Semi-Supervised Learning with Cover Trees

Avneesh Saluja
Carnegie Mellon University
avneesh@cmu.edu

Branislav Kveton
Technicolor Labs
branislav.kveton@technicolor.com

1 Introduction

Semi-supervised learning (SSL) has emerged in recent years as an important tool to tackle large amounts of data. Generally in large-data scenarios, one finds that the ratio of unlabeled to labeled data is very high, and that annotating and labeling data for training a learning algorithm is often resource-demanding and expensive. This issue makes semi-supervised techniques a natural way to explore, understand, and learn from these large datasets [14].

A subset of semi-supervised learning approaches that has been quite successful relies on constructing a similarity or a data adjacency graph between all the points in the training set, labeled and unlabeled, and using the structure of the graph as well as the labeled points to predict labels over the unlabeled data. These methods are collectively referred to as graph-based semi-supervised learning. Graph-based learning is $\Omega(n^2)$ in the number of datapoints in the training set [4], because it takes $\theta(n^2)$ time to build the similarity graph over all datapoints. This approach is impractical for large n.

Many approximations exist to circumvent or limit the quadratic time complexity, the most common approach being to select k representative vertices of the graph [9, 12, 13]. The basic idea is to maintain a compact representation of the entire similarity graph, and infer labels for these representative vertices using a graph-based SSL method. Subsequently, labels are propagated from the chosen vertices to the rest of the graph. For example, Liu *et al.* [9] pick a set of representative points known as anchors, and express all other points as a convex combination of the anchor points. The complexity of these methods is $\Omega(k^2 + k(n-k))$ because it takes $\theta(k^2)$ time to build the representative vertex similarity graph and $\theta(k(n-k))$ time to propagate labels to the rest of the data, in addition to the time required to compute labels for the unlabeled data using graph-based SSL. These methods are linear in n if k is independent of n, however in practice k and n are dependent, as a large k is required to get good coverage for large datasets (large n).

In this work, we investigate another way of summarizing the data. Instead of focusing on k representative vertices, we represent data as a tree at different levels of granularity. We show that our approach is scalable to large datasets, and compares favorably to representative vertex selection methods for scaling up graph-based SSL.

2 Approach

Our work concentrates on the scalability of a well-known graph-based SSL algorithm, the harmonic function solution (HFS) [15]. We note however, that the presented approach is applicable to any graph-based SSL algorithm and can result in speed-ups for practically all such methods.

We first construct a *cover tree* of the entire dataset prior to graph construction. The idea is that, given a set of points in a metric space defined by the metric d, we can preprocess the data such that nearest neighbor queries can be done efficiently. Generally, nearest neighbor search is $\theta(n)$, but by leveraging the structure of the data we can speed up the query time. The cover tree 'covers' the data at various levels of granularity. From Beygelzimer *et al.* [5][1]:

[1] we modify the indexing of the levels compared to the original definition, this has no effect on the actual data structure

Definition 1. *A cover tree T on a dataset S is a leveled tree where each level is a "cover" for the level beneath it. Each level is indexed by an integer scale i which increases as the tree is descended. Let C_i denote the set of nodes at level i. A cover tree T on a dataset S obeys the following invariants for all i:*

- *(nesting) $C_i \subset C_{i+1}$*
- *(covering tree) For every $p \in C_{i+1}$, there exists a $q \in C_i$ satisfying $d(p,q) \leq 2^i$, and exactly one such q is a parent of p*
- *(separation) For all $p, q \in C_i, d(p,q) > 2^i$*

Given a fixed intrinsic dimensionality (see [5] for more details), the cover tree construction time is is $O(n \log n)$, the running time of a nearest neighbor query is $O(\log n)$, and the space usage is only $O(n)$. The confluence of these properties allows us to scale to very large datasets. Compare these costs with conventional graph construction approaches, which are $\theta(n^2)$ in time and space. Furthermore, these costs also improve upon representative vertex-based approximation methods when k and n are dependent. For example if we choose $k = \sqrt{n}$, the time complexity for graph construction is $\Omega(n^{3/2})$, versus $O(n \log n)$ construction time for the cover tree[2]. Yet, we observed (Section 3) that we often need $k > \sqrt{n}$ to get meaningful performance for large n.

From the cover tree, we construct an ultra-sparse similarity matrix W_{uu} of the unlabeled datapoints. We include in the matrix W_{uu} an edge between the every datapoint in the tree and its parent. Thus, we have $n-1$ edges in our similarity graph. Note that this is the minimum number of edges needed to cover the dataset with a single connected component. Each unlabeled example is also connected to its closest labeled vertex (submatrix W_{ul}). Our representation can be viewed as a small world network [6]: each vertex is reachable from any other vertex in $O(\log n)$ steps. Furthermore, the distance between a parent at level i and its children is in the interval $[\frac{1}{2^{i+1}}, \frac{1}{2^i}]$. Afterwards, we compute a graph Laplacian from the similarity matrix, which is then used in the HFS computation. We take this opportunity to briefly remind the reader of the HFS. We wish to minimize the following objective function:

$$\min_{\boldsymbol{\ell} \in \mathbb{R}^n} \boldsymbol{\ell}^\intercal L \boldsymbol{\ell} \quad \text{s.t. } \ell_i = y_i \text{ for all } i \in l; \tag{1}$$

where $\boldsymbol{\ell}$ is the vector of predictions on both labeled and unlabeled examples, $L = D - W$ is the Laplacian of the similarity graph, which is given by a matrix W of pairwise similarities w_{ij}, and D is a diagonal matrix defined as $d_i = \sum_j w_{ij}$. This problem has a closed-form solution:

$$\boldsymbol{\ell}_u = (L_{uu})^{-1} W_{ul} \boldsymbol{\ell}_l, \tag{2}$$

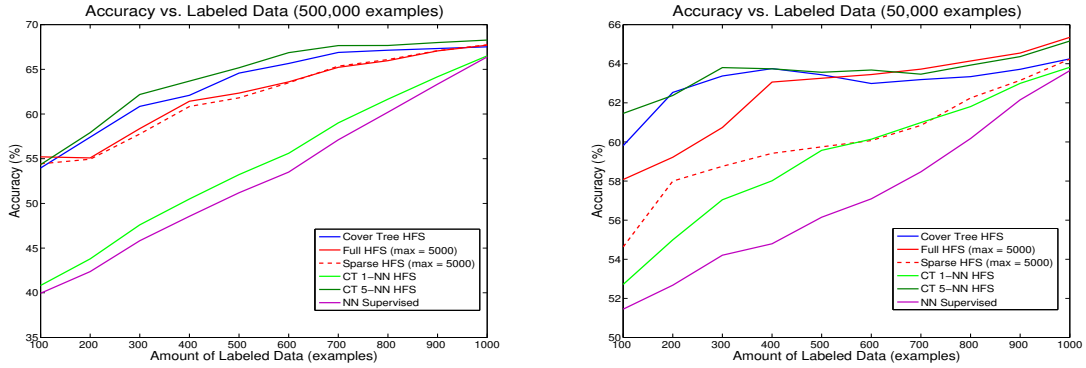
which is known as the *harmonic function solution* because it satisfied the *harmonic property* $\ell_i = \frac{1}{d_i} \sum_{j \sim i} w_{ij} \ell_j$.

By using a sparse Laplacian matrix, we can speed up the computation of Equation 2. Two such approaches for sparsifying are k-nearest neighbor (k-NN) sparsification, and edge-weight thresholding sparsification. In k-NN sparsification, we maintain edges between a point and its k nearest neighbors and discard the other edges. In edge-weight thresholding, all edge weights less than a chosen threshold are set to be 0. For small dataset sizes, these heuristics are appropriate, but for very large datasets they break down, since the heuristics rely on first computing the entire similarity matrix and then sparsifying. The cover tree-based approach, however, not only speeds up graph construction but also yields a naturally sparse representation for our similarity graph, since we only preserve edge weights between parents and children. The full similarity matrix is never computed before sparsifying.

3 Experiments & Results

We conducted experiments on two datasets. The first, the Covertype dataset ('Cover') [2], consisted of more than 580,000 examples, and the goal was to predict forest cover types from cartographic features only. The second dataset we used was the 2004 KDD Cup Particle Physics dataset ('Physics')

[2] in fact, $O(n \log n)$ is asymptotically smaller than $O(n^{1+\epsilon}),\ \epsilon > 0$



(a) 'Cover' dataset, 500,000 examples

(b) 'Physics' dataset, 50,000 examples

Figure 1: Accuracy as a function of the number of labeled examples the 'Cover' and 'Physics' datasets. The cover tree results (blue) are on par with 5-NN, despite being computed significantly faster.

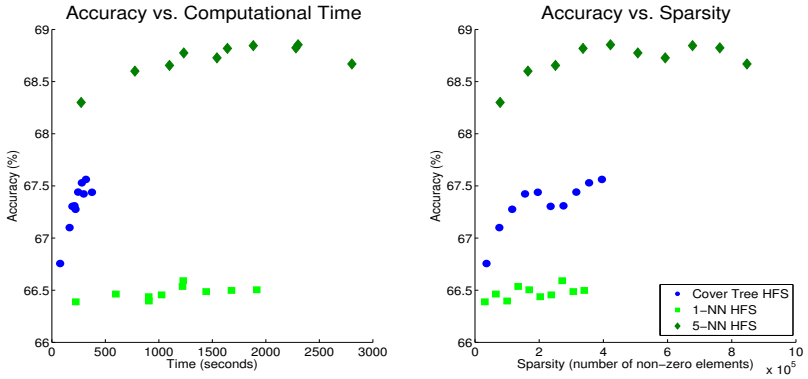


Figure 2: Accuracy as a function of computational time and sparsity for the cover tree, 1-NN and 5-NN HFS. Note that while the cover tree is similar in sparsity as 1-NN, it achieves higher accuracy.

[1], which consisted of 50,000 examples. All experiments were conducted on a 2.66 GHz Intel Core 2 Duo Macbook Pro with 8GB RAM. Despite the limited hardware, we were able to handle more than half a million datapoints with the cover tree HFS implementation on one laptop.

We compared our cover tree approach to a) a supervised, nearest-neighbor classifier, b) an approximate HFS method, which we call "subsetHFS", where we randomly selected 5000 representative vertices, solved the HFS over these vertices, and then used 1-nearest neighbor label propagation from the HFS-labeled points to label the remaining points in the dataset, and c) k-NN similarity matrices (for $k = \{1, 5\}$) that are constructed from the cover tree, i.e., we use the cover tree to find the k nearest neighbors and symmetrize the similarity matrix. Also, to compare the accuracy for similar levels of sparsity, we tried a variant of "subsetHFS" where we first computed the full similarity matrix with $k = 5000$ and then enforced a threshold for the edge weight. We chose the threshold to ensure the number of non-zero elements would be roughly equivalent to the cover tree representation. We fixed the total number of examples at 500,000 and 50,000 respectively, and measured accuracy as a function of the number of labeled examples (varying from 100 to 1000). Figure 1 shows these trends.

Lastly, to compare the cover tree and cover tree-derived 1-NN and 5-NN solutions, we varied the number of datapoints from 10,000 to 100,000 examples for the 'Cover' dataset and at each stage recorded the computational time taken to compute the requisite matrices and their respective sparsities. The results are presented in Figure 2. The computational time takes into account constructing the cover tree and k-NN graphs, computing the respective Laplacians and solving the HFS using a linear solver. Sparsity is measured by the number of non-zero elements (non-zero edges between unlabeled datapoints and unlabeled and labeled datapoints). The important point to bear in mind is

the k-NN graph construction is on top of the cover tree construction and adds significantly to the computation time (up to 6 times longer). Note that computational time and sparsity comparisons with "subsetHFS" were not done: for "subsetHFS", the time scales with $\Omega(k^2 + k(n-k))$, where $k = 5000$, and the number of non-zero elements is on a different scale, since we are dealing with a full 5000×50000 matrix.

From these results, we see that the our proposed cover tree approach compares favorably to both "subsetHFS"-based and cover tree-based k-NN approaches. Regarding subset selection, there are obviously better ways to select representative vertices for HFS computation rather than random selection, for example k-means clustering. But all of these preprocessing techniques come at a high price, especially for a high number of clusters k, and high dimensionality of the data (although we note that there is work on speeding up these selection methods). We achieve similar sparsity levels to a 1-NN graph, yet better accuracy in general. It is interesting to see that while our cover tree matrix is much more sparse compared to a 5-NN matrix, the results for the 'Cover' dataset are comparable. The ultra-sparse cover tree representation of maintaining edges only between parents and children, while computed in a fraction of the time as the other approaches (Figure 2), is on par with them in terms of accuracy.

4 Related Work

An alternative approach undertaken by Fergus *et al.* [7] is based on making significant assumptions about the distribution from which the data was generated. In particular, since the HFS amounts to solving a large linear system, one can reduce the size of the problem by only considering a smaller number of eigenvectors of the Laplacian. But if instead of computing over the graph Laplacian directly, one makes the assumption that the data is generated by a certain parametric distribution that can be written as the product of distributions over individual features, it may be simpler to compute the eigenfunctions of the probability function directly. This corresponds to computing eigenfunctions of the Laplace-Beltrami operator. This approximation becomes more valid as the number of datapoints $n \to \infty$. The method relies on the data being separable by dimension, an assumption which does not always hold in real-world datasets.

Some prior work has looked at using kd-trees [3] for data representation, but such approaches often fall prey to the curse of dimensionality, since kd-trees are notorious for scaling poorly in dimension [8]. Kd-trees are also dependent on the real dimensionality of the data, while cover trees have intrinsic dimensionality dependence. Parallelization [11] and distributed computing [10] perspectives have also been used for large-scale SSL, and while these methods are orthogonal to the direction that we pursue in this work, they can be easily combined with cover trees to yield additional scalability and speed-ups.

5 Discussion and Future Work

We observed that for k of moderate magnitude (i.e., $k = 5$), the accuracy of the cover tree-derived k-NN graph is higher than our ultra-sparse approach, where we include edges only between parents and children of the cover tree (however for low k this method of edge selection seems to perform better than k-NN). The major deficiency in our approach is that we do not allow connections between different sub-trees in the cover tree, whereas k-NN does not make the distinction between sub-trees. The comparison with cover tree-based k-NN graphs is insightful in that it suggests using the cover tree as a way to combine different nearest neighbor representations at various levels of the tree. The cover tree allows us to look at the coverage of the data at different levels of granularity and provides a way to trade off approximation quality and accuracy in a principled and intuitive manner.

Graph construction has not been closely looked at, despite its significant contribution to most graph-based SSL algorithms' computational costs. What we have shown in this work is a fast an efficient way to construct a naturally sparse similarity matrix or Laplacian, with resulting speed-ups in harmonic function solution computation. The main idea behind improved performance is to represent the data at different levels of granularity, and the techniques are broadly applicable to an array of graph-based SSL algorithms.

References

[1] The 2004 kdd-cup dataset. http://osmot.cs.cornell.edu/kddcup/.

[2] Uci machine learning repository. http://archive.ics.uci.edu/ml/.

[3] Andreas Argyriou. Efficient approximation methods for harmonic semi-supervised learning, 2004.

[4] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434, 2006.

[5] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pages 97–104, 2006.

[6] David Easley and Jon Kleinberg. *Networks, Crowds, and Markets*. Cambridge Books. Cambridge University Press, 2010.

[7] Rob Fergus, Y. Weiss, and Antonio Torralba. Semi-supervised learning in gigantic image collections. *Advances in Neural Information Processing Systems*, 1:1–9, 2009.

[8] Jacob E. Goodman, Joseph O'Rourke, and Piotr Indyk, editors. *Handbook of discrete and computational geometry (2nd ed.)*. CRC Press, Inc., Boca Raton, FL, USA, 2004.

[9] Wei Liu, Junfeng He, and S.F. Chang. Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1–8, 2010.

[10] Delip Rao and David Yarowsky. Ranking and semi-supervised classification on large scale graphs using map-reduce. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing - TextGraphs-4*, number August, page 58. Association for Computational Linguistics, 2009.

[11] Amarnag Subramanya and J.A. Bilmes. Entropic graph regularization in non-parametric semi-supervised classification. In *Neural Information Processing Society (NIPS)*, pages 1–9, 2009.

[12] A. Talwalkar, S. Kumar, and H. Rowley. Large-scale manifold learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2008.

[13] Michal Valko, B. Kveton, L. Huang, and D. Ting. Online Semi-Supervised Learning on Quantized Graphs. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, 2010.

[14] Xiaojin Zhu. Semi-Supervised Learning Literature Survey. Technical report, 2008.

[15] Xiaojin Zhu, Z. Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, volume 20, page 912, 2003.