ASHA
Date _____ Page ___

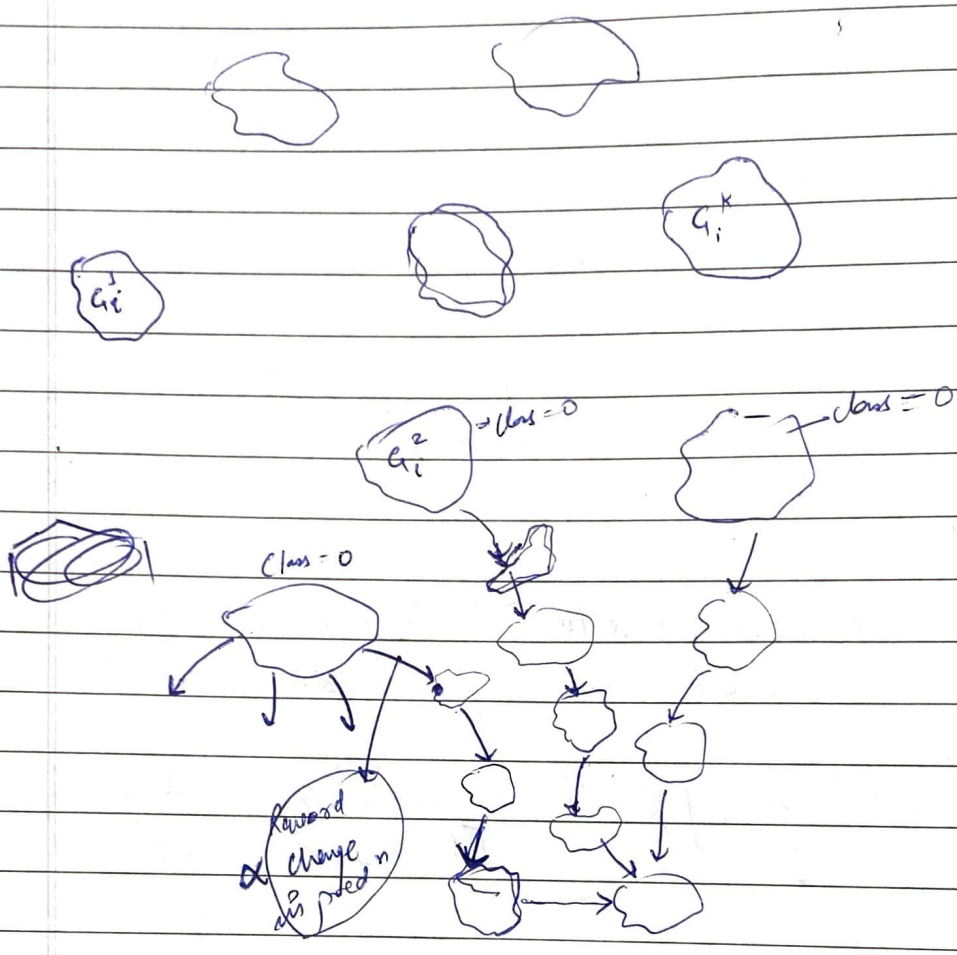
- IRC-Exp → sampling results ⇒
  Candidate
  C-L Pool size ⇒ 100 ⇒

New to avi gfx = 3469
Val, test = 433, 435 resp

- Feature size ⇒ Graph Embedding size ⇒ ~~20~~ [1 x 20]
  ↳ :- Last GCN Layer's output size = 20

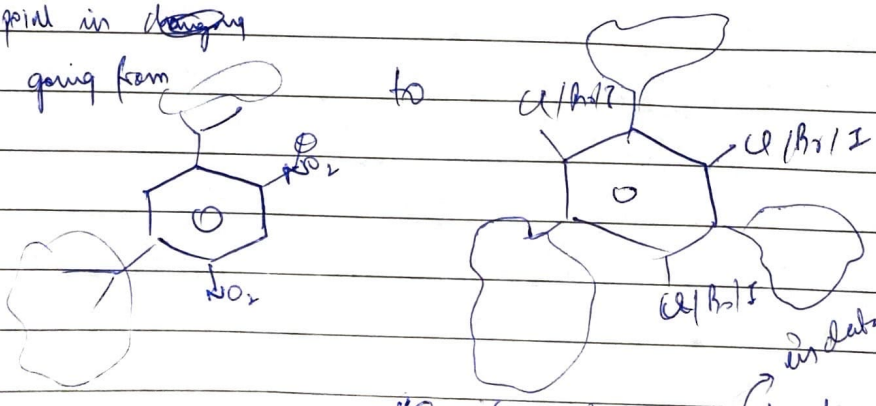
- Restricting the Random Walk to Data Domain Diss ⇒ Not going out of data manifold or expanding the data manifold too much
  ↳ In GCFE → they only go to a distance ~~θ~~ from any input graph
  ↳ fixed before hand (hyperparam)

⇒ simply perturbing the set of similar graphs (similar as per trained model, not necessarily based on similarity of just graph structure & features) till we reach a common cf ⇒ may (we check cf by the pred^n of trained model on proposed cf) ⇒ does not seem to be very appropriate in terms of explaining the model as it may not fully reflect the representations learnt by GNN that we may want to explain.
⇒ It would generate vCFs → but these may still not be CF explanations (in the true sense)
ie, → similar wrt classification



- How to make sure that the Random walk does not go to far from the data dist^n / manifold?

Ex → No point in changing going from to



If most graphs don't have a Cl/Br/I directly attached to the benzene in dataset

→ In check by computing GNN embedding of the new created graph → if the GNN embedding is outlier → we may not want to explore it further → But has expl^n vs exploitation tradeoff

↓

An intermediate graph may be too out of data domain but the final G may be in domain. ∴ ~~Ef needs~~ May keep this learnable by using RL (introducing a Reward setting) to automatically figure out what is good, what is not)

Here ① Approach 1 : ~~Re-compute clusters~~ of repr^n of input graphs, & check if the new graph belongs to any one cluster within a dist threshold

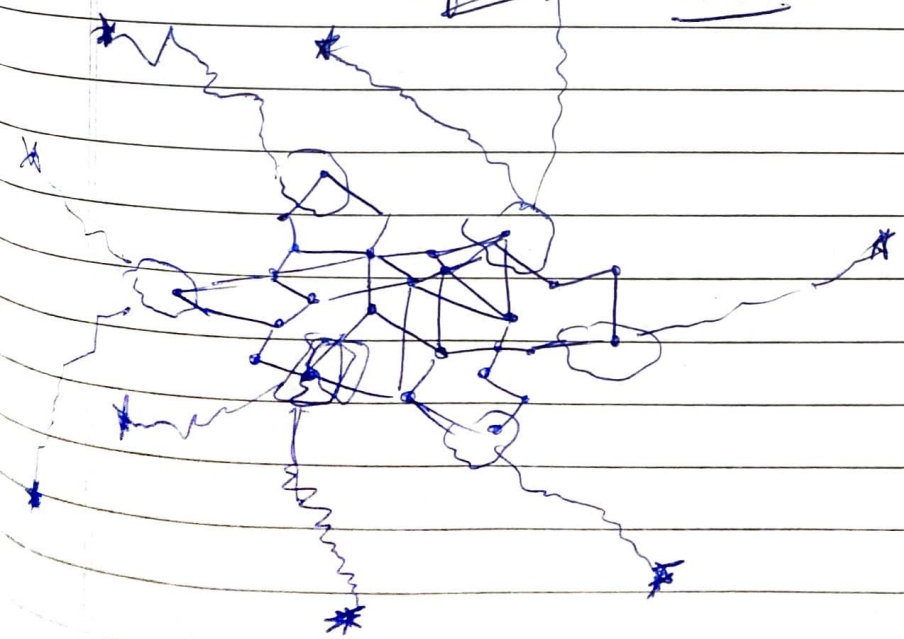
GNN embedding

→ ~~Can be~~ Feasible to compute

(Takes same time as checking if ~~a new~~ the new Graph is CF / not CF)

+ Distance from all clusters (centres) → O(k) ↳ # of clusters

② Approach 2 : After each new graph is formed → recluster embeddings of all the graphs till now → & then do Local Outlier Factor to check if it is outlier or not ? → Infeasible Approach (Not possible to recluster at each time step)




Class - 0
Class - 0
Class - 0 graphs
Class - 0
Class - 0
Class - 0
Class - 0
Class - 1
Pot CF2
Pot CF3
Pot CF4
Pot CF5
Pot CF1

→ Consider the convex hull of all embeddings (both class 0 & class 1) → Can be pre-computed

Then, a new graph can be checked to be an outlier if it is more than θ dist from convex hull

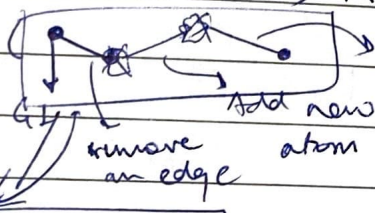
(Ideally, it's embedding should lie within the convex hull) → Complexity of this computation?

⇒ Penalise → Such potential CFs which are at high distance from this convex hull ⇒

(Also → read about some K-nearest nbrs like approaches for finding whether a new point is within data-manifold)

* Also → sometimes can't just remove an edge (for eg in molecular graphs → if removing an edge → will need to add some other edge to satisfy the valence)

↳ How to handle this? ⇒ Need multiple steps in one-go



Need 3 steps in 1 go

→ Can select among these pot CFs & arrange them in ↑ order of their ~~distance to~~ avg distance to all input graphs to be explained.

Can tackle this using immutable features ⇒ If a transition leads to a change in an immutable feature → keep making sequential changes till that immutable feature change is restored.

↳ If we simply don't allow such intermediate transitions we won't be able to create new graphs ⇒

As we need to remove an edge first at any cost for an edge removal action.

Reward for transition to a new graph ⇒ ~~Predictions~~

$\propto \uparrow$ in Prob^n for desired label / Decrease in prob^n for current (undesired label)

$$\propto \frac{1}{\text{Distance from convex hull of embeddings (data manifold)}} \text{ of its embedding}$$

(maybe) $\propto \frac{1}{\text{Avg dist from all graphs to be explained}}$

how to ~~quantify this~~ compute this efficiently

★ Still all this is not using the Representations learnt by the GNN or its learnt decision boundaries

~~Reward~~

$$R_{t+1} = e^{-\left[P(\hat{y} = y_{desired}) - P(\hat{y} = y_{current/new})\right]}$$

$e^{d_1} \cdot e^{d_2}$ $(d_1+\varepsilon)$ $(d_2+\varepsilon)$ $(d_3+\varepsilon)$

d_1 = distance from boundary of convex hull of all embeddings

d_2 → avg distance of G_{t+1} from all graphs to be explained

G_t —Action→ G_{t+1}

- remove edge
- add edge
- change edge type
- change node label
- change node / edge feats

We can have an estimate of avg d_2 using the dist of GNN Embedding of G_{t+1} to all graphs to be explained.

Also $\propto$ Data-density at the new pt G_{t+1}

(How to estimate that → one proxy could be

$$\frac{1}{\text{Sum of dist of K-nearest nbrs}}$$

d_3 = ~~Sum~~ distance of K-nearest nbrs (ie Actual graphs in the dataset → not the generated graphs)

↳ But this will take $O(N \log N + N)$ complexity where N = no of ~~points~~ trans samples → for each transition → Can't ~~go this~~ adopt this approach.

$$R_{t+1} = \frac{e^{[p(\hat{y} = y_{desired}) - p(\hat{y} = y_{current/non-desired}]}}{e^{d_1 (d_2 + \epsilon)(d_3 + \epsilon)}}$$

d_1 ⇒ distance from boundary of convex hull → $\mathbb{1}_{d_1}$ { ≥ 0 if pt outside C·H ; 0 o.w

d_2 ⇒ avg dist of embedding of G_{t+1} from all graphs to be explained

d_3 ⇒ Proxy for data density at G_{t+1} ⇒ sum of dist of K nearest nbrs.