Active GNN reading notes

S2: An efficient graph based active learning algorithm with application to nonparametric classification

- Problem setup
  - active learning for binary label prediction on a graph
  - nonparametric active learning, S2 sequentially select vertices to be labeled
    - *cut-set*: $C = \{\{x, y\} \in E : f(x) \neq f(y)\}$
    - *boundary*: $\partial C = \{x \in V : \exists e \in C \text{ with } x \in e\}$ goal is to identify ∂C
    - algorithm assume a noiseless oracle that return label of a multiset of vertices, noisy oracle version algorithms can be transferred from noiseless
    - can be extended to multi-class
- Datasets
  - Digits:
    - Cedar Buffalo binary digits database
    - construct symmetrized 10-nearest-neighbor graph
  - Congressional Voting Records (CVR):
    - 380 vertices, boundary size of 234
  - Grid:
    - synthetic example of a 15x15, positive core in the center
- Methods
  - S2: Shortest Shortest Path

Algorithm 1 S^2: Shortest Shortest Path

Input Graph $G = (V, E)$, BUDGET $\leq n$

$L \leftarrow \emptyset$
while 1 **do**
  $x \leftarrow$ Randomly chosen unlabeled vertex
  do
    Add $(x, f(x))$ to L
    Remove from G all edges whose two ends have different labels.
    if $|L|$ = BUDGET **then**
      Return LABELCOMPLETION(G, L)
    end if
  while $x \leftarrow$ MSSP(G, L) exists
end while

    - LABELCOMPLETION: Any off-the-shelf graph prediction algorithms

        - **MSSP**: return midpoint on the shortest among all the shortest-paths that connect oppositely labeled vertices in L

Sub-routine 2 MSSP

Input Graph $G = (V, E)$, $L \subseteq V$

1: **for** each $v_i, v_j \in L$ such that $f(v_i) \neq f(v_j)$ **do**
2: $P_{ij} \leftarrow$ shortest path between v_i and v_j in G
3: $\ell_{ij} \leftarrow$ length of P_{ij} (∞ if no path exists)
4: **end for**
5: $(i^*, j^*) \leftarrow \arg\min_{v_i, v_j \in L: f(v_i) \neq f(v_j)} \ell_{ij}$
6: **if** (i^*, j^*) exists **then**
7: **Return** mid-point of $P_{i^* j^*}$ (break ties arbitrarily).
8: **else**
9: **Return** $\emptyset$
10: **end if**

    - Can be seen as: random sampling + aggressive search
        - aggressive search: like binary search to find the cut-edge, then unzip the cut-edge
- Baselines
    - measure query complexity
    - AFS On the complexity of finding an unknown cut via vertex queries
    - ZLG Combining active learning and semi- supervised learning using Gaussian fields and harmonic functions
    - BND Towards active learning on graphs: An error bound minimization approach

---

Active Learning for Networked Data

- Problem setup
    - classifying nodes (labels prediction)
        - node features
        - graph structure
        - features/labels of neighbor nodes
    - collective classification:
        - simultaneously predicting labels of all nodes
    - active learning
        - request labels, with goals of decreasing number of labels needed
        - pool-based setting:
            - initially provided with pool of unlabeled examples
            - each step select batch of instances, remove from pool, add to labeled corpus
        - task:
            - collective classification as base learner

        - train: active learning learn CC, CO
        - test: ICA + CC
- Methods
    1. cluster nodes based on graph structure: modularity clustering
    2. iterate:
        1. re-train CO, CC
        2. score clusters based on CO/CC disagreement, pick top k clusters
        3. label one of unlabeled node from each of the k clusters, remove them from pool
            - the node with greatest disagreement LD between CO, CC, majority is picked
        4. Semi-supervision and Dimensionality reduction
            1. semi-supervised collective classification method, use CO to predict unobserved neighbor
            2. PCA
    - note:
        - CC: $P(Y_i|X_i, \text{aggr}(N_i))$, consider neighbor labels
        - CO: $P(Y_i|X_i)$ local classifier with only node features
- Datasets
    - Cora & CiteSeer
        - citation network
        - ignore directions
        - cleaned up
- Baselines
    1. Semi-supervision and Dimensionality Reduction (Base Learner)
        1. CO
        2. CC
        3. CC+Semi-supervision
        4. CC+Semi-supervision+PCA
    2. ALFNET
        1. Random
        2. Uncertainty sampling
    3. Ablation
        1. disagreement: no cluster structure
        2. clustering: select cluster randomly

Active Discriminative Network Representation Learning

- Problem setup

    - a network representation learning method under active learning principle
    - pool-based active learning setting, select most informative instance given **query strategy**
    - Then label the selected node, add to labeled set for network representation learning
- Method
    - AL **Query Strategy**
        - uncertainty measure:
            - **Information Entropy**: prediction from node embedding
        - representativeness measure:
            - **Node centrality**: PageRank centrality
            - **Information density**: K-means on node embedding
    - Active Node Selection
        - Multi-Armed Bandit Method
        - Reward Scheme
    - Active Discriminative Network Representation
        - GCN
- Datasets
    - Citeseer, Cora, Pubmed
        - sparse BOW feature vector for each document
        - citation links
- Baselines
    - GCN: randomly select node
    - AGE: linearly combination of AL query strategy
    - ANRMAB-exclude:
        - ANRMAB-entropy
        - ANRMAB-centrality
        - ANRMAB- density
    - Metrics:
        - Macro-F1
        - Micro-F1

Towards Active Learning on Graphs: An Error Bound Minimization Approach Quanquan

- Problem setup:
    - adaptive method
    - non-adaptive method: this paper developed

        - present data-dependent generalization error bound for LLGC, using transductive Rademacher Complexity
        - actively select nodes by minimizing the empirical transductive Rademacher complexity of LLGC on a graph
        - present sequential optimization algorithm to select labeled data
        - use LLGC as classfier on labeled data
- Method:
    - LLGC:
        - graph-based(semi-supervised) learning method
        - add Graph Regrlarization
    - Objective function (for selection):
        - expected error on unlabeled <= empirical error on the labeled data + empirical transductive Rademacher complexity + confidence term
        - non-adaptive, do not know empirical error on the labeled data until label nodes
        - only minimize empirical transductive Rademacher complexity for LLGC
    - Sequential optimization
        - sequentially select nodes to be labeled is to optimize the above objective.
        - **Algorithm 1** Active Learning on Graphs via Generalization Error Bound Minimization (**Bound**)

          Input: Adjacency matrix $\mathbf{W}$, number of nodes to select l, regularization parameter μ;
          Compute $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$
          Perform eigen decomposition $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$
          Initialize $\mathbf{H}_0 = \mathbf{\Gamma}^{-1}$, $\mathcal{L}_0 = \emptyset$
          for $k = 0 \to l-1$ **do**
            Compute $i_{k+1} = \arg\max_{i \subset \mathcal{V}/\mathcal{L}_k} \frac{\mathbf{u}_i^T \mathbf{H}_k^{-1} \mathbf{\Lambda}^{-1} \mathbf{H}_k^{-1} \mathbf{u}_i}{1+\mathbf{u}_i^T \mathbf{H}_k^{-1} \mathbf{u}_i}$;
            Update $\mathcal{L}_{k+1} = \mathcal{L}_k \cup \{i_{k+1}\}$
            Update $\mathbf{H}_{k+1}^{-1} = \mathbf{H}_k^{-1} - \frac{\mathbf{H}_k^{-1} \mathbf{u}_{i_{k+1}} \mathbf{u}_{i_{k+1}}^T \mathbf{H}_k^{-1}}{1+\mathbf{u}_{i_{k+1}}^T \mathbf{H}_k^{-1} \mathbf{u}_{i_{k+1}}}$
          end for
    - Learning pipline:
        - select nodes by active learning
        - train classifier(LLGC) on the graph
- Datesets
    - Cora
        - 2 datasets from DS, PL

        - largest connected component is used
        - undirected
    - Coauthor
        - from DBLP
        - author as nodes, weighted edge by papers co-authored
- Baselines
    - active learning:
        - Random
        - Variance Minimization (VM)
        - METIS
        - Ψ Maximization (Ψ-Max)
        - BOUND: Proposed
    - classifier:
        - LLGC: Proposed
        - GFHF: used in VM
        - MinCut: used in Ψ-Max

Active Learning on Graphs via Spanning Trees

- Problem setup:
    - node classification, focus on prediction that rely on network topology only
    - homophily bias: linked entities tend to have same class-->clusters
- Method:
    - define:
        - cutsize $\Phi(y)$: number of edges connecting nodes with different labels
        - function $\Psi^*(L)$: small value correspond to good choice of traninig nodes L that satisfy homophily
        - Together bound from above the number of mistakes on non-training nodes
    - approach:
        - use SEL to minimize Ψ^* on a spanning tree of graph
        - label propagation on entire graph, yields a fast classifier that is better than random selection
- Datasets
    - RCV1 corpus
        - build weighted graph using k-NN
    - DBLP network
    - CORA network

- Baselines
  - Spanning trees generation:
    - RST(Random)
    - BFST(Breadth-first)
  - random selection
  - high-degree

Bayesian Semi-supervised Learning with Graph Gaussian Processes

- Problem setup:
  - graph-based semi-supervised learning: relational graph of data points is available
  - propose **Gaussian process model**, **scalable variational** inducing approximation method to perform **inference**
  - review:
    - Gaussian Process
      - **priors** on functions in Bayesian Inference
    - Scalable Variational Inference for GP
    - Graph Laplacian
      - $$\mathbf{L} = \mathbf{D} - \mathbf{A}$$
      - Graph Laplacian can be viewed as operator on space of functions $g : V \to \mathbb{R}$

$$\mathbf{L}(g) = \sum_{v \in Ne(n)} [g(n) - g(v)]$$

- Method
  - Graph Gaussian Process
  - Variational Inference with Inducing Points
  - Active learning:
    - GGP as classification model paired with Σ- optimal (SOPT) to form active learner, select + retrain classifier for 50 times
- Datasets
  - Cora, Citeseer, Pubmed
- Baselines
  - semi-supervised settings
    - GCN, MoNet, DCNN, DeepWalk, Planetoid, ICA, LP, SemiEmb, ManiReg
  - Active learning setting:

- SOPT+GCN/LP
- RAND+GGP/GCN/LP

Batch Mode Active Learning for Networked Data

- Problem setup
  - Batch mode active learning
    - query k labels of instances S to improve quality of learned classification model
    - define an objective function for active selection Q
    - design efficient algorithms to maximize $Q(S)$
  - selection combine both context and link information
  - present objective function based on maximum uncertainty, maximum impact, and minimum redundancy
  - Underlying model: Framework of Random Walk
- Method
  - Objective function $Q(S)$: combine maximum uncertainty and the maximum impact, naturally satisfy minimum redundancy
  - Use Framework of Random Walk as classification model
  - Combining **Link Information**: integrate link information into **similarity matrix**, extend similarity measure with **Pagerank**
  - Optimization: Greedy selection, monitonic submodular, have good error bound

ALGORITHM 1: Greedy algorithm for batch mode active selection. The algorithm iteratively selects a sample that can maximize $Q(S)$

Input: U, L, G, $\mathbf{x}$, $\mathbf{y}$, k
Output: S, s.t.$|S| = k$
Calculate transition matrix P;
Calculate probability vector $\mathbf{p}$;
for $v \in U$ **do**
  $H[v] \leftarrow p_v \log \frac{1}{p_v} + (1 - p_v) \log \frac{1}{1-p_v}$;
  initialize: $C[v] \leftarrow 0, \max[v] \leftarrow 0$;
end
initialize: $S \leftarrow \emptyset$;
while $|S| < k$ **do**
  for $v \in U - S$ **do**
    $C[v] \leftarrow$ The summation over $j \in U$;
      of $(H[j])^{\beta} \left(\max\{\max[j], w(v, j)\}^{1-\beta}\right)$;
  end
  Find $v \in U - S$ to maximize:
    $\alpha C[v] + (1 - \alpha) H[v]$;
  update: $S \leftarrow S \cup \{v\}$;
  update: $\max[j] \leftarrow \max\{\max[j], w(v, j)\}$;
end

    - Speed up with parallel algorithm
- Datasets
    - Gaussian Synthetic Dataset
        - random generated, 17 Gaussian distributions
        - no link information
        - use KNN to learn classification model
    - The Networked Synthetic Dataset
        - without content information
    - Real world with links:
        - Cora, Citeseer, WebKB
        - cast as multiple binary
    - Real World without Links:
        - UCI 20 Newsgroup
          classification tasks
- Baselines
    - Real world with links
        - Random
        - Most Uncertainty: largest entropy
        - Active Learning Using Gaussian Fields
        - Hybrid
        - k-means
    - Real world without links
        - Random
        - Most Uncertainty: largest entropy
        - Active Learning Using Gaussian Fields
        - SVM
        - Fisher
    - On Gaussian Synthetic Dataset:
        - Maximum Uncertainty
        - Without Maximum Uncertainty
        - Without Minimum Redundancy

FEW-SHOT LEARNING ON GRAPHS VIA SUPER- CLASSES BASED ON GRAPH SPECTRAL MEASURES

- Problem setup:
    - few-shot setting:

        - training: classifier must generalize well after seeing abundant base-class samples
        - testing: very few samples from novel class
    - Details
        - training: set of *base class* labeled graphs $G_B = \{(g_i^{(B)}, y_i^{(B)})\}_{i=1}^{n}$, set of *novel class* labeled graphs $G_N = \{(g_i^{(N)}, y_i^{(N)})\}_{i=1}^{m}$ used for fine-tuning, where $m << n$, set of labels from base class graphs and novel class graphs are disjoint.
        - testing: set of ublabeled unseen graphs $G_U = \{g_i^{(U)}\}_{i=1}^{t}$
- Method:
    - Computing super classes
        - Prototype Graphs:
            - partitioned G_B according to graph labels, $G_B = \cup_{i=1}^{K} G^{(i)}$
            - pick prototype graph p_i for the i-th class with least average spectral distance to rest graphs in the same class
        - Clustering prototype graphs:
            - k-means
            - into k super classes
    - GNN
        - GIN as graph feature extractor
    - Classifier
        - build super-graph:
            - node is graph feature vector
            - build on a batch of *base-labeled* graphs as a collection of k-NN graphs
            - each constituent k-NN graph is built on the graphs belonging to the same super-class
        - super graph passed through multi-layered graph attention network GAT
        - graph embedding passed into MLP to learn associated super-class labels
        - add cross-entropy losses associated with GAT and MLP
    - fine-tune:
        - fix GIN, MLP, tune GAT
- Datasets:
    - Letter-High, TRIANGLES, Reddit-12K, and ENZYMES.
- Baselines:
    - k-NN search on embeddings
        - Supervised:
            - GIN, CapsGNN, Diffpool
        - Unsupervised:
            - AWE, Graph2Vec, Weisfeiler- Lehman subtree Kernel, Graphlet count kernel
    - Few-shot:
        - replace GAT with GCN

- replace classifier with k-NN: GIN-k-NN
