Weighting vectors for machine learning: numerical harmonic analysis applied to boundary detection

Eric Bunch
American Family Insurance
Madison, WI 53783
ebunch@amfam.com

Daniel Dickinson
American Family Insurance
Madison, WI 53783
ddickins@amfam.com

Jeffery Kline
American Family Insurance
Madison, WI 53783
jklin1@amfam.com

Glenn Fung
American Family Insurance
Madison, WI 53783
gfung@amfam.com

Abstract

Metric space magnitude, an active subject of research in algebraic topology, aims to quantify the effective number of distinct points in a space. The contribution of each point to a metric space's global magnitude, which is encoded by the *weighting vector*, captures much of the underlying geometry of the original metric space. When the metric space is Euclidean, the weighting vector also serves as an effective tool for boundary detection. This allows the weighting vector to serve as the foundation of novel algorithms for classic machine learning tasks such as classification, outlier detection and active learning. We demonstrate, using experiments and comparisons on classic benchmark datasets, the promise of the proposed magnitude and weighting vector-based approaches.

1 Introduction

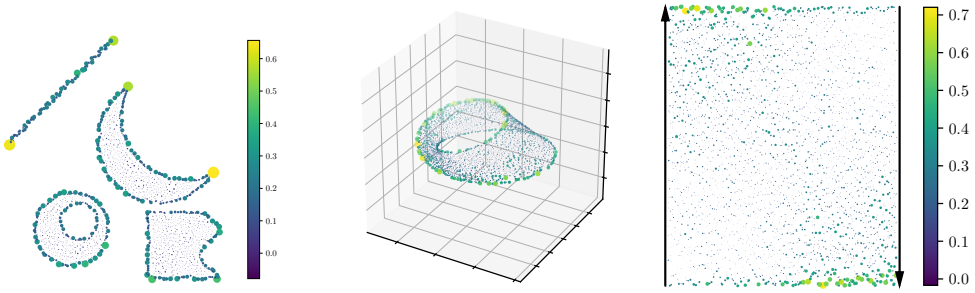

Figure 1: A visualization of two weighting vectors. The set in the left-hand figure is supported within four disjoint components, and they live in $\mathbb{R}^2$. The set in the right-hand figure is supported on an embedding of Möbius strip, and it lives in $\mathbb{R}^3$. In both images, the weight of each point is represented using color and point size.

Topological Data Analysis and Beyond Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Magnitude is a scalar quantity that has meaning for many different kinds of data, and as with other scalar quantities such as rank, diameter, and measure, it has wide applicability, an intuitive interpretation and a solid theoretical foundation. Magnitude has been discovered, and rediscovered multiple times in both practical and theoretical contexts. In this paper, our goal is to apply recent developments drawn from magnitude theory to machine learning, and to empirically demonstrate characteristics of magnitude that, while implicitly described by abstract theoretical results, have not, to our knowledge, been explicitly stated before, nor have they been leveraged for practical purpose.

Informally, magnitude aims to quantify the effective number of points in a space. Our aim is more subtle: we wish to identify *which points* are considered "effective" and "important." We do this using the *weighting vector*. The weighting vector appears naturally in the definition of magnitude, and we find that the weighting vector, under appropriate conditions, serves as an effective *boundary detector*. It is this behavior that makes the weighting vector especially well suited for machine learning tasks. The remainder of section 1 presents background and notation, section 2 demonstrates our algorithms, and seciton 3 has conclusions and directions for future work.

Definition 1. Let X be a finite metric space with metric d. Denote the number of points in X by $|X|$. The *similarity matrix* of X is defined to be $\zeta_X(i,j) := \exp(-d(x_i,x_j))$ for $1 \leqslant i,j \leqslant |X|$. Whenever the inverse of ζ_X exists, we define the *weighting vector* of X to be

$$w_X := \zeta_X^{-1}\mathbb{1},$$

where $\mathbb{1}$ is the $|X| \times 1$ column vector of all ones. The *magnitude* of X is defined to be the quantity

$$\mathrm{Mag}(X) := \mathbb{1}^T w_X = \mathbb{1}^T \zeta_X^{-1}\mathbb{1}.$$

That is, $\mathrm{Mag}(X)$ is the sum of all the entries of the weighting vector w_X.

Example. When X is a finite subset of Euclidean space, ζ_X is a symmetric positive definite matrix [Theorem 2.5.3, [5]]. In particular, ζ_X^{-1} is guaranteed to exist. Hence, the weighting vector and magnitude exist for finite subsets of $\mathbb{R}^n$.

It is not at all clear by inspection of the definition that a weighting vector might carry useful information about a set's boundary. There is early work empirically investigating the weighting vector for examples of finite metric spaces [11]. The theoretical foundation of using weighting vectors for boundary detection leverages the theory of Bessel potential functions and other machinery of harmonic analysis. We offer a hint at how the gap between established theory and empirical application is traversed. Let $\mathcal{X} \subset \mathbb{R}^n$ be a compact set that possesses a weighting (see [7] Theorem 4.1) and suppose $X \subset \mathcal{X}$ is a finite set, uniformly distributed about $\mathcal{X}$. The discrete sum that defines a weighting vector for X, namely $\zeta w_X = \mathbb{1}$, approximates an integral which, after introducing suitable concepts and notation, expresses a weighting function w as a function that satisfies $z * w = I_{\mathcal{X}}$, where $*$ denotes convolution and $I_{\mathcal{X}}$ is the characteristic function of $\mathcal{X}$. One then "solves" this linear equation to find $w = ZI_{\mathcal{X}}$, where Z is a generalized function. When n is odd, Z decomposes into a "nice" part on the interior of $\mathcal{X}$, and a singular part that is supported on $\partial\mathcal{X}$ ([1] Theorem 5 and [8]). More details are in Appendix B.

2 Algorithms

2.1 Weighting vector as a classifier

In this section, we develop an algorithm that uses metric space magnitude for a machine learning classification task. In a classification task, we are given a set X of m training examples in $\mathbb{R}^n$, $x_i \in X \subset \mathbb{R}^n$, $i \in \{1,2,...,m\}$, with $m < \infty$. Each x_i has an associated label, $l(x_i) \in L$, which is an element of a finite set of possible labels, $|L| = k < \infty$. Given an unlabeled new point, $x' \in \mathbb{R}^n$, we seek to assign it an associated label $l(x') \in L$. Since X is finite, the term "boundary" is not well-defined so we use the following convention: A point $x_i \in X \subset \mathbb{R}^n$ with $|X| < \infty$ is in the interior of X if its weight value is sufficiently small (where "sufficient" is context-specific). Note that our convention matches with intuition on densely sampled subsets of $Y \subset \mathbb{R}^n$, as all points away from $\partial\mathcal{Y}$ have small weight.

Let $L = \{L_1, L_2, ..., L_k\}$ be the set of labels, and $X_i = \{x \in X \mid l(x) = L_i\}$. If x' is an unlabeled point, the logic proceeds as follows. For each label L_i, compute w_i', the weight of x' in the set $\{x'\} \cup X_i$. Intuitively, if w_i' has a low value, it likely is an interior point of X_i and therefore $l(x') = L_i$

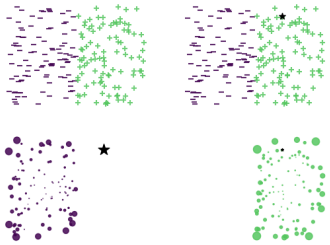

Figure 2: Upper left: Training data X, with $L_0 = -$ and $L_1 = +$. Upper right: $X \cup x'$, with a star denoting x'. Lower left: $\{x'\} \cup X_0$, with $w'_0 = 0.517$, and the sizes of markers indicate weight. Lower right: $\{x'\} \cup X_1$, with $w'_1 = 0.026$, and the sizes of markers indicate weight.

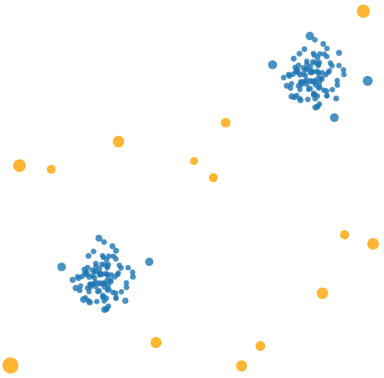

Figure 3: Outlier detection for synthetic data, $\tau = 0.2$. Inlier data was generated from two Gaussian distributions, and outliers were drawn from a uniform distribution.

is appropriate. However, if w'_i has a high value, it is likely not on the interior of X_i, so another label is more appropriate. Figure 2 shows an example.

If classes are imbalanced or have different underlying distributions, the values of w'_i will not necessarily be comparable. Our algorithm therefore incorporates a function $\text{SCALE}_i : (\mathbb{R}, \mathbb{R}^{|X_i|}) \to \mathbb{R}$, which serves to normalize w'_i relative to weights of other points with label L_i, for example absolute value or percentile.

We select a class label using a function DECIDE which operates on the w'_i after they have been scaled using SCALE_i, for example argmax. By allowing DECIDE to accept one additional threshold parameter, however, the algorithm can account for previously unseen classes as follows. If all w'_i are above the threshold parameter, it is likely the point is far from any of the labeled points, and thus from an unseen class, so it is assigned `NULL`. Otherwise, apply the decision function as described above. Note that for simplicity and readability, we omit the threshold parameter from the basic version presented in Algorithm 1.

Algorithm 1 Classification via weighting vector.

input: Data set X, $L = \{L_1, L_2, ..., L_k\}$ labels, function DECIDE : $\mathbb{R}^k \to \{1, 2, ..., k\}$, function $\text{SCALE}_i : (\mathbb{R}, \mathbb{R}^{|X_i|}) \to \mathbb{R}$ for each $i \in \{1, 2, ..., k\}$
input: unlabeled point x'
  p = []
  for $i \in \{1, 2, ..., k\}$ **do**
    $Y = \{x'\} \cup X_i$
    $w'_i = w_Y(x')$
    $w = \text{SCALE}_i(w'_i, W_{X_i})$
    p.append(w)
  let $j = \text{DECIDE}(p)$
output: L_j

2.2 Weighting vector for active learning

Let $\mathcal{L}$ (the labeled dataset) and $\mathcal{U}$ the (unlabeled dataset) be two subsets of the available pool of training data X, with $X = \mathcal{U} \cup \mathcal{L}$ and $\mathcal{U} \cap \mathcal{L} = \varnothing$. An iteration of the algorithm will pick some points in $\mathcal{U}$ to be labeled by an oracle (transferring them to $\mathcal{L}$). The current model will be then updated using the new updated dataset $\mathcal{L}$ and its corresponding labels. For simplicity we will state

the algorithm for a binary classification problem i.e. when $L = \{L_0, L_1\}$, however it can be trivially extended to a multi-class problem.

The intuition behind the algorithm is simple: at each iteration i, we assign every training data point to one of the sets $\tilde{X}_0$ or $\tilde{X}_1$ according to its predicted label by the current classifier f_i. We will calculate the corresponding weight vectors $w_{\tilde{X}_0}$ and $w_{\tilde{X}_1}$. Then, we choose to label the point with the minimum value (interior point) and the with the maximum value (likely to be in the boundary) for both sets $\tilde{X}_0$ and $\tilde{X}_1$. By choosing this way we are aiming to: (a) reinforce, validate and refine high confidence classifier information (labels) acquired in prior iterations (exploitation) and (b) to acquire labels in the predicted class boundaries where our classifier confidence is potentially lower (exploration). The proposed active learning algorithm is stated in Algorithm 2. We present some numerical experiments in Section A.

Algorithm 2 Active learning via weighting vector.

input: Data set X
$\mathcal{L} = \varnothing; \mathcal{U} = X$
initialize $\mathcal{L}$; $\mathcal{U} = X - \mathcal{L}$; with it's corresponding $\mathcal{Y}_{\mathcal{L}}$
$f = \mathbf{train\_classifier}(\mathcal{L}, \mathcal{Y}_{\mathcal{L}})$
while (not converged) **or** (labeling budget not reached) **do**
$\tilde{X}_i = \{x \in X \mid f(x) = i\}$ for $i = 0, 1$.
calculate weighting vectors $w_{\tilde{X}_i}$
$Q_{\min,i} = \arg\min_{\mathcal{U}} \text{abs}(w_{\tilde{X}_i})$ for $i = 0, 1$
$Q_{\max,i} = \arg\max_{\mathcal{U}} \text{abs}(w_{\tilde{X}_i})$ for $i = 0, 1$
$\mathcal{Y}_{\mathcal{Q}}$=query_labels($Q_{\min,0}$,$Q_{\max,0}$,$Q_{\min,1}$,$Q_{\max,1}$)
$\mathcal{L} = \mathcal{L} \cup \{Q_{\min,0}, Q_{\max,0}, Q_{\min,1}, Q_{\max,1}\}$
$\mathcal{Y}_{\mathcal{L}} = \mathcal{Y}_{\mathcal{L}} \cup \mathcal{Y}_{\mathcal{Q}}$
$\mathcal{U} = X - \mathcal{L}$;
$f = \mathbf{train\_classifier}(\mathcal{L}, \mathcal{Y}_{\mathcal{L}})$
output: f

2.3 Weighting vector for Outlier Detection

Suppose we have a data set $X \subset \mathbb{R}^n$, and wish to determine if a new point $x \in \mathbb{R}^n$ should be considered an outlier with respect to X. By looking at the value $\gamma_{Xx} := \text{Mag}(X \cup \{x\}) - \text{Mag}(X)$, we can see if adding x increased the magnitude substantially, thereby greatly extending the "border" of X. By Lemma 3.1.3 in [5] we have that $0 \leqslant \gamma_{Xx}$. Care must be taken, however; both the points on the boundary of X, and the outlier points will have high weight relative to the interior of the X. Thus we collect all points in X whose weight is below a fixed threshold, and denote this subset as X_{in}, the *inliers*. We say $X \backslash X_{in}$ are the *outlier candidates*. Next, for each $x \in X \backslash X_{in}$ with γ_{Xx} less than a user-defined threshold $0 \leqslant \tau$, we move from $X \backslash X_{in}$ to X_{in}. We record this algorithm in Algorithm 3, and Figure 3 displays example results of this algorithm.

Algorithm 3 Outlier detection via weighting vector.

input: dataset X, threshold τ
$X_{in} = \{x \in X \mid \text{abs}(w_X(x)) < \text{median}(w_X) + 1.5\text{std}(w_X)\}$
for $x \in X \backslash X_{in}$ **do**
if $\gamma_{Xx} < \tau$ **then**
$X_{in} \leftarrow x$
output: $X \backslash X_{in}$

3 Conclusions

We apply the concepts of metric space magnitude and weighting vector to a wide variety of classical machine learning tasks. We introduce practical algorithms that are suited to these tasks, and we

demonstrate performance that is competitive with, and in many cases, surpasses the performance of benchmark methods. Additionally, we introduce the notion that the weighting vector can accurately identify boundaries on scattered data that lives in a Euclidean space.

Prior work in the field of metric space magnitude has generally been theoretical and focused on the magnitude functional itself, and the properties of the weighting vector have been overshadowed. Practical aspects of metric space magnitude and the weighting vector is still an emergent field. Since magnitude and the weighting vector are well-defined for an extraordinarily wide class of sets, we believe that one natural aim of future work would be to develop vector weighting and magnitude into a robust, unifying foundation for the analysis of familiar, but also highly unusual, spaces.

Acknowledgments and Disclosure of Funding

We would like to thank Mark Meckes for extremely useful conversation related to this work.

References

[1] J. Barceló and A. Carbery. On the magnitudes of compact sets in Euclidean spaces. *American Journal of Mathematics*, 140(2):449–494, 2018.

[2] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[3] G. Folland. *Real analysis: modern techniques and their applications*. Pure and applied mathematics. Wiley, 1999.

[4] G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In *KDD '01*, 2001.

[5] T. Leinster. The magnitude of metric spaces. *Documenta Mathematica*, 18:857–905, 2013.

[6] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. *CoRR*, abs/cmp-lg/9407020, 1994.

[7] M. W. Meckes. Magnitude, diversity, capacities, and dimensions of metric spaces. *Potential Analysis*, 42(2):549–572, 2015.

[8] M. W. Meckes. Personal communication, September 2020.

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

[10] J. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293–300, 06 1999.

[11] S. Willerton. Heuristic and computer calculations for the magnitude of metric spaces, 2009.

A Results

A.1 Classification Experiments

To test the classification algorithm, we ran a set of ten experiments across 5 classic benchmark datasets from the UCI repository, a synthetic two-dimensional checkerboard dataset, as well as the scikit-learn digit and iris datasets, and multiple classifiers. Each experiment consisted of using a random stratified splitting method to partition the the data set into a training set consisting of 70% of the data, and a testing set consisting of the remaining 30%. The classifiers were trained without fine-tuning any parameters; the basic algorithm presented in 1 with ARGMAX for DECIDE and absolute value for SCALE_i, and the defaults in scikit-learn [9] for all parameters in the other algorithms. Table 1 records the average and standard deviation of the accuracy on the testing dataset for all classifiers.

Remark. Our model performed quite similarly to k-nearest neighbors in our experiments, which is quite remarkable given the dramatic differences between the algorithms. We also note the promise it implies: our initial attempt at using the boundary detection properties of the weighting vector in a machine learning setting have matched the performance of a well-established and widely-used model. We believe this will be improved upon and expanded as techniques using the weighting vector are adopted more widely.

dataset	K-Neighbors	Logistic Reg.	Rand. Forest	SVM	Weight
2-d checkerboard	0.92 ± 0.02	0.51 ± 0.04	$\mathbf{0.94 \pm 0.01}$	0.62 ± 0.04	0.92 ± 0.01
clevedata.mat	0.82 ± 0.04	$\mathbf{0.85 \pm 0.02}$	0.82 ± 0.03	0.84 ± 0.03	0.84 ± 0.03
dimdata.mat	0.94 ± 0.01	0.95 ± 0.01	0.95 ± 0.00	$\mathbf{0.96 \pm 0.00}$	0.93 ± 0.01
housingdata.mat	0.87 ± 0.02	0.87 ± 0.03	0.87 ± 0.02	0.87 ± 0.03	0.87 ± 0.02
ionodata.mat	0.84 ± 0.05	0.89 ± 0.02	0.94 ± 0.02	$\mathbf{0.95 \pm 0.02}$	0.81 ± 0.08
iris	0.94 ± 0.04	0.87 ± 0.05	0.94 ± 0.04	$\mathbf{0.96 \pm 0.03}$	0.85 ± 0.13
sklearn digits	0.97 ± 0.01	0.96 ± 0.01	0.95 ± 0.01	$\mathbf{0.98 \pm 0.01}$	0.97 ± 0.00
ticdata.mat	0.85 ± 0.02	0.69 ± 0.03	$\mathbf{0.93 \pm 0.02}$	0.88 ± 0.02	0.78 ± 0.03

Table 1: Classification performance results and benchmarks.

dataset	K-Neighbors	Logistic Reg.	Random Forest	SVM	Weight
2-d checkerboard	0.01 ± 0.00	0.01 ± 0.00	0.17 ± 0.02	0.02 ± 0.00	0.08 ± 0.01
clevedata.mat	0.01 ± 0.00	0.01 ± 0.00	0.15 ± 0.02	0.01 ± 0.00	0.02 ± 0.00
dimdata.mat	0.11 ± 0.01	0.02 ± 0.00	0.48 ± 0.06	0.07 ± 0.01	6.38 ± 0.62
housingdata.mat	0.01 ± 0.00	0.01 ± 0.00	0.15 ± 0.01	0.01 ± 0.00	0.03 ± 0.00
ionodata.mat	0.01 ± 0.00	0.01 ± 0.00	0.16 ± 0.01	0.01 ± 0.00	0.03 ± 0.00
iris	0.01 ± 0.00	0.01 ± 0.00	0.13 ± 0.01	0.01 ± 0.00	0.01 ± 0.00
sklearn digits	0.08 ± 0.01	0.09 ± 0.01	0.29 ± 0.04	0.10 ± 0.01	0.45 ± 0.05
ticdata.mat	0.02 ± 0.00	0.01 ± 0.00	0.16 ± 0.01	0.02 ± 0.00	0.09 ± 0.01

Table 2: Average with standard deviation of training times in seconds.

To demonstrate the `NULL` class label capabilities, we trained the magnitude classifier on examples of six and nine from the scikit-learn digits dataset, then predicted on images of ones, sixes, and nines. The confusion matrix with a `NULL` class threshold of $1 - 10^{-11}$ is shown in table 3.

A.2 Active learning Experiments

In order to assess the effectiveness of the weighting-vector-based active learning (AL) algorithm proposed in Section 2.2, we compared Algorithm 2 to the simplest but highly effective and most commonly used query AL framework: uncertainty sampling [6]. In this framework, the AL algorithm queries the instances for which it is least certain about how to label (i.e. for many algorithms $p(label \| x) \approx 0.5$ or where the decision function is close to 0). For simplicity we used a kernelized Ridge regression model [2] (also refer as to LS-SVM [10] or proximal SVM [4]). Laplacian kernels were used both as magnitude to calculate the weighting vector and as classification kernel ($k(x, y) = \exp(-\gamma \|x - y\|_1)$ with $\gamma = 0.1$. At each iteration of Algorithm 2 the classifier learned after obtained labels from the oracle has the form $f(x) = K(x, \mathcal{L})'w - w_0$, where w_0 is the bias term.

We performed experiments on five classic benchmark datasets from the UCI repository taking 67% of the data as training pool and the remaining 33% as a testing set. Note that the weighing-vector-

	null	6	9
null	53	0	1
6	1	53	0
9	1	0	54

Table 3: Confusion matrix for classifier with `NULL` class.

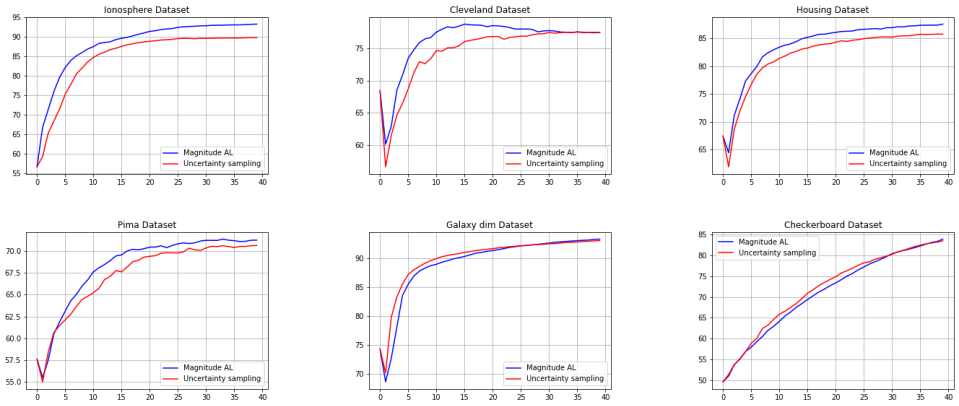

Figure 4: Active learning results comparing the weighting vector query strategy vs the uncertainty sampling strategy. Average over 100 runs.

inspired algorithm chooses 4 points per iterations so we picked the four more uncertain points for the uncertainty sampling algorithm to be fair.

Figure 4 shows average performance curves over 100 runs. The performance from the weighting vector algorithm seems to perform better in four out of the five datasets and slightly worse on the Galaxy dim. and Checkerboard datasets.

B Boundary detection

The purpose of this section is to state the theorem in [8], which contributes rigor to the above discussion about boundary detection. We begin by offering informal comments about how the finite, discrete sets of the applications relate to the infinite, continuous objects of the theorem. The background required for this initial part of the discussion is limited to basic familiarity with the Fourier transform. These comments also serve to present the moral case that weighting vectors ought to be useful as boundary detectors. The background required to interpret the theorem's statement includes substantial familiarity with tempered distributions and related theory.

Let $f : \mathbb{R}^n \to \mathbb{C}$ be a smooth integrable function. We define the Fourier transform of f as

$$(\mathcal{F}f)(\xi) := \int_{\mathbb{R}^n} e^{-2\pi i \xi' x} f(x)\, dx.$$

Then $\mathcal{F}$ may be extended unitarily to all square-integrable functions defined on $\mathbb{R}^n$, and since it is unitary, $\mathcal{F}$ has an inverse which we denote $\mathcal{F}^{-1}$. Under suitable conditions on f one has that

$$(2\pi i \xi)^k (\mathcal{F}f)(\xi) = \left(\mathcal{F}\left(\partial^k f\right)\right)(\xi). \tag{1}$$

Set $h_t(x) := e^{-2\pi t \|x\|}$. Then (see [3] Chapter 8)

$$(\mathcal{F}h_t)(\xi) = c_n \frac{t}{\left(t^2 + \|\xi\|^2\right)^{(n+1)/2}}, \tag{2}$$

where $c_n = \pi^{-(n+1)/2}\Gamma((n+1)/2)$.

Now suppose $X \subset [0,t]$ is a finite set of equispaced points selected from the interval $[0,t]$ (the equispacing condition may be relaxed to instead be a uniform random sample). Let $D(i,j) := e^{-\|x_i - x_j\|}$ denote the (i,j) entry of the matrix D. Then the linear equation that defines the weighting vector w is

$$Dw := \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

This statement has, via Riemann summation, a continuous analogue expression for a "weighting function" v for the entire interval, $[0, t]$. Let $h := h_{1/2\pi}$. This analogue has the form,

$$h * v(x) = \int_{\mathbb{R}} h(x-y)v(y)\,dy = \int_{\mathbb{R}} e^{-|x-y|}v(y)\,dy = I_{[0,t]}(x).$$

From Eq. 2,

$$\begin{aligned}\left(\mathcal{F}I_{[0,t]}\right)(\xi) &= \left(\mathcal{F}(h*v)\right)(\xi) \\ &= \left(\mathcal{F}h\right)(\xi)\left(\mathcal{F}v\right)(\xi) \\ &= \frac{2}{1+4\pi^2\xi^2}\left(\mathcal{F}v\right)(\xi),\end{aligned}$$

or

$$\frac{1}{2}(1+4\pi^2\xi^2)\mathcal{F}\left(I_{[0,t]}\right)(\xi) = \left(\mathcal{F}v\right)(\xi). \tag{3}$$

Applying Eq. 1 and the operator $\mathcal{F}^{-1}$ to Eq. 3, one has

$$\frac{1}{2}\left(I_{[0,t]} - \partial^2 I_{[0,t]}\right) = v. \tag{4}$$

In Eq. 4, the term $\partial^2 I_{[0,t]}$ vanishes everywhere except at the points 0 and t, where it behaves as second-order derivative operator. Informally, v is constant on the open interior $(0, t)$, and it approximates a discrete second-order derivative operator at the boundary points 0 and t. This informal argument may be adapted to $n > 1$ dimensions, where $I_{[0,t]}$ is replaced by more general $A \subset \mathbb{R}^n$.

We now turn to the theorem statement. The *Bessel potential space* is the Hilbert space of tempered distributions

$$H^s := \left\{\phi \in S'(\mathbb{R}^n) : (1+\|\cdot\|^2)^{s/2}\mathcal{F}\phi \in L^2(\mathbb{R}^n)\right\}$$

that is equipped with norm

$$\|\phi\|_{H^s} := \left(\int_{\mathbb{R}^n}\left(1+\|\xi\|^2\right)^s \left|\left(\mathcal{F}\phi\right)(\xi)\right|^2 d\xi\right)^{1/2}.$$

For for compact $A \subset \mathbb{R}^n$, the definition of a *weighting*, as well as necessary and sufficient conditions for A to have a weighting, can be found in Definition 3.3 and Theorem 4.1 of [7].

Theorem 2. *Let n be odd, $A \subset \mathbb{R}^n$ compact with weighting $z \in H^{-(n+1)/2}(\mathbb{R}^n)$, and let λ_A denote Lebesgue measure restricted to* $\operatorname{int} A$. *Then*

$$z = \frac{1}{n!\omega_n}\lambda_A + \nu \tag{5}$$

for some $\nu \in H^{-(n+1)/2}(\mathbb{R}^n)$ that is supported on ∂A. The constant $\omega_n := \pi^{n/2}/\Gamma(n/2+1)$ is the volume of the unit n-ball.

The decomposition of z stated in Eq. 5 agrees with informally-derived expression for v stated in Eq. 4. Numerically, we find that n odd does not seem to be required. Finally, we note that under extra regularity assumptions, a similar result follows from Theorem 5 of [1].

C Useful properties of magnitude

In this section, we offer some techniques that are useful when working with weighting vectors. We discuss how the computation of the weighting vector may be effectively computed by breaking the computation into smaller pieces and "gluing" the results together.

C.1 Inclusion-Exclusion for Weight and Magnitude

We demonstrate a practical way to calculate the weighting vector for a set $Z \coloneqq X \cup Y$ that is the union of two finite $X, Y \subset \mathbb{R}^n$. To approach this, first we investigate the case when X and Y are disjoint. Then we will look at the case $Y \subset X$, and show how to calculate either w_X or w_Y when one knows the other. Finally we will state an inclusion-exclusion principle for magnitude, as well as the weighting vector.

Before proceeding, we recall the definition of the *Schur complement*.

Definition 3. Let $M \coloneqq \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ be the block matrix where the matrices A, B, C, D are of dimensions $n \times n, n \times m, m \times n$, and $m \times m$ respectively. If D is invertible, then the *Schur complement* of D in M is the $n \times n$ matrix

$$M/D = A - BD^{-1}C.$$

Similarly, if A is invertible, then the Schur complement of A in M is the $m \times m$ matrix

$$M/A = D - CA^{-1}B.$$

Let $\varnothing \neq Y \subset X \subset \mathbb{R}^n$ be finite sets. Without loss of generality, we can index the points of X such that the first $|Y|$ of them correspond to those points in Y. Then we can see that ζ_X can be written as a block matrix

$$\zeta_X = \begin{bmatrix} \zeta_Y & \zeta_{Y,\bar{Y}} \\ \zeta_{Y,\bar{Y}}^T & \zeta_{\bar{Y}} \end{bmatrix}, \tag{6}$$

where $\bar{Y} = X \backslash Y$, and $\zeta_{Y,\bar{Y}}$ denotes the submatrix of ζ_X formed by taking the rows corresponding to Y and columns corresponding to $\bar{Y}$. We can now rewrite the formula $\zeta_X w = \mathbb{1}$ using equation 6 as the system of equations

$$\begin{aligned} \zeta_Y w_X\big|_Y + \zeta_{Y,\bar{Y}} w_X\big|_{\bar{Y}} &= \mathbb{1}_Y \\ \zeta_{Y,\bar{Y}}^T w_X\big|_Y + \zeta_{\bar{Y}} w_X\big|_{\bar{Y}} &= \mathbb{1}_{\bar{Y}}, \end{aligned}$$

where $\mathbb{1}_Y$ and $\mathbb{1}_{\bar{Y}}$ are respectively the $|Y| \times 1$ and $\left|\bar{Y}\right| \times 1$ column vectors of all ones. Since both ζ_Y and $\zeta_{\bar{Y}}$ are invertible, we can form both of the Schur complements ζ_X/ζ_Y and $\zeta_X/\zeta_{\bar{Y}}$. With these in hand, we can write

$$w_X\big|_Y = (\zeta_X/\zeta_{\bar{Y}})^{-1}(\mathbb{1}_Y - \zeta_{Y,\bar{Y}} w_{\bar{Y}}) \tag{7}$$

$$w_X\big|_{\bar{Y}} = (\zeta_X/\zeta_Y)^{-1}(\mathbb{1}_{\bar{Y}} - \zeta_{Y,\bar{Y}}^T w_Y), \tag{8}$$

where w_Y and $w_{\bar{Y}}$ are the weight vectors for Y and $\bar{Y}$ respectively, and $w_X\big|_Y$ is the weight vector of X, restricted to those indices corresponding to Y. Thus if we know w_Y and $w_{\bar{Y}}$, equations 7 and 8 give a way to compute w_X.

Next, for finite sets $Y \subset X \subset \mathbb{R}^n$ we wish to calculate either the weight vector w_X or w_Y given the other.

Definition 4. For a block matrix $M \coloneqq \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ with A invertible, define

$$\rho_{MA} \coloneqq \begin{bmatrix} A^{-1}B(M/A)^{-1}CA^{-1} & -A^{-1}B(M/A)^{-1} \\ -(M/A)^{-1}CA^{-1} & (M/A)^{-1} \end{bmatrix}.$$

Recall that for a block matrix M as in Definition 4,

$$M^{-1} = \begin{bmatrix} A^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \rho_{MA}. \tag{9}$$

Definition 5. For $Y \subseteq X \subset \mathbb{R}^n$ finite sets, assume ζ_X is in block matrix format as in Equation 6. Define the matrix

$$\rho_{XY} = \rho_{\zeta_X \zeta_Y}$$

where ρ_{XY} is taken to be the zero matrix when $Y = X$, and ρ_{XY} is taken to be ζ_X when $Y = \varnothing$.

Lemma 6. *For finite sets $Y \subset X \subset \mathbb{R}^n$, let P_{XY} be a permutation matrix such that*

$$P_{XY}\zeta_X P_{XY} = \begin{bmatrix} \zeta_Y & \zeta_{Y\bar{Y}} \\ \zeta_{Y\bar{Y}}^T & \zeta_{\bar{Y}}. \end{bmatrix}$$

Then

$$\begin{aligned} w_X &= P_{XY}\begin{bmatrix} w_Y \\ 0 \end{bmatrix} + P_{XY}\rho_{XY}\mathbb{1}, \quad \textit{and} \\ \mathrm{Mag}(X) &= \mathrm{Mag}(Y) + \mathbb{1}^T\rho_{XY}\mathbb{1}. \end{aligned}$$

Proof. This follows by setting $M = P_{XY}\zeta_X P_{XY}$, employing Equation 9, and multiplying on the right by $\mathbb{1}$. □

We can now calculate the weight vector of $X \cup Y$ where X and Y are not necessarily disjoint. This can be viewed as an inclusion-exclusion principle that applies to weight vectors as well as magnitude.

Theorem 7. *For finite sets $X, Y \subset \mathbb{R}^n$, set $Z = X \cup Y$. Then we have*

$$\begin{aligned} w_Z &= P_{ZX}\left(\begin{bmatrix} w_X \\ 0 \end{bmatrix} + \rho_{ZX}\mathbb{1}\right) + P_{ZY}\left(\begin{bmatrix} w_Y \\ 0 \end{bmatrix} + \rho_{ZY}\mathbb{1}\right) \\ &\quad - P_{ZX\cap Y}\left(\begin{bmatrix} w_{X\cap Y} \\ 0 \end{bmatrix} - \rho_{ZX\cap Y}\mathbb{1}\right), \quad \textit{and} \\ \mathrm{Mag}(Z) &= \mathrm{Mag}(X) + \mathrm{Mag}(Y) - \mathrm{Mag}(X \cap Y) \\ &\quad + \mathbb{1}^T\rho_{ZX}\mathbb{1} + \mathbb{1}^T\rho_{ZY}\mathbb{1} - \mathbb{1}^T\rho_{ZX\cap Y}\mathbb{1}. \end{aligned}$$

Proof. This follows by applying Lemma 6 to each subset considered, e.g.

$$w_Z = P_{ZX}\begin{bmatrix} w_X \\ 0 \end{bmatrix} + P_{ZX}\rho_{ZX}\mathbb{1}.$$

□

C.2 Numerical Considerations

In the setting where we have finite sets $Y \subset X \subset \mathbb{R}^n$, and we have calculated w_Y, we can calculate w_X without having to invert the entire matrix ζ_X using Corollary 6. Since

$$w_X = \begin{bmatrix} w_Y \\ 0 \end{bmatrix} + \rho_{XY}\mathbb{1},$$

we only need to invert the matrices ζ_Y–which we are assuming we have already done–and ζ_X/ζ_Y, which is an $|X\backslash Y| \times |X\backslash Y|$ matrix. Then all the matrix products must be performed in the block matrix formulation of ρ_{XY}. In particular, for the case when we are adding a single point to the set Y, ζ_X/ζ_Y is a scalar, and the products needed to form ρ_{XY} are matrix-vector products.