Title

Edit Distance Between Merge Trees

Authors

Raghavendra Sridharamurthy, Talha Bin Masood, Adhitya Kamakshidasan, Vijay Natarajan

Abstract

Topological structures such as the merge tree provide an abstract and succinct representation of scalar fields. They facilitate effective visualization and interactive exploration of feature-rich data. A merge tree captures the topology of sub-level and super-level sets in a scalar field. Estimating the similarity between merge trees is an important problem with applications to feature-directed visualization of time-varying data. We present an approach based on tree edit distance to compare merge trees. The comparison measure satisfies metric properties, it can be computed efficiently, and the cost model for the edit operations is both intuitive and captures well-known properties of merge trees. Experimental results on time-varying scalar fields, 3D cryo electron microscopy data, shape data, and various synthetic datasets show the utility of the edit distance towards a feature-driven analysis of scalar fields.

Summary

Let $(\mathbb{X}, f)$ be a 2-dimensional scalar field. We define the merge tree M of $(\mathbb{X}, f)$ by defining an equivalence relation similar to a Reeb graph except instead of looking at the level sets of f, we look at the sub-level sets $(-\infty, a]$. The merge tree (or join tree) is a well-defined tree. Let T be a rooted tree with vertex set V and edge set E. In addition, we have a label set Σ and a character $\lambda \notin \Sigma$ which represents the null or empty character, which corresponds to a 'gap'.

Defines an Edit distance whose goal is to be stable (less than maximum function distance) and discriminative (distance is always greater than bottleneck distance). The three elementary operations they use are

- *Relabel:* A relabel $a \to b$ corresponds to changing the label a is changed to the label b
- *Insertion:* An insert operation $\lambda \to b$ inserts a node n with label b as a child of another node m by moving all children of node m to be childrend of the new node n
- *Deletion:* A deletion operation $a \to \lambda$ removes a node n with label a and all the children of node n are now children of the parent node of n $(parent(n))$.

Note that we have not yet defnied a labeling on the merge trees.

We define an *Edit Distance Mapping* as being a triple (M_e, T_1, T_2), where T_1 and T_2 are two trees and M_e is a collection of ordered pairs (i, j). Furthermore, t_1 and t_2 are labelings of the nodes in T_1 and T_2. Lastly, the pairs in M_e satisfy the following:

- $i_1 = i_2$ iff $j_1 = j_2$
- $t_1[i_1]$ is an ancestor of $t_1[i_2]$ iff $t_2[j_1]$ is an ancestor of $t_2[j_2]$.

Then, the cost of transforming T_1 into T_2 would be the cost of all changes from i to j, as well as the deletion of i if there exists no corresponding j and insertion of j if there exists no corresponding i. We call this the cost of M_e, denoted as $\gamma(M_e)$. Minimizing this cost over every possible edit mappings has been shown to be NP-complete. Thus, we seek to constrain these problems to make them contractible.

Since the information encoded in merge trees is very specific, this limits the number of operations we can do. For example, if we were to delete a minima which connects to a saddle above and that saddle connects to no other minima, then this tree will now have a saddle as a leaf, which is no longer a merge tree. Thus, to make this correct, we sometimes need to delete minima and connecting saddles in pairs.

When measuring the edit distance, one issue that is encountered is small perturbations in the data set can lead to large differences in the cost. Consider two saddles that are have close function value. If the order of the saddles were reversed, then the pairings between these saddles and the minima below also changes. Since the edit cost is dependent on these persistence pairs, the edit cost is drastically affected. To alleviate this, they introduce a parameter ε which is known as the *stability parameter*. Before running the edit cost algorithm, each merge tree is run to a stabalizing algroithm where two saddles that are within ε of each other are merged into a single saddle (bottom up).

We can now define the set of costs we assign for merge trees. These costs are dependent on the persistence of these features. Let p and q be nodes in T_1 and T_2, respectively. Then, p and q are creators or destroyers of features in the merge trees. Let (b_p, d_p) and (b_q, d_q) be the birth and death times of these associated features, respectively. These can be represented as coordinates in the persistence diagram or as intervals on persistence barcodes.

We define the L_∞ cost C_W as follows:

$$\gamma(p \to q) = \min \begin{cases} \max\{|b_q - b_p|, |d_q - d_p|\}, \\ \frac{|d_p - b_p| + |d_q - b_q|}{2} \end{cases}$$

$$\gamma(p \to \lambda) = \frac{|d_p - b_p|}{2}$$

$$\gamma(\lambda \to q) = \frac{|d_q - b_q|}{2}$$

Essentailly, to relabel p as q, we take the minimum of the largest discrepency in birth or death time and the sum of half the distances from the diagonal of the persistence diagram. To remove p or insert q, we just take half the distance from the diagonal.

We defined the $\text{Overhange Cost } C_O$ as follows:

$$\gamma(p \to q) = \min \begin{cases} |b_q - b_p| + |d_q - d_p|, \\ |d_p - b_p| + |d_q - b_q| \end{cases}$$

$$\gamma(p \to \lambda) = |d_p - b_p|$$

$$\gamma(\lambda \to q) = |d_q - b_q|$$

Turns out that C_W is a metric.

They provide some experimental studies for the usefulness of this metric. First, they provide an example of three scalar fields f_1, f_2, f_3 (simple and predictable) whose merge tree is provided. In this example, they run the algorithm to compute the tree mappings based on both the tree edit distance D and the wasserstein distance W (I'm not currently sure how exactly they are using wasserstein distance in their cost model). The edit distance cost model maps the tree of f_1 correctly to f_2 and from f_2 correctly to f_3. However, the wasserstein cost model fails once in each pair. See the paper for details, as they provide an explanation as to why W fails in these cases. They claim that "D in general establishes mappings that are better than D_B and W_1 because it is aware of the structure of the merge tree and preserves the heirarchy captured in the tree".

I believe what they mean by the matchings created by Wasserstein distance is the standard way that Wasserstein distance has to determine a matching between two persistence features. That is, in order to compute W, we need to find the "best matching" between the points in the persistence diagram. This is NOT using any sort of algorithm based on tree edit distance. In contrast, the tree edit distance measure D assigns a mapping (i.e. finds a "best matching) differently. This is based completely on the algorithm described as the "constrained edit distance mapping" (possibly the restricted case). So this specifically doesn't have to do with the cost model but only the edit distance mappings described in previous papers.

They provide experimental results showing that D is more discriminative than W_1 and D_B. This is for all values of the stability parameter ε. However, increasing ε to a higher percentage of the maximum persistence (say from 1% to 20%) closes the gap between these metrics. This is because as the parameter increase the tree slowly loses the tree structure and eventually all the nodes in the tree become direct children of the root node.

Theoretical Contributions

The actual theoretical contributions from this paper are in the definition of the cost model for merge (and split) trees. It also includes their proofs that these cost models are indeed metrics.

Computational Contributions

Algorithm for finding the tree edit mappings is described in "A constrained edit distance between unordered labeled trees" by K. Zhang, where they then applied the discuessed edit costs. The inputs are two merge trees stabalized (see stabalization above). Note that the mappings created by the algorithm are depending on the choice of cost functions.

Applications and Experiments

Need to re-read experimental section