Lal, Thomas N, Weston, Jason, and Schölkopf, Bernhard. Learning with local and global consistency. In *Advances in neural information processing systems (NIPS)*, pp. 321–328, 2004.

Zhou, Denny, Hofmann, Thomas, and Schölkopf, Bernhard. Semi-supervised learning on directed graphs. In *Advances in neural information processing systems (NIPS)*, pp. 1633–1640, 2005.

Zhu, Xiaojin. Semi-supervised learning literature survey. *CMU Repository*, 2005.

Zhu, Xiaojin and Lafferty, John. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *International Conference on Machine Learning (ICML)*, pp. 1052–1059. ACM, 2005.

Zhu, Xiaojin, Ghahramani, Zoubin, and Lafferty, John D. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning (ICML)*, pp. 912–919, 2003a.

Zhu, Xiaojin, Lafferty, John D, and Ghahramani, Zoubin. Semi-supervised learning: From gaussian fields to gaussian processes, 2003b.

Zhu, Xiaojin, Goldberg, Andrew B, and Khot, Tushar. Some new directions in graph-based semi-supervised learning. In *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1504–1507. IEEE, 2009.