Boundary-Sensitive Approach for Approximate Nearest-Neighbor Classification

Alejandro Flores-Velazco
Department of Computer Science, University of Maryland, College Park, MD, USA

David M. Mount
Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD, USA

Abstract

The problem of *nearest-neighbor classification* is a fundamental technique in machine-learning. Given a training set P of n labeled points in $\mathbb{R}^d$, and an approximation parameter $0 < \varepsilon \leq \frac{1}{2}$, any unlabeled query point should be classified with the class of any of its ε-approximate nearest-neighbors in P. Answering these queries efficiently has been the focus of extensive research, proposing techniques that are mainly tailored towards resolving the more general problem of ε-approximate nearest-neighbor search. While the latest can only hope to provide query time and space complexities dependent on n, the problem of nearest-neighbor classification accepts other parameters more suitable to its analysis. Such is the number k_ε of ε-border points, which describes the complexity of boundaries between sets of points of different classes.

This paper presents a new data structure called Chromatic AVD. This is the first approach for ε-approximate nearest-neighbor classification whose space and query time complexities are only dependent on ε, k_ε and d, while being independent on both n and Δ, the spread of P.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases approximate nearest-neighbor searching, nearest-neighbor classification, geometric data structures, space-time tradeoffs

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.44

Funding Research partially supported by NSF grant CCF-1618866.

1 Introduction

Non-parametric classification is a fundamental technique in machine-learning. In this context, we are given a *training set* P consisting of n points in a metric space $(\mathcal{X}, \mathsf{d})$, with domain $\mathcal{X}$ and distance function $\mathsf{d} : \mathcal{X}^2 \to \mathbb{R}^+$. Additionally, the training set is partitioned into a finite set of *classes* C by associating each point $p \in P$ with a *label* $l(p)$, which indicates the class to which it belongs. Given an *unlabeled* query point $q \in \mathcal{X}$, the goal of a *classifier* is to predict q's label using the training set P.

The *nearest-neighbor rule* is among the best-known classification techniques [15]. It assigns a query point the label of its closest point in P according to the defined metric. This technique exhibits good classification accuracy both experimentally and theoretically [12, 13, 34], but it is often criticized due to its high space and time complexities. Despite the advent of more sophisticated techniques (*e.g.*, support-vector machines [11] and deep neural networks [32]), nearest-neighbor classification is still widely used in practice [9, 22, 26], proving its value in constructing resilient defense strategies against adversarial [27] and poisoning [29] attacks, as well as in achieving interpretable machine-learning models [28, 33].

As mentioned, the criticism towards nearest-neighbor classification lingers on the bases of exceedingly high query times and space requirements. The standard approach to answering these queries, even approximately, involves storing the entire training set P or at least a

sufficiently large part of it. This implies that the time spent to answer such queries depends to a large degree on the size and dimensionality of the training set of points stored, limiting the use of nearest-neighbor classification on large-scale applications.

In this paper we explore efficient methods for approximate nearest-neighbor classification. We are given the training set P and an approximation parameter $0 < \varepsilon \leq \frac{1}{2}$. The objective is to construct a data structure so that given any query point q, it is possible to efficiently classify q according to any valid ε-approximate nearest-neighbor in P. Throughout, we take the domain $\mathcal{X}$ to be d-dimensional Euclidean space $\mathbb{R}^d$, the distance function d to be the L_2 norm, and we assume that the dimension d is a fixed constant, independent of n and ε.

1.1 Related Work

In the standard *ε-approximate nearest-neighbor searching* (ε-ANN), the objective is to compute a point whose distance from the query point is within a factor of $1 + \varepsilon$ of the true nearest neighbor. This problem, referred to as "standard ANN" throughout, has been widely studied. In *chromatic ε-ANN search* the objective is to return just the class (or more visually, the "color") of any such point [6, 18]. We refer to this as *ε-classification*.

Clearly, chromatic ANN queries can be reduced to standard ANN queries. Hence, most of the efficiency improvements in nearest-neighbor classification have arisen from improvements to the standard ANN problem. While standard ANN has been well studied in high-dimensional spaces (see, e.g., [1]), in constant-dimensional Euclidean space, the most efficient data structures involve variants of the *Approximate Voronoi Diagram* (or AVD) (see [3–5, 23]). Arya *et al.* [6] proposed a data structure specifically tailored for ε-classification. Unfortunately, this work was based on older technology, and its results are not competitive when compared to the most recent advances on standard ANN search via AVDs.

All previous results have query and space complexities that depend on n, the total size of the training set P. In many cases, a much smaller portion of the training set may suffice to correctly ε-classify queries. Think of the boundaries between adjacent Voronoi cells of points of different classes (see Figure 1a). The points that define these boundaries are known as *border points*. Throughout, let k denote the number of such border points in P (clearly, $k \leq n$, and hopefully, $k \ll n$). Furthermore, the notion of border points can be generalized to the context of ε-classification (see Section 2 for a formal definition). Thus, denote k_ε as the number of ε-border points, where $k \leq k_\varepsilon \leq n$. Ideally, we would like the query and space complexities of answering chromatic ε-ANN queries to depend on k_ε instead of n.

In order to achieve this goal, previous research has focused on reducing the training set P by selecting a subset $R \subseteq P$. Once R is computed, it is assumed that this subset will be used to build a standard AVD for ε-classification. Research in this area is vast, but there are two broad approaches, depending on the type and size of the computed subsets, and the classification guarantees provided.

Heuristics: Most of the work has focused on proposing heuristics to compute smaller training sets $R \subseteq P$. These are often known as *condensation algorithms*, and the literature on these is extensive (see [25, 35] for comprehensive surveys, and [2, 7, 8, 21, 24, 30] for some of the proposed algorithms). The most recent condensation algorithms [16, 17, 20] show that it is possible to compute subsets of P of size $\mathcal{O}(k)$ in $\mathcal{O}(n^2)$ time. However, when AVDs are built from these subsets, the resulting data structures are likely to introduce classification errors [19], especially for query points that should be easily ε-classified. Thus, while often used in practice, these approaches do not guarantee that chromatic ε-ANN queries are answered correctly.

Coresets: Recent results propose a technique to compute a coreset for ε-classification [19]. A coreset R guarantees that every query point will be correctly classified when assigning the class of the point of R returned by the AVD. That is, for any query $q \in \mathbb{R}^d$, the point of R returned by the AVD belongs to the same class as one of q's ε-approximate nearest-neighbors in P. Unfortunately, the size of the computed coreset can be as large as $\mathcal{O}((k \log \Delta)/\varepsilon^{d-1})$, where Δ is the spread[1] of P.

1.2 Contributions

From the previous section, we have seen that existing approaches for ε-classification achieve only one of the following goals:

- The size of the resulting data structure is dependent only on ε, k_ε (the number of ε-border points) and d, while being independent from n and Δ.
- It guarantees correct ε-classification for any query point.

The main result of this paper is an approach that achieves both goals. We propose a new data structure built specifically to answer chromatic ε-ANN queries over the training set P, which we call a *Chromatic AVD*. Given any query point $q \in \mathbb{R}^d$, this data structure returns the class to be assigned to q, which matches the class of at least one of q's ε-approximate nearest-neighbors in P. More generally, our data structure returns a set of classes such that there is an ε-approximate nearest-neighbor of q from each of these classes.

Therefore, the Chromatic AVD can be used to correctly ε-classify any query point. The main result of this work is summarized in the following theorem, expressed in the form of a space-time tradeoff based on a parameter γ.

▶ **Theorem 1.** *Given a training set P of n labeled points in $\mathbb{R}^d$, an error parameter $0 < \varepsilon \leq \frac{1}{2}$, and a separation parameter $2 \leq \gamma \leq \frac{1}{\varepsilon}$. Let k_ε be the number of ε-border points of P. There exists a data structure for ε-classification, called Chromatic AVD, with:*

$$\textit{Query time: } \mathcal{O}\left(\log\left(k_\varepsilon \gamma\right) + \frac{1}{(\varepsilon\gamma)^{\frac{d-1}{2}}}\right) \qquad \textit{Space: } \mathcal{O}\left(k_\varepsilon \gamma^d \log \frac{1}{\varepsilon}\right).$$

Which can be constructed in time $\widetilde{\mathcal{O}}\left(\left(n + k_\varepsilon/(\varepsilon\gamma)^{\frac{3}{2}(d-1)}\right)\gamma^d \log \frac{1}{\varepsilon}\right)$.

By setting γ to either of its extreme values, we obtain the following query times and space complexities.

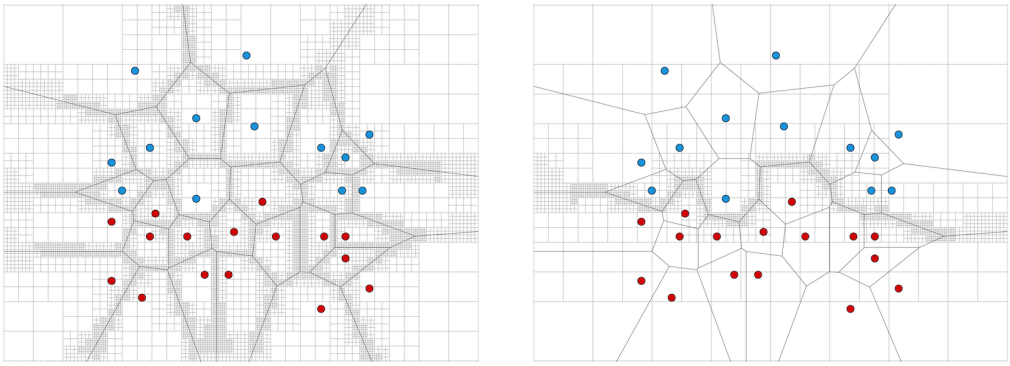
▶ **Corollary 2.** *The separation parameter γ describes the tradeoffs between the query time and space complexity of the Chromatic AVD. This yields the following results:*

$$\textit{If } \gamma = 2 \quad \longrightarrow \quad \textit{Query time: } \mathcal{O}\left(\log k_\varepsilon + \frac{1}{\varepsilon^{\frac{d-1}{2}}}\right) \qquad \textit{Space: } \mathcal{O}\left(k_\varepsilon \log \frac{1}{\varepsilon}\right).$$

$$\textit{If } \gamma = \frac{1}{\varepsilon} \quad \longrightarrow \quad \textit{Query time: } \mathcal{O}\left(\log \frac{k_\varepsilon}{\varepsilon}\right) \qquad \textit{Space: } \mathcal{O}\left(\frac{k_\varepsilon}{\varepsilon^d}\right).$$

The approach towards constructing this data structure is hybrid, combining a quadtree-induced partitioning of space (leveraging similar techniques to the ones used for standard AVDs), with the construction of coresets for only some cells of this partition. All other cells can be discarded, and a new quadtree can be built with only the remaining cells. The final size of the tree is bounded in terms of k_ε. This technique allows us to maintain coresets in the most critical regions of space, and thus, avoiding the dependency on the spread of P.

[1] The *spread* of a point set is defined to be the ratio between the largest and smallest pairwise distances.



(a) Standard AVD [3–5, 23]. **(b)** Chromatic AVD.

Figure 1 Examples of the space partitioning achieved by any standard AVD, compared to the Chromatic AVD data structure proposed in this paper. Our approach subdivides the space around the boundaries defined by the ε-border points, while ignoring other boundaries.

2 Preliminary Ideas and Intuition

Preliminaries. First, we need to introduce some preliminary definitions and notations that are relevant to the results presented in the remaining of the paper. Given any point $q \in \mathbb{R}^d$, denote its nearest-neighbor as $\text{nn}(q)$, and the distance between them by $\mathsf{d}_{\text{nn}}(q) = \mathsf{d}(q, \text{nn}(q))$.

Additionally, let's introduce a few concepts and related properties that will prove useful in the construction of the Chromatic AVD. These are Well-Separated Pair Decompositions [10] (WSDPs), Quadtrees [14, 31], and Approximate Voronoi Diagrams [3–5, 23] (AVDs).

Well-Separated Pair Decompositions: Given the point set P, and a separation factor $\sigma > 2$, we say that two sets $X, Y \subseteq P$ are *well separated* if they can be enclosed within two disjoint balls of radius r, such that the distance between the centers of these balls is at least σr. We say that X and Y form a *dumbbell*, where both sets are the *heads* of this dumbbell. Consider the line segment that connects the centers of both balls, and let z and ℓ be the center and length of this line segment, respectively (*i.e.*, the center and the length of the dumbbell). The following properties hold when $\sigma > 4$, for $x \in X$ and $y \in Y$:

$$\mathsf{d}(x, z) < \ell \qquad \ell < 2\mathsf{d}(x, y) \qquad \ell > \mathsf{d}(x, y)/2.$$

Furthermore, a well-separated pair decomposition of P is defined as a set $\mathcal{D} = \{(X_i, Y_i)\}_i$ where every X_i and Y_i are well separated, and for every two distinct points $p_1, p_2 \in P$ there exists a unique pair $\mathcal{P} = (X, Y) \in \mathcal{D}$ such that $p_1 \in X$ and $p_2 \in Y$, or vice-versa. It is known how to construct a WSPD of P with $\mathcal{O}(\sigma^d n)$ pairs in $\mathcal{O}(n \log n + \sigma^d n)$ time.

Quadtrees: These are tree data structures that provide a hierarchical partition of space. Each node in this tree consists of a d-dimensional hypercube, where non-leaf nodes partition its corresponding hypercube into 2^d equal parts. The root of this tree corresponds to the $[0, 1]^d$ hypercube. We will use a variant of this structure called a *balanced box-decomposition* tree (BBD tree) [6]. Such data structure satisfies the following properties:

1. Given a point set P, such a tree can be built in $\mathcal{O}(n \log n)$ time, having space $\mathcal{O}(n)$ such that each leaf node contains at most one point of P.
2. Given a collection $\mathcal{U}$ of n quadtree boxes in $[0, 1]^d$, such a tree can be built in $\mathcal{O}(n \log n)$ time, having $\mathcal{O}(n)$ nodes such that the subdivision induced by its leaf cells is a refinement of the subdivision induced by the Quadtree boxes in $\mathcal{U}$.
3. Given the trees from **1** or **2**, it is possible to determine the leaf cell containing any arbitrary query point q in $\mathcal{O}(\log n)$ time.

Approximate Voronoi Diagram: Generally, AVDs are quadtree-based data structures that can be used to efficiently answer ANN queries. The partitioning of space induced by this data structure is often generated from a WSPD of P. Additionally, every leaf cell w of this quadtree has an associated set of ε-representatives R_w that has the following property: for any query point $q \in w$, at least one point in R_w is one of q's ε-approximate nearest-neighbors in P.

New Ideas and Intuitions. Consider the space partitioning induced by a standard AVD, as previously described. By construction, any leaf cell w of this partition has an associated set of ε-representatives R_w. Evidently, for the purposes of ε-classification, the most important information related to this leaf cell comes from the classes of the points in R_w, and not necessarily the points themselves.

This leads to an initial approach to simplify an AVD. We distinguish between two types of leaf cells, based on the points inside their corresponding ε-representative sets. Any leaf cell w is said to be:

- **Resolved**: If every point in R_w belongs to the same class.
- **Ambiguous**: Otherwise, if at least two points in R_w belong to different classes.

Clearly, there is no need to store the set of ε-representatives of any resolved leaf cell, as instead, we can simply mark the leaf cell w as *resolved with the class* that is shared by all the points in R_w. This effectively reduces the space needed for such cells to be constant.

Furthermore, it seems that the bulk of the "work" needed to decide the class of a given query point can be carried out by the ambiguous leaf cells, along with some groupings of resolved leaf cells. The data structure presented in this paper, called Chromatic AVD, builds upon this hypothesis.

Additionally, we formally define the set of ε-border points of the training set P. This set, denoted as K_ε, contains any point $p \in P$ for which there exist some $q \in \mathbb{R}^d$ and $\bar{p} \in P$, such that p and $\bar{p}$ are ε-approximate nearest-neighbors of q, and both belong to different classes. Denote $k_\varepsilon = |K_\varepsilon|$ as the number of ε-border points of the training set P. Note that $K_\varepsilon \subseteq K_{\varepsilon'}$ if and only if $\varepsilon \leq \varepsilon'$. Additionally, note that K_0 defines the set of (exact) border points of P, where $k = k_0$.

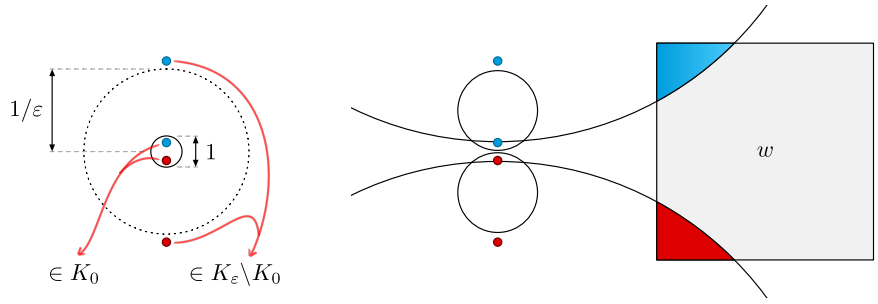


Figure 2 Intuition to think that K_ε (and not K_0) is needed to ε-classify some query points.

This generalization of the definition of border points seems better suited to analyze the problem of ε-classification, as illustrated in Figure 2. Figure 2b shows the ε-approximate bisectors between the two closest and two farthest points (the first two belong to K_0, while the others belong to K_ε but not K_0). A hypothetical leaf cell w is sufficiently separated from the only two exact border points, but intersects the ε-approximate bisectors between the

two farthest points. This implies that inside the cell w lie query points that *can only* be ε-classified with one class, and others with the other class, forcing this cell to be ambiguous. This suggests that K_0 is insufficient to account for the necessary complexity of ε-classification.

3 Chromatic AVD Construction

In this section, we describe our method for constructing the proposed Chromatic AVD. The following overview outlines the necessary steps followed to construct this data structure.

- *The Build step* (Section 3.1): Consists of building an initial quadtree-based subdivision of space, designed specifically to achieve the properties described in Lemma 3.
- *The Reduce step* (Section 3.2): Seeks to identify the leaf cells of the initial subdivision that are relevant for ε-classification, as well as those that can be ignored or simplified. This process consists of the following substeps.
  - Computing the sets of ε-representatives for every leaf cell of the initial quadtree.
  - Based on these sets, marking the leaf cells as either ambiguous or resolved.
  - Selecting those leaf cells which are relevant for ε-classification.
  - Building a new quadtree-based subdivision using the previously selected leaf cells.

3.1 The Build Step

We begin by constructing the tree T_{init} using similar methods as the ones used to construct a standard AVD. Thus, the first step is to compute a well-separated pair decomposition $\mathcal{D}$ of P using a constant separation factor of $\sigma > 4$. While the standard construction would use all pairs in this decomposition, for the purpose of the Chromatic AVD, we filter $\mathcal{D}$ to only keep bichromatic pairs. Denote $\mathcal{D}' \subseteq \mathcal{D}$ to be the set of bichromatic pairs in $\mathcal{D}$, where a pair $\mathcal{P} \in \mathcal{D}$ is said to be bichromatic if and only if the dumbbell heads separate points of different classes. Note that $\mathcal{D}'$ can be computed similarly to $\mathcal{D}$, using a simple modification of the well-known algorithm for computing WSPDs [10] (the details are left to the reader).

Next, we compute an initial set of quadtree boxes $\mathcal{U}(\mathcal{P})$ for every pair in $\mathcal{D}'$ as follows. This construction depends on two constants c_1 and c_2 whose assignment will be described later in this section. For $0 \leq i \leq \lceil \log(c_1\, 1/\varepsilon) \rceil$, we define $b_i(\mathcal{P})$ as the ball centered at z of radius $r_i = 2^i \ell$. Thus, this set of balls involves radius values ranging from ℓ to $\Theta(\ell/\varepsilon)$. For each such ball $b_i(\mathcal{P})$, let $\mathcal{U}_i(\mathcal{P})$ be the set of quadtree boxes of size $r_i/(c_2\gamma)$ that overlap the ball. Let $\mathcal{U}(\mathcal{P})$ denote the union of all these boxes over all the $\mathcal{O}(\log 1/\varepsilon)$ values of i.

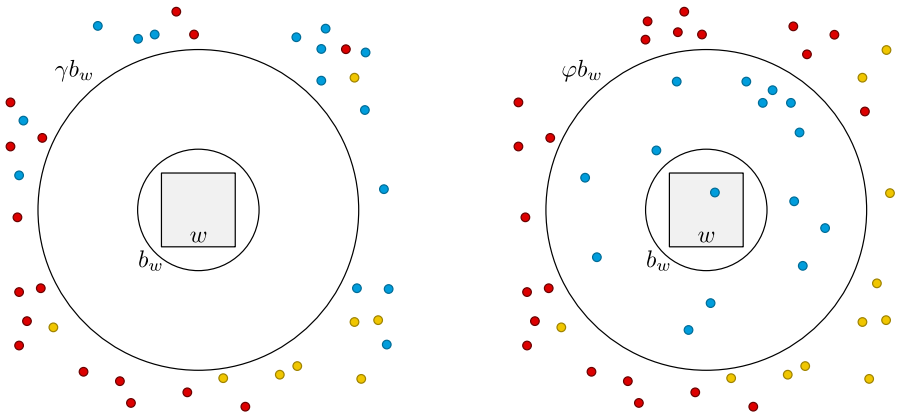
After performing this process on every pair of the filtered decomposition $\mathcal{D}'$, take the union of all these boxes denoted as $\mathcal{U} = \bigcup_{\mathcal{P} \in \mathcal{D}'} \mathcal{U}(\mathcal{P})$. Finally, build the tree T_{init} from the set of quadtree boxes $\mathcal{U}$, leveraging property 2 of quadtrees described in Section 2. Additionally, for each class i in the training set, build an auxiliary tree T^i_{aux} from the point set P_i (*i.e.*, the points of P of that are labeled with class i), using property 1 of quadtrees. These auxiliary trees will be used together with T_{init} in order to build our final tree T, the Chromatic AVD.

While the standard AVD construction satisfies that all resulting leaf cells of the tree have certain separation properties from the points of set P, the same is not true for tree T_{init}. However, the following result describes a relaxed notion of the separation properties, now based on the classes of the points, which are achieved by T_{init}.

▶ **Lemma 3** (Chromatic Separation Properties). *Given two separation parameters $\gamma > 2$ and $\varphi > 3$, every leaf cell w of the tree T_{init} satisfies at least one of the following separation properties, where b_w is the minimum enclosing ball of w:*

(i) *$P \cap \gamma b_w$ is empty (see Figure 3a), and hence b_w is concentrically γ-separated from P.*

(ii) *The cell w can be resolved with the classes present inside $P \cap \varphi b_w$ (see Figure 3b).*



(a) No points inside γb_w. **(b)** Leaf cell w can be resolved.

Figure 3 Basic separation properties achieved by during the construction of the Chromatic AVD.

Proof. Let w be any leaf cell of T_{init}, with center c_w and side length s_w, where its (minimum) enclosing ball b_w has radius $r_w = \sqrt{d}/2\, s_w$ and shares the center c_w. Additionally, let $x_i \in P_i$ be a 1-approximate nearest-neighbor of c_w among the points of P of class i. In other words, for each class of points we use the auxiliary trees T^i_{aux} to compute a 1-approximate nearest-neighbor of c_w. A few cases unfold from here:

The first case is rather simple. If $4\gamma\varphi b_w \cap \{x_i\}_i = \varnothing$, knowing that the points x_i are 1-approximate nearest-neighbors of c_w, this implies that the ball $2\gamma\varphi b_w$ is empty (*i.e.*, we know that $2\gamma\varphi b_w \cap P = \varnothing$). Clearly, this means the the first separation property holds for w.

Consider the case when $|4\gamma\varphi b_w \cap \{x_i\}_i| = 1$, and let i be the class of the point that lies inside $4\gamma\varphi b_w$. Following similar arguments to the previous case, this implies that only points of class i could potentially lie inside of $2\gamma\varphi b_w$. Then, check if x_i lies inside the smaller ball expansion $2\gamma b_w$. If not, we know that γb_w is empty (*i.e.*, $\gamma b_w \cap P = \varnothing$), making the first separation property hold for w. Otherwise, we know that $2\gamma b_w$ contains at least one point (*i.e.*, x_i), and additionally we know that $2\gamma b_w$ is φ-separated from points of all other classes but i (as $2\gamma\varphi b_w$ only contains points of class i). Given that $\varphi > 3$, the nearest neighbor of every query point inside $2\gamma b_w$ has class i. Therefore, w can be resolved with class i (namely, $C_w = \{i\}$), satisfying the second separation property.

Lastly, it is possible that $|4\gamma\varphi b_w \cap \{x_i\}_i| \geq 2$. However, it is possible to show that if this is the case, it immediately implies that every point inside $4\gamma\varphi b_w$ actually lies inside of some ball b'_w which is β-separated from w (see Figure 5a), where $\beta = 1/\varepsilon$. The details of this part of the proof are omitted, and left in the Appendix A, as the arguments are similar to the ones described in [5]. However, proving this provides insights into how to set constants c_1 and c_2, where $c_1 \geq 3(1+\varepsilon)$ and $c_2 \geq 20\varphi\sqrt{d}$.

Finally, if $|4\gamma\varphi b_w \cap \{x_i\}_i| \geq 2$, we know all points inside $4\gamma\varphi b_w$ are β-separated from w. We can now proceed similarly to the previous case, by checking if one of the computed 1-approximate nearest-neighbors lies inside the ball $2\gamma b_w$. If $2\gamma b_w \cap \{x_i\}_i = \varnothing$ we know that γb_w is empty (*i.e.*, $\gamma b_w \cap P = \varnothing$), making the first separation property hold for w. Otherwise, note that b'_w is completely contained inside $2\gamma(1+\varepsilon)b_w$. Given that $\varphi > 3$, it is possible to show that for any query point in w, all points in b'_w are valid ε-approximate nearest neighbors. This implies that we can resolve w with the class of any of the points inside of b'_w, thus satisfying the second separation property. In particular, we mark w as resolved with every class present in the inner cluster b'_w, namely, $C_w = \{l(p) \mid \forall\, p \in b'_w \cap P\} = \{i \mid x_i \in 4\gamma\varphi b_w\}$. ◀

3.2 The Reduce Step

From the initial partitioning as described in Lemma 3, every leaf cell w of T_{init} is either concentrically γ-separated from P (*i.e.*, $\gamma b_w \cap P = \varnothing$), or it is already marked as resolved. For every leaf cell w in the first case, we will compute a set of ε-representatives by leveraging the *concentric ball lemma* (see Lemma 5.1 in [5]). It states that there exists a set R_w of ε-representatives for w of size $\mathcal{O}(1/(\varepsilon\gamma)^{\frac{d-1}{2}})$, and provides a way to compute such set.

Instead of directly applying this result, we use it to compute a set of $\varepsilon/3$-representatives for any leaf cell w that is yet unresolved. Essentially, this leads to the same asymptotic upper-bound on the size of R_w, meaning that $|R_w| = \mathcal{O}(1/(\varepsilon\gamma)^{\frac{d-1}{2}})$. Once R_w is computed, we can proceed to mark w as either resolved or ambiguous as follows.

Procedure to Mark Leaf Cells. For every leaf cell w, this procedure marks w as either resolved or ambiguous, following a few defined cases that unfold from the contents of the set R_w of points selected as representatives for w. Let $r_w^- = \varepsilon\,(1-\gamma)\,r_w/3$.

1. If all the points in R_w belong to the same class.
   For every point $p \in R_w$ and class $i \in C$, compute a 1-approximate nearest-neighbor of p among the points of P_i, denoted as the point $x_{p,i}$. If $\mathsf{d}(p, x_{p,i}) < r_w^-$, then add $x_{p,i}$ to R_w. It is easy to show that $x_{p,i}$ would also be an ε-representative for w. Repeat this for every point originally in R_w, and every class in the training set.
   a. If any point $x_{p,i}$ was added to R_w, proceed with *Case* 2.
   b. Otherwise, mark w as resolved with the class of the points in R_w. Namely, let i be the class of every point in R_w, then $C_w = \{i\}$.
2. If R_w contains points of more than one class.
   Before proceeding, we will do some basic pruning of the set R_w. For every class i, compute a *net* among the points of R_w of class i, using a radius of r_w^- to compute the net, and replace the points of class i in R_w with the computed net. It is easy to see that the remaining points of R_w are a set of ε-representatives of w, and that every two points in R_w of the same class are at distance at least r_w^-.
   a. If the diameter of R_w is less than r_w^-, it is easy to prove that all the points in R_w are ε-representatives of any point inside b_w. Therefore, w can be labeled as resolved with the class of all of the points in R_w. That is, $C_w = \{l(p) \mid \forall\ p \in R_w\}$.
   b. If the diameter is greater than or equal to r_w^-, w is marked as ambiguous.

Let $\mathcal{A}$ and $\mathcal{R}$ be the sets of ambiguous and resolved leaf cells of T_{init}, respectively. We will use some of these cells to build the Chromatic AVD, while ignoring the remaining cells.

Consider the set of resolved leaf cells $\mathcal{R}$, we partition this set into two subsets $\mathcal{R}_b$ and $\mathcal{R}_i$ (named *boundary* and *interior* resolved leaf cells, respectively). We say a resolved leaf cell w_1 belongs to $\mathcal{R}_b$, if and only if there exists another resolved leaf cell w_2 adjacent to w_1, such that $C_{w_2} \setminus C_{w_1} \neq \varnothing$. Every other resolved leaf cell belongs to $\mathcal{R}_i$ (*i.e.*, $\mathcal{R}_i = \mathcal{R} \setminus \mathcal{R}_b$). Note that both sets $\mathcal{R}_b$ and $\mathcal{R}_i$ can be identified by a simple traversal over the leaf cells of T_{init}, using linear time in the size of the tree[2].

Finally, we build a new tree T from the set of ambiguous and boundary resolved leaf cells $\mathcal{A} \cup \mathcal{R}_b$. By well-known construction methods of quadtrees, as described in Section 2, the leaf cells of T either belong to $\mathcal{A} \cup \mathcal{R}_b$, or are "Steiner" leaf cells added during the construction of T that cover the remainer of the space that is uncovered by $\mathcal{A} \cup \mathcal{R}_b$.

[2] Two leaf cells are adjacent if and only if a vertex of one of the cells "touches" the other cell. This implies that the number of adjacency relations (*i.e.*, edges in the implicit graph where the leaf cells are the nodes) is $\mathcal{O}(2^d\, m)$, where m is the number of leaf cells of the tree T_{init}.

▶ **Lemma 4.** *For any leaf cell w in the tree T such that $w \notin \mathcal{A} \cup \mathcal{R}_b$, w must cover a space that is also covered by a collection of leaf cells of T_{init}, all of which are resolved with the same set of classes C_w.*

Proof. This becomes apparent from the construction of T. In the new tree T, consider any leaf cell w of T that is not part of $\mathcal{A} \cup \mathcal{R}_b$ (*i.e.*, a "Steiner" leaf cell added during the construction of the tree). Now, recall that the leaf cells of both T and T_{init} are a partitioning of (the same) space, which means that we can define $\mathcal{W}_w = \{w' \in T_{\text{init}} \mid w \cap w' \neq \varnothing\}$ as the collection of leaf cells of T_{init} that cover the same space covered by w.

Now, for any fixed w of T, it is easy to see that any two leaf cells $w_1, w_2 \in \mathcal{W}_w$ must be resolved with the same set of classes. Otherwise, at least one of these two would be part of the set $\mathcal{R}_b$, which would be a contradiction to the fact that w is a "Steiner" leaf cell of T. Therefore, any query inside w can be answered with the classes $C_w = C_{w_1} = C_{w_2}$, and this can be determined during preprocessing by a single query on T_{init} (*e.g.*, finding the leaf cell of T_{init} that contains the center c_w of w is sufficient to know the contents of C_w). ◀

This implies that after building tree T, and with some extra preprocessing to resolve the "Steiner" leaf cells of the tree, we can use the resulting data structure to correctly answer chromatic ε-approximate nearest-neighbor queries over the training set P. In other words, T can be used to answer ε-classification queries over P. We call this data structure T the Chromatic AVD.

▶ **Lemma 5.** *The construction of T takes $\widetilde{\mathcal{O}}\left(n\gamma^d \log \frac{1}{\varepsilon}\right)$ time.*

Proof. Let's analyze the total time needed to build our Chromatic AVD, namely the tree T, by analyzing the time required to perform each step of the construction.

- Building T_{init} has essentially the same complexity of building any standard AVD [3–5, 23]. This means that constructing T_{init} takes $\mathcal{O}(m \log m)$ time, where $m = n\gamma^d \log \frac{1}{\varepsilon}$. Note that during the construction, while computing the set of ε-representatives of each leaf cell, each leaf cell can already be marked as either ambiguous or resolved.
- Building the auxiliary trees T^i_{aux} for every class i, takes $\mathcal{O}(n \log n)$ time, as the number of classes of P is considered to be a constant. Recall that because these trees are only used to for 1-ANN queries, they only need to be standard Quadtrees, and not AVDs.
- Identifying the set $\mathcal{R}_b$ requires a traversal over the leaf-level partitioning of the space, which is linear in terms of the number of cells. Therefore, this step requires $\mathcal{O}(m)$ time.
- Once the sets of ambiguous and boundary resolved leaf cells are identified, namely, the sets $\mathcal{A}$ and $\mathcal{R}_b$, the final tree T can be built. Roughly, this step takes $\mathcal{O}(m \log m)$ time.
- Finally, we must resolve the "Steiner" leaf cells of T, which can be done by a single query over T_{init}, each taking $\mathcal{O}(\log m)$ time. Thus, this step takes $\mathcal{O}(m \log m)$ total time.

All together, the total construction time is dominated by the first step. Therefore, the time required to construct T is $\mathcal{O}(m \log m) = \widetilde{\mathcal{O}}(m) = \widetilde{\mathcal{O}}\left(n\gamma^d \log \frac{1}{\varepsilon}\right)$. ◀

4 Tree-size Analysis

4.1 Initial Size Bounds

Define the set of important leaf cells $\mathcal{I}$ of the tree T_{init} as those leaf cells w for which there exists two ε-border points inside $\rho\gamma b_w$ for some constant ρ, such that the distance between these points is lower-bounded by $\Omega(\varepsilon\gamma r_w)$. Formally, we define this set as $\mathcal{I} = \{w \in T_{\text{init}} \mid \exists p_1, p_2 \in \rho\gamma b_w \cap K_\varepsilon, \mathsf{d}(p_1, p_2) = \Omega(\varepsilon\gamma r_w)\}$.

▶ **Lemma 6.** *The number of important leaf cells of T_{init} is $|\mathcal{I}| = \mathcal{O}\big(k_\varepsilon \gamma^d \log \frac{1}{\varepsilon}\big)$.*

Proof. This proof follows from a charging argument on the set K_ε of ε-border points of P. More specifically, consider a well-separated pair decomposition $\mathcal{D}''$ of K_ε with constant separation factor of $\sigma > 4$, the charging scheme assigns every important leaf cell $w \in \mathcal{I}$ to a pair of $\mathcal{D}''$. Recall that $\mathcal{D}''$ can generally be consider to have $\mathcal{O}(k_\varepsilon)$ pairs, where $k_\varepsilon = |K_\varepsilon|$. It is important to note that both K_ε and $\mathcal{D}''$ need not be computed.

Given that $w \in \mathcal{I}$, we know there exist two points $p_1, p_2 \in \rho\gamma b_w \cap K_\varepsilon$. Let $\mathcal{P} \in \mathcal{D}''$ be the pair of $\mathcal{D}''$ that contains both p_1 and p_2, each in one of its dumbbell heads. We then charge w to the pair $\mathcal{P}$. Denote z and ℓ to be the center and length of $\mathcal{P}$, respectively, we know the following. First, note that the distance from c_w (the center of w) to z is $\mathsf{d}(c_w, z) \leq \rho\gamma r_w + \ell$. Additionally, we know that $\ell \geq \mathsf{d}(p_1, p_2)/2 = \Omega(\varepsilon\gamma r_w)$ by the properties of WSPDs described in Section 2. Therefore, this implies that the ratio $\mathsf{d}(c_w, z)/\ell = \mathcal{O}(1/\varepsilon)$.

Finally, fix some pair $\mathcal{P} \in \mathcal{D}''$ with center z and length ℓ, and consider all important leaf cells according to their distance to z. For any value of $i \in [0, 1, \ldots, \mathcal{O}(\log 1/\varepsilon)]$, consider all leaf cells that can charge $\mathcal{P}$ whose distance to z is between $2^i\ell$ and $2^{i+1}\ell$. From our previous analysis, $r_w \geq \mathsf{d}(c_w, z)/\rho\gamma \geq 2^i\ell/\rho\gamma$. By a simple packing argument, the number of such leaf cells is at most $\mathcal{O}(\gamma^d)$. Thus, a total of $\mathcal{O}(\gamma^d \log 1/\varepsilon)$ cells can charge $\mathcal{P}$. Note that no leaf cell whose distance to z is $\Omega(\ell/\varepsilon)$ can charge $\mathcal{P}$, as it would contradict the fact that both p_1 and p_2 are separated by a distance of $\Omega(\varepsilon\gamma r_w)$. Finally, the proof follows by knowing that there are at most $\mathcal{O}(k_\varepsilon)$ pairs in $\mathcal{D}''$. ◀

▶ **Lemma 7.** *Every ambiguous leaf cell of T_{init} is important, namely $\mathcal{A} \subseteq \mathcal{I}$.*

Proof. Consider any ambiguous leaf cell $w \in \mathcal{A}$ of the tree T_{init}. Knowing that w is ambiguous implies that there must exist some point $q \in \frac{\gamma}{2} b_w$ for which two of the ε-representatives of w are valid ε-approximate nearest neighbors for q, both points belong to different classes, and the distance between them is $\Omega(\varepsilon\gamma r_w)$. Formally, denote these points as $p_1, p_2 \in P$ such that $l(p_1) \neq l(p_2)$, $\mathsf{d}(p_1, p_2) \geq \varepsilon\,(1-\gamma)\, r_w/4$, and $\mathsf{d}(q, p_1), \mathsf{d}(q, p_2) \leq (1+\varepsilon)\, \mathsf{d}_{\text{nn}}(q)$.

We will see now how to bound the distance from c_w to any of these points as a constant factor of r_w (recall that $r_w = \sqrt{d}/2\, s_w$). From the proof of Lemma 6.3 in [5], we know that the ball $c_3\gamma b_w \cap P \neq \varnothing$, for some constant $c_3 \geq 1 + 2c_2/\sqrt{d}$. In other words, $\mathsf{d}_{\text{nn}}(c_w) \leq c_3\gamma\, r_w$. From this, we can say that $\mathsf{d}_{\text{nn}}(q) \leq (\frac{1}{2} + c_3)\gamma r_w$. Applying the triangle inequality yields that $\mathsf{d}(c_w, p_1) \leq \mathsf{d}(c_w, q) + \mathsf{d}(q, p_1) \leq \big(\frac{1}{2} + (1+\varepsilon)(\frac{1}{2} + c_3)\big)\, \gamma\, r_w$. Similarly, we can achieve the same bound for $\mathsf{d}(c_w, p_2)$.

Therefore, both $p_1, p_2 \in \rho\gamma b_w$ for sufficiently large constant ρ (*i.e.*, $\rho \geq \varepsilon(\frac{1}{2} + c_3) + c_3 + 1$). Knowing also that $\mathsf{d}(p_1, p_2) = \Omega(r_w^-) = \Omega(\varepsilon\gamma r_w)$ yields that the leaf cell $w \in \mathcal{I}$. ◀

▶ **Lemma 8.** *Every boundary resolved leaf cell of T_{init} is important, namely $\mathcal{R}_b \subseteq \mathcal{I}$.*

Proof. Let $w_1 \in \mathcal{R}_b$ be any boundary resolved leaf cell of the tree T_{init}, we know there exists another leaf cell $w_2 \in \mathcal{R}_b$ adjacent to w_1, such that there exists some class $i \in C_{w_2} \backslash C_{w_1}$. Let b_{w_1} and b_{w_2} be the corresponding bounding balls of w_1 and w_2. By definition, any point q on the boundary shared by w_1 and w_2 has at least one ε-representative from each cell, namely some points $p_1 \in R_{w_1}$ and $p_2 \in R_{w_2}$, where $l(p_1) \neq i$ and $l(p_2) = i$. Additionally, by similar arguments to the ones described in Lemma 7, we know that both $p_1, p_2 \in \rho\gamma b_w$ for sufficiently large constant ρ.

Now, we proceed to prove that $\mathsf{d}(p_1, p_2) \geq r_w^-/2$. First, note that if w_1 was resolved by the initial marking of leaf cells as described in Lemma 3, then p_2 must lie outside of γb_w. In such cases, clearly $\mathsf{d}(p_1, p_2) \geq r_w^-/2$. The remaining possibility is that w_1 was resolved after computing the set of representatives. From the described procedure, in Case 1, we know

that if $\mathsf{d}(p_1, p_2) < r_w^-/2$, the point $x_{p_1,i}$ (which is a 1-approximate nearest-neighbor of p_1 among points in P_i) would hold that $\mathsf{d}(p_1, x_{p_1,i}) < r_w^-$. Hence, $x_{p_1,i}$ should have been added to the set of representatives of w_1, contradicting the assumption that w_1 is resolved, or that C_{w_1} does not contain i. All together, we have that $\mathsf{d}(p_1, p_2) = \Omega(r_w^-) = \Omega(\varepsilon\gamma r_w)$. From the definition of the set of important leaf cells, $w \in \mathcal{I}$. ◀

Lemmas 7 and 8 imply that all the leaf cells used to build T belong to the set of important leaf cells (*i.e.*, $\mathcal{A} \cup \mathcal{R}_b \subseteq \mathcal{I}$), whose size is upper-bounded by Lemma 6. All together, and leveraging construction methods of quadtrees (see Section 2), the size of T is proportional to the total number of leaf cells used to build it, which we now know is $\mathcal{O}(k_\varepsilon \gamma^d \log \frac{1}{\varepsilon})$. However, we also need to account for the set of ε-representatives stored for each ambiguous leaf cell, leading to a worst-case upper-bound of $\mathcal{O}(k_\varepsilon \gamma^d \log \frac{1}{\varepsilon} \cdot 1/(\varepsilon\gamma)^{\frac{d-1}{2}})$ total space to store T.

4.2 Spatial Amortization

The previous result can be improved by applying a technique called *spatial amortization*, described by Arya *et al.* [5]. That is, we can remove the extra $\mathcal{O}(1/(\varepsilon\gamma)^{\frac{d-1}{2}})$ factor from the analysis of the space requirements for T.

This will be twofold process, as in order to successfully apply spatial amortization to the analysis of the data structure, we first need to further reduce the set of ε-representatives of every ambiguous leaf cell in the tree. Actually, the new set will no longer be a set of ε-representatives, but it will just be a *weak* ε-coreset for query points inside of each leaf cell.

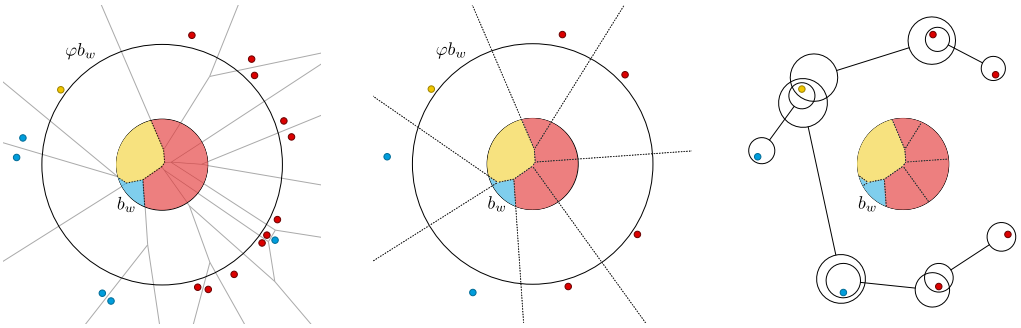
▶ **Lemma 9.** *The total space required to store the ambiguous leaf cells of T is $\mathcal{O}\big(k_\varepsilon \gamma^d \log \frac{1}{\varepsilon}\big)$.*

Consider any ambiguous leaf cell w of T, and in particular, consider the set R_w of ε-representatives of w. By construction, R_w has the property that every point $q \in b_w$ has at least one ε-approximate nearest-neighbor in the set R_w. However, note that the opposite is not necessarily true, as not every $p \in R_w$ is an ε-approximate nearest-neighbor of some point in b_w. Even worst, while the fact the w is ambiguous indicates that at least two points in R_w belong to K_ε, the remaining points of R_w might not, which in turn prevents the application of a spatial amortization analysis. Overall, this suggests some of the points of R_w might not be necessary to distinguish between the classes that change the classification of points inside b_w (see Figure 4a).

This can be resolved as follows. Suppose we have access to the Voronoi diagram of the set of points R_w, and consider the boundaries between adjacent cells of this diagram. Any boundary that separates two points of R_w of different classes, and that intersects b_w, is relevant to the classification any query point inside b_w. Formally, we define the set $R'_w \subseteq R_w$ of border points of R_w as (see Figure 4b):

$$R'_w = \{p \in R_w \mid \exists\, q \in b_w, p' \in R_w \text{ such that } l(p) \neq l(p') \wedge \mathsf{d}(q,p) = \mathsf{d}(q,p')\}$$

This new set R'_w has some useful properties. Note that for any query point $q \in b_w$, its (exact) nearest-neighbor in R'_w belongs to the same class as its (exact) nearest-neighbor in R_w, which itself is an ε-approximate nearest-neighbor of q among the points of P. In other words, R'_w is an ε-coreset for any query point in b_w. This implies that we can replace the set of ε-representatives of w with the set R'_w. Moreover, this means that by the definition of ε-border points, $R'_w \subseteq K_\varepsilon$. Note that we don't need to compute the Voronoi diagram of R_w, but instead just part of the 1-skeleton. While not immediately evident, the set R'_w can be computed in time $\mathcal{O}(|R_w|^3)$, by fixing every two pairs of R_w, and checking whether there exists a point $q \in b_w$ with the desired properties. The later step can be solved using Linear Programming via the lifting transform into a parabola in $d+1$ Euclidean space.



(a) Set of ε-representatives R_w. **(b)** Set of border points R'_w. **(c)** Relevant pairs of $\mathcal{D}''$.

Figure 4 The set R_w of ε-representatives of w can be reduced to the set R'_w. This later set is a subset of K_ε, and can be charged to a proportional number of relevant pairs of $\mathcal{D}''$.

Now, let's proceed with the charging argument over the pairs of the same WSPD $\mathcal{D}''$ used in Lemma 6. Instead of only charging w to a single pair (as described in Lemma 7), we charge every point stored in R'_w to a pair of $\mathcal{D}''$. Thus, consider the following procedure to find a sufficient number of pairs to charge all the points in R'_w, which is derived from a similar procedure proposed in [5]. See Figure 4c for an illustrative example.

1. Compute a net of R'_w using radius r_w^-, and denote this subset R''_w. Given that all the points of R'_w that belong to the same class are already separated by a distance of at least r_w^-, we know that $|R''_w| = \Theta(|R'_w|)$, hiding constants[3] that depend exponentially on d.
2. Find the two of points of $p_1, p_2 \in R''_w$ with smallest pairwise distance, and consider the pair of $\mathcal{P} \in \mathcal{D}''$ that contains both points p_1 and p_2, each in one of its dumbbell heads. Note that by having computed a net in the previous step, $\mathsf{d}(p_1, p_2) \geq r_w^-$.
3. Delete one of the two points from R''_w (without lost of generality, delete p_1).
4. Charge every point of R'_w that is covered by p_1 (*i.e.*, whose distance to p_1 is $\leq r_w^-$) to the pair $\mathcal{P}$. By the arguments described in step 1 on the size of R''_w, we know that $\mathcal{P}$ receives a charge from $\mathcal{O}(1)$ points of R'_w.
5. Repeat steps 2-4 with the remaining points of R''_w until the set is empty.

Evidently, the number of pairs found (and charged) equals $|R''_w| - 1$. All together, we have that the total space required to store all the ambiguous leaf cells is proportional to the sum of charges to every pair of $\mathcal{D}''$. Using the same arguments as Lemma 6, this implies that the total storage is $\mathcal{O}(k_\varepsilon \gamma^d \log \frac{1}{\varepsilon})$. This completes the proof of Theorem 1.

References

1 Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high dimensions. *arXiv preprint arXiv:1806.09823*, 2018.

2 Fabrizio Angiulli. Fast nearest neighbor condensation for large data sets classification. *IEEE Transactions on Knowledge and Data Engineering*, 19(11):1450–1464, 2007.

3 Sunil Arya, Guilherme D. da Fonseca, and David M. Mount. Optimal approximate polytope membership. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 270–288. SIAM, 2017.

[3] More specifically, we know that $|R''_w| \geq |R'_w|/\phi^{d-1}c$, where ϕ^{d-1} is the kissing number in d-dimensional Euclidean space, and c is the number of classes in P.

4 Sunil Arya, Guilherme D. Da Fonseca, and David M. Mount. Approximate polytope membership queries. *SIAM Journal on Computing*, 47(1):1–51, 2018.

5 Sunil Arya, Theocharis Malamatos, and David M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *Journal of the ACM (JACM)*, 57(1):1, 2009.

6 Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998. doi:10.1145/293347.293348.

7 Ricardo Barandela, Francesc J Ferri, and J Salvador Sánchez. Decision boundary preserving prototype selection for nearest neighbor classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(06):787–806, 2005.

8 Ahmad Biniaz, Sergio Cabello, Paz Carmi, Jean-Lou De Carufel, Anil Maheshwari, Saeed Mehrabi, and Michiel Smid. On the minimum consistent subset problem. In *Workshop on Algorithms and Data Structures*, pages 155–167. Springer, 2019.

9 Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

10 Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM (JACM)*, 42(1):67–90, 1995.

11 Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

12 T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13(1):21–27, January 1967. doi:10.1109/TIT.1967.1053964.

13 Luc Devroye. On the inequality of cover and hart in nearest neighbor discrimination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):75–78, 1981.

14 Raphael A. Finkel and Jon Louis Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9, 1974.

15 E. Fix and J. L. Hodges. Discriminatory analysis, nonparametric discrimination: Consistency properties. *US Air Force School of Aviation Medicine*, Technical Report 4(3):477+, 1951.

16 Alejandro Flores-Velazco. Social distancing is good for points too! In *Proceedings of the 32st Canadian Conference on Computational Geometry, CCCG 2020, August 5-7, 2020, University of Saskatchewan, Saskatoon, Saskatchewan, Canada*, 2020.

17 Alejandro Flores-Velazco and David M. Mount. Guarantees on nearest-neighbor condensation heuristics. In *Proceedings of the 31st Canadian Conference on Computational Geometry, CCCG 2019, August 8-10, 2019, University of Alberta, Edmonton, Alberta, Canada*, 2019.

18 Alejandro Flores-Velazco and David M. Mount. Coresets for the nearest-neighbor rule, 2020. arXiv:2002.06650.

19 Alejandro Flores-Velazco and David M. Mount. Coresets for the Nearest-Neighbor Rule. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms (ESA 2020)*, volume 173 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 47:1–47:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2020.47.

20 Alejandro Flores-Velazco and David M. Mount. Guarantees on nearest-neighbor condensation heuristics. *Computational Geometry*, 95:101732, 2021. doi:10.1016/j.comgeo.2020.101732.

21 Lee-Ad Gottlieb, Aryeh Kontorovich, and Pinhas Nisnevitch. Near-optimal sample compression for nearest neighbors. In *Advances in Neural Information Processing Systems*, pages 370–378, 2014.

22 Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, 2020. URL: https://arxiv.org/abs/1908.10396.

23 Sariel Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 94–103, 2001.

24 P. Hart. The condensed nearest neighbor rule (corresp.). *IEEE Trans. Inf. Theor.*, 14(3):515–516, 1968. doi:10.1109/TIT.1968.1054155.

25 Norbert Jankowski and Marek Grochowski. Comparison of instances selection algorithms I. Algorithms survey. In *Artificial Intelligence and Soft Computing-ICAISC 2004*, pages 598–603. Springer, 2004.

26 Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*, 2017.

27 Marc Khoury and Dylan Hadfield-Menell. Adversarial training with Voronoi constraints. *CoRR*, abs/1905.01019, 2019. arXiv:1905.01019.

28 Nicolas Papernot and Patrick McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

29 Neehar Peri, Neal Gupta, W. Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P. Dickerson. Deep k-nn defense against clean-label data poisoning attacks. In *European Conference on Computer Vision*, pages 55–70. Springer, 2020.

30 G. L. Ritter, H. B. Woodruff, S. R. Lowry, and T. L. Isenhour. An algorithm for a selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21(6):665–669, 1975.

31 Hanan Samet. *The design and analysis of spatial data structures*, volume 85. Addison-Wesley Reading, MA, 1990.

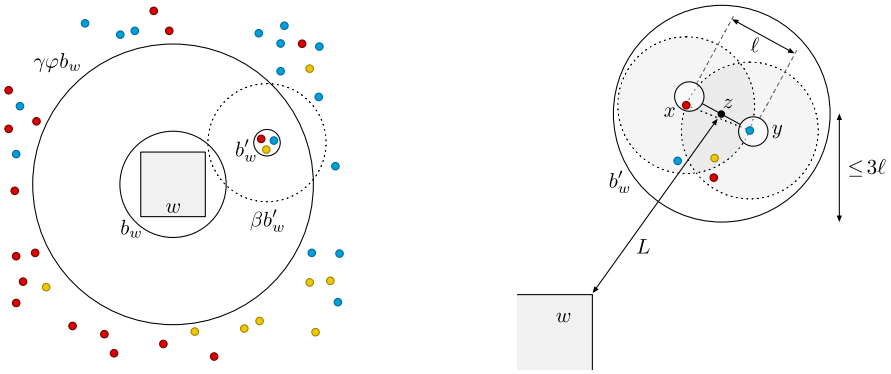
32 Jürgen Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014. arXiv:1404.7828.

33 Chawin Sitawarin and David Wagner. On the robustness of deep k-nearest neighbors. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 1–7. IEEE, 2019.

34 Charles J. Stone. Consistent nonparametric regression. *The annals of statistics*, pages 595–620, 1977.

35 Godfried Toussaint. Open problems in geometric methods for instance-based learning. In Jin Akiyama and Mikio Kano, editors, *JCDCG*, volume 2866 of *Lecture Notes in Computer Science*, pages 273–283. Springer, 2002. doi:10.1007/978-3-540-44400-8_29.

A Proof Details



(a) (b)

Figure 5 It is possible that points of ≥ 2 classes lie inside of $\gamma\varphi b_w$. However, this case can be reduced to the two separation properties illustrated in Figure 3.

To finish the proof of Lemma 3, we shall proof the assumption that if $|4\gamma\varphi b_w \cap \{x_i\}_i| \geq 2$, then every point inside $4\gamma\varphi b_w$ lies inside the ball b'_w which is β-separated from w.

Proof. Let $x, y \in 4\gamma\varphi b_w$ be the two points of different classes inside the ball $4\gamma\varphi b_w$ with largest pairwise distance. Thus, it is easy to show that all the points inside $4\gamma\varphi b_w$ lie inside the two balls centered at x and y with radii equal to $\mathsf{d}(x, y)$, as shown in Figure 5b. By definition of the (bichromatic) well-separated pair decomposition $\mathcal{D}'$, there exists a pair $\mathcal{P} \in \mathcal{D}'$ that contains x and y each in one of its dumbbells, with length ℓ and center z. Now, we define the ball b'_w with center at z and radius $r'_w = \max(\mathsf{d}(z, x), \mathsf{d}(z, y)) + \mathsf{d}(x, y)$. By definition of $\mathcal{P}$, we know that $\mathsf{d}(z, x), \mathsf{d}(z, y) \leq \ell$ and $\mathsf{d}(x, y) \leq 2\ell$, thus making $r'_w \leq 3\ell$. Let L be the distance from w to z, we distinguish two cases based on the relationship between L and ℓ:

- $L > c_1\beta\ell$. Consider the distance that separates the ball b'_w from the cell w.

$$\mathsf{d}(w, b'_w) = L - r'_w > c_1\beta\ell - r'_w \geq (c_1\beta/3 - 1)\, r'_w$$

  Since $\beta = 1/\varepsilon$, for all sufficiently large constants $c_1 \geq 3(1 + \varepsilon)$, the distance $\mathsf{d}(w, b'_w)$ exceeds $\beta r'_w$ which implies that b'_w is concentrically β-separated from w.
- $L \leq c_1\beta\ell$. We will show that this case cannot occur, since otherwise the dumbbell $\mathcal{P}$ would have caused w to be split, contradicting the assumption that it is a leaf cell of T_{init}. Since x, y, and w are all contained in the ball $4\gamma\varphi b_w$ whose center lies within w, we have both that $\mathsf{d}(x, w) \leq 4\gamma\varphi r_w$, and $\ell < 2\mathsf{d}(x, y) \leq 2(8\gamma\varphi r_w) = 16\gamma\varphi r_w$. Thus, by the triangle inequality, we have:

$$L \leq \mathsf{d}(x, y) + \mathsf{d}(x, w) < \ell + 4\gamma\varphi r_w < 16\gamma\varphi r_w + 4\gamma\varphi r_w = 20\gamma\varphi r_w$$

  Because $L \leq c_1\beta\ell$, it follows from our construction that there is at least one ball $b_i(\mathcal{P})$ (with $0 \leq i \leq \lceil \log c_1\beta \rceil$) that overlaps w. Let b denote the smallest such ball, and let r denote its radius. By the construction, we have that $r \leq \max(\ell, 2L)$. Since our construction generates all quadtree boxes of size $r/(c_2\gamma)$ that overlap b, it follows that $s_w \leq r/(c_2\gamma)$. Thus, we have:

$$r_w = s_w \frac{\sqrt{d}}{2} \leq \frac{r\sqrt{d}}{2c_2\gamma} \leq \frac{\max(\ell, 2L)\sqrt{d}}{2c_2\gamma} < \frac{20\gamma\varphi r_w\sqrt{d}}{c_2\gamma} = \frac{20\varphi r_w\sqrt{d}}{c_2}$$

  Choosing $c_2 \geq 20\varphi\sqrt{d}$ yields the desired contradiction.

  This completes the proof of Lemma 3. ◀