Journal of Machine Learning Research 25 (2024) 1-45 Submitted 10/22; Revised 11/23; Published 4/24

Margin-Based Active Learning of Multiclass Classifiers *

Marco Bressan MARCO.BRESSAN@UNIMI.IT
Dept. of Computer Science, Università degli Studi di Milano, Italy

Nicolò Cesa-Bianchi NICOLO.CESA-BIANCHI@UNIMI.IT
Dept. of Computer Science, Università degli Studi di Milano, Italy
& Politecnico di Milano, Italy

Silvio Lattanzi SILVIOL@GOOGLE.COM
Google Research

Andrea Paudice ANDREA.PAUDICE@UNIMI.IT
Dept. of Computer Science, Università degli Studi di Milano, Italy

Editor: Samory Kpotufe

Abstract

We study active learning of multiclass classifiers, focusing on the realizable transductive setting. The input is a finite subset X of some metric space, and the concept to be learned is a partition $\mathcal{C}$ of X into k classes. The goal is to learn $\mathcal{C}$ by querying the labels of as few elements of X as possible. This is a useful subroutine in pool-based active learning, and is motivated by applications where labels are expensive to obtain. Our main result is that, in very different settings, there exist interesting notions of *margin* that yield efficient active learning algorithms. First, we consider the case $X \subset \mathbb{R}^m$, assuming that each class has an unknown "personalized" margin separating it from the rest. Second, we consider the case where X is a finite metric space, and the classes are convex with margin according to the geodesic distances in the thresholded connectivity graph. In both cases, we give algorithms that learn $\mathcal{C}$ exactly, in polynomial time, using $\mathcal{O}(\log n)$ label queries, where $\mathcal{O}(\cdot)$ hides a near-optimal dependence on the dimension of the metric spaces. Our results actually hold for or can be adapted to more general settings, such as pseudometric and semimetric spaces.

Keywords: active learning, semimetric space, pseudometric space, convexity, margin

1. Introduction

We study the following active learning problem. Given a finite set X from some domain, we have access to an oracle O_h that, upon receiving $x \in X$, replies with $h(x)$, where $h : X \to [k]$ is an unknown function from some class $\mathcal{H}$. The queries to O_h can be adaptive, that is, each element x queried can depend on the answers to previous queries. The goal is to learn h exactly by making as few queries to O_h as possible. This problem, and active learning more in general, have gained a lot of attention in recent years. The reason is that collecting *unlabeled* data has become relatively cheap, and the most expensive part has become data annotation (which often requires human intervention). As a result, active learning is now a mainstream

*. Preliminary versions of this work appeared in the *Proceedings of the 34th Annual Conference on Learning Theory (COLT),* PMLR 134:775–803, 2021 (Bressan et al., 2021a) and in *Advances in Neural Information Processing Systems 34 (NeurIPS),* 34:25231–25243, 2021 (Bressan et al., 2021b).

©2024 Marco Bressan, Nicolò Cesa-Bianchi, Silvio Lattanzi, Andrea Paudice.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v25/22-1127.html.

field of machine learning, with a rich literature providing algorithms and bounds for several settings (Dasgupta, 2005; Dasgupta et al., 2005; Balcan et al., 2007; Settles, 2012; Balcan and Hanneke, 2012; Balcan and Long, 2013; Gonen et al., 2013; Hanneke, 2014; Dasarathy et al., 2015; Hanneke and Yang, 2015; Wiener et al., 2015; Beygelzimer et al., 2016; Kane et al., 2017; Hopkins et al., 2020a,b, 2021).

It is well known that active learning may provide exponential savings over passive learning. Consider for instance the class of thresholds $\mathcal{H} = \{\mathbb{I}_{x \le t} : t \in [0,1]\}$ over $\mathcal{X} = [0,1]$, and suppose one wants to learn a threshold that, with probability at least $1-\delta$, has error ε against the target threshold t^* with respect to some unknown distribution P over $\mathcal{X}$. In the passive model, one can draw a sample X of $n = \Theta\big(\frac{1}{\varepsilon}\log\frac{1}{\delta}\big)$ points x, obtain the pairs $\big(x, t^*(x)\big)$, and then take any threshold $\widehat{t}$ consistent with the labeled sample. However, if one is allowed to first draw X alone (that is, without the labels) from P, and then to adaptively query the points of X for their labels, then a simple binary search learns a good threshold by looking at only $\mathcal{O}(\log n) = \mathcal{O}\big(\log\frac{1}{\varepsilon} + \log\log\frac{1}{\delta}\big)$ labels. This idea can be applied to any $\mathcal{H}$ and any P. First, draw a set X of $n = n(\varepsilon,\delta)$ i.i.d. unlabeled points from P, where $n(\varepsilon,\delta)$ is the sample complexity of the passive case. Then, learn h exactly over X by making as few queries as possible; ideally only $\mathcal{O}(\log n)$, where the constants hidden in the $\mathcal{O}(\cdot)$ notation can depend on $\mathcal{H}$, $\mathcal{X}$, and other relevant parameters.

Clearly, if $\mathcal{H} = [k]^X$ then learning $\mathcal{H}$ even approximately with non-vanishing probability requires $\Omega(|X|)$ queries. Thus the interesting question is finding useful classes $\mathcal{H}$ that can be learned with $\mathcal{O}(\log n)$ queries over any n-point set X, possibly in polynomial time. The present work addresses precisely this question. As a starting point, consider the classic SVM margin assumption in $\mathbb{R}^m$. For notational convenience, we represent the target classifier $h^* : X \to [k]$ as a partition $\mathcal{C} = (C_1, \ldots, C_k)$ of X. The SVM margin assumption requires every class $C \in \mathcal{C}$ to be separated from $X \setminus C$ by a hyperplane in $\mathbb{R}^m$ with margin $\gamma > 0$. In this case one can learn $\mathcal{C}$ with $\mathcal{O}(\log n)$ queries, where the hidden constants are functions of m and γ (see Bressan et al., 2020). There are two key ingredients that enable such a query rate: the *convexity* of the classes and the *margin* between them. Inspired by these findings, in this work we devise novel, general conditions that allow one to attain a query rate of $\mathcal{O}(\log n)$, with constants that we can often prove to be near-optimal, and in polynomial time.

1.1 Our contributions

We study two settings. The first one considers the classic Euclidean case, $\mathcal{X} = \mathbb{R}^m$, and the more general case of finite pseudometric spaces (where the triangle inequality holds, but the distance between distinct points can be zero). The second setting is inspired by algorithms designed for "dense" classes, such as DBSCAN, and applies to any (finite) semimetric space (where the distance between distinct points is positive, but the triangle inequality may fail), including $\mathbb{R}^m$.

1.1.1 Euclidean spaces and pseudometric spaces

For the case $\mathcal{X} = \mathbb{R}^m$, we introduce a new notion of margin, called *convex hull margin* (Definition 1), enabling the margin of each class to be measured in terms of its own diameter and according to a "personalized" pseudometric. This notion strictly generalizes the spherical margin of Ashtiani et al. (2016), the ellipsoidal margin of Bressan et al. (2020), and (as a

consequence) the SVM margin. Using the convex hull margin, we develop a technique called *convex hull expansion trick*, based on recent results from convex geometry. This technique recovers a constant fraction of any class, with one-sided error, by sampling roughly $\left(1+\frac{1}{\gamma}\right)^{\frac{m-1}{2}}$ of its points. By a simple boosting argument this yields a polynomial-time algorithm that learns all classes using roughly $k^2\left(1+\frac{1}{\gamma}\right)^{\frac{m-1}{2}}\log n$ queries (Theorem 2), without knowing the pseudometrics that yield the margin for each class.[1] The dependence on γ is nearly optimal, as we prove that roughly $\left(1+\frac{1}{\gamma}\right)^{\frac{m-1}{2}}$ queries are necessary in the worst case (Theorem 3).

For the case of finite pseudometric spaces, we generalize the convex hull margin by introducing the *one-versus-all margin* (Definition 11). Interestingly, this notion captures as special cases standard notions of stability for clustering problems such as k-means or k-centers. We show that, if a classifier has one-versus-all margin γ, then it can be learned with $M(\gamma)\operatorname{poly}(k)\log n$ queries via a purely learning-theoretic approach (Theorem 12), where $M(\gamma)$ measures the dimension of the pseudometric space via packing numbers. We also show that the dependence on $M(\gamma)$ is essentially optimal and thus $M(\gamma)$ characterizes the learnability of multiclass classifiers in this setting.

Finally, we study the case when the classes are realized by some concept class $\mathcal{H}$ over $\mathcal{X}$ (that is, when for each class C_i there is a concept $h_i \in \mathcal{H}$ such that $X \cap h_i = C_i$). We show that if a certain combinatorial parameter, the *coslicing dimension* $\operatorname{cosl}(\mathcal{H})$, is bounded, then one can learn $\mathcal{C}$ with $\operatorname{cosl}(\mathcal{H})\operatorname{poly}(k)\log n$ label queries; otherwise, $\Omega(n)$ queries are needed in the worst case (Theorem 21). Moreover we show that, for all concept classes in $\mathbb{R}^m$ that are closed under affine transformations and well-behaved in a natural sense, finite coslicing dimension and positive one-versus-all margin are equivalent (Theorem 22).

1.1.2 Finite semimetric spaces

The second setting assumes X is a finite set and d is a semimetric over X (that is, a metric without the triangle inequality). For instance d could be the Wasserstein distance between images, the Levenshtein distance between strings, the cosine dissimilarity in document analysis, or the Pearson dissimilarity in bioinformatics. In such a general setting, it is not clear how to define notions of convexity or margin for the elements of a partition $\mathcal{C}$. We fill this gap by introducing (β,γ)-*convexity* (Definition 23), a property of classes based on the connectivity graph $\mathcal{G}(\varepsilon)$ where $\{x,y\} \subseteq X$ is an edge iff $d(x,y) \le \varepsilon$. Loosely speaking, a class C is (β,γ)-convex if the subgraph $\mathcal{G}(\varepsilon)[C]$ is connected and closed under taking nearly-shortest paths. When $\mathcal{X} = \mathbb{R}^m$ and d is the Euclidean distance, (β,γ)-convexity captures classes that are very far from being convex, but still natural in an intuitive way.

Our main result is that (β,γ)-convex classifiers can be learned deterministically with $\mathcal{O}\big(k^2\big(\log n + (6/\beta\gamma)^{\operatorname{dens}(X)}\big)\big)$ queries, provided that we know some point (a "seed") from each class (Theorem 24). Here, $\operatorname{dens}(X)$ is the *density dimension* of X (see Gottlieb and Krauthgamer, 2013), a generalization of the doubling dimension that is used to bound the size of packings in metric spaces. This dependence of our exponent on $\operatorname{dens}(X)$ is asymptotically optimal, as we prove that $\Omega(2^{\operatorname{dens}(X)})$ queries are necessary to learn a (β,γ)-convex classifier in the worst case. Thus, $\operatorname{dens}(X)$ plays a role similar to that of the dimensionality m in

1. Any upper bound B on the query cost of our algorithms should be intended as $\min(n,B)$ since obviously one never needs to query the same point twice. A similar observation holds for our running times, which are all polynomial in n,m,k.

Euclidean spaces. The algorithm is completely different from those of Section 1.1.1: rather than sampling and boosting, it computes certain *separators* between each class C and every other class in turn. To do so, it carefully exploits the structural properties of (β, γ)-convexity by analysing shortest paths and edge cuts between different classes in $\mathcal{G}(\varepsilon)$. The algorithm runs in time almost linear in the size of the encoding of (X, d) as a weighted graph.

1.1.3 Enriched queries and learning the parameters

Finally, we investigate the use of more powerful queries for learning the model parameters. In particular we consider a query SEED that, given a set $U \subset X$ and an index $i \in [k]$, either certifies that $C_i \subseteq U$ or returns some $x \in C_i \setminus U$. It is easy to see that k SEED queries are sufficient to test whether any given k-partition $\widehat{\mathcal{C}}$ coincides with $\mathcal{C}$; by an easy halving/guessing argument, this implies one can learn $\mathcal{C}$ without knowing γ (in the first setting) or one of γ and β (in the second setting) by running our algorithms $O\big(k \log \frac{p^\star}{p}\big)$ times where p is the unknown parameter and $p^\star$ a known upper bound on it. For the second setting we give several additional results. First we show that, without seed points, any algorithm needs $\Omega(n)$ label queries to learn $\mathcal{C}$ in the worst case; and if one allows for a different semimetric for each cluster, then $\Omega(n)$ queries are needed to learn $\mathcal{C}$ even if SEED is allowed. Next we show that, with $\mathcal{O}(k^2)$ additional SEED queries, our algorithms can learn classes that are (β, γ)-convex at different connectivity thresholds $\varepsilon_1, \dots, \varepsilon_k$, and thus are dense at different scales (Theorem 26). Moreover we show that $\mathcal{O}(k \log n)$ SEED queries are sufficient and $\Omega(k \log \frac{n}{k})$ are necessary to learn $\varepsilon_1, \dots, \varepsilon_k$ (Theorem 46). Finally, we show how to adapt our algorithms to the case that one or both of β and γ are unknown by using a small number of additional SEED queries and incurring a small computational overhead. Again, these adapted algorithms run in almost linear time.

2. Related work

Ashtiani et al. (2016) introduced semi-supervised active clustering (SSAC), a setting very similar to ours where the oracle supports *same-cluster queries* that reveal whether two given points are in the same class. They showed how to exactly recover the optimal k-means clustering with $\mathcal{O}\big(\operatorname{poly}(k) \log n\big)$ queries when each cluster lies inside a sphere centered in the cluster's centroid and well separated from the spheres of other clusters. Bressan et al. (2020) extended these results to clusters separated by arbitrary ellipsoids with arbitrary centers.[2] Compared to Bressan et al. (2020), our algorithms for the Euclidean case work under strictly more general conditions and reduce the queries by a factor of roughly $m^m(1 + 1/\gamma)^{\frac{m-1}{2}}$. We note that label queries can be simulated (up to a permutation of the classes) with k same-cluster queries, hence our upper bounds work for cluster recovery with an additional multiplicative factor of k. As same-cluster queries can be simulated with two label queries, our lower bounds hold for same-cluster queries as well.

Various notions of margin are central in active learning and cluster recovery (see Xu et al., 2004; Balcan et al., 2007; Balcan and Long, 2013; Kane et al., 2017; Bressan et al., 2021a). Our coslicing dimension is similar to the slicing dimension of Kivinen (1995) and

2. Thanks to their stronger notion of margin, the results and bounds of Ashtiani et al. (2016) are dimension-free, and can also be extended to infinite-dimensional Hilbert spaces.

the star number of Hanneke and Yang (2015). Our lower bounds, like many others, are inspired by a construction by Dasgupta (2005). Our arguments based on packing numbers are similar to those based on the inference dimension of Kane et al. (2017) or the lossless sample compression of Hopkins et al. (2021). A query bound similar to the one given by our convex hull expansion trick, but worse by a factor roughly 2^m, can be inferred by adapting arguments of Hopkins et al. (2020b). Combinatorial characterizations of multiclass learning have been proposed in the passive case by Ben-David et al. (1995), Rubinstein et al. (2009), and Daniely and Shalev-Shwartz (2014). Other learning settings related to one-sided and active learning are RPU learning (Rivest and Sloan, 1988) and perfect selective classification (El-Yaniv and Wiener, 2012)—see Hopkins et al. (2020a) for a discussion.

Concerning learning on graphs, Dasarathy et al. (2015) develop a probabilistic active classification algorithm, called S^2 ($s$hortest-$s$hortest-path), whose label complexity depends on the graph's structure. In particular, the query complexity is linear in the size of the boundary of the edge cut between nodes with different labels. Unfortunately, even under (β, γ)-convexity, in $G_X(\varepsilon)$ this boundary can have size $\Omega(n)$. Thiessen and Gärtner (2021) show a deterministic algorithm with label complexity proportional to the size of the shortest path cover of the graph (the smallest set of shortest paths that cover all nodes). Again, in $G_X(\varepsilon)$ this cover could have size $\Omega(n)$ even under (β, γ)-convexity. Active classification on unweighted graph has been also studied by Afshani et al. (2007), Cesa-Bianchi et al. (2010), and Guillory and Bilmes (2011), but only for approximate learning; to learn exactly, their algorithms may need $\Omega(n)$ queries.

A number of works studied cluster reconstruction under stochastic models using same-cluster queries. Mazumdar and Saha (2017a) prove a logarithmic query bound assuming some latent distribution of similarities between points, while Mazumdar and Saha (2017b) obtain a linear bound when the oracle can err with a certain probability. Stochastic block models (Zhang et al., 2014; Gadde et al., 2016) and geometric block models (Chien et al., 2020) have been also considered as generative models for active clustering on graphs. Finally, there is a rich stream of research on approximating the optimal solution to clustering optimization problems using oracles (see Ailon et al., 2018a,b; Gamlath et al., 2018; Mazumdar and Pal, 2017; Saha and Subramanian, 2019).

SEED and their variants are motivated and used by Hanneke (2009) as *positive example queries*, by Balcan and Hanneke (2012) as *conditional class queries*, and by Beygelzimer et al. (2016) and Attenberg and Provost (2010) as *search queries*. They are also used implicitly by Tong and Chang (2001), Doyle et al. (2011), and Vikram and Dasgupta (2016). It also easy to see that SEED queries are equivalent to the *partial equivalence* queries of Maass and Turán (1992) and to the *subset* plus *superset* queries of Angluin (1988).

3. Preliminaries and notation

Let $\mathcal{X}$ be a set. A function $d : \mathcal{X}^2 \to \mathbb{R}$ is a metric if for all $x, y \in \mathcal{X}^2$ we have (i) $d(x, y) = 0$ iff $x = y$, (ii) $d(x, y) = d(y, x)$, and (iii) $d(x, y) \leq d(x, z) + d(z, y)$ for all $z \in \mathcal{X}$. The function is a pseudometric if (i) is replaced by $d(x, x) = 0$, and is a semimetric if (iii) is removed. Let d be any (pseudo/semi) metric over $\mathcal{X}$. For $x \in \mathcal{X}$ and $r \geq 0$ let $B(x, r, d) = \{y \in \mathcal{X} : d(x, y) \leq r\}$ denote the closed ball of radius r centered at x. For any $S \subseteq \mathcal{X}$, the packing number $\mathcal{M}(S, r, d)$ of S in $\mathcal{X}$ is the maximum size of any $A \subseteq S$ such that $d(x, y) > r$ for all distinct

$x, y \in A$. For any $\eta > 0$ let $\mathcal{M}(\eta, d) = \sup\{\mathcal{M}\big(B(x, r, d), \eta r, d\big) : x \in \mathcal{X}, r > 0\}$. This is the largest packing number of any ball of radius r in $\mathcal{X}$ using balls of radius ηr. Note that $\mathcal{M}(\eta, d) \geq 1$ for all η, d. For any $A \subset \mathcal{X}$ let $\phi_d(A) = \sup_{x,x' \in A} d(x, x')$. For any $U, S \subset \mathcal{X}$ let $d(U, S) = \inf_{x \in U, y \in S} d(U, S)$. For any $X \subset \mathbb{R}^m$ let $\mathrm{conv}(X)$ be the convex hull of X. The unit sphere in $\mathbb{R}^m$ is $S^{m-1} = \{x \in \mathbb{R}^m : \|x\|_2 = 1\}$. We may omit d if clear from the context.

We recall some learning-theoretic facts. A concept class over $\mathcal{X}$ is a family $\mathcal{H} \subseteq 2^{\mathcal{X}}$ of measurable sets.[3] A set $X \subseteq \mathcal{X}$ is shattered by $\mathcal{H}$ if for every $S \subseteq X$ there is $h \in \mathcal{H}$ such that $S = X \cap h$. The Vapnik-Chervonenkis dimension of $\mathcal{H}$, denoted by $\text{vc-dim}(\mathcal{H}, \mathcal{X})$, is the size of the largest such set; we write $\text{vc-dim}(\mathcal{H})$ when $\mathcal{X}$ is clear from the context. The intersection class of $\mathcal{H}$ is $I(\mathcal{H}) = \bigcup_{i \in \mathbb{N}} \{h_1 \cap \ldots \cap h_i : h_1, \ldots, h_i \in \mathcal{H}\}$. For a given $X \subseteq \mathcal{X}$, the smallest concept in $I(\mathcal{H})$ consistent with X is $\bigcap\{h \in \mathcal{H} : h \cap \mathcal{X} = X\}$. An algorithm $\mathcal{A}$ learns $\mathcal{H}$ with one-sided error ε and confidence δ with r examples if, for any target concept $h^\star \in \mathcal{H}$ and any probability measure $\mathcal{P}$ over $\mathcal{X}$, by drawing r independent labeled examples from $\mathcal{P}$ the algorithm outputs a concept $h \subseteq h^\star$ such that $\mathcal{P}(h^\star \setminus h) \leq \varepsilon$ with probability at least $1 - \delta$.

The input to our problem depends on the setting. For the Euclidean setting, the input set is $X \subset \mathcal{X}$ where $\mathcal{X} = \mathbb{R}^m$. For the finite semimetric setting, the input is a weighted graph $\mathcal{G} = (X, \mathcal{E}, d)$ encoding the semimetric d over X. In both cases, the input also contains a pointer to a labeling oracle $O_{\mathcal{C}}$ consistent with a k-partition $\mathcal{C} = (C_1, \ldots, C_k)$ of X, so that $O_{\mathcal{C}}(x) = i$ if and only if $x \in C_i$. In line with previous work we assume k is known and we allow for empty classes. By "high probability" we mean a probability that, for any $a > 0$, can be made higher than $1 - n^{-a}$ by increasing the cost by no more than a multiplicative factor ca for some universal constant $c > 0$. Further definitions and notation are given when needed.

4. Learning classifiers in Euclidean and pseudometric spaces

In this section we consider the problem of learning a k-classifier over a finite input set X from some space $\mathcal{X}$ endowed with a number of metrics or pseudometrics. We start with the special case $\mathcal{X} = \mathbb{R}^m$ with pseudometrics induced by seminorms, and then we generalize to more abstract settings. The input to our problem always contains the pair $(X, O_{\mathcal{C}})$, and possibly some additional information, such as the margin γ. Depending on the setting, the algorithm may know or may not know the pseudometrics relevant to the classifier.

4.1 The convex hull margin in $\mathbb{R}^m$

Let $X \subset \mathbb{R}^m$ and let $\mathcal{C}$ be a k-partition of X. We introduce a novel notion of margin, called *convex hull margin*, which captures the idea that points not in a certain class should be well separated from that class in terms of the class' diameter. Instead of using the Euclidean metric, however, we allow distances to be measured by any pseudometric in $\mathbb{R}^m$, which the algorithm may not know, and which may be specific to each class. We only require that the pseudometric be homogeneous and invariant under translation (that is, that the pseudometric be induced by a seminorm).

3. We will interchangeably consider hypotheses as functions from $\mathcal{X}$ to $\{0, 1\}$ and as subsets of $\mathcal{X}$.

Definition 1 (Convex hull margin) *Let D be the family of all pseudometrics induced by seminorms over $\mathbb{R}^m$, and let $X \subset \mathbb{R}^m$ be a finite set. A k-partition $\mathcal{C} = (C_1, \ldots, C_k)$ of X has convex hull margin γ if for every $i \in [k]$ there exists $d_i \in D$ such that:*

$$d_i\big(X \setminus C_i, \mathrm{conv}(C_i)\big) > \gamma\, \phi_{d_i}(C_i)$$

Definition 1 has some interesting properties. First, it generalizes both the ellipsoidal margin of Bressan et al. (2020) and the spherical margin of Ashtiani et al. (2016), as D contains the pseudometric induced by $\|Wx\|_2$ for every positive semidefinite matrix $W \succeq 0$, and it generalizes also the classic SVM margin—see Lemma 4 below. Second, pseudometrics are versatile and can express, for instance, the Euclidean distance after a projection on a subspace, modeling scenarios where each class only "cares" about a certain subset of the features.

Our main contribution is the design and analysis of CheatRec (for Convex Hull ExpAnsion Trick Recovery), a polynomial-time algorithm for actively learning k-classifiers with positive convex hull margin.

Theorem 2 *Let $\mathcal{C}$ be a k-partition of $X \subset \mathbb{R}^m$ with convex hull margin $\gamma > 0$. Then* CheatRec$(X, O_{\mathcal{C}}, \gamma)$ *outputs $\mathcal{C}$, runs in time* $\mathrm{poly}(k, n, m)$*, and with high probability makes a number of label queries to $O_{\mathcal{C}}$ bounded by $k^2\, 2^{\mathcal{O}(m)}\, (1 + 1/\gamma)^{\frac{m-1}{2}} \log(1 + 1/\gamma) \log n$.*

The key ingredient behind CheatRec is the *convex hull expansion trick*, a technique that enables one-sided-error learning without knowing the pseudometrics that define the margin. Section 4.1.1 describes this technique and contains the proof of Theorem 2.

To put Theorem 2 in perspective, consider the algorithm of Bressan et al. (2020). Under an ellipsoidal margin of γ_{EL}, that algorithm achieves a query bound of roughly $\big(\frac{m}{\gamma_{EL}}\big)^m \log n$. One can check that an ellipsoidal margin of γ_{EL} implies a convex hull margin of $\gamma \geq \frac{\gamma_{EL}}{3}$ for all $\gamma_{EL} \leq 1$.[4] Hence, in this range, our dependence on γ is better by factors $(c_1 m)^m (c_2/\gamma)^{\frac{m+1}{2}}$ for some $c_1, c_2 > 0$. In fact, our dependence is nearly optimal:

Theorem 3 *For some $a > 0$ the following claim holds. For all $\gamma \in \big(0, \frac{1}{5}\big]$, all $m \geq 2$, and every (randomized) algorithm $\mathcal{A}$, there exists an instance $(X, O_{\mathcal{C}})$ with $X \subset \mathbb{R}^m$ such that: $|X| \geq \big(\frac{1+\gamma}{6\gamma}\big)^{\frac{m-1}{2}}$, $|\mathcal{C}| = 2$, $\mathcal{C}$ has convex hull margin γ, and, in order to return $\mathcal{C}$ with constant probability, $\mathcal{A}$ must make at least $a|X|$ queries to $O_{\mathcal{C}}$ in expectation.*

Theorem 3 is proven in Section 4.1.2. Between Theorem 3 and Theorem 2 there is a gap of at most $2^{\mathcal{O}(m)}\, \mathrm{poly}(k) \log(1 + \frac{1}{\gamma}) \log n$. We believe removing the $2^{\mathcal{O}(m)}$ factor requires deriving tight bounds on $\mathcal{M}(S^{m-1}, \gamma)$. We also note that, while the definition of convex hull margin could be generalized to arbitrary normed spaces, our algorithm CheatRec relies on several technical results (Section 4.1.1) whose generalizability beyond $\mathbb{R}^m$ is unclear.

Before continuing, we show that the convex hull margin generalizes the SVM margin. For any two sets $C_1, C_2 \in \mathbb{R}^m$ define their multiplicative SVM margin as

$$\frac{\sup_{u \in S^{m-1}(0,1)} \inf_{x \in C_1, y \in C_2} |\langle u, x - y \rangle|}{\sup_{x,y \in C_1 \cup C_2} \|x - y\|}$$

Note that this is precisely the usual SVM margin after scaling $C_1 \cup C_2$ so to have radius 1.

4. Their definition uses squared distances, so the relationship with our margin is $1 + \gamma \geq \sqrt{1 + \gamma_{EL}}$.

Lemma 4 *If C_1, C_2 have multiplicative SVM margin γ, then they have convex hull margin γ. Moreover, for every $\eta \in (0,1)$ there exists a partition $\mathcal{C} = (C_1, C_2)$ of a set $X \subset \mathbb{R}^2$ having multiplicative SVM margin at most η but convex hull margin γ for all $\gamma > 0$.*

Proof The first claim is immediate from the definition of multiplicative SVM margin. For the second claim let $X = C_1 \cup C_2$ where $C_1 = \{(\eta, 0), (1, 0)\}$ and $C_2 = \{(0, \eta), (0, 1)\}$. One can check that the multiplicative SVM margin between C_1 and C_2 is precisely η. Now, for any $x, y \in C$ let $d_1(x, y) = |x_2 - y_2|$ and $d_2(x, y) = |x_1 - y_1|$. Clearly, both d_1 and d_2 are induced by seminorms over $\mathbb{R}^2$, and $d_1(C_1, C_2) = d_2(C_1, C_2) > 0$ while $\phi_{d_1}(C_1) = \phi_{d_2}(C_2) = 0$. Hence $d_1(C_1, C_2) > \gamma\phi_{d_1}(C_1)$ and $d_2(C_1, C_2) > \gamma\phi_{d_2}(C_2)$ for all $\gamma > 0$. ■

4.1.1 CheatRec and proof of Theorem 2

The idea behind CheatRec is to boost learning with one-sided error. Suppose we draw a sample S_C of points from some class C that has convex hull margin γ with respect to a pseudometric d. Using S_C we would like to find out, say, half of C, by taking all points that are close to S_C under d. If we could to this, then we could learn $\mathcal{C}$ exactly by repeating the process $\mathcal{O}(k \log n)$ times, since there are k classes with at most n points each. Unfortunately, this approach has two problems. First, we do not know d. Second, even if we knew d, there may be very few points close to S_C. We show that both problems can be overcome as follows. Let $K = \mathrm{conv}(S_C)$, choose any $z \in K$, and let Q be the scaling of K about z by a factor $1 + \gamma$. Since d is homogeneous and invariant under translation (recall that d is induced by a seminorm), then any $y \in X$ such that $d(y, K) \leq \gamma\phi_d(K)$ is in C as well, hence $X \cap Q \subseteq C$. This solves the first problem. The second problem is less trivial and requires choosing carefully $|S_C|$ and z. We prove that, if $|S_C|$ is large enough and z is (approximately) the center of mass of K, then Q contains half of C with good probability. We call this the *convex hull expansion trick*.

Before continuing, we need some more definitions. Without loss of generality assume K has full rank (otherwise use the subspace spanned by S_C, which can be computed in time $\mathcal{O}(|S_C|\, m)$). Let $R \subseteq \mathbb{R}^m$ be any convex body. We denote by $\mu_R = \int_R x\, dx$ and ∂R respectively the center of mass and the boundary of R. We say R is in *isotropic position*[5] if $\mu_R = 0$ and $\int_R \langle x, u\rangle^2\, dx = 1$ for all $u \in S^{m-1}$. We let $\|\cdot\|_R = \|f_R(\cdot)\|_2$ where f_R is the unique invertible affine transformation such that $f_R(R)$ is in isotropic position, and let $d_R(x, y) = \|x - y\|_R$ for all $x, y \in \mathbb{R}^m$. For any two points $x, y \in \mathbb{R}^m$ and any $\alpha > 0$, let $\sigma(x, y, \alpha) = y + \alpha(x - y)$ be the scaling of x by a factor of α w.r.t. y, and for any $A \subset \mathbb{R}^m$ let $\sigma(A, y, \alpha) = \cup_{x \in A}\sigma(x, y, \alpha)$.

Lemma 5 *Suppose $C \subset \mathbb{R}^m$ has convex hull margin $\gamma > 0$. Let S_C be a sample of s independent uniform random points from C for $s = 2^{\mathcal{O}(m)}\big(1 + \frac{1}{\gamma}\big)^{\frac{m-1}{2}} \log_2\big(1 + \frac{1}{\gamma}\big)$ large enough, let $z \in K = \mathrm{conv}(S_C)$ be a (random) point such that $\mathbb{P}\left(d_K(z, \mu_K) \leq \frac{1}{2}\right) \geq \frac{3}{4}$, and let $Q = \sigma(K, z, 1 + \gamma)$. Then $Q \cap (X \setminus C) = \emptyset$ and $\mathbb{P}\big(|Q \cap C| \geq |C|/2\big) \geq \frac{1}{2}$.*

Proof We use the following results (slightly rephrased from their original version):

5. Not to be confused with the definition of Giannopoulos (2003), where the assumption $\int_R \langle x, u\rangle^2\, dx = 1$ is replaced by $\mathrm{vol}(R) = 1$.

Lemma 6 (Kannan et al. 1995) *If $R \subset \mathbb{R}^m$ is a convex body in isotropic position, then*

$$\sqrt{\frac{m+1}{m}}B(0,1) \subseteq R \subseteq \sqrt{m(m+1)}B(0,1)$$

Theorem 7 (Naszódi et al. 2020) *For every convex body $R \subset \mathbb{R}^m$ and every $\gamma > 0$ there is a polytope $P \subset \mathbb{R}^m$ on $2^{\mathcal{O}(m)}\big(1+\frac{1}{\gamma}\big)^{\frac{m-1}{2}}$ vertices such that $R \subset P \subset \sigma(R, \mu_R, 1+\gamma)$.*

Theorem 8 (Kupavskii 2020) *The VC dimension of the family of t-vertex polytopes in $\mathbb{R}^m$ is at most $8m^2 t \log_2 t$.*

First, let us prove that $Q \cap (X \setminus C) = \emptyset$. Let d be any pseudometric that is homogeneous and invariant under translation. Since $z \in K$, then any $y \in \sigma(K, z, 1+\gamma) \cap X$ satisfies $d(y, K) \leq \gamma\, \phi_d(K)$. But $K \subseteq \mathrm{conv}(C)$, therefore $\phi_d(K) \leq \phi_d(C)$. Hence $d(y, \mathrm{conv}(C)) \leq \gamma\, \phi_d(C)$. By the convex hull margin this implies $y \in C$. Hence $Q \cap X \subseteq C$, which in turn implies $Q \cap (X \setminus C) = \emptyset$.

Next we prove that $\mathbb{P}\big(d_K(z, \mu_K) \leq \frac{1}{2}\big) \geq \frac{3}{4}$ implies $\mathbb{P}\big(|Q \cap C| \geq |C|/2\big) \geq \frac{1}{2}$. Let $t = 2^{\mathcal{O}(m)}\big(1+\frac{1}{\gamma}\big)^{\frac{m-1}{2}}$ large enough and let $\mathcal{P}_{t,m}$ be the family of all polytopes on at most t vertices in $\mathbb{R}^m$. By Theorem 8,

$$\text{vc-dim}(\mathcal{P}_t) = 2^{\mathcal{O}(m)}\left(1+\frac{1}{\gamma}\right)^{\frac{m-1}{2}} \log\left(1+\frac{1}{\gamma}\right)$$

Thus, if $s = 2^{\mathcal{O}(m)}\left(1+\frac{1}{\gamma}\right)^{\frac{m-1}{2}} \log\left(1+\frac{1}{\gamma}\right)$ is large enough and S_C is a sample of s independent uniform points from C, then—by standard PAC bounds—any $P \in \mathcal{P}_{t,m}$ that is consistent with S_C (i.e., such that $P \supset S_C$) satisfies $\mathbb{P}\big(|P \cap C| \geq \frac{1}{2}|C|\big) \geq \frac{3}{4}$. It remains to show that, if $d_K(z, \mu_K) \leq \frac{1}{2}$, then there exists such a P that satisfies $P \subset Q$; the claim then follows by a union bound.

By Theorem 7 there is $P \in \mathcal{P}_{t,m}$ such that $K \subseteq P \subseteq \sigma\big(K, \mu_K, 1+\frac{\gamma}{2}\big)$; thus we only need to prove that $\sigma\big(K, \mu_K, 1+\frac{\gamma}{2}\big) \subset \sigma(K, z, 1+\gamma)$. Observe that, to this end, it is sufficient to show that $d_K\big(\partial\sigma(K, z, 1+\gamma), \sigma(K, \mu_K, 1+\frac{\gamma}{2})\big) > 0$, since $z \in K$ as implied by Lemma 6 and by $d_K(z, \mu_K) \leq \frac{1}{2}$. Now, by the triangle inequality:

$$\begin{aligned} d_K\Big(\partial\sigma(K, z, 1+\gamma), \sigma\Big(K, \mu_K, 1+\frac{\gamma}{2}\Big)\Big) \geq\ & d_K\Big(\partial\sigma(K, \mu_K, 1+\gamma), \sigma\Big(K, \mu_K, 1+\frac{\gamma}{2}\Big)\Big) \\ & - d_K\big(\partial\sigma(K, \mu_K, 1+\gamma), \partial\sigma(K, z, 1+\gamma)\big) \end{aligned}$$

On the one hand, by standard arguments and by Lemma 6,

$$d_K\Big(\partial\sigma(K, \mu_K, 1+\gamma), \sigma\Big(K, \mu_K, 1+\frac{\gamma}{2}\Big)\Big) \geq \frac{\gamma}{2} \inf_{x \in \partial K} \|x\|_K \geq \frac{\gamma}{2}\sqrt{\frac{m+1}{m}} > \frac{\gamma}{2}$$

On the other hand, since $\sigma(K, z, 1+\gamma) = \sigma(K, \mu_K, 1+\gamma) + \gamma(z - \mu_K)$,

$$d_K\left(\partial\sigma(K, \mu_K, 1+\gamma), \partial\sigma(K, z, 1+\gamma)\right) \leq \gamma\|z - \mu_K\|_K = \gamma\, d_K(z, \mu_K)$$

Therefore

$$d_K\Big(\partial\sigma(K, z, 1+\gamma), \sigma(K, \mu_K, 1+\frac{\gamma}{2})\Big) > \gamma\left(\frac{1}{2} - d_K(z, \mu_K)\right)$$

which is positive for $d_K(z, \mu_K) \leq \frac{1}{2}$, concluding the proof. ■

Next, we give a polynomial-time implementation of the convex hull expansion trick of Lemma 5, see Algorithm 1. The minimum volume enclosing ellipsoid (MVEE) of a set $P \subset \mathbb{R}^m$ is an ellipsoid $E \subseteq \mathbb{R}^m$ of smallest volume such that $P \subseteq E$. A convex body $P \subseteq \mathbb{R}^m$ is in *near-isotropic position* if the ratio between the smallest radius of a ball containing P and the largest radius of a ball contained in P is at most m. By *hit-and-run* we mean the hit-and-run algorithm of Lovász and Vempala 2006.

Algorithm 1 ExpandHull$(S_C, 1+\gamma)$

let $N = \Omega(m^2)$ large enough and $\varepsilon = \mathcal{O}(m^{-1})$ small enough
compute the MVEE of $\mathrm{conv}(S_C)$ and put $\mathrm{conv}(S_C)$ in near-isotropic position
for $i = 1, \ldots, N$ **do**
    X_i = output of $\mathrm{poly}(m/\varepsilon)$ hit-and-run steps over $\mathrm{conv}(S_C)$ starting from 0
let $z = \frac{1}{N}\sum_{i=1}^{N} X_i$
return $\sigma(S_C, z, 1+\gamma)$

Lemma 9 ExpandHull$(S_C, 1+\gamma)$ *can be implemented to run in time* $\mathrm{poly}(|S_C|, m)$. *Moreover, its output* $\sigma(S_C, z, 1+\gamma)$ *satisfies* $\mathbb{P}\left(d_K(z, \mu_K) \leq \frac{1}{2}\right) \geq \frac{3}{4}$ *where* $K = \mathrm{conv}(S_C)$.

Proof We start with the running time bound. Let $K = \mathrm{conv}(S_C)$. Computing the MVEE E of K takes time $|S_C|m^2\big(\ln m + \ln\ln|S_C|\big)$, see Khachiyan (1996). Putting K in near-isotropic position can be done by computing the affine transformation that maps E into the unit ball $B(0,1)$, which clearly takes time $\mathrm{poly}(m)$. At that point by John's theorem (John, 1985) we have $B(0, 1/m) \subseteq K$, hence 0 is at distance at least $\frac{1}{m}$ from ∂K. If we then execute hit-and-run from 0, we have:

Lemma 10 (see Lovász and Vempala 2006, Corollary 1.2) *For any* $\varepsilon > 0$, *the distribution of the random walk after* $t = \Theta\left(m^5 \ln\frac{m}{\varepsilon}\right)$ *steps is* ε*-uniform over* K.

It remains to implement hit-and-run in time $\mathrm{poly}(|S_C|, m)$. To this end we need to solve the following problem: given any $x \in K$ and any $u \in S^{n-1}$, determine the intersection of the ray $\{x + \alpha u\}_{\alpha \geq 0}$ with ∂K. This amounts to finding the maximum $\alpha \geq 0$ such that $x + \alpha u$ is in $\mathrm{conv}(S_C)$, which can be done via linear programming in time $t_K = \mathrm{poly}(|S_C|, m)$ with polynomial precision. Summarizing, the total time to draw $X_1, \ldots, X_N$ and compute z is:

$$\mathcal{O}\left(|S_C|m^2\big(\ln m + \ln\ln|S_C|\big) + N t_K\, m^5 \ln\frac{m}{\varepsilon}\right)$$

Setting $N = \mathcal{O}(m^2)$, $\varepsilon = \mathcal{O}(m^{-1})$, and $t_K = \mathrm{poly}(|S_C|, m)$ yields a bound of $\mathrm{poly}(|S_C|, m)$.

We now prove that $\sigma(S_C, z, 1+\gamma)$ satisfies $\mathbb{P}\left(d_K(z, \mu_K) \leq \frac{1}{2}\right) \geq \frac{3}{4}$. Recall that the uniform probability measure $\mathcal{U}$ over a body K is defined by $\mathcal{U}(K') = \frac{\mathrm{vol}(K')}{\mathrm{vol}(K)}$ for all measurable $K' \subseteq K$.

A probability measure $\mathcal{P}$ is ε-uniform if $|\mathcal{P}(K') - \mathcal{U}(K')| \leq \varepsilon$ for all measurable $K' \subseteq K$. Fix $\eta > 0$, $p > 0$, $\varepsilon \leq \frac{\eta}{4(m+1)}$, and $N \geq \frac{16(m+1)^2}{p^2\eta^2}$. We show that, if $X_1, \ldots, X_N$ are independent ε-uniform points from K and $\overline{X} = \frac{1}{N}X_i$, then:

$$\mathbb{P}\big(d_K\big(\overline{X}, \mu_K\big) \leq \eta\big) \geq 1 - p \tag{1}$$

which, for $\eta = \frac{1}{2}$ and $p = \frac{1}{4}$, implies the claim above since—by construction—$z = \overline{X}$. To prove (1) we need two ancillary results. Define $R_K(K) = \sup_{x \in K} \|x\|_K$. First,

$$R_K(K) \leq m + 1 \tag{2}$$

Consider indeed K in isotropic position, and let $K' = \vartheta K$ where $\vartheta = \mathrm{vol}(K)^{-1/m}$, so that $\mathrm{vol}(K') = 1$. Then, K' is in isotropic position according to the definition of Giannopoulos (2003). Define $R(K') = \sup_{x \in K'} \|x\|$. Theorem 1.2.4 of Giannopoulos (2003) implies $R(K') \leq (m+1)L_K$, where L_K is the *isotropic constant* which, for all $u \in S^{m-1}$, satisfies $\int_{K'} \langle x, u\rangle^2 \, dx = L_K^2$. Since $K' = \vartheta K$ and $\int_K \langle x, u\rangle^2 \, dx = 1$ by the isotropy of K, we have $L_K = \vartheta$. Hence $R(K') \leq (m+1)\vartheta$, that is, $R_K(K) \leq m + 1$.

Second, we claim that if X is ε-uniform over K then $\|\mathbb{E}X\|_K \leq 2\varepsilon(m+1)$. Indeed, since X is ε-uniform over K, then there exists a coupling (X, Y) with $\mathbb{P}(X \neq Y) \leq \varepsilon$ and Y uniform over K. Since $\|\mathbb{E}Y\|_K = 0$, we have:

$$\|\mathbb{E}X\|_K = \|\mathbb{E}[X - Y]\|_K \leq \mathbb{P}(X \neq Y) \sup_{x,y \in K} d_K(x, y) \leq \varepsilon \, 2R_K(K) \leq 2\varepsilon(m+1)$$

where the last inequality is given by (2).

We can now prove (1). Consider K in isotropic position; clearly the X_i are still independent and ε-uniform over K, and (1) becomes $\mathbb{P}\big(\|\overline{X}\|_2 \leq \eta\big) \geq 1 - p$. As $\|\overline{X}\|_2 \leq \|\mathbb{E}\overline{X}\|_2 + \|\overline{X} - \mathbb{E}\overline{X}\|_2$, proving that $\|\mathbb{E}\overline{X}\|_2 \leq \frac{\eta}{2}$ and $\mathbb{P}(\|\overline{X} - \mathbb{E}\overline{X}\|_2 \leq \frac{\eta}{2}) \geq 1 - p$ is sufficient. For the first part, by the above bound on $\|\mathbb{E}X\|_K$ and since $\varepsilon \leq \frac{\eta}{4(m+1)}$,

$$\|\mathbb{E}\overline{X}\|_2 = \|\mathbb{E}X_i\|_2 \leq 2\varepsilon(m+1) \leq \frac{\eta}{2}$$

For the second part, by (2) $\|X_i\|_2 \leq m + 1$ for all i. Therefore, letting $Y_i = X_i - \mathbb{E}X_i$ for all i, we have $\|Y_i\|_2 \leq 2(m+1)$, and thus $\|Y_i\|_2^2 \leq 4(m+1)^2$. Let $\overline{Y} = \frac{1}{N}\sum_{i=1}^N Y_i$. Since the Y_i are independent with $\mathbb{E}Y_i = 0$, then $\mathbb{E}\langle Y_i, Y_j\rangle = 0$ whenever $i \neq j$ and therefore:

$$\mathbb{E}\|\overline{Y}\|_2^2 = \mathbb{E}\bigg(\frac{1}{N^2}\sum_{i,j=1}^N \langle Y_i, Y_j\rangle\bigg) = \frac{1}{N^2}\sum_{i=1}^N \mathbb{E}\|Y_i\|_2^2 \leq \frac{4(m+1)^2}{N}$$

Using $N \geq \frac{16(m+1)^2}{p^2\eta^2}$ and Jensen's inequality yield $\big(\mathbb{E}\|\overline{Y}\|_2\big)^2 \leq \mathbb{E}\|\overline{Y}\|_2^2 \leq \frac{p^2\eta^2}{4}$. Therefore $\mathbb{E}\|\overline{Y}\|_2 \leq \frac{p\eta}{2}$, which by Markov's inequality implies $\mathbb{P}\big(\|\overline{Y}\|_2 > \frac{\eta}{2}\big) < p$. Substituting $\overline{Y}$ with $\overline{X} - \mathbb{E}\overline{X}$ completes the proof of (1). ■

We can now prove Theorem 2 by analysing the pseudocode in Algorithm 2. Let a *round* be a single iteration of the **while** loop in CheatRec. For the correctness just note that

Algorithm 2 CheatRec$(X, O_{\mathcal{C}}, \gamma)$

let $s = 2^{\mathcal{O}(m)}\left(1+\frac{1}{\gamma}\right)^{\frac{m-1}{2}} \ln\left(1+\frac{1}{\gamma}\right)$ large enough
while $X \neq \emptyset$ **do**
  if $|X| \leq ks$ **then** query $O_{\mathcal{C}}$ to label each point in X and **return**
  draw ks i.i.d. uniform random points S from X
  learn the labels of S by querying $O_{\mathcal{C}}$
  for $i = 1, \ldots, k$ **do**
    let S_i be the points of S having label i
    $S_i^* = \textsc{ExpandHull}(S_i, 1+\gamma)$
    compute $\widehat{C}_i = \mathrm{conv}(S_i^*) \cap X$ via LP
    label all points of $\widehat{C}_i$ with label i
    $X = X \setminus \widehat{C}_i$

$\widehat{C}_i \subseteq C_i$ for all $i \in [k]$ by Lemma 9. For the query bound, since $|S| = ks$ some $i \in [k]$ satisfies $|S_i| \geq s$, so by Lemma 9 with probability at least $\frac{1}{2}$ we have $|\widehat{C}_i \cap X| \geq \frac{|C_i|}{2}$. As shown in Lemma 3 of Bressan et al. (2020), this implies that CheatRec performs more than $4k \log n + 6a\sqrt{k} \log n$ rounds with probability at most n^{-a} for all $a > 0$. This in turn implies the query bound by substituting s and noting that each round makes ks queries. For the running time note that drawing S, asking for all labels, and computing the sets S_i requires time $\mathrm{poly}(n, m)$. By Lemma 9, each call to ExpandHull costs time $\mathrm{poly}(|S_i|, m)$. Computing $\mathrm{conv}(S_i^*) \cap X$ amounts to solving a polytope membership problem for each $x \in X$, which takes time $\mathrm{poly}(|S_i^*|, m)$ via linear programming. Since at each round at least one new point is labeled, the algorithm makes at most n rounds, proving that CheatRec runs in time $\mathrm{poly}(n, m) = \mathrm{poly}(n, k, m)$.

4.1.2 Proof of Theorem 3

Let $\varepsilon = \sqrt{5\gamma}$; clearly $\varepsilon \in (0, 1]$ by the assumption on γ. Let $B \subset \mathbb{R}^{m-1}$ be the unit $(m-1)$-dimensional ball centered at the origin, and Y be an ε-packing of B of maximum size. It is well known that $|Y| \geq \left(\frac{1}{\varepsilon}\right)^{m-1}$, see Vershynin (2018). With our choice of ε this gives:

$$|Y| \geq \left(\frac{1}{5\gamma}\right)^{\frac{m-1}{2}} \geq \left(\frac{1+\gamma}{6\gamma}\right)^{\frac{m-1}{2}}$$

where the second inequality comes from $\gamma \leq \frac{1}{5}$. Let $f : Y \to \mathbb{R}^m$ be defined by:

$$f(y_1, \ldots, y_{m-1}) = (\sqrt{1 - \|y\|^2}, y_1, \ldots, y_{m-1})$$

for all $y = (y_1, \ldots, y_{m-1}) \in Y$. Define $X = \{f(y) : y \in Y\}$, and let d be the Euclidean distance. Note that $d(f(y), f(y')) \geq d(y, y')$ for all $y, y' \in Y$. Thus $X \subseteq S^{m-1}$ is an ε-packing of S^{m-1} of size $n \geq \left(\frac{1+\gamma}{6\gamma}\right)^{\frac{m-1}{2}}$.

Now choose any $x \in X$. Let $C_1 = \{x\}$, $C_2 = X \setminus C_1$, and $\mathcal{C} = (C_1, C_2)$. We claim that both C_1 and C_2 have convex hull margin γ under d. For C_1 this is trivial since $\phi_d(C_1) = 0$ and $d(X \setminus C_1, \mathrm{conv}(C_1)) > 0$. For C_2, consider any $x' \in X \setminus \{x\}$ and let p be the projection

of x' on the subspace span(x) spanned by x. As the hyperplane orthogonal to span(x) and containing p separates $X \setminus C_2$ and C_2,

$$d(X \setminus C_2, \mathrm{conv}(C_2)) \geq d(x, p)$$

Let θ be the angle between x and x'. Standard trigonometric arguments show that $d(x,p) = d(x,x') \sin\frac{\theta}{2}$, and since $\theta \geq d(x,x') > \varepsilon$, then $d(x,p) > \varepsilon \sin\frac{\varepsilon}{2}$. As $\sin a > 0.8\, a$ for all $a \in (0, 1/2]$, then $d(x,p) > 0.4\,\varepsilon^2$. We conclude that $d(X \setminus C_2, \mathrm{conv}(C_2)) > 0.4\,\varepsilon^2$ too. Since $\phi(C_2) < 2$, the convex hull margin of C_2 is at least $0.2\,\varepsilon^2 = \gamma$.

Now consider an instance $(X, O_{\mathcal{C}})$ obtained by drawing x uniformly at random from X and letting $\mathcal{C} = (X \setminus \{x\}, \{x\})$ as described above. Clearly, any deterministic algorithm that returns $\mathcal{C}$ with constant probability must make at least $a|X|$ queries to $O_{\mathcal{C}}$ for some universal $a > 0$. By Yao's principle (see Motwani and Raghavan 1995) this implies the claim.

4.2 The one-versus-all margin

This section generalizes the setting of Section 4.1 by letting $\mathcal{X}$ be a generic set equipped with a set of pseudometrics. As in Section 4.1, we want a notion of margin between classes. However, we cannot express the margin in terms of diameter of convex hulls, since we do not assume $\mathcal{X}$ to be a vector space or to have any notion of convexity. Thus, we introduce a notion called *one-versus-all margin*. We prove that classifiers with positive one-versus-all margin can be learned with $\mathcal{O}(\log n)$ oracle queries. However, we do not provide a running time analysis; the implementation of our algorithm depends on $\mathcal{X}$ and on properties of $\mathcal{C}$, and in general may take time superpolynomial in the size of the input.

Definition 11 (One-versus-all margin) *Fix k pseudometrics $d_1, \ldots, d_k$ over $\mathcal{X}$. A partition $\mathcal{C} = (C_1, \ldots, C_k)$ of a finite set $X \subset \mathcal{X}$ has one-versus-all margin γ with respect to $d_1, \ldots, d_k$ if, for all $i \in [k]$, we have $d_i(X \setminus C_i, C_i) > \gamma\, \phi_{d_i}(C_i)$.*

Note that, unlike the convex hull margin (Definition 1), here $d_1, \ldots, d_k$ are fixed in advance. The reason is that, as said above, $\mathcal{X}$ may not be a linear space, $C_1, \ldots, C_k$ may not be convex, and $d_1, \ldots, d_k$ may not be induced by seminorms. These properties appear to be crucial for learning $\mathcal{C}$ without knowing $d_1, \ldots, d_k$ (see Section 4.1). Note that if a clustering $\mathcal{C}$ in $\mathbb{R}^m$ has convex hull margin γ then it has one-versus-all margin γ, too, since $d_i(X \setminus C_i, C_i) \geq d_i(X \setminus C_i, \mathrm{conv}(C_i)) > \gamma\, \phi_{d_i}(C_i)$. In this sense the one-versus-all margin is a generalization of the convex hull margin.

Under the one-versus-all margin we prove two main results. First, in Section 4.2.1 we prove that the one-versus-all margin subsumes well-known notions of *stability* that yield polynomial-time algorithms for center-based clusterings, such as k-means or k-medians. Second, in Section 4.2.2 we give an algorithm, mRec, that learns $\mathcal{C}$ with a query rate of $\mathcal{O}(\log n)$. Formally, recalling $\mathcal{M}(\gamma, \cdot)$ from Section 3 and letting $M(\gamma) = \max_{i \in [k]} \mathcal{M}(\gamma, d_i)$, we prove: We prove:

Theorem 12 *Let $\mathcal{C}$ be a k-partition of X with one-versus-all margin $\gamma > 0$ with respect to $d_1, \ldots, d_k$. Then* mRec$(X, O_{\mathcal{C}}, \gamma, d_1, \ldots, d_k)$ *outputs $\mathcal{C}$ while making $\mathcal{O}\big(M(\gamma)\, k \log k \log |X|\big)$ queries to $O_{\mathcal{C}}$ with high probability. Moreover, for some universal $b > 0$, every (randomized) algorithm that learns classifiers with one-versus-all margin $\gamma > 0$ makes at least $b\, M(2\gamma)$ label queries in the worst case.*

Due to the generality of the setting, mRec is based on purely learning-theoretical arguments. Thus, unlike what we did for CheatRec (Section 4.1), we do not prove running time bounds for mRec, as they depend on the particular semimetric space at hand.

4.2.1 One-versus-all-margin and stability of center-based clusterings

Fix any pseudometric d over $\mathcal{X}$. A partition $\mathcal{C} = (C_1, \dots, C_k)$ of X is a *center-based clustering* if there exist k points $c_1, \dots, c_k \in \mathcal{X}$, called *centers*, such that for every $i \in [k]$ and every $x \in C_i$ we have $d(x, c_j) > d(x, c_i)$ for all $j \neq i$. In words, every point is assigned to the nearest center. It is well known that computing center-based clusterings that minimize objective functions such as k-means or k-centers is NP-hard in general (Cohen-Addad and Srikanta, 2019). However, those problems become polynomial-time solvable if the optimal solution $\mathcal{C}$ meets certain *stability* properties. We show that two such properties, the α-center proximity of Awasthi et al. (2012) and the $(1+\varepsilon)$-perturbation resilience of Bilu and Linial (2012), imply positive one-versus-all margin. Let us recall these properties. A function d' (not necessarily a pseudometric) is a $(1+\varepsilon)$-perturbation of d if $d \leq d' \leq (1+\varepsilon)d$.

Definition 13 *Let $\mathcal{C}$ be a center-based clustering.*

- *$\mathcal{C}$ satisfies the α-center proximity property with $\alpha > 1$ if, for all $i \in [k]$, for all $x \in C_i$ and all $j \neq i$ we have $d(x, c_j) > \alpha\, d(x, c_i)$.*
- *$\mathcal{C}$ is $(1+\varepsilon)$-perturbation resilient with $\varepsilon > 0$ if it is induced by the same centers $c_1, \dots, c_k$ under any $(1+\varepsilon)$-perturbation of d.*

It is known that $(1+\varepsilon)$-perturbation resilience implies α-center proximity with $\alpha = 1+\varepsilon$, see (Awasthi et al., 2012). We show:

Theorem 14 *If $\mathcal{C}$ satisfies α-center proximity, then it has one-versus-all margin $\frac{(\alpha-1)^2}{2(\alpha+1)}$. Hence, if $\mathcal{C}$ satisfies $(1+\varepsilon)$-perturbation stability, then it has one-versus-all margin $\frac{\varepsilon^2}{2(\varepsilon+2)}$.*

Proof Consider any cluster C_i with center c_i, let $x' = \arg\max_{x \in C_i} d(x, c_i)$, and choose any $x \in C_i$ and any $y \in C_j$ with $j \neq i$. If $d(x', c_i) = 0$ then clearly $\phi_d(C_i) = 0$, and α-center proximity implies $d(y, x) = d(y, c_i) > \alpha\, d(y, c_j) \geq 0 = \gamma \phi_d(C_i)$ for all $\gamma > 0$. Suppose then $d(x', c_i) > 0$; we consider two cases.
Case 1: $d(x, c_i) = 0$**.** Then $d(y, c_i) = d(y, x)$ and $d(x', c_i) = \phi_d(C_i)$. We obtain:

$$\begin{aligned}
\phi_d(C_i) &= d(x', c_i) \\
&< \frac{1}{\alpha} d(x', c_j) \\
&\leq \frac{1}{\alpha} \Big(d(x', c_i) + d(c_i, y) + d(y, c_j) \Big) \\
&< \frac{1}{\alpha} \left(\phi_d(C_i) + d(y, x) + \frac{1}{\alpha} d(y, c_i) \right) \\
&= \frac{1}{\alpha} \phi_d(C_i) + \frac{1}{\alpha} d(y, x) + \frac{1}{\alpha^2} d(y, x)
\end{aligned}$$

from which we infer $d(y, x) > \frac{\alpha(\alpha-1)}{\alpha+1} \phi_d(C_i) > \frac{(\alpha-1)^2}{2(\alpha+1)} \phi_d(C_i)$.

Case 2: $d(x, c_i) > 0$**.** We start by deriving:

$$\begin{aligned} d(y, x) &\geq d(x, c_j) - d(y, c_j) \\ &> d(x, c_j) - \frac{1}{\alpha} d(y, c_i) \\ &\geq d(x, c_j) - \frac{1}{\alpha}\big(d(y, x) + d(x, c_i)\big) \\ &= -\frac{1}{\alpha} d(y, x) + d(x, c_j) - \frac{1}{\alpha} d(x, c_i) \end{aligned}$$

and thus $d(y, x)\left(\frac{\alpha+1}{\alpha}\right) > d(x, c_j) - \frac{1}{\alpha} d(x, c_i)$, which yields

$$d(y, x) > \frac{\alpha}{\alpha+1} d(x, c_j) - \frac{1}{\alpha+1} d(x, c_i) \tag{3}$$

Let $\beta = \frac{d(x', c_i)}{d(x, c_i)}$. Note that $\phi_d(C_i) \leq 2\beta d(x, c_i)$, thus $d(x, c_i) \geq \frac{\phi_d(C_i)}{2\beta}$. We consider two cases. First, suppose $\beta \leq \frac{\alpha+1}{\alpha-1}$. In this case apply $d(x, c_j) > \alpha d(x, c_i)$ to (3) to obtain:

$$\begin{aligned} d(y, x) &> \frac{\alpha^2}{\alpha+1} d(x, c_i) - \frac{1}{\alpha+1} d(x, c_i) \\ &= d(x, c_i)(\alpha - 1) \\ &\geq \frac{\phi_d(C_i)}{2\beta}(\alpha - 1) \\ &\geq \phi_d(C_i) \frac{(\alpha-1)^2}{2(\alpha+1)} \end{aligned}$$

Suppose instead $\beta > \frac{\alpha+1}{\alpha-1}$. Since we chose $x' \in C_i$ such that $d(x', c_i) = \beta d(x, c_i)$,

$$\begin{aligned} d(x, c_j) &> d(x', c_j) - d(x, x') \\ &> \alpha d(x', c_i) - \big(d(x, c_i) + d(x', c_i)\big) \\ &= \alpha\beta d(x, c_i) - \big(d(x, c_i) + \beta d(x, c_i)\big) \\ &= d(x, c_i)((\alpha - 1)\beta - 1) \end{aligned}$$

Combining this with (3), we obtain:

$$\begin{aligned} d(y, x) &> d(x, c_i)\left(\frac{\alpha}{\alpha+1}((\alpha-1)\beta - 1) - \frac{1}{\alpha+1}\right) \\ &= d(x, c_i)\left(\frac{\alpha(\alpha-1)\beta}{\alpha+1} - 1\right) \\ &\geq \frac{\phi_d(C_i)}{2\beta}\left(\frac{\alpha(\alpha-1)\beta}{\alpha+1} - 1\right) \\ &= \phi_d(C_i)\left(\frac{\alpha(\alpha-1)}{2(\alpha+1)} - \frac{1}{2\beta}\right) \\ &> \phi_d(C_i)\left(\frac{\alpha(\alpha-1) - (\alpha-1)}{2(\alpha+1)}\right) \\ &= \phi_d(C_i)\frac{(\alpha-1)^2}{2(\alpha+1)} \end{aligned}$$

Hence in all cases $d(y,x) > \phi_d(C_i)\frac{(\alpha-1)^2}{2(\alpha+1)}$. This concludes the proof. ■

4.2.2 mRec and proof of Theorem 12

The idea behind mRec is a classic one: use the smallest hypothesis consistent with the labeled sample. To be more formal, we need:

Definition 15 *For any finite $X \subset \mathcal{X}$, any pseudometric d over $\mathcal{X}$, and any $\gamma > 0$, the* effective concept class *with one-versus-all margin γ with respect to d over X is:*

$$H(X,d,\gamma) = \{C \subseteq X \,:\, d(X \setminus C, C) > \gamma\, \phi_d(C)\}$$

Lemma 16 *Let d be any pseudometric over $\mathcal{X}$ and $H = H(X,d,\gamma)$. Then $H = I(H)$.*

Proof Let $C_1, C_2 \in H$ and $C = C_1 \cap C_2$. Let $y \in X \setminus C$; without loss of generality we may assume $y \notin C_1$. Since $C \subseteq C_1$, then $\phi_d(C_1) \geq \phi_d(C)$ and $d(y,C) \geq d(y,C_1)$. Moreover, $d(y,C_1) > \gamma\, \phi_d(C_1)$ since $C_1 \in H$. Therefore:

$$d(y,C) \geq d(y,C_1) > \gamma\, \phi_d(C_1) \geq \gamma\, \phi_d(C)$$

Thus C has one-versus-all margin γ. As this holds for all $C = C_1 \cap C_2$, then $H = I(H)$. ■

Let $d_1, \ldots, d_k$ be the pseudometrics of Definition 11. In each round, mRec draws a uniform random sample S from X, labels it and partitions it accordingly into $S_1, \ldots, S_k$. Then, for each $i \in [k]$, it computes the smallest set $\widehat{C}_i \in H(X, d_i, \gamma)$ consistent with S_i and labels all of $\widehat{C}_i$ as i. By "smallest set" we mean the intersection of all elements of $H(X, d_i, \gamma)$ that include S. It is easy to see that all assigned labels are correct. To prove the upper bounds of Theorem 12 it remains to show that, if S is sufficiently large compared to $\mathcal{M}(\gamma)$, then every round labels a good fraction of X. To begin with, we prove that vc-dim$(H(X, d_i, \gamma))$ is bounded by $1 + \mathcal{M}(\gamma)$. Recall that $I(H)$ denotes the intersection class of H.

Algorithm 3 mRec$(X, O_{\mathcal{C}}, \gamma, d_1, \ldots, d_k)$

```
while X ≠ ∅ do
    draw a sample S of Θ(M(γ) k ln k) points uniformly at random from X
    use O_C to label S and partition it into S_1, ..., S_k
    for i ∈ [k] do
        let Ĉ_i be the smallest set in H(X, d_i, γ) such that S_i ⊆ Ĉ_i
        label every x ∈ Ĉ_i with i
    X = X \ ∪_{i∈[k]} Ĉ_i
```

Lemma 17 *Let d be any pseudometric over $\mathcal{X}$ and $H = H(X,d,\gamma)$. Then* vc-dim$(H,X) \leq 1 + \mathcal{M}(\gamma, d)$.

Proof By Lemma 16 vc-dim(H,X) = vc-dim$(I(H),X)$. Now we use the following results:

Definition 18 (Kivinen 1995, Definition 5.11) *Let $\mathcal{X}$ be any set and $\mathcal{H} \subseteq 2^{\mathcal{X}}$. We say that $\mathcal{H}$ slices $X \subset \mathcal{X}$ if, for each $x \in X$, there is $h \in \mathcal{H}$ such that $X \cap h = X \setminus \{x\}$. The slicing dimension of $(\mathcal{H}, \mathcal{X})$, denoted by $\mathrm{sl}(\mathcal{H}, \mathcal{X})$, is the maximum size of a set sliced by $\mathcal{H}$. If $\mathcal{H}$ slices arbitrarily large sets, then we let $\mathrm{sl}(\mathcal{H}, \mathcal{X}) = \infty$.*

Lemma 19 (Kivinen 1995, Lemma 5.19) $\text{vc-dim}(I(\mathcal{H}), \mathcal{X}) \leq \mathrm{sl}(\mathcal{H}, \mathcal{X})$.

Next, we show that $\mathrm{sl}(H, X) \leq 1 + \mathcal{M}(\gamma, d)$; by Lemma 19 this implies the claim.

Let $S \subseteq X$ be any set sliced by H. We show that $|S| \leq 1 + \mathcal{M}(\gamma, d)$. The case $|S| \leq 2$ is trivial since $\mathcal{M}(\gamma, d) \geq 1$. Suppose then that $|S| \geq 3$. Choose $a, b \in S$ such that $d(a, b) = \phi_d(S)$. Since S is sliced by H, for any $x \in S$ we have $S \setminus \{x\} = S \cap C$ for some $C \in H$. Since $S \setminus \{x\} \subseteq C$, we have $d\big(S \setminus \{x\}, x\big) \geq d(C, x)$ and $\phi_d(C) \geq \phi_d(S \setminus \{x\})$. Moreover, $d(C, x) > \gamma\, \phi_d(C)$, since $x \in X \setminus C$ and $C \in H$. Therefore:

$$d\big(S \setminus \{x\}, x\big) \geq d(C, x) > \gamma\, \phi_d(C) \geq \gamma\, \phi_d(S \setminus \{x\})$$

Hence, $d(S \setminus \{x\}, x) > \gamma\, \phi_d(S \setminus \{x\})$ for all $x \in S$.

Now, suppose first that $\gamma \geq 1$. Since $|S| \geq 3$, any $x \in S \setminus \{a, b\}$ yields the absurd:

$$\begin{aligned}
\phi_d(S) &\geq d(S \setminus \{x\}, x) \\
&> \phi_d(S \setminus \{x\}) && \text{since } \gamma \geq 1 \\
&= d(a, b) && \text{since } a, b \in S \setminus x \\
&= \phi_d(S) && \text{by the choice of } a, b
\end{aligned}$$

Hence $|S| \leq 2 \leq 1 + \mathcal{M}(\gamma, d)$.

Suppose instead that $\gamma < 1$. Then, for any two distinct points $x, y \in S$, we have $d(x, y) > \gamma \phi_d(S)$. This is trivially true if $x = a$ and $y = b$; otherwise, assuming $x \notin \{a, b\}$, it follows by the fact that $d(x, y) \geq d(S \setminus \{x\}, x) > \gamma \phi_d(S \setminus \{x\}) = \gamma \phi_d(S)$, as seen above. Moreover, S is contained in the closed ball $B(x, \phi_d(S))$ for any $x \in S$. Therefore, $\mathcal{M}(B(x, r), \gamma r, d) \geq |S|$ for $r = \phi_d(S)$ and some $x \in \mathcal{X}$. By definition of $\mathcal{M}(\gamma, d)$ this implies that $|S| < 1 + \mathcal{M}(\gamma, d)$. This concludes the proof. ■

We can finally prove Theorem 12. First, we prove the lower bound. By definition of $M(2\gamma)$, there is a set X of $M(2\gamma)$ points that, according to some $d \in \{d_1, \ldots, d_k\}$, lies within a ball of radius $r > 0$, and thus has diameter at most $2r$, and such that $d(x, y) > 2\gamma r$ for all distinct $x, y \in X$. Now choose $x \in X$ uniformly at random, and define $\mathcal{C} = (\{x\}, X \setminus \{x\})$. The same argument used in the proof of Lemma 17 shows that $d(x, X \setminus \{x\}) > \gamma\, \phi_d(X \setminus \{x\})$, which implies that $\mathcal{C}$ has one-versus-all margin γ. Clearly, in expectation over the distribution of $\mathcal{C}$, any algorithm that learns $\mathcal{C}$ exactly makes $b\,|X| = b\,M(2\gamma)$ queries for some universal $b > 0$. By Yao's principle, then, any such algorithm makes $b\,M(2\gamma)$ queries on some instance.

We turn to the upper bounds. Consider a generic iteration of mRec and let $i \in [k]$. By Lemma 17 and standard generalization bounds, if $|S| = \Theta\big(\frac{1}{\varepsilon}\big(\mathcal{M}(\gamma) \log \frac{1}{\varepsilon} + \log \frac{1}{\delta}\big)\big)$ then:

$$\mathbb{P}\left(\left|C_i \setminus \widehat{C}_i\right| \leq \varepsilon |X|\right) \geq 1 - \delta$$

Set $\varepsilon = \frac{1}{2k}$ and $\delta = \frac{1}{2}$. This makes $|S| = \Theta\big(\mathcal{M}(\gamma)\, k \log k\big)$, and since $|C_i \setminus \widehat{C}_i| = |C_i| - |\widehat{C}_i|$, the probability bound above yields:

$$\mathbb{P}\left(\big|\widehat{C}_i\big| \geq |C_i| - \frac{|X|}{2k}\right) \geq \frac{1}{2}$$

which, since $\big|\widehat{C}_i\big| \geq 0$, implies:

$$\mathbb{E}\big|\widehat{C}_i\big| \geq \frac{1}{2}\left(|C_i| - \frac{|X|}{2k}\right) = \frac{|C_i|}{2} - \frac{|X|}{4k}$$

Thus, the expected number of points mRec removes from X at the end of a generic round is:

$$\mathbb{E}\left|\widehat{C}_1 \cup \ldots \cup \widehat{C}_k\right| = \sum_{i=1}^{k} \mathbb{E}\big|\widehat{C}_i\big| \geq \sum_{i=1}^{k}\left(\frac{|C_i|}{2} - \frac{|X|}{4k}\right) = \frac{|X|}{2} - \frac{|X|}{4} = \frac{|X|}{4}$$

Standard concentration bounds show that with high probability mRec makes $\mathcal{O}(\log n)$ rounds and thus $\mathcal{O}(M(\gamma)\, k \log k \log n)$ queries, concluding the proof of Theorem 12.

4.3 One-versus-all classifiers

Let $\mathcal{X}$ be any domain, $X \subset \mathcal{X}$ a finite subset, and $\mathcal{C} = (C_1, \ldots, C_k)$ a partition of X. Let $\mathcal{H}$ be any concept class over $\mathcal{X}$. For example, for the ellipsoidal clusters of Bressan et al. (2020) $\mathcal{X} = \mathbb{R}^m$ and $\mathcal{H}$ is the set of all ellipsoids in $\mathbb{R}^m$; for the convex classes of Section 4.1, $\mathcal{X} = \mathbb{R}^m$ and $\mathcal{H}$ is the family of all polytopes in $\mathbb{R}^m$. Recall that $\mathcal{C}$ is realized by $\mathcal{H}$ if for all $i \in [k]$ there is some $h_i \in \mathcal{H}$ such that $C_i = X \cap h_i$. We start by introducing the *coslicing dimension*, a combinatorial quantity similar to the star number of Hanneke and Yang (2015) and the slicing dimension of Definition 18.

Definition 20 *$\mathcal{H}$ coslices $X \subseteq \mathcal{X}$ if for all $x \in X$ there exist $h_x^+, h_x^- \in \mathcal{H}$ such that $X \cap h_x^+ = \{x\}$ and $X \cap h_x^- = X \setminus \{x\}$. The coslicing dimension of $\mathcal{H}$ is $\operatorname{cosl}(\mathcal{H}) = \sup\{|X| : X$ is cosliced by $\mathcal{H}\}$. If $\mathcal{H}$ coslices arbitrarily large sets then we let $\operatorname{cosl}(\mathcal{H}) = \infty$.*

Consider for example $\mathcal{X} = \mathbb{R}^m$. If $\mathcal{H}$ is the set of linear separators, then $\operatorname{cosl}(\mathcal{H}) = \infty$; if instead $\mathcal{H}$ is the set of axis-aligned boxes, it is not hard to see that $\operatorname{cosl}(\mathcal{H}) = 2m$.

We give two results. First, we show that $\mathcal{C}$ is actively learnable if and only if $\operatorname{cosl}(\mathcal{H}) < \infty$ (Theorem 21). Second, for a wide family of concept classes in $\mathbb{R}^m$, we show that $\operatorname{cosl}(\mathcal{H}) < \infty$ is equivalent to a positive one-versus-all margin (Theorem 22).

Theorem 21 *Suppose $\operatorname{cosl}(\mathcal{H}) < \infty$. There is an algorithm that, given $(X, O_{\mathcal{C}})$ such that $\mathcal{C}$ is realized by $\mathcal{H}$, learns $\mathcal{C}$ with $\mathcal{O}(\operatorname{cosl}(\mathcal{H})\, k \log k \log |X|)$ queries to $O_{\mathcal{C}}$ with high probability; moreover, any algorithm $\mathcal{A}$ makes $\Omega(\operatorname{cosl}(\mathcal{H}))$ label queries in expectation to learn $\mathcal{C}$. If instead $\operatorname{cosl}(\mathcal{H}) = \infty$, then for any algorithm $\mathcal{A}$ that learns $\mathcal{C}$ there are instances $(X, O_{\mathcal{C}})$ with X arbitrarily large where $\mathcal{A}$ makes $\Omega(|X|)$ queries in expectation in the worst case.*

Proof The proof is along the lines of the proof of Theorem 12. For the lower bound, let X be a set cosliced by $\mathcal{H}$. Let $x \in X$ be uniform random and let $\mathcal{C} = (\{x\}, X \setminus \{x\})$. By definition of cosliced set $\mathcal{C}$ is realised by $\mathcal{H}$. The same arguments of the proof of Theorem 12

show that any algorithm makes $\Omega(|X|)$ queries in the worst case. This implies the lower bounds for both $\mathrm{cosl}(\mathcal{H}) < \infty$ and $\mathrm{cosl}(\mathcal{H}) = \infty$. For the upper bounds, let $P_k(X)$ be the set of all k-partitions of X that are realised by $\mathcal{H}$. For each $i \in [k]$ define:

$$H_i = \left\{C' \,:\, C' = C'_i \wedge (C'_1, \ldots, C'_k) \in P_k(X)\right\}$$

Adapt mRec (Algorithm 3) by (i) drawing S of size $\Theta(\text{vc-dim}(I(H_i), X)\, k \ln k)$, (ii) letting $\widehat{C}_i$ be the smallest hypothesis in $I(H_i)$ consistent with S_i. If we prove that $\mathrm{sl}(H_i, X) \leq \mathrm{cosl}(\mathcal{H})$, then the proof of Lemma 17 carries on with $\text{vc-dim}(I(H_i), X) \leq \mathrm{cosl}(\mathcal{H})$ and yields our bound. To prove that $\mathrm{sl}(H_i, X) \leq \mathrm{cosl}(\mathcal{H})$, suppose that $U = \{x_1, \ldots, x_\ell\} \subseteq X$ is sliced by H_i. By construction of H_i this means that there are ℓ classes $\mathcal{C}_1, \ldots, \mathcal{C}_\ell$, each one realised by $\mathcal{H}$, such that $\mathcal{C}_i = (\{x_i\}, U \setminus \{x_i\})$ for all $i \in [k]$. Thus U is cosliced by $\mathcal{H}$, so $|U| \leq \mathrm{cosl}(\mathcal{H})$, and $\mathrm{sl}(H_i, X) \leq \mathrm{cosl}(\mathcal{H})$. We conclude that the adaptation of mRec learns $\mathcal{C}$ by making with high probability $\mathcal{O}(\mathrm{cosl}(\mathcal{H}) k \log k \log n)$ queries. ■

For our second result, let us say that $\mathcal{H}$ is *non-fractal* if there exists $h \in \mathcal{H}$ such that both h and $\mathcal{X} \setminus h$ contain some ball of positive radius; this avoids pathological cases such as $\mathcal{H}$ containing only affine transformations of Cantor's set.

Theorem 22 *Let $\mathcal{H}$ be a concept class in $\mathbb{R}^m$ that is non-fractal and closed under affine transformations. There is an algorithm that, given any instance $(X, O_\mathcal{C})$ where $\mathcal{C}$ is realized by $\mathcal{H}$ and has one-versus-all margin $\gamma > 0$, learns $\mathcal{C}$ by making $\mathcal{O}\big((1 + 4/\gamma)^m\, k \log k \log |X|\big)$ queries to $O_\mathcal{C}$ with high probability. Moreover, for any randomized algorithm $\mathcal{A}$ there are instances $(X, O_\mathcal{C})$, with X arbitrarily large and $\mathcal{C}$ realized by $\mathcal{H}$ with arbitrarily small one-versus-all margin, where $\mathcal{A}$ makes $\Omega(|X|)$ label queries in expectation to learn $\mathcal{C}$.*

Proof The upper bound follows from Theorem 12, by recalling that in the Euclidean metric the unit ball has covering number $\mathcal{N}(B(x,1), \varepsilon) \leq (1 + 2/\varepsilon)^m$ for all $\varepsilon > 0$ and $\mathcal{M}(B(x,1), \varepsilon) \leq \mathcal{N}(B(x,1), \varepsilon/2)$. For $\varepsilon = \gamma$ this yields $M(\gamma) \leq (1 + 4/\gamma)^m$.

We now prove the lower bound. Choose $\gamma > 0$ arbitrarily small. Take any $h \in \mathcal{H}$ such that both h and $\overline{h} = \mathbb{R}^m \setminus h$ contain a ball of positive radius. Then for any $\rho > 0$ there exists a closed ball B with radius $r > 0$ such that:

$$B \subseteq h$$
$$\exists x \in \overline{h} \,:\, d(B, x) \leq \rho$$

Let c be the center of B. Consider a sphere S of radius r' that contains x and whose center c' lies on the affine subspace $x + \alpha(c - x)$, see Figure 1. Let $\eta = \sup_{y \in S \setminus B} d(x, y)$ and let X be an η-packing of S. By making r' and $\frac{\eta}{r'}$ arbitrarily small we can make X arbitrarily large. Now consider $x' \in X \setminus \{x\}$. Since by construction $d(x', x) > \eta$, and as $r' < r$, then $x' \in B \subseteq h$. Hence h satisfies $X \cap h = X \setminus \{x\}$. Now, for any $x' \in X$ there is an invertible affine transformation $R_{x'}$ (a rotation) such that $R_{x'}(x') = x$. Let then $h_{x'} = R_{x'}^{-1} h_x$. Clearly $X \cap h_{x'} = X \setminus \{x'\}$, and since $\mathcal{H}$ is closed under affine transformations, then $h_{x'} \in \mathcal{H}$. Thus for every $x \in X$ there exists $h_x \in \mathcal{H}$ such that $X \cap h_x = X \setminus \{x\}$.

Now consider the complement $\overline{h}$ of h. The argument above applies to $\overline{h}$, too, hence for every $x \in X$ there exists an invertible affine transformation $R_{x'}$ such that $\overline{h}_x := R_{x'} \overline{h}$

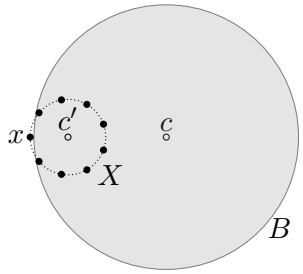

Figure 1: The η-packing X of S is in B, and thus in h, except for x that lies in $\overline{h}$. By taking B arbitrarily close to x we can make η arbitrarily small and X arbitrarily large.

satisfies $X \cap \overline{h}_x = X \setminus \{x\}$. The complement h_x of $\overline{h}_x$ thus satisfies $X \cap h_x = \{x\}$, and since $h_x = R_{x'}h$, then $h_x \in \mathcal{H}$.

We conclude that for every $x \in X$ there exist $h_x^-, h_x^+ \in \mathcal{H}$ such that $X \cap h_x^- = X \setminus \{x\}$ and $X \cap h_x^+ = \{x\}$; thus every 2-clustering $\mathcal{C}$ of X in the form $C_1 = \{x\}, C_2 = X \setminus \{x\}$ is realized by $\mathcal{H}$. This immediately implies that any algorithm makes $\Omega(|X|)$ queries to learn $\mathcal{C}$ on some instance $(X, O_{\mathcal{C}})$. As we can make X arbitrarily large, this completes the proof. ■

Note that Theorem 22 applies to many fundamental concept classes—for instance linear separators, ellipsoids, polytopes, and more in general convex bodies (bounded or not, and possibly degenerate), as well as disjoint unions of those concepts.

5. Learning classifiers in finite semimetric spaces

In this section we consider learning classifiers in (finite) semimetric spaces. The use of semimetrics is common in domains where the notion of distance is strongly non-Euclidean (see Gottlieb et al., 2017). Classic examples include the Wasserstein distance in vision, the Levenshtein distance in string matching, the cosine dissimilarity in document analysis, the Pearson dissimilarity in bioinformatics. In all these cases, the notions of convexity and separability with margin are lost, or exist only in certain generalized forms, hence the techniques of Section 4.1 do not apply anymore. To fill this gap, we introduce a novel notion of class convexity that can be applied to any finite semimetric space and exploited algorithmically. Let X be a finite set and d a semimetric over X. We assume that (X, d) is given as a weighted graph $\mathcal{G} = (X, \mathcal{E}, d)$ where $\{x, y\} \in \mathcal{E}$ if and only if $d(x, y) > 0$. We would like to use $\mathcal{G}$ to define our abstract notion of convexity. We start by considering *geodesic convexity* (Pelayo, 2013), a well-known generalization of Euclidean convexity. A set $C \subseteq X$ is geodesically convex if every shortest path in $\mathcal{G}$ between pairs of vertices of C lies in C. Unfortunately, this condition is too lax. To see this, let X consist of n distinct points on a circle with the Euclidean metric. In $\mathcal{G}$, the shortest path between any two points x, y is the edge $\{x, y\}$. Thus, any $C \subseteq X$ will be geodesically convex, hence $\mathcal{H} = [k]^X$ and $\Omega(n)$ queries are needed to learn $\mathcal{C}$ in the worst case. However, a variant of this approach gives a suitable notion of convexity. For any $\varepsilon > 0$ let G_ε be the unit disk graph of $\left(X, \frac{d}{\varepsilon}\right)$; this is the simple graph where $x, y \in X$ are connected if and only if $d(x, y) \leq \varepsilon$. For any simple

graph $G = (V, E)$ and any $x, y \in V$ denote by $d_G(x, y)$ the shortest-path distance between x and y in G.

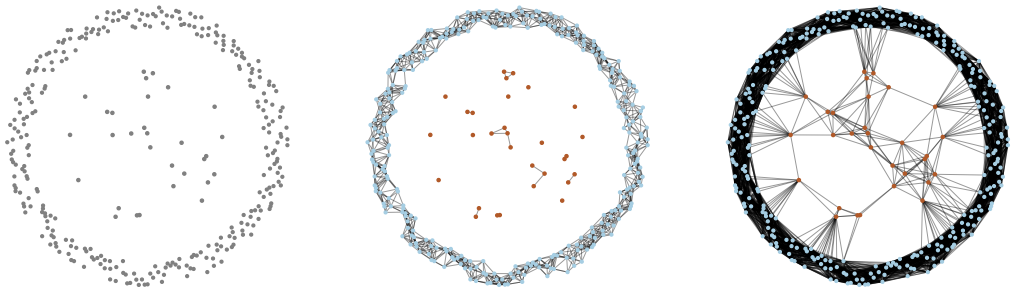

Figure 2: Left: a set $X \subset \mathbb{R}^2$. Middle and right: the graphs G_{ε_1} and G_{ε_2} where $\varepsilon_1 < \varepsilon_2$ are the radii of the "outer" class C_1 and the "inner" class C_2. The partition is (β, γ)-convex for $\beta \leq .5$ and $\gamma \leq .1$.

Definition 23 *Let $\beta, \gamma \in (0, 1]$. A k-partition $\mathcal{C} = (C_1, \ldots, C_k)$ of X is (β, γ)-convex if there is $\varepsilon > 0$ such that for each $i \in [k]$ the following properties hold:*

1. connectedness*: $G_\varepsilon[C_i]$ is connected*
2. local metric margin*: for all $x, y \in X$, if $x \in C_i$ and $y \notin C_i$ then $d(x, y) > \beta\varepsilon$*
3. geodesic convexity with margin*: if $x, y \in C_i$, then in G_ε any simple path between x and y of length at most $(1 + \gamma) d_{G_\varepsilon}(x, y)$ lies in C_i*

The smallest value of ε satisfying the three properties is called the radius *of the classes.*[6]

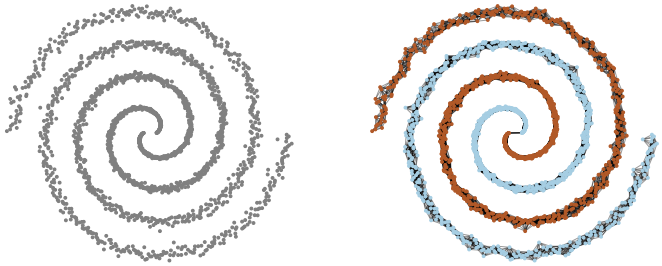

Figure 3: Left: a toy point set. Right: the graph $G_\varepsilon[X]$ and a valid (β, γ)-convex partition for X for any $\beta < \frac{1}{2}$ and $\gamma > 0$ (classes encoded by the color of the points).

Although definition 23 might seem artificial at first sight, it captures pretty interesting scenarios. Indeed, (β, γ)-convex classes can be strongly non-convex in $\mathbb{R}^m$, see Figure 3. Section 5.3 gives a more elaborate version of Definition 23 allowing for classes with different radii, as in Figure 2. These examples suggest that one can view (β, γ)-convexity as a way of translating density into convexity.

Moreover, we can show that (β, γ)-convexity becomes uninteresting if we drop any one of the conditions of Definition 23. Let indeed $X = \mathbb{R}^2$, let $k = 2$, and let d be the Euclidean distance. We use Figure 4 for reference.

6. Actually, our algorithm in Section 5.2 works with *any* such ε; we use the minimum to disambiguate.

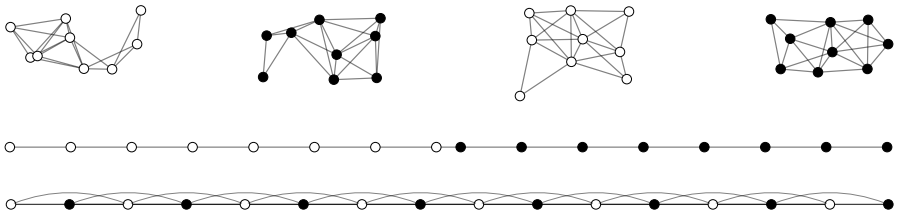

Figure 4: Example of classes that sastisfy all properties of Definition 23 except: connectivity (top), local metric margin (middle), geodesic convexity with margin (bottom). Empty nodes are in C_1, filled nodes are in C_2. The edges shown are those of $G_\varepsilon[X]$.

Removing connectivity. Choose any $\beta, \gamma \leq 1$. Let X consist of disjoint subsets, each one with diameter $\leq \varepsilon$, and sufficiently far from each other. Label any subsets as C_1 and the rest as C_2. Note that the second and third property in Definition 23 are satisfied.
Removing local metric margin. Choose any $\gamma \leq 1$. Let C_1 consist of collinear points spaced by ε, and the same for C_2, so that two extremal points of C_1 and C_2 are at arbitrarily small distance $\delta < \varepsilon$. Note that the first and third property in Definition 23 are satisfied.
Removing geodesic convexity with margin. Choose any $\beta < \frac{1}{2}$. Let X consist of collinear points spaced by $\frac{\varepsilon}{2}$, with the points of C_1 and C_2 interleaved. Note that the first and second property in Definition 23 are satisfied, but the third one is violated for any $\gamma = \omega(1/n)$: take the two extreme nodes of C_1 and change their shortest path to use a point of C_2.

Our main result is LearnBGC (Algorithm 4 in Section 5.2), an efficient active learning algorithm for (β, γ)-convex classifiers.

Theorem 24 *Let $\mathcal{C} = (C_1, \ldots, C_k)$ be a (β, γ)-convex partition of X with radius ε and let $\boldsymbol{s} = (s_1, \ldots, s_k)$ where $s_i \in C_i$ for each i. Then* LearnBGC$(\mathcal{G}, \beta, \gamma, \varepsilon, \boldsymbol{s}, O_{\mathcal{C}})$ *learns $\mathcal{C}$ exactly in time $\mathcal{O}(k^2(n+m))$ using $\mathcal{O}\big(k^2 \log n + k^2 \mathcal{M}\big(\frac{\beta\gamma}{2+\gamma}\big)\big) = \mathcal{O}\big(k^2 \log n + k^2 (6/\beta\gamma)^{\mathrm{dens}(X)}\big)$ queries.*

In many aspects, Theorem 24 cannot be significantly improved: as we show in Section 5.6, the dependence on $\mathcal{M}$ and dens is essentially tight, and any algorithm needs $\Omega(n)$ queries if the *seed nodes* $s_1, \ldots, s_k$ are not available or if one changes the definition of (β, γ)-convexity to allow for a different semimetric for each class.

To relax the assumptions of Theorem 24, we introduce SEED queries. For any $S \subseteq X$ and $i \in [k]$, SEED(S, i) returns an arbitrary point $s_i \in C_i \cap S$, or NIL if $C_i \cap S = \emptyset$. We show that, at the price of some SEED queries, one can learn $\mathcal{C}$ without knowing ε or $s_1, \ldots, s_k$. In fact, if we adopt a "hereditary" version of the geodesic convexity property, then we can even allow each class C_i to have its own radius ε_i, which captures classes that are dense at a different scale, as in Figure 2. Formally:

Definition 25 *Let $\beta, \gamma \in (0, 1]$. A partition $\mathcal{C} = (C_1, \ldots, C_k)$ of X is* hereditary (β, γ)-convex *if for some $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_k) \in \mathbb{R}^k_{\geq 0}$ the following properties hold for all $i \in [k]$:*

1. connectedness*: $G_{\varepsilon_i}[C_i]$ is connected*
2. local metric margin*: for all $x, y \in X$, if $x \in C_i$ and $y \notin C_i$, then $d(x, y) > \beta\varepsilon_i$*
3. hereditary geodesic convexity with margin*: for any $\varepsilon \leq \varepsilon_i$, if $x, y \in C_i$ and $d_{G_\varepsilon}(x, y) < \infty$, then in G_ε any simple path between x and y of length at most $(1+\gamma) d_{G_\varepsilon}(x, y)$ lies in C_i*

Our main result is LearnHBGC (Algorithm 10 in Section 5.3), an efficient algorithm based on label and SEED queries to learn hereditary (β, γ)-convex classifiers.

Theorem 26 *Let $\mathcal{C}$ be hereditary (β, γ)-convex. Then* LearnHBGC$(\mathcal{G}, \beta, \gamma, \boldsymbol{\varepsilon}, O_{\mathcal{C}})$ *learns $\mathcal{C}$ exactly, has the same runtime as* LearnBGC*, and uses the same number of label queries as* LearnBGC *plus $\mathcal{O}(k^2)$* SEED *queries.*

Next, we investigate the cost of learning $\mathcal{C}$ when the model parameters are unknown. First we show how, when β and/or γ are unknown, one can learn a (hereditary) (β, γ)-convex classifier $\mathcal{C}$ for a small overhead in terms of queries and running time. Formally, we prove:

Theorem 27 *Suppose $\mathcal{C}$ is (hereditary) (β, γ)-convex. If only one of β, γ is unknown then one can learn $\mathcal{C}$ in time $O\big(\big(k^2 + \log \frac{2}{p}\big)(n+m)\big)$ using $\mathcal{O}\big(k^2 \log n + \big(k^2 + \log \frac{2}{p}\big)\, \mathcal{M}\big(\frac{\beta\gamma}{2}\big)\big)$ label queries and $\big(3k^2 + 2\log \frac{2}{p}\big)$* SEED *queries, where $p \in \{\beta, \gamma\}$ is the unknown parameter. If both β, γ are unknown then one can learn $\mathcal{C}$ in time $O\big(\big(k^2 + \log^2 \frac{2}{\beta\gamma}\big)(n+m)\big)$ using $\mathcal{O}\big(k^2 \log n + \big(k^2 + \log^2 \frac{2}{\beta\gamma}\big)\, \mathcal{M}\big(\frac{\beta\gamma}{2}\big)\big)$ label queries and $\big(3k^2 + 2\log \frac{2}{\beta\gamma}\big)$* SEED *queries.*

Finally, we show one can efficiently learn the radii of a hereditary (β, γ)-convex classifier.

Theorem 28 *The radii $\varepsilon_1, \ldots, \varepsilon_k$ can be learned using $\mathcal{O}(k \log n)$* SEED *queries in time $\mathcal{O}(m\, \alpha(m,n) + kn \log n)$, where $m = |\mathcal{E}|$ and $\alpha(m,n)$ is the functional inverse of the Ackermann function.*[7] *Moreover, any (randomized) algorithm needs $\Omega(k \log \frac{n}{k})$* SEED *and/or label queries to learn the radii of all classes with constant probability.*

5.1 Additional notation

We denote by $G = (V, E)$ a generic unweighted graph. We denote by $\rho(G)$ the number of connected components of G; for any $U \subseteq V$ we denote by $G[U]$ the subgraph of G induced by U, and by $\Gamma_G(U)$ the set of edges of G with exactly one endpoint in U. An edge in $\Gamma_G(U)$ is called a *cut edge* of U. We often treat such edges as oriented by writing $(x, y) \in \Gamma_G(U)$ to specify that $x \in U$ and $y \notin U$. We assume the algorithm is given a vector $\boldsymbol{s} = (s_1, \ldots, s_k)$ of *seed nodes* $s_i \in C_i$ for each $i \in [k]$ and that the radius ε is known. Without $\boldsymbol{s}$ any algorithm needs $\Omega(n)$ queries, see Section 5.6. These assumptions may be avoided by introducing *seed queries*: for any $S \subseteq X$ and $i \in [k]$, SEED(S, i) returns an arbitrary point $s_i \in C_i \cap S$, or NIL if $C_i \cap S = \emptyset$. Given a bipartition $(S, X \setminus S)$ of X and $i \in [k]$, with two SEED queries one can check whether C_i is cut by the partition, yielding a kind of separation oracle.

Recall $\mathcal{M}(\eta)$ from Section 3. Following Gottlieb et al. (2017), we define the *density constant* of X as $\mu(X) = \mathcal{M}(1/2)$ and the *density dimension* of X as $\operatorname{dens}(X) = \log_2 \mu(X)$.[8] It is easy to see that $\mathcal{M}(\eta) \le \mu(X)^{\lceil \log_2 1/\eta \rceil} \le (2/\eta)^{\operatorname{dens}(X)}$ for every $\eta > 0$.

5.2 LearnBGC and proof of Theorem 24

This section describes LearnBGC and proves the correctness and query bounds of Theorem 24. The proofs of the running time bounds require some care, and are deferred to Section 5.4. To simplify the notation we omit $\mathcal{G}, \beta, \gamma, O_{\mathcal{C}}$ from the parameters passed to the algorithms

7. For all practical purposes, α can be considered constant. For instance, $\alpha(m,m) \le 4$ for all $m \le \frac{1}{8} 2^{2^{2^{65536}}}$.
8. While Gottlieb et al. (2017) uses open balls, we use closed balls.

and their subroutines. LearnBGC first computes the graph G_ε and then invokes a second algorithm, LearnClass, which provides:

Lemma 29 LearnClass$(G_\varepsilon, \varepsilon, \boldsymbol{s}, i)$ *returns* C_i *using* $\mathcal{O}\Big(k \log n + k\, \mathcal{M}\big(\frac{\beta\gamma}{2+\gamma}\big)\Big)$ *queries.*

The correctness and query bounds of Theorem 24 follow immediately from Lemma 29 and the pseudocode of LearnBGC below. Hence the rest the section deals with LearnClass and the proof of Lemma 29.

Algorithm 4 LearnBGC$(\varepsilon, \boldsymbol{s})$

compute G_ε
for $i = 1, \dots, k$ **do**
    $C_i :=$ LearnClass$(G_\varepsilon, \varepsilon, \boldsymbol{s}, i)$
return $(C_1, \dots, C_k)$

5.2.1 Margin-based separation, and binary search on shortest paths.

We begin with a simple routine MBS (for Margin-Based Separator) that is useful for learning the labels of any set of small radius.

Algorithm 5 MBS(Z, ε)

1: $Z_1 := \cdots := Z_k := \emptyset$
2: **for** each connected component H of $G_{\beta\varepsilon}[Z]$ **do**
3:     query $O_{\mathcal{C}}$ for the label i of any $x \in V(H)$
4:     add $V(H)$ to Z_i
return $(Z_1, \dots, Z_k)$

Lemma 30 *For any* $Z \subseteq X$ *such that* $Z \subseteq B(z, r)$ *for some* $z \in X$ *and* $r < \infty$, MBS(Z, ε) *returns* $(Z \cap C_1, \dots, Z \cap C_k)$ *using at most* $\mathcal{M}\big(\frac{\beta\varepsilon}{r}\big)$ *queries.*

Proof If H is a connected component of $G_{\beta\varepsilon}[Z]$ then $d(x, y) \le \beta\varepsilon$ for all $\{x, y\} \in E(H)$, hence by the local metric margin $V(H) \subseteq C_i$ for some i. This proves the correctness. For the query bound, note that the vertices queried at line 4 form a set P independent in $G_{\beta\varepsilon}[Z]$, thus $d(x, y) > \beta\varepsilon$ for all distinct $x, y \in P$. By definition of $\mathcal{M}(\cdot)$ then $|P| \le \mathcal{M}\big(\frac{\beta\varepsilon}{r}\big)$. ■

Next, we give a condition for finding efficiently a cut edge of C_i. A path $\pi = (x_1, \dots, x_{|\pi|})$ is *C_i-prefixed* if there exists $j^* \in \big[|\pi|\big]$ such that $x_j \in C_i$ if and only if $j \in \{1, \dots, j^*\}$.

Lemma 31 *Let* $C_i \subseteq R \subseteq X$ *such that* $G_\varepsilon[R]$ *is connected. Then in* $G_\varepsilon[R]$ *any shortest path between any* $s_i \in C_i$ *and any* $s \in R \setminus C_i$ *is* C_i*-prefixed.*

Proof Suppose $G_\varepsilon[R]$ contains a shortest path π between $s_i \in C_i$ and $s \in R \setminus C_i$ that is not C_i-prefixed. Then some prefix π' of π is a shortest path between $s_i \in C_i$ and $s_i' \in C_i$ that intersects $R \setminus C_i$. Now observe that π' is a shortest path in G, too. This holds because any shortest path between s_i and s_i' in G_ε lies in $G_\varepsilon[C_i]$ by geodesic convexity, and thus in $G_\varepsilon[R]$ as $C_i \subseteq R$. By geodesic convexity this implies $\pi' \subseteq G_\varepsilon[C_i]$, a contradiction. ■

Finally, in a C_i-prefixed simple path, a cut edge of C_i can be found via binary search, yielding a routine FindCutEdge with the following guarantees whose proof is straightforward:

Lemma 32 *Given a simple C_i-prefixed path π,* FindCutEdge(π) *returns the unique cut edge of C_i in π using $\mathcal{O}(\log n)$ queries.*

Algorithm 6 FindCutEdge($\pi(s_i, s_h)$)

1: **if** $|\pi(s_i, s_h)| = 1$ **then return** $\pi(s_i, s_h)$
2: choose a median node x in $\pi(s_i, s_h)$
3: **if** $O_{\mathcal{C}}(x) = i$ **then return** FindCutEdge($\pi(x, s_h)$) **else return** FindCutEdge($\pi(s_i, x)$)

5.2.2 Class separators

We introduce the notion of class separator, which is at the heart of LearnClass.

Definition 33 *Let $R \subseteq X$ and $C_i, C_j \in \mathcal{C}$. A partition (S_i, S_j) of R is an (i, j)-separator of R if $S_i \cap C_j = S_j \cap C_i = \emptyset$.*

A class separator is similar to half-spaces in abstract closure systems (Seiffarth et al., 2019), which require $C_i \subseteq S_i$ and $C_j \subseteq S_j$. We use the weaker condition $S_i \cap C_j = S_j \cap C_i = \emptyset$ because in some of our algorithms $R \subsetneq X$ and thus $C_j \subseteq R$ might not hold. However we will always ensure that $C_i \subseteq R$, and thus $C_i \subseteq S_i$, where C_i is the class we are learning. Therefore, if we could compute an (i, j)-separator for all $j \neq i$, then we could compute C_i as a simple intersection. To compute (S_i, S_j) efficiently, suppose we know a cut edge (u_i, u_j) between C_i and C_j. First, we compute $Z = \{x \in R : d_{G_\varepsilon}(x, u_i) < \frac{1}{\gamma}\}$ and learn $Z \cap C_i$ using MBS (Algorithm 5). For every $x \in R \setminus Z$, instead, we show that the condition $d_{G_\varepsilon}(x, u_i) \leq d_{G_\varepsilon}(x, u_j)$ tells whether to place x in S_i or S_j without making queries:

Lemma 34 *Let $(u_i, u_j) \in G_\varepsilon$ be a cut edge between C_i and C_j and suppose $\frac{1}{\gamma} \leq d_{G_\varepsilon}(u_i, x) < \infty$. If $d_{G_\varepsilon}(u_i, x) \leq d_{G_\varepsilon}(u_j, x)$ then $x \notin C_j$, and if $d_{G_\varepsilon}(u_i, x) \geq d_{G_\varepsilon}(u_j, x)$ then $x \notin C_i$.*

Proof Assume $d_{G_\varepsilon}(u_i, x) \leq d_{G_\varepsilon}(u_j, x)$ and $x \in C_j$; we show this leads to a contradiction. Consider the path $\pi = \pi(x, u_i) + (u_i, u_j)$. First, π is a simple path, as otherwise $u_j \in \pi(x, u_i)$, which implies $d_{G_\varepsilon}(u_j, x) \leq d_{G_\varepsilon}(u_i, x) - 1$, contradicting the assumption. Second:

$$\begin{aligned} |\pi| &= d_{G_\varepsilon}(x, u_i) + 1 \\ &\leq d_{G_\varepsilon}(x, u_j) + 1 && \text{since } d_{G_\varepsilon}(u_i, x) \leq d_{G_\varepsilon}(u_j, x) \\ &\leq d_{G_\varepsilon}(x, u_j)\,(1 + \gamma) && \text{since } d_{G_\varepsilon}(x, u_j) \geq d_{G_\varepsilon}(x, u_i) \geq 1/\gamma \end{aligned}$$

Thus, π is a simple path between two points of C_j, with length at most $(1 + \gamma)$ times their distance in G_ε, and containing a point of C_i. This violates the geodesic convexity of C_j. We conclude that $x \notin C_j$. The other case is symmetric. ■

We can now introduce FindSeparator (Algorithm 7), a routine for computing an (i, j)-separator given a cut edge between C_i and C_j.

Algorithm 7 FindSeparator$(G_\varepsilon, u_i, u_j)$

$Z := \{x \in V(G_\varepsilon) : d_G(u_i, x) < 1/\gamma\}$
$(Z_1, \ldots, Z_k) := \mathrm{MBS}(Z, \varepsilon)$
$U_i := \emptyset, \quad U_j := \emptyset$
for each $x \in V(G_\varepsilon) \setminus Z$ **do**
    if $d_{G_\varepsilon}(x, u_i) \le d_{G_\varepsilon}(x, u_j)$ **then** add x to U_i **else** add x to U_j
return $(Z_i \cup U_i,\ (Z \setminus Z_i) \cup U_j)$

Lemma 35 *Let $(u_i, u_j) \in G_\varepsilon$ be a cut edge between C_i and C_j.* FindSeparator$(G_\varepsilon, u_i, u_j)$ *returns an (i,j)-separator (S_i, S_j) of $V(G_\varepsilon)$ using at most $\mathcal{M}(\beta\gamma)$ queries.*

Proof The query bound follows from Lemma 30 by observing that $Z \subseteq B(u_i, \varepsilon/\gamma)$. For the correctness, we show that $(Z_i, Z \setminus Z_i)$ is an (i,j)-separator of Z and (U_i, U_j) is an (i,j)-separator of $V(G_\varepsilon) \setminus Z$. For Z, Lemma 30 guarantees that $Z_i = C_i \cap Z$. So $Z_i \cap C_j = (Z \setminus Z_i) \cap C_i = \emptyset$, and $(Z_i, Z \setminus Z_i)$ is an (i,j)-separator of Z. Consider now any $x \in V(G_\varepsilon) \setminus Z$. By definition of Z, we have $d_{G_\varepsilon}(x, u_i) \ge \frac{1}{\gamma}$. Therefore, by Lemma 34, if the algorithm assigns x to U_i then $x \notin C_j$, and if the algorithm assigns x to U_j then $x \notin C_i$. Therefore $U_i \cap C_j = U_j \cap C_i = \emptyset$, and (U_i, U_j) is an (i,j)-separator of $V(G_\varepsilon) \setminus Z$. ■

5.2.3 Learning a single class with LearnClass

We can finally introduce LearnClass (Algorithm 8) and prove Lemma 29. LearnClass starts by computing the set of vertices R_i reachable from s_i in G_ε, and the induced subgraph $G_{\varepsilon,i} = G_\varepsilon[R_i]$. Note that, by the connectedness of the classes, R_i is simply the union of C_i and possibly other classes. Now, consider a seed node $s_h \in \boldsymbol{s}$ such that $s_h \in R_i \setminus \{s_i\}$. If no such s_h exists, then $R_i = C_i$ and we are done. Otherwise, compute the shortest path π between s_h and s_i in $G_{\varepsilon,i}$. By Lemma 31, π is C_i-prefixed, and so by Lemma 32 we can find a cut edge (u_i, u_j) of C_i with $\mathcal{O}(\log n)$ queries. Using (u_i, u_j) we can then compute an (i,j)-separator (S_i, S_j) of $X = V(G_\varepsilon)$ using FindSeparator. Finally, we update $G_{\varepsilon,i}$ to be the connected component of s_i in $G_\varepsilon[R_i \cap S_i]$, and R_i to be $V(G_{\varepsilon,i})$. By definition of (S_i, S_j), this removes from $G_{\varepsilon,i}$ all vertices of C_j, so we have reduced by at least one the number of classes other than C_i intersected by R_i. After at most $k-1$ of these rounds we have $R_i = C_i$.

This process has an issue: it can run out of seeds. Indeed, by taking $R_i \cap S_i$ we could remove every seed node s_h, even if $R_i \cap S_i$ still contains points of C_h. If this is the case then LearnClass would be stuck, that is, unable to compute a new cut edge. One is tempted to ignore that $s_h \notin R_i$, and compute the shortest path between s_h and s_i for all $h \neq i$ nonetheless. This however does not work, as the only cut edge on all those shortest paths could be (u_i, u_j). We bypass this obstacle by exploiting the separators found by LearnClass. By carefully analysing the cuts of those separators we devise an algorithm, FindNewSeed, that either finds some new seed $s_h \in R_i \setminus C_i$ or certifies that $R_i = C_i$. FindNewSeed is invoked by LearnClass at the beginning of each round and is described below.

Algorithm 8 LearnClass($G_\varepsilon, \varepsilon, \boldsymbol{s}, i$)

1: $G_{\varepsilon,i} := \{$the connected component of s_i in $G_\varepsilon\}$, $R_i := V(G_{\varepsilon,i})$
2: $\boldsymbol{u} :=$ an empty vector
3: **while** true **do**
4: $\quad s_h :=$ FindNewSeed($G_\varepsilon, R_i, \varepsilon, \boldsymbol{s}, \boldsymbol{u}, i$)
5: $\quad$ **if** $s_h =$ NIL **then return** R_i
6: $\quad \pi(s_i, s_h) :=$ ShortestPath($G_\varepsilon[R_i], s_i, s_h$)
7: $\quad (u_i, u_j) :=$ FindCutEdge($\pi(s_i, s_h)$)
8: $\quad$ add u_i to $\boldsymbol{u}$
9: $\quad (S_i, S_j) :=$ FindSeparator(G_ε, u_i, u_j)
10: $\quad G_{\varepsilon,i} := \{$the connected component of s_i in $G_{\varepsilon,i}[R_i \cap S_i]\}$, $R_i := V(G_{\varepsilon,i})$

Proof [of Lemma 29] For each $\ell = 1, 2, \ldots$, denote by R_i^ℓ the set R_i at the beginning of the ℓ-th iteration, and by $\kappa(R_i^\ell)$ the number of distinct classes that R_i^ℓ intersects. We show that, at the beginning of the ℓ-th iteration, the following invariants hold:

I1. $G_\varepsilon[R_i^\ell]$ is connected
I2. $C_i \subseteq R_i^\ell$
I3. $\kappa(R_i^\ell) \leq \kappa(R_i^1) - (\ell - 1)$

Note that I2 and I3 imply that the algorithm stops and returns C_i (line 5) after at most k iterations. Invariant I1 holds by the construction of R_i^ℓ, so we focus on proving I2 and I3.

Suppose first $\ell = 1$. Then, I2 holds by the assumptions of the lemma, and I3 is trivial. Suppose then I1, I2, I3 hold for some $\ell \geq 1$, and that iteration $\ell + 1$ exists; we show that I2, I3 hold at iteration $\ell + 1$ as well. First, if iteration $\ell + 1$ exists, then $C_i \subsetneq R_i^\ell$. Then, by Lemma 36, FindNewSeed returns $s_h \in R_i^\ell \setminus C_i$. As $G_\varepsilon[R_i^\ell]$ is connected by I1, the shortest path $\pi(s_i, s_h)$ exists in $G_\varepsilon[R_i^\ell]$, and is C_i-prefixed by Lemma 31 applied to R_i^ℓ. Then by Lemma 32 FindCutEdge($\pi(s_i, s_h)$) returns a cut edge (u_i, u_j) of C_i in $\pi(s_i, s_h)$. At this point the hypotheses of Lemma 35 are satisfied, so FindSeparator(G_ε, u_i, u_j) returns an (i, j)-separator (S_i, S_j) of $V(G_\varepsilon)$, hence $C_i \subseteq R_i^\ell \cap S_i$. This implies that the connected component of s_i in $G_\varepsilon[R_i^\ell \cap S_i]$ still contains C_i. The vertex set of this connected component is precisely $R_i^{\ell+1}$ (line 10), hence I2 holds. Moreover $u_j \in R_i^\ell$, since $u_j \in \pi(s_i, s_h) \subseteq R_i^\ell$. Therefore $R_i^\ell \cap C_j \neq \emptyset$. However, by construction, $R_i^{\ell+1} \subseteq R_i^\ell \cap S_i$ and $S_i \cap C_j = \emptyset$ since (S_i, S_j) is an (i, j)-separator of $V(G_\varepsilon)$. Therefore $\kappa(R_i^{\ell+1}) \leq \kappa(R_i^\ell) - 1$. As I3 holds at iteration ℓ, then $\kappa(R_i^{\ell+1}) \leq \kappa(R_i^1) - (\ell - 1) - 1 = \kappa(R_i^1) - ((\ell + 1) - 1)$. So I3 holds too.

Finally, we bound the number of queries. By Lemma 36 (see below), FindNewSeed makes at most $k\,\mathcal{M}\big(\frac{\beta\gamma}{2+\gamma}\big)$ SCQ in total across all iterations. By Lemma 32, FindCutEdge makes $\mathcal{O}(\log n)$ queries at each iteration. By Lemma 35, FindSeparator makes at most $\mathcal{M}(\beta\gamma)$ queries at each iteration. Summing the three terms yields the bound. ■

Finding new seed nodes with FindNewSeed**.** We describe the routine FindNewSeed (Algorithm 9) that, given $R_i \supseteq C_i$, either decides that $R_i = C_i$ or finds a node $s_h \in R_i \setminus C_i$. The key idea behind FindNewSeed is this: if R_i does not contain any seed $s_h \in \boldsymbol{s}$ other than s_i, then for each $h \neq i$ either $C_h \cap R_i = \emptyset$ or, by the connectedness of $G_\varepsilon[C_h]$, the cut

$\Gamma_{G_\varepsilon}(R_i)$ must contain some edge of $G_\varepsilon[C_h]$. Therefore the task boils down to finding such an edge, or deciding that there is none. Clearly, checking all edges in $\Gamma_{G_\varepsilon}(R_i)$ would require too many queries. Fortunately, the following approach works: for all those edges that are close to cut edges previously used, we perform an explicit search using MBS(Algorithm 5). For the remaining edges, geodesic convexity tells whether a cut edge exists without making queries.

Algorithm 9 FindNewSeed$(G_\varepsilon, R_i, \varepsilon, \boldsymbol{s}, \boldsymbol{u}, i)$

1: **if** $\boldsymbol{s} \cap R_i \neq \{s_i\}$ **then return** any $s \in \boldsymbol{s} \cap R_i \setminus \{s_i\}$
2: **for** each $u \in \boldsymbol{u}$ **do**
3: $\quad Z = \{x \in R_i \,:\, d_{G_\varepsilon}(u, x) < 2/\gamma + 1\}$
4: $\quad (Z_1, \ldots, Z_k) := \mathrm{MBS}(Z, \varepsilon)$
5: $\quad$ **if** $Z \setminus Z_i \neq \emptyset$ **then return** any $x \in Z \setminus Z_i$
6: **if** $\exists\, (x, y) \in \Gamma_{G_\varepsilon}(R_i)$ such that $\forall u \in \boldsymbol{u} \,:\, d_{G_\varepsilon}(u, x) \geq 2/\gamma + 1$ **then return** any such x
7: **else return** NIL

Lemma 36 *Consider the beginning of any iteration of* LearnClass$(G_\varepsilon, \varepsilon, \boldsymbol{s}, i)$. *Then, the call to* FindNewSeed$(G_\varepsilon, R_i, \varepsilon, \boldsymbol{s}, \boldsymbol{u})$ *returns a point $x \in R_i \setminus C_i$ if $R_i \neq C_i$, or* NIL *if $R_i = C_i$, using at most $k\,\mathcal{M}\big(\frac{\beta\gamma}{2+\gamma}\big)$ queries. Moreover,* FindNewSeed *can be adapted so as to make at most $k\,\mathcal{M}\big(\frac{\beta\gamma}{2+\gamma}\big)$ queries over the entire execution of* LearnClass.

In order to prove Lemma 36 we need to two ancillary results, Lemma 37 and Lemma 38.

Lemma 37 *Let (S_i, S_j) be an (i, j)-separator for $V(G_\varepsilon)$ obtained from* FindSeparator$(V(G_\varepsilon)$, $u_i, u_j)$. *If $(x, y) \in \Gamma_{G_\varepsilon}(S_i)$ and $d_G(u_i, x) \geq \frac{2}{\gamma} + 1$ then $x \notin C_i$.*

Proof We show that $x \in C_i$ violates the geodesic convexity of C_i. First $(x, y) \in \Gamma_{G_\varepsilon}(S_i)$ implies $y \in S_j$. Moreover, $d_{G_\varepsilon}(u_i, y) \geq d_{G_\varepsilon}(u_i, x) - 1 > \frac{1}{\gamma}$. Now recall the code of FindSeparator. Since $d_{G_\varepsilon}(u_i, y) > \frac{1}{\gamma}$, then $y \notin Z_j$. But $y \in S_j = Z_j \cup U_j$, and therefore $y \in U_j$. This means that FindSeparator executed line 5, which happens only if:

$$d_{G_\varepsilon}(u_j, y) < d_{G_\varepsilon}(u_i, y) \tag{4}$$

Now consider the path $\pi = (u_i, u_j) + \pi(u_j, y) + (y, x)$ from u_i to x where $\pi(u_j, y)$ has length $d_{G_\varepsilon}(u_j, y)$. First, we show that π is a simple path. Suppose indeed that π was not simple. Since $\pi(u_j, y)$ is simple (it is a shortest path), and since $u_i \neq x$ (because $d_{G_\varepsilon}(u_i, x) \geq 1$ by assumption), then $u_i \in \pi(u_j, y)$ or $x \in \pi(u_j, y)$. If $u_i \in \pi(u_j, y)$, then $d_{G_\varepsilon}(u_j, y) > d_{G_\varepsilon}(u_i, y)$, which contradicts (4). If instead $x \in \pi(u_j, y)$, then $d_{G_\varepsilon}(u_j, y) > d_{G_\varepsilon}(u_j, x)$, which gives:

$$\begin{aligned} d_{G_\varepsilon}(u_j, y) &> d_{G_\varepsilon}(u_j, x) \\ &\geq d_{G_\varepsilon}(u_i, x) && \text{since } x \in S_i \\ &\geq d_{G_\varepsilon}(u_i, y) - 1 && \text{since } (x, y) \in E(G_\varepsilon) \end{aligned}$$

which, since d_{G_ε} is integral, implies $d_{G_\varepsilon}(u_j, y) \geq d_{G_\varepsilon}(u_i, y)$. This contradicts again (4). Thus, π is a simple path. Next, we show that $|\pi| \leq d_{G_\varepsilon}(u_i, x)\,(1 + \gamma)$. From (4):

$$\begin{aligned} d_{G_\varepsilon}(u_j, y) &\leq d_{G_\varepsilon}(u_i, y) - 1 && \text{since } d_{G_\varepsilon} \in \mathbb{N} \\ &\leq d_{G_\varepsilon}(u_i, x) + d_{G_\varepsilon}(x, y) - 1 && \text{since } (x, y) \in E(G_\varepsilon) \\ &= d_{G_\varepsilon}(u_i, x) \end{aligned}$$

And therefore:

$$\begin{aligned} |\pi| &= d_{G_\varepsilon}(u_j, y) + 2 \\ &\le d_{G_\varepsilon}(u_i, x) + 2 && \text{since } d_{G_\varepsilon}(u_j, y) \le d_{G_\varepsilon}(u_i, x) \\ &\le d_{G_\varepsilon}(u_i, x)\,(1+\gamma) && \text{since } d_{G_\varepsilon}(u_i, x) \ge \frac{2}{\gamma} \end{aligned}$$

Hence π is a simple path violating the geodesic convexity of C_i. So $x \notin C_i$, as claimed. ■

Recall LearnClass$(G_\varepsilon, \varepsilon, \boldsymbol{s}, i)$ in Algorithm 8. Let R_i^ℓ be the value of R_i at the beginning of the ℓ-th iteration, and let (S_i^ℓ, S_{j_ℓ}) be the separator computed by FindSeparator at the ℓ-th iteration.

Lemma 38 *Let $\ell \ge 2$. If $(x,y) \in \Gamma_{G_\varepsilon}(R_i^\ell)$, then $(x,y) \in \Gamma(S_i^\tau)$ for some $\tau \in \{1, \ldots, \ell-1\}$.*

Proof Let $\tau = \min\left\{1 \le t \le \ell - 1 \,:\, (x,y) \in \Gamma_{G_\varepsilon}(R_i^{t+1})\right\}$. First, $x \in S_i^\tau$. Indeed $x \in R_i^{\tau+1}$, and by construction $R_i^{\tau+1} \subseteq R_i^\tau \cap S_i^\tau$. Second, $y \notin S_i^\tau$. Suppose indeed by contradiction that $y \in S_i^\tau$. Since $x \in R_i^\tau$ and $(x,y) \notin \Gamma_{G_\varepsilon}(R_i^\tau)$, then $y \in R_i^\tau$. Thus $y \in S_i^\tau \cap R_i^\tau$. So y is adjacent to x in $G_\varepsilon[S_i^\tau \cap R_i^\tau]$, and therefore $y \in R_i^{\tau+1}$ as well, by construction of $R_i^{\tau+1}$ as a connected component. But then $(x,y) \notin \Gamma_{G_\varepsilon}(R_i^{\tau+1})$, which contradicts our hypothesis. Therefore $x \in S_i^\tau$ and $y \notin S_i^\tau$. We conclude that $(x,y) \in \Gamma_{G_\varepsilon}(S_i^\tau)$. ■

Proof [of Lemma 36] The query bound holds by Lemma 30 noting that $Z \subseteq B\big(u, \varepsilon\big(\frac{2}{\gamma}+1\big)\big) = B\big(u, \varepsilon\frac{2+\gamma}{\gamma}\big)$. Let us now prove the correctness. In the first iteration of LearnClass R_i is the union of C_i and zero or more other classes, $\Gamma_{G_\varepsilon}(R_i) = \emptyset$, and $\boldsymbol{u} = \emptyset$. In that case, if $R_i = C_i$ then FindNewSeed returns NIL, otherwise it returns some $s \in \boldsymbol{s} \cap R_i \setminus \{s_i\}$. Consider now any subsequent iteration of LearnClass. Suppose first that $R_i = C_i$. Then $\boldsymbol{s} \cap R_i = \{s_i\}$. Moreover, at each **for** iteration, by Lemma 30 $Z_i = Z$, so $Z \setminus Z_i = \emptyset$. Finally, note that no edge $(x,y) \in \Gamma_{G_\varepsilon}(R_i)$ exists such that $d_{G_\varepsilon}(u,x) \ge \frac{2}{\gamma} + 1$ for all $u \in \boldsymbol{u}$. Indeed, if such an edge existed, Lemma 38 would imply $(x,y) \in \Gamma_{G_\varepsilon}(S_i)$ in some previous round of LearnClass, which by Lemma 37 implies $x \notin C_i$—a contradiction, since $x \in R_i = C_i$. Hence, FindNewSeed reaches line 7 and returns NIL. Now suppose that $R_i \ne C_i$. If $s_h \in R_i$ for some $s_h \ne s_i$, then FindNewSeed returns s_h at line 1. Otherwise, we must have $C_h \cap R_i \ne \emptyset$ but $C_h \not\subseteq R_i$ for some $h \ne i$. By the connectedness of C_h in G_ε this implies the existence of an edge $(x,y) \in \Gamma_{G_\varepsilon}(R_i)$ with $x \in C_h$. If any such edge exists with $d_{G_\varepsilon}(u,x) < \frac{2}{\gamma} + 1$ for some $u \in \boldsymbol{u}$, then line 5 finds x and returns it. Otherwise, any such edge has $d_{G_\varepsilon}(u,x) \ge \frac{2}{\gamma} + 1$ for all $u \in \boldsymbol{u}$. In this case, line 6 returns x such that $(x,y) \in \Gamma_{G_\varepsilon}(R_i)$ and $d_{G_\varepsilon}(u,x) \ge \frac{2}{\gamma} + 1$ for all $u \in \boldsymbol{u}$, in which case $x \notin C_i$ by Lemma 38 and Lemma 37.

To make FindNewSeed use at most $k\,\mathcal{M}\big(\frac{\beta\gamma}{2+\gamma}\big)$ queries over the execution of LearnClass, we keep track of $Z \setminus Z_i$. At each invocation compute the set $Z^u = \{x \in R_i \,:\, d_{G_\varepsilon}(u,x) < \frac{2}{\gamma} + 1\}$, where u is the last node added to $\boldsymbol{u}$. Invoke MBS on Z^u, retrieve Z_i^u, and add $Z^u \setminus Z_i^u$ to $Z \setminus Z_i$. Finally, remove from $Z \setminus Z_i$ all points not in R_i. This sequence of operations costs $\mathcal{M}\big(\frac{\beta\gamma}{2+\gamma}\big)$ queries, and the resulting set is exactly the $Z \setminus Z_i$ computed at line 5 of FindNewSeed. Hence the behavior of LearnClass is unchanged, but the total number of queries is at most $k\,\mathcal{M}\big(\frac{\beta\gamma}{2+\gamma}\big)$. ■

5.3 LearnHBGC and proof of Theorem 26

This section describes LearnHBGC (Algorithm 10) and proves Theorem 26. We start by noting that, if the conditions of Definition 25 hold, then they hold for the smallest ε_i such that C_i is connected in G_{ε_i},[9] so without loss of generality we can use such an ε_i.

Lemma 39 *Let $\mathcal{C}$ be a hereditary (β,γ)-convex partition of X, and for $i \in [k]$ let:*

$$\zeta_i = \min\{\zeta : \forall\, x, y \in C_i \,:\, d_{G_\zeta}(x,y) < \infty\} \tag{5}$$

$$\zeta_i^* = \min\{\zeta : \rho(G_\zeta[C_i]) = 1\} \tag{6}$$

Then $\zeta_i = \zeta_i^*$, and all properties of Definition 25 hold when $\varepsilon_i = \zeta_i = \zeta_i^*$.

Proof First, note that $\zeta_i \le \zeta_i^*$, since $\rho(G_\zeta[C_i]) = 1$ implies $\forall\, x, y \in C_i \,:\, d_{G_\zeta}(x,y) < \infty$, and so the minimum in (5) is taken over a superset of that of (6). Now consider the graph G_{ζ_i}. For $x, y \in C_i$, since $\zeta_i \le \zeta_i^*$ and $d_{G_{\zeta_i}}(x,y) < \infty$, the geodesic convexity implies that any shortest path in G_{ζ_i} between x and y lies in C_i. Thus the subgraph induced by C_i in G_{ζ_i} is connected. By definition of ζ_i^* this implies $\zeta_i^* \le \zeta_i$, so $\zeta_i = \zeta_i^*$.

For the second claim, let $\varepsilon_i = \zeta_i^*$. The connectedness of $G_{\zeta_i^*}[C_i]$ holds by definition of ζ_i^*. Now let ζ be any value such that the three properties hold when $\varepsilon_i = \zeta$. Then $\zeta \ge \zeta_i^*$, because for $\varepsilon_i < \zeta_i^*$ the connectedness fails, by definition of ζ_i^*. The hereditarity implies that the local metric margin and the geodesic convexity with margin hold for $\varepsilon_i = \zeta_i^*$, since they hold for $\varepsilon_i = \zeta \ge \zeta_i^*$. ■

We now turn to LearnHBGC (Algorithm 10). The basic idea is to learn each C_i separately by invoking LearnClass (Algorithm 8) on the graph $G_{\varepsilon_i}[X]$. Unfortunately, this does not work: any C_j with $\varepsilon_j < \varepsilon_i$ may not satisfy geodesic convexity in G_{ε_i}, so LearnClass may fail. However, we can make things work under the following precautions. First, recover the classes in nondecreasing order of radius. Second, when recovering C_i, restrict G_{ε_i} to the connected component of s_i. Third, after recovering C_i, delete it from X. Note that this procedure learns each C_i on a graph thresholded at a different radius and with only a subset of the original vertices. Thus, its correctness is not be obvious. In particular, it is not obvious that the partition induced by the sub-instance used at the i-th iteration is (β,γ)-convex, which is necessary for LearnClass to work. We show that it is: at every iteration, the residual instance is (β,γ)-convex, and the guarantees of Lemma 29 hold. In all this, the role of SEED queries is to find a seed node for each class in the connected component of G_{ε_i} containing s_i, as required by LearnClass.

After one last technical lemma, we can describe LearnHBGC and prove Theorem 26.

Lemma 40 *Let $\mathcal{C}$ be a hereditary (β,γ)-convex partition of X. Let ε^* be the smallest radius of any class and let X^* be the vertex set of any connected component of G_{ε^*} that contains a class with radius ε^*. Finally, let $\mathcal{C}^* = \big(C_1 \cap X^*, \ldots, C_\kappa \cap X^*\big)$. Then $\mathcal{C}^*$ is a (β,γ)-convex partition of X^* with radius ε^* in in $G_{\varepsilon^*}[X^*]$.*

9. Note that this is different from requiring that $G_{\varepsilon_i}[C_i]$ is connected; here we are only requiring that any two points of C_i have a connecting path in G_{ε_i}. Lemma 39 however shows that the smallest ε_i that satisfies either condition is actually the same.

Proof We verify that the three properties of Definition 23 hold with radius ε^*. This implies that ε^* is also the smallest such value, since the corresponding classes becomes disconnected in $G_\varepsilon[X^*]$ for any $\varepsilon < \varepsilon^*$. Let $C_i^* = C_i \cap X^*$ for all $i \in [\kappa]$. To simplify the notation let $G^* = G_{\varepsilon^*}[X^*]$.

Connectivity: the subgraph induced by C_i^* in G^* is connected. Consider two points $x, y \in C_i^*$; obviously $x, y \in C_i$. Since G^* is connected, $d_{G^*}(x,y) < \infty$. Moreover, $d_{G^*}(x,y) = d_{G_{\varepsilon^*}}(x,y)$, since by construction G^* is the connected component of G_{ε^*} containing x, y. Thus, $d_{G_{\varepsilon^*}}(x,y) < \infty$. Moreover, $\varepsilon^* \leq \varepsilon_i$ by assumption. Thus, $x, y \in C_i$, and $d_{G_{\varepsilon^*}}(x,y) < \infty$ with $\varepsilon^* \leq \varepsilon_i$. Then, by the hereditary geodesic convexity of $\mathcal{C}$ on X, any shortest path π between x and y in G_{ε^*} lies in C_i. Since again G^* is the connected component of G_{ε^*} containing x, y, then π lies in X^*. We conclude that $\pi \in C_i \cap X^* = C_i^*$. This holds for any choice of x, y. Therefore, $G^*[C_i^*]$ is connected.

Local metric margin: for all $x, y \in X^*$, if $x \in C_i^*$ and $y \notin C_i^*$, then $d(x,y) > \beta\varepsilon^*$. Since $x \in C_i^*$ and $y \notin C_i^*$, then $x \in C_i$ and $y \notin C_i$. By the local metric margin of $\mathcal{C}$, and since $\varepsilon_i \geq \varepsilon^*$, we have $d(x,y) > \beta\varepsilon_i \geq \beta\varepsilon^*$.

Geodesic convexity with margin: if $x, y \in C_i^*$, then in G^* any simple path between x and y of length at most $(1+\gamma)d_{G^*}(x,y)$ lies in C_i^*. By the same argument of connectivity, we invoke the hereditary geodesic convexity for $x, y \in C_i$, for $\varepsilon = \varepsilon^* \leq \varepsilon_i$. We obtain that G_{ε^*} contains no simple path of length at most $(1+\gamma)d_{G_{\varepsilon^*}}(x,y)$ between x and y that leaves C_i. This implies that no such path exists in G^* as well, since $G^* \subseteq G_{\varepsilon^*}$. Moreover, since $C_i^* = C_i \cap X^*$, no such path exists in G^* that leaves C_i^* (otherwise it would leave C_i). Recalling that $(1+\gamma)d_{G_{\varepsilon^*}}(x,y) = (1+\gamma)d_{G^*}(x,y)$, we deduce that in G^* there is no path of length at most $(1+\gamma)d_{G^*}(x,y)$ between x and y that leaves C_i^*. ■

Algorithm 10 LearnHBGC$(X, \boldsymbol{\varepsilon}, \boldsymbol{s})$

assume $\varepsilon_1 \leq \ldots \leq \varepsilon_k$
for $i = 1, \ldots, k$ **do**
  $G^* :=$ the connected component of s_i in $G_{\varepsilon_i}[X]$, $X^* := V(G^*)$
  for $j = i+1, \ldots, k$ **do** $s_j^* :=$ SEED(X^*, j)
  $\boldsymbol{s}^* := (s_i, s_{i+1}^*, \ldots, s_k^*)$
  $\widehat{C_i} :=$ LearnClass$(G^*, \varepsilon_i, \boldsymbol{s}^*, i)$
  output $\widehat{C_i}$
  $X := X \setminus \widehat{C_i}$

Proof [of Theorem 26] For each $i = 1, \ldots, k$ let X_i be the value of X at the beginning of the i-th iteration of LearnHBGC$(X, \boldsymbol{\varepsilon}, \boldsymbol{s})$. We prove the following three invariants:

I1. $X_i = C_i \cup \ldots \cup C_k$

I2. $(C_i, \ldots, C_k)$ is a hereditary (β, γ)-convex partition of X_i

I3. if $i > 1$ then the algorithm has output $C_1, \ldots, C_{i-1}$ so far.

For $i = 1$ we have $X_i = X$ and I1-I3 clearly hold. Now assume I1-I3 hold at the beginning of the i-th iteration for some $i \geq 1$. Observe that if LearnHBGC sets $\widehat{C_i} = C_i$ then I1-I3

hold at iteration $i+1$. For I1 and I3 this is trivial; for I2, note that deleting a class never invalidates the properties of Definition 25, thus, $(C_{i+1}, \dots, C_k)$ will be a hereditary (β, γ)-convex partition of $X_{i+1} = C_{i+1} \cup \dots \cup C_k$. Hence, we prove that LearnHBGC sets $\widehat{C_i} = C_i$. Consider the subgraph G^* and its node set V^* computed at line 3. Let $\mathcal{C}_i = (C_i, \dots, C_k)$, and let $\mathcal{C}_i^* = (C_i \cap X^*, \dots, C_k \cap X^*)$. Since ε_i is the smallest radius of all classes in $\mathcal{C}_i$, then by Lemma 40 $\mathcal{C}_i^*$ is a hereditary (β, γ)-convex partition of X_i with radius ε_i. Furthermore, by construction $\boldsymbol{s}^*$ contains one seed for each nonempty class in $\mathcal{C}^*$. Thus by Lemma 29, LearnClass$(G^*, \varepsilon, \boldsymbol{s}^*, i)$ returns C_i, so $\widehat{C_i} = C_i$. The query bound is straightforward. ■

5.4 Bounds on the running time of LearnBGC and LearnHBGC

We show how to implement our algorithms in time linear or essentially linear in the size of $\mathcal{G}$. Without loss of generality assume that, for any graph G and any $x \in V(G)$, the edges incident with x can be listed in constant time per edge. The following claim is straightforward:

Claim 1 *For every $\varepsilon \geq 0$ the graph G_ε can be computed in time $\mathcal{O}(n+m)$. This holds more in general for thresholding any subgraph of $\mathcal{G}$.*

By Claim 1 we can implement LearnClass in time $\mathcal{O}(k^2(n+m))$: essentially, for $\mathcal{O}(k^2)$ times the algorithm computes the distances of all nodes of $G_\varepsilon[X]$ from some given node u, which takes time $\mathcal{O}(n+m)$ via breadth-first search. Since LearnClass is invoked once per each class, this would give a total running time of $\mathcal{O}(k^3(n+m))$ for both LearnBGC and LearnHBGC. With some extra effort, however, we show how to adapt LearnClass so that it runs in time $\mathcal{O}(k(n+m))$, by amortizing the cost of FindNewSeed. In the end, we obtain a running time of $\mathcal{O}(k^2(n+m))$ for both LearnBGC and LearnHBGC.

Lemma 41 MBS(Z, ε) *and* FindSeparator(G, u_i, u_j) *both run in time $\mathcal{O}(n+m)$.*

Proof For MBS(Z, ε), computing $G_{\beta\varepsilon}$ takes time $\mathcal{O}(n+m)$ by Claim 1, and thereafter computing $G_{\beta\varepsilon}[Z]$ and listing its connected components takes again time $\mathcal{O}(n+m)$. Consider now FindSeparator$(G_\varepsilon, u_i, u_j)$. Computing $d_{G_\varepsilon}(u_i, x)$ and $d_{G_\varepsilon}(u_j, x)$ for all $x \in G_\varepsilon$ takes time $\mathcal{O}(n+m)$ using a BFS from u_i and u_j. Computing Z takes time $\mathcal{O}(n)$. MBS(Z, ε) takes time $\mathcal{O}(n+m)$ as said, and the loop at line 4 takes time $\mathcal{O}(n)$. ■

Lemma 42 *In* LearnClass$(G_\varepsilon, \varepsilon, \boldsymbol{s}, i)$*, each call to* FindNewSeed$(G_\varepsilon, R_i, \varepsilon, \boldsymbol{s}, \boldsymbol{u}, i)$ *takes time $\mathcal{O}(k(n+m))$. By adapting both algorithms, this can be reduced to $\mathcal{O}(n+m)$ while adding at most an additive $\mathcal{O}(n+m)$ to the running time of each iteration of* LearnClass.

Proof Let us start with the $\mathcal{O}(k(n+m))$ bound given by a "naive" implementation of FindNewSeed. Line 1 computes $\boldsymbol{s} \cap R_i$ in time $\mathcal{O}(k|R_i|) = \mathcal{O}(kn)$ and performs the check in constant time. Line 2 makes $|\boldsymbol{u}| \leq k$ iterations. Each iteration computes Z in time $\mathcal{O}(n+m)$ with a BFS from u, runs MBS(Z, ε) in time $\mathcal{O}(n+m)$ by Lemma 41, and possibly searches for $x \in Z \setminus Z_i$ in time $\mathcal{O}(n)$. Hence the entire loop of line 2 takes time $\mathcal{O}(k(n+m))$. Finally, line 6 computes $\{(x, y) \in \Gamma(R_i) : \forall u \in \boldsymbol{u} : d_{G_\varepsilon}(u, x) \geq \frac{2}{\gamma} + 1\}$. To this end, first compute

the set $\{x \in R_i : \forall u \in \boldsymbol{u} : d_{G_\varepsilon}(u,x) \geq \frac{2}{\gamma}+1\}$ in time $\mathcal{O}(k(n+m))$ using a BFS from u. For each x in that set list all its edges $(x,y) \in E(G_\varepsilon)$; if some edge satisfies $y \notin R_i$ then return y, otherwise return NIL. Thus, this part takes time $\mathcal{O}(k(n+m))$. Therefore a single call to FindNewSeed takes time $\mathcal{O}(k(n+m))$. Since FindNewSeed is called at most k times, the total running time is $\mathcal{O}(k^2(n+m))$.

Let us now see how to reduce to $\mathcal{O}(n+m)$ the running time of FindNewSeed by adding at most $\mathcal{O}(n+m)$ to each iteration of LearnClass. First, consider line 1 of FindNewSeed. We keep $\boldsymbol{s}$ updated so that $\boldsymbol{s} \cap R_i = \boldsymbol{s} \setminus \{s_i\}$; in this way we can run line 1 in constant time. To this end modify LearnClass as follows. First, we store s_i separately from $\boldsymbol{s}$. Second, after updating $G_{\varepsilon,i}$ and R_i at line 10 of LearnClass, we replace $\boldsymbol{s}$ with $\boldsymbol{s} \cap R_i$. To this end encode R_i as a bitmap of size $\mathcal{O}(n)$, for every $x \in bs$ check if $x \in R_i$, and if so then append x to a new vector $\boldsymbol{s}'$; then replace $\boldsymbol{s}$ with $\boldsymbol{s}'$. Summarizing, we spend an additional $\mathcal{O}(n)$ time at each iteration of LearnClass, and line 1 of FindNewSeed will run in constant time.

Now consider the loop at line 2 of FindNewSeed. Let $Z(u)$ be the value of Z computed for $u \in \boldsymbol{u}$ at line 3, and $Z_i(u)$ the corresponding value of Z_i computed by MBS. Define:

$$\overline{Z_i}(\boldsymbol{u}) := \left\{ x \in R_i \setminus C_i \,:\, \exists u \in \boldsymbol{u} \,:\, d_{G_\varepsilon}(u,x) < \frac{2}{\gamma}+1 \right\}$$

Note that $\overline{Z_i}(\boldsymbol{u}) = \cup_{u \in \boldsymbol{u}} Z(u) \setminus Z_i(u)$; thus the loop detects precisely if $\overline{Z_i}(\boldsymbol{u}) \neq \emptyset$, in which case it returns some $x \in \overline{Z_i}(\boldsymbol{u})$. We adapt LearnClass so as to maintain $\overline{Z_i}(\boldsymbol{u})$ as a linked list across subsequent invocations of FindNewSeed; we can then replace the loop of line 2 with $\mathcal{O}(1)$ elementary operations (checking $\overline{Z_i}(\boldsymbol{u}) \neq \emptyset$ and returning, say, its first element). Start by initializing $\overline{Z_i}(\boldsymbol{u})$ to an empty list. After updating $G_{\varepsilon,i}$ and R_i at line 10, perform the following operations. First, make LearnClass compute $Z(u_i)$. Second, run MBS$(Z(u_i),\varepsilon)$ to obtain $Z_i(u_i) = Z(u_i) \cap C_i$. Third, compute $\overline{Z_i}(u_i) = Z(u_i) \setminus Z_i(u_i)$. Fourth, add $\overline{Z_i}(u_i)$ to $\overline{Z}(\boldsymbol{u})$. Finally, replace $\overline{Z}(\boldsymbol{u})$ with $\overline{Z}(\boldsymbol{u}) \cap R_i$. The last two steps can be performed in time $\mathcal{O}(n)$ by encoding the sets as bitmaps, and every other step requires time $\mathcal{O}(n+m)$; thus, overall the running time of each iteration of LearnClass increases by $\mathcal{O}(n+m)$.

Finally, consider line 6 of FindNewSeed. At the beginning of the first iteration of LearnClass mark all nodes of R_i as *active*. At each iteration of LearnClass, after computing (u_i,u_j), mark as *inactive* every $x \in R_i$ such that $d_{G_\varepsilon}(u_i,x) < \frac{2}{\gamma}+1$; this takes time $\mathcal{O}(n+m)$ by running a BFS from u_i. Finally, for every active $x \in R_i$ list all its edges (x,y), until finding $y \notin R_i$ or concluding that no such y exists (the check $y \notin R_i$ takes time $\mathcal{O}(1)$ by again encoding R_i as a bitmap). This makes line 6 run in time $\mathcal{O}(n+m)$ while making the running time of each iteration of LearnClass increase by $\mathcal{O}(n+m)$. ■

Lemma 43 LearnClass$(G_\varepsilon,\varepsilon,\boldsymbol{s},i)$ *runs in time* $\mathcal{O}(k(n+m))$.

Proof At every instant, computing $G_{\varepsilon,i}$ and R_i takes time $\mathcal{O}(n+m)$. Now consider each iteration of the main loop. By Lemma 42, we can make FindNewSeed$(G_\varepsilon,R_i,\varepsilon,\boldsymbol{s},\boldsymbol{u},i)$ run in time $\mathcal{O}(n+m)$ while increasing the overall running time of the iteration of LearnClass by $\mathcal{O}(n+m)$. ShortestPath$(G_\varepsilon[R_i],s_i,s_h)$ takes time $\mathcal{O}(n+m)$ via BFS, and FindCutEdge$(\pi(s_i,s_h))$ clearly runs in time $\mathcal{O}(\log n)$. Adding u_i to $\boldsymbol{u}$ takes constant time. FindSeparator(G,u_i,u_j) runs in time $\mathcal{O}(n+m)$, see Lemma 41. Hence each iteration takes time $\mathcal{O}(n+m)$ overall.

To conclude, note that at most k iterations are made. ■

We can finally prove the running time bounds of Theorem 24 and Theorem 26. First, consider LearnBGC$(\mathcal{G}, \varepsilon, \boldsymbol{s})$. By Claim 1, computing G_ε takes time $\mathcal{O}(n+m)$. Each call to LearnClass$(G_\varepsilon, \varepsilon, \boldsymbol{s}, i)$ takes time $\mathcal{O}(k(n+m))$ by Lemma 43. As there are k classes, the $\mathcal{O}(k^2(n+m))$ bound of Theorem 24 follows. Now consider LearnHBGC. At every iteration, computing G^* takes time $\mathcal{O}(n+m)$ via a BFS. The SEED part takes time $\mathcal{O}(k) = \mathcal{O}(n)$, as does the construction of $\boldsymbol{s}^*$. The call to LearnClass takes time $\mathcal{O}(k(n+m))$ by Lemma 43. Writing out $\widehat{C_i}$ takes time $\mathcal{O}(n)$. This proves the bounds of Theorem 26.

5.5 Learning the radii and the convexity parameters

In this section we show how to use SEED queries to learn the radii of the classes and to adapt our algorithms when β or γ are unknown.

5.5.1 Learning the radii

This section proves Theorem 28.

Upper bounds. The upper bounds of Theorem 28 hinge on three observations. First, ε_i is actually the smallest ε such that all nodes of C_i belong to a single connected component of $G_\varepsilon[X]$, see above. Second, with $\mathcal{O}(1)$ SEED queries we can test whether C_i belongs to a single connected component of G for any given G. Third, if T is a minimum spanning tree of $\mathcal{G}$, then the connected components of $G_\varepsilon[X]$ are exactly the connected components of T_ε. Our algorithm starts by computing T; for each $i \in [k]$ it then performs a binary search to find the smallest edge weight ε such that C_i is connected in T_ε.

We start with a simple routine for checking if $C_i \subseteq V$ is connected in a graph $G = (V, E)$.

Algorithm 11 IsConnected(G, i)

1: $u :=$ SEED$(V(G), i)$
2: $U :=$ the connected component of u in G, or $\emptyset$ if $u =$ NIL
3: **return** (SEED$(V(G) \setminus U, i) =$ NIL)

Claim 2 IsConnected(G, i) *returns* TRUE *if and only if* $d_G(x, y) < \infty$ *for all* $x, y \in C_i$.

Now let T be a minimum spanning tree of $\mathcal{G} = (X, \mathcal{E}, d)$. For $\varepsilon > 0$ let T_ε be the forest formed by the edges (x, y) of T such that $d(x, y) \le \varepsilon$. Observe the following basic fact:

Claim 3 *The connected components of* T_ε *are the connected components of* G_ε.

As a consequence, we have:

Claim 4 IsConnected$(T_\varepsilon, i) =$ IsConnected(G_ε, i), *for any* $i \in [k]$ *and any* $\varepsilon > 0$.

We can now give the algorithm for learning the radius of a single class.

Lemma 44 *If* T *is a MST of* $\mathcal{G} = (X, \mathcal{E}, d)$, *then* GetEpsilon$(T, i)$ *returns* ε_i *in time* $\mathcal{O}(n \log n)$ *using* $\mathcal{O}(\log n)$ SEED *queries where* $n = |X|$.

Algorithm 12 GetEpsilon(T, i)

$\boldsymbol{w} := (w_1, \ldots, w_\ell)$, the distinct edge weights of T in increasing order
$lo := 1, \quad hi := \ell$
while $lo < hi$ **do**
    $mid := \lfloor \frac{lo+hi}{2} \rfloor$
    if IsConnected$(T_{w_{mid}}, i)$ **then** $hi := mid$ **else** $lo := mid + 1$
return w_{hi}

Proof Clearly GetEpsilon(T, i) makes $\mathcal{O}(\log n)$ iterations, since $\boldsymbol{w}$ has length $\mathcal{O}(n^2)$. Moreover, since T has $\mathcal{O}(n)$ edges, every call to IsConnected$(T_{w_{mid}}, i)$ takes time $\mathcal{O}(|T|) = \mathcal{O}(n)$. This gives the time bound of $\mathcal{O}(n \log n)$. Next we show that GetEpsilon(T, i) returns ε_i, or equivalently by Lemma 39, $w^* = \min\{w \in \boldsymbol{w} : C_i \text{ is connected in } G_w[X]\}$. Consider the test $lo < hi$ at beginning of each iteration. We claim that $w^* \le w_{hi}$. Indeed, C_i is connected in $G_{w_{hi}}[X]$ before the **while** loop starts, when $w_{hi} = w_\ell$, and at each iteration hi is set to mid only if IsConnected$(T_{w_{mid}}, i) =$ TRUE. We claim that $w^* \ge w_{lo}$, too. Indeed $w^* \ge w_0$ at the first iteration, and at each iteration lo is set to $mid+1$ only if IsConnected$(T_{w_{mid}}, i) =$ FALSE. Therefore at return time $w_{lo} = w_{hi} = w^*$, hence GetEpsilon(T, i) returns w^*. ■

We conclude with the algorithm to learn the radii and with the upper bounds of Theorem 28.

Algorithm 13 GetEpsilons$(\mathcal{G} = (X, \mathcal{E}, d), k)$

$T := \text{MST}(\mathcal{G})$
for $i = 1, \ldots, k$ **do** $\widehat{\varepsilon}_i :=$ GetEpsilon(T, i)
return $\widehat{\varepsilon}_1, \ldots, \widehat{\varepsilon}_k$

Lemma 45 GetEpsilons$(\mathcal{G}, k)$ *returns the radii* $\varepsilon_1, \ldots, \varepsilon_k$ *in time* $\mathcal{O}(m\,\alpha(m, n) + kn \log n)$ *using* $\mathcal{O}(k \log n)$ SEED *queries, where* $\alpha(m, m)$ *is the functional inverse of Ackermann's function (Chazelle, 2000).*

Proof One can compute MST$(\mathcal{G})$ in time $\mathcal{O}(m\,\alpha(m, n))$, see (Chazelle, 2000), and by Lemma 44 GetEpsilon(T, i) returns ε_i in time $\mathcal{O}(n \log n)$ using $\mathcal{O}(\log n)$ SEED queries. ■

Lower bounds. The lower bounds of Theorem 28 are given by:

Theorem 46 *For any* $k \ge 2$ *and any sufficiently large* n *there exists a distribution of (hereditary)* $(1/2, 1)$*-convex classifiers on* n *points such that any (randomized) algorithm needs* $\Omega(k \log \frac{n}{k})$ SEED *and/or label queries to learn the radii of all classes with constant probability.*

Proof Suppose first $k = 2$. Let $\mathcal{G} = (X, \mathcal{E}, d)$ be a path with increasing edge weights. Formally, let $X = [n]$, let $\mathcal{E} = \{\{j, j+1\} : j \in [n-1]\}$, and let $d(j, j+1) = 1 + \frac{\beta j}{n}$ for each $j \in [n-1]$. Draw j^* uniformly at random in $\{2, \ldots, n-1\}$, let $C_1 = \{1, \ldots, j^* - 1\}$, and $C_2 = \{j^*, \ldots, n\}$.

First, we prove that C_1 and C_2 are (β, γ)-convex with radii respectively $\varepsilon_1 = d(j^* - 1, j^*)$ and $\varepsilon_2 = d(n-1, n)$. For the connectivity, clearly $\rho(G_{\varepsilon_1}[C_1]) = \rho(G_{\varepsilon_2}[C_2]) = 1$. For the local metric margin note that the choice of β and d ensure $d \geq 1$ and $\varepsilon_1, \varepsilon_2 \leq \frac{3}{2}$. Thus $d(x, y) > \beta \max(\varepsilon_1, \varepsilon_2)$ for any two distinct $x, y \in X$. Therefore the local metric margin is satisfied. For geodesic convexity, note that there is only one edge between C_1 and C_2 in $\mathcal{G}$, thus no simple path between two points of one class intersects the other class. Thus, the properties of Definition 23 are satisfied by $\varepsilon_1, \varepsilon_2$. To show that Definition 25 is satisfied as well, we show $\varepsilon_1, \varepsilon_2$ are the smallest such values, see Lemma 39. To this end, simply note that $\varepsilon_1 = \min\{\zeta : \rho(G_\zeta[C_1]) = 1\}$, since $d(j^* - 1, j^*) = \varepsilon_1$ and $d(j, j+1) \leq \varepsilon_1$ for all $j = 1, \ldots, j^* - 1$. Similarly, $\varepsilon_2 = \min\{\zeta : \rho(G_\zeta[C_2]) = 1\}$, since $d(n-1, n) = \varepsilon_2$ and $d(j, j+1) \leq \varepsilon_1$ for all $j = j^*, \ldots, n-1$.

Finally, we show that any algorithm needs $\Omega(\log n)$ queries to learn ε_1. Clearly, if the algorithm learns ε_1 then it can also output the index j^*, which is a function of ε_1. Therefore, we show that finding j^* requires $\Omega(\log n)$ queries. First, we observe that SEED queries can be emulated by label queries. Consider indeed any $U \subseteq X$, and recall that if $U \cap C_i \neq \emptyset$ then $\textsc{seed}(U, i)$ may return *any* $x \in U \cap C_i$. We then let $\textsc{seed}(U, i)$ return $\min(U \cap C_i)$ if $i = 1$, and $\max(U \cap C_i)$ if $i = 2$. Now, observe that $\min(U \cap C_1) = \min U$ and $\max(U \cap C_1) = \max U$. Therefore $\textsc{seed}(U, i)$ can be emulated by asking the labels of $\min(U)$ and $\max(U)$. Since every label query reveals at most one bit of information and j^* is uniform over $\Omega(n)$ elements, learning j^* with constant probability requires $\Omega(\log n)$ label queries.

To extend the construction to $k > 2$, take $K = \frac{k}{2}$ disjoint weighted paths on $\frac{n}{K}$ nodes each (without loss of generality we can assume k is even). Each such path is weighted as in the construction above, with the weights of the h-th path all smaller than the weights of the $(h+1)$-th path. For each path, we draw j^* uniformly at random like above, and form two classes. The same proof used above shows that any algorithm needs $\Omega(k \log \frac{n}{k})$ queries to learn the radii of all classes with constant probability. ■

5.5.2 Learning β and γ

We adapt our algorithms, using SEED queries, to the case where β and/or γ are unknown (that is, it is known only that $\beta, \gamma \leq 1$). We begin with an alternative implementation of FindNewSeed, see Algorithm 14. It is easy to see that it has the same guarantees as FindNewSeed.

Algorithm 14 FindNewSeed-2$(R_i, \boldsymbol{c}, i)$

```
for each j ∈ c do
    s_j = SEED(R_i, j)
    if s_j ≠ NIL then return c, s_j
    c := c \ {j}
return c, NIL
```

Next, we describe LearnClass-2, an alternative implementation of LearnClass that learns one or both β, γ along the way. The subroutine UpdateParameters is described in the proofs below; it depends on whether one knows β, γ, or neither. It is crucial to observe that if

a classifier $\mathcal{C}$ is (hereditary) (β,γ)-convex then it is also (hereditary) $(\widehat{\beta},\widehat{\gamma})$-convex for all $\widehat{\beta}\le\beta$ and $\widehat{\gamma}\le\gamma$. This implies that if we run our algorithms with $\widehat{\beta},\widehat{\gamma}$ in place of β,γ then the output will be correct (and this holds for subroutines too).

Algorithm 15 LearnClass-2$(G_\varepsilon,\varepsilon,i)$

1: $s_i := \textsc{seed}(V,i)$
2: $G_{\varepsilon,i} := \{\text{the connected component of } s_i \text{ in } G_\varepsilon\},\ \ R_i := V(G_{\varepsilon,i})$
3: $\boldsymbol{c} := [k]$
4: **while** true **do**
5: $\quad \boldsymbol{c}, s_h := \text{FindNewSeed-2}(R_i,\boldsymbol{c},i)$
6: $\quad$ **if** $s_h = \textsc{nil}$ **then return** R_i
7: $\quad \pi(s_i,s_h) := \text{ShortestPath}(G_\varepsilon[R_i],s_i,s_h)$
8: $\quad (u_i,u_j) := \text{FindCutEdge}(\pi(s_i,s_h))$
9: $\quad$ **while** true **do**
10: $\quad\quad (S_i,S_j) := \text{FindSeparator}(G_\varepsilon,u_i,u_j)$
11: $\quad\quad$ **if** $\textsc{seed}(S_i,j) = \textsc{seed}(S_j,i) = \textsc{nil}$ **then break**
12: $\quad\quad \text{UpdateParameters}()$
13: $\quad G_{\varepsilon,i} := \{\text{the connected component of } s_i \text{ in } G_\varepsilon[R_i\cap S_i]\},\ \ R_i := V(G_{\varepsilon,i})$
14: $\quad \boldsymbol{c} := \boldsymbol{c}\setminus\{j\}$

We start by proving correctness and bounds for LearnClass-2 when only one of β,γ is unknown; when both are unknown we will only need to adapt UpdateParameters.

Lemma 47 *Let $\mathcal{C}$ be (β,γ)-convex. Suppose* LearnClass-2 *and its subroutines use $\widehat{\gamma}$ in place of γ and that each call to* UpdateParameters *halves $\widehat{\gamma}$. Then* LearnClass-2$(G_\varepsilon,\varepsilon,i)$ *returns C_i in time $O\Big(\big(k+\log\frac{\gamma_0}{\gamma_1}\big)(n+m)\Big)$ using $\mathcal{O}\Big(k\log n+\big(k+\log\frac{\gamma_0}{\gamma_1}\big)\,\mathcal{M}(\beta\gamma_1)\Big)$ label queries and at most $3k+2\log\frac{\gamma_0}{\gamma_1}$ SEED queries, where γ_0,γ_1 are the values of $\widehat{\gamma}$ at the beginning and at the end of the execution. The same holds with $\beta,\widehat{\beta},\beta_0,\beta_1$ in place of $\gamma,\widehat{\gamma},\gamma_0,\gamma_1$.*

Proof For the correctness, note that FindNewSeed-2 behaves exactly as FindNewSeed (see Lemma 36) and that when line 13 is executed (S_i,S_j) is an (i,j)-separator. The running time bounds follow by the analysis of Section 5.4, noting that FindNewSeed-2 runs in time $\mathcal{O}(k)=\mathcal{O}(n+m)$ and that lines 10–11 are repeated at most $\log\frac{\gamma_0}{\gamma_1}$ times.

For the label query bound, FindCutEdge is called at most k times and by Lemma 32 each call uses $\mathcal{O}(\log n)$ queries, while FindSeparator is called at most $k+\log\frac{\gamma_0}{\gamma_1}$ times and by Lemma 35 every call uses at most $\mathcal{M}(\beta\widehat{\gamma})\le\mathcal{M}(\beta\gamma_1)$ queries. For the SEED query bound, note that LearnClass-2 makes at most $2k+2\log\frac{\gamma_0}{\gamma_1}$ queries at line 11, while FindNewSeed-2 makes at most k queries overall as after each query some element is deleted from $\boldsymbol{c}$ (either by FindNewSeed-2 itself or by LearnClass-2 at line 14).

Finally, note that all arguments hold with $\beta,\widehat{\beta},\beta_0,\beta_1$ in place of $\gamma,\widehat{\gamma},\gamma_0,\gamma_1$. ■

We can now prove Theorem 27. First, assume $\mathcal{C}$ is (β,γ)-convex. The first claim follows by Lemma 47 when calling LearnClass-2$(G_\varepsilon,\varepsilon,i)$ for each $i\in[k]$. For the second claim, for every $j=0,1,\ldots$ let $\delta=2^{-j}$ and consider the $j+1$ pairs of values $(\delta,1),(2\delta,\frac{1}{2}),\ldots,(1,\delta)$. This

yields a succession of values for $(\widehat{\beta}, \widehat{\gamma})$. Starting with $\widehat{\beta} = \widehat{\gamma} = 1$, each call to UpdateParameters updates $\widehat{\beta}$ and $\widehat{\gamma}$ by moving to the next term in the succession. The claim follows by the same argument as above. Suppose now $\mathcal{C}$ is hereditary (β, γ)-convex. Consider LearnHBGC-2, see Algorithm 16; it is an adaptation of LearnHBGC that uses LearnClass-2 in place of LearnClass. Note that LearnHBGC-2 does not need to receive or compute the list of seed vertices; any necessary seed is found already by LearnClass-2. The correctness of LearnHBGC-2 follows by the same arguments used for LearnHBGC, see the proof of Theorem 26. The bounds on the number of queries follow by the same arguments of Theorem 27 (note that SEED is used only by LearnClass-2; the $\mathcal{O}(k^2)$ SEED queries appearing in Theorem 26 are not used anymore). The claim on the running time holds again by Theorem 27.

Algorithm 16 LearnHBGC-2$(X, \boldsymbol{\varepsilon})$

```
assume ε_1 ≤ ... ≤ ε_k
for i = 1, ..., k do
    G* := the connected component of s_i in G_{ε_i}[X],   X* := V(G*)
    Ĉ_i := LearnClass-2(G*, ε_i, i)
    output Ĉ_i
    X := X \ Ĉ_i
```

5.6 Lower bounds

In this section we show that some of our parameters and assumptions are necessary.

Theorem 48 (Dependence on $\mathrm{dens}(X)$**)** *Choose any $\beta, \gamma \in (0, 1)$. There is a distribution of (hereditary) (β, γ)-convex classifiers $\mathcal{C}$, where $n = |X| = \mu(X) = 2^{\mathrm{dens}(X)}$ is arbitrarily large, such that any (randomized) algorithm needs $\Omega(\mu(X)) = \Omega(2^{\mathrm{dens}(X)})$ label and/or* SEED *queries to learn $\mathcal{C}$ with constant probability. This holds even if $\beta, \gamma, \varepsilon$ are known.*

Proof Let $\mathcal{G} = (X, \mathcal{E}, d)$ where $(X, \mathcal{E})$ is the complete graph on n nodes and $d = 1$, and let $\mathcal{C} = (C_1, C_2)$ be a uniform random partition of X. We claim that $\mathcal{C}$ is (hereditary) (β, γ)-convex for $\varepsilon = 1$. The connectivity of $G_\varepsilon[C_1]$ and $G_\varepsilon[C_2]$ holds trivially. The local metric margin holds as well, since any two distinct points $x, y \in X$ satisfy $d(x, y) = d > \beta d$, as $\beta < 1$. To see that geodesic convexity holds, too, note that for any $x, y \in C_1$ we have $d_{G_\varepsilon}(x, y) \leq 1$ and any (simple) path between x and y that contains a point in $X \setminus C_1$ has length at least $2 > (1 + \gamma) d_{G_\varepsilon}(x, y)$. Finally, note that $G_{\varepsilon'}$ is an independent set for any $\varepsilon' < \varepsilon$. Hence $\mathcal{C}$ satisfies both Definition 23 and Definition 25.

Since $\mathcal{C}$ is chosen uniformly at random among all bipartitions of X, then $\Omega(n)$ label and/or SEED queries are needed to learn $\mathcal{C}$ exactly with non-vanishing probability. Noting that $\mu(X) = n$ and recalling that $\mathrm{dens}(X) = \log_2 \mu(X)$ completes the proof. ■

Theorem 49 (Necessity of seeds) *Choose any $\beta, \gamma \in (0, 1]$. There is a distribution of (hereditary) (β, γ)-convex classifiers $\mathcal{C}$, with $n = |X|$ arbitrarily large, such that any (randomized) algorithm needs $\Omega(n)$ label queries to learn $\mathcal{C}$ with non-vanishing probability if no seed nodes are given. This holds even if $\gamma, \beta, \varepsilon$ are known.*

Proof Let $X = \mathrm{UP} \cup \mathrm{LOW} \subset \mathbb{R}^2$ where (see Figure 5):

$$\mathrm{UP} = \bigcup_{j=1}^{n/3} \{(2j, 1)\}, \quad \mathrm{LOW} = \bigcup_{j=1}^{2n/3} \{(j, 0)\}$$

Moreover let $\mathcal{G} = (X, \mathcal{E}, d)$ where $\mathcal{E} = \binom{X}{2}$ and d is the Euclidean distance.

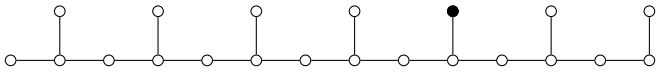

Figure 5: the graph G_ε for $\varepsilon = 1$. All points are in C_1, save for the filled point in C_2, chosen uniformly at random in UP.

Choose a point z uniformly at random from UP, and let $C_1 = X \setminus \{z\}$ and $C_2 = \{z\}$. One can check that all properties of Definition 23 and Definition 25 hold for $\varepsilon = 1$. In particular, in G_ε no simple path between two points of C_1 contains z. Hence $\mathcal{C}$ is (hereditary) (β, γ)-convex. Learning $\mathcal{C}$ requires finding z, and finding z with constant probability requires $\Omega(n)$ queries in expectation even if γ, β and ε are known. ■

Next, we show that relaxing our notion of (β, γ)-convexity by allowing a different semimetric for each class leads to an $\Omega(n)$ query lower bound. To this end, for each $i \in [k]$ let $\mathcal{G}^i = (X, \mathcal{E}^i, d^i)$ be the weighted graph encoding the semimetric d^i. The notation and definitions formulated for $\mathcal{G}$ apply to $\mathcal{G}^i$ with the obvious modifications.

Definition 50 *Let $\beta, \gamma \in (0, 1]$. A k-partition $\mathcal{C} = (C_1, \ldots, C_k)$ of X is* personalized (β, γ)-convex *if there is $\varepsilon > 0$ such that for each $i \in [k]$ the following properties hold:*

1. connectedness*: $G^i_\varepsilon[C_i]$ is connected*
2. local metric margin*: for all $x, y \in X$, if $x \in C_i$ and $y \notin C_i$, then $d^i(x, y) > \beta\varepsilon$*
3. geodesic convexity with margin*: if $x, y \in C_i$, then in G^i_ε any simple path between x and y of length at most $(1 + \gamma) d_{G^i_\varepsilon}(x, y)$ lies in C_i*

The smallest value of ε satisfying the three properties is called the radius *of the classes.*

We prove:

Theorem 51 (Necessity of a universal semimetric) *Choose any $\beta, \gamma \in (0, 1)$. There is a distribution of personalized (β, γ)-convex 2-class classifiers $\mathcal{C}$, with $n = |X|$ arbitrarily large, such that any (randomized) algorithm needs $\Omega(n)$* SEED *and/or label queries to learn $\mathcal{C}$ with non-vanishing probability, even if $\gamma, \beta, \varepsilon$ are known.*

Proof Let $\varepsilon > 0$, let $X = \{a_1, a_2\} \cup M$ where $M = [n]$, and for $i = 1, 2$ and all $x, y \in X$ let:

$$d^i(x, y) = \begin{cases} \varepsilon & x \in M, y = a_i \\ 2\varepsilon & \text{otherwise} \end{cases}$$

Note that G^i_ε is the union of an isolated vertex and a star centered in a_i with M as leaves. Note also that for any $X' \subseteq X$ the set $C_i = \{a_i\} \cup X'$ satisfies all properties of Definition 50.

Thus any bipartition (C_1, C_2) of X such that $a_i \in C_i$ for $i = 1, 2$ is personalized (β, γ)-convex. Taking the distribution of classifiers where $a_i \in C_i$ and every $x \in M$ is independently assigned to C_i with probability $\frac{1}{2}$ proves the claim. ■

6. Conclusions and future work

We have shown that, in very different settings, one can devise meaningful notions of margins that allow for efficient and exact active learning of multiclass classifiers. Interestingly, this can be achieved with very different techniques, from boosting learners with one-sided errors through expansion of convex hull, to computing abstract separators that play the role of halfspaces in vector spaces. Our work also shows that exact learning of multiclass classifiers requires an inevitable exponential dependence on the dimensionality of the space. There are two main natural questions that our work leaves open. The first one is to what extent these results can be adapted to the non-realizable case, that is, to noisy oracles. The second one is whether a parsimonious use of more powerful queries, such as SEED, can remove the exponential dependence on the dimensionality of the ambient space while keeping the running time polynomial.

Acknowledgments

The authors gratefully acknowledge partial support by the Google Focused Award "Algorithms and Learning for AI" (ALL4AI). Nicolò Cesa-Bianchi acknowledges the financial support from the FAIR (Future Artificial Intelligence Research) project, funded by the NextGenerationEU program within the PNRR-PE-AI scheme (M4C2, investment 1.3, line on Artificial Intelligence) and the EU Horizon CL4-2022-HUMAN-02 research and innovation action under grant agreement 101120237, project ELIAS (European Lighthouse of AI for Sustainability).

References

Peyman Afshani, Ehsan Chiniforooshan, Reza Dorrigiv, Arash Farzan, Mehdi Mirzazadeh, Narges Simjour, and Hamid Zarrabi-Zadeh. On the complexity of finding an unknown cut via vertex queries. In *Proceedings of the International Computing and Combinatorics Conference (COCOON)*, pages 459–469, 2007. doi: 10.1007/978-3-540-73545-8_45.

Nir Ailon, Anup Bhattacharya, and Ragesh Jaiswal. Approximate correlation clustering using same-cluster queries. In *Proceedings of the Latin American Symposium on Theoretical Informatics (LATIN)*, pages 14–27, 2018a. doi: 10.1007/978-3-319-77404-6_2.

Nir Ailon, Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Approximate clustering with same-cluster queries. In *Proceedings of Innovations in Theoretical Computer Science (ITCS)*, pages 40:1–40:21, 2018b. doi: 10.4230/LIPIcs.ITCS.2018.40.

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988. doi: 10.1023/A:1022821128753.

Hassan Ashtiani, Shrinu Kushagra, and Shai Ben-David. Clustering with same-cluster queries. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3216–3224, 2016.

Josh Attenberg and Foster Provost. Why label when you can search? Alternatives to active learning for applying human resources to build classification models under extreme class imbalance. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, page 423–432, 2010. doi: 10.1145/1835804.1835859.

Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1):49–54, 2012. doi: 10.1016/j.ipl.2011.10.006.

Maria Florina Balcan and Steve Hanneke. Robust interactive learning. In *Proceedings of the Annual Conference Computational Learning Theory (COLT)*, pages 20.1–20.34, 2012.

Maria-Florina Balcan and Phil Long. Active and passive learning of linear separators under log-concave distributions. In *Proceedings of the Annual Conference Computational Learning Theory (COLT)*, pages 288–316, 2013.

Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In *Proceedings of the Annual Conference Computational Learning Theory (COLT)*, pages 35–50, 2007.

Shai Ben-David, Nicolo Cesa-Bianchi, David Haussler, and Philip Long. Characterizations of learnability for classes of $\{0, ..., n\}$-valued functions. *Journal of Computer and System Sciences*, 50(1):74–86, 1995. doi: 10.1006/jcss.1995.1008.

Alina Beygelzimer, Daniel J Hsu, John Langford, and Chicheng Zhang. Search improves label for active learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, page 3350–3358, 2016.

Yonatan Bilu and Nathan Linial. Are stable instances easy? *Combinatorics, Probability and Computing*, 21(5):643–660, 2012. doi: 10.1017/S0963548312000193.

Marco Bressan, Nicolò Cesa-Bianchi, Silvio Lattanzi, and Andrea Paudice. Exact recovery of mangled clusters with same-cluster queries. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9324–9334, 2020.

Marco Bressan, Nicolò Cesa-Bianchi, Silvio Lattanzi, and Andrea Paudice. Exact recovery of clusters in finite metric spaces using oracle queries. In *Proceedings of the Annual Conference Computational Learning Theory (COLT)*, pages 775–803, 2021a.

Marco Bressan, Nicolò Cesa-Bianchi, Silvio Lattanzi, and Andrea Paudice. On margin-based cluster recovery with oracle queries. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 25231–25243, 2021b.

Nicolò Cesa-Bianchi, Claudio Gentile, Fabio Vitale, and Giovanni Zappella. Active learning on trees and graphs. In *Proceedings of the Annual Conference Computational Learning Theory (COLT)*, pages 320–332, 2010.

Bernard Chazelle. A minimum spanning tree algorithm with inverse-Ackermann type complexity. *Journal of the ACM*, 47(6):1028–1047, 2000. doi: 10.1145/355541.355562.

Eli Chien, Antonia Tulino, and Jaime Llorca. Active learning in the geometric block model. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3641–3648, 2020.

Vincent Cohen-Addad and Karthik Srikanta. Inapproximability of clustering in *L-p* metrics. In *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 519–539, 2019. doi: 10.1109/FOCS.2019.00040.

Amit Daniely and Shai Shalev-Shwartz. Optimal learners for multiclass problems. In *Proceedings of the Annual Conference Computational Learning Theory (COLT)*, pages 287–316, 2014.

Gautam Dasarathy, Robert Nowak, and Xiaojin Zhu. S2: An efficient graph based active learning algorithm with application to nonparametric classification. In *Proceedings of the Annual Conference Computational Learning Theory (COLT)*, pages 503–522, 2015.

Sanjoy Dasgupta. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 337–344, 2005.

Sanjoy Dasgupta, Adam Tauman Kalai, and Claire Monteleoni. Analysis of Perceptron-based active learning. In *Proceedings of the Annual Conference Computational Learning Theory (COLT)*, pages 249–263, 2005.

Scott Doyle, James Monaco, Michael Feldman, John Tomaszewski, and Anant Madabhushi. An active learning based classification strategy for the minority class problem: application to histopathology annotation. *BMC Bioinformatics*, 12(1), 2011.

Ran El-Yaniv and Yair Wiener. Active learning via perfect selective classification. *Journal of Machine Learning Research*, 13(2), 2012.

Akshay Gadde, Eyal En Gad, Salman Avestimehr, and Antonio Ortega. Active learning for community detection in stochastic block models. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, pages 1889–1893, 2016. doi: 10.1109/ISIT.2016.7541627.

Buddhima Gamlath, Sangxia Huang, and Ola Svensson. Semi-supervised algorithms for approximately optimal and accurate clustering. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 57:1–57:14, 2018. doi: 10.4230/LIPIcs.ICALP.2018.57.

Apostolos Giannopoulos. Notes on isotropic convex bodies, 2003. URL `http://users.uoa.gr/~apgiannop/isotropic-bodies.pdf`.

Alon Gonen, Sivan Sabato, and Shai Shalev-Shwartz. Efficient active learning of halfspaces: an aggressive approach. *Journal of Machine Learning Research*, 14(1):2583–2615, 2013.

Lee-Ad Gottlieb and Robert Krauthgamer. Proximity algorithms for nearly doubling spaces. *SIAM Journal on Discrete Mathematics*, 27(4):1759–1769, 2013. doi: 10.1137/120874242.

Lee-Ad Gottlieb, Aryeh Kontorovich, and Pinhas Nisnevitch. Nearly optimal classification for semimetrics. *The Journal of Machine Learning Research*, 18(1):1233–1254, 2017.

Andrew Guillory and Jeff Bilmes. Active semi-supervised learning using submodular functions. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 274–282, 2011.

Steve Hanneke. *Theoretical Foundations of Active Learning*. PhD thesis, Carnegie Mellon University, 2009. URL `http://reports-archive.adm.cs.cmu.edu/anon/ml2009/CMU-ML-09-106.pdf`. AAI3362265.

Steve Hanneke. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 7(2-3):131–309, 2014. doi: 10.1561/2200000037.

Steve Hanneke and Liu Yang. Minimax analysis of active learning. *The Journal of Machine Learning Research*, 16(1):3487–3602, 2015.

Max Hopkins, Daniel Kane, and Shachar Lovett. The power of comparisons for actively learning linear classifiers. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6342–6353, 2020a.

Max Hopkins, Daniel Kane, Shachar Lovett, and Gaurav Mahajan. Noise-tolerant, reliable active classification with comparison queries. In *Proceedings of the Annual Conference Computational Learning Theory (COLT)*, pages 1957–2006, 2020b.

Max Hopkins, Daniel Kane, Shachar Lovett, and Michal Moshkovitz. Bounded memory active learning through enriched queries. In *Proceedings of the Annual Conference Computational Learning Theory (COLT)*, pages 2358–2387, 2021.

Fritz John. Extremum problems with inequalities as subsidiary conditions. *Birkhäuser Boston eBooks*, Jan 1985. doi: 10.1007/978-1-4612-5412-6_25.

Daniel Kane, Shachar Lovett, Shay Moran, and Jiapeng Zhang. Active classification with comparison queries. In *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 355–366, 2017. doi: 10.1109/FOCS.2017.40.

Ravi Kannan, László Lovász, and Miklós Simonovits. Isoperimetric problems for convex bodies and a localization lemma. *Discrete & Computational Geometry*, 13(3):541–559, 1995. doi: 10.1007/BF02574061.

Leonid G Khachiyan. Rounding of polytopes in the real number model of computation. *Mathematics of Operations Research*, 21(2):307–320, 1996. doi: 10.1287/moor.21.2.307.

J. Kivinen. Learning reliably and with one-sided error. *Mathematical Systems Theory*, 28(2): 141–172, 1995. doi: 10.1007/BF01191474.

Andrey Kupavskii. The VC-dimension of k-vertex d-polytopes. *Combinatorica*, 40(6):869–874, 2020. doi: 10.1007/s00493-020-4475-4.

László Lovász and Santosh Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006. doi: 10.1137/S009753970544727X.

Wolfgang Maass and György Turán. Lower bound methods and separation results for on-line learning models. *Machine Learning*, 9(2):107–145, 1992. doi: 10.1007/BF00992674.

Arya Mazumdar and Soumyabrata Pal. Semisupervised clustering, and-queries and locally encodable source coding. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6489–6499, 2017.

Arya Mazumdar and Barna Saha. Query complexity of clustering with side information. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4682–4693, 2017a.

Arya Mazumdar and Barna Saha. Clustering with noisy queries. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5788–5799, 2017b.

Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, USA, 1995. ISBN 0521474655. doi: 10.1017/CBO9780511814075.

Márton Naszódi, Fedor Nazarov, and Dmitry Ryabogin. Fine approximation of convex bodies by polytopes. *American Journal of Mathematics*, 142(3):809–820, 2020. doi: 10.1353/ajm.2020.0018.

Ignacio M. Pelayo. *Geodesic convexity in graphs*. Springer-Verlag New York, 2013. doi: 10.1007/978-1-4614-8699-2.

Ronald L. Rivest and Robert Sloan. Learning complicated concepts reliably and usefully. In *Proceedings of the Annual Conference Computational Learning Theory (COLT)*, pages 69–79, 1988.

Benjamin I. P. Rubinstein, Peter L. Bartlett, and J. Hyam Rubinstein. Shifting: One-inclusion mistake bounds and sample compression. *Journal of Computer and System Sciences*, 75 (1):37–59, 2009. doi: 10.1016/j.jcss.2008.07.005.

Barna Saha and Sanjay Subramanian. Correlation clustering with same-cluster queries bounded by optimal cost. In *Proceedings of the European Symposium on Algorithms (ESA)*, pages 81:1–81:17, 2019. doi: 10.4230/LIPIcs.ESA.2019.81.

Florian Seiffarth, Tamás Horváth, and Stefan Wrobel. Maximal closed set and half-space separations in finite closure systems. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 21–37, 2019.

Burr Settles. Active learning. Technical report, University of Wisconsin-Madison, 2012. URL `http://digital.library.wisc.edu/1793/60660`.

Maximilian Thiessen and Thomas Gärtner. Active learning of convex halfspaces on graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 23413–23425, 2021.

Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ACM international conference on Multimedia (ICM)*, page 107–118, 2001. doi: 10.1145/500141.500159.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018. doi: 10.1017/9781108231596.

Sharad Vikram and Sanjoy Dasgupta. Interactive Bayesian hierarchical clustering. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 48, pages 2081–2090, 2016.

Yair Wiener, Steve Hanneke, and Ran El-Yaniv. A compression technique for analyzing disagreement-based active learning. *Journal of Machine Learning Research*, 16:713–745, 2015.

Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1537–1544, 2004.

Pan Zhang, Cristopher Moore, and Lenka Zdeborová. Phase transitions in semisupervised clustering of sparse networks. *Physical Review E*, 90(5):052802, 2014. doi: 10.1103/PhysRevE.90.052802.