Semi-Supervised Learning with Competitive Infection Models

Nir Rosenfeld
Harvard University

Amir Globerson
Tel Aviv University

Abstract

The goal of semi-supervised learning methods is to effectively combine labeled and unlabeled data to arrive at a better model. Many methods rely on graph-based approaches, where labels are propagated through a graph over the input examples. In most current methods, the propagation mechanism underlying the learning objective is based on random walks. While theoretically elegant, random walks suffer from several drawbacks which can hurt predictive performance.

In this work, we explore *dynamic infection processes* as an alternative propagation mechanism. In these, unlabeled nodes can be "infected" with the label of their already infected neighbors. We provide an efficient, scalable, and parallelizable algorithm for estimating the expected infection outcomes. We also describe an optimization view of the method, relating it to Laplacian approaches. Finally, experiments demonstrate that the method is highly competitive across multiple benchmarks and for various learning settings.

1 Introduction

The supervised learning framework underlies much of the empirical success of machine learning systems. Nonetheless, results in unsupervised learning have demonstrated that there is much to be gained from unlabeled data as well. This has prompted considerable interest in the *semi-supervised learning* [11] setting, where the data includes both labeled and unlabeled examples. Methods for semi-supervised learning (SSL) are especially useful for applications in which unlabeled examples are ample, but labeled examples are scarce or expensive.

One of the most wide-spread approaches to SSL, and our focus in this paper, is the class of *graph-based methods*. In these, part of the problem input is a graph that specifies which input points should be considered *close*. Graph-based methods assume that proximity in the graph implies similarity in labels. There are many variations on this idea [5, 7, 36, 40], each using smoothness and graph distance differently. However, they all share the intuition that the classification function should be smooth with respect to the graph.

One way for encouraging smoothness in predictions is by optimizing an objective based on the graph Laplacian. This approach is prevalent in both classic SSL methods such as Label Propagation (LabelProp) [46] and its variants [45, 4, 41], as well as in recent deep-embedding methods [44, 35]. Optimizing the quadratic learning objective in these methods can often be done by iterative local averaging of labels. This can be thought of as propagating labels under a certain averaging dynamic process, whose steady state corresponds to the optimum. Equivalently, the optimum can also be expressed as the probability that a random walk terminates at a certain state. Due to their elegance, computational properties, and empirical performance, random walks and local averaging have become the standard go-to mechanisms for propagating information in many applications. Nonetheless, they also have several shortcomings, which we address here.

First, many of the guarantees of such methods hold only for undirected graphs. For directed graphs, the Laplacian is not necessarily PSD, meaning that the objective is no longer convex, and that the quadratic smoothness interpretation breaks down. Optimization in directed graph Laplacians is much harder and far less understood [42], and sampling is computationally prohibitive, slow to converge, and unstable [28, 29].

Second, such methods were originally designed for graphs that approximate the density of the data in feature space. As such, they can fail when applied to real graphs, especially large networks with a community structure. This is because random walks are prone to get stuck in local neighborhoods [9], because visiting all nodes can require an expected $O(n^3)$ steps [2],

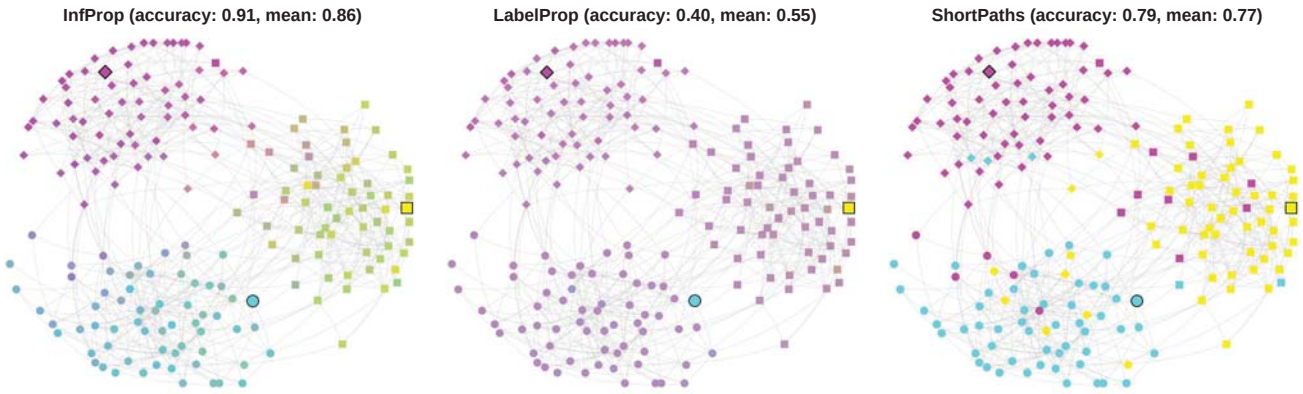

Figure 1: For graphs with weakly inter-connected components, infection dynamics (our method, left) propagate labels better than random walks (middle) or shortest paths (right). Labeled nodes are outlined, shapes denote true labels, and probabilistic predictions are encoded by CMY color values. See supp. material for more details.

and since the limit distribution can be uninformative for large graphs [43] or when labels are rare [32].

Third, extending such methods beyond the vanilla multiclass setting has proven to be quite challenging. For instance, outputting confidence in predictions is possible, but leads to extremely low values [41]. Label priors can be utilized, but only in determining the classification rule, rather than being incorporated into the model [46]. Most methods for an active semi-supervised setting are either heuristic [22] or based on pessimistic worst-case objectives [23]. Finally, supporting structured labels is far from straightforward and can be computationally demanding [3].

Due to the above, in this work we advocate for considering alternative mechanisms for propagating labels over a graph, and propose an approach which addresses most of the above issues. Our method, called *Infection Propagation* (InfProp), views the process of labeling the graph as a *dynamic infection process*. Initially, only the labeled nodes are "infected" with their known labels. As time unfolds, unlabeled nodes can, with some probability, be infected by their neighbors. When this happens, they inherit the label of their infector. The labeled nodes therefore compete over infecting the unlabeled nodes with their labels. Since the infection process is stochastic we can calculate the probability that a given node was infected by a given label, and label the node according to this probability.

Exact calculation of the above infection probabilities is #P hard. However, we provide a fully polynomial-time randomized approximation scheme (FPRAS), which exploits an equivalence between the infection process and shortest paths in random graphs. The resulting algorithm is easy to parallelize, making the method highly scalable. It also extends to various learning settings, such as multilabel prediction and active SSL.

InfProp is motivated by the idea that different graph types may require different dynamics for efficient propagation of information. It is inspired by propagation dynamics found in the natural and social worlds, and draws on the successful application of infection models in different contexts [25, 19, 13]. InfProp is especially efficient for graphs with highly intra-connected but lightly inter-connected components, a characteristic of many real-world networks. Fig. 1 illustrates this for a small synthetic random network with three clusters (see supplementary material for details). As can be seen, InfProp propagates information correctly, even when the seed set is very small. In comparison, LabelProp provides uninformative and almost uniform predictions which are prone to error, and shortest paths over the weighted graph err due to cross-cluster links.

Since InfProp is a graph based method, it is intriguing to understand its relation to label propagation and random walk based methods. Our analysis in Sec. 5 provides some surprising connections. Specifically, we show that InfProp can be viewed as optimizing a quadratic objective, in which weights are related in an intricate manner to the undelying diffusion process. We conclude with an extensive set of experiments which demonstrate the effectiveness of our approach.

2 Propagating Labels with Infections

In this section we present our infection-based method for semi-supervised learning. We are given as input a directed weighted graph $G = (V, E, W)$, as well as a subset of labeled nodes $S \subseteq V$ referred to as the *seed set*. Each seed node $s \in S$ also comes with a true label $y_s \in \mathcal{Y}$. We denote the unlabeled nodes by $U = V \setminus S$, and set $n = |V|$, $m = |E|$, $L = |\mathcal{Y}|$, and $k = |S|$. In some settings additional node features are available.

We focus on the transductive setting, where the goal is to predict the labels of all non-seed nodes $u \in U$.

The core idea of our method is to propagate labels from labeled to unlabeled nodes using infection dynamics. The process is initialized with all seed nodes in an infected state and all unlabeled nodes in a null state $\varnothing$. Then, a stochastic model of infection dynamics is used to determine how the infectious state of nodes in the graph changes over time, typically as a function of the states of neighboring nodes. To allow for multiple label classes, we consider *competitive* infection models. In these, seed nodes $s \in S$ are initialized with their true labels y_s, and compete in infecting unlabeled nodes.

The models we consider are stochastic and converge to a steady state. This means that, after some time point, the labels of all nodes will not change anymore (we refer to this as process termination, or steady state). Since the process itself is stochastic each instantiation will result in a different value for the labels at termination. For a given infection model, let $Y_{v\ell}$ be the binary random variable indicating whether node v is infected by label ℓ at steady state.[1] Since our goal is to reason about the labels of the nodes, it will be natural to utilize the infection dynamics to generate probabilistic predictions. For each node v, our method outputs a distribution over labels f_v. Each entry $f_{v\ell}$ corresponds to the probability that v had value ℓ at steady state, as a function of the seed set S and its labels. Formally:

$$f_{v\ell}(S, y) = \mathbb{P}\left[Y_{v\ell} = 1\right] = \mathbb{E}\left[Y_{v\ell}\right] \tag{1}$$

Note that ℓ can take values in $1, \ldots, L$ but also $\ell = 0$ for $\varnothing$. The entry f_{v0} therefore describes the (possibly non-negative) probability that v remained uninfected.

As we show in Sec. 2.2, even for simple infection models, computing f exactly is #P-hard. Like many other infection-based methods [25, 16, 15], we resort to a Monte-Carlo approach and estimate f by averaging over infection outcomes Y. Our final predictor $\hat{f}$ is:

$$\hat{f}_{v\ell}(S, y) = \frac{1}{N} \sum\nolimits_{i=1}^{N} Y_{v\ell}^{(i)} \tag{2}$$

where $Y_{v\ell}^{(i)}$ is an indicator for the i^{th} random instance.

In principle, outcomes Y can be evaluated by simulating the infection dynamics. This however is not straightforward for several of the models we consider, such as those with continuous time. In the next section we describe some infection models, and show how $\hat{f}$ can be efficiently computed for them using an alternative graphical representation of the infection process.

We conclude by stating an approximation bound for f. As we can calculate $\hat{f}$ efficiently (see next section) this implies that our method yields an efficient approximation scheme for the true infections probabilities.

Proposition 1. *For every $\epsilon, \delta \in [0, 1]$, if $N \geq \frac{1}{2\epsilon^2} \log \frac{2n(L+1)}{\delta}$, then with probability of at least $1 - \delta$, Algorithm 2 returns $\hat{f}$ such that $\|\hat{f} - f\|_\infty \leq \epsilon$.*

Proof. Note that each $Y_{v\ell}$ is a random variable in $\{0, 1\}$. Furthermore, $\hat{f}$ is an average of Y, and f is the corresponding expectation. The result is obtained by applying the Hoeffding and union bounds. □

2.1 Competitive Infection Models for Graph Labeling

As mentioned above, our SSL method relies on an infection process which infects nodes of the graph with labels. There are of course many variants of infection processes, and we described some relevant ones below.

2.1.1 The Independent Cascade model

Since its introduction in [18], the simple but powerful Independent Cascade (IC) model has been used extensively. The original IC model, briefly reviewed below, is a discrete-time, network-dependent interpretation of the classic Susceptible-Infected-Recovered (SIR) epidemiological model [26]. At time $t = 0$, seed nodes are initialized to an infected state, and all other nodes to a susceptible state. If node u is infected at time step t, then at time $t + 1$ it attempts to infect each of its non-infected out-neighbors $v \in Nei(u)$, and succeeds with probability p_{uv}. If successful, we refer to the edge (u, v) as *active* or *activated*, mark the infection time of v as $\tau_v = t+1$, and set v's *infector* to be u, which we denote by $\rho(v) = u$. The model is therefore parametrized by the set of all edge infection probabilities $\{p_{uv} \mid (u, v) \in E\}$ (given as input via $W_{uv} = p_{uv}$). Once a node becomes infected, it remains in this state. As infections are probabilistic, not all nodes are necessarily infected. The process terminates either when all nodes are infected, or (more commonly) when all infection attempts at some time step are unsuccessful.

The IC model describes the propagation of a *single* infectious content. Hence, it can tell us only *when* and *how* a node is infected, but not by *what*. This motivated a class of *competitive* infection models which support multiple content types. Several competitive IC variants have been proposed [8, 10, 12, 24]. The common theme in these is that nodes inherit the label of their earliest infector (with tie-breaking when needed). All of these are supported by our method. In the supplementary material we show how our approach can also be applied to threshold models [25].

[1] For multilabel tasks, $\mathcal{Y}$ is the set of seed node identities, and f becomes a weighted sum of their labels.

Algorithm 1 BasicInfProp(G, S, y, p, N)

for $i = 1, \ldots, N$ **do**
  Initialize $Y^{(i)}_{u\ell} \leftarrow 0$ for all $u \in U$, $\ell \in \mathcal{Y} \cup \varnothing$
  for $(u, v) \in E$ **do**
    $W_{uv} \leftarrow 1$ with probability p_{uv}, and ∞ o.w.
  for $s \in S$ **do**
    $\texttt{dist}[s][\cdot] \leftarrow$ Dijkstra(G, W, s)
  for $u \in U$ **do**
    $Y^{(i)}_{u,\alpha(u)} \leftarrow 1$ where $\alpha(u) \in \operatorname{argmin}_s \texttt{dist}[s][u]$
Return $\hat{f} = \frac{1}{N}\sum_{i=1}^{N} Y^{(i)}$

2.1.2 Continuous Time Dynamics

While simple and elegant, the IC dynamics are somewhat limited in their expressive power. One important generalization is the Continuous-Time IC model (CTIC) [19]. This model is well suited for SSL as it is flexible, does not require tie-breaking, and allows for incorporating node priors. In this model, a successful infection attempt entails an "incubation period", after which the node becomes infected. Hence, if u succeeds in infecting v at time $\tau_u \in \mathbb{R}^+$, it draws an incubation time $\delta_{uv} \sim D(\theta_{uv})$, and v can become infected at time $\tau_{uv} = \tau_u + \delta_{uv}$. As in the IC model, v inherits the label of its earliest infector $\rho(v) = \operatorname{argmin}_u \tau_{uv}$. The competitive CTIC model generalizes the competitive IC model for an appropriate choice of D, where δ_e is set to 1 with probability p_e, and ∞ with probability $1-p_e$. We will therefore consider a general mixture distribution of activations and incubation times $D(p, \theta)$. Since infections are determined by the earliest successful attempt, the shortest-paths interpretation and algorithm (Sec. 2.2.1) hold for the random graph $G^\delta = (V, E, \delta)$.

2.2 Computing Infections Efficiently

For infection models as in Sec. 2.1, we would like to calculate predictions $\hat{f}$ as in Eq. (2). A naive approach would be to do this via simulating the infection process N times and averaging. This, however, is inefficient for discrete-time IC, requires continuous time simulation for CTIC, and does not apply to general models. We hence provide an equivalent efficient alternative below.

2.2.1 Infections as Shortest Paths

We now present an alternative view of the sampling process, which facilitates efficient implementation and extensions. Consider first the discrete time IC process. For a single instantiation of the process, recall that if u succeeded in infecting v, the edge (u, v) is considered *active*. We use the set of active edges $A \subseteq E$ (sampled throughout the instantiation until termination) to construct the *active graph* $G^A = (V, E, W^A)$ with weights $W^A_e = 1$ for $e \in A$ and $W^A_e = \infty$ for $e \in E \setminus A$. An important observation is that node v is infected at termination iff there exists a path in G^A from *some* seed node $s \in S$ to v with finite weight. We refer to this as an *active path*. Since v's actual infection time τ_v is set by the earliest successful infection, it is also the length of the shortest active path from some $s \in S$.

The above formulation allows us to replace time with graph distances. Let $d_A(u, v)$ be the distance from u to v in G^A. Due the recursive nature of label assignment, it follows that a node v inherits its label from the seed node $s \in S$ whose distance to v is shortest. We refer to s as v's *ancestor*, denoted by $\alpha(v)$ and set $\alpha(v) = \varnothing$ when there are no paths from S to v. We can now express the infection outcomes $Y_{v\ell}$ using distances:

$$Y_{v\ell} = \mathbb{1}\{\ell = y_{\alpha(v)}\}, \qquad \alpha(v) = \operatorname*{argmin}_{s \in S} d_A(s, v) \quad (3)$$

Recall that our motivation here was to compute Y without simulating the dynamics. Since distances d_A depend on edge activations, it is not yet clear why Eq. (3) is useful. An important result by [25] shows ancestors can be computed over a simpler random graph model. Specifically, let $\widetilde{A} \subseteq E$ be a random edge set, where *each edge* $(u, v) \in E$ is sampled *independently* to be in $\widetilde{A}$ with probability p_{uv}. Then, for an appropriately defined $G^{\widetilde{A}}$ and $d_{\widetilde{A}}$, we have:

$$\alpha(v) = \operatorname*{argmin}_{s \in S} d_{\widetilde{A}}(s, v) \quad (4)$$

Thus, to compute each $Y^{(i)}_{v\ell}$ (and hence $\hat{f}$), it suffices to sample edges independently, and compute shortest paths on $G^{\widetilde{A}}$, bypassing the need for simulation. Under this view, f can be thought of as an ensemble of shortest-path predictors, whose weights are set by the dynamics. Algorithm 1 provides a simple implementation of this idea for the discrete time IC model. After sampling edges, the algorithm computes shortest paths (using Dikjstra) from each $s \in S$ to all $u \in U$. Then, each node u is assigned the label of its ancestor $\alpha(u)$. This approach applies to a large class of infection models that admit to a similar graphical form [25].

2.2.2 Improved Efficiency via Modified Dijkstra

Recall that for a single infection instance, a node inherits its label from the closest seed node. Based on this, Algorithm 1 offers a direct approach for computing f, where shortest paths are computed from each of the k seed nodes to every unlabeled node $v \in U$ using k calls to Dijkstra. While correct, this method suffers an unnecessary factor of k on its runtime. To reduce

this overhead, we change Dijkstra's initialization and updates, so that only a single call would suffice. Algorithm 2 implements this idea for the more general CTIC model and allows for node priors (Sec. 2). The correctness of the algorithm is stated below, and a proof is provided in the supplementary material.

Proposition 2. *Algorithm 2 correctly computes the average infection probabilities $\hat{f}$ in Eq. (1).*

The worst-case complexity of Dijkstra, and hence of each iteration in Algorithm 2, is $O(m+n\log n)$. Other implementations of Dijkstra which support further parallelization or GPUs [30] can also be modified for our setting. Nonetheless, the practical run time of Algorithm 2 can be, and typically is, much better, for two reasons. First, note that only the subset of active edges are traversed (and sampled on the fly), and only nodes which are reachable from S are processed. The infection parameters p therefore induce a trade-off between the influence diameter of S and the run time (empirical demonstration in Fig. 3 (left)). Second, many settings require "hard" predictions $\hat{y} \in \mathcal{Y}$, typically set by $\hat{y}_v = \text{argmax}_\ell \hat{f}_{u\ell}$. Hence, for $\hat{y}_v$ to be correct, it suffices that $\hat{f}_{u,y_u} \geq \hat{f}_{u\ell}$ for all $\ell \in \mathcal{Y}$, which does not require the full convergence stated in Proposition 1 (empirical demonstration in Fig. 3 (right)).

In this section we showed how infection outcomes can be computed efficiently. It is therefore only natural to ask - what is it that infections compute? In the next section we show that f is in fact the solution to a quadratic optimization objective, whose weights depend on the infection dynamics.

Algorithm 2 InfProp (G, S, y, D, q, N)

for $i = 1, \ldots, N$ **do**
  Initialize $Y^{(i)}_{u\ell} \leftarrow 0$ for all $u \in U$, $\ell \in \mathcal{Y} \cup \varnothing$
  for $v \in U$ **do**
    $\texttt{dist}[v] \leftarrow \infty$, $\texttt{y}[v] \leftarrow \varnothing$
  for $s \in S$ **do**
    $\texttt{dist}[s] \leftarrow 0$, $\texttt{y}[s] \leftarrow y_s$
    push s into min-queue Q
  while Q is not empty **do**
    pop v from Q ▷ break ties randomly
    for $u \in Nei(v)$ **do**
      sample $\delta_{vu} \sim D(\theta, p)$ ▷ incubation time
      if $\delta_{vu} = \infty$ **then** continue
      $\texttt{alt} \leftarrow \texttt{dist}[v] + w^A_{vu} + q_u(\texttt{y}[v])$ ▷ penalize
      if $\texttt{alt} < \texttt{dist}[u]$ **then**
        $\texttt{dist}[u] = \texttt{alt}$
        $\texttt{y}[u] \leftarrow \texttt{y}[v]$ ▷ u inherits label from parent v
        update u in Q with $\texttt{dist}[u]$
  $Y^{(i)}_{u,\texttt{y}[v]} \leftarrow 1$ for all $u \in U$
Return $\hat{f} = \frac{1}{N}\sum_{i=1}^{N} Y^{(i)}$

3 What do infections optimize?

Many SSL methods propose an optimization objective which encodes some notion of smoothness. For instance, the classic LabelProp algorithm [46] encourages adjacent nodes to agree on their predicted labels by minimizing a quadratic penalty term:

$$f_{\text{LP}} = \underset{f'}{\text{argmin}} \sum_{\ell} \sum_{u,v} W_{uv}(f'_{u\ell} - f'_{v\ell})^2 \qquad (5)$$

for predictions f' and symmetric and normalized weights W, subject to $f'_s = y_s$ for all $s \in S$. In this section we show that InfProp has a related interpretation. Specifically, we show that the InfProp predictions f minimize the quadratic objective in Eq. (13).

While similar in structure, the fundamental difference between Eqs. (5) and (13) lies in how the weights are determined. In LabelProp (and variants), edge weights are given as input, and are typically set according to some feature-based similarity measure. In this sense, each W_{uv} is a local function of the features of u and v. In contrast, weights in Eq. (13) are set in a global manner. As we show next, each weight is a function of the infection dynamics, of the specific seed set S, and, if available, of the features of *all* nodes. To demonstrate this, and to see why Eq. (13) holds, it will be helpful to analyze InfProp from a spectral perspective.

3.1 A Laplacian Interpretation for InfProp

An interesting property of LabelProp is that its objective can be expressed via the graph Laplacian. For a directed weighted graph, the standard Laplacian is:

$$\mathcal{L}_{\text{LP}} = D - W \qquad (6)$$

where D is a diagonal matrix with $D_{uu} = \sum_v W_{uv}$ (and W is symmetric). The output of LabelProp can be computed by solving the system $\mathcal{L}_{\text{LP}} f' = 0$ for the unlabeled nodes. We now show that the infection-based predictions of InfProp also correspond to the solution of a certain Laplacian system which is determined by the seed set and the infection dynamics.

Consider a single infection instance, and denote by $T_{uv}(S)$ the random variable indicating whether u was infected by v for seed S, namely $T_{uv}(S) = \mathbb{1}_{\{u=\rho(v)\}}$. We refer to the matrix T as the *infector matrix*. Further denote by $\boldsymbol{T}$ the expected infector matrix $\boldsymbol{T}(S) = \mathbb{E}[T(S)]$. We use this to define the following Laplacian:

$$\mathcal{L}(S) = I - \boldsymbol{T}(S) \qquad (7)$$

Note that $\mathcal{L}$ is defined over the same graph G, but need not be symmetric. We now show that $\mathcal{L}$ is indeed a Laplacian matrix, and that it can be used to infer f.

Lemma 1. *The infection-based predictions f in Eq. (2) are also the solution to the Laplacian system:*

$$\mathcal{L}(S)f = \boldsymbol{b}(S) \tag{8}$$

where:

$$\boldsymbol{b}_{u\ell}(S) = \sum_v b^{(S)}_{vu\ell}, \qquad b^{(S)}_{vu\ell} = cov\left[T_{vu}(S), Y_{u\ell}\right]$$

For conciseness, we defer the full proof to the supplementary material, and show here a useful special case.

Lemma 2. *If T and Y are uncorrelated, then the infection-based predictions f in Eq. (2) are also the solution to the homogeneous Laplacian system:*

$$\mathcal{L}(S)f = 0 \tag{9}$$

Proof. We first show that $\mathcal{L}$ is a graph Laplacian, namely that the sum of each row in $\boldsymbol{T}$ equals the corresponding diagonal element in I, which is 1. Since rows in T have only one non-zero entry of value one, each row in $\boldsymbol{T}$ is positive and sums to one. Note that $\boldsymbol{T}_{u\cdot}$ provides a distribution over the infectors of u.

We now prove Eq. (9). By definition, the label of each node at steady state is set to be that of its infector, namely $Y_{v\ell} = Y_{\rho(v),\ell}$ for all v and ℓ, or simply $Y = TY$. Using Eq. (2) and applying expectation, we have:

$$f(S) = \mathbb{E}\left[Y\right] = \mathbb{E}\left[T(S)Y\right] \tag{10}$$

When T, Y are uncorrelated, $\mathbb{E}\left[TY\right] = \mathbb{E}\left[T\right]\mathbb{E}\left[Y\right]$, hence $f = \boldsymbol{T}f$. Rearranging concludes our proof. □

3.2 InfProp as Optimization

We next use the Laplacian insight above to provide an objective minimized by the InfProp solution. Begin by noting that for LabelProp, Eq. (6) can be restated as:[2]

$$\begin{aligned} f_{\text{LP}} &= \operatorname*{argmin}_{f'} \|\mathcal{L}_{\text{LP}} f'\|_2^2 \\ &= \operatorname*{argmin}_{f'} \sum_\ell \sum_u \left(f'_u - \sum_v W_{uv} f'_v\right)^2 \end{aligned} \tag{11}$$

where minimization is only over the unlabeled nodes. This gives an alternative quadratic objective, which lower-bounds Eq. (5),[3] and directly expresses the steady-state of LabelProp's averaging dynamics. In a similar fashion, we can restate Eq. (8) as:

$$f(S) = \operatorname*{argmin}_{f'} \|\mathcal{L}(S)f' - \boldsymbol{b}(S)\|_2^2 \tag{12}$$

[2]Using the symmetry and normalization of W

[3]Using Jensen's inequality

Expanding and denoting $w^{(S)}_{uv} = \boldsymbol{T}_{uv}(S)$ provides the general objective of our method:

$$\min_{f'} \sum_\ell \sum_u \left(f'_{u\ell} - \sum_v \left(w^{(S)}_{uv} f'_{v\ell} + b^{(S)}_{uv\ell}\right)\right)^2 \tag{13}$$

Note that Eq. (13) and Eq. (11) are structurally equivalent up to the bias terms, which disappear when the conditions of Lemma 2 hold. The critical difference is that the weights in Eq. (13) are intricate functions of the dynamics, rather than just scalars. Through their dependence on $\boldsymbol{T}$ and Y, the weights and bias terms in Eq. (13) are in fact functions of the dynamics. In this sense, $w^{(S)}_{uv}$ quantifies how well v relays information from S to u, which depends on the entire graph. Similarly, the term $b^{(S)}_{uv\ell}$ quantifies consistency between the identity of u's infector (v) and the inherited label (ℓ). This means that frequent yet indecisive infectors are penalized, while reliable nodes remain unbiased.

Finally, note that the optimization interpretation above does not offer a better optimization scheme, since calculating the weights $w(S)$ and $b(S)$ would require sampling, and hence our InfProp sampling algorithms from Sec. 2.2 would be a simpler approach.

4 Other Learning Settings

In this section we briefly describe how our method extends to other learning settings used in our experiments. For more details please see the supp. material.

Incorporating features and priors: Many network based datasets include additional node features or priors. Our method incorporates priors directly into the CTIC dynamics by penalizing incubation times. Denote by $\boldsymbol{\rho}_{v\ell}$ the prior for labeling v with ℓ, and let $q : [0, 1] \to \mathbb{R}$ be a penalty function. If u succeeds in infecting v with ℓ, the incubation time δ_{uv} is penalized by an additional $q(\boldsymbol{\rho}_{v\ell})$. For a decreasing q , high priors induce low penalties, and vice versa. Although penalties are deployed locally, they delay the global propagation of the penalized label across the graph.

Confidence and active learning: Recall that v remains uninfected with probability f_{v0}. Hence, $\sigma_v(S) = 1 - f_{v0}$ serves as a natural measure of confidence. We use this as a selection criteria for an active setting where the goal is to choose a seed set of size k. The objective we consider coincides with the well-studied notion of influence [25], which is monotone and submodular and admits to an efficient greedy approximation scheme. Our method offers a tractable alternative to existing active SSL methods [23, 17, 21].

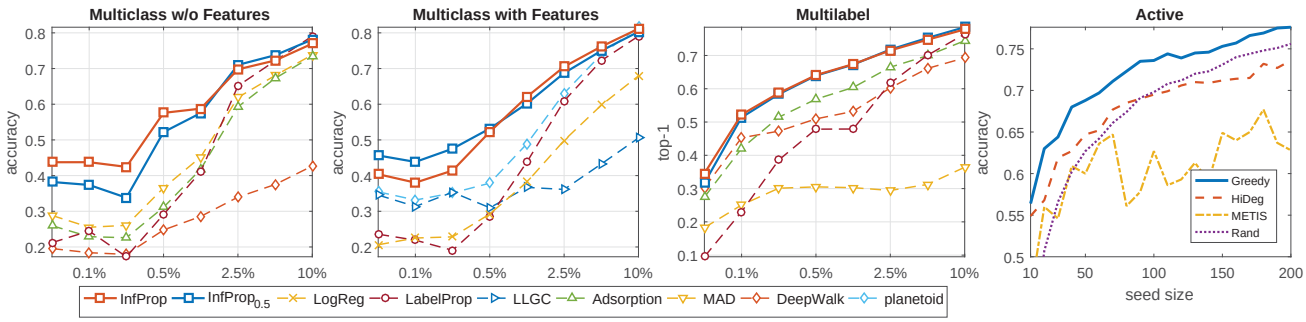

Figure 2: Results on the CoRA dataset for various learning settings.

5 Related Work

As in unsupervised learning, methods for semi-supervised learning are often based on assumptions regarding the structure of the unlabeled data. One such assumption is that of *smoothness*, which states that examples that are close are likely to have similar labels. In the classic Label Propagation algorithm [46], adjacent nodes in the graph are encouraged to agree on their labels via a quadratic penalty. Some variants add regularization terms [4], allow for label uncertainty [41], or include normalization and unanchored seeds [45]. An interesting property of this approach is that, for undirected graphs, the objective can be interpreted as the solution to a Laplacian system, as the outcome of a random walk process, and as the fixed point of a dynamic process of label propagation.

The above methods were originally designed for graphs that approximate the data density by encoding similarity in feature space. In many cases, these graphs need to be constructed from the sample set. Since many modern datasets originate from real-world networks, several SSL methods have been recently introduced that utilize the graph as an additional source of information. Motivated by the success of deep embedding techniques [31], these methods embed the nodes of a graph into a low-dimensional vector space, which can then be used in various ways. When the data includes only the graph, the embeddings can be used as input for an off-the-shelf predictor [35]. When the data includes additional node features, the embedding can act as a regularizer for a standard loss over the labeled nodes [44, 27]. In contrast to classic methods, these methods propagate *features* rather than labels.

An alternative but related method for utilizing graphs is to consider shortest paths as a measure of closeness between nodes. The authors of [1] show that Laplacians and shortest paths are in some sense special cases of resistance distances, and propose (but do not evaluate) a new regularizer. Other methods construct ad-hoc graphs whose shortest paths approximate density-based distances [34, 6]. Our method, which can be viewed as an ensemble of shortest paths, can be applied to any graph. A recent work [14] proposes a method for SSL in directed graphs based on distance diffusion. As they consider distances from unlabeled to labeled nodes, each instance of their model is computationally intensive, and requires an approximation scheme. In contrast, we consider distances from labeled to unlabeled nodes, which can be computed efficiently. While for a specific setting (symmetric weights and a certain link function) both models overlap, in this paper we consider a more general setup.

As mentioned, our method draws on the rich literature of infection models and diffusion processes over networks. Infection models have been used for describing the propagation of information, innovation, behavioral norms, influence, and others. Such models were popularized in a seminal work on choosing a seed set which maximizes influence [25]. In our work we leverage this concept for choosing nodes in an active semi-supervised learning setting. Other methods have utilized infection models for network inference [19], influence maximization [25] estimation [16, 15] and prediction [37], and personalized marketing [13].

6 Experiments

We evaluated our method on various learning tasks over three benchmark dataset collections, which include network based data for multiclass learning with

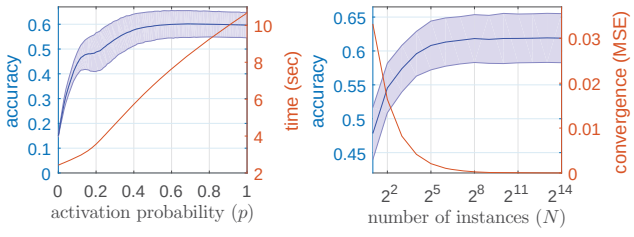

Figure 3: Activation tradeoff and convergence

	Multiclass (Accuracy / MSE)					Multilabel (Top-1 / AUC)					
	CoRA	DBLP	Flickr	IMDb	Industry	Amazon	CoRA	IMDb	PubMed	Wikipedia	YouTube
InfProp	**0.59** / **0.56**	0.73 / **0.42**	**0.79** / **0.38**	0.56 / **0.49**	**0.21** / 0.91	0.38 / 0.72	**0.52** / **0.83**	0.15 / 0.52	**0.68** / **0.83**	0.46 / **0.84**	**0.24** / **0.76**
InfProp$_{0.5}$	0.58 / 0.64	0.74 / 0.46	0.78 / 0.38	0.55 / 0.49	**0.21** / **0.90**	**0.39** / 0.71	0.51 / 0.82	**0.16** / 0.52	0.67 / 0.80	0.45 / 0.83	0.23 / 0.76
ShortPaths	0.53 / 0.87	0.63 / 0.74	0.65 / 0.70	0.55 / 0.90	0.16 / 1.57	0.31 / 0.61	0.39 / 0.66	0.15 / 0.51	0.58 / 0.68	0.24 / 0.59	0.16 / 0.56
LabelProp	0.41 / 0.74	0.60 / 0.59	0.33 / 0.90	0.50 / 0.63	0.14 / 0.99	0.36 / **0.75**	0.23 / 0.69	0.16 / **0.56**	0.40 / 0.66	0.12 / 0.59	0.04 / 0.58
Adsorption	0.42 / 0.99	0.54 / 0.99	0.72 / 0.99	0.56 / 0.99	0.14 / 0.99	0.34 / 0.66	0.42 / 0.75	0.14 / 0.55	0.57 / 0.70	0.34 / 0.75	0.16 / 0.66
MAD	0.45 / 0.99	0.20 / 1.00	0.75 / 0.99	**0.58** / 0.99	0.16 / 1.00	0.34 / 0.66	0.25 / 0.49	0.13 / 0.55	0.60 / 0.71	0.04 / 0.00	0.19 / 0.68
DeepWalk	0.29 / 0.86	**0.77** / 0.62	0.49 / 0.73	0.50 / 0.56	0.17 / 0.92	0.10 / 0.55	0.45 / 0.64	0.11 / 0.50	0.38 / 0.53	**0.58** / 0.65	0.10 / 0.58

Table 1: Results for experiments on data without features.

[39] and without features [38], and multilabel learning [33]. The datasets include diverse networks such as social networks, citation and co-authorship graphs, product and item networks, and hyperlink graphs. Dataset statistics can be found in the supplementary material.

Our experimental setup follows the standard graph-based semi-supervised learning evaluation approach. Specifically, in each instance we draw a seed set of size k uniformly at random, acquire its labels, and then use the graph and labeled seed set to generate labels for all nodes. We repeat this procedure for 10 random seed set selections and for various values of k (where k is set to be a fixed proportion of the number of nodes in the graph) and report average results.

We compared our method to current state-of-the-art baselines, which include spectral methods as well as deep embedding methods. For tasks which do not include features, these included LabelProp [46], Adsorption [4], MAD [41], and the feature-agnostic deep method DeepWalk [35]. For tasks which do include features, we compared to the prior-supporting spectral method LLGC [45], the recent feature-based deep method Planetoid [44], LabelProp as a graph-only baseline, logistic regression (LogReg) as a features-only baseline, and a baseline where labels are set by shortest paths in G (ShortPaths). For the active setting (Fig. 2), we compared our approach (Greedy) to METIS [22]), to choosing high-degree nodes (HiDeg), and to random seeds (Rand).

For our method (InfProp) we used exponential incubation times $\delta \sim \text{Exp}(\theta)$. As in many works (e.g., [25, 14]), we used $\theta_{uv} = 1/d_u$ for all node pairs $(u, v) \in E$, where d_u is the out-degree of u. We set the number of random instances to $N = 1,000$. Fig. 3 (right) demonstrates accuracy and convergence as a function of N. We show results for two variants: InfProp, where we set activation probabilities to $p = 1$ for all edges, and InfProp$_{0.5}$, where $p = 0.5$. In addition to providing a confidence measure, InfProp$_{0.5}$ is much faster, while on average achieving 0.99% of the performance of InfProp. Fig. 3 (left) demonstrates the tradeoff in accuracy and runtime when varying p.

Most methods we consider naturally output probabilistic "soft" labels as predictions. We therefore evaluate performance using both probabilistic or order-based performance measures, as well as performance measures for "hard" labels. For all methods, hard labels were generated by choosing the label with the highest value. Tables 1 and 2 include results for all datasets for $k = 0.1\%$ of the data. Fig. 2 shows results for various values of k on the CoRA dataset, since it appears in all learning tasks. As can be seen, InfProp consistently performs well across all settings.

7 Conclusions

In this work we presented an SSL method where labels propagate over the graph using dynamic infection models. These models have a strong connection to short-path ensembles and to graph Laplacians, allow for efficient computation, and show empirical potential. Our work was motivated by the idea that different graph types may require different dynamics, which led us to consider alternatives to random walks and averaging dynamics. We used a competitive CTIC variant, but other infection models (and other dynamics in general) can be considered. The choice of dynamics can serve as a means for expressing prior knowledge and for encoding structure and dependencies.

The models we use have very few tunable parameters. Nonetheless, one can consider highly parametrized models. Such parameters can be used to control infection probabilities, be node or label specific, relate to features, and even adjust the dynamics themselves. The stochastic nature of the models and the nonlinearity of the dynamics makes learning these parameters a challenging task, which we leave for future work.

(Acc. / MSE)	CiteSeer	CoRA	PubMed
InfProp	0.47 / 0.72	**0.62** / 0.59	**0.74** / 0.46
InfProp$_{0.5}$	**0.48** / 0.74	0.60 / **0.57**	0.72 / **0.41**
ShortPaths	0.39 / 0.73	0.44 / 0.72	0.68 / 0.51
LogReg	0.44 / 0.78	0.37 / 0.81	0.45 / 0.65
LabelProp	0.39 / 0.77	0.38 / 0.78	0.40 / 0.67
LLGC	0.45 / **0.71**	0.49 / 0.69	0.44 / 0.67
Planetoid[4]	0.41 / 0.94	0.53 / 0.89	0.68 / 0.64

Table 2: Results on data with features.

[4]Our results differ from those in [44] since their evaluation is based on a specific seed of fixed size, chosen by a different procedure, evaluated only on 1000 test samples, and early-stopped using test data.

References

[1] Alamgir, M., and Luxburg, U. V. Phase transition in the family of p-resistances. In *Advances in Neural Information Processing Systems* (2011), pp. 379–387.

[2] Aleliunas, R., Karp, R. M., Lipton, R. J., Lovasz, L., and Rackoff, C. Random walks, universal traversal sequences, and the complexity of maze problems. In *Foundations of Computer Science, 1979., 20th Annual Symposium on* (1979), IEEE, pp. 218–223.

[3] Altun, Y., McAllester, D., and Belkin, M. Maximum margin semi-supervised learning for structured variables. *Advances in neural information processing systems 18* (2006), 33.

[4] Baluja, S., Seth, R., Sivakumar, D., Jing, Y., Yagnik, J., Kumar, S., Ravichandran, D., and Aly, M. Video suggestion and discovery for youtube: taking random walks through the view graph. In *Proceedings of the 17th international conference on World Wide Web* (2008), ACM, pp. 895–904.

[5] Belkin, M., and Niyogi, P. Semi-supervised learning on Riemannian manifolds. *Machine Learning 56*, 1 (2004), 209–239.

[6] Bijral, A. S., Ratliff, N., and Srebro, N. Semi-supervised learning with density based distances. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence* (2011), UAI'11, pp. 43–50.

[7] Blum, A., and Chawla, S. Learning from labeled and unlabeled data using graph mincuts. *Proc. 18th International Conf. on Machine Learning* (2001).

[8] Borodin, A., Filmus, Y., and Oren, J. Threshold models for competitive influence in social networks. In *WINE* (2010), vol. 6484, Springer, pp. 539–550.

[9] Broder, A. Z., Karlin, A. R., Raghavan, P., and Upfal, E. Trading space for time in undirected s-t connectivity. *SIAM Journal on Computing 23*, 2 (1994), 324–334.

[10] Budak, C., Agrawal, D., and El Abbadi, A. Limiting the spread of misinformation in social networks. In *Proceedings of the 20th international conference on World wide web* (2011), ACM, pp. 665–674.

[11] Chapelle, O., Schölkopf, B., Zien, A., et al. *Semi-supervised learning*. MIT press Cambridge, 2006.

[12] Chen, W., Collins, A., Cummings, R., Ke, T., Liu, Z., Rincon, D., Sun, X., Wang, Y., Wei, W., and Yuan, Y. Influence maximization in social networks when negative opinions may emerge and propagate. In *Proceedings of the 2011 SIAM International Conference on Data Mining* (2011), SIAM, pp. 379–390.

[13] Chen, W., Wang, C., and Wang, Y. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (2010), ACM, pp. 1029–1038.

[14] Cohen, E. Semi-supervised learning on graphs through reach and distance diffusion. *arXiv preprint arXiv:1603.09064* (2016).

[15] Cohen, E., Delling, D., Pajor, T., and Werneck, R. F. Sketch-based influence maximization and computation: Scaling up with guarantees. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (2014), ACM, pp. 629–638.

[16] Du, N., Song, L., Rodriguez, M. G., and Zha, H. Scalable influence estimation in continuous-time diffusion networks. In *Advances in neural information processing systems* (2013), pp. 3147–3155.

[17] Gadde, A., Anis, A., and Ortega, A. Active semi-supervised learning using sampling theory for graph signals. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), ACM, pp. 492–501.

[18] Goldenberg, J., Libai, B., and Muller, E. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing letters 12*, 3 (2001), 211–223.

[19] Gomez Rodriguez, M., Leskovec, J., and Krause, A. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2010), KDD '10, pp. 1019–1028.

[20] Gomez-Rodriguez, M., Song, L., Du, N., Zha, H., and Schölkopf, B. Influence estimation and maximization in continuous-time diffu-

sion networks. *ACM Trans. Inf. Syst. 34*, 2 (Feb. 2016), 9:1–9:33.

[21] Gu, Q., and Han, J. Towards active learning on graphs: An error bound minimization approach. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on* (2012), IEEE, pp. 882–887.

[22] Guillory, A., and Bilmes, J. A. Label selection on graphs. In *Advances in Neural Information Processing Systems* (2009), pp. 691–699.

[23] Guillory, A., and Bilmes, J. A. Active semi-supervised learning using submodular functions. *arXiv preprint arXiv:1202.3726* (2012).

[24] He, X., Song, G., Chen, W., and Jiang, Q. Influence blocking maximization in social networks under the competitive linear threshold model. In *Proceedings of the 2012 SIAM International Conference on Data Mining* (2012), SIAM, pp. 463–474.

[25] Kempe, D., Kleinberg, J., and Tardos, É. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (2003), ACM, pp. 137–146.

[26] Kermack, W. O., and McKendrick, A. G. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 115*, 772 (1927), 700–721.

[27] Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[28] Lin, F., and Cohen, W. W. The multi-rank bootstrap algorithm: Self-supervised political blog classification and ranking using semi-supervised link classification. In *Proceedings of the Second International Conference on Weblogs and Social Media, ICWSM 2008, Seattle, Washington, USA, March 30 - April 2, 2008* (2008).

[29] Lofgren, P., Banerjee, S., Goel, A., and Comandur, S. FAST-PPR: scaling personalized pagerank estimation for large graphs. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014* (2014), pp. 1436–1445.

[30] Martín, P. J., Torres, R., and Gavilanes, A. Cuda solutions for the sssp problem. In *International Conference on Computational Science* (2009), Springer, pp. 904–913.

[31] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (2013), pp. 3111–3119.

[32] Nadler, B., Srebro, N., and Zhou, X. Semi-supervised learning with the graph laplacian: The limit of infinite unlabelled data. *Advances in neural information processing systems 21* (2009).

[33] Nandanwar, S., and Murty, M. N. Structural neighborhood based classification of nodes in a network. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), KDD '16, pp. 1085–1094.

[34] Orlitsky, A., et al. Estimating and computing density based distance metrics. In *Proceedings of the 22nd international conference on Machine learning* (2005), ACM, pp. 760–767.

[35] Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), ACM, pp. 701–710.

[36] Rifai, S., Dauphin, Y., Vincent, P., Bengio, Y., and Muller, X. The manifold tangent classifier. In *NIPS* (2011), vol. 271, p. 523.

[37] Rosenfeld, N., Nitzan, M., and Globerson, A. Discriminative learning of infection models. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining* (2016), WSDM '16, pp. 563–572.

[38] Saha, T., Rangwala, H., and Domeniconi, C. Flip: active learning for relational network classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2014), Springer, pp. 1–18.

[39] Sen, P., Namata, G. M., Bilgic, M., Getoor, L., Gallagher, B., and Eliassi-Rad, T. Collective classification in network data. *AI Magazine 29*, 3 (2008), 93–106.

[40] Sindhwani, V., Niyogi, P., and Belkin, M. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of the 22nd international conference on Machine learning* (2005), ACM, pp. 824–831.

[41] Talukdar, P. P., and Crammer, K. New regularized algorithms for transductive learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2009), Springer, pp. 442–457.

[42] Vishnoi, N. K., et al. Lx= b. *Foundations and Trends® in Theoretical Computer Science 8*, 1–2 (2013), 1–141.

[43] von Luxburg, U., Radl, A., and Hein, M. Getting lost in space: Large sample analysis of the resistance distance. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.* (2010), pp. 2622–2630.

[44] Yang, Z., Cohen, W. W., and Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning* (2016), ICML'16, pp. 40–48.

[45] Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. Learning with local and global consistency. In *NIPS* (2003), vol. 16, pp. 321–328.

[46] Zhu, X., Ghahramani, Z., Lafferty, J., et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML* (2003), vol. 3, pp. 912–919.

Semi-Supervised Learning with Competitive Infection Models: Supplementary Material

Nir Rosenfeld
Harvard University

Amir Globerson
Tel Aviv University

1 Proof of Proposition 2 in Main Text

In this section we prove the correctness of our algorithm. The proof considers the more general CTIC infection dynamics and allows for node features or priors (via a penalty function).

In the infection dynamics presented in the paper, once a node's label is set, it remains fixed. In contrast, during the course of the algorithm a node's label may change with each distance update. It therefore remains to show that the algorithm outputs the desired labels. For basing our claim it will be easier to assume that instead of initially inserting all seed nodes into Q, we add a dummy root node r to V, with edges of length $w_{rs} = 0$ to all $s \in S$, and initialize Q to include only r. It is easy to see that after extracting r from Q, we return to our original algorithm.

Recall that the standard single-source Dijkstra algorithm offers three important guarantees: (1) the estimated distance of an extracted node is correct (and remains unchanged), (2) nodes are extracted in increasing order of their true distance, and (3) the distance estimates always upper-bound the true distances. Now, let $v \in U$ be a node that has just been extracted, and assume by induction that the labels of all previously extracted nodes (which include all seed nodes) are correct.[5] The above guarantees tell us that the distance from r to v is correct, and all nodes on the shortest path from r to v have already been extracted. This is true even when a penalty is incurred, as it can only increase the distance estimate. As these nodes are assumed to be correctly labeled, v inherits the correct label as well, as by construction its shortest path from r goes through exactly one seed node. The correctness of the seed node labels gives the induction basis, which concludes the proof.

[5] Correct in the sense of the algorithm, not in the sense of their true labels.

2 Extensions

In this section we describe in more depth several useful extensions of our method. These include applying out method to the Linear Threshold model, incorporating node features and priors into the infection dynamics, and a framework for using our method in an active SSL setting.

2.1 The Linear Threshold model

In this section we show how InfProp can be applied with the Linear Threshold (LT) dynamics, rather than the IC or CTIC dynamics discussed in the text. This includes adapting the algorithm for computing expected labels to the LT model, as well as supporting node features and priors.

The input to the LT model is a weighted graph $G = (V, E, W)$ and an initial set of infected seed nodes S. We assume that weights are positive, and that for each $v \in V$, the sum of incoming weights $\sum_v W_{uv}$ is at most 1 (thought can be strictly less than 1). Before the process begins, each node u is assigned a threshold η_u sampled uniformly at random from the interval $[0, 1]$. The dynamics then progress in discrete time steps, where at time t, a susceptible node v becomes infected if the weighted sum of its infected neighbors exceeds its threshold. Denoting by $I_u(t)$ an indicator of whether u is infected at time t, v is infected at time t if:

$$\sum_u W_{uv} I_v(t-1) \geq \eta_u \tag{14}$$

Note that the randomness in this model comes from the threshold η; given η, the dynamics are deterministic.

Intriguingly, the authors of [25] show that the LT model can also be equivalently expressed using a graphical perspective using active edge sets. Here, however, edges are no longer sampled independently. Instead, for each node v, only at most one incoming edge will become active in each instance. Specifically, for each node v, an edge $(u, v) \in E$ is going to be se-

lected to be the active incoming edge with probability $p_{uv} = W_{uv}$, where no incoming edges are active with probability $W_{-u} = 1 - \sum_v W_{uv}$. Then, for a given instance, v is infected if and only if there is an active path to v from some seed node in S.

An interesting interpretation of the above is that the chosen active edge (u, v) can be thought of as corresponding to the node u whose infection triggered the infection of v by crossing the threshold. Under this view, a type-dependent specification of the above model is one where v inherits its label from its triggering neighbor v, which we refer to as his infector. This model readily applies to the competitive setting which we consider. In terms of implementation, the only necessary modification to the algorithm is the way in which active edges are sampled.

The competitive LT model can also incorporate node priors using penalty terms. Specifically, the node-label prior $\boldsymbol{\rho}_{v\ell}$ will induce a multiplicative penalty $q_v(\ell) \in [0, 1]$ on the original weights W_{uv} when u tries to infect v with label ℓ. Thus, given that u has label ℓ, the penalty reduces the probability that it will be the infector of v. To implement this, when u is expanded, the edge (u, v) is sampled to be active with the penalized probability, and all other incoming weights (including the complementing W_{-u}) are re-normalized.

2.2 Incorporating node features and priors

In addition to the graph, many network based datasets include node or edge features. These can be used to generate node-specific class priors. In this section we describe a novel generalization of the competitive infection models introduced above which incorporates class priors into the dynamics. In this setting, our approach is to first train a probabilistic classifier (e.g., logistic regression) on the labeled seed set, and then use its predictions on the unlabeled nodes as a prior for our model.

Our method utilizes node priors by transforming them into penalties on incubation times. Consider a single instance of an infection process. Assume node u has just been infected with label $\ell \in \mathcal{Y}$, and succeeded in its attempt to infect node v with an incubation time of δ_{uv}. If δ_{uv} is small, then it is very likely that v will get infected with ℓ as well. On the other hand, if δ_{uv} is large, then other nodes might have a chance to infect v with other labels. This motivates the idea of further *penalizing* the infection time of a node according to its prior. We do this by adding a label-dependent penalty $q_v(\ell)$ to δ_{uv}, as a function of the prior $\boldsymbol{\rho}_{v\ell}$. We use the link function $q_v(\ell) = -\log(\boldsymbol{\rho}_{v\ell})$, which maps low priors into large penalties, and high priors into low penalties, where $\boldsymbol{\rho}_{v\ell} = 1$ entails no penalty. Hence, setting $\boldsymbol{\rho}_{v\ell} = 1$ for all v, ℓ recovers the original model.

Note that while the priors are deployed locally, their effect is in fact global, as penalizing a node's infection time delays the potential propagation of its acquired label throughout the graph. This increases the significance of nodes which are central to the infection process, and reduces the significance of those which play a small role in it, a property captured by our notion of confidence (Sec. 2.3). The strength of the above formulation lies in its ability to introduce non-linear label dependencies to the actual infection dynamics. To see this, we can write the original predictions as:

$$f = \mathbb{E}_{\delta \sim D}\left[\mathcal{A}^{\delta}\mathcal{S}\right] = \mathbb{E}_{\delta \sim D}\left[\mathcal{A}^{\delta}\right]\mathcal{S} \tag{15}$$

where $\mathcal{A}^{\delta}_{vs} = \mathbb{1}_{\{s=\alpha(v)\}}$ indicates ancestors in G^{δ}, and $\mathcal{S}_{s\ell} = \mathbb{1}_{\{y_s=\ell\}}$ indicates the seed nodes' true labels. This shows that predictions are non-linear in the propagation of the seed nodes, but linear in the labels. In the prior-dependent model, the above no longer holds, as activation times are now label-dependent. In the supplementary material we show how to efficiently compute predictions for this model as well.

2.3 Confidence and Active Learning

Recall that a node v has a probability f_{v0} of not being infected by *any* label. This suggests a very natural measure of *confidence* in our prediction, namely:

$$\sigma_v(S) = 1 - f_{v0} = \sum_{\ell=1}^{L} f_{v\ell}, \qquad \sigma(S) = \sum_v \sigma_v(S) \tag{16}$$

The function σ quantifies the confidence *in the labeling*. This is conceptually different from confidence *in a label*. Our model supports both concepts distinctly. The former is controlled by the activations p, as they determine reachability in the active graph and are agnostic to labels. The latter is controlled by θ, as it affects the speed of propagation of the labels.

The notion of confidence allows us to apply our method to an active learning setting. Instead of assuming the seed is given as input, in this setting we are allowed to *choose* the seed set, often under a cardinality budget constraint. The goal is then to choose the seed set which leads to a good labeling. Various graph-based notions have been suggested as objectives for active seed selection, such as those based on graph cuts [23], graph signals [17], and generalization error [21]. Such methods however either optimize an adversarial objective, or simply offer a heuristic solution. In contrast, using σ as a seed-selection criterion offers an optimistic alternative, as summing over all classes makes it indifferent to the actual (latent) labels. In Sec. 6 we show that this also leads to good predictions.

The confidence term σ coincides with the well-studied notion of *influence*, defined as the expected number of nodes a seed will infect. In [25] it is shown that for various settings, influence is submodular, and therefore admits to a greedy $(1-1/\epsilon)$-approximation scheme. Any algorithm for maximizing influence efficiently (e.g., [15, 20]), can therefore be adopted for out setting.

3 Non-Homogeneous Laplacian

Here we prove that:

$$\mathcal{L}(S)f = \boldsymbol{b}(S)$$

where:

$$\boldsymbol{b}_{u\ell}(S) = \sum_v b^{(S)}_{vu\ell}, \qquad b^{(S)}_{vu\ell} = \mathrm{cov}\left[T_{vu}(S), Y_{u\ell}\right]$$

For clarity we drop the notational dependence on S. We begin by expanding $f_{u\ell}$ using Y and T:

$$\begin{aligned}
f_{u\ell} &= \mathbb{E}\left[Y_{u\ell}\right] = \mathbb{E}\left[T_{u\cdot}Y_{\cdot\ell}\right] \\
&= \mathbb{E}\left[\sum_v T_{uv}Y_{v\ell}\right] \\
&= \sum_v \mathbb{E}\left[T_{uv}Y_{v\ell}\right] \\
&= \sum_v \left(\mathbb{E}\left[T_{uv}\right]\mathbb{E}\left[Y_{v\ell}\right] + \mathrm{cov}\left[T_{uv}Y_{v\ell}\right]\right) \\
&= \sum_v \left(\boldsymbol{T}_{uv}f_{v\ell} + b_{uv\ell}\right)
\end{aligned}$$

where the final step is true for the product of general random variables. Rewriting in matrix form gives: $f = \boldsymbol{T}f + \boldsymbol{b}$. Rearranging we get: $(I - \boldsymbol{T})f = \mathcal{L}f = \boldsymbol{b}$, as required.

The objective function can then be expressed as:

$$\begin{aligned}
\|\mathcal{L}f' - \boldsymbol{b}\|_2^2 &= \sum_u \sum_\ell \left(\mathcal{L}_{u\cdot}f_{\cdot\ell} - \boldsymbol{b}_{u\ell}\right)^2 \\
&= \sum_u \sum_\ell \left(\sum_v \mathcal{L}_{uv}f_{v\ell} - b_{uv\ell}\right)^2 \\
&= \sum_u \sum_\ell \left(\sum_v (\mathbb{1}_{\{u=v\}} - \boldsymbol{T}_{uv})f_{v\ell} - b_{uv\ell}\right)^2 \\
&= \sum_u \sum_\ell \left(f_{u\ell} - \sum_v (w_{uv}f_{v\ell} + b_{uv\ell})\right)^2
\end{aligned}$$

where $w_{uv} = \boldsymbol{T}_{uv}$.

4 Details for the Illustrative Synthetic Experiment

Our hypothesis in this work is that infection dynamics are a good candidate for propagating label information over real networks. To illustrate this, we designed a synthetic experimental setup in which our goal was to capture the structure of real world networks. One well-known property of such networks is that they often have a community-like structure, with many intra-community edges, but few inter-community edges. In many cases, only a few specific nodes within a community are also connected to other communities. Hence, we randomly created small networks with the above properties.

Specifically, each network was set to have 3 communities, each with 64 nodes. Edges were randomly added between these nodes with probability 0.1. For community A, 8 nodes were assigned to community B, and an additional 8 to community C (and similarly for the other communities). These edges were also added with probability 0.1. To account for some noise, all other edges were added with probability 0.05. The seed set included one randomly chosen node from each community, giving $|S| = 3$. The figure in the main text displays a random instance of the above setting, providing both the instance specific accuracies, as well as the average accuracy over 1,000 random instances.

Recall that InfProp can be interpreted both as the expected result of a dynamic infection process, and as a stochastic ensemble of shortest paths. We therefore compared our method to two baselines. To compare the dynamics, we used Label Propagation (LabelProp) which is based on the more standard random-walk dynamics. As we argue in the text, these dynamics are prone to getting stuck in dense clusters. As can be seen, while InfProp provides almost exact predictions, the predictive values of LabelProp are almost uniform and hence extremely error-prone. This demonstrates the inability of label information to propagate efficiently over the network.

To demonstrate the power of using a stochastic ensemble of paths, we compared to simply setting labels according to the deterministic shortest paths given by the original graph. While correctly classifying most labels, shortest paths can be very sensitive to cross-community or noisy edges. In contrast, InfProp mitigates this noise by considering a distribution over shortest-paths.

5 Datasets

We evaluated our method on various learning tasks over three collections of benchmark datasets, which include network based datasets for multi-class learning with features[6] [39], multi-class learning without

[6] http://linqs.umiacs.umd.edu/projects//projects/lbc/

features[7] [38], and multi-label learning[8] [33]. The following table provides some statistics.

	Dataset	Nodes	Edges	Classes	Features	Avg. $\|y\|$
Multiclass[7]	CoRA	2,708	5,278	7	-	1
	DBLP	5,329	21,880	6	-	1
	Flickr	7,971	478,980	7	-	1
	IMDb	2,411	12,255	22	-	1
	Industry	2,189	11,666	12	-	1
Features[6]	CiteSeer	3,132	4,713	6	3,703	1
	CoRA	2,708	5,278	7	1,433	1
	PubMed	19,717	44,324	3	500	1
Multilabel[8]	Amazon	83,742	190,097	30	-	1.546
	CoRA	24,519	92,207	10	-	1.004
	IMDb	19,359	362,079	21	-	2.300
	PubMed	19,717	44,324	3	-	1
	Wikipedia	35,633	49,538	16	-	1.312
	YouTube	22,693	96,361	47	-	1.707

[7] http://cs.gmu.edu/ tsaha/Homepage/Projects.html
[8] http://github.com/sharadnandanwar/snbc