Voronoi Boundary Classification: A High-Dimensional Geometric Approach via Weighted Monte Carlo Integration

Vladislav Polianskii [1] **Florian T. Pokorny** [1]

Abstract

Voronoi cell decompositions provide a classical avenue to classification. Typical approaches however only utilize point-wise cell-membership information by means of nearest neighbor queries and do not utilize further geometric information about Voronoi cells since the computation of Voronoi diagrams is prohibitively expensive in high dimensions. We propose a Monte-Carlo integration based approach that instead computes a weighted integral over the boundaries of Voronoi cells, thus incorporating additional information about the Voronoi cell structure. We demonstrate the scalability of our approach in up to 3072 dimensional spaces and analyze convergence based on the number of Monte Carlo samples and choice of weight functions. Experiments comparing our approach to Nearest Neighbors, SVM and Random Forests indicate that while our approach performs similarly to Random Forests for large data sizes, the algorithm exhibits non-trivial data-dependent performance characteristics for smaller datasets and can be analyzed in terms of a geometric confidence measure, thus adding to the repertoire of geometric approaches to classification while having the benefit of not requiring any model changes or retraining as new training samples or classes are added.

1. Introduction and Related Work

The problem of classifying a set of n points $\{x_1, \dots, x_n\}$ in d-dimensional Euclidean space, each labelled with one of k classes has been tackled with a large variety of approaches in recent decades. In many application fields, Deep Neural networks are today leading the field in terms of classification rates, particularly in the area of image classification (Krizhevsky et al., 2012). A draw-back of these approaches to date is the difficulty of reasoning about the learned representations in terms of convergence guarantees, stability and the complex structure of learned embeddings leading to challenging phenomena such as adverserial examples (Kurakin et al., 2016). In contrast, a benefit of purely geometric approaches such as nearest neighbors – while not always having optimal performance – is their simplicity and asymptotic convergence under suitable assumptions (Döring et al., 2017). A further advantage is the lack of dependence on initialization since no stochastic gradient descent or complex variational inference procedure is required. This work contributes to this category of *optimization-free*, *non-Bayesian* and *purely geometric* approaches to classification with a focus on *simpliticy* and *mathematical interpretability*.

Recent works such as (Sharif Razavian et al., 2014) have demonstrated that classical methods such as SVM classifiers can provide competitive results when combined with representations determined by a neural network. Similarly, metric learning techniques (Weinberger & Saul, 2009) may be used to first determine an appropriate metric and/or embedding before applying geometric methods such as a nearest neighbor classifier. The Voronoi cell based approach proposed here can hence also be viewed as a component in the toolbox that may be combined with Bayesian or Neural Network based methods. In the present paper, we focus on the definition, algorithms, mathematical properties and an initial stand-alone evaluation of our approach.

While nearest-neighbor classifiers (Biau & Devroye, 2015) make a classification decision by testing which *training data Voronoi cell* a test data point $t \in \mathbb{R}^d$ belongs to, this membership test can be performed efficiently with approximate geometric lookup datastructures (Kushilevitz et al., 2000) and does not require a computation of the full geometry of the Voronoi cells themselves. Our approach is instead focused on the Voronoi cell $V(t)$ around the test data point t and balances the influence of neighboring training data on the classification decision by means of a weighted integral over the Voronoi boundary. A related approach consists of the Natural Neighbor regression method (Sibson, 1981), where volumes of intersections of Voronoi cells before and

[1]Division of Robotics, Perception and Learning, EECS, KTH Royal Institute of Technology, Stockholm, Sweden. Correspondence to: Vladislav Polianskii <vpol@kth.se>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

after insertion of a test-point are used as weights for interpolation. By considering a ranking of these weights this method could be considered as a related classification approach, but since current implementations rely on numerical integration and an explicit computation of the Voronoi cell structure, these are not applicable for high-dimensional data in their current form.

The key contributions of this work are: 1) We introduce and provide a convergence analysis of a conceptually simple Voronoi-based classifier based on a weighted integral formulation over boundary faces of $V(t)$. 2) We show how to overcome the computational complexity challenge of working with Voronoi cells in high dimensions by avoiding explicit calculation of the Voronoi structure. This is achieved by means of a Monte Carlo integral approximation method and, to the best of our knowledge, is the first time an integral over Voronoi cell geometry has been proposed and used for classification in high dimensions. 3) We evaluate our approach relative to Nearest Neighbors, SVM and Random Forests on 3 datasets, including CIFAR-10 and MNIST.

2. Methodology and Algorithm

We consider a test data point $t \in \mathbb{R}^d$ and a labeled training dataset $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ where each data point $x_i \in \mathbb{R}^d$ lies in Euclidean space and each x_i is associated to a unique class $c_i \in \{1, \ldots, k\}$. We assume furthermore that the points in X are in general position, which for data with uniform noise happens with probability one and which can alternatively be enforced by an arbitrarily small perturbation of the input data. For any $x \in \mathbb{R}^d$, we denote the Voronoi cell at x with respect to $\{t, x_1, \ldots, x_n\}$ by $V(x) = \{y \in \mathbb{R}^d : \|y - x\| \le \|y - x_j\|$ for all $i \in \{1, \ldots, n\}$ and $\|y - x\| \le \|y - t\|\}$.

We seek to determine a classification label for t and consider the Voronoi cell $V(t)$ as the basic geometric object upon which a classification decision will be made. Here, we think of each boundary face of $V(t)$ as being colored by a unique color corresponding to the class label of the neighboring x_i (see Fig.1). In intuitive terms, our basic motivation for the proposed method is that an observer placed at t may decide upon the classification label for t by ranking the relative influence of the observed colored faces of $V(t)$ by weighting *the distances between* t *and points on the boundary* and *the total area of the observed colored faces*. I.e. the class of t is determined by "how much of a given color is observed from t, weighted by distance". We introduce a weight function $w : \mathbb{R}^d \to \mathbb{R}_+$ that decays with distance from t and our proposed method utilizes a w-weighted integral over the boundary faces of $V(t)$ to achieve the above intuitive goal. Crucially, we will show in the following that this formulation can be implemented *without requiring the computation of the full Voronoi cell structure*, which is of complexity $O(n^{\lceil \frac{d}{2} \rceil})$ (Aurenhammer, 1991) and thus infeasible in high dimensions.

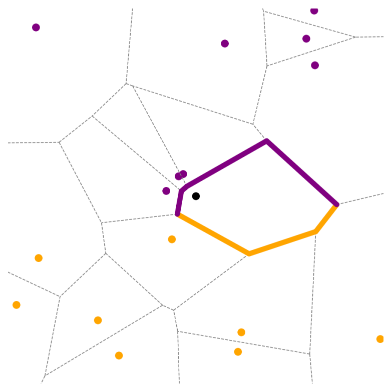


Figure 1. Planar classification problem with two classes of training data (purple / orange) as well as test data point t in black. Our approach ranks the influence of each neighboring class on t, by approximating a weighted integral over the Voronoi cell boundary of t for each color.

Definition 1. *Consider a labeled training dataset* $\{(x_1, c_1), \ldots, (x_n, c_n)\}$, *where* $x_i \in \mathbb{R}^d$ *and* $c_i \in \{1, \ldots, k\}$ *for all* $i \in \{1, \ldots n\}$. *For any integrable function* $w : \mathbb{R}^d \to \mathbb{R}_+$, $t \in \mathbb{R}^d$ *and* $i \in \{1, \ldots, k\}$, *the* (weighted) Voronoi Boundary Rank of class i at t *is denoted by* $r_i(t)$ *and given by*

$$r_i(t) = \sum_{j : c_j = i} \int_{V(t) \cap V(x_j)} w(x) \mathrm{dVol},$$

where dVol *denotes the Volume form on* $V(t) \cap V(x_j)$ *induced by the Euclidean metric in* $\mathbb{R}^d$. *The (weighted) Voronoi Boundary Classifier is given by* $VBC(t) = \operatorname{argmax}_{i \in \{1, \ldots, k\}} r_i(t)$.

Observe that, in the above, each face $V(t) \cap V(x_j)$ is $d - 1$ dimensional (or empty) and may be unbounded. As a result, for w to be integrable over this domain, we consider w that decay with distance to t. As we shall explore later, exponentially decaying weight functions constitute a suitable choice, for example. We now turn to the problem of evaluating the integral $r_i(t)$. We approach this by first changing variables and performing the integration over a unit sphere instead.

Computation of Voronoi Boundary Rank

Lemma 1 (Change of Variables and Monte Carlo). *Let* $P = V(t) \cap V(x_j)$. *If* $P \neq \emptyset$, P *lies in some hyperplane* $\Pi = \{x \in \mathbb{R}^d : \langle x, n \rangle = c\}$, *where* n *is of unit norm. Denote by* $\mathbb{S}^{d-1}$ *the* $d - 1$ *dimensional unit sphere centered at the origin and denote by* f_j *the map which maps* $m \in \mathbb{S}^{d-1}$ *to to the intersection between the ray starting at* t *in direction* m *with* P *if this intersection exists. Let* $I_j =$

$\int_{V(t)\cap V(x_j)} w(x)\mathrm{dVol}$, *then*

$$I_j = \int_{f_j^{-1}(V(t)\cap V(x_j))} w(f_j(m)) \frac{\|f_j(m)-t\|^{d-1}}{|\langle m,n\rangle|} \mathrm{dVol}_{\mathbb{S}^{d-1}} \tag{1}$$

where $\mathrm{dVol}_{\mathbb{S}^{d-1}}$ *denotes the standard induced volume form of the sphere in Euclidean space. Secondly, consider a sequence* $\{m_i\}_1^T$ *of uniform samples on* $f_j^{-1}(V(t)\cap V(x_j)) \subset \mathbb{S}^{d-1}$. *We have*

$$I_j = V_j \lim_{T\to\infty} \frac{1}{T}\sum_{i=1}^{T} w(f_j(m_i)) \frac{\|f_j(m_i)-t\|^{d-1}}{|\langle m_i,n\rangle|} \tag{2}$$

where V_j *denotes the volume of the set* $f_j^{-1}(V(t)\cap V(x_j))$ *with respect to the standard induced metric on the unit sphere in Euclidean space.*

Proof. The first part above follows from the general integral change of variable formula and a consideration of Sylvester's determinant theorem to simplify the resulting expression, while the second part follows from the standard Monte Carlo integration convergence (Davis & Rabinowitz, 2007). See the supplementary material for additional details. □

We note that the above sampling based approximation of I_j still requires us to 1) perform uniform sampling on $f_j^{-1}(V(t)\cap V(x_j)) \subset \mathbb{S}^{d-1}$ and 2) to determine its volume V_j. We note that this can be achieved by means of rejection sampling by noting the following lemma.

Lemma 2. *Let* $t \in \mathbb{R}^d$ *and* $m \in \mathbb{S}^{d-1}$ *and define*

$$l_m^*(x) = \frac{\|x-t\|^2}{2\langle m, x-t\rangle}$$
$$\mathrm{face}(m) = \arg\min_{i\in\{1,\dots,n\}} \{l_m^*(x_i) \mid l_m^*(x_i) > 0\}.$$

If there is no $i \in \{1,\dots,n\}$ *such that* $l_m^*(x_i) > 0$, *then the ray* R *starting at* t *and in direction* m *is fully contained in* $V(t)$ *(and* $V(t)$ *is unbounded in direction* m *starting at* t*). Otherwise* m *lies in* $f_j^{-1}(V(t)\cap V(x_{\mathrm{face}(m)}))$.

Proof. See the supplementary material for a proof. □

Given Lemma 2, we can now uniformly sample T samples $S_T = \{m_1,\dots,m_T\}$ on $\mathbb{S}^{d-1}$ and count the number $T_j(S_T) = \#\{m \in S_T : \mathrm{face}(m) = j\}$ of samples mapped to $V(t)\cap V(x_j)$. This yields $V_j = \mathrm{Vol}(\mathbb{S}^{n-1}) \lim_{T\to\infty} \frac{T_j(S_T)}{T}$. We now have: $\frac{I_j}{\mathrm{Vol}(\mathbb{S}^{n-1})} = \lim_{T\to\infty}\frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{\mathrm{face}(m_i)=j} w(f(m_i)) \frac{\|f_j(m_i)-t\|^{d-1}}{|\langle m_i,n\rangle|}$ where $\mathbb{1}_{\mathrm{face}(m_i)=j}$ denotes the indicator function that is 1 if $\mathrm{face}(m_i) = j$ and zero otherwise. Since $r_i(t) = \sum_{j:c_j=i} I_j$, this provides the basis for computing the Voronoi Classification scheme we propose here. Observe that we in particular did not require an explicit computation of the boundary face geometry or volumes of $V(t)$. Additionally, a notable quality of the algorithm is that the Monte Carlo integration rate of convergence is $O(\frac{1}{\sqrt{T}})$ in arbitrary dimension (Doucet et al., 2001).

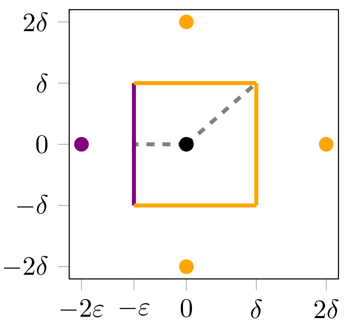


Figure 2. A counter-example for a bounded weight function.

Weight functions We now come to a discussion of suitable weight functions w for our proposed method. Observe that, since Voronoi Cells may be unbounded, we may encounter an integral over an unbounded domain in the definition of the Voronoi rank. To ensure that all integrals are finite we hence require that w decays with distance to the test data point t. In the following experiments, we in fact only consider $w(x) = \hat{w}(\|x-t\|)$ for some integrable function $\hat{w} : \mathbb{R}_+ \to \mathbb{R}_+$ that is monotonically decreasing.

An obvious choice may be a weight function of the form $\hat{w}(z) = e^{-0.5\sigma^{-2}z^2}$ which does indeed result in finite integrals, but does - as we shall now explain - have the property of favoring volume over distance in the relative contribution of the boundary of $V(t)$. To illustrate this, consider points in $\mathbb{R}^2$, placed as shown on Figure 2: a test point t is placed at the origin, one point of the first class at coordinates $(-2\varepsilon, 0)$ for some positive ε and 3 points of the second class at coordinates $(0, 2\delta)$, $(0,-2\delta)$ and $(2\delta, 0)$ for some positive δ. In this setting, it is easy to show that $r_1(t) \le \hat{w}(\varepsilon)\cdot 2\delta$ and $\hat{w}(\delta\sqrt{2})(4\delta + 2\varepsilon) \le r_2(t)$. If we decrease ε towards 0, the point of the first class starts to move closer to the test point. However, a simple analysis shows that if $\delta < \frac{1}{\sqrt{2}}\hat{w}^{-1}(0.5)$, then no matter how close the first point is moved towards the origin, $r_2(t) > r_1(t)$ and the classification chooses the second (orange) class - in effect the larger volume of the orange boundary component of $V(t)$ always dominates the classification.

This behavior may be considered unnatural if a balance between distance and volume contribution is desired, because an arbitrarily close point may be considered class-defining under the assumption that the data does not contain erroneous training labels.

If instead we consider $\hat{w}(z) = z^{-p}e^{-0.5\sigma^{-2}z^2}$ for $p \geq d$, this problem can be avoided and the integrals also converges for $\sigma = 0$. Furthermore, for $p = d$, the factors of $\|f_j(x)-t\|$ cancel in the Monte Carlo integration formula of Lemma 2.

Some examples of the classification boundary, including edge cases, produced by this weight function with different σ parameters are displayed in the top row of Figure 3. When σ is close to 0, the result becomes reminiscent of 1-Nearest Neighbor classification. This behavior is further discussed in Section 2. On the contrary, when σ is considerably large, the role of the distances decreases relative to the role of the boundary volumes. Another example that shows a comparison of the decision boundaries between SVM and our algorithm for a particular family of data is presented in the bottom row of Figure 3 where we illustrate that our classifier remains largely unchanged even when we introduce an imbalanced data split between classes. This is in contrast to the performance of support vector machines and artificial neural networks (Wang et al., 2016) on imbalanced data.

Optimized implementation We implemented our algorithm with a few optimization considerations in mind. Firstly, we utilized weight functions with a $\|f_j(m) - t\|^d$ factor to benefit from the previously discussed simplification and to avoid precision problems with computing large powers of real values in high dimensions. Another set of optimizations is based on the fact that most of the elements of the Monte Carlo integration formula can be unfolded and pre-calculated to reduce the computational complexity of these elements to constant lookup $O(1)$ during the main execution of the algorithm. Algorithm 1 presents the complete algorithm, optimized for a test dataset, rather than a single test point t. This batch implementation allows for efficient matrix operations as compared to a sequential implementation and was implemented using C++ with the use of the GPU library OpenCL for the most computationally intensive sections.

In the algorithm, initially we compute all pairwise distances between test and training data points. Then, during each iteration of ray sampling, we also precompute all inner products between data points and the sampled vector m. After that, by looking at each pair of points, we can find $l_m^*(\cdot)$ in constant time and perform an update to the current estimate of a corresponding area. By considering the nested for-loop structure of Alg 1, we observe that the running time is of order $O(NMD + TR_D + TND + TMD + TMN)$, where N denotes the number of training data points, M denotes the number of test data points, D the dimension of the space containing the data and R_D the complexity of generating a sample on the unit sphere in $\mathbb{R}^D$.

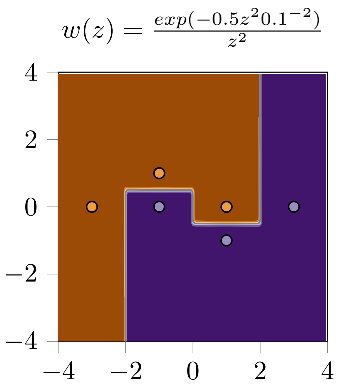
Algorithm 1 Classification Algorithm

```
 1: **Input:**
 2: Train, test data $X \in \mathbb{R}^{N\times D}$, $\hat{X} \in \mathbb{R}^{M\times D}$
 3: Train labels $Y \in \{1,\dots,k\}^N$
 4: Number of iterations $T \in \mathbb{N}$
 5: Weight function $w : \mathbb{R}_+ \to \mathbb{R}_+$
 6: **Output:**
 7: Predicted labels $\hat{Y} \in \{1,\dots,k\}^M$
 8: Compute matrix $dst \in \mathbb{R}^{N\times M}$ as
 9:     $dst_{ij} \leftarrow \|x_i - \hat{x}_j\|^2$
10: Initialize matrix $areas \in \mathbb{R}^{M,k}$ with zeros
11: **for** $it = 1$ **to** $T$ **do**
12:     Sample random $m \in \mathbb{R}^D$ from $Uni(S^{D-1})$
13:     Compute vector $p \in \mathbb{R}^N$ as
14:         $p_i \leftarrow \langle m, x_i\rangle$
15:     Compute vector $\hat{p} \in \mathbb{R}^M$ as
16:         $\hat{p}_i = \langle m, \hat{x}_i\rangle$
17:     **for** $j = 1$ **to** $M$ **do** // $t \leftarrow \hat{x}_j$
18:         Initialize $i_m = 0$, $l_m = +\infty$
19:         **for** $i = 1$ **to** $N$ **do**
20:             $l_{cur} \leftarrow dst_{ij}/(2\cdot(p_i - \hat{p}_j))$ // $l_{cur} \leftarrow l_m^*(x_i)$
21:             **if** $0 < l_{cur} < l_m$ **then**
22:                 $i_m \leftarrow i$, $l_m \leftarrow l_{cur}$
23:             **end if**
24:         **end for**
25:         **if** $i_m \neq 0$ **then**
26:             $q \leftarrow (p_{i_m} - \hat{p}_j)/sqrt(dst_{i_m j})$ // $q \leftarrow \langle m, n\rangle$
27:             Add $w(l_m)\cdot {l_m}^{D-1}/q$ to $areas_{j,y_{i_m}}$
28:         **end if**
29:     **end for**
30: **end for**
31: **for** $j = 1$ **to** $M$ **do**
32:     $\hat{y}_j \leftarrow \arg\max_l areas_{j,l}$
33: **end for**
```

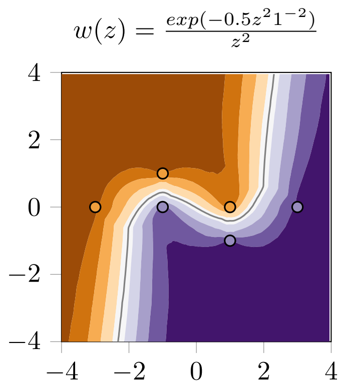
Approximate Nearest Neighbor Search for Large Data Note that, while our algorithm consists mainly of vectorized sequential operations and benefits from parallelizability, a key bottleneck for large datasets is the computation of all pairwise distances between train and test data points requiring $O(MN)$ storage and compute time. As an alternative, we propose an approximate algorithm, which can make use of a chosen approximate nearest-neighbor search algorithm such as (Muja & Lowe, 2014).

The key difference of the alternative method lies in the way of finding the $\mathrm{face}(m)$, which was introduced in Lemma 2. For a given sampled direction m we consider the intersection of the ray emanating from a test point t in direction m and use a binary search on the distance to the intersection from t, with the initial bounds set to 0 and MAX, where the latter is some chosen maximum value, such as the diameter of the dataset or a value where the weight function's value is sufficiently close to zero to be considered insignificant for the classification task.

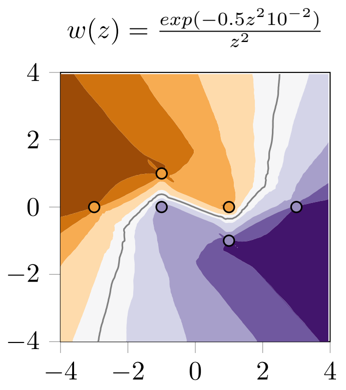
Note that, if the intersection occurs at a distance ρ, then



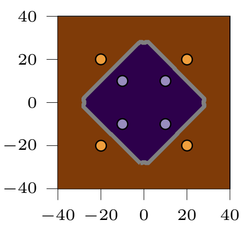
(a) $\sigma = 0.1$, comparable to 1NN.



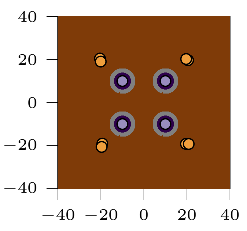
(b) $\sigma = 1.0$.



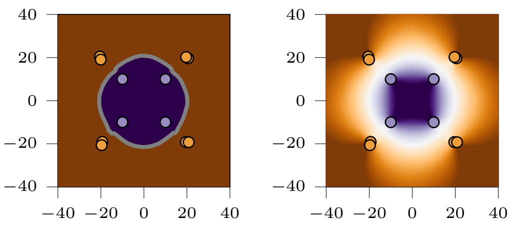
(c) $\sigma = 10.0$.



(d) SVM($\gamma = 0.1$) (4 vs 4).



(e) SVM($\gamma = 0.1$) (8 vs 4).



(f) VBC($\sigma = 10^2$) (8 "outer" points vs 4 "inner" points).

Figure 3. Top row: An example of a classification boundary on 6 training points. Bottom row: Comparison of SVM decision boundaries to our algorithm. The intensity of the color represents the ratio of the largest Voronoi rank divided by the sum of all Voronoi ranks at a given point (split into ten bins in 3a, 3b, 3c, two bins in 3f (left) and continuous in 3f (right)). Observe that as we double the outer training data density with a slight displacement from 4 to 8 between figure (d) and (e), the SVM classifier boundary changes drastically. In figure (f) we display the corresponding decision boundary for 8 points for VBC and the full level-set of the decision boundary for 4 points. The decision boundary in our approach remains largely unchanged since the Voronoi cell geometry is not affected significantly.

points on the ray with distances from the interval $[0, \rho]$ belong to test point's cell, and points with distances from (ρ, MAX) belong to other cells. Therefore, if t is our test point and $\mathrm{nearest}(\cdot)$ is the nearest neighbor function that returns the closest datapoint to a given one, then the binary search is performed over the function $h(l) = \mathbb{1}[\mathrm{nearest}(t + l \cdot m) = t]$, searching for the argument at which the function's value changes from 1 to 0. After the seach is complete, we can decide what cell is getting crossed by the ray by running the neareset neighbor algorithm for the intersection point one more time.

The nearest neighbor algorithm in this case may be replaced with an approximate search version, which would allow us to replace the mentioned part from the algorithm's complexity with an asymptotically smaller function. The complexity now becomes

$$O(\mathrm{prep}(N, D) + TM \log \frac{\mathrm{diam}(X)}{\varepsilon_0}(\mathrm{nn}(N, D) + D)) \quad (3)$$

where $\mathrm{prep}(N, D)$ is the time needed to construct a data structure for a nearest-neighbor search, $\mathrm{nn}(N, D)$ is the complexity of the search over N D-dimensional points (which might be considered as $D \log N$ for some algorithms (Arya et al., 1998)) and ε_0 is the desired precision for the binary search.

Convergence to 1-Nearest-Neighbor classifier As was observed empirically in Fig. 3, certain weight functions appear to return classification results comparable to a 1-nearest neighbor classifier. Here, we present a theorem making this observation precise:

Theorem 1. *Consider a labeled dataset $D = \{(x_1, c_1), \ldots, (x_n, c_n)\}$ of data points $x_i \in \mathbb{R}^d$ and corresponding class labels $c_i \in \{1, 2, \ldots, k\}$ and a test point $t \in \mathbb{R}^d$, where $t \neq x_i$ for all $i \in \{1, \ldots, n\}$. Assume that t only has a single nearest neighbor among $\{x_1, \ldots, x_n\}$. Consider a sequence of weight functions $\{w_n\}_{n=1}^{\infty}$, $w_n : \mathbb{R}_+ \to \mathbb{R}_+$ which are each monotonically decreasing and where*

$$\lim_{n\to\infty} \int_{z_2}^{+\infty} \frac{w_n(z)}{w_n(z_1)} z^{d-2}\, dz = 0 \quad \textit{for all} \quad 0 < z_1 < z_2 \quad (4)$$

Then, for sufficiently large n, the class $VBC(t|D, w_n)$ assigned by the Voronoi Boundary Classifier of t is equal *to the class $1NN(t|D)$ assigned by the nearest neighbor classifier for t.*

Proof. Please consult the supplementary material. □

Our standard weight function $\hat{w}(z) = \frac{e^{-0.5z^2\sigma^{-2}}}{z^p}$ in particular can be considered in the light of the above result. Consider two sequences: $w_n(z) = \hat{w}(z|\sigma = n^{-1}, p = p_0 \geq 1)$ and $w_n(z) = \hat{w}(z|\sigma = \sigma_0, p = n)$ for $n \in \mathbb{N}$. It is easy to see that both sequences follow the theorem and hence converge to nearest neighbor classification when used as weight functions for Voronoi Boundary Classification.

3. Experimental Evaluation

The following weight function was used in all experiments: $w_\sigma(z) = \|z\|^{-d}e^{-0.5\sigma^{-2}\|z\|^2}$, where d is the dimensionality of the data. The parameter σ varies for different datasets. We noted that in higher-dimensional spaces the classification performance appeared to improve with a larger σ.

Monte Carlo Convergence Firstly, we investigate the convergence of our proposed Monte Carlo Integration procedure. For these experiments, we used generated: 2D, 20D and 1000D normally distributed pointclouds of 100 points each and also considered the MNIST dataset which is 784 dimensional. We assigned all data points to a single class label 1 and study the convergence of the Voronoi Boundary Rank r_1 for a selection of random test points. All tests in this subsection use $\sigma = 10^3$ as the weight parameter. Each subplot on Figure 4 demonstrates 25 curves of values of $r_1(t)$ as we increase the number of Monte Carlo samples. We observe after only 10^5 iterations the approximated integrals start to stabilize. It should be noted that in this experiment the points are considered to belong all to 1 class, while in a classification task those calculated areas for each cell are split among several classes neighboring the test Voronoi cell. Note that, for MNIST, approximately 10^4 iterations were enough for areas to stabilize, while for the synthetic datasets, 10^5 iterations were necessary for 20D and 1000D cases.

Figure 5 furthermore focuses in on the Voronoi Boundary Rank convergence for two of these datasets, with low- and high-dimensional data, but for one randomly picked test data point and shows the standard deviations for 100 repeated random initializations of the algorithm. As expected from the theoretical Monte Carlo integration foundations, the variance curve can be approximated with a function $cn^{-0.5}$, where n is the number of iterations and this fitting is also displayed in the figure.

Classification Performance We now study the performance of our algorithm on the following datasets:

1. *Anuran Calls* – Frog sounds (MFCCs), 22 attributes, 10 species. 7195 data points split into 3598 training, 3597 test points (Dheeru & Karra Taniskidou, 2017).
2. *MNIST* – a classic dataset with $28 \times 28 \times 1 = 784$ dimensional dataset of images with handwritten digits. 60000 training data points, 10000 test data points (Deng, 2012).
3. *CIFAR-10* – a dataset with $32 \times 32 \times 3 = 3072$ dimensional image data which corresponds to 10 different classes; 50000 training data points, 10000 test data points (Krizhevsky et al., 2014).

Figure 6 first reports results of the convergence of the classification accuracy depending on the number of Monte Carlo samples. As can be seen, after 10^4 iterations the changes to the accuracy value become negligible and thus all classification tests were conducted with 10^4 Monte Carlo samples.

Our algorithm (VBC with 10000 samples) was then tested against 1NN, (KNN with $K \in \{2, \dots, 10\}$ results are not presented since examining only one neighbor was always the most optimal for these datasets), SVM (Chang & Lin, 2011) with an RBF kernel and default parameters from `scikit-learn 0.20.0` and RandomForest (Breiman, 2001) with 1500 estimators. The comparison is presented in Table 1, where results are averaged over 25 runs.

Table 1. Test results

	FROGS	MNIST	CIFAR-10
VBC($10k$)	**.986**±.0002	.969±.0003	**.494**±.0011
1NN	.982	.969	.354
SVM	.903	.944	.440
RF(1500)	.974±.0004	**.972**±.0004	**.495**±.0016

We observe that Voronoi Boundary Classification obtains satisfactory results, comparable to the best performance of the classical algorithms. In the case of the CIFAR dataset, where the nearest neighbor algorithm obtains low accuracy, *our algorithm, despite its superficial resemblance to 1NN, has a significantly better accuracy than 1NN* and is in line with the performance of random forests.

The running time of our algorithm was measured on the MNIST dataset, on an Intel Core i7-7700HQ processor and GeForce GTX 1060 graphics card. Performing the classification with 10000 Monte Carlo samples took 4 minutes 53 seconds for all 10000 test images, of which 15 seconds were spent on initialization of the algorithm and about 0.277 seconds for each ray casting iteration on average.

While these initial tests show the feasibility of VBC-classification on medium-sized datasets, we defer an in depth performance evaluation to future work since optimizations such as the mentioned approximate nearest neighbor search implementation and large scale implementation will require further work.

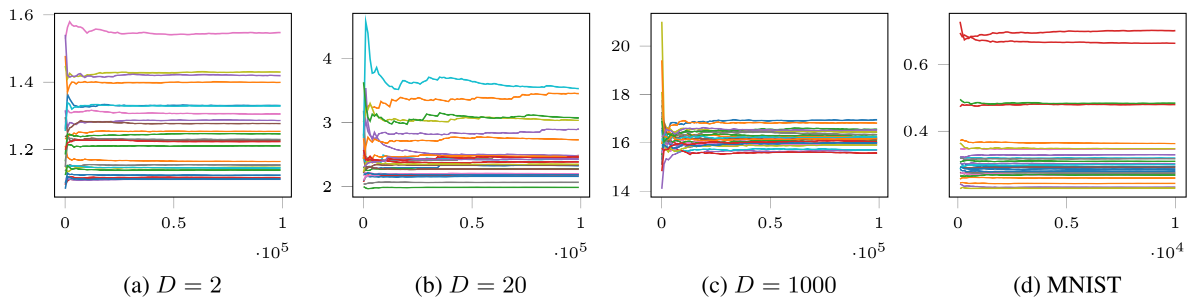


Figure 4. Area convergence results for normally distributed data and MNIST with $\sigma = 10^3$.

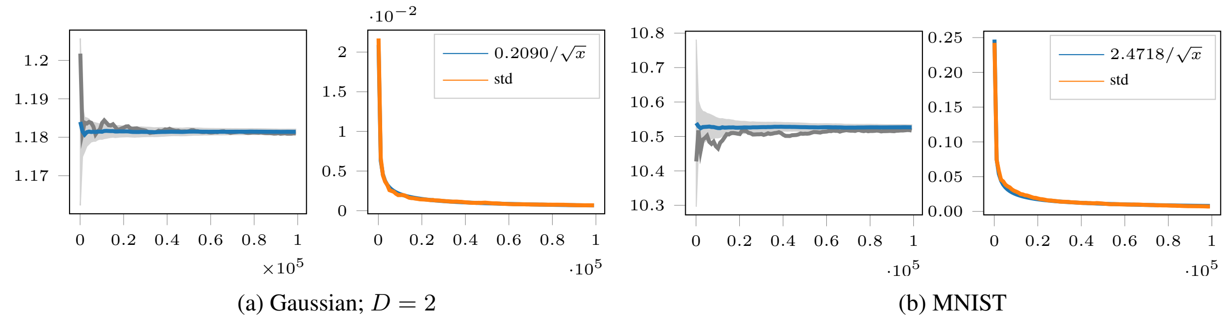


Figure 5. Area convergence for a single cell with weight function parameter $\sigma = 10^3$. The left plot in each subfigure represents the average Voronoi Boundary Rank, its standard deviation over 100 executions in grey. The dark gray curve represents a single run and the blue curve depicts the mean Voronoi Boundary Rank. The second and fourth figure each show the standard deviation and its MSE approximation with $c/\sqrt{x}$.

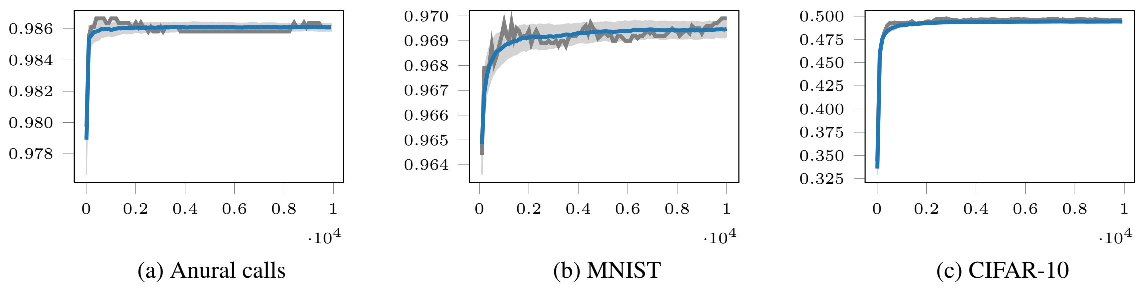


Figure 6. Accuracy plots on the given datasets. We display mean accuracy (blue) and standard deviation (light gray) error bounds over 100 iterations of our algorithm and an example run (dark gray) of our approach.

Performance dependence on training data size In Fig. 7, we study how the accuracy of the algorithms changes with the size of the training dataset. As all algorithms quickly improve in accuracy as the training dataset size increases, we display the difference in performance to the mean accuracy over all 4 studied algorithms. The results were obtained by sampling 25 random subsets of the data. The displayed graphs represent mean performance over those 25 subsets for each training data size setting. Note that we do not display the results for SVM which were significantly worse than the displayed values.

The intuition is that our algorithm should benefit, compared to say 1NN, when it becomes possible to approximate the Voronoi cell geometry.

We note that the Voronoi boundary classification algorithm performs well on both small and large datasets. On the contrary, most of the other algorithms do not have this feature or require additional tuning, that is dependent on the size of the data (as in the case with neural networks). We note that, particularly for CIFAR, and VBC performance for larger σ parameter, VBC performance is similar for Random Forests which performed well for large data size, while for small σ VBC performed similar to 1NN for large training datasets in line with Theorem 1.

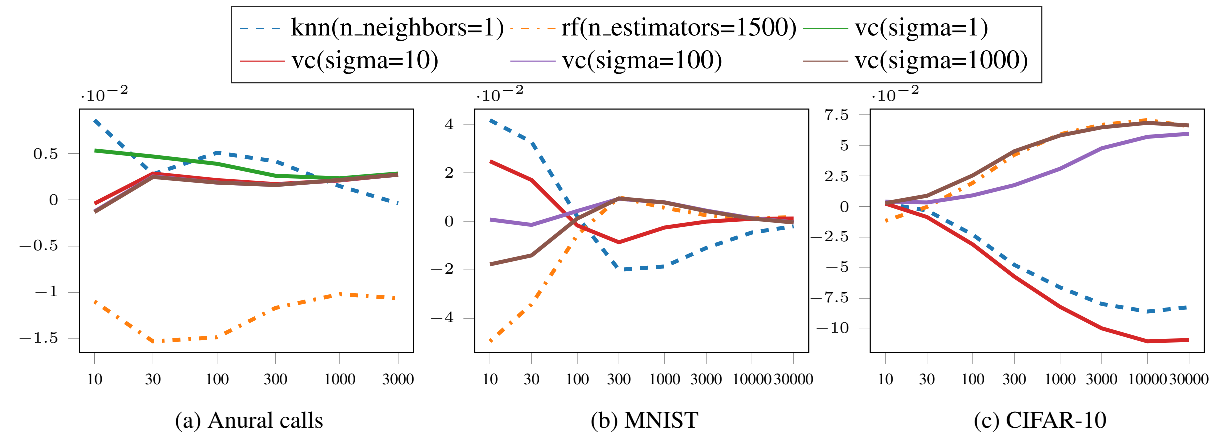


Figure 7. For each training data set size, we compute the relative difference between the algorithms' accuracy and mean accuracy over all 4 studied algorithms for that training data set size. Results are furthermore averaged over 25 random training datasets for each training data size setting.

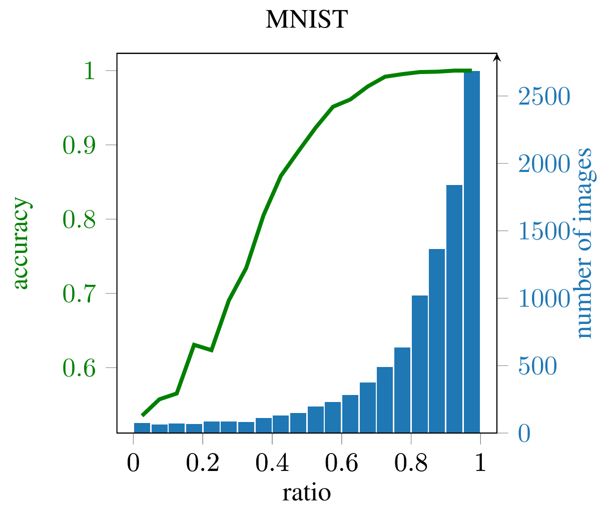


Figure 8. Correlation between accuracy and confidence value.

Confidence measurement We now consider a measure of confidence in our classifier provided by the formula $(S_1 - S_2)/(S_1 + S_2)$, where S_1 is the largest Voronoi Boundary Rank for a given point, and S_2 is the second largest Voronoi Boundary Rank.

The value of this ratio lies in $[0, 1]$, with 1 corresponding to the fact that only one class was encountered during integration, while 0 indicates that the algorithm has to pick randomly between at least two classes, due to corresponding integrals being equal. Figure 8 shows how the accuracy of our algorithm correlates with the described confidence levels on the MNIST dataset. Note that we observe that the algorithm has accuracy close to 100% for confidence values above 0.8, as well as a low accuracy of approximately 50% for confidence values close to 0.

We believe that confidence measures such as the proposed one may be useful in scenarios when the task of classification is extended by having an option of not assigning a class label in case of high uncertainty.

4. Conclusions and Future Work

We have proposed a geometric classification algorithm based on a weighted integral over Voronoi cell boundaries and demonstrated the performance of the algorithm on a set of example datasts.

We believe that the simplicity of the geometric formulation of this algorithm will allow for extensive analysis in terms of error-bounds, behavior depending on training data distribution assumptions etc. In future work, we hope to evaluate the performance of our approach on larger datasets and would like to generalize the ray-casting Monte Carlo integration based approach of the proposed algorithm to a larger family of geometric classification algorithms. Finally, we intend to test our approach also as a hybrid method in conjunction with Bayesian and Deep Neural Network based approaches.

5. Supplementary Material and Source Code

The authors will maintain updated versions of supplementary information for this paper at their respective websites `https://people.kth.se/~vpol` and `https://people.kth.se/~fpokorny`. Source code will be available at `https://github.com/vlpolyansky/voronoi-boundary-classifier`

6. Acknowledgements

This work has been supported by the Knut and Alice Wallenberg Foundation.

References

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

Aurenhammer, F. Voronoi diagramsa survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

Biau, G. and Devroye, L. *Lectures on the nearest neighbor method*. Springer, 2015.

Breiman, L. Random forests. *Machine learning*, 45(1): 5–32, 2001.

Chang, C.-C. and Lin, C.-J. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

Davis, P. J. and Rabinowitz, P. *Methods of numerical integration*. Courier Corporation, 2007.

Deng, L. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Dheeru, D. and Karra Taniskidou, E. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Döring, M., Györfi, L., and Walk, H. Rate of convergence of k-nearest-neighbor classification rule. *The Journal of Machine Learning Research*, 18(1):8485–8500, 2017.

Doucet, A., De Freitas, N., and Gordon, N. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pp. 3–14. Springer, 2001.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Krizhevsky, A., Nair, V., and Hinton, G. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Kushilevitz, E., Ostrovsky, R., and Rabani, Y. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.

Muja, M. and Lowe, D. G. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36, 2014.

Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: An astounding baseline for recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

Sibson, R. *A brief description of natural neighbor interpolation (Chapter 2), Interpolating Multivariate Data*. John Wiley, 1981.

Wang, S., Liu, W., Wu, J., Cao, L., Meng, Q., and Kennedy, P. J. Training deep neural networks on imbalanced data sets. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pp. 4368–4374. IEEE, 2016.

Weinberger, K. Q. and Saul, L. K. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.