Scaling HDBSCAN Clustering with kNN Graph Approximation

Jacob Jackson
Petuum, Inc.
jacobbfjackson@gmail.com

Aurick Qiao
Petuum, Inc.
Carnegie Mellon University
aurick.qiao@petuum.com

Eric P. Xing
Petuum, Inc.
Carnegie Mellon University
eric.xing@petuum.com

1 INTRODUCTION

Clustering algorithms have found applications in a variety of fields involving exploratory data analysis, including document retrieval [1], image search [2], and bioinformatics [3]. HDBSCAN [4] is one such algorithm which is well-suited for these applications, being able to model clusters in many naturally occurring data. However, traditional implementations of HDBSCAN scale poorly with the size of data, requiring a full kNN graph [5] to be constructed over all points in the dataset.

Simpler clustering models such as k-means [6] are popular for data-intensive applications because they can be efficiently scaled to large datasets [7]. However, they also tend to have restrictions for the data and/or user, limiting their usefulness for real-world data. For example, they may assume that clusters are normally distributed around a mean point [8], the user knows the number of clusters ahead of time, or that there are no noise points in the data. These shortcomings are avoided by many hierarchical and density-based clustering algorithms like HDBSCAN.

In our work, we investigate the feasibility of scaling up HDBSCAN. We approach the problem by using two key optimizations:

(1) Replace the computationally expensive kNN graph construction with a more scalable approximation algorithm.
(2) Replace a step which finds a minimum spanning tree of the complete graph of all data points, by finding the minimum spanning tree of the kNN graph instead.

Using these optimizations, we obtain an efficient approximation to HDBSCAN which works well on large datasets. We implement this approach as a distributed system, and show that it achieves over 40x better performance than a popular exact implementation of HDBSCAN. Additionally, we find that the approximations we used result in minimal difference when compared with exact HDBSCAN results. We obtain reasonable output from a large word embedding dataset containing hundreds of clusters, demonstrating the flexibility of the approach.

2 BACKGROUND

2.1 HDBSCAN

HDBSCAN is a hierarchical density-based clustering algorithm. It takes N points with dimensionality D and a distance metric d as input. The algorithm works in three phases:

(1) Find the K-th nearest neighbor of each point according to the distance metric d, where K is a hyperparameter. By a_p we denote the distance between point p and its K-th nearest neighbor. In subsequent steps, HDBSCAN uses the following modified distance metric:

$$\tilde{d}(x,y) = \max(a_x, a_y, d(x,y))$$

The purpose of using this modified metric is to reduce sensitivity to outliers.

(2) Find the minimum spanning tree of the points, where for each pair of points x and y there is an edge connecting them with weight $\tilde{d}(x,y)$.
(3) Extract clusters from the minimum spanning tree.

Steps (1) and (2) are computationally expensive: they both have time complexity $O(N^2 D)$. An algorithm which is quadratic in the number of points will not scale to datasets containing millions of points.

One technique for dealing with such datasets is sub-sampling, where the algorithm is run on a fraction of the dataset. This can be effective, but it is not suitable in all instances. For an example, consider the word embedding dataset described in Sec. 4. This dataset contains hundreds of clusters of fewer than 10 points, which would be difficult to extract from a sub-sample of the dataset.

Previous work on HDBSCAN [9] used k-d trees [10] to optimize step (1), but the worst-case complexity of step (1) is unchanged. As our experiments will show, this worst-case complexity is realized when the dimensionality D is sufficiently large. This is why we turn to approximation to scale the algorithm to large N and D.

2.2 NN-Descent

NN-Descent [11] is an approximate kNN graph construction algorithm. It is based on the principle that "a neighbor of a neighbor is likely to be a neighbor". A list of K candidate nearest neighbors is maintained for each point. In each step, for each point p, the algorithm examines all neighbors of neighbors of p and adds them to the neighbor list of p if they are closer than the last entry of p's neighbor list (thus replacing this last entry). This improves the accuracy of the kNN graph in each iteration. The algorithm stops when the number of updates in an iteration falls below a certain threshold. NN-Descent possesses several desirable qualities which make it well-suited for this application:

- It has been shown to perform well on a variety of datasets and distance metrics [11].
- It supports useful non-metric spaces such as the space induced by the cosine similarity metric.
- It has an $O(n^{1.14})$ empirical runtime, which is consistent with our results and makes it scalable to larger datasets.
- It can be easily distributed to multiple cores and machines.

SysML'18, Stanford, CA
. . . $15.00
DOI:

Dimension	FM score	Clusters (theirs)	Clusters (ours)
1	0.896	6267	6285
2	0.959	840	842
3	0.934	1129	1135
4	0.941	1117	1115
5	0.942	1104	1100
6	0.948	1075	1079
7	0.956	1044	1050
8	0.957	1028	1029
9	0.958	1027	1023
10	0.961	1020	1018

Table 1: FM score measuring the similarity of our implementation's output with the output of the exact implementation by McInnes et al.

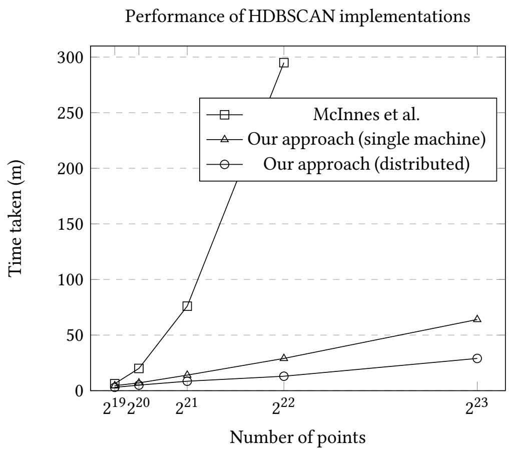

Figure 1: Performance of our implementation compared with the exact implementation by McInnes et al.

3 DESIGN AND IMPLEMENTATION

To approximate step (1) of HDBSCAN, we run NN-Descent.

To approximate step (2), we find the minimum spanning tree of the graph produced by NN-Descent (which has KN edges) rather than the all-pairs graph (which has $N(N-1)/2$ edges). This choice is justified as follows. The only important edges of the minimum spanning tree used by step (3) are those which connect points in the same cluster. Thus we expect that removing edges not present in the k-NN graph will preserve most of the important edges.

We implemented our HDBSCAN application using Litz [12], a framework for distributed machine learning. The minimum spanning tree is found using Kruskal's algorithm [13] when the points fit on one machine's memory and Borůvka's algorithm [14] when they must be split across machines. Most of the computational cost lies in running NN-Descent, so we devoted significant effort to optimizing this part of the algorithm.

We modified NN-Descent to achieve constant-factor improvement in the distributed case. In NN-Descent, points are partitioned evenly among the machines, and each machine keeps track of the nearest neighbors of its points. To update these nearest neighbor lists, an update is sent consisting of x, y, and $d(x,y)$ to the machine holding x. The update is accepted if $d(x,y)$ is less than the distance between x and the most distant element of the nearest neighbor list of x, denoted $d_{\max}(x)$.

Sending data over the network is more expensive than accessing local memory, so we send $d_{\max}(x)$ to all neighbors of x at the start of each iteration of updates. That allows those neighbors to discard updates to x before sending them through the network if the distance of the update is larger than $d_{\max}(x)$. This technique led to a 42% decrease in runtime when running NN-Descent on the GloVe dataset [15] (described in the next section) with 4 machines.

4 RESULTS

To test our implementation on real-world data, we ran it on the GloVe dataset containing 400,000 points with dimension 300. We imposed a maximum cluster size of 1,000 to avoid assigning all points to the same cluster. Using the dot product metric and running on 4 AWS m4.xlarge instances, our implementation finished in 143 seconds, producing 329 clusters as output. We find that the clusters produced are qualitatively valid, examples include:

- loaf loaves rigatoni tofu spaghetti orzo tacos snacks fettuccine penne dente linguine couscous sausage pasta cheese bread noodles sandwiches
- infinitive imperfective noun nominative pronoun dative genitive verb participle subjunctive accusative
- phi kappa epsilon theta sigma

To obtain a more quantitative measure of the accuracy of our approximation, we compare with the implementation by McInnes et al. using the Fowlkes-Mallows index [16] running on a synthetic dataset. The dataset consists of 100,000 points, with 1,000 clusters, where 20% of the points were noise. The remaining points were randomly distributed among the clusters; the points in a cluster were normally distributed about the center of the cluster. Table 1 summarizes these results.

We performed a similar experiment to test efficiency. We generated 10 synthetic clusters of points with dimension 10 and varied the number of points. We found that with 2^{23} points, our single-machine implementation was over 40 times faster. Fig. 1 shows how the implementations scale with the number of points. The distributed tests are run using 4 AWS m4.xlarge instances. The performance of our implementation is consistent with the $O(n^{1.14})$ empirical runtime of NN-Descent.

5 CONCLUSION AND FUTURE WORK

We have described an approach for approximating the result of HDBSCAN with high accuracy and efficiency. This allows HDBSCAN to be used on larger datasets than was previously possible.

Future work may focus on the theoretical properties of this approach, deriving bounds for the inaccuracy due to approximation error. Many methods exist for finding approximate k-NN, and alternatives to NN-Descent deserve investigation.

REFERENCES

[1] Ellen M Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing & Management*, 22(6):465–476, 1986.

[2] Nadav Ben-Haim, Boris Babenko, and Serge Belongie. Improvingweb-based image search via content based clustering. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pages 106–106. IEEE, 2006.

[3] Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.

[4] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. *Density-Based Clustering Based on Hierarchical Density Estimates*, pages 160–172. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[5] Preparata Franco and Michael Ian Preparata Shamos. Computational geometry: an introduction. 1985.

[6] J. MacQueen. Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967)., 1967.

[7] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM, 2010.

[8] J. Lücke and D. Forster. k-Means is a Variational EM Approximation of Gaussian Mixture Models. *ArXiv e-prints*, April 2017.

[9] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11), mar 2017.

[10] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[11] Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 577–586, New York, NY, USA, 2011. ACM.

[12] Aurick Qiao, Abutalib Aghayev, Weiren Yu, Haoyang Chen, Qirong Ho, Garth A Gibson, and Eric P Xing. Litz: An elastic framework for high-performance distributed machine learning. 2017.

[13] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[14] O. Borů vka. Über ein Minimalproblem. Práce moravské přirodovědecké společnosti 3 (1926), 37-58 (1926)., 1926.

[15] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[16] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.