Ricci Curvature in Network Embedding and Clustering

Ginte Petrulionyte

4th Year Project Report
Computer Science and Mathematics
School of Informatics
University of Edinburgh

2020

Abstract

Graph embedding is one of the natural problems faced when dealing with networks - if and how it is possible to represent the same information conveyed by the graph structure as points in Euclidean n-dimensional space. It is an active research area, with different methodologies proposed to improve both embedding quality and the accuracy of tasks performed on the data post embedding, like clustering and prediction.

To improve the representations obtained by embedding methods, we incorporate information provided by graph curvature, in particular the discrete Ollivier-Ricci curvature. It provides a way to describe the intrinsic shape of a network, at least locally, drawing on analogies to differential manifolds.

In this work we first explore and establish the positive impact on post-embedding clustering of applying Ricci flow to the original graph. It is an iterative process which makes the curvature of the graph closer to that of Euclidean space, suggesting a closer match for embedding. However, the computational cost of applying Ricci flow is very high. Noting this, we propose a novel way to regularize random walk based embedding algorithms - using weights derived from the Ollivier-Ricci curvature values, in a way similar to the flow procedure. We show that the addition of this regularization term improves the performance of the original models in clustering tasks. In addition, the benefit observed exceeds that of adding a related simpler regularization term. In addition to these, we provide analysis of the impact Ricci curvature has on clustering via non-negative matrix factorization. This serves an example of a more direct application in clustering, while maintaining behavior similar to other methods described.

Acknowledgements

Thank you to my supervisor Rik Sarkar, and PhD student Benedek Rozemberczki - for answering all my questions, holding brainstorming sessions, sharing their ideas, and sometimes having more optimism regarding my work than I did myself.

A less serious, but nonetheless important thank you - to the people who asked “So what is your project about?” and actually had the patience to listen to my attempts of explaining differential geometry and its connection to graphs (all of which I had yet to understand myself). With a special mention for all those who did not seem like they regret ever asking the question around two minutes in.

Table of Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Contributions . . . 3
- 1.3 Report Overview . . . 4

2 The Theoretical Background on Ricci Curvature **5**
- 2.1 The Geometric Notion of Curvature . . . 5
- 2.2 Discretization of Ricci Curvature . . . 8

3 Network Embeddings and Discrete Ricci Curvature in Network Analysis **11**
- 3.1 Ricci Curvature in Network Analysis . . . 11
- 3.2 Network Embeddings . . . 12

4 Exploration of Embedding After Ricci Flow **15**
- 4.1 Ricci Flow and its Discretization . . . 15
- 4.2 Preliminary Experiments . . . 17
  - 4.2.1 Impact of Ricci Flow on Simple Graphs . . . 17
  - 4.2.2 Experiment Design . . . 18
  - 4.2.3 Types of Graphs Examined . . . 19
  - 4.2.4 Embeddings Used . . . 19
  - 4.2.5 Clustering Evaluation Method . . . 21
  - 4.2.6 Results . . . 21

5 Ricci Curvature in Random Walk Based Embedding **24**
- 5.1 Motivation and Hypothesis . . . 24
- 5.2 Setup and Incorporation into the Loss Function . . . 25
  - 5.2.1 DeepWalk Loss Function . . . 25
  - 5.2.2 GEMSEC Loss Function . . . 26
  - 5.2.3 Ricci Curvature Based Regularization . . . 27
- 5.3 Datasets Used for Evaluation . . . 28
- 5.4 Evaluation of Clustering on Embedded Networks . . . 29
  - 5.4.1 Modularity . . . 29
  - 5.4.2 Implementation . . . 30
  - 5.4.3 Experimental Results . . . 30
  - 5.4.4 Regularization Coefficient Weighting Strategies . . . 31
  - 5.4.5 Computational Cost . . . 33

6 Ricci Curvature in Embedding and Clustering via Matrix Factorization 36
6.1 Non-negative Matrix Factorization 36
6.2 Relation to Clustering 37
6.3 Experimental Evaluation 37
6.3.1 Setup and Matrix Construction 37
6.3.2 Results 38
6.3.3 Impact of Different α Values 39

7 Conclusions and Future Work 40
7.1 Proposed Extensions 40
7.1.1 Impact of Ricci Curvature Based Regularization on Classification Tasks 40
7.1.2 Other Potential Extensions 42
7.2 Critical Discussion 43
7.3 Conclusion 44

Bibliography 46

Chapter 1

Introduction

1.1 Motivation

The start of the 21st century is often referred to as the "Age of Data". More sources and different datasets are available than ever before. However, not all information is limited to the structure of neat multidimensional features. In fact, many types of data are best represented as networks when their salient features are related to the interaction between entities. Examples of such data include people and their connections in a social network, roads and the layout of cities, interactions of different drugs, biological structures. While this is a very diverse set of examples, they have a rather glaring similarity - the relation of different items to each other carries crucial information, sometimes more important than the specifics of the objects themselves.

This data can be quite naturally phrased as a set of nodes - items - and a set of edges - the connections between pairs of the items. In other words, a graph[1], $G = (V, E)$.

This poses a challenge. Modern machine learning methods allow us to analyze data with an unparalleled level of efficiency and obtain results in tasks like classification and clustering with more accuracy than ever seen before. But there is a caveat - most conventional methods are created to analyze data represented as points in high-dimensional vector spaces, not graphs and their edges. However, losing the information provided by these relations could reduce the usefulness of a dataset represented by a network greatly and is undesirable.

A possible solution is to create brand new methods (an example of this is graph neural networks [40]). This is not only a task as difficult as the initial development of known methods, but also takes a step back from the expertise and extensive studies available when using more conventional methods. A different point of view suggests adapting the data instead. Finding a representation compatible with the established framework for it. In particular, we must find a good mapping from a network to a vector space - often the familiar d-dimensional Euclidean space, $\mathbb{R}^d$. This is known as graph embedding - another challenging problem related to networks. The main challenge stems from the need for this to be a good representation - i.e. preserve as much

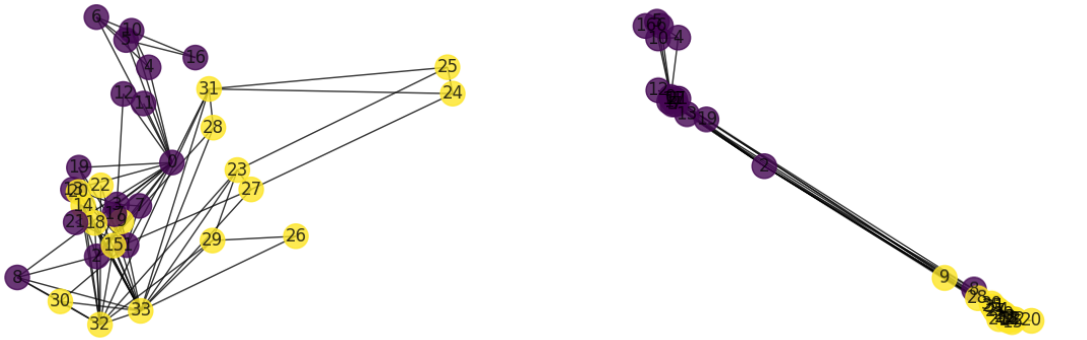
[1] the words "graph" and "network" are used interchangeably throughout this report

of the information inherent to the connections as possible. Solving it requires, beyond just implementation and design, both clarifying what the important properties are, and finding ways to measure how accurate the representation is.

A property shared between both graphs and objects in vector spaces is a notion of "shape". Recent research has shown that the concept of curvature - deviation from flatness, which inherently represents the shape of an object in some sense - can be transferred to the discrete context. Ricci curvature and its discretization [29, 23] are our focus in this report, translating a general notion of curvature from the context of differential manifolds to that of networks. While this has been used to perform tasks like community identification [28, 9] or analysis of network robustness [39], many of its applications are still not explored extensively.

We seek to utilize the information provided by the the discrete Ricci curvature, as defined by Ollivier [29] - something that inherently depends on the edges in the network - and explore if and how it can be used in a solution to the network embedding problem. We hope that using this piece of extra knowledge about the shape and flatness (or lack thereof) of the graph we are working with will allow us to improve the representations in the (flat by definition) target Euclidean space.

Part of the complexity of embedding graphs comes from the fact that obtaining this representation is usually an intermediate step. Afterwards we may seek to visualise the data, classify nodes, identify communities, just to mention a few of the applications. Ideally, it would be best if these tasks could be performed efficiently, and thus a representation in a low-dimensional space is preferable. However, the lower the number of dimensions, the more difficult it is to, for example, preserve distances between nodes, or maintain a complex, potentially overlapping, cluster structure. Clustering can be used to evaluate embeddings - it is expected that a good embedding makes existing clusters more distinct and easier to detect, something we focus on in this report.



(a) Embedded using DeepWalk

(b) Embedded using DeepWalk-Ricci, a Ricci curvature regularized version of the embedding

Figure 1.1: Zachary's karate club graph [51], colored according to club membership and embedded with different methods - DeepWalk [31], an established random walk based method, and a version of it using Ricci curvature based regularization. A clear distinction of the clusters can be seen when the regularization is used.

One natural way to utilize Ricci curvature to improve graph embeddings in Euclidean space it to use the discrete Ricci flow. It is an iterative process, transforming the graph so as to make the curvature everywhere in it close to zero. As Euclidean space is "flat" - has curvature equal to 0 at every point - it is reasonable to expect that a graph transformed by Ricci flow will be easier to represent than the potentially "non-flat" original. However, the Ricci flow computation is a very demanding process, taking several minutes even on small examples (graphs with 100 nodes). This makes the technique hardly feasible on real-world networks, which can easily have tens of thousands of nodes.

A more direct incorporation of Ricci curvature is also non-trivial. A lot of contemporary graph embedding methods are based on minimizing a cost function, thus the inclusion of curvature values, which can be strongly negative, can create imbalance within the optimization procedure.

1.2 Contributions

Summarizing the work completed, we highlight the following contributions as particularly significant:

- We propose a novel way to incorporate Ricci curvature in embeddings, as a regularization term, requiring only a fraction of the computational cost needed for Ricci flow. We note that curvature on edges in dense, well connected parts of the graph tends to be positive. Such areas could be likened to a sphere in the usual sense of curvature - all points can be seen as dense and geodesic lines tend to drift towards each other. The opposite holds for so called "bridge" edges, ones connecting clusters of tighter-knit nodes. This can similarly be likened to a hyperbolic plane, where geodesic lines would drift apart from a common starting point. Thus, if we encourage the endpoints of strongly positively curved edges to be embedded close to each other via regularization, we could expect a cluster structure to become more pronounced, as demonstrated in figure 1.1.
- We show that when Ricci curvature is combined with standard, gradient descent based graph embedding techniques in this way, the performance in clustering of the embedded networks improves significantly. This is shown using the specific examples of the algorithms we propose, DeepWalk-Ricci and GEMSEC-Ricci.
- We demonstrate that applying Ricci flow before performing an embedding improves the clustering performance, showing the initial intuition to have been correct.
- We show Ricci curvature can be used in clustering more directly via non-negative matrix factorization (NMF), improving the performance by up to 30% when compared to factorizing the adjacency matrix.
- We explore different ways to transform and compute Ricci curvature, showing that applying a parametrized sigmoidal transformation to the values to ensure non-negativity leads to the most significant improvement, both when curvature values are used as regularization weights and in NMF.

1.3 Report Overview

In this report we seek to provide a comprehensive summary of the work done, as well as the relevant background and context. It is structured as follows:

- In chapter 2 we provide a brief overview of the different notions of curvature in geometry, as well as the discretization of Ricci curvature and the method used for computing its values.
- Chapter 3 consists of a literature overview for the application of graph Ricci curvature in network analysis as well as an overview of different embedding methods.
- Chapter 4 describes the initial exploratory work, using the discrete Ricci flow (a method of transforming a graph so as to obtain near-zero curvature for every edge), synthetic graphs and simple embedding methods - multi dimensional scaling and spectral embedding. Here we perform comparison of clustering performance on the graph embeddings before Ricci flow is applied, and after.
- In chapter 5 we propose a way to incorporate graph Ricci curvature values in the loss function of a skipgram-based embedding algorithm as well as the results in clustering obtained post-embedding. Here we also provide a summary of the impact on the runtimes of the algorithms and analysis of performance when different strategies to compute the regularization weights are used.
- Chapter 6 presents a way to use Ricci curvature in conjunction with non-negative matrix factorization to improve clustering results, in comparison to the baseline of using the adjacency matrix.
- Lastly, in chapter 7 we provide some concluding remarks and discussion, as well as pointing out possible extensions to follow this work.

Chapter 2

The Theoretical Background on Ricci Curvature

One of the fundamental concepts underlying the work presented in this report is graph curvature, specifically a discretized version of Ricci curvature. In this chapter we present a summary of the theoretical background required to understand the significance of this concept as well as its relevance in relation to graph embeddings.

2.1 The Geometric Notion of Curvature

In geometry, and, by extension, differential geometry, several different notions of curvature can be found. Varying in the complexity and sophistication of the description, as well as restrictions on the domain where they can be applied, all of these measures are referred to as "curvature" as they seek to represent and quantify the same basic concept - the deviation from flatness. Typically the values are defined so as to be large when the object examined deviates strongly from its flat equivalent, and to be close to zero when it is nearly flat.

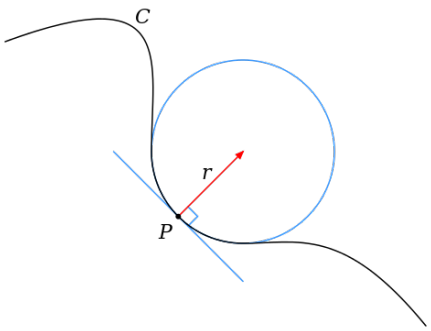


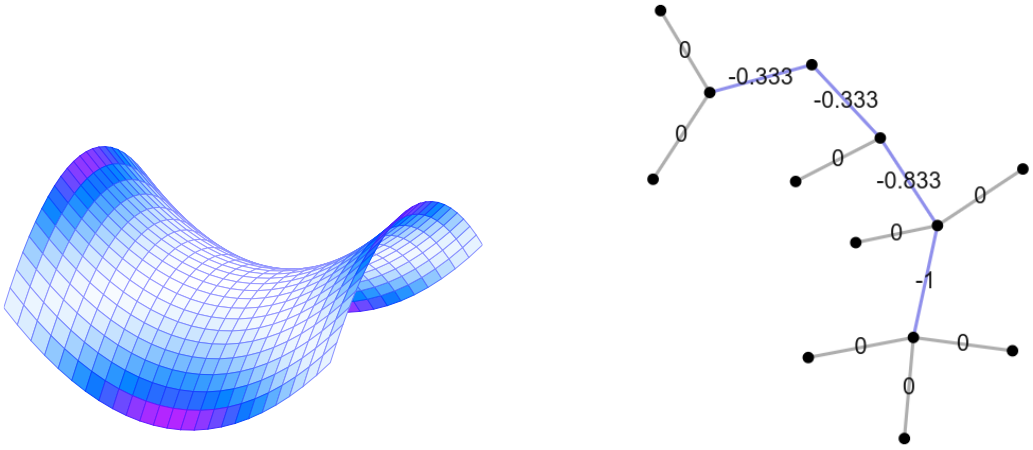
Figure 2.1: Osculating circle to a 2-dimensional curve C [50]

To illustrate this, we first consider the simplest notion of curvature - one for curves in two dimensions. The deviation from flatness here can be understood as the deviation from the curve being a straight line at any particular point. This is defined via an osculating circle (see figure 2.1) - the circle which provides the best approximation

of the curve at a given point. Intuitively it can be seen as the trajectory traced by an imaginary car driving along the path traced by the curve, which had its steering wheel become stuck at the point in question. Naturally, the smaller the radius of the osculating circle, r, the less "flat" the curve is ("turns" more sharply at the point). It is desirable for this to be reflected by higher curvature values. To achieve this, the curvature is defined to be the inverse of the radius, $\kappa = \frac{1}{r}$.

Taking a step-up in complexity, we have the notion of Gaussian curvature, defined for surfaces in three dimensional space. It is a signed value, defined as the product of the two principal curvature values at the given point - the maximum and minimum curvatures of the curves obtained at the intersection of the surface with a normal plane.

More rigorously, consider a point p on a surface M. Let $\mathbf{n}$ be the normal vector and $\mathbf{t}$ be a tangent vector to the surface at p. A normal plane is then one which contains both $\mathbf{n}$ and $\mathbf{t}$. The intersection of this plane with M will produce a curve through p. We can then compute the curvature of this curve at p. Varying the direction of $\mathbf{t}$ will yield different curves. We are interested in the highest and lowest curvature values obtainable this way. Examples of surfaces with positive, negative and zero curvature can be found in figures 2.4a, 2.2a and 2.3a respectively.



(a) Surface with Negative Curvature

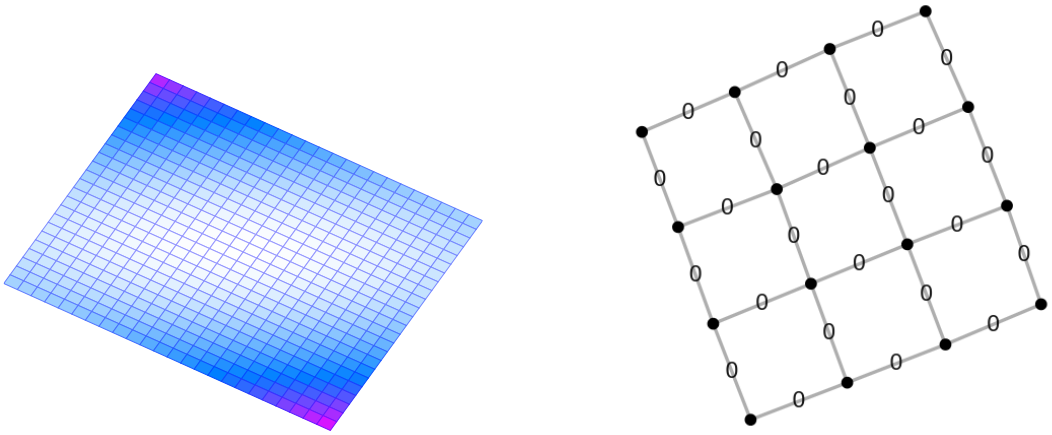
(b) Graph with negative Curvature

Figure 2.2: Parallel Between Surfaces and Graphs - Negative

It is worth noting this measure is intrinsic to the surface and does not depend on the way it is embedded in Euclidean space. Even before taking a closer look at graph specific notions, we can see that defining a similar, intrinsic measure on networks has the potential to give us information on what the representations of these networks in Euclidean space could be.

Furthermore, regions with highly positive curvature tend to be more densely packed than ones with negative curvature. A metric which behaves similarly on graphs could be expected to have a correlation with the community and cluster structure. Indeed, in figures 2.1, 2.1 and 2.1[1] we can see a comparison between graphs and surfaces of

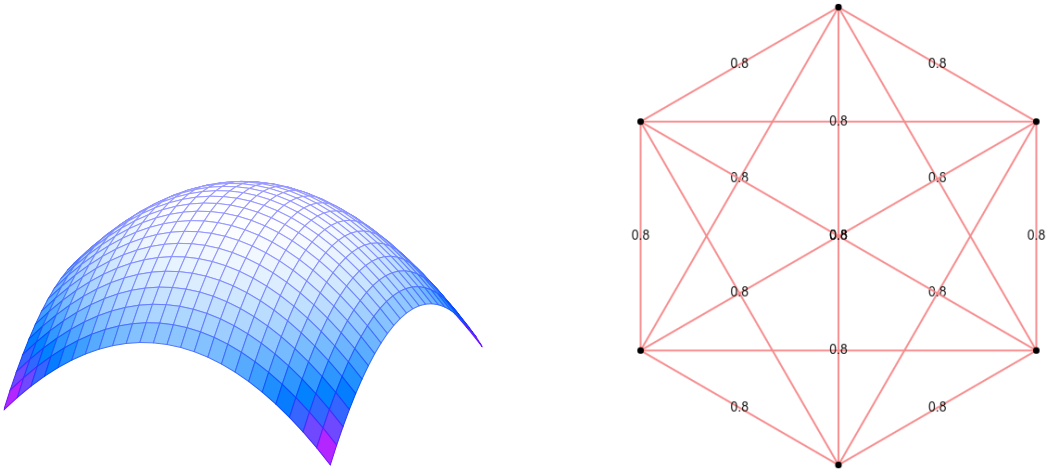
[1]Graph visualisations generated with the graph Ricci curvature calculator [9]



(a) Surface with Zero Curvature

(b) Graph with Zero Curvature

Figure 2.3: Parallel Between Surfaces and Graphs - Zero



(a) Surface with Positive Curvature

(b) Graph with Positive Curvature

Figure 2.4: Parallel Between Surfaces and Graphs - Positive

similar curvature, reflecting this density hypothesis. In particular, we see a regular grid graph has zero curvature, a clique yields a positive value, and a sparser, tree-like structure has negative curvature.

Ricci curvature can be seen as a generalisation of the more basic notions described above, translating the idea of measuring the deviation from flatness into the context of differential manifolds. It represents the amount by which the volume of a geodesic[2] ball on the manifold deviates from the volume of a standard ball in Euclidean space. With the latter being the baseline of flatness, we can see how this is an extension of the common idea referenced before. This particular notion has proven to be extendable to the context of graphs, which is discussed in the following section.

[2] a generalized notion of a straight line, representing an interpretation of "shortest path" on the surface

2.2 Discretization of Ricci Curvature

There have been several approaches proposed to generalize the notion of curvature, and specifically Ricci curvature, to be applied in discrete settings. Several explorations of this curvature on general metric spaces have been conducted by Bakry and Emery [2], as well as Lott and Villani [24]. The first definition of Ricci curvature on graphs was given by Chung and Yau in 1996 [8], a definition on Markov chains was developed by Ollivier [29]. Another notion of the discretization is presented in Forman's work [11].

In this report we focus on the discretization of Ricci Curvature proposed by Ollivier [29] as coarse Ricci curvature. This formalisation was further explored in the context of graphs by Lin, Lu and Yau in 2011 [23] and the explanation here draws heavily on their work. This approach translates the idea behind Ricci curvature quite directly, by comparing the distance between two balls to the distance between their centers.

In order to present the mathematical interpretation of this notion of Ricci curvature on graphs, we first define **Wasserstein distance** (also known as the earth-mover distance). Given two probability measures on a metric space M, μ and ν, a coupling is a probability measure on M, $\gamma(\mu,\nu)$ such that the respective marginal distributions correspond to μ and ν. Let $\Gamma(\mu,\nu) = \{\gamma|\gamma(\mu,\nu) \text{ is a coupling}\}$. Then, given the distance measure d, the Wasserstein distance is defined as

$$W(\mu,\nu) = \inf\{\int_{M\times M} d(x,y)d\gamma(x,y)|\gamma \in \Gamma(\mu,\nu)\} \tag{2.1}$$

In the particular case of graphs, we define a parametrized probability measure on the neighbourhood of a vertex. First, given an undirected graph $G = (V,E)$, for a vertex $v \in V$ let $C(v) = \{u|(v,u) \in E\}$, and let $N(v) = C(v) \cup \{v\}$ be the neighbourhood of v and $d_v = |C(v)|$ be the degree of v. We then define a probability measure on the graph, parametrized by $\alpha \in [0,1]$ as stated in the following equation.

$$m_v^\alpha(u) = \begin{cases} \alpha & \text{if } u = v \\ (1-\alpha)/d_v & \text{if } u \in C(v) \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Wasserstein distance is fundamentally related and, in discrete cases, equivalent to the optimal transport problem, originally posed by G. Monge in 1781 as seeking the optimal way to move iron ore from mines to factories, which can then use it for production, while minimizing the cost of this transportation. In a discrete case, we can see the probability measure in equation 2.2 as a mass distribution, which needs to be transported to another specific arrangement. We thus produce a transport plan, A such that $\sum_{y\in M} A(x,y) = \mu(x)$ and $\sum_{x\in M} A(x,y) = \nu(y)$, equivalent to the couplings defined before (here M refers to the discrete space of places which can hold mass or probability values, such as nodes in a graph).

Now given two nodes $x,y \in V$, and some value $\alpha \in [0,1]$, we can define m_x^α and m_y^α, and compute $W(m_x^\alpha, m_y^\alpha)$. The discrete version of the computation is presented in the following equation.

$$W(m_x^\alpha, m_y^\alpha) = \inf_A \{ \sum_{x_i, y_j \in V} d(x_i, y_j) A(x_i, y_j) | A \text{ is a transportation plan} \} \tag{2.3}$$

We use the length of the shortest path between x and y as the distance metric $d(x,y)$. The α-Ricci curvature for the pair of nodes, $\kappa_\alpha(x,y)$ is then obtained by a comparison of the Wasserstein distance to the distance between the nodes:

$$\kappa_\alpha(x,y) = 1 - \frac{W(m_x^\alpha, m_Y^\alpha)}{d(x,y)} \tag{2.4}$$

Even though this value can be computed for any arbitrary pair of vertices, it is usually restricted to adjacent pairs (with the Ricci curvature being computed for each edge). A visualisation for this case, with x, y being adjacent nodes, can be seen in figure 2.5. The transportation problem pictured here requires moving the mass from $N(x)$ (green) to $N(y)$ (blue), with the lowest possible amount of work (computed as amount of mass multiplied by the distance it is to be transported over).

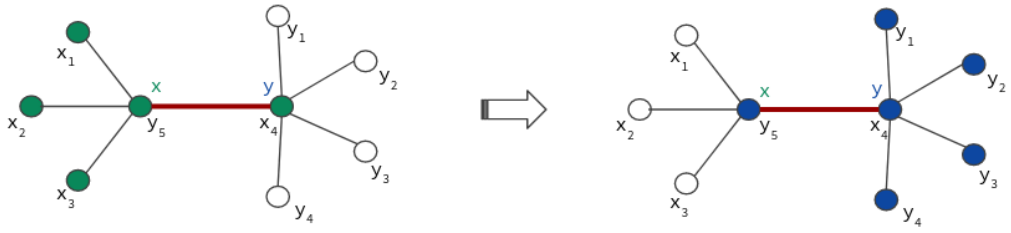


Figure 2.5: Optimal transportation problem - the mass distribution is moved from the neighbourhood of x (green) to the negbourhood of y (blue)

Note that in this case the optimal transportation distance calculation can be expressed as a linear programming optimization problem, presented in equation (2.5). Here we denote the proportion of mass originally at x_i transported to y_j by ρ_{ij}. The first constraint ensures that exactly the amount of mass available at each node in the neighbourhood of x is moved, and is moved in its entirety. The second one ensures the distribution m_y^α is emulated correctly by the transportation result. These constraints together ensure ρ values will comprise a valid transporatation plan.

$$\begin{aligned} \text{Min}: \quad & \sum_j \sum_i d\left(x_i, y_j\right) \rho_{ij} m_x^\alpha\left(x_i\right) \\ \text{S.t.} \quad & \sum_j \rho_{ij} = 1 \quad \forall i \\ & \sum_i \rho_{ij} m_x^\alpha\left(x_i\right) = m_y^\alpha\left(y_j\right) \quad \forall j \\ & 0 \leq \rho_{ij} \leq 1 \quad \forall i, j \end{aligned} \tag{2.5}$$

This optimization is a computationally expensive task, impacting the performance of any algorithm which uses Ricci curvature on larger real-world networks. This is especially prevalent on large and densely connected networks, where there are many

possible paths which need to be considered when computing the optimal transport distance. To reduce the complexity of the calculations, we can also compute the average transportation distance, $A(m_x^\alpha, m_y^\alpha)$. Here we consider sending the same amount of mass from $x_i \in C(x)$ to every $y_i \in C(y)$, repeating this for all $x_i \in C(x)$, and transferring the mass form x to y. This can be computed directly (only requires computing shortest paths, which can be done efficiently with algorithms like Dijkstra and Floyd-Warshall). Due to the transportation plan (represented by the ρ_{ij} values, computed as described above) being fixed, this method does not require solving a linear programming optimization problem.

In this report we consider a version of the discrete Ricci curvature with a fixed value of the parameter, $\alpha = 0.5$, a decision consistent with [26, 27] and other works. This choice allows us to focus on the impact the addition of curvature-specific information has on other algorithms, rather than tuning curvature specific hyperparameters. As it has been shown to provide good results in past work, it is safe to presume this decision will not have a large detrimental impact on our results. Noting this convention, for simplicity and readability we use $\kappa(x,y)$ to refer to $\kappa_{0.5}(x,y)$.

Chapter 3

Network Embeddings and Discrete Ricci Curvature in Network Analysis

There has been a considerable amount of research on the potential application of Ricci curvature on graphs over the past few year, since the initial papers defining it were published. In this section we briefly review the current results of using it in complex networks as well as the reasoning behind the choices made. In addition to this, we cover the background on the other area relevant to this report - network embeddings.

3.1 Ricci Curvature in Network Analysis

The most fundamental application of the discrete Ricci curvature is providing an interpretation for the shape and structure of networks. An example of a study carried out in this context is the 2015 paper analyzing the Ricci curvature of the internet topology [27]. Here the authors show that the internet network predominantly negatively curved, as measured by Ricci curvature. This is presented as consistent with pre-existing work, which used various other ways to examine graph curvature. Different robustness indications which can be derived from this information are also studied, as well as their relation to known geographical distances. The latter provides an indication that positively curved edges tend to be geographically short - the nodes connected by them are quite close together.

In the aforementioned paper the authors also suggest graph curvature could be used in connection with network embeddings. The proposed application is slightly different than the one explored by us - the suggestion here is to consider embeddings to non-euclidean spaces, as they could provide a closer fit for the inherent structure of the graph. This deviates slightly from the traditional approach to graph embedding, characterised by seeking a representation in n-dimensional Euclidean space. The approach we take is partly motivated by the connection proposed here, however we have chosen to maintain the convention of fitting to Euclidean spaces.

Expanding on the observation above, with respect to the endpoints of positively curved edges being “close” in some sense, naturally leads to a different application in network analysis - finding dense clusters. Ricci curvature on graphs provides information on

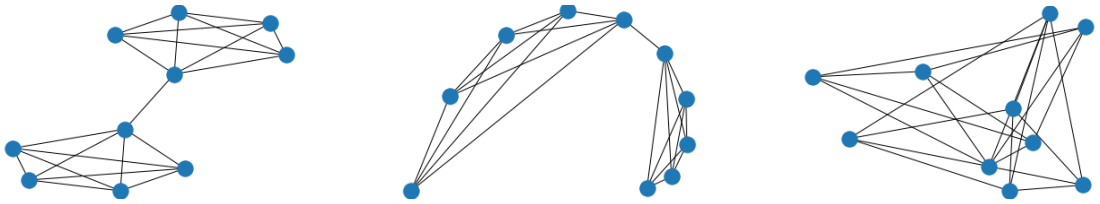
the local structure of the graph, and is connected to how densely it is connected. This can be seen both from the mathematical definition presented in section 2.2, as well as the examples provided in figures 2.2, 2.4 and 2.3. As in real world networks the more densely connected regions tend to bear some similarity to each other, the problem of identifying them arises. Usually, this is phrased as the problem of community detection, although the precise definition of a community varies. It is also closely related to, and sometimes treated as interchangeable with, graph clustering.

The problem is addressed via graph curvature in two recent scientific reports by Ni et al. [28] and Sia et al. [42]. The authors take quite different approaches. The former proposes simply using the Ollivier-Ricci curvature as an indicator for edges which should lie between communities rather than within them. The latter suggests using the information less directly, via Ricci flow (explained in more detail in section 4.1). Both approaches are shown to yield considerable results, and encouraged the further exploration related to clustering presented in our work.

Other applications can be found in a variety of fields. Due to its relation to the robustness of complex networks, also pointed out in previous paragraphs, it has been applied in economics for pre-crisis market fragility study [39], as well as for differentiating cancer networks [38]. In addition to this, uses for discrete Ricci curvature on graphs have been found in the research of wireless networking [45, 46] and quantum computation [47, 48].

3.2 Network Embeddings

Embeddings are nearly inseparable from our understanding of graphs. Most of the time we think of one, we are visualising a two or three dimensional representation, with the nodes positioned in specific locations and edges stretching in between. However there are numerous such representations available, as shown in figure 3.1. Even at a glance, we can tell that they provide us with different amounts of information about the graph, thus finding a good, informative embedding is an important but complex task.



(a) Kamada-Kawai layout (b) Spiral layout (c) Random layout

Figure 3.1: Different embeddings of the same graph (two 5-cliques connected by a single edge) into two dimensions. It can be difficult to determine all pictures represent the same underlying structure at a glance.

More formally, a network embedding is defined as a mapping from the initial structure to a vector space - usually, the familiar Euclidean space. Typically the goal is to preserve the inherent properties of the network as well as possible. This is often done in order to create a representation which can then be used with the standard data

science methods for tasks like clustering and classification. The desire to keep those methods working efficiently has encouraged the study of embeddings into relatively low-dimensional spaces. Such representations are also used for visualisation and link prediction (a more detailed summary of applications can be found in the paper by Goyal and Ferrara [12]).

In this report we focus on vertex embeddings, where the nodes of a network are mapped to coordinates in Euclidean space. Given a graph $G = (V, E)$, they can be seen as a function $f : V \to \mathbb{R}^d$, where d is the dimension of the target space.

The approaches to graph embedding are quite diverse. One method is factorization - where the similarity between every pair of nodes is defined via a chosen metric, a matrix of these values is constructed and an embedding is sought via a factorization of it. An example of a matrix factorization based embedding is Graph Factorization [1], which seeks a factorization of the adjacency matrix[1]. Multidimensional scaling (MDS) and the spectral embedding, both more extensively explained in chapter 4 also fall into this category. Further examples include HOPE [30], which depends on a similarity measure used to construct the matrix to factorize, and others [35, 5].

Some embedding methods rely on deep learning exclusively. Techniques like deep autoencoders, previously used for informative dimensionality reduction, have been adapted to be used on graphs and generate representations of the nodes [6, 49, 18].

Another category of embedding methods utilize random walks on the graph. They follow a similar pattern - sample random walks on the graph (even though the sampling strategies may vary) and then use the SkipGram model to compute the embedding representations. This model is also used in language processing, in algorithms such as Word2Vec [25]. It maximizes the probability that words co-occur when they appear within a specific window in a sentence. The adaptation of this model to graph embedding algorithms treats the sampled random walks as these sentence windows, implying that the nodes appear in close relation to each other.

An early example of random walk based embedding algorithms is DeepWalk [31], proposed in 2014. The random walks here are sampled by uniformly choosing a start node, and then progress to a neighbour, again, uniformly at random. This is contrast to later algorithms, such as Node2Vec [13], which can utilise biased random walks. Multiple random walks of a given length are sampled and then used to train the aforementioned SkipGram model to obtain an embedding.

A variation of DeepWalk we base some work on is GEMSEC (Graph Embedding with Self-Clustering) [36]. This method uses the same random walk sampling approach as before, however the difference lies in the loss function optimized afterwards. GEMSEC uses a clustering cost term, somewhat similar to k-means, to encourage nodes belonging to the same clusters to be embedded close to each other. It also proposes an addition of a social network cost, based on the overlap of the neighbourhoods for every pair of neighbouring nodes, again as an aid in community detection and clustering. This part is what we propose to reinterpret with curvature values in chapter 5. This part of the report also presents more detailed descriptions of DeepWalk and GEMSEC.

[1] A matrix A such that $A_{i,j} = w(i, j)$ where if $(i, j) \in E$, then $w(i, j)$ is the weight of that edge; $w(i, j) = 0$ otherwise

One of the challenges arising alongside the graph embedding task is performance evaluation, as well as the lack of a strict definition of what a good embedding ought to be like. Typically, the tasks mentioned above as applications are performed on the embedded network and the results are then used to compare different methods. Naturally, this makes comparing different approaches a non-trivial task itself. For example, some embedding methods might be better suited to a specific application, rather than seeking to optimise all at once. In this report we have chosen to focus most evaluation efforts on clustering performance.

Chapter 4

Exploration of Embedding After Ricci Flow

This part of our work is focused on developing an understanding of how Ricci flow affects simple embeddings, by studying the clustering performance on the embedded graphs when compared against a ground truth valuation. The main goal here was to produce a feasibility study of the basic method, with the motivation being that applying the flow procedure, which fundamentally attempts to flatten the graph structure, could allow it to be more easily represented via embeddings in Euclidean (and as such, non-curved) space.

We examine the effects on two different embedding methods, multi dimensional scaling (MDS) and spectral embedding (described in section 4.2.4). A positive result, in line with the hypothesis above, is observed when MDS is used. Here post embedding clustering results improve when evaluated with respect to a known ground truth assignment quite consistently. A similar effect is seen when using the Spectral embedding (specifically, the Laplacian Eigenmaps algorithm), although slightly less consistently. For this algorithm the weights on the edges assigned by Ricci flow have to be transformed to reflect closeness rather than distance. Overall, these results imply that Ricci flow can consistently improve some embeddings in terms of clustering performance.

4.1 Ricci Flow and its Discretization

Ricci flow, in its original formulation, is a process of deforming manifolds, formally analogous to the diffusion of heat. Via this process, introduced by Hamilton in 1982 [14]. The space is modified in proportion to the Gaussian curvature so as to make it as uniform as possible. The Ricci flow process smooths out irregularities - regions with positive curvature shrink, while ones with negative curvature tend to expand [28].

Formally, given a Riemannian metric g_{ij}, surface Ricci flow is defined by the differential equation 4.1 which characterizes the evolution of the metric. Here t is the time parameter and $K(t)$ is the Gaussian curvature induced by the metric.

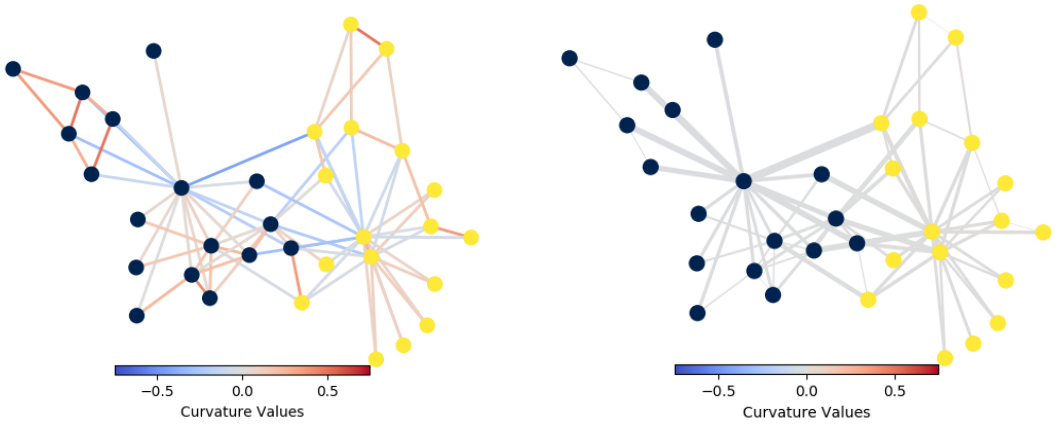
$$\frac{dg_{ij}(t)}{dt} = -2K(t)g_{ij}(t) \tag{4.1}$$

The discrete Ricci flow on surfaces was analyzed by Chow and Luo [7] and an optimization based algorithm for its computation was given in a later paper [15].

In this project we have focused on the discrete Ollivier-Ricci curvature, and the corresponding version of Ricci flow. This procedure was described in the work by Ni et al. studying network alignment [26] as the discrete Ollivier-Ricci flow. Here the curvatures are amended using an iterative process on the edge weights in the graph, aiming to modify them so as to make the curvature uniformly zero. For any edge (x,y), we use the edge weights at step i (denoted by $w_{xy}^{(i)}$), shortest path distances at step i induced by the relevant weights (denoted by $d^{(i)}$) and the Ricci curvature values κ (as defined in section 2.2) to compute the weights at step $i+1$ as follows:

$$w_{xy}^{(i+1)} = d^{(i)}(x,y) - \kappa_{xy}^{(i)} \cdot d^{(i)}(x,y) \tag{4.2}$$

In order to compute any weights for step $i+2$ we thus must update all others up to step $i+1$. Similarly as on manifolds, this process transforms the graph via edge weights so as to achieve as close to uniformly zero curvature everywhere as possible. An illustration of the discrete Ricci flow being applied can be seen in figure 4.1.



(a) The original graph

(b) Graph after Ricci flow has been applied

Figure 4.1: Zachary's karate club graph before and after Ricci flow has been applied. The nodes are colored according to community membership, the edge colors represent the corresponding curvature values, edge thickness corresponds to the edge weight. The layout used is the spring layout for the original unweighted graph.

The research on applications of Ricci flow, alongside the discrete version of curvature in graphs is quite recent, started with the aforementioned work on network alignment in 2018 [26]. Here the weights computed with Ricci flow were used to define a distance measure on the graph (the length of the shortest path in the appropriately weighted version of the graph), which is then used to align graphs via distances from landmark nodes, for which the correspondence is determined from prior knowledge.

This paper was soon followed by another work by the same authors, examining the uses of discrete Ricci flow in community identification [28]. Here the communities are determined by removing the edges for which the weights assigned by Ricci flow exceed

a chosen threshold value. Due to the tendency of the process to contract high-curvature edges (often inner ones in clusters) and lengthen the ones with negative curvature (often bridges between highly connected groups of nodes), this is expected to recover the community structure reasonably well. The authors provide a theoretical proof of this for a specific family of graphs, as well as experimental results, showing performance comparable to or better than other community identification methods on both synthetic and real-world networks.

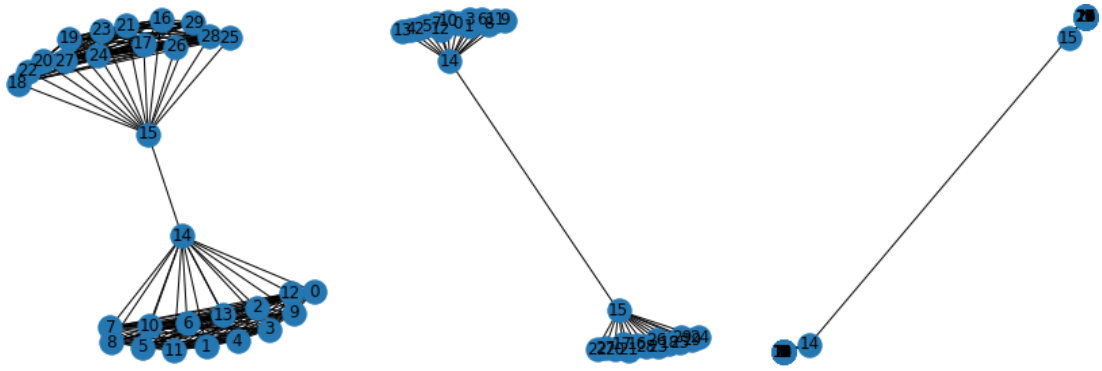
An important takeaway from both papers for us is the general behaviour of Ricci flow - via the iterative process, it contracts areas of high curvature, assigning low weights to originally positively curved edges, and expand areas of negative curvature (assigning large weights to negatively curved edges). As edges within more densely connected clusters are often positively curved, and ones between clusters tend to have negative curvature, this means Ricci flow could move clusters further away from each other, while making them tighter by assigning low weights to inner edges.

The work described in this chapter builds on this notion and aims to explore the impact Ricci flow has on graph embedding and how it can be used to construct meaningful representations.

4.2 Preliminary Experiments

As an initial exploration of the impact of Ricci flow on the quality of embedding and clustering, a series of simple experiments were carried out. The results of these are not entirely conclusive, due to the high variance obtained, however an improvement is typically observed. It can be seen both qualitatively, when visualising 2-dimensional embeddings, and via numerical evaluation, measuring correspondence to a

4.2.1 Impact of Ricci Flow on Simple Graphs



(a) The original graph

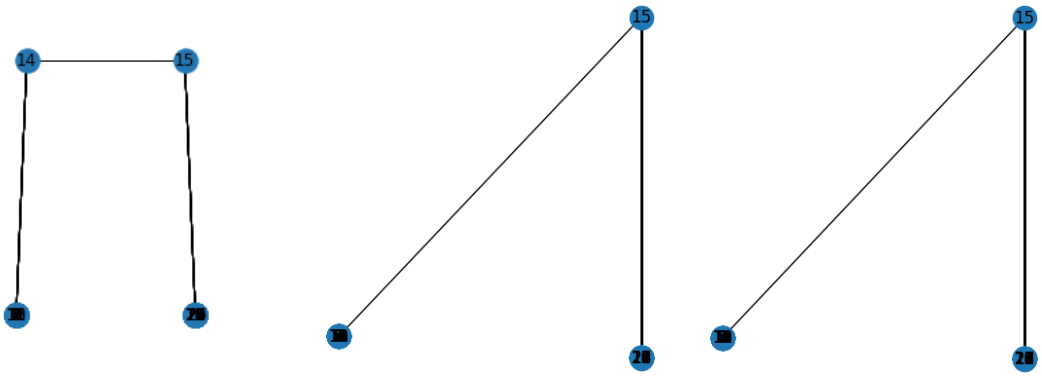
(b) After 2 iterations of Ricci flow

(c) After 100 iterations of Ricci flow

Figure 4.2: Effects on a barbell graph embedded via MDS - the nodes belonging to the cliques are embedded very close together after the graph has been transformed by Ricci flow.

Initial trials were focused on providing a visualisation of the effects Ricci flow has on the 2-dimensional representations of some simple graphs. The visualisations were produced using the embeddings described in 4.2.4.

Figure 4.2 shows a typical example of what is the expected and desired behaviour under Ricci flow: the barbell graph, consisting of two cliques joined by a single edge is gradually contracted into two distinct points, making the clusters very well separated. As mentioned before, in general the positively curved edges are contracted by Ricci flow, and the negatively curved ones are extended. This allows us to expect edges inside tight communities to be contracted and bridge edges - ones between two more connected clusters - to be extended, as per their original curvatures.



(a) The original graph

(b) After 2 iterations of Ricci flow

(c) After 100 iterations of Ricci flow

Figure 4.3: Effects on a Barbell graph embedded with spectral embedding - the nodes belonging to cliques are initially embedded close by, effect of Ricci flow does not after the second iteration and does not move the bridge edge endpoints towards their clusters.

Similar, although not identical behaviour is observed when the spectral embedding is used, as shown in figure 4.3. Here the effects of Ricci flow appear limited, not changing whether 2 or 100 iterations have been applied. This could be due to the simple structure of the graph, as well as the initial performance of spectral embedding - the clusters are embedded to single points from the very start. A similar trend - results when using the spectral embedding being slightly less consistent than when embedding using MDS - persists in other graphs, as well as the numerically evaluated experiments described in the following sections and summarised in 4.2.6.

The observations suggest that Ricci flow and the information provided by curvature values is useful and relatively smoothly incorporated into at least some embeddings. However, the results obtained can have quite a high variance and not be entirely consistent when using different methods.

4.2.2 Experiment Design

Each of the experiments carried out for numerical evaluation of clustering performance on the embedded graphs followed the same general steps:

- Generate a graph with a known clustering assignment (with the types of graphs used described in 4.2.4).
- Embed the graph, using MDS or the spectral method (as described in later in section 4.2.4), into n-dimensional Euclidean space.
- Apply K-means clustering for a chosen value of K and evaluate its correspondence to the ground truth (the method of evaluation is described in 4.2.5).
- Apply Ricci flow, for a set number of iterations, to the original graph.
- Perform the same kind of embedding on the newly transformed graph.
- Apply K-means clustering with the same value of K as before and evaluate its correspondence to the ground truth.
- Evaluate the performance difference between the two clustering attempts.

4.2.3 Types of Graphs Examined

For interpretability of the experiments, a ground truth clustering allocation is desirable. Due to this, as well as runtime considerations, we chose to run the initial exploration on artificial graphs.

Types of graphs tested on and the relevant parameters used to specify them are summarized below:

- Gaussian partition graph. These graphs are specified by the number of nodes (n), the mean size of the clusters (c_{avg}) within the graph, a shape parameter (s), effectively controlling the variance of the cluster sizes, and probabilities for an edge (u, v) existing if the nodes u, v both belong to the same cluster (p_{in}) and if u, v are in different clusters (p_{out}) respectively. Generating these graphs with the NetworkX tool provides us with a ground truth clustering allocation.
- Lancichinetti–Fortunato–Radicchi (LFR) Benchmark Graph. Introduced by the titular authors in 2008 [20], this is another synthetic graph with a predetermined ground truth cluster (or community) structure. Here both the node degrees and the community sizes follow a power law distribution, with the respective exponents taken as parameters (τ_1 and τ_2 respectively) by the algorithm generating the graph. Other defining parameters are the number of nodes (n), the average degree of a node in the graph (d_{avg}), minimal size of a community (c_{min}), and a "mixing" parameter (μ), reflecting the average fraction of the nodes in a neighbourhood for a given node which do not belong to the same community as the one under consideration. When working with a specific framework, other parameters are available, however these are the main one which will be referred to. These graphs mimic a real-world network structure and account for the heterogeneous distribution of node degrees and community sizes.

4.2.4 Embeddings Used

The embeddings used in these experiments were multi-dimensional scaling (MDS) and spectral embedding. Both approaches are well studied and established, and as such were chosen as interpretable baseline methods.

MDS is applicable to any general kind of data with a measure of distance. We choose the length of the shortest path between the pair of nodes, weighted with the Ricci flow weights if present, to represent this measure. The method requires a dissimilarity matrix and attempts to optimize the placement of objects in euclidean space so as to preserve the pairwise dissimilarities, interpreted as the euclidean distances between embedded points.

More formally, this method minimizes the stress, which, given a dissimilarity matrix D is expressed as follows:

$$stress = \sqrt{\frac{\sum_{i<j}(d_{ij} - \hat{d}_{ij})^2}{\sum_{i<j} d_{ij}^2}} \tag{4.3}$$

Here d_{ij} denotes the (i, j)'th element of D and $\hat{d}_{ij}$ - the distance between the representations of i and j in the embedding. This is done by first converting the matrix D to an equivalent positive semi-definite one and performing PCA (principal component analysis) on the result.

The spectral method rests more on graph-specific properties, positioning the nodes according to the directions of the eigenvectors of the corresponding Laplacian matrix [3].

To perform a spectral embedding, first we compute the adjacency matrix A, corresponding to the input graph. Then we obtain its Laplacian, $L = D - A$ (where D is now the diagonal matrix holding the degree for each node) and calculate the partial eigenvalue decomposition for it. Finally, to embed into d dimensional space the eigenvectors corresponding to the d smallest (but non-zero) eigenvalues are used.

An important thing to note is that when using the spectral embedding the weights of the edges are interpreted as a measure of similarity, rather than distance. Thus edges with high weights will, generally, have their endpoints embedded close to each other and the ones with low weights will have endpoints placed further apart. However, the Ricci flow iteration weights are most naturally interpreted as distance. Due to this we need to transform the post-flow edge weights in order to apply the spectral embedding. Given the adjacency matrix $A \in \mathbb{R}^{n \times n}$, we compute the transformed matrix $A' \in \mathbb{R}^{n \times n}$ as follows:

$$w_{max} = \max\{A_{i,j} | 1 \leq i, j \leq n\} \tag{4.4}$$

$$A'_{i,j} = \begin{cases} 0 & \text{if } A_{i,j} = 0 \\ w_{max} + \varepsilon - A_{i,j} & \text{otherwise} \end{cases} \tag{4.5}$$

Here ε is a small value, ensuring the minimal similarity measure is still greater than zero. A linear transformation was chosen to preserve the scaling due to Ricci flow as well as possible

4.2.5 Clustering Evaluation Method

The method chosen for the evaluation of clustering correspondence to the ground truth cluster or community assignments is adjusted mutual information. It is a measure derived from mutual information, which, on a high level, describes the amount of information shared between the two cluster assignments. Adjusted mutual information not only normalizes the score, but also accounts for the similarity which would be obtained given a random clustering. In general, the score for mutual information tends to be higher as the number of clusters increases [44], thus the adjustment for chance allows for a more independent comparison than simply using the normalised measure would. A similar adjustment for chance is also present in another commonly used measure for performance of clustering compared to the ground truth assignment, Adjusted Rand Index. An AMI score of 1 indicates a perfect correspondence while the expected value of the AMI score for randomly assigned partitions is 0 (although the score itself can be negative).

4.2.6 Results

The results reflect a very similar trend to the barbell graph example presented in section 4.2.1. When the MDS embedding is used, Ricci flow consistently improves the performance in k-means clustering on the embedded graph, in terms of similarity to the ground truth (as measured by AMI). This is true both in Gaussian partition graphs and LFR networks. The spectral embedding results follow a similar pattern, although here an improvement is seen less consistently, with one experiment displaying an overall deterioration in performance instead.

Table 4.1: Setup of the experiments on Gaussian partition graphs

No.	Trials	n	c_{avg}	p_{in}	p_{out}	dim	n_{it}
1	30	50	10	0.6	0.2	2	30
2	30	50	10	0.6	0.3	2	30
3	30	80	10	0.6	0.2	2	30

Several experiments were conducted on both types of graphs, varying the parameters slightly. The setup for each experiment on Gaussian partition graphs is described in Table 4.1. Note that throughout these experiments the shape parameter s was held fixed at 10, encouraging a low variance on the cluster sizes. The embedding dimension is referred to as dim and the number of Ricci flow iterations as n_{it}, both in this table and in the subsequent ones.

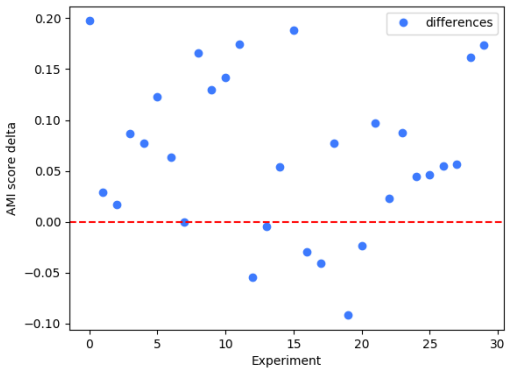
Table 4.2 presents the results of the respective Gaussian partition graph experiments. We can see here that, when using both the MDS and spectral embeddings, an improvement in the mean AMI score can be seen consistently. However, the magnitude of the improvement is quite small and the variance is relatively high, leading to values of the standard error of the mean being significant as well.

An example of the actual differences under the MDS embedding, obtained over the runs of experiment 3 can be seen in figure 4.4a. In fact, this particular experiment is

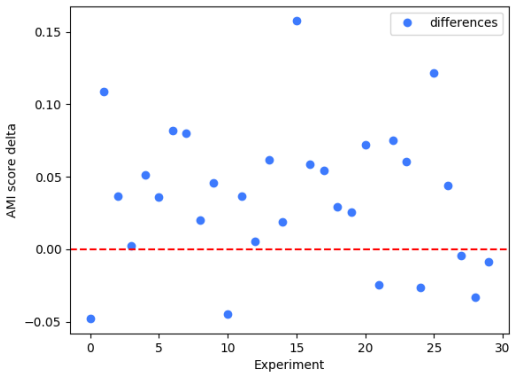
Table 4.2: Results of the experiments on Gaussian partition graphs. A consistent improvement in clustering performance after both of the embeddings is observed when graphs are embedded after Ricci flow.

No.	Embedding	Mean AMI pre-flow	Mean AMI post-flow	Mean Change	Std. Error on the Mean Change
1	MDS	0.4819	0.5069	0.0250	0.0240
	Spectral	0.4453	0.5304	0.0850	0.0115
2	MDS	0.2067	0.2527	0.0460	0.0164
	Spectral	0.2368	0.2676	0.0309	0.0094
3	MDS	0.2733	0.3408	0.0675	0.0139
	Spectral	0.2674	0.3162	0.0488	0.0089

one where the improvement was the most consistent, as can be seen in the plot and the lower standard error value. This is likely due to the graphs being slightly larger than the other ones tested, allowing the effects to become more pronounced.



(a) Gaussian Partition Graph Experiment 3, MDS Embedding



(b) LFR Graph Experiment 2, Spectral Embedding

Figure 4.4: Scatter plots showing, as points, the differences for each generated graph of the AMI value achieved after embedding the graph post Ricci flow and after embedding the original graph. It can be seen a positive effect is quite consistent and high positive values are much more common than negative ones.

Similarly as before, table 4.3 describes the setup of the experiments run on LFR networks, and table 4.4 summarises the results obtained in each of the experiments, with a sample scatter plot of the differences under the spectral embedding in experiment 2 shown in figure 4.4b. The overall trend here is very similar as before - a slight, but consistent improvement in the AMI score can be seen when the MDS embedding is used and Ricci flow applied, however the clustering performed on the spectral embedding of the network does not see a consistent improvement, with experiment 1 indicating deterioration in performance. The other two experiments, however, indicate that under appropriate conditions a consistent improvement can be seen.

In these experiments we make a conscious choice to embed into two dimensions - we

Table 4.3: Setup of the experiments on LFR networks

No.	Trials	n	τ_1	τ_2	μ	d_{avg}	c_{min}	dim	n_{it}
1	30	100	2	1.2	0.3	13	10	2	30
2	30	100	3	2	0.3	10	15	2	30
3	30	150	3	2	0.3	15	20	2	30

Table 4.4: Results of the experiments on LFR networks. A consistent improvement in clustering performance after MDS embedding is observed when graphs are embedded after Ricci flow. When the spectral embedding is used the improvements are also present, but less consistent.

No.	Embedding	Mean AMI pre-flow	Mean AMI post-flow	Mean Change	Std. Error on the Mean Change
1	MDS	0.1715	0.1978	0.0262	0.0071
	Spectral	0.2305	0.2139	-0.0167	0.0082
2	MDS	0.3802	0.4903	0.1101	0.0170
	Spectral	0.4030	0.4394	0.0364	0.0088
3	MDS	0.2832	0.3581	0.0749	0.0177
	Spectral	0.3665	0.3797	0.0132	0.0068

explore relatively small graphs, to examine a feasibility of such a method. Embedding into two dimensions has the advantage of both being easy to visualise and presenting a challenging task even for smaller graphs. Furthermore, relatively small graphs were chosen as benchmarks here in order to keep the computational cost of the experiments manageable - Ricci flow requires the recomputation of curvatures n_{it} times, which can be very time consuming on larger networks.

Overall, the results support the statement made when discussing the effects on a barbell graph in section 4.2.1 - Ricci flow can be useful when dealing with some embeddings, such as MDS or the spectral method, but we should not assume it would improve an arbitrary method consistently. This is evidenced by the slight inconsistency in the results when considering the spectral embedding of the graphs in question.

Chapter 5

Ricci Curvature in Random Walk Based Embedding

The initial explorations have shown that while Ricci flow can be useful for embeddings, it is not guaranteed to consistently improve any arbitrary method's performance, at least in clustering tasks. This information encourages us to seek further, more specific applications of both Ricci curvature and Ricci flow, moving to more complex embedding techniques which have the potential to use the information with more flexibility. In this chapter we focus on using Ricci curvature more directly. We propose a way to incorporate its values as part of the loss function in DeepWalk [31] and another method based on it, and evaluate the performance of clustering on graphs embedded using these methods.

5.1 Motivation and Hypothesis

We have see that using Ricci curvature and Ricci flow has the potential to improve embeddings for some tasks (with clustering correspondence to a ground truth assignment being used as the reference point in the initial explorations). Even though the results are not entirely conclusive, we can expect the extra information gained from computing curvature to be beneficial in some embedding methods, like MDS and the spectral embedding in the previous chapter.

One of the concerns when using Ricci flow directly is the computational efficiency - it requires repeated re-calculation of Ricci curvature and thus the repeated solving of many small optimization problems in order to compute the optimal transportation distances. This can significantly extend the time required for an embedding to be performed. For example, a single trial for the experiments described in the previous chapter would take several minutes to run, even though the graphs used are significantly smaller than most real world networks and the embeddings themselves take seconds.

However, we can emulate behaviour, intuitively similar to that of Ricci flow via an unconstrained minimization problem in the context of embeddings. We make a couple of assumptions, which can be considered consistent with the background on embeddings provided in section 3.2:

1. Given a graph $G = (V, E)$, for $u, v \in V$ and an embedding function $f : V \to \mathbb{R}^d$, the embedding is good if the distance $\|f(u) - f(v)\|_2$, corresponds to a measure of closeness or similarity of the nodes in the graph.
2. Nodes connected by positively curved edges are likely to belong to the same cluster or communities (backed up by the results in community detection using Ricci curvature [28, 42]).

Thus if we add a term corresponding to the post-embedding distance of nodes connected by positively curved edges to the cost function of a gradient based minimiser, we will encourage those nodes to be embedded close to each other. This effectively emulates the effect of the distance between the nodes in the graph being very low - something that Ricci flow would produce, due to the tendency to contract positively curved edges. As there is a correlation between positive curvature and nodes belonging to the same cluster, this would then tend to embed clusters compactly, while discouraging shortening distances between different clusters, an idea compatible with the clustering quality measures defined by Ackerman and Ben-David in their 2009 paper [4].

We hypothesise that such an amendment to an embedding should lead to an improvement in clustering performance.

5.2 Setup and Incorporation into the Loss Function

We take an approach to incorporating the Ricci curvature values inspired by the social network cost regularization term, proposed in the paper describing GEMSEC [36], which used normalised node neighbourhood overlap values. In order to see the reasoning behind the formulation, we first cover the algorithms the method is based on, as well as the loss functions used in them.

5.2.1 DeepWalk Loss Function

The embedding methods we ultimately consider are based on the DeepWalk algorithm [31], and use its loss function as a component of the final loss guiding the embedding procedure. This part is thus the first one we examine.

The defining property of DeepWalk is the random walk based sampling. To begin the embedding procedure, some fixed number of random walks of a given length are performed for each node (using a uniform distribution to choose which connection to use at each point). This creates a sample of sequences, S.

Again, let $f : V \to \mathbb{R}^d$ represent the embedding function. Note that given a sequence of nodes, we can find the window w in which a node v occurs - the set of nodes that appear within a number of steps of the random walk from v (either before it, or after). Then, given a sample of random walks, S, for any node v we can define $N_S(v)$ as the collection of such windows in the samples in S which contain v.

The goal of the embedding is to find the representation which most closely corresponds to the observed random walks. Equivalently, to achieve this, we wish to minimize the

negative log likelihood of the observed neighbourhoods $N_S(v)$ given the relevant embedding representations, $f(v)$. This objective is also given in the following equation:

$$\min_f \sum_{v \in V} -\log P(N_S(v)|f(v)) \tag{5.1}$$

The probability function is then considered to follow two properties, analogously as in the language processing applications [25] the method is adapted from. The first one is conditional independence with respect to the embeddings. With this property, we are able to factor the probability $P(N_S(v)|f(v))$ as a product of the probabilities for individual nodes, observed within the relevant windows.

$$P(N_S(v)|f(v)) = \prod_{n_i \in N_S(v)} P(n_i \in N_S(v)|f(v), f(n_i)) \tag{5.2}$$

The second property is symmetry - a pair of nodes should always have a symmetric effect on each others locations. To satisfy this, a specific function which exhibits the required property (softmax on dot products) for $P(n_i \in N_S(v)|f(v), f(n_i))$ is chosen.

$$P(n_i \in N_S(v)|f(v), f(n_i)) = \frac{\exp(f(n_i) \cdot f(v))}{\sum_{u \in V} \exp(f(u) \cdot f(v))} \tag{5.3}$$

Thus the final expression for the loss function, obtained by substituting the equations above into the objective specified in equation 5.1, is the following:

$$\mathcal{L}_{\mathcal{D}}(f) = \sum_{v \in V} \left[\log \left(\sum_{u \in V} \exp(f(u) \cdot f(v)) \right) - \sum_{n_i \in N_S(v)} \exp(f(n_i) \cdot f(v)) \right] \tag{5.4}$$

As mentioned before, the embedding objective is then to find representations for the nodes, which minimize this loss function, and can be done using standard minimization techniques, like gradient descent on neural networks.

5.2.2 GEMSEC Loss Function

The second paper our method draws from and later compares the results with is Graph embedding with self Clustering [36]. Here the embedding method, based on DeepWalk itself, focuses on the applications in clustering, incorporating a clustering cost in the loss function as a parametrized additive term. The proposed clustering cost is similar to the one optimized by the well known k-means algorithm. We seek, with a set of clusters denoted as C, to minimize the sum of the Euclidean distances between the nodes and the cluster center (μ_c) closest to them. To do this, the algorithm keeps track not only of the assigned coordinates for the nodes, but also those of the current cluster centers.

$$\mathcal{L}_C = \sum_{v \in V} \min_{c \in C} \|\mu_c - f(v)\|_2 \tag{5.5}$$

Thus the full loss function used by GEMSEC is

$$\begin{aligned}\mathcal{L}_G &= \mathcal{L}_D + \gamma \cdot \mathcal{L}_C \\ &= \sum_{v \in V} \left[\log \left(\sum_{u \in V} \exp(f(u) \cdot f(v)) \right) - \sum_{n_i \in N_S(v)} \exp(f(n_i) \cdot f(v)) \right] \\ &+ \gamma \cdot \sum_{v \in V} \min_{c \in C} \|\mu_c - f(v)\|_2\end{aligned} \tag{5.6}$$

This paper also proposes the use of a regularization term, representing the social network cost, as another additive term in the loss function. This is done to further encourage the embedding to preserve any community structure which may be present in the graph. In the paper, this cost is computed by iterating over the edges and computing the normalised neighbourhood overlap values. These are later used to scale the embedding distances for the relevant node pairs in the loss function, encouraging the edge endpoints with high neighbourhood overlap to be embedded close to each other.

5.2.3 Ricci Curvature Based Regularization

We suggest incorporating Ricci curvature values as a regularization term in the cost function of algorithms based on DeepWalk and GEMSEC.

Let r_e denote the Ricci curvature value for the edge e (equivalently, $r_{(u,v)}$ on the edge (u,v)). To incorporate these values into the cost function of a gradient based embedding algorithm, we also consider a transformation on the curvature values, $t_\alpha : \mathbb{R} \to [0,1]$, defined as $t_\alpha(x) = \frac{e^{\alpha x}}{e^{\alpha x}+1}$. This is done to both keep the weights non-negative and restrict their values to a bounded interval.

We seek to achieve a regularization effect, similar to that used for GEMSEC [36]. In order to do this, we define a regularization cost to be added to the embedding cost function:

$$\Lambda = \lambda \cdot \sum_{(u,v) \in E} t_\alpha(r_{(u,v)}) \cdot \|f(v) - f(u)\|_2$$

Here both λ and α are adjustable hyperparameters which let us configure the level of impact the curvature based cost has on the overall loss and the steepness of the sigmoidal function respectively.

This regularization term encourages behaviour empirically resembling Ricci flow. For edges of high curvature ($r_{(u,v)}$ close to 1), we will obtain a high value of $t(r_{(u,v)})$, and thus the distance between the embedding locations of the endpoints, $\|f(u) - f(v)\|_2$ is penalised, and encouraged to shrink.

We examine the effect of using this form of regularization with both the DeepWalk and the non-regularised GEMSEC cost functions. The former will be referred to as the DeepWalk-Ricci embedding, using the loss function specified in equation 5.7.

$$\begin{aligned}\mathcal{L}_R &= \mathcal{L}_D + \Lambda \\ &= \sum_{v \in V} \left[\log \left(\sum_{u \in V} \exp(f(u) \cdot f(v)) \right) - \sum_{n_i \in N_S(v)} \exp(f(n_i) \cdot f(v)) \right] \\ &+ \lambda \cdot \sum_{(u,v) \in E} t_\alpha(r_{(u,v)}) \cdot \|f(v) - f(u)\|_2\end{aligned} \tag{5.7}$$

The algorithm based on GEMSEC will be referred to as GEMSEC-Ricci and use the cost function specified in equation 5.8.

$$\begin{aligned}\mathcal{L}_R &= \mathcal{L}_D + \gamma \cdot \mathcal{L}_C \Lambda \\ &= \sum_{v \in V} \left[\log \left(\sum_{u \in V} \exp(f(u) \cdot f(v)) \right) - \sum_{n_i \in N_S(v)} \exp(f(n_i) \cdot f(v)) \right] \\ &+ \gamma \cdot \sum_{v \in V} \min_{c \in C} \|\mu_c - f(v)\|_2 + \lambda \cdot \sum_{(u,v) \in E} t_\alpha(r_{(u,v)}) \cdot \|f(v) - f(u)\|_2\end{aligned} \tag{5.8}$$

A comparison of the embeddings achieved with and without the use of this regularization term for DeepWalk can be seen in in figure 5.1. The most notable thing here is that the network cluster structure is much more defined when curvature based regularization is used, in line with the assumptions this term would encourage the inner edges of clusters to become short.

(a) Kamada-Kawai layout (b) Embedded using DeepWalk (c) Embedded using DeepWalk-Ricci

Figure 5.1: A Gaussian partition graph, as described in section 4.2.3, with 50 nodes, $p_{in} = 0.7$ and $p_{out} = 0.05$ embedded into two dimensions using different methods. Nodes are colored according to the ground truth cluster assignment.

5.3 Datasets Used for Evaluation

To evaluate the performance of the algorithms we used the datasets made available along with the GEMSEC paper. This decision was made due to both the accessibil-

ity of the data and the ease of comparison with the preexisting results, providing a way to validate the baseline performance as well as ensuring the correctness of the implementations.

The networks chosen are Facebook page networks - the nodes are different pages and the connections represent mutual likes among their pairs. The networks are distinguished by types and purpose of the pages. The types present are politicians, government entities, TV shows, athletes, companies and public figures.

The datasets have somewhat sparse edges, and represent a real world example of a social network, a structure where we could expect to find communities, clusters of more densely interconnected nodes. A brief summary of their properties is provided in the table below.

Table 5.1: Summary information for the datasets used for clustering evaluation.

Dataset	$\|V\|$	$\|E\|$	**Edge Density**	**Average Degree**
Athletes	13866	86858	0.00090	12.53
Company	14113	52310	0.00053	7.41
Government	7057	89455	0.00359	25.35
Politician	5908	41729	0.00239	14.13
Public Figure	11565	67114	0.00100	11.61
TV Show	3892	17262	0.00228	8.87

5.4 Evaluation of Clustering on Embedded Networks

We chose to evaluate the embedding obtained using our methods via the quality of the clustering obtained from the representation. However, in contrast to the clustering results presented in chapter 4, where the evaluation was based on a known ground truth partition, the datasets we are examining do not have pre-existing cluster assignments. Due to this, we require different means to evaluate them, since AMI is not usable. We have chosen to replace these values with the modularity of the clustering obtained.

The overall observations are as follows:

- Methods using Ricci curvature based regularization (DeepWalk-Ricci and GEMSEC-Ricci) consistently outperform their unregularized counterparts,
- Ricci curvature based regularization is more effective in improving clustering performance than regularization using normalised neighbourhood overlap, implying the local information provided by Ricci curvature has a stronger relation to the underlying cluster structure.

5.4.1 Modularity

Modularity values provide a way to evaluate a clustering without referring to a known ground truth partition of the nodes. High modularity values mean the network has

dense edges within the assigned clusters and the edges between clusters are sparse in comparison. It is computed by subtracting the expected number of edges within a given group, if the edges were distributed randomly, from the actual inner edge count of the group. In particular, if we allocate k clusters, $C_i \subseteq V$ for $i \in \{1,\dots,k\}$, we define

$$e_{ii} = \frac{|\{(u,v)|u,v \in C_i \text{ and } (u,v) \in E\}|}{|E|} \tag{5.9}$$

$$a_i = \frac{|\{(u,v)|u \in C_i \text{ and } (u,v) \in E\}|}{|E|} \tag{5.10}$$

Then e_{ii} represent the percentage of edges within the cluster i and a_i represents the percentage of edges with at least one endpoint in the cluster. Modularity is then computed as

$$M = \sum_{i=1}^{k} e_{ii} - a_i^2 \tag{5.11}$$

5.4.2 Implementation

When implementing the algorithms and our regularization method, we have used the code made available alongside the GEMSEC [36] paper as a starting point. This decision ensures that we are not introducing any inconsistencies in terms of the variations of DeepWalk and GEMSEC we refer to, and could immediately shift the focus to the novel parts of the algorithms proposed. The implementation expanded on the code available, adding the two algorithms we suggest, DeepWalk-Ricci and GEMSEC-Ricci, as well as various implementations relating to chapters 4 and 6, methods for testing and visualisations. Throughout we have also made use of the Python package GraphRicciCurvature, created and made available by the authors of some of the papers we cite [26, 28, 27].

The source code used is available together with this report, including the testing methods and the raw results obtained.

5.4.3 Experimental Results

Over the initial trials, most hyperparameters have been kept constant. Referring to the values mentioned in equations 5.8, 5.7, we use the following: $\lambda = 0.0625$, $\gamma = 0.1$, $\alpha = 4.0$. For training the embedding network we use the Adam optimizer [17], as a standard optimization method for gradient descent. The learning rate is initially set to 0.01 and allowed to decrease down to 0.001.

Table 5.2 provides a summary of the results obtained in terms of modularity values. The values represent the mean results over 10 runs of the given algorithms on each graph. Each graph has been embedded into 16 dimensions and the partitioned into 20 clusters - either by performing k-means clustering on the embedded data, or using the cluster centers found by GEMSEC based algorithms. DeepWalk and GEMSEC are used as baselines. The former due to the wide availability and application, the latter due to the fact it has been shown to outperform other common methods on these datasets

Table 5.2: Comparison of clustering modularity after different embeddings. Table contains mean values over 10 experiments on each graph, the numbers in parentheses correspond to standard deviation.

Algorithm \ Graph	Athletes	Company	Govt.	Politician	Public Figure	TV Show
DeepWalk	0.395 (±0.0135)	0.493 (±0.0121)	0.525 (±0.0260)	0.761 (±0.0347)	0.341 (±0.0180)	0.813 (±0.0145)
Regularized DeepWalk	0.489 (±0.0044)	0.555 (±0.0089)	0.669 (±0.0078)	0.827 (±0.0052)	0.455 (±0.0306)	0.841 (±0.0020)
DeepWalk-Ricci	0.584 (±0.0044)	0.606 (±0.0065)	0.682 (±0.0046)	0.841 (±0.0062)	0.585 (±0.0121)	0.837 (±0.0138)
GEMSEC	0.524 (±0.0113)	0.552 (±0.0092)	0.623 (±0.0253)	0.827 (±0.0152)	0.499 (±0.0136)	0.819 (±0.0055)
Regularized GEMSEC	0.586 (±0.0106)	0.606 (±0.0057)	0.673 (±0.0092)	0.847 (±0.0053)	0.588 (±0.0262)	0.839 (±0.0028)
GEMSEC-Ricci	0.639 (±0.0100)	0.644 (±0.0026)	0.690 (±0.0063)	0.855 (±0.0046)	0.623 (±0.0063)	0.842 (±0.0049)

as per the original paper [36]. We also include regularized DeepWalk and regularized GEMSEC. These algorithms use the neighbourhood overlap based regularization term described at the end of section 5.2.2. They are included to provide a comparison with a less computationally expensive regularization method.

In the table we can see that the GEMSEC-Ricci algorithm consistently outperforms the other methods, often by a reasonably large margin. The most significant part here is, however, the fact that GEMSEC-Ricci consistently outperforms the basic version of GEMSEC, and, similarly DeepWalk-Ricci outperforms the baseline version of DeepWalk. This implies that the Ricci regularization term has a positive impact on the clustering performance of real-world network embeddings.

The performance of Ricci curvature regularized methods also consistently exceeds that of the versions using neighbourhood overlap regularizers. This motivates the use of additional time taken for computation, and shows the information provided by curvature values has a substantial relative impact.

5.4.4 Regularization Coefficient Weighting Strategies

The results presented earlier have been obtained using the Ricci curvature computations based on Optimal Transport Distance (OTD), the problem outlined in 2.2. As pointed out before, this computation is responsible for most of the added runtime required.

An alternative could be using Average Transport Distance (ATD), as done in some previous work [26]. This would replace the optimisation problems with the simpler task of counting the neighbours and computing shortest paths between their pairs, greatly reducing the computational cost.

Another notable detail is that we only used the curvature values transformed via the stretched sigmoid function to lie in the interval $[0,1]$. However, we emphasised the parallel of this procedure to Ricci flow, which is somewhat incomplete. With the transformation applied, our cost function only encourages the endpoints of positively curved edges to be embedded close to each other, while having little to no effect on the endpoints of negatively curved edges.

We could consider using the raw edge curvature values as regularization weights. This would encourage the endpoints of negatively curved edges to be embedded further away from each other, by effectively rewarding this distance because of the negative contribution to the cost function. However, since we are using a regular cost function minimisation algorithm, this could lead to instability and require very careful tuning of the λ hyperparameter. In this case it would be difficult to ensure that these negative contributions do not dominate the cost function, allowing it to disregard the other costs (such as clustering cost and the observation log likelihood optimization).

Table 5.3 presents the mean modularity values obtained over 10 experiments for each of the graphs, this time using different ways to compute the weights in the regularization term. We compare three weighting alternatives - OTD values, transformed via t with $\alpha = 4.0$ as before, ATD values under the same transformation, and raw OTD values. The rest of the hyperparameters are kept the same as in the previously described experiments.

Table 5.3: Comparison of clustering modularity after embedding when using different curvature weighting strategies. The table contains mean values over 10 experiments, numbers in parentheses correspond to standard deviation.

Algorithm \ Graph	Athletes	Company	Govt.	Politician	Public Figure	TV Show
Transformed ATD DeepWalk-Ricci	0.517 (±0.0123)	0.575 (±0.0103)	0.672 (±0.0148)	0.834 (±0.0112)	0.526 (±0.0263)	0.839 (±0.0044)
Raw OTD DeepWalk-Ricci	0.040 (±0.0078)	0.154 (±0.0127)	0.436 (±0.0275)	0.766 (±0.0340)	0.068 (±0.0048)	0.819 (±0.0083)
Transformed OTD DeepWalk-Ricci	0.584 (±0.0044)	0.606 (±0.0065)	0.682 (±0.0046)	0.841 (±0.0062)	0.585 (± 0.0121)	0.837 (±0.0138)
Transformed ATD GEMSEC-Ricci	0.601 (±0.0092)	0.627 (±0.0089)	0.679 (±0.0087)	0.851 (±0.0025)	0.616 (±0.0088)	0.843 (±0.0029)
Raw OTD GEMSEC-Ricci	0.108 (±0.0127)	0.243 (±0.0165)	0.586 (±0.0253)	0.809 (±0.0065)	0.189 (±0.0166)	0.820 (±0.0043)
Transformed OTD GEMSEC-Ricci	0.639 (±0.0100)	0.644 (±0.0026)	0.690 (±0.0063)	0.855 (±0.0046)	0.623 (±0.0063)	0.842 (±0.0049)

The results here suggest that using the OTD measure of curvature can provide significantly better results on the clustering task. Even though the outcomes are still reasonably good, this trade-off could mean any advantage over the performance of non-regularised algorithms, or, more importantly, the algorithms regularized via the neighbourhood overlap values, is lost. As such the change from OTD to ATD could

not be made lightly and further proof it can remain beneficial would be needed. Due to this, we continue using the OTD values in this report.

The use of raw curvature values, including negative ones, as weights often has a strong negative effect on the modularity of the clustered graph. As this could be caused by an under-tuned λ hyperparameter (as per equations 5.7 and 5.8, the scaling factor for the contribution of Ricci curvature weights to the loss function), we present the results under different choices of λ in table 5.4 using DeepWalk-Ricci as the examined algorithm.

Table 5.4: Comparison of clustering modularity after embedding when using raw curvature values as weights for DeepWalk-Ricci. The table contains mean values over 10 experiments, numbers in parentheses correspond to standard deviation.

Value \ Graph	Athletes	Company	Govt.	Politician	Public Figure	TV Show
$\lambda = 0$ (DeepWalk)	0.395 (±0.0135)	0.493 (±0.0121)	0.525 (±0.0260)	0.761 (±0.0347)	0.341 (±0.0180)	0.813 (±0.0145)
$\lambda = 0.03125$	0.166 (±0.0097)	0.313 (±0.0164)	0.517 (±0.0234)	0.768 (±0.0276)	0.195 (±0.0093)	0.822 (±0.0067)
$\lambda = 0.0625$	0.040 (±0.0078)	0.154 (±0.0127)	0.436 (±0.0275)	0.766 (±0.0340)	0.068 (±0.0048)	0.819 (±0.0083)
$\lambda = 0.125$	-0.018 (±0.0023)	0.050 (±0.0099)	0.379 (±0.0232)	0.706 (±0.0493)	0.017 (±0.0052)	0.823 (±0.00343)
$\lambda = 0.25$	-0.031 (±0.0016)	0.019 (±0.0068)	0.259 (±0.0215)	0.603 (±0.0229)	0.005 (±0.0030)	0.814 (±0.0078)

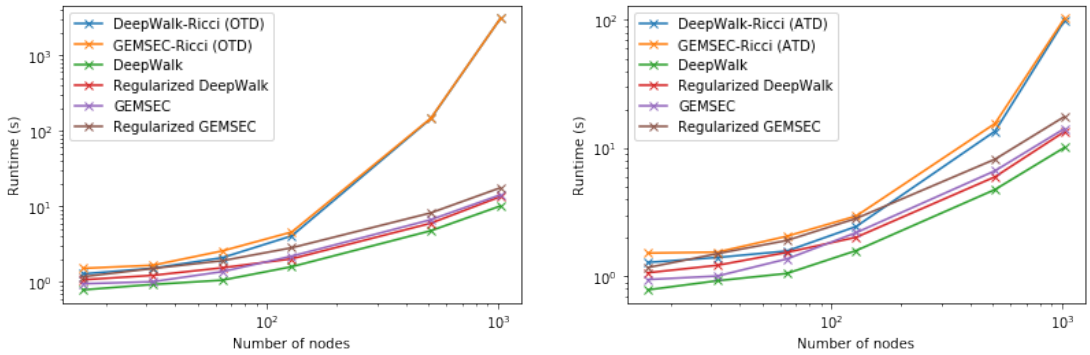
The results show that there is almost no observable consistent improvement over the use of DeepWalk alone when we choose to use raw Ricci curvature values as weights, no matter what λ value is chosen. In fact, in 4 out of the 6 networks we tested on, only a deterioration in modularity has been observed. This suggests using transformed values is much more suitable in the context of random walk based algorithms like DeepWalk and GEMSEC.

5.4.5 Computational Cost

Although the results presented suggest an improvement in performance on post-embedding clustering tasks, the Ricci curvature regularization weights are somewhat expensive to compute. The main cost comes from the initial computation of curvature values for each edge in the graph. This, in its original form, requires solving an optimization problem, typically phrased in terms of linear programming, for each edge. While this is a polynomial-time task, it is still non-trivial and can require quite a lot of resources. It does, however, greatly improve on the computation time which would be required if we wished to apply Ricci flow to achieve the similar improvement in clustering we described in chapter 4, as this requires many repeated calculations of curvature.

In this case, for each embedding, curvature computation is a one-time cost. The design of the regularizing term only requires the values to be computed once, thus the impact

of the computational cost is spread out if many iterations of the embedding procedure are performed.



(a) Ricci curvature computed with OTD

(b) Ricci curvature computed with ATD

Figure 5.2: Running time comparison for the embedding algorithms examined, on Erdos-Renyi graphs with $p = 0.1$. Both x and y axes use logarithmic scaling.

The graph in figure 5.2 shows the running time of the proposed embedding procedure on progressively larger Erdos-Renyi graphs. The Erdos-Renyi model refers to random graphs with a specified number of nodes, n, and a probability p of an edge existing between any given pair of nodes. For these experiments the value of $p = 0.1$ was kept constant, while connected graphs with progressively larger numbers of nodes were generated. The set-up was chosen so that the expected edge density remains constant.

In these experiments the use of Ricci curvature, especially with OTD computations, represents a significant increase in runtime. However, a 10% edge density is considerably higher than the density observed in many real world networks.

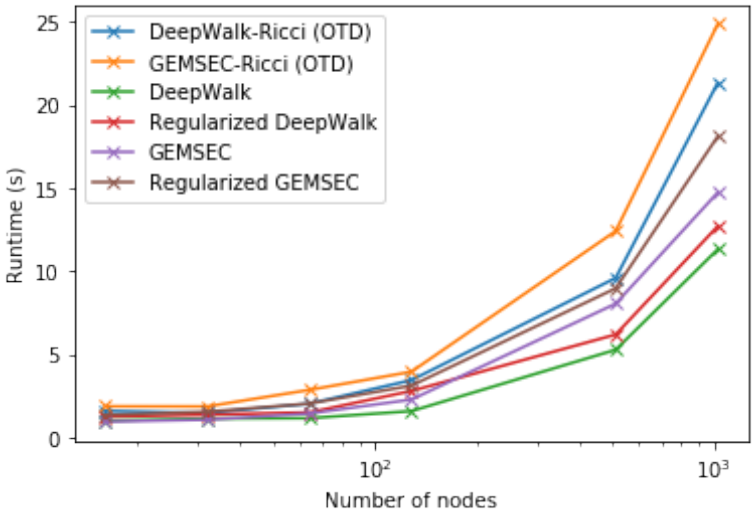


Figure 5.3: Running time comparison for the embedding algorithms examined on LFR networks with constant parameters and varied number of nodes. Only the x-axis uses logarithmic scaling.

For comparison, we also evaluate the running time on LFR networks (as described in 4.2.3, here we set the average degree to 4, maximum degree to 10, $\tau_1 = 2$, $\tau_2 = 1.5$, $\mu = 0.3$ and ensure the graph is connected). While this creates rather sparse graphs, they serve as a slightly more accurate representation of networks in real datasets. These runtimes are visualised in graph 5.3. Note we no longer need to use logarithmic scaling on the vertical axis, as even when the computation time for Ricci curvature using OTD is included, DeepWalk-Ricci and GEMSEC-Ricci are only slightly slower than their counterpart regularized via community overlap.

The runtime analysis on LFR networks suggests that on some types of sparser networks the embedding methods which utilize Ricci curvature do not result in extensive computational overhead. Thus these methods remain feasible, even when using optimal transportation distance values. However, in denser networks it may be necessary to use average transportation distance to keep the running time reasonably efficient.

Chapter 6

Ricci Curvature in Embedding and Clustering via Matrix Factorization

The results obtained in chapter 4 showed that Ricci flow (and by extension the information provided by Ricci curvature) had a positive impact when clustering graphs embedded using MDS or spectral embedding. One notable thing about both of the methods, is that they based on finding a matrix factorization, whether for a transformed distance matrix or the Laplacian of the graph. In addition to this, the skipgram network based models like DeepWalk, which we examined and adapted in chapter 5, have also been shown to be equivalent to implicit matrix factorization [22, 33].

These similarities motivate further exploration of using Ricci curvature in combination with matrix factorization methods. This chapter presents the investigation of using Ricci curvature values in non-negative matrix factorization and its applications to clustering. A consistent and significant improvement in clustering performance, as measured by modularity, is seen when compared to simply factorising the adjacency matrix for this purpose.

6.1 Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) [21] refers to a matrix factorization method which, given a matrix $M \in \mathbb{R}^{n\times m}$ with non-negative elements, finds an approximate factorization into two matrices, $W \in \mathbb{R}^{n\times k}$ and $H \in \mathbb{R}^{k\times m}$, while maintaining the non-negativity constraint. As with other similar methods, k is usually chosen to be significantly smaller than n or m, thus producing a representation for the input matrix in a space with comparatively low dimensionality.

The algorithm minimizes the Froebenius norm between the original matrix and its reconstruction WH. Thus the optimization problem solved is the following:

$$\min_{W,H\geq 0} \|M - WH\|_F^2 = \sum_{i,j} (M - WH)_{ij}^2 \tag{6.1}$$

Several variations of NMF exist [19, 16], most imposing additional constraints, such as sparsity on the factorization matrix H [16], however in this report we focus on

the basic variation, seeking to determine whether the use of Ricci curvature yields an improvement, rather than create new cutting-edge approaches.

6.2 Relation to Clustering

Although such factorisation can be seen as an embedding of the original matrix in the k-dimensional space, there is an established explicit relation of NMF to clustering the columns of the original matrix [16, 43].

NMF, given a matrix M, computes $M \approx WH$. To determine the clustering assignment we can interpret the columns of $W \in \mathbb{R}^{n \times k}$ as the cluster centers, with the assignment for node i determined by taking the i'th column of H, finding the largest element and assigning the node to the cluster which has the center (column in W) corresponding to the position of this value in its column. An intuitive way this can be seen is presented by thinking about the approximation of M via WH - each column of M will be a linear combination of the columns of W, scaled by the coefficients in the relevant column of H. The highest relevant value in H will then indicate the most signifficant contribution to the reconstruction.

The symmetric version of this method has been shown to be equivalent to a relaxation of k-means clustering (kernel k-means) [10]. Other parallels between NMF and k-means have also been shown, for example Kim and Park show in their report [16] that imposing certain restrictions on NFM can lead it to behave like k-means clustering. However, overall experimental results imply NMF is a more flexible method, less prone to getting stuck in local minima.

6.3 Experimental Evaluation

The goal of this exploration was to establish whether Ricci curvature alone would provide enough information to obtain considerable results in clustering via NMF. In order to test this we have chosen to compare the results obtained when factorizing a matrix of Ricci curvature values to the one observed when only the adjacency matrix is supplied - the most basic matrix representation of a graph. A significant and consistent improvement in the clustering modularity can be seen, differing from the baseline by up to 35%. A further comparison of these values to the ones obtained in chapter 5 shows that the restriction to the use of Ricci curvature information exclusively allows us to achieve results comparable to ones obtained via well established methods, such as DeepWalk.

6.3.1 Setup and Matrix Construction

To construct a non-negative matrix M using Ricci curvature values we employ a strategy equivalent to the computation of curvature based weights, described earlier in section 5.2.3. Given a value $\alpha > 0$ and an undirected graph $G = (V, E)$, we compute the following (again referring to the raw Ricci curvature value for an edge (u, v) as $r_{(u,v)}$):

$$t_\alpha(x) = \frac{e^{\alpha x}}{e^{\alpha x} + 1} \tag{6.2}$$

$$M_{u,v} = \begin{cases} 0 & \text{if } (u,v) \notin E \\ t_\alpha(r_{(u,v)}) & \text{otherwise} \end{cases} \tag{6.3}$$

The transformation hyperparameter α regulates how much we exaggerate the differences between curvature values - they is effectively scaled by this factor when the values are passed to the sigmoidal t function. A higher value of α will effectively lead to more extreme values (closer to 0 or 1 than 0.5) being used in the matrix.

6.3.2 Results

The results obtained when testing clustering via non-negative matrix factorization are presented in table 6.1. We have chosen to use the same datasets that the random walk based methods were evaluated on, with the description found in section 5.3 - a decision allowing us to compare the results obtained directly. Again, experiments on each graph were repeated 10 times, with mean modularity values presented in the table, alongside their standard deviations.

Table 6.1: Comparison of clustering modularity after when using non-negative matrix factorization. Table contains mean values over 10 experiments on each graph, the numbers in parentheses correspond to standard deviation.

Matrix \ Graph	Athletes	Company	Govt.	Politician	Public Figure	TV Show
Adjacency	0.537 (±0.0006)	0.416 (±0.0015)	0.531 (±0.0032)	0.693 (±0.0026)	0.278 (±0.0142)	0.666 (±0.0048)
Ricci curvature	0.540 (±0.0006)	0.533 (±0.0003)	0.613 (±0.0007)	0.727 (±0.0001)	0.372 (±0.0017)	0.754 (±0.0000)

The first thing to note here is the consistency in improvement - considering more extensive local information than just the presence of an edge, provided by the Ricci curvature values, improves the clustering performance on every graph tested. However, the magnitude of this effect varies significantly. Another notable thing is that using the Ricci curvature matrix results in much lower standard deviation of the clustering modularity values. This implies that the factorization obtained is quite consistent throughout different trials, or at least the clustering assignment quality does not vary greatly. This could be a desirable property for certain applications, increasing the certainty we have about the algorithm's behaviour.

Although these results are still slightly inferior to those obtained with DeepWalk-Ricci and GEMSEC-Ricci, they are similar to and sometimes exceed the baseline DeepWalk and GEMSEC version performance. There are two main differentiating factors here: NMF is not treated as an intermediate embedding, but rather a direct clustering methods, and the method using Ricci curvature directly, as the only information available (alongside the edge structure, implicitly represented by the placement of zero elements in the matrix).

6.3.3 Impact of Different α Values

The main curvature related hyperparameter in this method is α. The initial experiments were performed with $\alpha = 4$, however changing this value could have an impact on the clustering performance, as it directly affects how the original matrix is constructed. To examine this, we perform further experiments, evaluating the results when different values of $\alpha > 0$ are used. The values are maintained positive, as we want to maintain the effect of large curvature values resulting in matrix elements close to 1 and negative values being mapped to numbers close to zero. The results of these experiments are presented in table 6.2.

Table 6.2: Comparison of clustering modularity after when using non-negative matrix factorization with varied α values in t_α. Table contains mean values over 10 experiments on each graph, the numbers in parentheses correspond to standard deviation.

Value \ Graph	Athletes	Company	Govt.	Politician	Public Figure	TV Show
$\alpha = 1$	0.506 ($\pm$0.0029)	0.493 ($\pm$0.0002)	0.603 ($\pm$0.0007)	0.770 ($\pm$0.0000)	0.310 ($\pm$0.0017)	0.752 ($\pm$0.0000)
$\alpha = 4$	0.540 ($\pm$0.0006)	0.533 ($\pm$0.0003)	0.613 ($\pm$0.0007)	0.727 ($\pm$0.0001)	0.372 ($\pm$0.0017)	0.754 ($\pm$0.0000)
$\alpha = 16$	0.545 ($\pm$0.0013)	0.562 ($\pm$0.0024)	0.656 ($\pm$0.0015)	0.720 ($\pm$0.0133)	0.378 ($\pm$0.0032)	0.772 ($\pm$0.0032)
$\alpha = 64$	0.567 ($\pm$0.0183)	0.591 ($\pm$0.0063)	0.658 ($\pm$0.0015)	0.716 ($\pm$0.0099)	0.409 ($\pm$0.0041)	0.786 ($\pm$0.0081)
$\alpha = 256$	0.546 ($\pm$0.0265)	0.576 ($\pm$0.0237)	0.667 ($\pm$0.0021)	0.673 ($\pm$0.0064)	0.419 ($\pm$0.0161)	0.793 ($\pm$0.0070)

Here we observe that higher values of α often lead to a slight increase in performance, with the majority of the graphs having the best recorded performance when tested with $\alpha \geq 16$. A potentially detrimental effect of using large values is an increase in the variability of performance. This can be seen especially clearly in the standard deviation values for the clustering modularity of the Athletes graph. This is likely due to an exaggeration of the differences in close but not equal curvature values, as noted in section 6.3.1, causing overall distortion. In addition to this, we can also note that extremely high values of α can lead to a drop in performance, which can be explaine by the same argument.

Chapter 7

Conclusions and Future Work

In this chapter we discuss potential further work and options to build on the progress made so far. One of the main areas we propose to explore is the impact of Ricci curvature based regularization on classification performance - there has been some, although limited, progress made already and a promising direction for future inquiry has been identified.

We also summarise the overall contributions and observations presented in this report, providing some concluding remarks.

7.1 Proposed Extensions

Given the limited time and the open-ended nature of the project, there are quite a few natural extensions which could be added to the work presented here. Part of the progress made so far is also briefly summarised here, so as to motivate one of the extensions which was partially implemented and explored alongside the other experiments described in earlier chapters.

7.1.1 Impact of Ricci Curvature Based Regularization on Classification Tasks

Prediction tasks are one of the key applications of network embeddings. We have carried out partial examination of the impact Ricci curvature based regularization has on classification, with limited improvements observed. The testing so far has been limited to a single dataset, due to unforeseen impact of timing constraints. However, we have identified classification in datasets with few known labels as a promising area where the inclusion of curvature values do make a difference.

7.1.1.1 Initial Progress and Results

For embedding evaluation based on classification we use the CORA dataset[1] [41]. The nodes here are different machine learning research papers, with the connections

[1] Obtained from the network repository [34]

representing citations. Each node is assigned one of 7 categories, specifying which sub-field of machine learning it focuses on. The dataset represents a connected graph which has 2708 nodes and 5429 edges, with an average degree of 4.

To carry out classification itself we use logistic regression. In particular, the labels are encoded as one-hot vectors, a softmax activation and a cross-entropy loss functions are used. The training is performed with a gradient descent optimizer. To compute testing accuracy we compare the *argmax* of the softmax output for a query node embedding coordinates with the one-hot ground truth encoding, considering the prediction to be correct if and only if the label assigned the highest probability by the softmax function matches the location of the non-zero entry in the target encoding,

Logistic regression was chosen as a simple and well-studied classification model. The goal of this investigation is not to obtain the highest possible prediction accuracy, but rather to investigate the effect of using the Ricci curvature based normalization term. We wish to focus on the relative performance on, for example, embeddings obtained with DeepWalk and DeepWalk-Ricci or GEMSEC and GEMSEC-Ricci.

The initial experiments used the majority of the dataset available (80%) for training, with the rest comprising the testing set. No validation set was used as we did not perform extensive hyperparameter tuning. The results obtained were mostly uniform for all the embedding methods used in chapter 5. In particular, the mean classification accuracy values over 10 independent trials, obtained when using DeepWalk, regularized DeepWalk and DeepWalk-Ricci respectively, are 67.84%, 67.56% and 67.38%. Similar results were observed for the GEMSEC variations. These results do not allow us to draw any conclusions on the impact of incorporating Ricci curvature values outright, as they provide very little distinction between any of the methods.

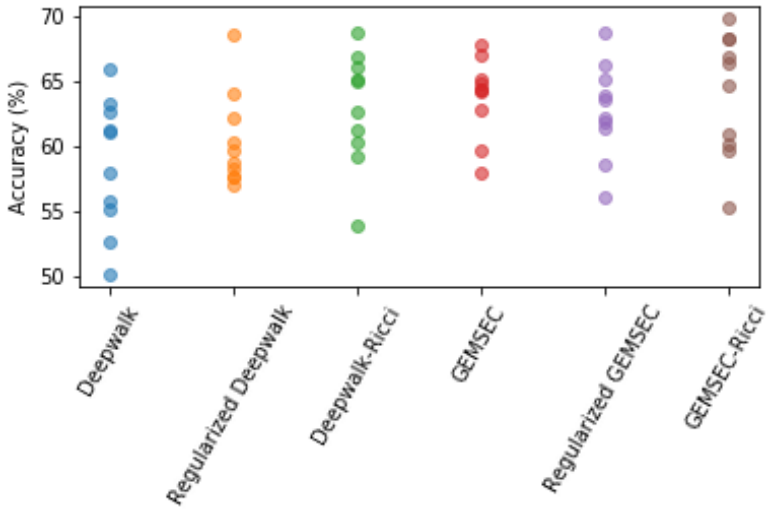


Figure 7.1: Classification accuracy on the CORA dataset, with the training set comprised of 5% of the dataset and testing performed on 20% of the available data. Each dot represents the outcome of a single experiment using the corresponding embedding method.

However, a slight distinction between the methods can be seen when only a small part of the network is used for training. An example is illustrated in figure 7.1. Here only

5% of the data was used for training (selected randomly at the start of each experiment), and the testing was performed on a subset of the same size as previously (20% of the dataset's original nodes). As expected, the reduction in the size of the training sets causes some deterioration in overall accuracy, however in addition to this we can also observe that the Ricci curvature regularized versions of the algorithms perform better than their baseline counterparts. In particular, the mean results for DeepWalk and DeepWalk-Ricci here are 58.56% and 62.87% respectively.

7.1.1.2 Node Classification in Networks with Few Known Labels

The results obtained so far suggest that methods which incorporate Ricci curvature could aid in classification tasks where the labels of only a small subset of nodes are known. Thus we propose this area as one of the main parts of further work.

Ricci curvature provides insight into not only the overall shape of the network, but also on the neighbourhood a specific edge lies in. Thus, given a network with a small set of labelled nodes, and the task of predicting the unknown labels, we wish to find the best way to utilize this information. An idea which could be expanded upon is changing the edge weights in the graph which will be used for the curvature computation so as to highlight the connections to the known labelled nodes. Intuitively, if, for example, the graph has an overall cluster structure correlated to the label assignments, we would like to put extra emphasis on the edges near a labelled node which are also in the same cluster. In connection to Ricci curvature, we would like to make the curvatures near the known label assignments more extreme - if there is a positively curved edge we would like the value to grow, and become even lower for negatively curved edges.

The challenge, which should be tackled in future work, is creating an efficient weight adjustment strategy which would allow us to achieve this, as well as examining how it impacts the classification performance in methods incorporating Ricci curvature.

7.1.2 Other Potential Extensions

In chapter 5 we show that adding the Ricci curvature regularization term to the loss function of the original algorithms improves the clustering performance of both DeepWalk and GEMSEC. This strongly suggests similar results could be seen with other random walk based embedding methods (such as Node2Vec [13] or Walklets [32]), as well as ones based on deep learning. A further investigation into the magnitude of the improvement achievable with methods not explored in this report, as well as the trade-off in computational cost faced after including this term could be investigated.

Another potential direction of future work could be investigating similar embeddings into non-Euclidean spaces, as suggested in some earlier research [27]. This could eliminate the need for Ricci flow-like transformations by matching the topology of the target vector space to that of the network in question. It would, however, raise new questions - for example, how to determine the overall topology and shape of the graph, since the discrete Ricci curvature is a highly localised measure.

Lastly, while we have chosen to focus on the definition of discrete Ricci curvature as presented by Ollivier [29], in line with most pre-existing work, this is not the only

known way to compute curvature on graphs. The concepts and methods presented in this report could be redefined to use, for example, the notion of discrete Ricci curvature proposed by Forman [11]. Examining this has the potential to provide insight on whether the results observed so far are tied to the use of Ollivier-Ricci curvature specifically, or inherent to using a reasonable interpretation of the local shape of the graph. The use of Forman-Ricci curvature could also aid in countering some of the computational overhead caused by the use of curvature in embeddings and clustering, as it is more efficient to compute on large real world networks than Ollivier-Ricci curvature, as noted in a paper by Samal et al. comparing the two methods [37].

7.2 Critical Discussion

In this report we have shown that the use of Ricci flow and Ricci curvature in network embeddings (as well as more direct methods, like NMF) can improve the performance in clustering tasks.

Although the results are encouraging, it is worth noting that we only present empirical evidence. The methods we apply are based on mathematical insights into what a neighbourhood around a positively or negatively curved edge is likely to look like and how it fits in to the broader cluster structure of the graph. However, these do not constitute a proof, and the project explored the applicability of these insights. Thus we are unable to provide specific guarantees or theoretical bounds on the effects observed when Ricci curvature is included in embedding cost. A similar point can be made about the initial method we used, applying Ricci flow before performing an embedding. This is further complicated by the variety of embedding techniques we choose to apply, as any theoretical proof would have to depend on the specifics of the methods.

However, the reliance on empirical evidence does not make our work stand out among the pre-existing work on the applications of Ricci curvature in complex networks. A similar pattern can be observed in various papers [26, 27, 42]. The notable exception here is the work by Ni et al. [28] exploring community detection via Ricci flow, where the authors provide mathematical proof the community detection method is successful on a specific type of graphs. This result encourages further exploration of the theoretical implications of Ricci curvature and suggests it could be possible to provide guarantees for some of our methods. In particular, given more time we could look into a possibility of a similar proof regarding the initial method of applying Ricci flow to improve the embeddings and clustering after.

In terms of evaluation, we have chosen to focus on two measures of clustering performance. In chapter 4 we used AMI to compare the clustering generated with a ground truth assignment. In chapters 5 and 6, to evaluate clustering without knowing a ground truth assignment, we employed modularity. The measures were chosen due to being commonly used and providing inter-comparability with other work. However, it should be noted that other measures for clustering exist (such as Adjusted Rand Index for comparison with known values, clustering purity and entropy). Evaluating the clustering performance using a larger variety of methods could lead to further insights on the nature of the improvements observed. It would also make the argument for claiming the improvements are observed consistently more robust. In addition to this, as pointed

out in section 7.1, there are other ways to evaluate embeddings, such as classification or link prediction performance. Focusing on these methods for evaluation could lead to slightly different results and conclusions, although some improvement could be expected due to the relation of these tasks to clustering. If additional exploration of these applications (as per the proposed work on classification in section 7.1) yielded positive results, we could claim embedding quality is improved in a well-rounded way, not only in the perspective of clustering.

The Ricci curvature based regularization term, proposed in chapter 5 can be added to any embedding algorithm which relies on the minimization of some specified loss function. We might expect this to be a relatively universal term, as the effects of its addition, when scaled appropriately, should be similar. However this cannot be concluded solely from the results we have obtained, as we only explored a few select embedding methods. This could be further tested with a wider variety of embedding methods, beyond DeepWalk and GEMSEC, which have been our focus in this report. In addition to this, the computational cost of computing Ricci curvature values, even without performing the iterations of Ricci flow, is still often orders of magnitude higher than that of standard embeddings, rendering the method to be of limited use for dense or large graphs. Use of ATD (the average transport distance) instead of OTD (the optimal transport distance) in calculations could mitigate this effect. However, as pointed out in section 5.4.4, the benefit observed when the former is used is lower than with OTD, and thus further exploration is required in order to ensure the proposed methods are efficient and beneficial in practice.

Lastly, although we have explored several ways to compute and incorporate Ricci curvature, we may not have found the best transformation to use for either the regularization term weights or NMF matrix entries. The sigmoidal function used in chapters 5 and 6 will encourage the endpoints of strongly positively curved edges to be embedded near each other. A closer emulation of the behaviour of Ricci flow might be desirable here, requiring endpoints of negatively curved edges to be pushed further away. We have shown that the direct use of curvature values (with the identity function in place of the transformation) does not improve the performance, however a different function yielding better results might still be found.

7.3 Conclusion

The overall results and findings presented in this report have shown that the discrete Ollivier-Ricci curvature can be a useful tool when considering graph embedding and clustering. It provides insights into the shape and cluster structure of the graph, although the improvements often come with a high computational cost.

The explorations of the impact Ricci flow has on embeddings and clustering performed afterwards provide new insight as to how the procedure affects the graph structure. We have shown that the changes observed under simple matrix factorization based embedding methods are consistent with the established understanding that Ricci flow contracts positively curved edges which tend to lie inside clusters, and extends negatively curved bridge edges. It was also shown that this has a consistent positive effect on

clustering results, when comparing to ground truth assignments on various synthetic graphs.

Although Ricci flow was shown to be effective in improving post-embedding clustering performance, the computational cost and time required to perform this procedure is often very high. This is due to the need to recompute Ricci curvature values in every iteration, solving many LP problems each time. With this in mind, we have shown how to achieve similar improvements in embedding algorithms based on random walk sampling and general loss function minimization, while avoiding the direct use of Ricci flow. We proposed a novel way to regularize the loss functions of these algorithms, incorporating Ricci curvature values directly. The addition of this regularization term to the DeepWalk and GEMSEC algorithms has proven to consistently increase the quality of the clustering on the embedded graphs, as measured by modularity. This allows us to improve well established algorithms while requiring less computational overhead than applying Ricci flow would. Although we now need to modify the internal cost function of the embedding at hand, rather than keeping the curvature use and embedding separate. The addition of this regularization term encourages behaviour somewhat similar to Ricci flow - endpoints of positively curved edges are encouraged to be embedded close to each other, with endpoints of negatively curved ones allowed to remain far without penalty.

Lastly, we have noted that most of the methods where improvements were observed can be seen as variations of matrix factorization. This encouraged further exploration of methods based on this framework and their applications. While investigating these we have shown that when Ricci curvature is used in clustering based on non-negative matrix factorization, the performance consistently improves by a significant margin.

Even though this work is not an exhaustive summary of the relation Ricci curvature has to embeddings and clustering, with a lot of potential extensions and questions open for future work, we have shown that incorporating the discrete Ricci curvature or Ricci flow when performing these tasks can lead to noticeable and consistent improvements.

Bibliography

[1] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48, 2013.

[2] Dominique Bakry and Michel Émery. Diffusions hypercontractives. In *Séminaire de Probabilités XIX 1983/84*, pages 177–206. Springer, 1985.

[3] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, June 2003.

[4] Shai Ben-David and Margareta Ackerman. Measures of clustering quality: A working set of axioms for clustering. In *Advances in neural information processing systems*, pages 121–128, 2009.

[5] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, page 891–900, New York, NY, USA, 2015. Association for Computing Machinery.

[6] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[7] Bennett Chow and Feng Luo. Combinatorial ricci flows on surfaces. *Journal of Differential Geometry*, 63(1):97–129, Jan 2003.

[8] Fan RK Chung and S-T Yau. Logarithmic harnack inequalities. *Mathematical Research Letters*, 3(6):793–812, 1996.

[9] David Cushing, Riikka Kangaslampi, Valtteri Lipiäinen, Shiping Liu, and George W Stagg. The graph curvature calculator and the curvatures of cubic graphs. *Experimental Mathematics*, pages 1–13, 2019.

[10] Chris Ding, Xiaofeng He, and Horst D Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 606–610. SIAM, 2005.

[11] Robin Forman. Bochner's method for cell complexes and combinatorial ricci curvature. *Discrete and Computational Geometry*, 29(3):323–374, 2003.

[12] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, Jul 2018.

[13] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[14] Richard S. Hamilton. Three-manifolds with positive ricci curvature. *J. Differential Geom.*, 17(2):255–306, 1982.

[15] M. Jin, J. Kim, F. Luo, and X. Gu. Discrete surface ricci flow. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1030–1043, 2008.

[16] Jingu Kim and Haesun Park. Sparse nonnegative matrix factorization for clustering. Technical report, Georgia Institute of Technology, 2008.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[19] Da Kuang, Chris Ding, and Haesun Park. Symmetric nonnegative matrix factorization for graph clustering. In *Proceedings of the 2012 SIAM international conference on data mining*, pages 106–117. SIAM, 2012.

[20] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4), Oct 2008.

[21] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[22] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[23] Yong Lin, Linyuan Lu, and Shing-Tung Yau. Ricci curvature of graphs. *Tohoku Mathematical Journal, Second Series*, 63(4):605–627, 2011.

[24] John Lott and Cédric Villani. Ricci curvature for metric-measure spaces via optimal transport. *Annals of Mathematics*, pages 903–991, 2009.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[26] Chien-Chun Ni, Yu-Yao Lin, Jie Gao, and Xianfeng Gu. Network alignment by discrete ollivier-ricci flow. *Graph Drawing and Network Visualization*, page 447–462, 2018.

[27] Chien-Chun Ni, Yu-Yao Lin, Jie Gao, Xianfeng David Gu, and Emil Saucan. Ricci curvature of the internet topology. *CoRR*, abs/1501.04138, 2015.

[28] Chien-Chun Ni, Yu-Yao Lin, Feng Luo, and Jie Gao. Community detection on networks with ricci flow. *Scientific Reports*, 9(1):9984, 2019.

[29] Yann Ollivier. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 256(3):810 – 864, 2009.

[30] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM*

SIGKDD international conference on Knowledge discovery and data mining, pages 1105–1114, 2016.

[31] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 701–710, New York, NY, USA, 2014. Association for Computing Machinery.

[32] Bryan Perozzi, Vivek Kulkarni, and Steven Skiena. Walklets: Multiscale graph embeddings for interpretable network classification. *arXiv preprint arXiv:1605.02115*, 2016.

[33] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 459–467, 2018.

[34] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[35] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[36] Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 65–72, 2019.

[37] Areejit Samal, RP Sreejith, Jiao Gu, Shiping Liu, Emil Saucan, and Jürgen Jost. Comparative analysis of two discretizations of ricci curvature for complex networks. *Scientific reports*, 8(1):1–16, 2018.

[38] Romeil Sandhu, Tryphon Georgiou, Ed Reznik, Liangjia Zhu, Ivan Kolesov, Yasin Senbabaoglu, and Allen Tannenbaum. Graph curvature for differentiating cancer networks. *Scientific reports*, 5(1):1–13, 2015.

[39] Romeil S Sandhu, Tryphon T Georgiou, and Allen R Tannenbaum. Ricci curvature: An economic indicator for market fragility and systemic risk. *Science advances*, 2(5):e1501495, 2016.

[40] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[41] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[42] Jayson Sia, Edmond Jonckheere, and Paul Bogdan. Ollivier-ricci curvature-based method to community detection in complex networks. *Scientific reports*, 9(1):1–12, 2019.

[43] Ali Caner Türkmen. A review of nonnegative matrix factorization methods for clustering. *arXiv preprint arXiv:1507.03194*, 2015.

[44] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1073–1080, New York, NY, USA, 2009. Association for Computing Machinery.

[45] Chi Wang, Edmond Jonckheere, and Reza Banirazi. Wireless network capacity versus ollivier-ricci curvature under heat-diffusion (hd) protocol. In *2014 American Control Conference*, pages 3536–3541. IEEE, 2014.

[46] Chi Wang, Edmond Jonckheere, and Reza Banirazi. Interference constrained network control based on curvature. In *2016 American Control Conference (ACC)*, pages 6036–6041. IEEE, 2016.

[47] Chi Wang, Edmond Jonckheere, and Todd Brun. Ollivier-ricci curvature and fast approximation to tree-width in embeddability of qubo problems. In *2014 6th International Symposium on Communications, Control and Signal Processing (ISCCSP)*, pages 598–601. IEEE, 2014.

[48] Chi Wang, Edmond Jonckheere, and Todd Brun. Differential geometric treewidth estimation in adiabatic quantum computation. *Quantum Information Processing*, 15(10):3951–3966, 2016.

[49] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[50] Wikipedia, the free encyclopedia, user Cepheus. Osculating circle for curvature, 2006. [Online; Public domain; accessed April 17, 2020].

[51] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.