APPLICATIONS OF PERSISTENT HOMOLOGY TO IDENTIFY CLUSTERS IN GRAPHS

M. SKILLETER, K. TURNER, M. RIZOIU

Abstract. We study practical applications of persistent homology to identifying bridges in graphs. These graphs arise when analysing social networks, where bridges represents users who are members of multiple communities. In this paper, we outline several variations on an algorithm that utilises persistent homology and a metric on persistence diagrams, called the Wasserstein Distance, to identify cluster points. We also provide a mathematical justification for the efficacy of this algorithm and discuss ways in which it can be optimised to be applied to large data sets.

1. Introduction

Society is just beginning to see the effect of dedicated diffusers of malicious information. Private and state agencies are using platforms such as Facebook and Twitter to push their ideas, and the problem of identifying these information distributors is becoming more important. In this paper, we will demonstrate how techniques from the area of topological data analysis can be used to address this problem. Specifically, we will use ideas from persistent homology to identify bridges in graphs.

The field of topological data analysis (TDA) has been rising in prominence since the start of the early 21st century. It utilises ideas from an area of mathematics called algebraic topology to draw inferences about large data sets. As it becomes more widespread, various programming languages have added support for TDA. A full implementation of our algorithm in the language Julia can be found at [insert link here], making use of the package Eirene.

We will now give the important definitions, as well as state some useful theorems (without proof).

1.1. Graph Theory.

A graph is a tuple (V, E, μ) with V a set of vertices, E a set of edges and $\mu : V \cup E \to \mathbb{R}_{\geq 0}$ a weight function. An edge between vertices v_0 and v_1 will be denoted by (v_0, v_1). A graph is called *undirected* if $(v_0, v_1) = (v_1, v_0)$ for every edge.

An important example of a weight function is the *shortest path* weight function, which can be computed using Djikstra's Algorithm. Fix a vertex v and for any other vertex w, define $\mu(w)$ to be the length of the shortest path from w to v. For an edge (v_0, v_1), we define $\mu((v_0, v_1)) = \max\{\mu(v_0), \mu(v_1)\}$.

The *path-component* of a vertex v is the subgraph of all vertices that can be reached from v. A graph is called *connected* if it only has one path-component. Notice that if v and w are in different path components then the distance between them under the shortest path weight function is ∞.

References

[1]
[2] David R. Wilkins: *Getting started with Latex*,

http://www.maths.tcd.ie/~dwilkins/LaTeXPrimer/

[3] http://en.wikibooks.org/wiki/LaTeX

Australian National University, Mathematical Sciences Institute