Active Learning Convex Halfspaces on Graphs

Maximilian Thiessen [1] **Thomas Gärtner** [1]

Abstract

We systematically study the query complexity of learning geodesically convex halfspaces on graphs. Geodesic convexity is a natural generalisation of Euclidean convexity and allows the definition of convex sets and halfspaces on graphs. We prove upper bounds on the query complexity linear in the treewidth and the minimum hull set size but only logarithmic in the diameter. We show tight lower bounds corresponding to well-established separation axioms and identify the Radon number as a central parameter bounding the query complexity and the VC dimension. While previous bounds typically depend on the cut size of the labelling, all parameters in our bounds can be computed from the unlabelled graph. We empirically compare our proposed approach with other active learning algorithms and provide evidence that communities in real-world graphs are often convex.

1. Introduction

We present a systematic characterisation of active learning geodesically convex halfspaces on graphs. While unlabelled graphs such as social networks are readily accessible, obtaining many labels is a tedious and labour intensive task. Active learning reduces the labelling effort by iteratively and carefully selecting the vertices to be labelled. Abstract convexity spaces in general and geodesic convexity on graphs are natural generalisations of Euclidean convexity. However, while learning convex classes and halfspaces has been intensively researched in the machine learning community for over half a century, geodesic convexity on graphs has been largely overlooked so far. To test the hypothesis that geodesic convexity is indeed a natural concept on graphs, we performed a preliminary experimental study that confirms that many communities in social networks are indeed convex. For a concise presentation of our theoretical results in this paper, we concentrate on binary classification, that is, active learning geodesically convex halfspaces.

[1]Machine Learning, TU Wien, Vienna, Austria. Correspondence to: Maximilian Thiessen <maximilian.thiessen@tuwien.ac.at>.

After introducing the terminology and tools of graph and abstract convexity theory (in Section 2), we show that general, worst-case upper and lower bounds holding for the set of all graphs are inherently loose. To derive tighter lower bounds, we structure the space of graphs along the so called separation axioms of abstract convexity theory. We show that an upper bound—related to treewidth, diameter, and hull set of the graph—is tight across all sets of graphs induced by the separation axioms and derive tighter lower bounds for each of these sets. This allows us to significantly close the gap between upper and lower bounds and present the first precise characterisation of active learning geodesically convex halfspaces on graphs (in Section 3). Note that a similar characterisation does not carry over to Euclidean convexity: While the separation axioms partition the space of all graphs into non-empty sets, all separation axioms hold in all linear spaces such as the Euclidean ones. We discuss related areas like active learning convex sets in Euclidean spaces, passive learning geodesically convex halfspaces, active learning bounds in terms of the cut size between the classes, and other related learning theoretic concepts (in Section 4). In particular, we also provide a novel bound on the VC-dimension of geodesically convex halfspaces. After presenting an empirical validation of our assumptions and approach (in Section 5), we conclude with a discussion of promising furture work.

2. Convexity spaces

In this section, we introduce important and necessary concepts of convexity theory. For a more thorough introduction on convexity theory, we refer the reader to van de Vel (1993) and Pelayo (2013).

For a set X and a family $\mathcal{C} \subseteq 2^X$ of subsets, the tuple $(X, \mathcal{C})$ is a *convexity space* if *(i)* $\emptyset, X \in \mathcal{C}$, *(ii)* $\mathcal{C}$ is closed under intersection, and *(iii)* $\mathcal{C}$ is closed under nested union. For finite set systems, property *(iii)* always holds. Any set in $\mathcal{C}$ is called *convex*. If a set C and its complement $X \setminus C$ are convex, both are called *halfspaces*. Two disjoint sets A, B are *separated* by a halfspace C if $A \subseteq C$ and $B \subseteq X \setminus C$. A mapping $\sigma : 2^X \to 2^X$ is a *convex hull* (or *closure*) *operator* if for all $A, B \subseteq X$ with $A \subseteq B$ *(i)* $\sigma(\emptyset) = \emptyset$, *(ii)* $\sigma(A) \subseteq \sigma(B)$, *(iii)* $A \subseteq \sigma(A)$, and *(iv)* $\sigma(\sigma(A)) = \sigma(A)$. Any convexity space $(X, \mathcal{C})$ induces a convex hull operator by $\sigma(A) = \bigcap\{C \in \mathcal{C} \mid A \subseteq C\}$. Note that a set $A \subseteq X$ is convex, that is $A \in \mathcal{C}$, if and only if is equal to its convex hull, $A = \sigma(A)$.

Convexity spaces can be characterised by their ability to separate sets via halfspaces, following a related notion in topological spaces.

Definition 1 (Separation axioms (van de Vel, 1993))**.** *A convexity space $(X, \mathcal{C})$ is:*

> S_1 *if and only if each singleton $x \in X$ is convex.*
>
> S_2 *if and only if each pair of distinct elements x, y is halfspace separable.*
>
> S_3 *if and only if each convex set C and elements $x \in X \setminus C$ are halfspace separable.*
>
> S_4 *if and only if any two disjoint convex sets are halfspace separable.*

If S_1 holds, the remaining axioms are increasingly stronger: $S_2 \Leftarrow S_3 \Leftarrow S_4$. For classical convex sets in Euclidean spaces, all four separation axioms hold (Kakutani, 1937).

We also use the following concepts. A set $H \subseteq X$ is a *hull set* if its convex hull is the whole space, $\sigma(H) = X$. For $A, B \subseteq X$, the set $A/B = \{x \in X \mid A \cap \sigma(B \cup \{x\}) \neq \emptyset\}$ is the *extension of* A (*away from* B). For $a, b \in X$, the extension $\{a\}/\{b\}$ is called a *ray* a/b. Two disjoint sets A_1, A_2 form a *partition* of $A \subseteq X$ if $A_1 \cup A_2 = A$. A partition A_1, A_2 of A is a *Radon partition* if $\sigma(A_1) \cap \sigma(A_2) \neq \emptyset$. The *Radon number* is the minimum number r such that any subset of X of size r or larger has a Radon partition.

A particular type of convexity is an *interval convexity*. Apart from the convex hull $\sigma(\cdot)$, they have an *interval mapping* $I : X \times X \to 2^X$ such that for all $x, y \in X$, *(i)* $x, y \in I(x, y)$ and *(ii)* $I(x, y) = I(y, x)$. We call $I(x, y)$ the *interval* between x and y. We denote $I(A) = \bigcup_{a,b \in A} I(a, b)$. A set A in an interval convexity space is convex if and only if $A = I(A)$. Denoting $I^1(\cdot) = I(\cdot)$ and $I^{k+1}(\cdot) = I(I^k(\cdot))$, it holds that $\sigma(A) = \bigcup_{k=1}^{\infty} I^k(A)$. A well-known instance of interval convexity spaces are metric spaces (X, d). There, the interval contains all the points for which the triangle inequality holds with equality: $I_d(x, y) = \{z \in X \mid d(x, y) = d(x, z) + d(z, y)\}$. In Euclidean space, this corresponds to a line segment and leads to the classical notion of convexity.

We are interested in the geodesic convexities induced by graphs. The geodesic convexity (or shortest path convexity) of a connected graph $G = (V, E)$ is given by the interval mapping I_d, where d is the *shortest path distance* in the graph. For unweighted graphs it is the number of edges on any shortest path and for graphs with edge weights, $w : E \to \mathbb{R}_{>0}$, it is the sum of the edge weights on any shortest path. Using this definition, a subset of vertices $C \subseteq V$ is convex if and only if every shortest path with endpoints in C stays in C, corresponding to the Euclidean case.

We denote the Radon number of the induced geodesic convexity space of a graph as $r(G)$ and the size of the minimum hull set as $h(G)$. The notion of rays simplifies for graphs: for an edge $\{a, b\}$ in an unweighted graph $a/b = \{v \in V \mid d(a, v) < d(b, v)\}$ and for a weighted graph $a/b = \{v \in V \mid d(b, v) = d(b, a) + d(a, v)\}$. A vertex v is *extreme* if $V \setminus \{v\}$ is convex, that is, $\{v\}$ is a halfspace. We will denote the set of extreme vertices of a graph G as $\mathrm{Ext}(G)$. In the unweighted case, v is extreme if and only if its neighbours form a clique. The *diameter* $d(G)$ of a weighted or unweighted graph G is the maximum number of edges in any shortest path in G.

We say a graph is S_i, for $i = 1, \dots, 4$, if for the induced geodesic convexity space the respective separation axiom holds. Moreover, there are graphs that are S_i but not S_{i+1} for $i = 1, 2, 3$ (Bandelt, 1989), see supplementary for some examples. In this regard, graph convexity spaces are more general than the Euclidean one, for which always all four separation axioms hold.

3. Active learning halfspaces on graphs

Having introduced the necessary concepts from convexity theory, we now present the main theoretical results of the paper. In active learning on graphs, an undirected graph $G = (V, E)$ with unknown vertex labels $\lambda : V \to \{0, 1\}$ is given and the goal is to accurately predict all labels using as few as possible iterative vertex queries $\lambda(v)$. Edges between vertices with different labels are called *cut edges*. We consider active learning on graphs where the vertices of the same class form a geodesically convex set. We achieve upper (Theorem 6) and lower bounds (Theorem 8) on the number of queries required to deduce all labels of the graph for binary classification on connected undirected graphs. In this case, the classes form halfspaces, we call the labelling *halfspace separable*, and denote the least number of queries required to identify any halfspace separable labelling in the worst-case, the *query complexity*, by $\mathrm{qc}(G)$. To reduce the gap (Proposition 9) between the upper and lower bounds, we inspect the graph families induced by the separation axioms and derive increasingly tighter lower bounds (Theorem 10). All proofs, multi-class settings, and adaptations for directed and disconnected graphs can be found in the supplementary.

3.1. Upper bounds on the query complexity

To derive a first simple upper bound we consider that one immediate consequence of the halfspace assumption is that any shortest path can have at most one cut edge. Therefore, either the endpoints of a shortest path P have the same label or we can find a cut edge by binary search with at most $1 + \lceil \log d(G) \rceil$ queries, as $|V(P)| - 1 \leq d(G)$, where log is the base 2 logarithm. We can generalise this approach to the whole graph using *shortest path covers*. A shortest path cover $\mathcal{S}$ is a set of shortest paths whose vertices jointly cover the graph: $\bigcup_{P \in \mathcal{S}} V(P) = V(G)$. Binary searching each of the paths in the cover gives our first upper bound.

Proposition 2. *For any weighted graph G with minimum shortest path cover $\mathcal{S}^*$ the query complexity can be bounded as* $\mathrm{qc}(G) \leq |\mathcal{S}^*|(2 + \lceil \log d(G) \rceil)$.

By considering a particular fully connected weighted graph with an arbitrary number of paths with independent cut edges, we can show that this bound is tight:

Proposition 3. *For any $d, s \in \mathbb{N}$, there exists a weighted graph G with diameter $d(G) = d$ and minimum shortest path cover $\mathcal{S}^*$ of size s such that $qc(G) \geq |\mathcal{S}^*| \log d(G)$.*

For many graphs, however, we can do much better, as the labels of different paths in the cover are typically not independent. Consider for instance a k-dimensional hypergrid with edge length ℓ, that is, the Cartesian product of k paths with ℓ vertices each: More than $k^{\ell-1}/\ell$ shortest paths are needed to cover the hypergrid but $1 + \lceil \log(k\ell) \rceil$ queries suffice to identify the labelling of the graph.

This leads directly to an idea for improving the bound and deriving our algorithm for active learning geodesically convex halfspaces: after each query, we deduce additional labels using convex hulls and extensions. Our proposed algorithm first queries the vertices of a hull set until it finds two vertices with different labels or concludes that all vertices in the graph have the same label. Then it performs a binary search on a shortest path between these two vertices to identify a cut edge $\{a, b\}$. Finally, Algorithm 1 efficiently queries the remaining vertices. It initialises the sets A and B with the convex hull of the rays a/b and b/a, respectively. The main loop queries any vertex v in the region $\hat{W}_{=ab} = V(G) \setminus (\sigma(a/b) \cup \sigma(b/a))$ and updates the known labels with the new vertex using convex hulls and extensions. Lemma 4 states the queries needed in the first two steps.

Lemma 4. *Let G be a graph with halfspace separable labels. We can either find a cut edge or determine that all vertices of the graph have the same label using $h(G) + \lceil \log d(G) \rceil$ queries.*

Lemma 5 shows that the initialisation and updates of A and B in Algorithm 1 are indeed valid.

Lemma 5. *Let G be a weighted graph with halfspace separable labels λ and let $A \subseteq \{x \in V(G) \mid \lambda(x) = 1\}$ and $B \subseteq \{x \in V(G) \mid \lambda(x) = 0\}$. For any $v \in \sigma(A/B)$, it holds that $\lambda(v) = 1$*

We can bound the number of queried vertices of Algorithm 1 using the set $W^*_{=ab} = \hat{W}_{=ab} \setminus \{v \in \hat{W}_{=ab} \mid \exists w \in \hat{W}_{=ab}$ such that $v \in w/a \cap w/b\}$. Intuitively, it is enough to know the labels of the vertices in $W^*_{=ab}$ to deduce all labels of $\hat{W}_{=ab}$. Theorem 6 shows that Algorithm 1 is correct and summarises our main upper bound.

Algorithm 1 Halfspace querying

```
Input: graph $G$, cut edge $\{a, b\} \in E(G)$
$A := \sigma(a/b)$,   $B := \sigma(b/a)$
while $A \cup B \neq V(G)$ do
  query any vertex $v \in V(G) \setminus (A \cup B)$
  if $\lambda(v) = \lambda(a)$ then
    $A := \sigma((A \cup \{v\})/B)$,   $B := \sigma(B/(A \cup \{v\}))$
  else
    $A := \sigma(A/(B \cup \{v\}))$,   $B := \sigma((B \cup \{v\})/A)$
```

Theorem 6. *Let G be a weighted or unweighted graph. Finding a cut edge with a minimum hull set and then applying Algorithm 1, results in the query complexity*

$$\mathrm{qc}(G) \leq h(G) + \lceil \log d(G) \rceil + \max_{\{a,b\} \in E(G)} |W^*_{=ab}| .$$

Note that Algorithm 1 does not need to explicitly compute the set $W^*_{=ab}$. To give a more intuitive bound for the unweighted case, we relate the last term of the bound to the *treewidth* $\mathrm{tw}(G)$, a well-studied graph-theoretic parameter (Bodlaender, 1996) measuring the 'tree-likeness' of a graph: Any cut edge $\{a, b\}$ together with $W^*_{=ab}$ has a complete bipartite minor $K_{2,|W^*_{=ab}|}$ which implies $\mathrm{tw}(G) \geq |W^*_{=ab}|/2$ (Bodlaender et al., 1997). This means that many real-world graphs such as molecules (Horváth and Ramon, 2010) and infrastructure networks (Maniu et al., 2019) which have small treewidth, and thus small $W^*_{=ab}$, can be queried efficiently by Algorithm 1. Corollary 7 summarises this result.

Corollary 7. *Let G be a unweighted graph with hull set H and treewidth $\mathrm{tw}(G)$. Then, for the query complexity it holds that* $\mathrm{qc}(G) \leq h(G) + \lceil \log d(G) \rceil + 2\,\mathrm{tw}(G)$.

3.2. Lower bounds under separation axioms

Having discussed our upper bounds on the query complexity, we now turn to a simple lower bound based on the extreme vertices $\mathrm{Ext}(G)$ of the graph, recall each of them is a halfspace by definition.

Theorem 8. *For a weighted graph G with extreme vertices $\mathrm{Ext}(G)$, it holds that* $\mathrm{qc}(G) \geq |\mathrm{Ext}(G)|$.

Considering a graph consisting of paths that coincide only in their endpoints, we can show that this bound is also tight and therefore that the gap between it and both our upper bound can be arbitrarily large.

Proposition 9. *For any $d, s \in \mathbb{N}$, there exists an unweighted graph G with $d(G) = d$ and minimum shortest path cover $\mathcal{S}^*$ of size s, such that* $\mathrm{qc}(G) \leq 2$ *and* $\mathrm{Ext}(G) = \emptyset$.

This shows that the lower bound in Theorem 8 is best possible for general graphs, even when $d(G)$ and $|\mathcal{S}^*|$ are large. The main insight of our work is that structuring the set of all graphs along separation axioms gives increasingly tighter lower bounds.

Theorem 10. *For a weighted graph G, the following holds for the query complexity* $\mathrm{qc}(G)$.

- *G is S_2:* $\mathrm{qc}(G) \geq \max(\log d(G), |\operatorname{Ext}(G)|)$,
- *G is S_3:* $\mathrm{qc}(G) \geq \max(\log d(G), h(G))$, *and*
- *G is S_4:* $\mathrm{qc}(G) \geq \max(\log d(G), h(G), r(G) - 1)$.

Each bound is tight in the respective family and stronger axioms lead to tighter bounds.

To see the final statement, notice that all extreme vertices are by definition contained in any hull set and hence $|\operatorname{Ext}(G)| \leq h(G)$. The two parameters $h(G)$ and $r(G)$ are independent of each other, in the sense that there are graphs with $h(G) < r(G)$ but also graphs with $r(G) < h(G)$. Theorems 6 and 10 together imply tight bounds (Table 1, supplementary) for various graph families studied in metric graph theory (Bandelt and Chepoi, 2008), convexity theory, and machine learning (Seiffarth et al., 2019).

3.3. Computational aspects

Above, we derived lower and upper bounds on the query complexity of learning halfspaces on graphs. We now discuss computational aspects of our algorithm and bounds.

Computing a minimum shortest path cover $\mathcal{S}^*$ is an open problem (Manuel, 2018). We show that greedily covering the graph with longest possible shortest paths gives a logarithmic approximation:

Theorem 11. *For a given weighted graph G, we can compute a $\mathcal{O}(\log d(G))$-approximation for the minimum shortest path cover $\mathcal{S}^*$ in time $\mathcal{O}(|V|^4)$.*

Computing the Radon number of a graph is even hard to approximate within any factor sublinear in $|V|$ (Coelho et al., 2015). However, we can bound the Radon number with the size of the largest clique minor (the *Hadwiger number*), which in turn is upper bounded by the treewidth.

Proposition 12. *For any graph G, it holds that $r(G) \leq \mathrm{tw}(G) + 2$.*

Computing intervals and hulls is possible in $\mathcal{O}(|V(G)|^3)$ time by first solving the all-pairs-shortest-path problem and afterwards check the triangle inequality for each triplet of vertices. The diameter $d(G)$ is computable by a generalised Dijkstra algorithm, see supplementary. Computing the set of extreme vertices is possible in time $\mathcal{O}(|V||E|)$ by evaluating the neighbourhood of each vertex. **Computing the minimum hull set size** $h(G)$ is APX-hard (Coelho et al., 2015). For graphs of bounded treewidth (Kanté et al., 2019) however, the problem is solvable in polynomial time and our algorithm achieves then the bound of Corollary 7 in polynomial time. For other graphs heuristic approaches can compute a hull set H and achieve the query complexity bounds in Theorem 6 with $|H|$ queries instead of $h(G)$. **Deciding whether all separation axioms hold** is possible in time $\mathcal{O}(|V(G)|^7)$ by examining all quadruplets of vertices, so-called *Pasch* combinations (Seiffarth et al., 2019). Such strategies for the other cases are unknown.

4. Discussion

In this section, we discuss our assumption as well as related active and *passive* learning results.

Convexity in real-world datasets In some applications convexity-based assumptions are already implicitly or explicitly used. For example in gene similarity networks, it is well-known that shortest paths largely preserve functional relationships and the category in the gene ontology (Zhou et al., 2002). Similar results were achieved on protein-protein-interaction networks, where shortest paths between known cancer-related genes are used to identify a candidate set of novel genes that are likely to be cancer-related, as well (Li et al., 2012; 2013). Aside from protein and gene networks, Marc and Šubelj (2018) and Šubelj et al. (2019) recently found that connected subgraphs of real-world graphs like collaboration networks are often convex. We performed preliminary experiments on different networks with ground truth communities. Table 2 (in the supplementary) shows the number of convex communities in six datasets from SNAP (Leskovec and Sosič, 2016). We found that on the DBLP network more than 85% of the 5000 communities are convex, supporting the results of Šubelj et al. (2019). On the Amazon product network we have similar results with roughly 80%. On the Youtube social network we found that roughly 60% of the communities are convex. On the remaining datasets less than a third of the communities are convex. Clearly convexity on graphs is application dependent.

Relationships to Euclidean convexity Halfspaces in Euclidean space have been at the core of various learning algorithms, like the Perceptron (Rosenblatt, 1958; Novikoff, 1963) and support vector machines (Boser et al., 1992; Cortes and Vapnik, 1995), and are still an active research area (Daniely, 2016; Diakonikolas et al., 2020; Hopkins et al., 2020). Graph and other finite convexity spaces, however, are not equivalent to the Euclidean convexity in the sense that it is not always possible to embed a graph into $\mathbb{R}^m$ or the other way around while preserving the convex hull $\sigma(\cdot)$. Euclidean spaces are always S_4 while graphs do not have to be. Further discussion are in the supplementary.

Cut-based bounds Most previous bounds for active learning on graphs are linear in the size of the set of cut edges C or *cut vertices* ∂C, that are the vertices incident to the cut edges. In the batch setting, one such bound was given by Guillory and Bilmes (2009), see supplementary for further discussions. We will present two bounds in the iterative setting here on the k-dimensional hypercube graph labelled according to some halfspace. Theorem 6 shows that one binary search on a shortest path connecting opposing corners is enough to deduce all labels with $\mathcal{O}(\log k)$ queries. Afshani et al. (2007) gave a worst-case query complexity of $c + |\partial C|(1 + \log(|V|/|\partial C|))$ under balancedness assumptions where $c \geq 2$ is the number of connected components in G without the cut edges C. As $\partial C = V$, this results in a bound $> |V|$ for the hypercube. The more recent bound of Dasarathy et al. (2015), besides the number of cut vertices, depends on the *clusteredness* κ of the cut, which is at least the largest distance between any two cut vertices of the same class in the same connected component of the graph with cut edges removed. Their upper bound is at least $|\partial C|(1 + \log \kappa)$. On the hypercube $\kappa \geq k - 1$ and $\partial C = V$ resulting again in a bound $> |V|$. We see that both bounds can be vacuous and exponentially larger than our bound.

Margin-based bounds Bressan et al. (2021) study the query complexity of identifying geodesically convex clusters in a nearest neighbour graph given by a semi-metric space using *same-cluster* queries. Even though they rely on additional assumptions, like a large margin, their upper bound is very similar to our main result. They use same-cluster queries, which are binary queries on a pair of points to test whether they are in the same cluster, instead of vertex label queries. However, these two query types are equivalent in terms of their query complexity up to a multiplcative factor given by the number of classes. They work on an ε neighbour graph, assume that the weight of any cut edge is $> \beta\varepsilon$ for an $\beta \in (0, 1]$, and use an interval with margin $\gamma \in (0, 1]$: $I_\gamma(a, b) = \{x \in V \mid d(a, x) + d(x, b) \leq (1 + \gamma)d(a, b)\}$ to define convex sets. The regular geodesic convexity corresponds to $\gamma = 0$. Using these, they state an $\mathcal{O}\left(\log |X| + \left(\frac{6}{\beta\gamma}\right)^{\mathrm{dens}(X)}\right)$ upper bound on the query complexity for any semi-metric space X with binary labels under the assumption that two vertices with different labels are already given, where $\mathrm{dens}(X)$ is the *density dimension* of the space (Gottlieb and Krauthgamer, 2013). Even though they make stronger assumptions to achieve this bound, there exist graph families where our bound in Theorem 6 is exponentially better. For example, take the $2 \times k$ grid. If we allow any halfspace, the margin is $\gamma < 2/k$. This results in a bound of at least $\log(2k) + (3k)^{\mathrm{dens}(X)}$ with $\mathrm{dens}(X) \geq 1$ while our bound predicts that $2 + \log(k + 1)$ queries are enough. Even with a large margin our bound remains slightly better.

Passive learning halfspaces Passive learning halfspaces on graphs was studied by Seiffarth et al. (2019; 2020) and de Araújo et al. (2019). Seiffarth et al. (2019) rely on the separation axiom S_4 to bound the number of convex hull computations needed to construct a halfspace. Stadtländer et al. (2021) investigate the more general problem of learning *weakly convex* sets in a metric space (X, d). Here, weakly convex sets with parameter $\theta \geq 0$ are given by the interval mapping $I_\theta(a, b) = \{x \in X \mid d(a, x) + d(x, b) = d(a, b) \leq \theta\} \cup \{a, b\}$ corresponding to our interval mapping on a pair a, b with $d(a, b) \leq \theta$. Moran and Yehudayoff (2019) established that the VC dimension of halfspaces of any convexity space is smaller than the Radon number of the space. Building on that, we can show that VC dimension is exactly determined by the Radon number in S_4 convexity spaces. This includes the classical result that in $\mathbb{R}^m$ the VC dimension of halfspaces is $m + 1$.

Proposition 13. *The VC dimension of the hypothesis class of halfspaces of an S_4 convexity space is exactly one less than the Radon number of the space.*

For S_4 graphs, $r(G)$ determines the VC dimension and gives a lower bound on the query complexity (Theorem 10), establishing the Radon number as a central parameter for learning geodesic halfspaces. Proposition 12 together with the above bound of Moran and Yehudayoff (2019) establishes:

Proposition 14. *The VC dimension of halfspaces in a graph G is smaller than $\mathrm{tw}(G) + 2$.*

5. Experiments

Having discussed our bounds on the query complexity and the conceptional benefits of the halfspace assumption in terms of theoretical upper bounds, we want to see whether the same holds experimentally, as well. Two practical querying strategies based on Algorithm 1 are *greedy* and *selective sampling*.The greedy strategy tries to maximise the number of known vertex labels with each query. For that, it queries the vertex v that would maximise the minimum number of known labels after the update of A and B in Algorithm 1. More precisely, after the update with a vertex v we will know the labels of the set $\sigma((A \cup \{v\})/B) \cup \sigma(B/(A \cup \{v\}))$ or the set $\sigma(A/(B \cup \{v\})) \cup \sigma((B \cup \{v\})/A)$. The greedy strategy thus selects the vertex v that maximises the size of the smaller one of these two sets, corresponding to the minimum number of labels we will know after the update. As performing the described greedy maximisation for each query is rather computationally expensive, we propose a simpler strategy based on selective sampling (Cohn et al., 1994) as an alternative. Instead of performing the actual maximisation, this strategy simply picks a vertex uniformly at random from $V(G) \setminus (A \cup B)$ in Algorithm 1. To find the first cut edge, we use a hull set as in Lemma 4. As discussed, computing a minimum hull set is in general not tractable.

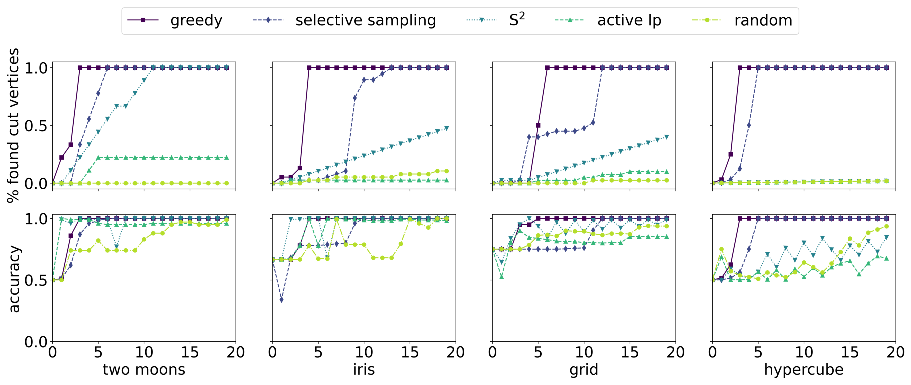


Figure 1: Found cut vertices and accuracy plotted against number of queries

Therefore, we propose to use the selection strategy (greedy or selective sampling) also to iteratively construct a hull set until two vertices with different labels are found. For that we start with $A, B = \emptyset$ and select vertices using the selection strategy and update A and B according to Algorithm 1, until we find two different labels. Implicitly this process will construct a (typically non-minimum) hull set H. That way the query complexity bound in Theorem 6 still holds, as it is independent of the specific selection strategy used; only the $h(G)$ is replaced with the size of the heuristically computed H. We compare these two strategies with the state-of-the-art graph-based active learning algorithm S^2 (Dasarathy et al., 2015), the classical active label propagation of Zhu et al. (2003b), and baseline non-active random sampling. The code is available at Github.[1] We use the ε-nearest-neighbour graphs of two moons [2] and Iris [3]. We also use a 20×20 grid and a 2^{10} hypercube, labelled with a random but fixed halfspace. More details and further experiments are in the supplementary.

Query evaluation Dasarathy et al. (2015) propsed to count the number of found cut vertices after each query as a measure to compare querying algorithms without relying on any classification method. An efficient graph-based active learner should detect these as fast as possible. The results for 20 iterative queries in the upper half of Figure 1 show that our approaches identify the cut vertices more efficiently than S^2, active label propagation, and random sampling by deducing many labels after each query. Our greedy approach requires less than 5 queries to deduce all vertex labels on all four datasets emphasising the strength of convexity-based assumptions. The selective sampling approach is only slightly worse using 2-8 queries more. The other approaches can find at most one cut vertex per query as they do not rely on convexity assumptions.

Predictive performance We also evaluated the predictive performance of the chosen queries. Our two approaches predict the labels of the computed hulls and default to the majority of known labels when they do not know the label of a vertex. The other three approaches perform label propagation (Zhu et al., 2003a) with the default Gaussian similarity and $\sigma = 1$. In the lower half of Figure 1, we can see that again our greedy approach identifies the vertex labels of the whole graph on all datasets within 5 queries. The accuracy of the other three approaches is significantly worse, even unstable. The main reason is that label propagation favours small cuts (weighted by similarity), and thus has problems predicting halfspaces with large cuts.

6. Conclusion and future work

In this work, we employed convexity theoretical concepts to achieve novel bounds on the query complexity of learning halfspaces on graphs. On the one hand, we derived two general upper bounds, one based on shortest path covers and the other based on the diameter, hull sets, and the quantity $\max_{\{a,b\}\in E(G)} |W^*_{=ab}|$, which we in turn bounded by the treewidth of the graph. On the other hand, we stated a simple general lower bound based on extreme vertices and increasingly tighter lower bounds for graph families corresponding to separation axioms. We compared our bounds to previous results based on the cut-size and found that the halfspace assumption enabled learning with large cuts, while previous bounds often become vacuous in this case. On the practical side, we found that ground-truth communities in real-world networks tend to be convex. We compared our approaches with two previous active learning strategies and found that our algorithms require considerably less queries to identify the correct halfspace. Unifying the ideas of Seiffarth et al. (2019), Stadtländer et al. (2021), and Bressan et al. (2021) with ours by developing general querying and prediction strategies is an interesting research direction.

[1] `https://github.com/maksim96/subsetml_active_graph_halfspaces`

[2] `make_moons(noise=0.1, random_state=0)` in `scikit-learn` (Pedregosa et al., 2011); $\varepsilon = 0.14$

[3] first class vs the other two (Fisher, 1936); $\varepsilon = 0.3$

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 *International Computing and Combinatorics Conference*, 2007.

Hans-Jürgen Bandelt. Graphs with intrinsic s3 convexities. *Journal of graph theory*, 13(2):215–228, 1989.

Hans-Jürgen Bandelt and Victor Chepoi. Metric graph theory and geometry: a survey. *Contemporary Mathematics*, 453:49–86, 2008.

Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *ICML*, 2001.

Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on computing*, 25(6):1305–1317, 1996.

Hans L. Bodlaender, Jan van Leeuwen, Richard Tan, and Dimitrios M. Thilikos. On interval routing schemes and treewidth. *information and computation*, 139(1):92–109, 1997.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, 1992.

Marco Bressan, Nicolò Cesa-Bianchi, Silvio Lattanzi, and Andrea Paudice. Exact recovery of clusters in finite metric spaces using oracle queries. In *COLT*, 2021.

Nicolò Cesa-Bianchi, Claudio Gentile, Fabio Vitale, and Giovanni Zappella. Active learning on trees and graphs. In *COLT*, 2010.

Gary Chartrand and Ping Zhang. The geodetic number of an oriented graph. *European Journal of Combinatorics*, 21(2):181–189, 2000.

Victor Chepoi. Separation of two convex sets in convexity structures. *Journal of Geometry*, 50(1-2):30–51, 1994.

Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

Erika M. M. Coelho, Mitre C. Dourado, and Rudini M. Sampaio. Inapproximability results for graph convexity parameters. *Theoretical Computer Science*, 600:49–58, 2015.

David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15 (2):201–221, 1994.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Amit Daniely. Complexity theoretic limitations on learning halfspaces. In *STOC*, 2016.

Gautam Dasarathy, Robert Nowak, and Xiaojin Zhu. S2: An efficient graph based active learning algorithm with application to nonparametric classification. In *COLT*, 2015.

Paulo H.M. de Araújo, Manoel Campêlo, Ricardo C. Corrêa, and Martine Labbé. The geodesic classification problem on graphs. *Electronic Notes in Theoretical Computer Science*, 346:65–76, 2019.

Ilias Diakonikolas, Vasilis Kontonis, Christos Tzamos, and Nikos Zarifis. Learning halfspaces with massart noise under structured distributions. In *COLT*, 2020.

Reinhard Diestel. *Graph Theory*. Springer, 5th edition, 2017.

Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

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

Andrew Guillory and Jeff Bilmes. Label selection on graphs. In *NIPS*, 2009.

Andrew Guillory and Jeff Bilmes. Active semi-supervised learning using submodular functions. In *UAI*, 2011.

Max Hopkins, Daniel M. Kane, Shachar Lovett, and Gaurav Mahajan. Point location and active learning: Learning halfspaces almost optimally. In *FOCS*, 2020.

Tamás Horváth and Jan Ramon. Efficient frequent connected subgraph mining in graphs of bounded tree-width. *Theoretical Computer Science*, 411(31-33):2784–2797, 2010.

Shizuo Kakutani. Ein Beweis des Satzes von M. Eidelheit über konvexe Mengen. *Proceedings of the Imperial Academy*, 13(4):93–94, 1937.

Mamadou Moustapha Kanté, Thiago Marcilon, and Rudini Sampaio. On the parameterized complexity of the geodesic hull number. *Theoretical Computer Science*, 1: 1–18, 2019.

Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8 (1):1, 2016.

Bi-Qing Li, Tao Huang, Lei Liu, Yu-Dong Cai, and Kuo-Chen Chou. Identification of colorectal cancer related genes with mrmr and shortest path in protein-protein interaction network. *PloS one*, 7(4):e33393, 2012.

Bi-Qing Li, Jin You, Lei Chen, Jian Zhang, Ning Zhang, Hai-Peng Li, Tao Huang, Xiang-Yin Kong, and Yu-Dong Cai. Identification of lung-cancer-related genes with the shortest path approach in a protein-protein interaction network. *BioMed research international*, 2013.

Silviu Maniu, Pierre Senellart, and Suraj Jog. An experimental study of the treewidth of real-world graph data. In *International Conference on Database Theory*, 2019.

Paul Manuel. Revisiting path-type covering and partitioning problems. *arXiv preprint arXiv:1807.10613*, 2018.

Tilen Marc and Lovro Šubelj. Convexity in complex networks. *Network Science*, 6(2):176–203, 2018.

Shay Moran and Amir Yehudayoff. On weak epsilon-nets and the Radon number. In *Symposium on Computational Geometry*, 2019.

Albert B. Novikoff. On convergence proofs for perceptrons. Technical report, Stanford Research Institute, Menlo Park, California, 1963.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Ignacio M. Pelayo. *Geodesic convexity in graphs*. Springer, 2013.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Florian Seiffarth, Tamás Horváth, and Stefan Wrobel. Maximal closed set and half-space separations in finite closure systems. In *ECMLPKDD*, 2019.

Florian Seiffarth, Tamás Horváth, and Stefan Wrobel. Maximum margin separations in finite closure systems. In *ECMLPKDD*, 2020.

J. L. Sobrinho. Algebra and algorithms for QoS path computation and hop-by-hop routing in the internet. *IEEE/ACM Transactions on Networking*, 10(4):541–550, 2002.

Eike Stadtländer, Tamás Horváth, and Stefan Wrobel. Learning weakly convex sets in metric spaces. In *ECMLPKDD*, 2021.

Lovro Šubelj, Dalibor Fiala, Tadej Ciglarič, and Luka Kronegger. Convexity in scientific collaboration networks. *Journal of Informetrics*, 13(1):10–31, 2019.

Marcel L.J. van de Vel. *Theory of convex structures*. Elsevier, 1993.

Xianghong Zhou, Ming-Chih J Kao, and Wing Hung Wong. Transitive functional annotation by shortest-path analysis of gene expression data. *Proceedings of the National Academy of Sciences*, 99(20):12783–12788, 2002.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003a.

Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *Workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, 2003b.

A. Proof details

We state some facts that we will often use in the following proofs. For that let $V_+ = \{x \in V(G) \mid \lambda(x) = 1\}$ and $V_- = \{x \in V(G) \mid \lambda(x) = 0\}$ be the labelled halfspaces.

Fact.

1. *If $S_+ \subseteq V_+$ then $I(S_+) \subseteq \sigma(S_+) \subseteq V_+$. Equivalently:*
   If $S_- \subseteq V_-$ then $I(S_-) \subseteq \sigma(S_-) \subseteq V_-$.

2. *If $S_+ \subseteq V_+$ and $S_- \subseteq V_-$ then $S_+/S_- \subseteq V_+$ and $S_-/S_+ \subseteq V_-$.*

Proof.

1. Since $S_+ \subseteq V_+$, we have $\sigma(S_+) \subseteq \sigma(V_+) = V_+$ by the monotonicity of $\sigma(\cdot)$. $I(S_+) \subseteq \sigma(S_+)$ by definition. $S_- \subseteq V_-$ follows analogously.

2. Assume there is an $x \in (S_+/S_-) \cap V_-$. By definition of S_+/S_- there is a $y \in S_+ \cap \sigma(S_- \cup \{x\})$. Using the previous Fact 1, we get $\sigma(S_- \cup \{x\}) \subseteq V_-$ but also $S_+ \subseteq V_+$, which implies that $y \in V_+ \cap V_-$, clearly a contradiction. $S_-/S_+ \subseteq V_-$ follows analogously.

□

Proposition 2. *For any weighted graph G with minimum shortest path cover $\mathcal{S}^*$ the query complexity can be bounded as* $\mathrm{qc}(G) \leq |\mathcal{S}^*|(2 + \lceil \log d(G) \rceil)$.

Proof. For each path in $\mathcal{S}^*$ first query the endpoints. If they are the same, the whole path must have this label by the halfspace assumption. Otherwise, it has exactly one cut edge, that is, an edge where the label change. We can find this edge with binary search using $\leq \lceil \log d(G) \rceil$. □

Proposition 3. *For any $d, s \in \mathbb{N}$, there exists a weighted graph G with diameter $d(G) = d$ and minimum shortest path cover $\mathcal{S}^*$ of size s such that $qc(G) \geq |\mathcal{S}^*| \log d(G)$.*

Proof. Fix $d, s \in \mathbb{N}$. Consider the unit vectors $e_1, \ldots, e_s \in \mathbb{R}^{s+1}$ and shifted copies of them: $e_j + i \cdot e_{s+1}$, for $j \in \{1, \ldots, s\}$ and $i \in \{1, \ldots, d\}$. Construct the fully connected graph of these $s \cdot d$ points with edge weights corresponding to the Euclidean distance. The minimum shortest path cover will have size s and the diameter will be d. We can label each shortest path P_j of the form $(e_j, e_j + e_{s+1}, \ldots, e_j + d \cdot e_{s+1})$ with at most one cut edge at an arbitrary position, independently of the other paths. The resulting labelling is halfspace separable, because the only shortest path between two vertices on different paths P_j,$P_{j'}$ is the edge connecting them. Hence, to deduce all labels we will need $\geq \log d$ queries per path. □

Lemma 4. *Let G be a graph with halfspace separable labels. We can either find a cut edge or determine that all vertices of the graph have the same label using $h(G) + \lceil \log d(G) \rceil$ queries.*

Proof. Let H be a hull set of size $h(G)$. The convex hull of H is the whole graph. If all vertices in H have the same label, the whole graph must have this label by Fact 1. Otherwise, we can take two vertices in H with different labels and find one cut edge on a shortest path between them using $\lceil \log d(G) \rceil$ queries. □

Lemma 5. *Let G be a weighted graph with halfspace separable labels λ and let $A \subseteq \{x \in V(G) \mid \lambda(x) = 1\}$ and $B \subseteq \{x \in V(G) \mid \lambda(x) = 0\}$. For any $v \in \sigma(A/B)$, it holds that $\lambda(v) = 1$.*

Proof. The claim directly follows by applying Fact 2 to A/B and then Fact 1 to $\sigma(A/B)$. □

Theorem 6. *Let G be a weighted or unweighted graph. Finding a cut edge with a minimum hull set and then applying Algorithm 1, results in the query complexity*

$$\mathrm{qc}(G) \leq h(G) + \lceil \log d(G) \rceil + \max_{\{a,b\} \in E(G)} |W^*_{=ab}|.$$

Proof. By Lemma 4, we can use $h(G) + \lceil \log d(G) \rceil$ queries to find the first cut edge $\{a, b\}$ or deduce that the all vertices have the same label. Applying Lemma 5 to a/b and b/a, we only have to deduce the labels of $\hat{W}_{=ab} = V(G) \setminus (\sigma(a/b) \cup \sigma(b/a))$.

Algorithm 1 queries iteratively a vertex label in $V \setminus (\sigma(A/B) \cup \sigma(B/A)) \subseteq \hat{W}_{=ab}$. We want to bound the number of queries made by Algorithm 1 by the size of the set $W^*_{=ab} = \{v \in \hat{W}_{=ab} \mid \nexists w \in \hat{W}_{=ab}$ such that $v \in (w/a) \cap (w/b)\}$. For that we show that the algorithm deduces at least one new vertex label from $W^*_{=ab}$ after each update of A and B. If we query a vertex in $W^*_{=ab}$ the claim holds, as we never query the same vertex twice, because it is afterwards either in A or B.

If we query a vertex $v \in \hat{W}_{=ab} \setminus W^*_{=ab}$, we know by the definition of $W^*_{=ab}$ that there is a vertex $w \in W^*_{=ab}$ with $v \in (w/a) \cap (w/b)$, which means that $v \in \sigma(\{a, w\}) \cup \sigma(\{b, w\})$. By applying Fact 1 to one of the convex hulls $\sigma(\{a, w\})$ or $\sigma(\{b, w\})$, depending on the label of w, we get the label of v. The vertex v will be hence correctly added to A or B in the following update where the convex hulls $\sigma(A \cup \{w\}/B) \supseteq \sigma(\{a, w\})$ or $\sigma(B \cup \{w\}/A) \supseteq \sigma(\{b, w\})$ are used.

This shows that after at most $|W^*_{=ab}|$ iterations, we will know the labels of $W^*_{=ab}$. It remains to show that we will also have all missing labels in $\hat{W}_{=ab} \setminus W^*_{=ab}$. For that, let $v \in \hat{W}_{=ab} \setminus W^*_{=ab}$, which implies that there is a $w \in W^*_{=ab}$

such that $v \in (w/a) \cap (w/b)$. By applying Fact 2 to w/a or w/b, depending on the known label of w, we get the label of v. Hence, v will be correctly added to A or B in the following update, because $w/a \subseteq \sigma(B \cup \{w\}/A)$ or $w/b \subseteq \sigma(B \cup \{w\}/A)$. □

Corollary 7. *Let G be a unweighted graph with treewidth $\operatorname{tw}(G)$. For the query complexity it holds that $\operatorname{qc}(G) \leq h(G) + \lceil \log d(G) \rceil + 2\operatorname{tw}(G)$.*

Proof. Let $\{a, b\}$ be a cut edge. We claim that in an unweighted graph, the vertices a and b together with $W^*_{=ab}$ form a $K_{2,|W^*_{=ab}|}$ minor of G. To show that, we first prove that any vertex $v \in W^*_{=ab}$ is adjacent to a/b and b/a, that is, there is an edge going from v to a/b and to b/a. Assume this does not hold. Then, without loss of generality, v is not adjacent to a/b and there exists a $w \in W^*_{=ab}$ such that $w \in I(a, v)$. By the definition of $\hat{W}_{=ab} \supseteq W^*_{=ab}$, the distance from a and b to any vertex in $\hat{W}_{=ab}$ is the same and hence $d(b, v) = d(a, v) = d(a, w) + d(v, w) = d(b, w) + d(v, w)$. This means $w \in I(b, v)$, which implies that $v \in (w/a) \cap (w/b)$. This is a contradiction to $v \in W^*_{=ab}$. It follows, that v is adjacent to a/b and b/a.

For each $v \in W^*_{=ab}$, we can identify an edge $e_v = \{v_a, v\}$ with $v_a \in a/b$ on a fixed shortest path from a to v. The same holds for a fixed shortest path from b to v with an edge $e'_v = \{v_b, v\}$ and $v_b \in b/a$. Contracting all edges on these shortest paths, but the e_v and e'_v for all v results in a complete bipartite graph $K_{2,|W^*_{=ab}|}$, with the two vertices a and b as one partition and all v as the other. This complete bipartite graph is a minor of G. The largest k such that $K_{2,k}$ is a minor of G is bounded as $k \leq 2\operatorname{tw}(G)$ by the treewidth (Bodlaender et al., 1997), which in turn bounds $|W^*_{=ab}| \leq k \leq 2\operatorname{tw}(G)$. □

Theorem 8. *For a weighted graph G with extreme vertices $\operatorname{Ext}(G)$, it holds that $qc(G) \geq |\operatorname{Ext}(G)|$.*

Proof. As any extreme vertex s is a halfspace, we have $s \notin \sigma(V(G) \setminus \{s\})$. Thus, to decide between the labelling according to the halfspaces $(\{s\}, V(G) \setminus \{s\})$ and the case where all vertices have the same label, we have to query s. □

The above result still holds, if we only allow proper halfspaces (neither empty nor $V(G)$) in general, as can be seen by the clique graph where each vertex is extreme.

Proposition 9. *For any $d, s \in \mathbb{N}$ with $s \geq 3$ and $d \geq 2$, there exists an unweighted graph G with diameter $d(G) = d$ and minimum shortest path cover $\mathcal{S}^*$ of size s, such that $\operatorname{qc}(G) \leq 2$ and $\operatorname{Ext}(G) = \emptyset$. The graph G is S_1 but not S_2.*

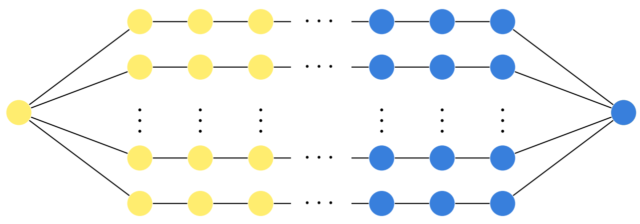


Figure 2: Example graph $P_{d,s}$ that is not S_2 and has at most two proper halfspaces.

Proof. Consider the graph $P_{d,s}$ consisting of s interior vertex-disjoint paths of length d with same endpoints, see Figure 2. It follows that $d(G) = d$, the minimum shortest path cover $\mathcal{S}^*$ has size s and the number of extreme vertices is 0. We claim that if d is even, the graph has only two non-proper halfspaces $V(P_{d,s})$ and $\emptyset$ and if d is odd additionally two halfspaces cutting each path in two halves of same length.

To show that, we identify three of the paths with vertices $(v_1, \dots, v_{d+1})$,$(v_1, w_2, \dots, w_d, v_{d+1})$, and $(v_1, x_2, \dots, x_d, v_{d+1})$. Without loss of generality we can assume that there is a cut edge v_k, v_{k+1} with $k < (d+1)/2$ and labels $\lambda(v_k) = 0$, $\lambda(v_{k+1}) = 1$. The ray v_{k+1}/v_k implies by Fact 2 that $\lambda(v_{d+1}) = 1$, and the opposite ray v_k/v_{k+1} that $\lambda(v_1) = 0$, $\lambda(w_{d+1-k}) = 0$ and $\lambda(x_{d+1-k}) = 0$. Finally, $v_{d+1} \in I(w_{d+1-k}, x_{d+1-k})$, which is a contradiction. This implies that a cut edge $\{v_k, v_{k+1}\}$ can only be placed at $k = (d+1)/2$ for an odd d.

This also means that $P_{d,s}$ is not S_2. □

The graph $P_{2,3}$ is just the complete bipartite $K_{2,3}$.

Theorem 10. *For a weighted graph G the following holds for the query complexity $\operatorname{qc}(G)$:*

- *if G is S_2, then $\operatorname{qc}(G) \geq \max(\log d(G), |\operatorname{Ext}(G)|)$,*
- *if G is S_3, then $\operatorname{qc}(G) \geq \max(\log d(G), h(G))$, and*
- *if G is S_4, then $\operatorname{qc}(G) \geq \max(\log d(G), h(G), r(G) - 1)$.*

Each bound is tight in the respective family and stronger axioms lead to tighter bounds.

Proof.

- Any two different vertices with a halfspace (S_2): The lower bound using extreme vertices $\operatorname{Ext}(G)$ holds for all graphs by Theorem 8 and hence also for S_2 graphs. We can place a cut edge anywhere on any fixed shortest path with size of the diameter on any S_2 graph, which requires $\log d$ queries to be found in the worst-case.

- Any convex set and a vertex not in this set can be separated with a halfspace (S_3):
  On S_3 graphs with minimum hull set H^*, we can separate any vertex $h \in H^*$ from the ramaining hull set. This means that we have to query all vertices in H^* in the worst-case.

- Any two disjoint convex sets can be separated with a halfspace (S_4):
  For any graph, there exists a set R of size $r(G) - 1$ that has no Radon partition. In S_4 graphs we can thus extend any partition of R to halfspaces and thus have to query each vertex in R in the worst-case.

□

We shortly discuss the bounds in Table 1. It contains graph families often studied in metric graph theory (Bandelt and Chepoi, 2008) and convexity theory, such as *partial cubes*, *weakly median* and *meshed* graphs. Meshed graphs satisfy the following *triangle condition* (Bandelt and Chepoi, 2008): for any three vertices u, v, w with $1 = d(v, w) < d(u, v) = d(u, w)$ there exists a vertex x adjacent to v and w such that $d(u, v) = d(u, x) + 1$. For example, the triangle conditions holds for *meshed graphs*, including *weakly modular* graphs. S_4 weakly modular graphs are the *weakly median* graphs (Chepoi, 1994). Bipartite graphs and their S_3 variant *partial cubes* (Bandelt, 1989) are also a classical topic in graph convexity theory. For bipartite graphs, $S_2 \iff S_3$. Table 1 also contains trees, and $K_{2,3}$ minor-free graphs a superset of *outerplanar graphs*, studied by Seiffarth et al. (2019).

Trees The lower bound follows by Theorem 10, because rees are S_4 and $h(G) = |\operatorname{Ext}(G)|$. The upper bound follows by Theorem 6 by the fact that for any edge $\{a, b\}$, we have that $\hat{W}_{=ab} = \emptyset$.

$K_{2,3}$ minor-free graphs Seiffarth et al. (2019) shows that $K_{2,3}$ minor-free graphs are S_4, which gives the lower bound by Theorem 10. The upper bound follows by Corollary 7. In fact, if k is a constant the upper bound also holds for any $K_{2,k}$ minor-free graph.

Partial cubes Partial cubes are exactly the S_2 (equivalently S_3) bipartite graphs, which gives the lower bound by Theorem 10. The upper bound holds by Theorem 6 as in bipartite graphs for any edge $\{a, b\}$, we have that $\hat{W}_{=ab} = \emptyset$. If there would be an $x \in \hat{W}_{=ab}$ it would form a cycle of odd length with a and b, which is not possible in bipartite graphs.

Weakly median graphs Weakly median graphs are S_4 and hence by Theorem 10 we get the lower bound. The upper bound holds for the much broader class of meshed graphs. In fact, we only need the triangle condition to achieve that the number of queries of Algorithm 1 is $\leq r(G) - 1$.

By the triangle condition, we have that for any $v \in W^*_{=ab}$, there exists a common neighbour w of a and b that is on the shortest path of a to v, respectively b to v. So if $d(a, v) = d(b, v) > 1$, we would have $v \in w/a \cup w/b$ and thus $v \notin W^*_{=ab}$, which is not possible. So, $v = w$ and adjacent to a and b.

$|W^*_{=ab}| \leq r(G)$ does not hold in general and therefore we can not use the query bound of Theorem 6. Instead, we will directly bound the size of the set of vertices $Q \subseteq W^*_{=ab}$ queried by Algorithm 1. We claim that any two different vertices $v, v' \in Q$ are adjacent. If not, there will be a shortest path from v to v' through a and b respectively, because v and v' are adjacent to a and b. This implies $v \in a/v' \cap b/v'$ and $v' \in a/v \cap b/v$. Without loss of generality v is queried before v'. Therefore, after the update in Algorithm 1 with v, A or B contains v'. That means, v' will no be queried, a contradiction. Thus, all vertices in Q are adjacent resulting in a clique. Any clique has no Radon partition, because any subset of a clique is convex, and thus $r(G) > |Q|$.

Tightness of the bounds For some example graphs that are S_i but not S_{i+1} for $i \in \{1, 2, 3\}$ and an S_4 graph, see Figure 3. For any choice of parameters (diameter, minimum hull set size, Radon number) in the bounds of Table 1 there are graphs achieving these in the respective graph family. That means that we have tight bounds for S_3 graphs (that are not S_4) and S_4 graphs. The lower bound of Theorem 10 matches asymptotically the upper bound of Theorem 6. The same holds for S_2 graphs that are not S_3, which can be seen by the graph family depicted in Figure 4. Each graph in this graph family is S_2 (and not S_3) and by adjusting the length of the path in the middle and the number of leaves on top, we can freely select the diameter and the number of extreme vertices. Again, the lower bound and the upper bound on the query complexity asymptotically coincide, as $h(G) = \operatorname{Ext}(G) + 2$ and $|W^*_{=ab}| \leq 1$ for all edges $\{a, b\}$.

Shortest path covers and generalised Dijkstra In this section we discuss how to compute the diameter and small shortest path covers using a generalised Dijkstra's algorithm.

Theorem 11. *For a given weighted graph G, we can compute a $\mathcal{O}(\log d(G))$-approximation for the minimum shortest path cover $\mathcal{S}^*$ in time $\mathcal{O}(|V|^4)$.*

Proof. We first describe a procedure to compute a shortest

Table 1: Tight bounds for specific unweighted graph families

graph family	sep. axiom	bound on $\mathrm{qc}(G)$	remark
trees	S_4	$\Theta(\log d(G) + \lvert \mathrm{Ext}(G)\rvert)$	$\mathrm{Ext}(G)$ are exactly the leaves
$K_{2,3}$ minor-free	S_4	$\Theta(\log d(G) + h(G))$	including outerplanar graphs
partial cubes	S_3	$\Theta(\log d(G) + h(G))$	$\mathcal{O}(\cdot)$ holds for bipartite graphs
weakly median	S_4	$\Theta(\log d(G) + h(G) + r(G))$	$\mathcal{O}(\cdot)$ holds for meshed graphs

path with a maximum number of vertices. More generally, we will describe an iterative procedure covering the vertices of the graph. Each round we will compute a shortest path with a maximum number of vertices that have not been covered yet.

We can assume that the graph is directed, because if it is undirected, we can transform the graph into an equivalent directed graph by duplicating each edge $\{x, y\}$ to (x, y) and (y, x) of same weight. For that, we define new weights w^* for each directed edge $e = (a, b)$ in the graph as $w^*(e) = (w(e), 0)$ if b is already covered and $(w(e), -1)$ otherwise. We note that $w(e) > 0$ for all edges e by assumption. Let us define the lexicographic order $\prec$ on the w^* tuples: $(a, b) \prec (c, d)$ if and only if $(a < b)$ or $(a = b$ and $c < d)$. Note that $\prec$ gives a total order on the w^* tuples. The w^* weight of a path is the component-wise sum of its edges w^* weight. This results in the following behavior. Two paths of the same w-weight, but with one path covering a larger number of not yet covered vertices, have different weights according to w^* and $\prec$. The path covering more vertices has a smaller w^* weight. We thus can apply the generalized Dijkstra's algorithm of Sobrinho (2002) to compute a w^*-shortest path, which gives us a w-shortest path that covers the maximum number of not yet covered vertices.

Iteratively computing such w^*-shortest paths results in a shortest path cover $\mathcal{S}$. Our problem of iteratively covering a graph using shortest path is an instance of the general *set cover problem*. Our covering approach selecting the shortest path covering the maximum number of yet uncovered vertices is an instance of the *greedy* algorithm for the general set cover problem (Chvatal, 1979). Hence, we achieve at least the same approximation ratio as this more general algorithm, which gives us $|\mathcal{S}| \leq (1 + \ln(d + 1))|\mathcal{S}^*|$. ☐

In a similar way, we can compute the diameter by applying the generalised Dijkstra of Sobrinho (2002) to each vertex in the graph again with the weights w^*.

Proposition 12. *For any graph G, it holds that $r(G) \leq \mathrm{tw}(G) + 2$.*

Proof. Consider a set $F \subseteq V(G)$ of size $r(G) - 1$ that has no Radon partition. For each pair of vertices $a, b \in F$, we can take a path in $I(a, b)$. The set of such paths for all vertex pairs do not overlap on the interior vertices, because otherwise we could construct a Radon partition using the overlapping paths. Contracting the edges on such path to one edge we get a clique of size $|F|$. The *Hadwiger number* is the maximum size of any such clique minor, which is known to be $\leq \mathrm{tw}(G) + 1$. This can be seen by the fact that any clique C has treewidth $\mathrm{tw}(C) = |C| - 1$ and that any graph minor F of a graph G has treewidth $\mathrm{tw}(F) \leq \mathrm{tw}(G)$, see for example Diestel (2017, Lemma 12.4.1). ☐

A.1. Passive learning halfspaces

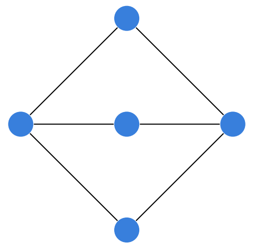
Moran and Yehudayoff (2019) established that the VC dimension of halfspaces of any convexity space is smaller than the Radon number of the space. We prove it here for completeness. Let $(X, \mathcal{C})$ be a convexity space and $r(X, \mathcal{C})$ its Radon number. We only have to show we can not shatter a set $R \subseteq X$ of size $r(X, \mathcal{C})$. By definition, R has a Radon partition R_1, R_2 with $\sigma(R_1) \cap \sigma(R_2) \neq \emptyset$. Assume there is a halfspace C separating R_1 and R_2, that is $R_1 \subseteq C$ and $R_2 \subseteq X \setminus C$. Then by the definition of halfspaces and the monotonicity of the convex hull we have that $\sigma(R_1) \subseteq \sigma(C) = C$ and $\sigma(R_2) \subseteq \sigma(X \setminus C) = X \setminus C$, contradicting that the two sets overlap.

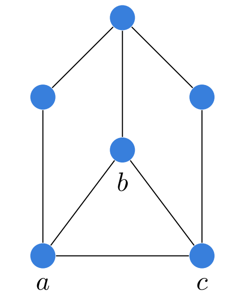
Proposition 13. *The VC dimension of the hypothesis class of halfspaces of an S_4 convexity space is exactly one less than the Radon number of the space.*

Proof. It only remains to show that we can shatter a set F of size $r(G) - 1$ using halfspaces. The convex hulls $\sigma(F_1), \sigma(F_2)$ of any partition F_1,F_2 of F do not overlap and hence they can be separated by halfspaces in an S_4 space. Thus, halfspaces shatter F. ☐

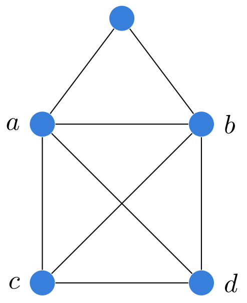
Proposition 14. *The VC dimension of halfspaces in a graph G is smaller than* $\mathrm{tw}(G) + 2$.

Proof. Combining the fact of Moran and Yehudayoff (2019) with Proposition 12 it follows that the VC dimension of halfspaces in G is $< r(G) \leq \mathrm{tw}(G) + 2$. ☐


(a) Graph is S_1 but not S_2. There are no proper halfspaces.



(b) Graph is S_2 but not S_3. Vertices a and c are not separable from vertex b.



(c) Graph is S_3 but not S_4. Vertices a and d are not separable from vertices b and c.

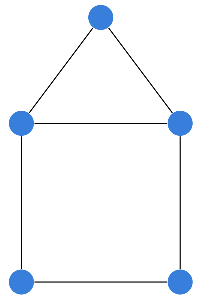

(d) Graph is S_4. Any two disjoint convex sets are separable.

Figure 3: Examples for S_1 to S_4 graphs. Inspired by Bandelt (1989).

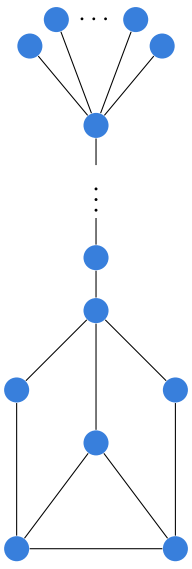


Figure 4: Example of a S_2 graph family that is not S_3 with tight bounds.

Table 2: Number of convex communities.

dataset	convex communities
DBLP	$4308/5000$
Amazon	$3999/5000$
Youtube	$2990/5000$
LiveJournal	$1649/5000$
Orkut	$363/5000$
Eu-core	$7/42$

B. Additions to the discussion

Graph and other finite convexity spaces are not equivalent to the Euclidean convexity in the sense that it is not always possible to embed a graph into $\mathbb{R}^m$ or the other way around while preserving the convex hull $\sigma(\cdot)$. As Euclidean convexity spaces are always S_4 while graphs do not have to be, there is no straightforward way to transform bounds on the query complexity from one setting to the other. But even for S_4 interval spaces, the convex hulls given by the interval mapping I_d in a finite metric space (X, d) with $X \subseteq \mathbb{R}^m$ do not always coincide with the classical convex hulls in $(\mathbb{R}^m, d)$.

To see that, take any tree graph with at least 3 leaves, which has the property that the convex hull of any two leaves contains a non-leaf vertex but no additional leaf. However, if we try to reconstruct this behaviour in $\mathbb{R}^m$ for arbitrary m, any three points, with at least one non-leaf vertex, must lie on one straight line and hence all points lie on the same straight line. But this implies that either one non-leaf vertex is not contained in the convex hull of one pair of leaves or the convex hull of two leaves contain another leaf.

For the other direction, take the triangle in the Euclidean plane with points a, b, c and a point d in its interior, which has the property that any two points are convex, and the convex hull $\sigma(\{a, b, c\})$ contains d, see Fig. 1c) in the main paper. Reconstructing this with any finite metric convexity space, including graphs, is not possible. If we assume convex edges, $I(\{a,b,c\}) = I(a,b) \cup I(a,c) \cup I(b,c) = \{a,b,c\}$, the set $\{a,b,c\}$ is convex and does not contain d.

We can phrase this in terms of halfspaces. In the Euclidean plane, a, b, c and d are not halfspace separable, however in any finite metric convexity space that has convex edges they are. In general, if two sets are linearly halfspace separable in Euclidean space, they will be separable by halfspaces in the corresponding fully connected graph with Euclidean distances.

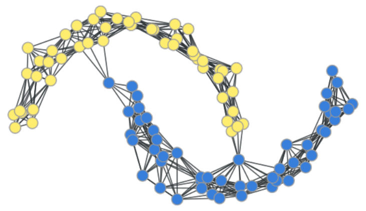
Cut-based bounds In the batch setting, the first query complexity bound was given by Guillory and Bilmes (2009), subsequently further improved (Guillory and Bilmes, 2011) and extended by the line of work of Cesa-Bianchi et al. (2010). Let $\delta(T)$ denote the set of outgoing edges of $T \subseteq V$. Guillory and Bilmes (2009) introduced $\Psi(L) = \min_{\emptyset \neq T \subseteq (V \setminus L)} |\delta(T)|/|T|$ to bound the error of the min-cut-based prediction strategy (Blum and Chawla, 2001) as $|C|/\Psi(L)$. Guillory and Bilmes (2011) thus propose to select a set L that maximises $\Psi(L)$ to make the bound smallest possible. On the hypercube with halfspace separable labels, the cut size $|C|$ is always $V/2$.

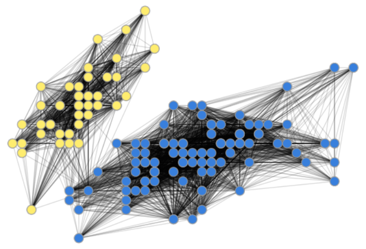
Let $G = (V, E)$ be $2 \times k$ grid vertices, where k is a multiple of four. We will show that for any L of size $|L| \leq |V|/4$, it holds that $|C|/\Psi(L) \geq |V|/2$. That means that even if the algorithm is allowed to query a quarter of the graph, it can still only guarantee that it will be correct on one half of the graph. Note that this is $2/3$ of the unqueried number of vertices.

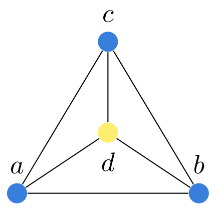
The worst-case halfspace will contain exactly one shortest path with k vertices, resulting in k cut vertices to ther other half of the graph. Thus $|C| = |V|/2$. As k is a multiple of four we can divide the grid into 2×2 blocks. Using the fact that $|L| \leq |V|/4$ we see that there must be one block $F \subseteq V \setminus L$ of four vertices that is unqueried. The number of outgoing edges of F is bounded as $|\delta(F)| \leq 4$. This implies that $\Psi(L) = \min_{\emptyset \neq T \subseteq (V \setminus L)} |\delta(T)|/|T| \leq |\delta(F)|/|F| \leq 1$. Plugging this into the error bound of Guillory and Bilmes (2009) we see that $|C|/\Psi(L) \geq |C|/1 = |V|/2$. This shows that even after $|V|/4$ arbitrarily selected queries, the bound predicts an error not better than random guessing.

Margin-based bounds We shortly discuss here, why $\operatorname{dens}(V) \geq 1$ for the $2 \times k$ grid graph. For a set $A \subseteq V$ let $\mathcal{M}(A, r)$ be the maximum cardinality of any subset $A' \subseteq A$ such that all distinct $a, b \in A'$ satisfy $d(a,b) > r$. Let $B(a, r) = \{b \in V \mid d(a,b) \leq r\}$ denote the ball of radius r around a. The density constant of a semimetric space is $\mu(V) = \min\{\mu \in \mathbb{N} \mid (v \in V \text{ and } r > 0) \Rightarrow \mathcal{M}(B(v,r), {}^{r}\!/_{2}) \leq \mu\}$. The density dimension of a semimetric space is defined as $\operatorname{dens}(V) = \log_2 \mu(V)$.

For a unweighted graph $B(v, 1)$ is simply the neighbours of v, including v. Thus, $\mathcal{M}(B(v,1), {}^{1}\!/_{2})$ is also again the set of all neighbours v, because any pair of vertices a, b has distance $d(a,b) \geq 1 > {}^{1}\!/_{2}$. This means that $\mu(V)$ is larger than the maximum degree of the graph. Concluding we have $\mu(V) \geq 2$ for any graph with at least one edge and thus $\operatorname{dens}(V) \geq 1$.


(a) Two moons NN-graph.


(b) Iris dataset NN-graph.



(c) Halfspace separable in the graph, but not in $\mathbb{R}^2$.

Figure 5: Halfspace separations on different graphs.

C. Experiments

We made all implementations and information on how to reproduce the experiments publicly available[4]. As described in the main text, we evaluated our two approaches *greedy* and *selective sampling* against S^2 (Dasarathy et al., 2015), active label propagation (Zhu et al., 2003b) and random sampling. We performed 10 independent runs on each dataset, where the first vertex query was the same for all 5 approaches. It was sampled uniformly at random from the same class in the dataset on each run, to guarantee the same starting accuracy for each appraoch.

For each dataset, the average accuracy after each query is depicted in Figure 6 including error bars representing the

[4]https://github.com/maksim96/subsetml_active_graph_halfspaces

10% and 90% quantile of the 10 runs. The same is visualised in Figure 7 with the average number of found cut vertices after each query instead of the accuracy.

We find that our greedy approach identifies the correct halfspace using 5-6 queries most of the time on all datasets. Only occasionally two more queries on the two moons and Iris dataset are reqiored. The selective sampling based approach takes some more queries, usually requiring roughly 1.5 to 2 times as many queries as the greedy approach. We thus conclude that, at least on these benchmark datasets the greedy maxisation significantly reduces the number of required queries.

Inspecting the other approaches, we find that they not only take substantially more queries to get close to the correct prediction but also that all of them are unstable. This is meant in the sense that their performance depends a lot on the first drawn vertex and the inherent randomness of the algorithms. Sometimes this even leads to a significant accuracy decrease with more queries, as can be especially be seen on the Iris dataset, see Figure 6b. Here, the accuracy of the predictions start to decrease after the 8th query for S^2 and after the 17th query for active label propagation. One reason might be that label propagation favours small cuts (weighted by Gaussian similarity) and hence has problems classifying the cut vertices correctly. Indeed, the Iris dataset has 38 cut edges while two moons only six and here all three label propagation based methods perform well. For example, S^2 identifies the correct halfspace after 14 queries.

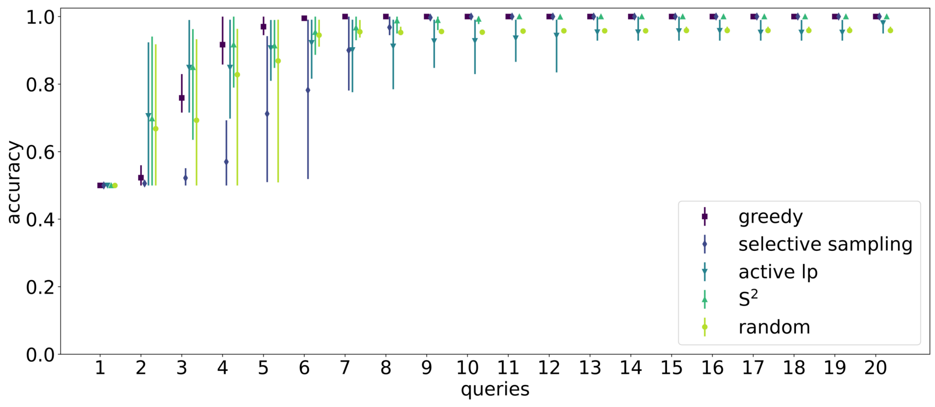
In terms of the found cut vertices per query, see Figure 7, our approaches use convex hulls and extensions to deduce many labels after each single query. The other approaches do not rely on the halfspace assumptions and hence can only identify at most one cut vertex per query. S^2 performs well still, identifying a cut vertex on almost every query after a few initial queries. This is not surprising however, as S^2 was designed with explicitly this goal in mind (Dasarathy et al., 2015). The other two approaches, active label propagation and random sampling, have difficulties finding cut vertices.

D. Extensions

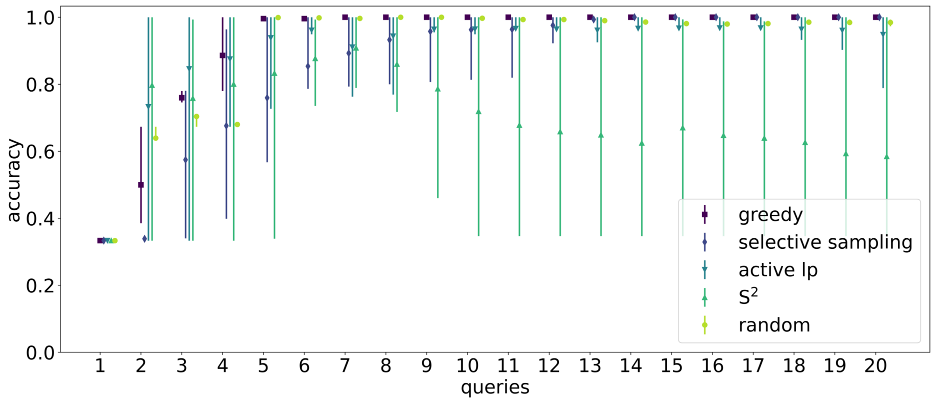
Multi-class settings Our results for halfspaces can be generalised to the case where we have k different labels and each corresponding vertex set is convex. Using shortest path covers we get the query complexity upper bound $\mathcal{O}(|\mathcal{S}^*|k \log d(G))$ in this case, as we have to binary search each path at most $k-1$ times to identify the at most $k-1$ possible cut edges. Also our main approach Lemma 4 can be adapted to find a cut edge for each pair of different labels using $h(G) + k^2 \log d(G)$ queries. However, Lemma 5 does not hold anymore in general. Algorithm 1 can be adapted to the multi-class setting but it is not clear how to generalise the $W^*_{=ab}$ based upper bound. We are convinced that similar ideas to the ones of Bressan et al. (2021) can be used to achieve an upper bound like $\mathcal{O}(h(G) + k^2(\log d(G) + \max_{\{a,b\}\in E(G)} |W^*_{=ab}|))$.

Directed graphs For directed graphs one has to adjust the definition of the interval mapping $I(x, y)$. The default way (Chartrand and Zhang, 2000; Pelayo, 2013) to fix the symmetry property, $I(x, y) = I(y, x)$, is by first defining a *directed interval* $I_D(x, y) = \{z \mid d(x, z) + d(z, y)\}$, where d is the directed shortest path distance which is not necessarily symmetric, and then simply taking the union of both directions, $I(x, y) := I_D(x, y) \cup I_D(y, x)$. The remaining definitions follow as before. Note that this implies that $I(x, y)$ can contain vertices from shortest paths of different length, as the shortest x-y-path can be shorter than its reversed counterpart.

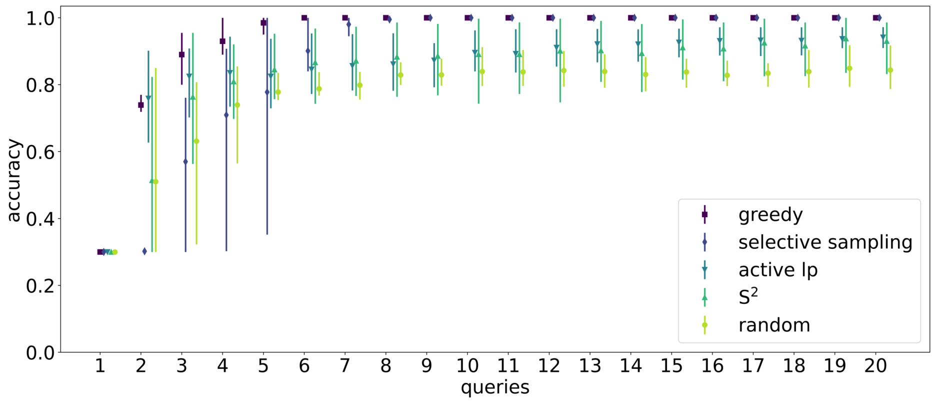
Disconnected graphs If the graph G is connected all convex vertex sets will induce a connected subgraph. Usually, for a vertex set to be convex in a disconnected graph it is additionally required that the induced subgraph is connected (Pelayo, 2013). To allow halfspace separation in disconnected graphs on multiple connected components, we ignore this restriction. The main difference to connected graphs is that in disconnected graphs the shortest path distance may become unbounded/undefined (for disconnected vertices). In such cases, we have to redefine the interval $I(x, y) = \{x, y\}$ if x and y are disconnected and keep the original definition $I(x, y) = \{z \in V \mid d(x, y) = d(x, z) + d(z, y)\}$ for connected vertices.



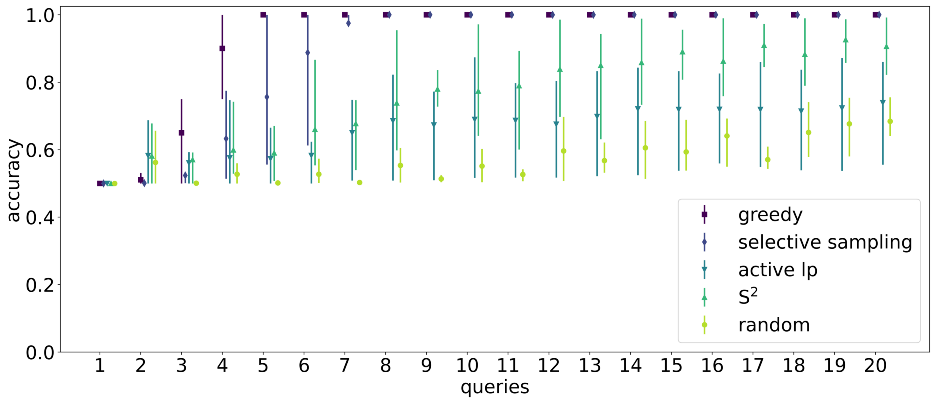
(a) Two moons.



(b) Iris.

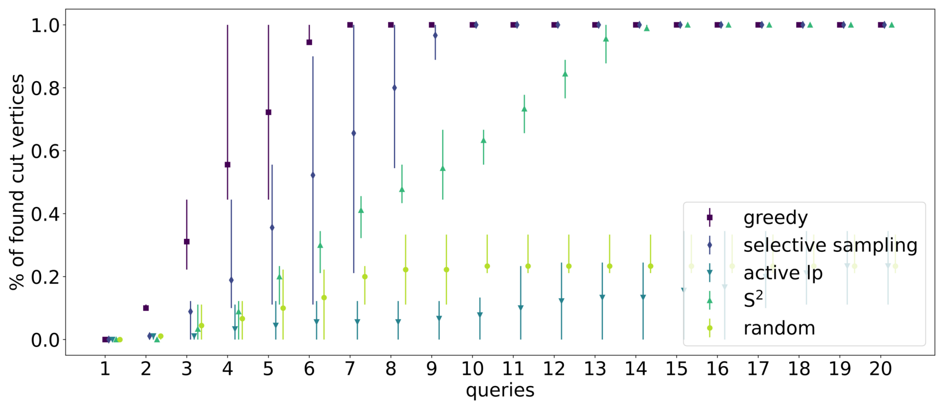


(c) 20×20 grid.

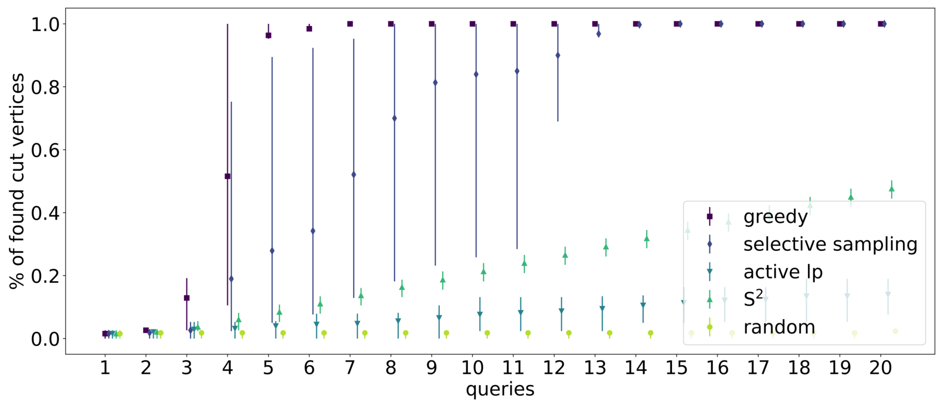


(d) 2^{10} hypercube.

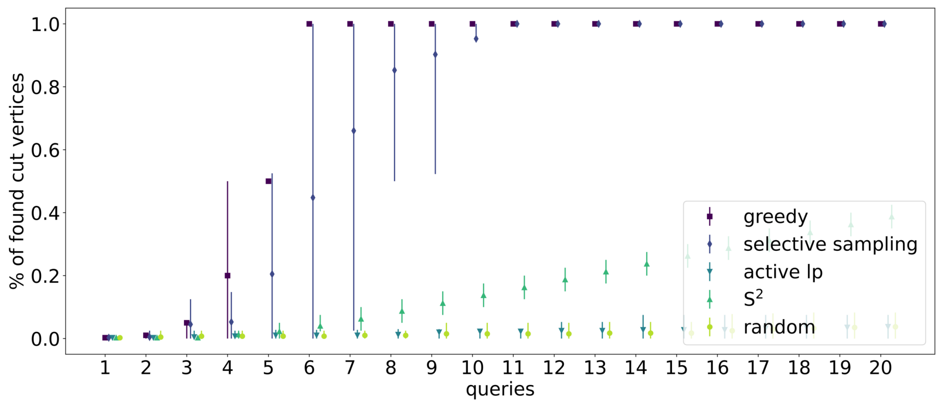
Figure 6: Accuracy against number of queries on four benchmark datasets.



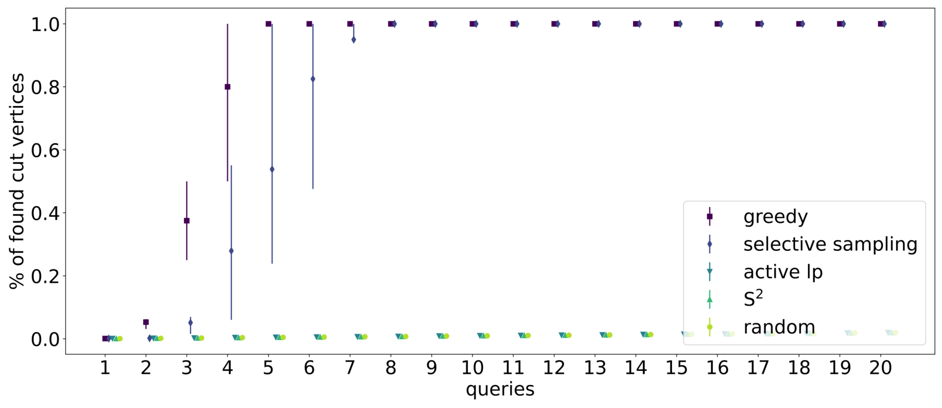
(a) Two moons.



(b) Iris.



(c) 20×20 grid.



(d) 2^{10} hypercube.

Figure 7: Number of found cut vertices against number of queries on four benchmark datasets.