Active GNN

- 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 vectices 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 transfered 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 $G = (V, E)$, $L \subseteq V$

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