Ollivier-Ricci Curvature for Directed Weighted Graphs

Karthik Sivaramakrishnan
Advisor: Abhishek Halder
University of California, Santa Cruz

December 22, 2020

Contents

Chapter 1

Graphs

1.1 What is a graph?

A graph is a diagram which depicts the relationship between two or more variables and can be expressed in various ways. The variables are representing using the notion of points, vertices or nodes. The relationship between each of these nodes are represented as lines, branches or links, which are formally known as edges, that connect each point to its corresponding pair. In mathematical notation, a graph is denoted as $G = (V, E)$ in which E is the set containing the edges and V is the set containing the nodes. One thing to note is that edges create relationships amongst all nodes in a graph, which is called an adjacency relation. This adjacency relation is quantified by a square matrix known as an adjacency matrix. In an adjacency matrix, the relation between each node is indicated by whether or not an edge exists between two nodes. We will be focusing on two types of graphs: directed and weighted graphs.

1.2 Directed Graph

Edges in a graph may have directions. In particular, when a direction is specified between any two nodes, we refer to the graph as a directed graph. A directed graph and an edge, known as an arc in this case, are denoted by [Gui] as ordered pairs of the form $G = (V, E)$ as well as (b, a) or (a, b) respectively, where a, b are nodes. Each arc is distinct going from node a to b and vice versa, shown in Figure 1.1. Additionally, if a directed graph does not contain any self-loops nor multiple arcs, then the graph is *simple*. Next we can define a walk in a directed graph to be a sequence of the form $n_1, e_1, ..., n_{m-1}, e_{m-1}$ thus $e_k = (n_j, n_{j+1})$, and a walk is *closed* given that $n_1 = n_k$. Using this idea of a walk, the definition of a path can be understood as a walk through distinct nodes.

1.2.1 Adjacency Matrix of a Directed Graph

In a directed graph, an adjacency matrix A is a square matrix such that each element of the square matrix consists of 0 if an edge does not exist between two nodes or 1 if an edge does exist. This specific adjacency matrix often exhibits zeros along the diagonal as the nature of the directed graph does not allow for self-loops nor multiple arcs. These are characteristics of

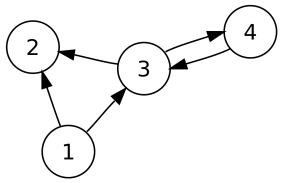

Figure 1.1: Here we have a graph, from [Com06], with four nodes or vertices labeled as $1, 2, 3, 4$. The directed edges between the vertices are shown too.

a simple finite directed graph. This adjacency matrix can be interpreted in two different ways: **(1)** Each element $A_{ab} > 0$ designates an edge from nodes a to b or **(2)** each element $A_{ab} > 0$ designates an edge from nodes b to a. The first interpretation is often referenced and utilized widely in graph theory, whereas the second interpretation is often used in understanding graphs related to, for instance, linear dynamical systems.

1.2.2 Network and Flow

A network is a type of directed graph which contains a distinct origin o and a distinct destination d, where every arc e has a capacity such that $c(e) > 0$. Networks are often used in visualizing settings such as transportation of goods across a physical network, as well as transportation of data through a digital network. We define the flow in a network to be a function, $g : c(e) \to \mathbb{R}$, where $g(e) \in [0, c(e)]$ $\forall e$, thus for every node that is not the origin nor the destination observe

$$\sum_{e \in E_n^-} g(e) = \sum_{e \in E_n^+} g(e),$$

where E_n^+ and E_n^- express the sets of arcs (a, b) and (b, a) respectively.

1.3 Weighted Graph

A weighted graph is defined as a graph that has specific values associated with the edges. A weighted graph can also be considered a network in that the edge weights can provide meaningful information over that network. This type of graph, as depicted in Figure 1.2, is often associated with notable problems such as the travelling salesman problem and the minimum path length problem.

1.4 Centrality Measure

Centrality measures quantify the relative importance of nodes in a graph. This can be expressed as a real-valued function on the nodes of a network such that the measurement will rank each essential node. Of course there are a few notable centrality measures, outlined by [Eat13], which are of importance when mentioning directed graphs.

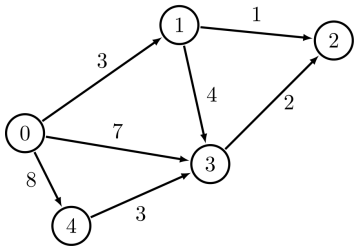

Figure 1.2: Here we have a weighted graph, from [Aca], with five nodes or vertices labeled as $0, 1, 2, 3, 4$. This graph has values assigned to its edges, which are denoted as weights.

1.4.1 Degree Centrality

Degree centrality is the number of links a certain node has. Degree may refer to information, risks, and numerous other factors. In terms of a directed graph, there exists two types of degree measures called indegree and outdegree. Indegree represents the amount of links passing into a node, while outdegree represents amount of links pointing towards other nodes. Indegree and outdegree can be expressed as positive and negative characteristics in relation to the number of links pointing towards a single node and directed outwards to other nodes. Let n be a node on a network such that its degree centrality is given to be $C_D(n) = \deg(n)$. We define the node with maximum centrality to be n^* and $S = (Z, U)$ be the $|Z|$-node network which maximizes

$$F = \sum_{i=1}^{|Z|} C_D(z^*) - C_D(z_i),$$

where z^* denotes the node with maximum centrality in S. In terms of expressing degree centralization,

$$C_D(G) = \frac{\sum_{i=1}^{|V|} C_D(n^*) - C_D(n_i)}{\sum_{i=1}^{|Z|} C_D(z^*) - C_D(z_i)}.$$

Referring back to Figure 1.2, we can work out a simple example of degree centrality on a weighted directed graph. We can construct an adjacency matrix of the form

$$A = \begin{pmatrix} 0 & 1 & 2 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 \end{pmatrix},$$

and multiply a column vector of ones to obtain the total number of links each node has, which can be written as

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \\ 0 \\ 1 \\ 1 \end{pmatrix}.$$

Thus we find that node 0 has the highest degree centrality measure of 3. Even though node 0 has a high number of total links, this does not tell us anything about the information on the edges.

1.4.2 Closeness Centrality

In a graph or network where every node is connected (meaning that there exists at least one directed path or sequence of arrows), we define the closeness of a node to be the average minimum path length from a single node or central node to all other nodes in a graph. This closeness can be defined in the following way,

$$C(a) = \frac{1}{\sum_b d(b,a)},$$

where a, b are nodes of a graph and $d(b,a)$ represents the distance between the nodes a and b. We can recall Figure 1.2 to work out a simple example of closeness centrality on a weighted directed graph. Lets construct an adjacency matrix of the form

$$A = \begin{pmatrix} 0 & 1 & 2 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 \end{pmatrix}.$$

We can take the sum of each row to obtain the total minimum path distance, which can be written as

$$\begin{pmatrix} 0 & 1 & 2 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1+2+1+1 \\ 1+1 \\ 0 \\ 1 \\ 2+1 \end{pmatrix} = \begin{pmatrix} 5 \\ 2 \\ 0 \\ 1 \\ 3 \end{pmatrix}.$$

Thus, the node with the highest closeness centrality is node 0 with a value of 5. Although node 0 may be the closest to most nodes in the graph, this does not provide insight to the information on the edges or the pairwise interactions between the nodes.

1.4.3 Betweenness Centrality

The betweenness centrality enumerates the amount of times a node becomes a "bridge" along a path with minimum length between two nodes. For example, if a minimum path between two nodes was picked randomly which had a set of nodes with high probability of occurrence, then these set of nodes would be considered to have high betweenness. We can compute this centrality measure by finding the minimum path between two nodes, then compute the fraction of minimum paths which pass through a certain node of interest, and finally we take the sum of these fractional amounts. Let p_{ab} denote every minimum path from nodes a to b and let $p_{ab}(n)$ represent

the amount of minimum paths from the total which actually pass through a specific node n, therefore we obtain

$$C_B(n) = \sum_{n \in N} \frac{p_{ab}(n)}{p_{ab}}, \quad n \neq a \neq b.$$

We can use Figure 1.2 to figure out the betweenness centrality, so observe

$$C_B(0) = 0, \quad C_B(2) = 0, \quad \text{and} \quad C_B(4) = 0,$$

since there does not exist pairs of nodes which connect to the above nodes as "bridges". Next, we can calculate the betweenness measure for node 1, 3 and 4:

$$C_B(1) = \frac{p_{02}(1)}{p_{02}} + \frac{p_{03}(1)}{p_{03}} + \frac{p_{04}(1)}{p_{04}} = \frac{1}{2} + \frac{0}{1} + \frac{0}{1} = \frac{1}{2},$$
$$C_B(3) = \frac{p_{02}(3)}{p_{02}} + \frac{p_{04}(3)}{p_{04}} + \frac{p_{12}(3)}{p_{12}} + \frac{p_{42}(3)}{p_{42}} = \frac{1}{2} + \frac{0}{1} + \frac{0}{1} + \frac{1}{1} = \frac{3}{2}.$$

Therefore, node 3 has the highest betweenness centrality measure of 3/2. Even though node 3 may be the most common "bridge", this centrality measure does not utilize the weights on the edges. Similar to the previous centrality measures, this example does not give us additional details about the edges of the graph. This is the primary reason that motivates the notions of graph curvatures to understand this pairwise interaction between every node, which will be explained in detail in the next chapter.

Chapter 2

Graph Curvature

2.1 What are curvatures in a graph?

We have discussed a bit about the ideas of graphs, nodes and edges, but now we can move into a topic that dives deep into the geometry of graphs and networks. Specifically, we will focus on the *curvature* of a graph.

The idea of a curvature is intuitive for a smooth surface. Suppose there are two paths on a smooth surface that are parallel to one another. For a surface with positive curvature, the two paths become closer to each other. Whereas if the surface had negative curvature, the paths would go further away. In Figure 2.1, we can visualize the shapes of these positive and negative curvatures. It turns out this idea of curvature can be generalized from smooth surfaces to discrete structures such as graphs. Next, we explain different notions of curvature for a graph as presented in [Oll13].

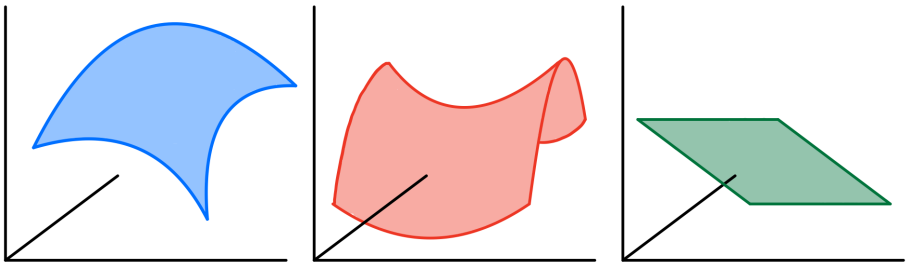

Figure 2.1: The left graph depicts positive curvature, the middle graph depicts negative curvature and the right graph is for zero curvature.

2.1.1 Sectional Curvature

Let (X, d) be a Riemannian manifold. Suppose v and w_x are two unit-length tangent vectors at a certain point $x \in X$ with $\epsilon, \delta > 0$. Let y be the endpoint of δv and w_y be obtained by parallel transport of w_x starting from point x to point y. Therefore, as (ϵ, δ) goes to 0, we have

$$d\left(\exp_x(\epsilon w_x), \exp_y(\epsilon w_y)\right) = \delta\left(1 - \frac{1}{2}\left(\epsilon K(v, w) + O(\epsilon^3 + \epsilon^2\delta)\right)\right).$$

This produces the sectional curvature at point x in the directions v and w, denoted as $K(v, w)$. Sectional curvature is dependent on two tangent vec-

tors, whereas Ricci curvature, introduced next, depends on a single tangent vector. This is achieved by averaging $K(v, w)$ across every directions w.

2.1.2 Ricci Curvature

Let us define x as a point in N-dimensional Riemannian manifold, and a unit tangent vector v at the point x. We can express the Ricci curvature along v, $\mathrm{Ric}(v)$, as the product of the average of $K(v, w)$ and N, which the average is taken over w running over the unit sphere in the tangent space T_xX. In this expression, N is denoted as a scaling factor and it originates from the usual definition of Ricci curvature as the trace of a linear map. Although this does not result in the average over a unit sphere, this does turn into a sum over some basis. Utilizing the previous statement for sectional curvature, we can also show that the average distance between two balls is

$$W = \delta\left(1 - \frac{\epsilon^2}{2(n+2)}\mathrm{Ric}(v) + O(\epsilon^3 + \epsilon^2\delta)\right),$$

which can be written out further to represent $\mathrm{Ric}(v)$,

$$\frac{W}{\delta} = 1 - c\mathrm{Ric}(v), \text{where } c \text{ is a constant } \implies \mathrm{Ric}(v) = 1 - \frac{W}{\delta}.$$

Curvatures refer to the distribution of the edge weights, so we can use this to define Ricci Curvature for graphs. Suppose we have a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Let $\mu_i(j), \mu_j(k)$ be probability vectors such that

$$\mu_i(j) = \frac{w_{ij}}{\sum_{j\in N(i)} w_{ij}} \quad \text{and} \quad \mu_j(k) = \frac{w_{jk}}{\sum_{k\in N(j)} w_{jk}},$$

where w represents the weights along the edges and $N(i) := \{j \in V \mid e_{ij} \in E\}$ is the neighborhood of node i (same definition for $N(j)$). This defines a set of probability measures $\mathcal{M} = \{\mu_1, \mu_2, \mu_3, ..., \mu_m\}$, where $n = |V|$ is the number of nodes of the graph. Therefore for any $\mu_i \in \mathcal{M}$, μ_i is a probability measure in the neighborhood of node i, $N(i)$.

2.2 Ricci Curvature Computation

Let us expand upon the previous paragraph where we defined Ricci Curvature. Let $\kappa(i, j)$ be the curvature between a node x and a node y (for an edge $\overline{xy}$). Let $W_1(\mu_i, \mu_j)$ denote the 1-Wasserstein distance explained in [Vil03] (also known as Earth-mover distance). Let $d(i, j)$ denote the hop distance or combinatorial graph distance, where if nodes i and j are connected then $d(i, j) \neq 0$, otherwise $d(i, j) = 0$. Therefore, we find that our computation involves solving

$$\kappa(i, j) = 1 - \frac{W_1(\mu_i, \mu_j)}{d(i, j)}.$$

The 1-Wasserstein distance $W_1(\mu_i, \mu_j)$ can be written as a Linear Programming (LP) problem

$$\begin{aligned} W_1(\mu_i, \mu_j) = \min_{\mu^{ij}(x,y)} \quad & \sum_{x \in N(i), y \in N(j)} d(x,y)\mu^{ij}(x,y) \\ \text{subject to} \quad & \mu^{ij}(x,y) \geq 0, \\ & \sum_{y \in N(j)} \mu^{ij}(x,y) = \mu_i(x), \\ & \sum_{x \in N(i)} \mu^{ij}(x,y) = \mu_j(y). \end{aligned}$$

In a MATLAB program, we implemented the curvature computation for a weighted adjacency matrix on a directed graph. We used the optimization parser CVX (`http://cvxr.com/cvx/`) to solve the LP mentioned above. Next, we explain the computational procedure using a simple weighted adjacency matrix.

Suppose we have **4** nodes, where the weighted adjacency matrix is

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0.9157 & 0 & 0.7577 & 0 \\ 0.7922 & 0.8491 & 0 & 0.7060 \\ 0.9595 & 0.9340 & 0 & 0 \end{pmatrix},$$

then we can insert this weighted adjacency matrix into the function `G = digraph(A)`, which then creates a weighted directed graph, as shown in Figure 2.2, that we can plot.

With the weighted directed graph created, the next step is to find the out-neighborhood for each node in the graph. This consists of assigning a vector for the out-neighborhood nodes (`N{i} = neighbor_nid`), the associated weight of the edges for each node to the out-neighbor (`{i} = G.Edges.Weight(neighbor_eid)`) and finally the probability distribution on out-neighborhood for each node (`mu{i} = w{i}/sum(w{i})`. Next, we find the shortest path distance matrix for every pair of nodes in `G`, which can be determined by using a convenient MATLAB function `d = distances(G)`.

Before proceeding to the computation, the variables of 1-Wasserstein distance as well as Ricci curvature are initialized. Next, we determine what nodes are reachable from node i and check to see if there is at the least one node which is reachable from node i. If so, then the code moves forward into a for loop that contains an if-condition which verifies that the j^{th} reachable node from node i has at least one single-hop neighbor. Once we have every out-neighborhood of each node in the graph and the reachable nodes from each node, we can construct the neighborhood distance (`C = d(N{i},N{reachable_nodes{i}(j)})`). The script (EMD_Primal.m), which is provided the neighborhood distances and both probability distributions of node i including the j reachable node from node i, calculates using CVX, the 1-Wasserstein distance (`OneWasserstein(i,reachable_nodes{i}(j)) = EMD_Primal(C, mu{i}, mu{reachable_nodes{i}(j)})`). With all

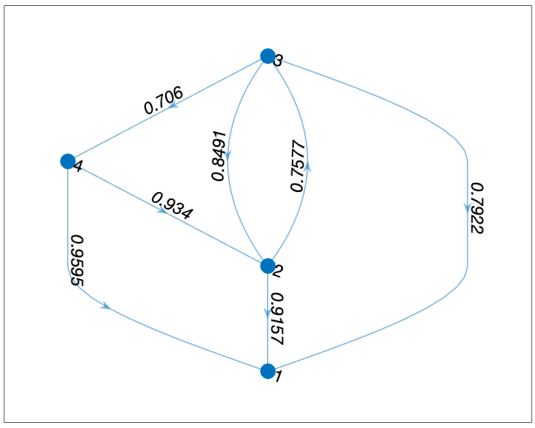

Figure 2.2: This is our example of the weighted directed graph with 4 nodes, given a 4×4 adjacency matrix with predetermined weights.

the components we have thus far, we are able to proceed to the curvature computation which will yield a matrix κ consisting of curvature values,

$$\kappa = \begin{pmatrix} \text{und} & \text{und} & \text{und} & \text{und} \\ \text{und} & \text{und} & -\infty & -\infty \\ \text{und} & 0.2588 & \text{und} & 0.5960 \\ \text{und} & 0.5930 & -\infty & \text{und} \end{pmatrix},$$

where curvatures that are undefined are denoted as **und**.

In the above matrix, we find that there are curvatures which are undefined. The curvatures on the diagonal of κ are undefined since there are no self-loops in the graph, thus there either does not exist any reachable nodes from itself or any single-hop neighbors from each node to itself. Similarly, we find that there either does not exist any reachable nodes from node 1 or nodes $2, 3$ and 4 from node 1 do not have any single-hop neighbors (curvatures are also undefined from nodes $2, 3,$ and 4 to node 1). This is due to the fact that node 1 as an origin or destination is not directed towards any other node in the graph. We also observe that the above matrix exhibits curvatures of $-\infty$ since these *inter-ball vertex* hop distances are not reachable, implying that the respective entries in the distance matrix are set to $+\infty$. The entries of the distance matrix become the coefficients of the linear objective function in the LP, therefore the problem outputs $+\infty$, which proves that the problem is infeasible for these 3 entries (i.e. $\kappa(i,j) = 1 - (W_1(\mu_i, \mu_j)/d(i,j))$, where $W_1(\mu_i, \mu_j) = +\infty$, thus $\kappa(i,j) = -\infty$).

Conclusion

In this paper, we discussed the notion of curvature on a graph to understand the pairwise interaction between nodes. This is motivated by the fact that traditional graphs typically utilize centrality measures, which do not capture these pairwise interactions and the idea that curvatures can be generalized from smooth surfaces to discrete structures such as graphs. We utilized Ricci curvature and formulated the 1-Wasserstein distance as an LP to demonstrate the curvature computation for a directed weighted graph. As a result, these curvature values can illustrate the essential information between nodes that normally would not translate over in centrality measures.

A possible scenario in which this method would be useful in is the application to COVID-19 data. Specifically analyzing the transmission of the virus across all counties in the United States, where automobile transportation is heavily relied on. We can visualize this on a directed weighted graph by letting the nodes denote the counties, where the nodes contain the number of COVID-19 cases. Then, let the edges of the graph denote the number of people travelling between counties on a daily basis. By performing the curvature computations, we can describe the meanings of positive and negative curvatures in relation to this data. For spaces with positive curvature, this means that there are small amounts of foot traffic between the counties implying less spreads of the virus. For spaces with negative curvature, this means that there are significant amounts of foot traffic between the counties implying larger spreads of the virus. Thus, this information will help in making decisions to optimally intervene in the spread of COVID-19.

Bibliography

[Vil03] Cédric Villani. *Topics in Optimal Transportation*. American Mathematical Society, 2003.

[Com06] Wikimedia Commons. *Directed Graph*. Last accessed 18 December 2020. 2006. URL: https://commons.wikimedia.org/wiki/File:Directed_graph.svg.

[Eat13] Eric Eaton. *Network Centrality*. Last accessed 18 December 2020. 2013. URL: https://cs.brynmawr.edu/Courses/cs380/spring2013/section02/slides/05_Centrality.pdf.

[Oll13] Yann Ollivier. "A visual introduction to Riemannian curvatures and some discrete generalizations". In: *Analysis and Geometry of Metric Measure Spaces: Lecture Notes of the 50th Séminaire de Mathématiques Supérieures (SMS), Montréal, 2011*. Ed. by Galia Dafni, Robert McCann, and Alina Stancu. AMS, 2013, pp. 197–219. URL: https://hal.archives-ouvertes.fr/hal-00858008.

[Aca] Jet Brains Academy. *Theory: Weighted Graph*. Last accessed 18 December 2020. URL: https://hyperskill.org/learn/step/5645.

[Gui] David Guichard. *Combinatorics and Graph Theory*. Last accessed 18 December 2020. URL: https://www.whitman.edu/mathematics/cgt_online/book/.