Learning from Hierarchy via Persistence Forests

With applications to graph learning

Sergei Burkin

University of Tokyo, Graduate School of Mathematical Sciences

Summary

Here all functors are functors from categories corresponding to posets. Persistent homology $\mathcal{PH}_*$ of a filtered space is functor from $(\mathbb{R}, \leq)$. We introduce **persistence forest**, functor $\mathcal{PF}_*$ from merge forest MF_f ([2]) of filtered space (X, f). This functor mimics the decomposition of homology of a space into direct sum of homology of its connected components.

Persistence forests of spaces can be used as input to machine learning algorithms. In particular, the problem of learning from graphs can be reduced to the problem of learning from directed forests with additional information in vertices.

Definition of persistence forest

- A *filtered space* (X, f) is a space X with a function $f : X \to \mathbb{R}$.
- For any $t \in \mathbb{R}$, the *level subset* F_t is the space $f^{-1}((-\infty, t])$.
- The *persistent homology* of (X, f) is the functor $\mathcal{PH}_* : (\mathbb{R}, <) \to R\text{-grMod} : t \mapsto H_*(F_t)$.
  - Every map $\mathcal{PH}_*(s \to t)$ is induced by the canonical inclusion of F_s into F_t.
- The *merge forest* of (X, f) is the space $MF_f = \{(x, t) \in X \times \mathbb{R}, t \geq f(x)\}/\sim$, where $(x_1, t_1) \sim (x_2, t_2)$, if $t_1 = t_2$ and the points x_1 and x_2 lie in the same connected component of F_{t_1}.
- Points of MF_f correspond to connected components of spaces F_t: there is a bijection $MF_f \to \cup_{t\in\mathbb{R}}(\pi_0(F_t), t) : [(x, t)] \mapsto (F_{t,j}, t)$, where $x \in F_{t,j}$.
- Merge forest is a poset: $(F_{t_1,j_1}, t_1) \prec (F_{t_2,j_2}, t_2)$, if $t_1 < t_2$ and $F_{t_1,j_1} \subset F_{t_2,j_2}$.
- The *persistence forest* of (X, f) is the functor $\mathcal{PF}_{f,*} : MF_f \to R\text{-grMod} : (F_{t,j}, t) \mapsto H_*(F_{t,j})$.

Interleaving pseudo-distance

- There is *interleaving (pseudo)-distance* between persistence forests $d_{PF}((X, f), (Y, g)) = d_{PF}(\mathcal{PF}_{f,*}, \mathcal{PF}_{g,*}) \in [0, +\infty]$.
- d_{PF} generalizes d_{PH}, the similar notion for persistent homology.
- Persistence forests are *stable* with respect to small deformations of a filtered space: $d_{PF}((X, f), (X, g)) \leq \|f - g\|_\infty$.
- Persistence forest is a more sensitive invariant than persistent homology: $d_{PF}((X, f), (Y, g)) \geq d_{PH}((X, f), (Y, g))$.

Partitional clustering, hierarchical clustering

- *Partitional clustering* is an algorithm that from a finite subset S of $\mathbb{R}^n$ or, in general, from a finite metric space (S, μ) constructs **a partition of the set** S.
- *Hierarchical clustering* constructs **a heighted tree**, i.e. a tree in which each vertex has a height, and the height of a parent vertex is always bigger than the height of its children. Elements of S are vertices of height zero.
- Hierarchical clustering generalizes partitional clustering: given a hierarchical clustering for every choice of height there is corresponding partitional clustering.
- There are several constructions of a filtered space (X, f) from finite space S.
- For these constructions, the merge forest MF_f is a hierarchical clustering of S.
- Merge forest is a "canonical basis" of zero persistent homology.
- Thus $\mathcal{PH}_*$ and $\mathcal{PF}_*$ are generalizations of hierarchical clustering.

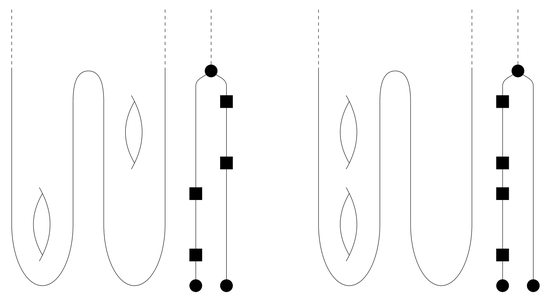
Learning *via* graphs

- There are machine learning algorithms that can learn from graphs.
- These algorithms are usually based on partitional clustering of vertices of a graph.
- If an algorithm L can learn from graphs, it can learn from forests.
- If $Spaces$ is some type of spaces and $F : Spaces \to Forests$ is a hierarchical clustering, then $L \circ F$ is an algorithm that can learn from $Spaces$.
- Computing persistence forest is a particular case of such F.

Learning *from* graphs

- The set $Spaces$ can be a set of graphs.
- On a weighted graph (V, E) define a filtration by $f(x) = min_{v\in V} d(x, v)$.
- On an abstract graph define a weight: the weight of edge (v_1, v_2) is $g(deg(v_1), deg(v_2))$, where g is any symmetric function.
- Compute persistent forests of such graphs and then use the approach above.

Illustration of persistence forests



The two filtered spaces above have the same persistent homology but different persistence forests, i.e. the corresponding functors $\mathcal{PH}_*$ are isomorphic, thus the distance d_{PH} between these spaces is zero, but the distance d_{PF} is non-zero.

Here the filtration function is given by the height of point. Circles and squares denote "critical points" for H_0 and H_1. Both forests start with two zero homology classes being born, then first homology classes are born (and they never die), then the two zero homology classes merge into one.

Applications

Graph learning. The approach described on the left works well in practice. Let L be the algorithm $DGCNN$ (Zhang et al, 2018). The classification accuracy of $L \circ PF$, compared to L alone, is 2-5% lower on biochemical datasets and 3-8% higher on social datasets. Note that biochemical graphs contain additional labeling of vertices, and PF ignores it. $L \circ PF$ outperforms L if these labels are removed.

The training of $L \circ PF$ converges faster than that of L.

Most likely, $L \circ PF$ learns from the hierarchical structure of a graph at different scales.

Reducing sizes of graph datasets. If the task is to learn from graphs, then the number of edges can be reduced by at least $|E|/2|V|$.

Which clustering/weighting is more natural for a dataset? The one, whose combination with (one or several different algorithms) L, achieves better classification accuracy. The difference in accuracy for different weightings given by $g \in \{max, min, x + y, x^2+y^2, |x-y|+1\}$ is usually 5-10%. The author did not test whether combination of several choices of g produces better accuracy. If it does, one might say that different weightings provide essentially different information.

Finding weaknesses in graph learning algorithms. Two different versions of PF followed by $DGCNN$ were tested: version with labels by $\dim H_1$ in vertices and version without labels. The version without labels performed better. In principle, this should not happen. Likewise, $L \circ PF$ should not outperform L, but in this case it did.

Comparison with another approach. DiffPool (Ying et al, 2018) is another approach based on hierarchical clustering followed by graph learning. In case of DiffPool hierarchical clustering is not given beforehand, but learned during training. It can use labels on vertices of graphs, and, unlike our approach, it outperforms the standard approach on biochemical datasets. The downside is that it is impossible to reduce the size of a dataset before training, and it might be hard to understand how the learned clustering works.

Further developments

Merge forest is a "canonical basis" of zero persistent homology. To obtain such bases for higher persistent homology one might try to use Hodge theory. André Lieutier made a presentation about this idea several years ago.

Acknowledgements

We thank Toshitake Kohno, Genki Sato, Jun Yoshida for helpful comments and discussions, and Yuanyuan Bao for asking about Hodge version. This work was supported by Program for Leading Graduate Schools, Japan Society of the Promotion of Science "Leading Graduate Course for Frontiers of Mathematical Sciences and Physics" (FMSP), and Japanese Government (Monbukagakusho: MEXT) Scholarship.

References

[1] Jon M. Kleinberg, *An impossibility theorem for clustering*, Advances in Neural Information Processing Systems 15 (S. Becker, S. Thrun, and K. Obermayer, eds.), MIT Press, 2003, pp. 463--470.

[2] Dmitriy Morozov, Kenes Beketayev, and Gunther Weber, *Interleaving distance between merge trees*, (2013).

Contact

E-mail: sburkin@ms.u-tokyo.ac.jp