Semi-Supervised Learning with Competitive Infection Models: Supplementary Material

Author 1
Institution 1

Author 2
Institution 2

Author 3
Institution 3

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.[1] 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.

[1] 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:

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

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

Intriguingly, the authors of [7] 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 selected 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:

$$f = \mathbb{E}_{\delta \sim D}\left[\mathcal{A}^{\delta}\mathcal{S}\right] = \mathbb{E}_{\delta \sim D}\left[\mathcal{A}^{\delta}\right]\mathcal{S} \tag{2}$$

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:

$$\sigma_v(S) = 1 - f_{v0} = \sum_{\ell=1}^{L} f_{v\ell}, \qquad \sigma(S) = \sum_v \sigma_v(S) \tag{3}$$

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 [6], graph signals [2], and generalization error [5]. 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. **??** 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 [7] 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., [1, 4]), 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[2] [10], multi-class learning without fea-

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

tures[3] [9], and multi-label learning[4] [8]. The following table provides some statistics.

	Dataset	Nodes	Edges	Classes	Features	Avg. $\lvert y \rvert$
Multiclass[3]	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[2]	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[4]	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

References

[1] 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.

[2] 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.

[3] 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.

[4] Gomez-Rodriguez, M., Song, L., Du, N., Zha, H., and Schölkopf, B. Influence estimation and maximization in continuous-time diffusion networks. *ACM Trans. Inf. Syst. 34*, 2 (Feb. 2016), 9:1–9:33.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[3] http://cs.gmu.edu/~tsaha/Homepage/Projects.html

[4] http://github.com/sharadnandanwar/snbc