Ricci curvature on Graphs

Arghya Bhattacharya[1] Chinmaya Dehury Aayush Sureka

Theory

We will first present the theoretical framework. Subsequently we will discuss about our implementation and design details and then the experimentation and its applications.

Lemma 1 : A temporally changing network can be thought of a discrete Riemannian manifold, equipped with a probability measure, dynamically changing w.r.t. time.

Definition 2 : Curvature of a geometric surface is defined as the deviation from being flat in this context whereas, intrinsic curvature, is defined at each point on a Riemannian manifold, independent on the embedding, i.e. how the manifold is realized in an ambient euclidean space.
For instance, for a surface M in R^3, a point $p : p \in M$, the plane containing the tangent vector T and surface normal N, cut out a curve of surface M through p. The normal curvature of M is defined to be the curvature of this curve. The principal curvature k_1 and k_2 are defined to be the maximum and minimum of the normal curvatures. The Gaussian curvature is defined to be $k_1 k_2$. It is positive for spheres, negative for one-sheet hyperboloid and zero for planes. It also has the character of being locally saddle (-ve) or locally convex (+ve).

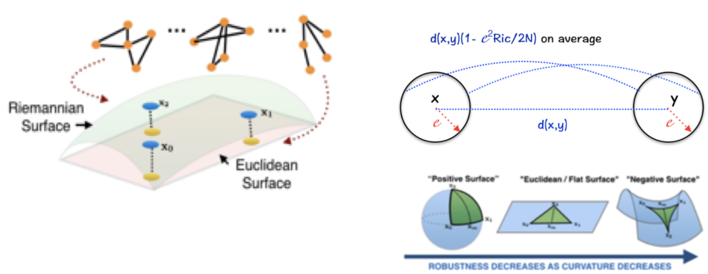

Figure 1: [Left] Visualisation of the manifold; [Top Right] Definition of curvature; [Bottom Right] Positive, zero and negative curvature

Definition 3 : Let M be a n-dimensional Riemannian manifold, $x \varepsilon M$ and $T_x M$ is the tangent space at x and $u_1, u_2 \varepsilon T_x M$ are orthonormal vectors. For geodesics $\gamma_i = exp(t\mu_i) : i = 1, 2$, sectional curvature, deviation of geodesics w.r.t. euclidean geometry, $K(u_1, u_2)$ is defined as

$$d(\gamma_1(t), \gamma_2(t)) = \sqrt(2)t(1 - \frac{K(u_1, u_2)}{12}t^2 + O(t^4))$$

[1]Algorithms Lab, Department of Computer Science, Stony Brook University, Stony Brook, NY, 11794-2424 USA. Email: `{argbhattacha}@cs.stonybrook.edu`.

The Ricci-curvature is defined to be the average sectional curvature. For a given unit vector $u\varepsilon T_x M$,

$$Ric(u) = \frac{1}{n-1}\sum_{2}^{n} K(u, u_i)$$

Corollary 4 : As we mentioned, in the previous formal definition, if point x and y are on surface M, and v is a tangent vector from x to y, let's take another tangent vector w_x and transport it along v till y, to become a tangent vector w_y. if x' and y' are the other end points of w_x and w_y,

$$\text{the surface is flat} \iff xx'y'y \text{ is a triangle}$$

Otherwise, $x'y' \neq |\overline{v}|$, and this difference is used to define sectional curvature. Ricci curvature is the average sectional curvature over all directions of w, it only depends on v.
Definition 5 : Monge's Problem defined in 1781, where it is defined as: some mass has to be transferred from one space X with Borel probability measure μ to another space Y equipped with ν, where the cost of transporting a unit mass from x to y is $c(x, y)$ where $c : X \times Y \to R$, where $R \geq 0$ assuming the cost of transportation per unit distance is constant. The solution should look for an optimal transport plan $T : X \to Y$ with $inf \int_X c(x, T(x))d\mu(x)$ | T : transportation

Definition 6 : Let X be a metric measure space equipped with distance d where μ_1 and μ_2 are two measures of same mass. Now, $\mu : \mu_1 \to \mu_2$ is a $X \times X$ coupling or a mass preserving transportation plan, $\mu\varepsilon\Pi(\mu_1, \mu_2)$ such that,

$$\int_y d\mu(x, y) = d\mu_1(x)$$

$$\int_x d\mu(x, y) = d\mu_2(y)$$

Wasserstein/ Earth Mover/ Kantorovich-Rubenstein distance between x and y is defined as

$$W_p(\mu_1, \mu_2) = [\inf_{\mu\varepsilon\Pi(\mu_1,\mu_2)} \int\int d(x, y)^p d\mu(x, y)]^{1/p}$$

Definition 7 : According to analytical definition of convexity, if a function $f : R^n \to R$ is C^2, the convexity can be characterized by $\nabla^2 f(x) \geq 0$. On the other hand, according to synthetic definition of convexity, f is convex if $f((1-t)x + ty) \leq (1-t)f(x) + tf(y)\ \forall x, y\varepsilon R^n$ and $t\varepsilon[0, 1]$

Lemma 8 : In a Riemannian manifold setting M, if P(M) is a probability measure equipped with a Wasserstein-2 distance metric, and μ_t is the geodesic curve on P(M),

$$Ric \geq 0 \iff t \to \int \mu_t \log \mu_t \text{ curve is convex in t}$$

Hence, the lower bound of Ricci curvature is related to entropy.

Definition 9 : If (X, d, m) is a geodesic space, a probability measure P can be set as,

$$P(X, d, m) = \mu \geq 0 : \int_x \mu dm = 1$$

$$P^*(X, d, m) = \mu\varepsilon P(X, d, m) : \overline{\lim}_{\varepsilon\to 0^+} \int_{\mu\geq\varepsilon} \mu \log \mu dm < \infty$$

Hence it can be defined,

$$H(\mu) = \overline{\lim}_{\varepsilon \to 0^+} \int_{\mu \geq \varepsilon} \mu \log \mu dm \, , \forall \mu \varepsilon P^*(X, d, m)$$

It can be noted, that Boltzmann's Entropy $S(\mu) = -H(\mu)$ and hence, if $H(\mu)$ is convex, $S(\mu)$ is concave.

Theorem 10 : If for the geodesic space, X's ricci curvature is bounded from below by $k \, \forall \mu_0, \mu_1 \, \varepsilon P(X) \, \exists$ a constant speed geodesic μ_t w.r.t. Wasserstein metric connecting μ_0 and μ_1 such that,

$$S(\mu_t) \geq tS(\mu_0) + (1-t)S(\mu_1) + \frac{kt(1-t)}{2} W(\mu_0, \mu_1)^2 : 0 < t < 1$$

$$\implies \Delta S \times \Delta Ric \geq 0$$

Theorem 11 : The fluctuation theorem for networks consider random fluctuation of a given network that result in perturbation of some observed variable. If $p_\varepsilon(t)$ is the probability that the mean of the observed variable deviates more than ε from the unperturbed value at time t. Hence, $p_\varepsilon(t) \to 0$ at $t \to \infty$. The relative rate function is defined as $R = \lim_{t \to \infty}(-\frac{1}{t} \log p_\varepsilon(t))$. It is worth noting that R is negatively correlated with the amount of deviation. And,

$$\Delta S \times \Delta R \geq 0$$

Corollary 12 : From Theorem 6 and Theorem 7, it can be safely extended to claim $\Delta R \times \Delta Ric \geq 0$.

Definition 13 : Let's say (X, d) is a metric space equipped with a probability measure $\mu_x : x \in X$. Ollivier-Ricci curvature $\kappa(x, y)$, in a discrete setting such as a graph, along the geodesic connecting x and y is defined as

$$W_1(\mu_x, \mu_y) = (1 - \kappa(x, y))d(x, y)$$

where d is geodesic distance on the graph and it is measured as $d_x = \sum_y w_{xy}$ and w_{xy} is the weight of the edge connecting x and y, (it is zero when there is no such edge), hence one step random walk from x in the graph is measured as $\mu_x(y) = \frac{w_{xy}}{d_x}$.

Lemma 14 : As have been suggested by past researchers, in a discrete graphical setting G, for a vertex $X \in G$ with degree of freedom $k = |\Gamma(x)|$, let the neighbors be $\Gamma(x) = x_1, x_2, ...x_k$. The probability measure μ_x is defined as.

$$\begin{aligned} \mu_x^\alpha &= \alpha : x_i = x \\ &= (1-\alpha)/k : x_i \in \Gamma(x) \\ &= 0 : otherwise \end{aligned}$$

Lemma 15 : With linear programming (LP) model, the Wasserstein distance can be formulated between two nodes in a graph, only with the local knowledge. The LP model for $W(\mu_x^\alpha, \mu_y^\alpha)$ is represented as a $m \times n$ matrix, where $\rho_{ij} \geq 0$ refers to the mass transported from vertex x_i to y_j.

We try to find:

$$W(\mu_x^\alpha, \mu_y^\alpha) = inf[\sum_j \sum_i d(x_i, y_j)\rho_{ij}\mu_x^\alpha(x_i)] : \sum_j \rho_{ij} = 1 \, \forall i, \rho_{ij} \varepsilon [0, 1] \, \forall i, j,$$

$$\sum \rho_{ij}\mu_x^\alpha(x_i) = \mu_y^\alpha(y_j) \, \forall j$$

Definition 16 : The discrete Ricci flow on a network is defined as an evolving process, where in each iteration, we update the edge weights by the following formula:

$$w_{xy}^{i+1} = d^i(x,y) - k_{xy}^i \times d^i(x,y)$$

Definition 17 : The scalar curvature in general is defined as $S(x) = \sum_y \kappa(x,y)$. So, in discrete setting it can be defined as $\overline{S}(x) = \sum_y \kappa(x,y)\mu_x(y)$. We keep on repeating this until $|k_{xy}^i - k_{xy}^{i-1}| < \varepsilon$

Definition 18 : Entropy in a discrete setting is defined as $\overline{S}_x = -\frac{1}{\log \overline{d}_x} \sum_y \mu_x(y) \log \mu_x(y)$

Implementation:

The algorithm to get Ricci curvature for the edges of a weighted undirected graph can be presented as follows. The complete implementation is available at `https://github.com/srkaysh/Graph-Ricci-Curvature.git`.

1. we take an adjacency matrix as an input
2. V = number of vertices, we ensure that number of edges, $E \geq 0.8\binom{V}{2}$, so that the graph is connected
3. we get the all pairs shortest path using Dijkstra's algorithm
4. for each pair of adjacent nodes we do the following:

   we calculate the neighbors of the source node and target node and keep them in two arrays, $source - neighbors$ and $target - neighbors$, let's say their size is m and n.

   $d_x = \sum$ weight(source-neighbors) and $d_y = \sum$ weight(target-neighbors)

   $\mu_x = \frac{source-neighbors}{d_x}$ and $\mu_y = \frac{target-neighbors}{d_y}$

   we define $d_{m\times n}$, such that d_{ij} = weight-shortest-path(node i, node j)

   we define ρ, a transportation plan, of the size $n \times m$.

   objective: $min(\mu_x d_{m\times n} \rho_{n\times m})$

   constraints: $\rho_{n\times m}(\mu_x)_{m\times 1} = (\mu_y)_{n\times 1}$ [mass preservation law], $\sum \rho_{\text{row-wise}} =$ [1, 1, ... m terms] and $0 \leq \rho_{i,j} \leq 1$ ∀ i,j

   m being the solution, Ricci-curvature = $1 - \frac{m}{\text{hop-distance(source, target)}}$

Results:

Experiment 1:

The graph is shown in picture and the edge weights taken are as follows:

1. source node = 1, target node = 2, weight = 6
2. source node = 1, target node = 5, weight = 3
3. source node = 1, target node = 6, weight = 3
4. source node = 2, target node = 3, weight = 20

```
('number of nodes', 7)
('number of edges', 9)
```

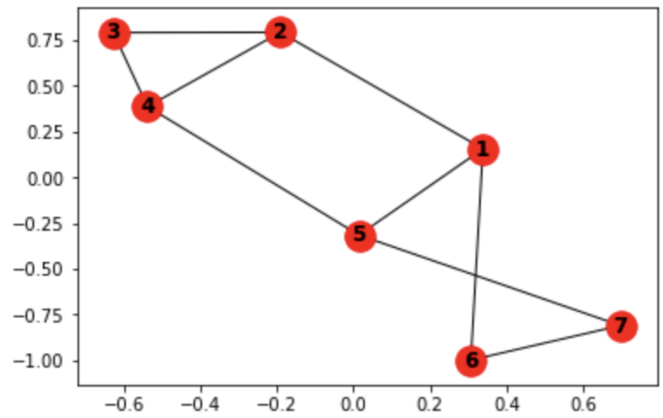

Figure 2: A synthetic graph to have the experiments

5. source node = 2, target node = 4, weight = 6
6. source node = 3, target node = 4, weight = 25
7. source node = 4, target node = 5, weight = 6
8. source node = 5, target node = 7, weight = 6
9. source node = 6, target node = 7, weight = 6

For all the pairs of nodes, the discrete Ricci curvatures are as follows.

1. source node = 1, target node = 2, 'Olivier-Ricci curvature: ', -1.5520833334097541
2. source node = 1, target node = 5, 'Olivier-Ricci curvature: ', -0.9000000000675072
3. source node = 1, target node = 6, 'Olivier-Ricci curvature: ', -1.5000000000529874
4. source node = 2, target node = 3, 'Olivier-Ricci curvature: ', 0.24548611091372918
5. source node = 2, target node = 4, 'Olivier-Ricci curvature: ', 0.5456081080504109
6. source node = 3, target node = 4, 'Olivier-Ricci curvature: ', 0.341861861836402
7. source node = 4, target node = 5, 'Olivier-Ricci curvature: ', -2.542342343262393
8. source node = 5, target node = 7, 'Olivier-Ricci curvature: ', 0.0999999996485249
9. source node = 6, target node = 7, 'Olivier-Ricci curvature: ', 0.16666666666666696

Experiment 2:

Later we devised a random graph generation protocol, that will create a connected graph and measure the Ricci and scalar curvature. Please take a look in out Github link provided. The adjacency matrix is given as:

0.	0.	3.	5.	0.	0.	0.	4.
0.	0.	7.	0.	0.	0.	1.	0.
3.	7.	0.	0.	0.	7.	0.	6.
5.	0.	0.	0.	6.	2.	4.	0.
0.	0.	0.	6.	0.	8.	1.	0.
0.	0.	7.	2.	8.	0.	0.	0.
0.	1.	0.	4.	1.	0.	0.	0.
4.	0.	6.	0.	0.	0.	0.	0.

The result that we had is as follows:

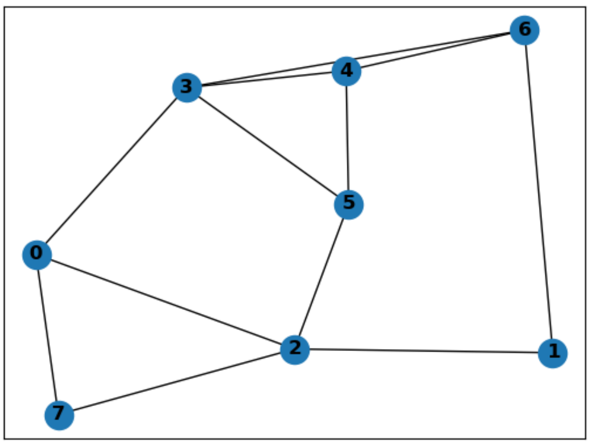

Figure 3: A randomly generated graph to generate Ricci and scalar curvature

1. source: 0, target: 2, Olivier-Ricci curvature: 0.007246375949974571
2. source: 0, target: 3, Olivier-Ricci curvature: -0.050000000043875836
3. source: 0, target: 7, Olivier-Ricci curvature: -0.03333333418645945
4. source: 1, target: 2, Olivier-Ricci curvature: 0.21894409913674573
5. source: 1, target: 6, Olivier-Ricci curvature: -6.00000000054448
6. source: 2, target: 5, Olivier-Ricci curvature: 0.4184782598317718
7. source: 2, target: 7, Olivier-Ricci curvature: 0.11594202540880472
8. source: 3, target: 4, Olivier-Ricci curvature: 0.0711538459805473
9. source: 3, target: 5, Olivier-Ricci curvature: -0.15625000005265233
10. source: 3, target: 6, Olivier-Ricci curvature: 0.23958333330425508
11. source: 4, target: 5, Olivier-Ricci curvature: 0.26510988913964584
12. source: 4, target: 6, Olivier-Ricci curvature: -1.4358974359231937
13. The scalar curvature is: -6.339022941998916

Future Work:

We want to extend this work to calculate discrete Ricci flow on networks in a C++ implementation with a novel fast convex optimization library written in C++ and apply it on different networks (like social network, internet, cancer network) to design an experimental framework for the mathematical advancement made in these genre.

Acknowledgments

We would particularly like to thank Prof. Xiangfeng Gu for some useful conversations in earlier stages of this work.

References

[1] F. L. . J. G. Chien-Chun Ni, Yu-Yao Lin. Community detection on networks with ricci flow. *Scientific Reports*, 2019.

[2] J. G.-D. G. Ni, Lin and E. Saucan. Ricci curvature of the internet topology. *IEEE Conference on Computer Communications*, pages 2758–2766, 2015.

[3] J. J. E. S. R P Sreejith, Karthikeyan Mohanraj and A. Samal. Forman curvature for complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2016.

[4] E. R. L. Z. I. K. Y. S. . A. T. Romeil Sandhu, Tryphon Georgiou. Graph curvature for differentiating cancer networks. *Scientific Reports, Nature*, 2015.

[5] T. G. Romeil Sandhu and A. Tannenbaum. Ricci curvature: An economic indicator for market fragility and systemic risk. *Scientific Advances*, 2016.