1 Motivation


Consider a weighted undirected graph $G = (V, E, \ell)$, with vertex values $v_0 : T \rightarrow \mathbb{R}$. Our goal is to assign values to nodes in $V \setminus T$ such that the values between nodes are as smooth as possible. The minimal Lipschitz extension of $v_0$ is a vector $v$ that minimizes

$$\max_{(x, y) \in E} \ell(x, y)^{-1} \lVert v(x) - v(y) \rVert,$$

subject to $v(x) = v_0(x)$ for all $x \in T$. We call such a vector an inf-minimizer. Inf-minimizers are not unique. So, among inf-minimizers we seek vectors that minimize the second-largest absolute value of $\ell(x, y)^{-1} \lVert v(x) - v(y) \rVert$ across edges, and then the third-largest given that, and so on. We call such a vector $v$ a lex-minimizer.

Previously, Xiao Shi (CPSC 490, Fall 2015) proposed an iterative algorithm for solving the lex-minimizer and proved its convergence using fixed-point analysis. Suppose we perform Lipschitz learning on the graph iteratively. The algorithm is terminated when the vertices' values are within some error bound $\epsilon$ of their true values. This algorithm is as follows:

\begin{algorithm}
\caption{IterLex for Uniformly Weighted Graphs}
1: $v_1|T \leftarrow v_0|T$  //Set all unassigned vertices to values of vertices in $T$
2: for all $x \in V$ do
3: \hspace{1em} $v_1(x) \leftarrow c$
4: \hspace{1em} $t \leftarrow 0$
5: \hspace{1em} while termination criterion is not met do $v_{t+1}|T \leftarrow v_0|T$
6: \hspace{2em} for all $x \in V \setminus T$ do
7: \hspace{3em} $v_{t+1}(x) = \frac{1}{2}(\max_{(x, y) \in E} v_t(y) + \min_{(x, z) \in E} v_t(z))$
8: \hspace{1em} $t \leftarrow t + 1$
\end{algorithm}

This lex-minimizer can easily be generalized to incorporate non-uniform weights, which can be read about in Section 3 of Xiao Shi’s CPSC 490 Thesis.

2 Approaches on Rate of Convergence

Xiao Shi’s heuristic leaves one question open: What is its rate of convergence? The algorithm presupposes that the true vertex values are already known, and it terminates the algorithm if the numerically computed vertex values approach the true values within a bound of $\pm \epsilon$. Here, I will address two approaches to analysis.

2.1 Convexity of vertex values

When applying this algorithm, vertex values have previously been proven to be monotonically increasing. The rate of convergence could possibly be geometrically analyzed by determining how
quickly groups of nodes approach their true values. We can achieve this through exploiting the shape of convex functions, geometrically determining the rate of change in chord lengths lying above a piecewise function that adequately captures and preserves vertex values at each time step. Here, I suggest two ways in which to do this:

1. At $t = 0$, sort all vertices $\in V \setminus T$ in increasing order. For these sorted vertices, plot each vertex value $v(x)$. The result is a convex plot.

Of course, at each subsequent time step, different vertices may increase in value at different ways. For example, at $t = 0$, suppose that there are 2 vertices $u_0 < v_0$. However, $u_t > v_t$. In other words, nodes increasing at different rates would destroy the convexity of our function. To preserve convexity, we can repeatedly re-sort the nodes at each $t = 1, 2, \ldots$. However, doing so would be to assume that nodes and their values are interchangeable, which may not be the case.

2. Another approach is to arbitrarily order the vertices and cumulatively sum the values of the vertices to attain a convex function. But this approach assumes that a particular node’s rate of convergence is dependent on preceding, arbitrarily ordered vertices. This may also not be the correct approach.

2.2 Paths of vertices

We could look at lengths of paths between $u \in V \setminus T$ and $v \in T$. My intuition is also to investigate the sum of degrees of each vertex on a particular path. In other words, for some path $P$, examine

$$\sum_{v \in P} \deg(v')$$

where $v', \ldots, v^n$ are the nodes in the path.

Perhaps there could be something to be said about density/connectedness with respect to rate of convergence.

3 List of Deliverables

This semester, I will accomplish the following:

1. Find a method by which I can correctly approach addressing the rate of convergence.

2. Determine the rate of convergence for Shi’s algorithm and find a bound on it.

3. Implement Xiao Shi’s algorithm via Julia, perform tests on it, and numerically verify that my bound is correct.

4. If time permits, explore other topics.

Other topics I may investigate if time permits (sorry that I didn’t consult you on this, Dan–but it just popped in my head the other day!). Lex-minimization may have the capacity to inform and encode the degrees of each vertex as well as their density/sparseness with respect to the terminal nodes.

1. Different selections of terminal nodes $T$ on a graph $G$ for which the lex-minimization yields the same result: It may turn out that one particular selection of $T$ yields a unique lex-minimization that will always be different from if we were to select another set of terminal nodes, $T'$, given that we’re not selecting some symmetrical set of nodes. However, if lex-minimizers are non-unique among different selections of $T$, perhaps we could gain some insight on convergence by investigating the properties of these settings.

2. Different graphs under which the lex-minimization yields the same values: If such sets of graphs exist, what can be said about the commonalities between them?