Learning Evolving Networks via Local Probing

Rajesh Jayaram
Advisor: Eli Upfal

May 2, 2017

Evolving Networks: Motivation and Application

Suppose we have a network $G = (V, E)$, where each node $v \in G$ has an *opinion*, or label.

- As time goes on, the opinions of some vertices's change (according to some distribution).
- Want to keep track of all opinions, but we can only observe information from a small (i.e. constant or polylogarithmic) portion of the graph at any time (our resources are bounded).
- What guarantees can we make between # of errors and # of probes?

Applications: Computer Vision

G is the $n \times n$ grid, where each vertex is a pixel in an $n \times n$-pixel image. Our model can encapsulate:

1. Foreground-background segmentation.
2. Object detection.
3. Movement detection.

In changing (and possibly noisy) video streams.

Applications: Social Networks

V is a set of people, edges correspond to "friendships".

- Everyone has a political party $\in \{0, 1\}$
- At time $t = 0$ we know most people's affiliations.
- Time progresses, some people change affiliations.
- (e.g. a traditional party surprisingly supports an untraditional candidate)

Probably hard to compute a political party looking only at one profile. Perhaps easier to compare two friends who agree on most things or not.

Evolving Networks: Our (noiseless) Model

Notation:

- Let G be a graph with vertices $v_1, \ldots, v_n$.
- Let $v_{i,t} \in \{0, 1\}$ be the label of vertex v_i at time t, and let $G_t = \{v_{i,t} \mid i = 1, 2, \ldots, n\}$.
- Let $h_t(v_i) \in \{0, 1\}$ be our hypothesis of v_i at time t.

Our error at time t is defined as:

$$\#\text{Error at time } t = \sum_{i=1}^{n} \left| v_{i,t} - h_t(v_i) \right|$$

The above is the objective function which we want to minimize w.h.p. at all time steps.

Evolving Networks: Our (noiseless) Model

We assume our initial hypothesis $h_0(G)$ is wrong on an ϵ_0 fraction of the vertices.

1. At the beginning of time t, each vertex $v \in G$ flips its value with probability $\frac{1}{2n}$.
2. Afterwards, we query k edges, and observe the `XOR` of the adjacent vertices of each probed edge.
3. We then update our hypothesis $h_t(G)$ based on $h_{t-1}(G)$ and the edge observations.

Why This Model?

Bounded number of probes is clear: we have limited computational resources. But why can we only probe `XOR` of an edge?

- In many practical examples, it is often difficult or unclear how to classify an individual vertex.
- However, it may be much easier to decide whether two adjacent vertices have the same classification.

For instance, in the video stream example, classifying foreground vs background via RGB of one pixel is not great! A better idea is to take the ℓ_1 distance between adjacent pixels, and ask if it is above a threshold.

Generalizations

Several generalizations to consider, namely where the vertices of G_t flip with according to some distribution D.

1. Vertices more likely to flip if they have many disagreeing neighbors
2. More complicated dependencies (rumor spreading, ect).

Notably, given that $k \in \Omega(\log(n))$, our second algorithm is independent of D as long as

$$Pr(\#\text{ flips at time } t > k) < \frac{1}{\text{Poly}(n)}$$

For i.i.d. with probability of flipping $\frac{1}{n}$, $k \in O(\log(n))$ is sufficient

A straightforward lower bound

Suppose our vertices flip i.i.d. with probability β.

Theorem

In the stochastic network model where vertices are flipped by nature with probability β, any algorithm which uses at most k probes on every time step never accumulates less than $\beta\frac{n^2}{4k} - \beta^2\frac{n}{2k}$ errors at any time step in expectation.

- If $\beta \in \Theta(\frac{1}{n})$, then this represents a *linear* probe-error tradeoff.

A straightforward lower bound

Theorem

In the stochastic network model where vertices are flipped by nature with probability β, any algorithm which uses at most k probes on every time step never accumulates less than $\beta\frac{n^2}{4k} - \beta^2\frac{n}{2k}$ errors at any time step in expectation.

Proof.

The proof is a simple counting argument in the expectation. At best, there are k vertices which have not been seen since the last step, $2k$ that have not been seen in the since the last 2 steps, and so on (in the best case all n vertices have been seen in the last $\lceil \frac{n}{k} \rceil$ steps). Summing over the probabilities that these vertices have flipped since last seen gives the desired expectation. □

k-Trailing Walks Algorithm: Intuition

Suppose we know $h_t(v_i) = v_{i,t}$ with probability 1. Then if (v_i, v_j) is an edge, I can probe it and determine $v_{j,t}$ correctly with probability 1.

- if $v_{i,t} = 1, v_{j,t} = 0$, then probing (v_i, v_j) I observe that $v_{i,t} \oplus v_{j,t} = 0$.
- Since I know $h_t(v_i) = v_{i,t} = 1$ for certain, it must be the case that $v_{h,t} = 0$.

I can then probe a neighbor of v_j, say v_r and be correct on it with certainty. Moving on in this fashion I can fix everything.

k-Trailing Walks Algorithm: Intuition

But note that this only works if:

- I know for sure I am correct about $h_t(v_i)$.
- What happens if v_j flips between the time I probe (v_i, v_j) and (v_j, v_r)?
  - $v_{j,t} = 0$ but $v_{j,t+1} = 1$.
  - I fix $v_{j,t} = 0$ correctly but then incorrectly assign $v_{k,t+1}$.
  - From here on, I incorrectly assign everything since I assumed I was right on $v_{k,t+1}$.

This is no good!!

k-Trailing Walks Algorithm

Keep track of a trailing set of vertices T, in a path, which we verify to be correct on w.h.p. at the beginning of every time step. So on every time step:

- Probe all the edges between the vertices in this "trailing path".
- This restricts the possible values for the vertices in T to two possible vectors in $\{0,1\}^{|T|}$.
- Assign our hypothesis to be the vector of values, of the two, which is closer to our old hypothesis.
  - The probability we are wrong is the probability that more than half the vertices flip in one time step.
  - This occurs with probability $\leq (\frac{1}{n})^{|T|/2}$
- Continue probing probing vertices in a path starting from the last vertex in T.

k-Trailing Walks Algorithm (initialization)

Suppose G is has a Hamiltonian cycle $v_1, v_2, \ldots, v_n, v_1$ (we will generalize soon!). Then given inputs r, s and k', where $k = k'(r+s)$:

1. Sample $j \xleftarrow{U} \{1, 2, \ldots, n\}$
2. Probe all edges between v_j and $v_{(j+r) \pmod n}$, pick assignment closest to initial hyp.
3. Iteratively probe the next vertex in cycle, starting from $j + r + 1 \pmod n$ and ending at $j + r + s \pmod n$, and assign value agreeing with edge & hypothesis for prior vertex.
4. Mark the ending position $j + r + s \pmod n$
5. Repeat steps 1-4 k' times.

k-Trailing Walks Algorithm (time t)

Then at time t

1. For each marked position $j \in \{1, \ldots, n\}$:
2. "Unmark" j. Then probe all edges between v_j and $v_{(j+r) \pmod n}$, pick assignment closest to hypothesis on last step.
3. Iteratively probe the next vertex in cycle, starting from $j + r + 1 \pmod n$ and ending at $j + r + s \pmod n$, and assign value agreeing with edge & hypothesis for prior vertex.
4. Mark the ending position $j + r + s \pmod n$

k-Trailing Walks Algorithm: Error Bound

Theorem

Fix any $k = k'(r+s)$ and $\delta \geq 1$. Then with probability at least $(1 - \frac{1}{k'})$, the (k', r, s)-Trailing Walks Algorithm has no more than $4(\log(k'))^2 n\epsilon_0 + (1+\delta)\frac{n}{2sk'}$ errors at any time at all before time $T \in O(n^{(r+1)/2}\frac{1}{k'})$.

- Note if $\epsilon_0 = 0$ then the above theorem holds with probability $O(1 - \frac{1}{\text{Poly}(n)})$.

k-Trailing Walks Algorithm: any graph

Of course, nobody designs practical algorithms for Hamiltonian graphs. So:

Theorem

Let G be any graph, and let $V(G) = C_1 \coprod \cdots \coprod C_d \coprod X$ be any partition of the vertices such that each C_i is a cycle. Then the errors obtained by running the k-Trailing Walks algorithm on each subgraph C_i admit the prior error bounds when applied to each subgraph, plus at most the additional cost $|X|$ at any time step.

- X are just the vertices you can't put into cycles (should be $\emptyset$!).

Our α-noisy Model

Again, start with $n\epsilon_0$ errors. Let $D(G_{t-1})$ be any distribution over the vertices of G_t. Assume for $q \geq 1$:

$$Pr\big(\#\text{ vertex flipped } > O(\log(n))\big) \leq \frac{1}{n^q}$$

Then at time t

1. $v_{i,t} \xleftarrow[D(G_{t-1})]{} \{0,1\}$ for $i = 1, 2, \ldots, n$
2. We probe k edges.
3. For each edge (v_i, v_j) probed, we observe $v_{i,t} \oplus v_{j,t}$ with probability $1 - \alpha$, and observe $\neg v_{i,t} \oplus v_{j,t}$ with probability α.
4. Must then update our hypothesis vector $h_t(G)$.

Expander Walks Algorithm

Given $k = k'r$, where r is the length of the path and r is the number of paths:

1. Sample a random path of length k'.
2. Probe all edges in the path, assign hypthesis to be the vertex values that minimize ℓ_1 distance from prior hyp.
3. Repeat 1-2 a total of r times.

Pretty straightforward... but this walk needs to be mixing. We need good expansion, or large k'!

Expander Walks Algorithm: why expansion matters

Now why cant we just probe any path or set of vertices with their neighborhoods? Where is the difficulty?

- The prob. of not fixing a bad vertex is the prob. that $\geq 1/2$ of the probed vertices are incorrect conditioned on the fact that we have a bad vertex.
  - If the errors are distributed uniformly, there is no problem!
- But they are not! We make more errors if we start around errors, altering the distribution
- Errors cluster around other errors,
  - We cannot fix large connected components of errors!
- The result is algorithmic failure.

We need to guarantee that the number of errors on our path is close to the expected number!

Expander Walks Algorithm: why expansion matters

There are two ways for this to happen:

1. Sample a long enough random walk P so that $|\Pr(v_i \in P) - \frac{|P|}{n}| \leq \epsilon$ for some small ϵ.
2. Pick a graph where our current number of allowed probes does this already (pick a good expander).

In fact, the number of required probes will be lower bounded by a function of the spectral gap of G.

Theorem (Expander Walk Sampling)

Let G be a graph with normalized second largest eigenvalue λ_2. Let $f : V(G) \to \{0,1\}$ be any function, and let $\mu = \frac{1}{n}\sum_{v_i \in V(G)} f(v_i)$ be its mean. If $Y_0, Y_1, \ldots, Y_{k'}$ is a k'-step random walk starting at a random vertex Y_0, then we have for all $\gamma > 0$:

$$\Pr\left[\frac{1}{k'}\sum_{i=0}^{k'} f(Y_i) - \mu > \gamma\right] \leq e^{-\frac{\gamma^2(1-\lambda_2)k'}{10}}$$

and

$$\Pr\left[\frac{1}{k'}\sum_{i=0}^{k'} f(Y_i) - \mu < -\gamma\right] \leq e^{-\frac{\gamma^2(1-\lambda_2)k'}{10}}$$

Expander Walks Algorithm: Two Paths to Error Bounds

Our analysis results in two bounds using two techniques, and applying the previous walk sampling theorem in both.

1. Model the # of errors as a bias random walk towards the expectation. Use results of Bias Gambler's Ruin.
2. Model the # of errors exceeding the mean as a supermartingale $X_0, X_1, \ldots,$ with $\mathbb{E}[X_0]$ the expected number of errors.

Approach 1 gives a weaker bound for (any degree) polynomially many time steps. Approach 2 gives a better bound for $O(n^2)$ time.

Remark

Note that in all of these bounds we require $\alpha^{-1} \in \Omega(k)$. For α constant there are strong arguments for impossibility.

Expander Walks Algorithm: Bias Random Walks

Theorem

Given $k \in O(q\frac{\log(n)}{(1-\lambda_2)})$ *probes per time step, with high probability the maximum number of errors encountered before time* n^q *is at most* $O\left(n\left(\sqrt{\frac{q\log(n)}{(1-\lambda_2)k}} + \alpha k\right) + qk\log(n)\right)$.

Proof.

Steps of proof:

1. Bound number of vertices that flip at time t w.h.p.
2. Prove high prob upper/lower bounds on # bad vertices on path using expander walk sampling.
3. Break interval $\{1, 2, \ldots, n\}$ int subintervals of size $2k$, and model the errors as a random walk on these subintervals.
   - Cannot skip subintervals between steps (some detail involved)!
4. Use results of bias gambler's ruin.

Expander Walks Algorithm: Easy Generalization

We can generalize to any distribution over how vertices flip. Suppose that for some $\zeta < k$ the distribution satisfies:

$$\Pr[\max \#\text{ flips at time at any time before } n^q \geq \zeta] \leq \frac{1}{\text{Poly}(n)}$$

Theorem

Given $k \in O(q\frac{\log(n)}{(1-\lambda_2)})$ probes per time step, with high probability the maximum number of errors encountered before time n^q is at most $O\left(n\left(\sqrt{\frac{q\log(n)}{(1-\lambda_2)k}} + \alpha k + \frac{\zeta}{k}\right) + qk\log(n)\right)$.

- Note that in the last slide $\zeta \leq 6q\log(n)$, so it is strictly dominated by the $\sqrt{\frac{q\log(n)}{(1-\lambda_2)k}}$ term in the Big-O.

Expander Walks Algorithm: Azuma's Inequality for Supermartingales

Theorem

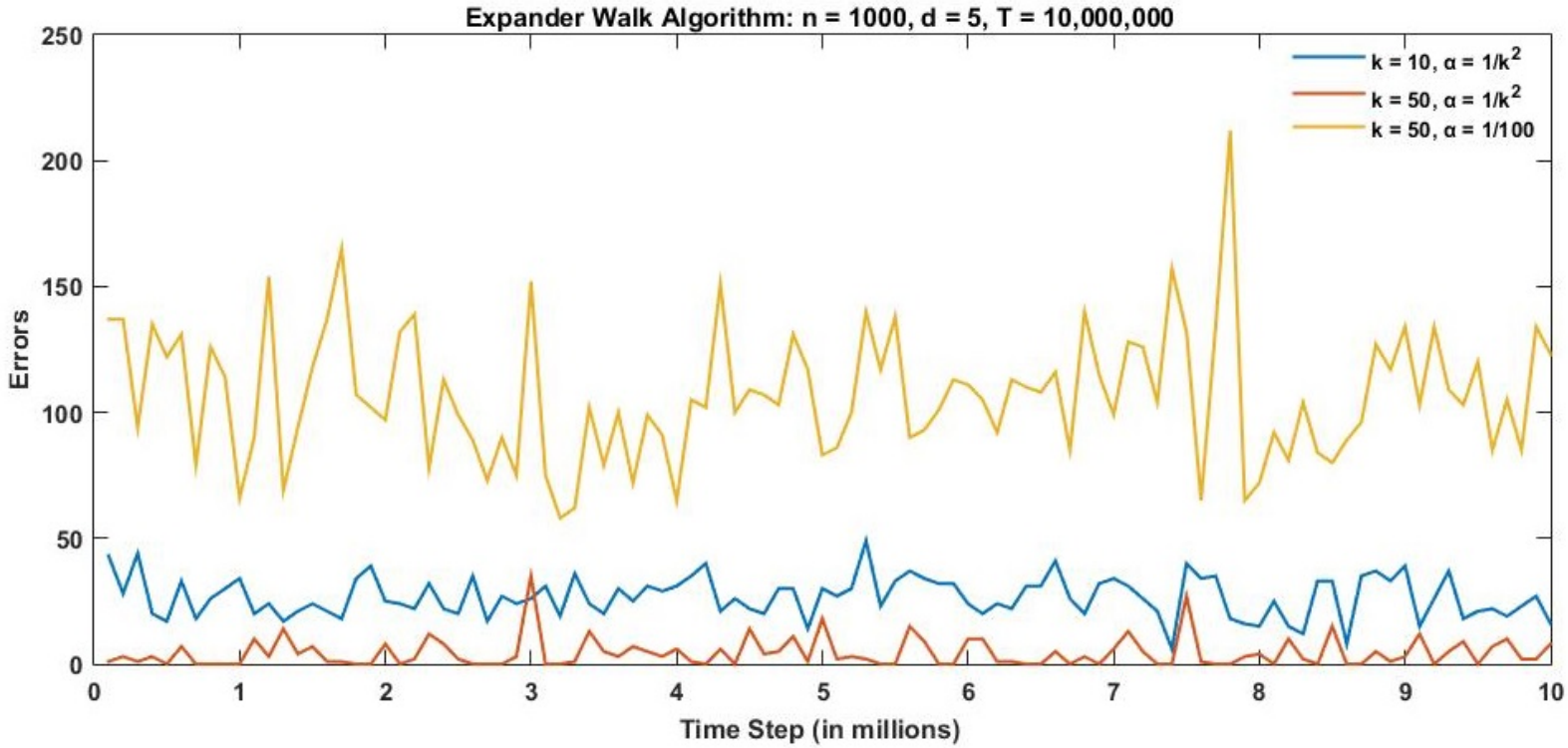
Fix integers $r, c \geq 1$, *and* $k = k'r \geq 2560rc\frac{\log(n)}{(1-\lambda_2)}$. *Assume* $\alpha^{-1} > 2(k')^2 r$. *Then with probability* $\geq (1 - \frac{1}{n^{c-2}})$, *the maximum number of errors accumulated at any time before* $\frac{n^2}{k^4}$ *is at most* $\frac{n}{r(1-\alpha k')}\left(1 + \sqrt{8c\log(n)}\right)$.

Proof.

Let ϵ_t be fraction of errors at time t.

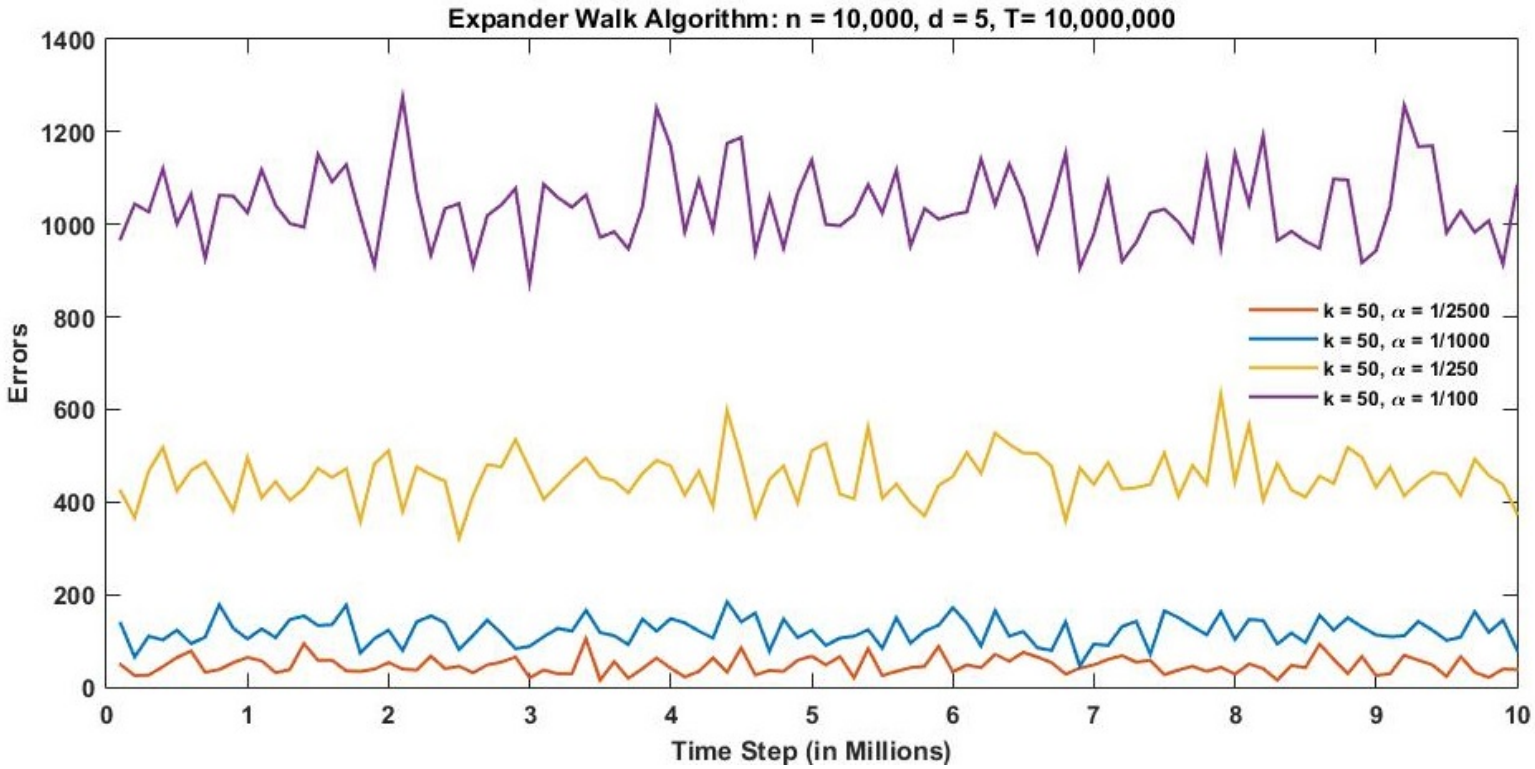
1. Compute $\mathbb{E}[\epsilon_t]$ (fixed point of the random process).
2. Prove that w.h.p. $\epsilon_t < \frac{1}{4}$ for polynomially many time steps (constant bounded from 1/2).
3. Prove $X_t = \min\{\max\{n\mathbb{E}[\epsilon_t], n\epsilon_t\}, \frac{n}{4}\}$ is a supermartingale upper bounding $n\epsilon_t$ w.h.p.
4. Apply Azuma's inequality for Supermartingales

Experiments



Parameters	Max Errors	Min	Mean	Median	σ^2
$k = 10, \alpha = \frac{1}{k^2}$	72	4	26.62	26	60.58
$k = 50, \alpha = \frac{1}{k^2}$	4	0	4.97	3	36.21
$k = 50, \alpha = \frac{1}{100}$	212	22	104.65	103	594.32

Experiments



Parameters	Max Errors	Min	Mean	Median	σ^2
$k = 50, \alpha = \frac{1}{2500}$	142	4	50.19	48	365.79
$k = 50, \alpha = \frac{1}{1000}$	268	38	123.93	122	897.65
$k = 50, \alpha = \frac{1}{250}$	702	298	466.58	465	3178.04