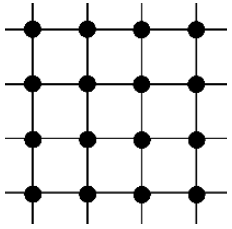
Graph Embedding with Adapative-curvature

Project Proposal

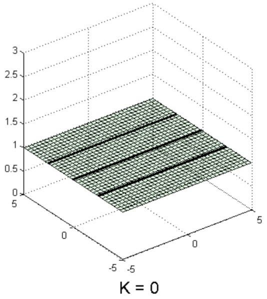
11/04/2022

Jialin Chen

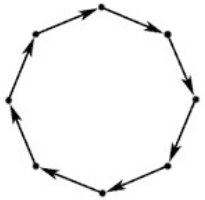
Motivation



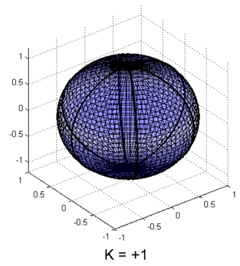
Embed



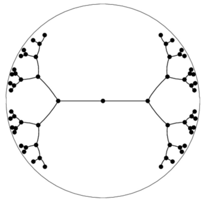
0 curvature



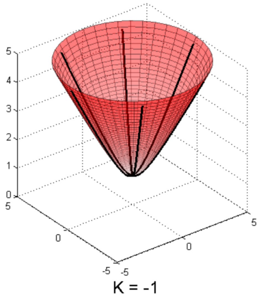
Embed



+ curvature

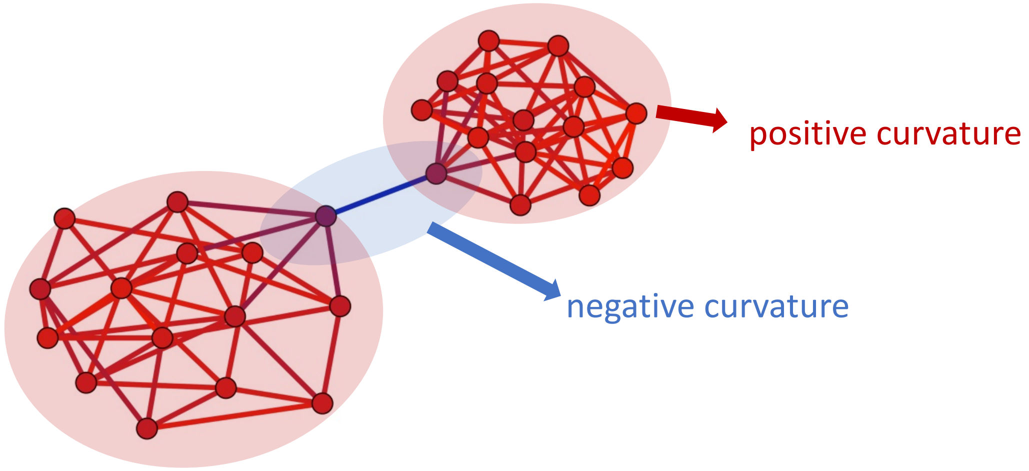


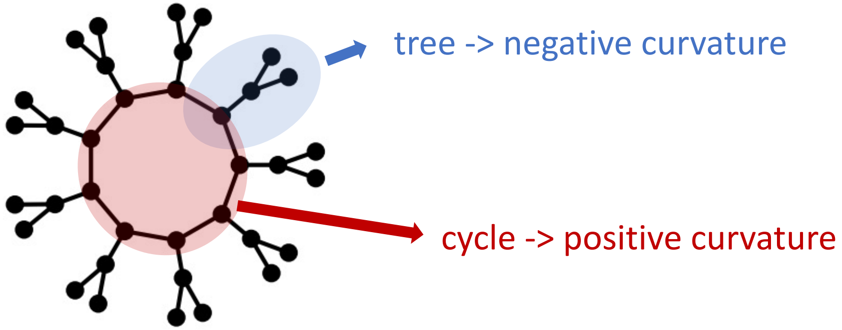
Embed



- curvature

What about graphs with local structures of different curvatures?





Curvature on Graph

- **Forman curvature** $F(i,j)$ for $e = (i,j) \in E$

$$F(i,j) = w_e\left(\frac{w_i}{w_e} + \frac{w_j}{w_e} - \sum_{e_i \sim e, e_j \sim e} \left[\frac{w_i}{\sqrt{w_e w_{e_i}}} + \frac{w_j}{\sqrt{w_e w_{e_j}}}\right]\right.$$

where w_e, w_i denote the weight on edge e and node i.

- For the case that all $w_e = w_i = 1$:

$$F(i,j) = 4 - \deg(i) - \deg(j)$$

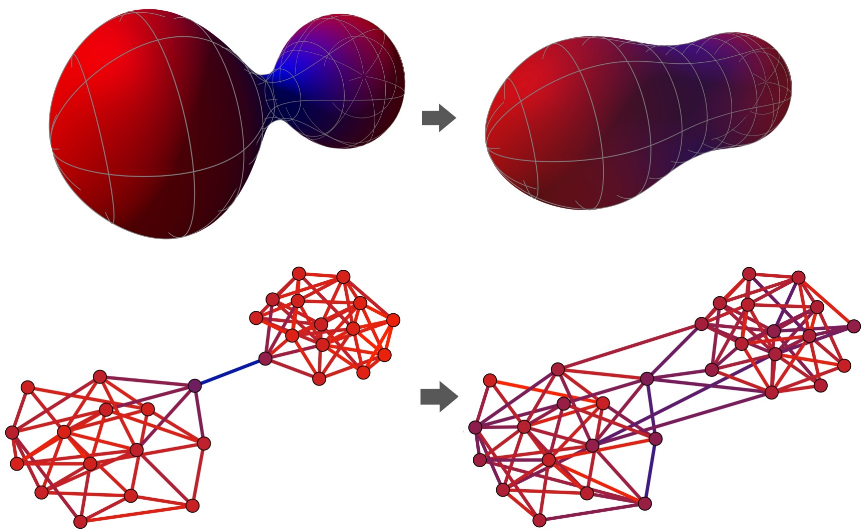
- **Augmented Forman Curvature:**

$$F'(i,j) = 4 - \deg(i) - \deg(j) + 3\gamma\#_\Delta(i,j)$$

  - γ considers the contribution of triangles
  - $\#_\Delta(i,j)$ denotes the number of triangles based at $i \sim j$
  - Consider at most 2-hop information

Curvature on Graph

- Two Contributions of [Understanding Oversquashing paper:]
  - Propose a new edge-based **Balanced Forman Curvature for graph**
  - Discover the relation between **the proposed curvature** and **local structure** of the graph (e.g., Cheeger constant)



weaker connection between large communities
=> **more negative** curvature!

Graph G		Ric_G
Cycles	C_3	$\frac{3}{2}$
	C_4	1
	$C_{n\geq 5}$	0
Complete K_n		$\frac{n}{n-1}$
Grid G_n		0
Tree T_r		$\frac{4}{r+1} - 2$

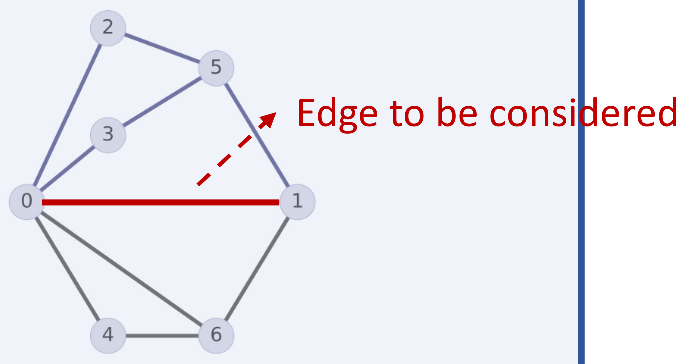
Intuitive understanding of
the proposed curvature

Balanced Forman Curvature for graph

- $\#_{\Delta}(\boldsymbol{i},\boldsymbol{j}) = S_1(i) \cap S_1(j)$ are the triangles based at $i\sim j$
- $\#_{\blacksquare}^{i}(\boldsymbol{i},\boldsymbol{j}) = \{k \in S_1(i) \backslash S_1(j), k \neq j \colon \exists w \in (S_1(k) \cap S_1(j)) \backslash S_1(i)\}$ are the neighbors of i *forming a* 4-cycle based at the edge $i\sim j$ **without diagonals inside.**
- $\boldsymbol{\gamma_{max}}(\boldsymbol{i},\boldsymbol{j})$ is the maximal number of 4-cycles based at $i\sim j$ traversing a common node
- $\boldsymbol{S_r}(\boldsymbol{i}) = \{j \in V \colon d_G(i,j) = r\}$
- $\#_{\Delta}(\boldsymbol{i},\boldsymbol{j})$ **is related to positive curvature;** $\#_{\blacksquare}^{i}$ **is related to negative curvature**

Example:

- $\#_{\Delta}(0,1)$=1
- $\#_{\blacksquare}^{0}(0,1) = \{2,3\}$; $\#_{\blacksquare}^{1}(0,1) = \{5\}$
- $\gamma_{max}(0,1) = 2$, as there exist two 4-cycles passing through node 5



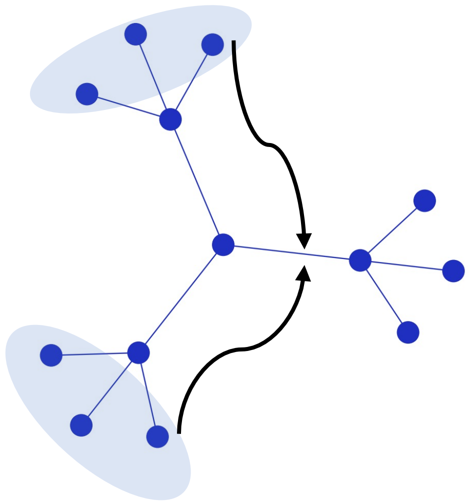
Balanced Forman Curvature for graph

$$Ric(i,j) = \frac{2}{d_i} + \frac{2}{d_j} - 2 + 2\frac{|\#_{\Delta(i,j)}|}{\max\{d_i,d_j\}} + \frac{|\#_{\Delta(i,j)}|}{\min\{d_i,d_j\}} + \frac{\gamma_{max}^{-1}}{\max\{d_i,d_j\}}(\#_{\blacksquare}^{i}(i,j) + \#_{\blacksquare}^{j}(i,j))$$

Think

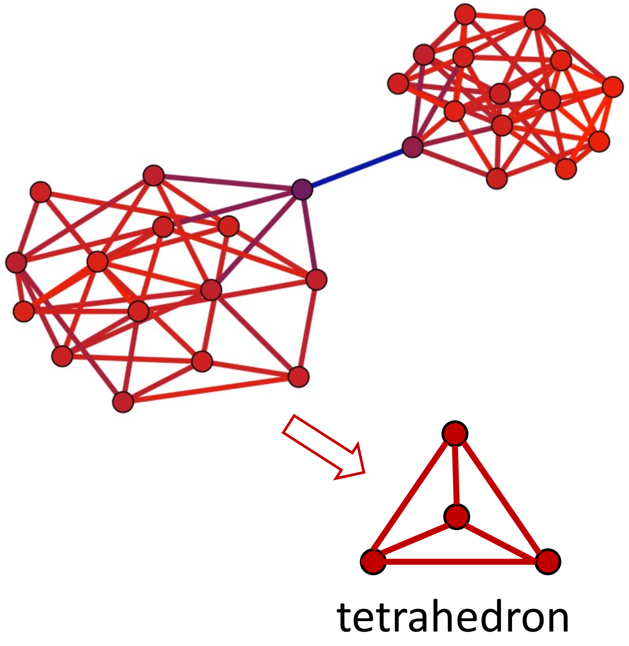
Cons:

1. Does not account for tree-like structures explicitly
2. Consider only (at most) 3-hop information



Improvements:

1. design a tree indicator (Gromov's δ-hyperbolicity /hierarchy of the nodes)
2. involve multi-hop information into the curvature
3. consider higher-dimensional structures (simplicial complex, polyhedron, etc.)



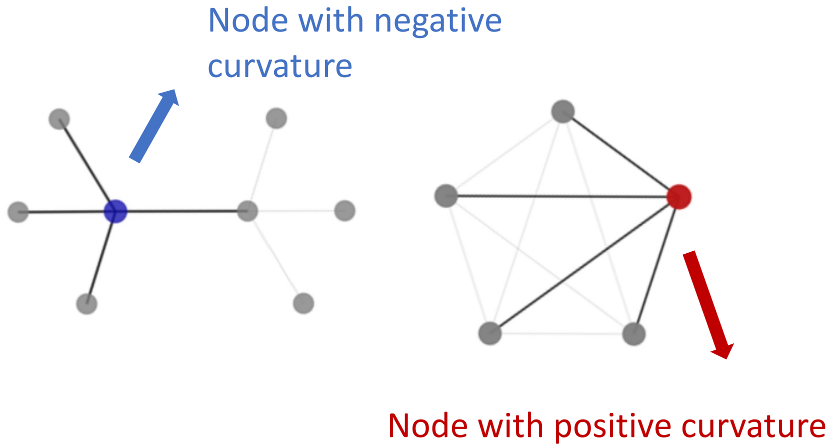
Node-based Curvature

- Graph embedding aims to map each node in the graph to an vector in the embedding space
- Need to define **node-based curvature**
- Node-based Forman scalar curvature:

$$F(i) = \frac{1}{d_i} \sum_{e=(i,j)\in E} F(i,j)$$

- Node-based Balanced Forman scalar curvature:

$$Ric(i) = \frac{1}{d_i} \sum_{e=(i,j)\in E} Ric(i,j)$$



Cons:

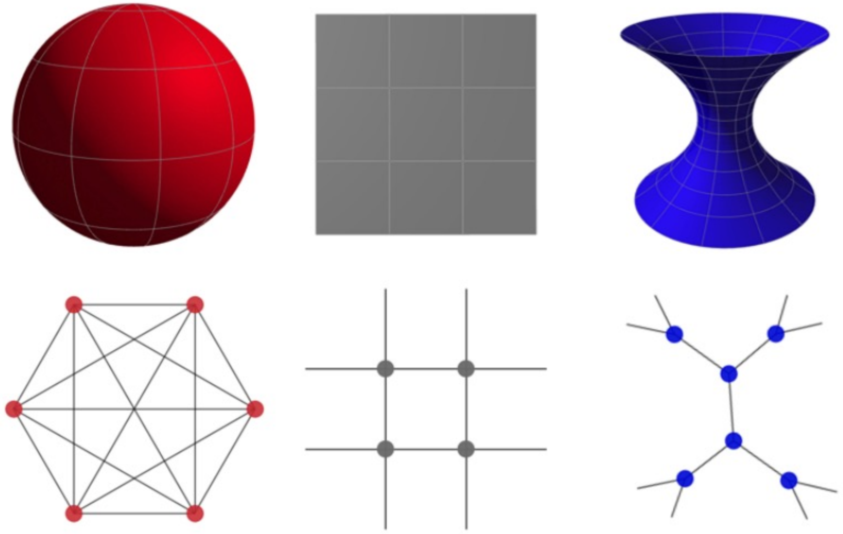
Positive curvature and negative curvature may cancel each other

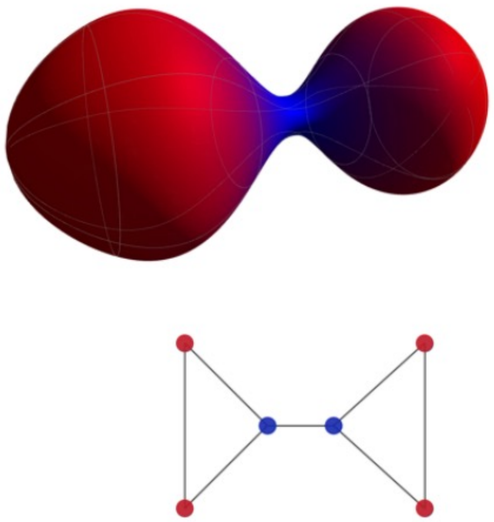
Nodes with the same curvature may have different local structure

Our Goal

- Choose **an appropriate curvature definition** that can reflect local structures of interest (or propose a new curvature!)
    - Tree/cycle/grid structure; community connectivity; multi-hop structure, etc.
- Give a manifold with **non-constant curvature** that is suitable for graphs with different local structures
- **Train a model** to embed the graph (map each node to a vector on the manifold)
    - Loss function design: embedding distortion / curvature matching
    - Metric: mean average precision / down-stream task performance

Homogenous v.s. Heterogenous





Homogenous manifold
(Sphere, Euclidean space, Hyperboloid)
Curvature is independent of the point

Heterogenous manifold
Non-constant curvature
Suitable for node-wise embedding

Hyperboloid: $\mathbb{H}^{d,K} = \left\{ \boldsymbol{x} \in \mathbb{R}^{d+1} \colon \langle \boldsymbol{x}, \boldsymbol{x} \rangle_H = \frac{1}{K} \right\},\ K < 0$ is the negative curvature

Sphere: $\mathbb{S}^{d,K} = \left\{ \boldsymbol{x} \in \mathbb{R}^{d+1} \colon \langle \boldsymbol{x}, \boldsymbol{x} \rangle_S = \frac{1}{K} \right\},\ K > 0$ is the curvature

Previous Method

- **Product Manifold**

  Given two Riemannian manifolds (M_1, g_1) and (M_2, g_2), their Cartesian product $M = M_1 \otimes M_2$ is also a Riemannian manifold with $g = g_1 \otimes g_2$.

  The scalar curvature of the product manifold:

$$R_g(p_1, p_2) = R_{g_1}(p_1, p_2) + R_{g_2}(p_1, p_2)$$

- **To enable changeable curvature:**

  Let $M = M_h \otimes M_\varphi$

  M_φ is a **rotatially symmetric manifold** in $\mathbb{R}^3$.

  The curvature of M_φ: $R_\varphi(r)$ depends on the radial distance r and (pre-defined) **radial function $\boldsymbol{\varphi(r)}$**.

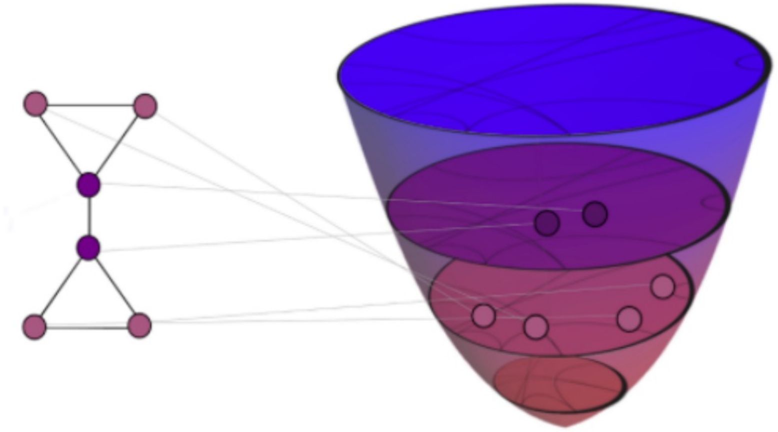
  M_h is a homogenous manifold (constant curvature)

Previous Method

- For $x \in G \rightarrow f(x) = (z(x), r(x), \theta)$

$$R_M(z, r, \theta) = \underbrace{R_h}_{\text{constant}} + \underbrace{R_\varphi(r)}_{\text{changeable}}$$

- **Con:** $M = M_h \otimes M_\varphi$, where the homogenous manifold M_h can only represent single type of structure



Di Giovanni, Francesco, Giulia Luise, and Michael Bronstein. "Heterogeneous manifolds for curvature-aware graph embedding." *arXiv preprint arXiv:2202.01185* (2022).

Ours: Fusion Manifold

- To involve hyperbolic and spherical space:
  - $M = \lambda_1(M_h \otimes M_{\varphi_1}) \otimes \lambda_2(M_s \otimes M_{\varphi_2})$ with $g = \lambda_1(g_1 \otimes g_{\varphi_1}) \otimes \lambda_2(g_2 \otimes g_{\varphi_2})$
- **Fused curvature:**

$$R_M(z_1, r_1, \theta_1, z_2, r_2, \theta_2) := \frac{1}{\lambda_1} R_h + \frac{1}{\lambda_2} R_s + \frac{1}{\lambda_3} R_\varphi(r)$$

  - where $z_1 \in M_h$, $z_2 \in M_s$, θ_i is unrelated to the curvature
  - Euclidean information is incoporated in M_{φ_i}
  - λ_1 , λ_2, λ_3 are learnable
  - z_1: hyperbolic component of node embedding
  - z_2: spherical component of node embedding

HGNN Algorithm

- **Input Transformation:** $\boldsymbol{x}^{0,M} := \exp_o^{R_0(x)}((0, \boldsymbol{x}^{0,E}))$
- **Message**: $\boldsymbol{h}_i^{l,M} = \left(W^l \otimes^{R_{l-1}(x_i)} \boldsymbol{x}_i^{l-1,M}\right) \oplus^{R_{l-1}(x_i)} \boldsymbol{b}^l$
  - Hyperbolic linear: $W \otimes^{R(x)} \boldsymbol{x}^{l,M} := \exp_o^{R(x)}(W \log_o^{R(x)}(\boldsymbol{x}^{l,M}))$
  - Mobius addition: $\boldsymbol{x}^M \oplus^{R(x)} \boldsymbol{b} := \exp_{x^M}^{R(x)}(P_{o\to x^H}^{R(x)}(\boldsymbol{b}))$

Edge curvature

- **Aggregation**: $AGG(x^M)_i := \exp_{\boldsymbol{x}_i^M}^{R_{l-1}(x_i)}(\sum_{j\in\mathcal{N}(i)} \boxed{w_{ij}} \log_{\boldsymbol{x}_j^M}^{R_{l-1}(x_j)}(\boldsymbol{x}_j^M))$
- **Update**: $\mathrm{Update}^{R_{l-1},R_l}(\boldsymbol{x}^M) := \exp_{\boldsymbol{o}}^{R_l(x)}(\sigma\left(\log_{\boldsymbol{o}}^{R_{l-1}(x)}(\boldsymbol{x}^M))\right)$

Improvement:

- Incorporate multi-hop / higher-dimension message passing
- be consistent with curvature definition

Loss

- Loss consists of two components
  - Average Distance Distortion

$$L_d(f) = \sum_{i,j} \left| \frac{d_E^2\left(f(x_i), f(x_j)\right)}{d_G^2(x_i, x_j)} - 1 \right|$$

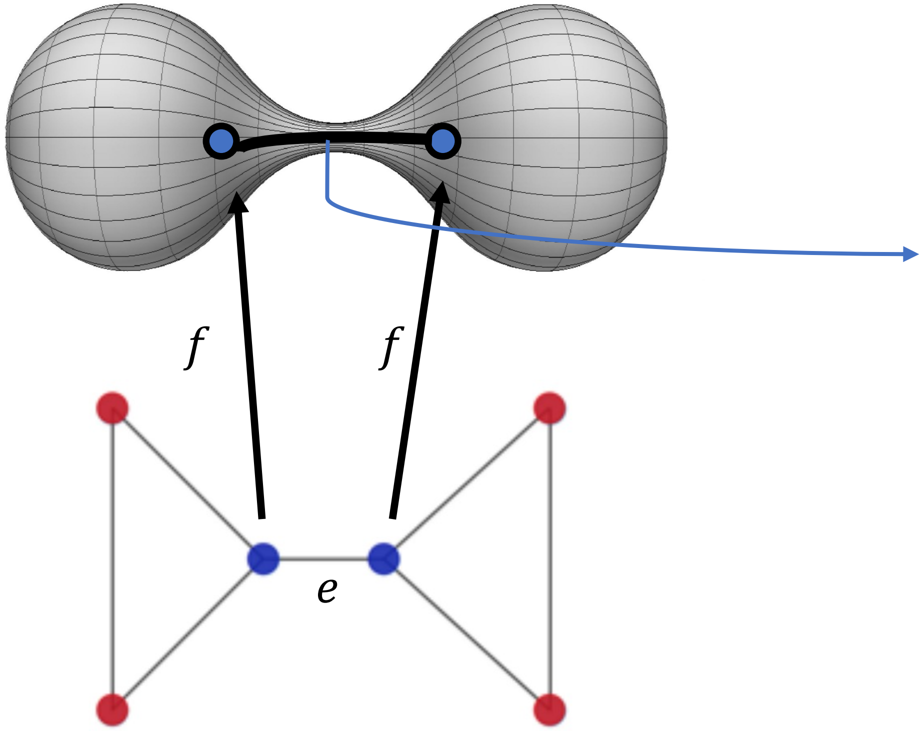
  - Node-wise Curvature Matching

$$L_{cn}(f) = \sum_i \frac{\left(F(x_i) - R\left(f(x_i)\right)\right)^2}{(|F(x_i)| + \varepsilon)^2}$$

  - E denotes the embedding space, f denotes the embedding function
  - F is the node-wise curvature on graph, R is the curvature on the manifold
- However, nodes with the same curvature may have different geometric structures (e.g., connects the edges with different curvatures)

Improvement

- Preserve the curvature along the geodesic between two embeded points on the manifold



Geodesic l corresponds to edge e

If e has negative curvature, then the model should make sure the curvature along l is always negative

Edge-wise Curvature Matching:

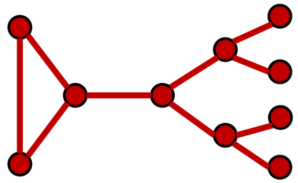
$$L_{Ce}(f) = \sum_{(i,j)\in E} F(i,j) - \lambda \frac{\int_0^l R(t)dt}{gl(f(x_i), f(x_j))}$$

$gl(\cdot,\cdot)$ is the geodesic length function. With explicit formula of the embedding space,

$\frac{\int_0^l R(t)dt}{gl(f(x_i), f(x_j))} := h(x_i, x_j)$ will be easy to calculate

Evaluation

- Graph Reconstruction:
  - Average distance distortion: $\sum_{i,j} |\frac{d_E^2\left(f(x_i),f(x_j)\right)}{d_G^2\left(x_i,x_j\right)} - 1|$
  - Synthesic: Ring of trees / SBM / Erdös-Rényi / Barabási-Albert
  - Real-world datasets: Cities / CS PhDs / Power / WebEdu /Facebook
- Neighbor Information Searching
  - Curvature on the embedding space helps to discover the **neighbor structure** of the node (linked triangles, hierarchy level, etc.)



- Down-stream Task performance
  - node classification $P(i = y|\boldsymbol{f(x_i)}, \boldsymbol{R(f(x_i))})$
  - link prediction $P((i,j) \in E|\boldsymbol{f(x_i)}, \boldsymbol{f(x_j)}, \boldsymbol{R(f(x_i))}, \boldsymbol{R(f(x_j)}, \int_{\boldsymbol{f(x_i)}}^{\boldsymbol{f(x_j)}} \boldsymbol{R(t)dt})$