Practical applications of metric space magnitude and weighting vectors

Eric Bunch, Daniel Dickinson, Jeffry Kline, Glenn Fung

American Family Insurance
Madison, WI 53783
{ebunch, ddickins, jklin1, gfung}@amfam.com

Abstract

Metric space magnitude, an active subject of research in algebraic topology, originally arose in the context of biology, where it was used to represent the effective number of distinct species in an environment. In a more general setting, the magnitude of a metric space is a real number that aims to quantify the effective number of distinct points in the 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.

Surprisingly, 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.

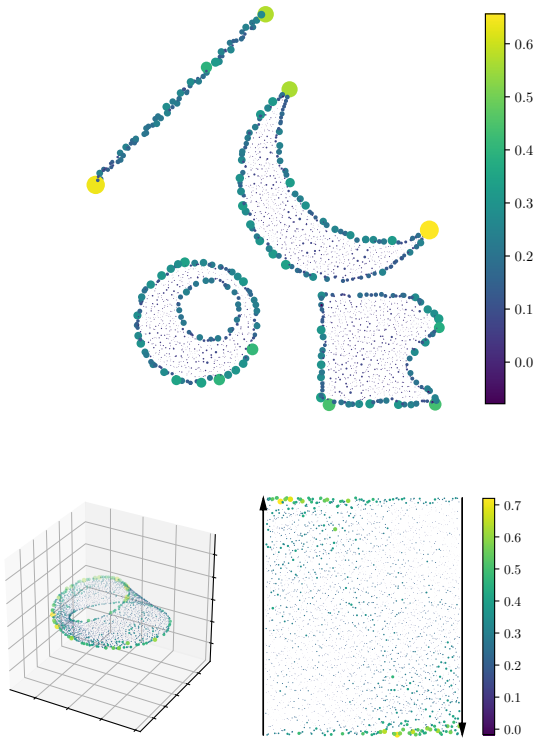

Figure 1. A visualization of two weighting vectors. The set in the top figure is supported within four disjoint components, and they live in $\mathbb{R}^2$. The set in the bottom figure is supported on an embedding of Möbius strip, and it lives in $\mathbb{R}^3$. In both images, the weight of each point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

1. Introduction

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.

1.1. Background, notation and examples

We now define magnitude and the weighting vector, we present several canonical examples, and we state several central theorems of the field. While our focus is largely on subsets of $\mathbb{R}^n$, we note that the concept *magnitude* and *weighting vector* can be defined for far more general types of sets.

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, (Leinster, 2013)]. In particular, ζ_X^{-1} is guaranteed to exist. Hence, the weighting vector and magnitude exist for finite subsets of $\mathbb{R}^n$.

Example. Given an undirected, unweighted graph G, one can define a metric space whose points are given by the vertices of G, and whose metric is taken to be the length of the shortest path between two vertices. The weighting vector of this metric space is not guaranteed to exist.

Definition 2. For an arbitrary subset $X \subseteq \mathbb{R}^n$, the magnitude of X is defined as

$$\mathrm{Mag}(X) = \sup\{\mathrm{Mag}(Y) \mid Y \text{ is a finite subset of } X\}.$$

Example. In 1 dimension, and for $t > 0$, one has that $\mathrm{Mag}([0,t]) = 1 + t/2$. This was shown by Leinster in (Leinster, 2013). The magnitude of the ball with radius r in $\mathbb{R}^{2n+1}$ is a rational function of r, and this was recently demonstrated by Barceló and Carbery (Barceló & Carbery, 2018).

For a finite metric space (X,d), and any $t \in [0,\infty]$, we can define a new metric space (tX,td) in the following way. The points of tX are the same as those of X, and the metric td is d scaled by t: $td(x,y) := t \cdot d(x,y)$. The *magnitude function of* X is the map $t \mapsto \mathrm{Mag}(tX)$, and it is well-defined whenever ζ_{tX} is invertible. Although the inverse of ζ_{tX} may not be defined in general, it has been shown in [Proposition 2.2.6 (Leinster, 2013)] that for finite subsets of $\mathbb{R}^n$, the magnitude function is analytic on $(0,\infty)$. We also have the following:

Theorem 3 (Proposition 2.2.6 (Leinster, 2013))**.** *For $X \subset \mathbb{R}^n$ finite, $\lim_{t\to\infty} \mathrm{Mag}(tX) = |X|$.*

The above proposition is one of the reasons underlying the informal interpretation of magnitude as quantifying the effective number of points in a space. The following very recent theorem gives a connection between the magnitude of $X \subset \mathbb{R}^n$ and the n-volume of X.

Theorem 4 (Theorem 1 (Barceló & Carbery, 2018))**.** *For $X \subset \mathbb{R}^n$ nonempty and compact, we have*

$$\lim_{t\to 0^+} \mathrm{Mag}(tX) = 1, \quad \textit{and}$$
$$\lim_{t\to\infty} \frac{\mathrm{Mag}(tX)}{t^n} = \frac{\mathrm{Vol}(X)}{n!\,\mathrm{Vol}(B_n)},$$

where $B_n \subset \mathbb{R}^n$ is the unit ball.

1.2. Properties of the weighting vector

The weighting vector plays a central role in the applications that are discussed below, but it is not obvious by inspection of its definition what useful information the individual entries of the weighting vector carries. To provide some intuition about this vector, we now highlight its key features. Our present aim is to convey a qualitative sense of things, so our focus is on numerical examples and basic facts. Note that the ability of the weighting vector to perform boundary detection is more than conjecture: it may be completely explained using harmonic analysis (Folland, 1999; Meckes, 2015). But for reasons of space and scope, we limit our focus.

Let $X \subset \mathbb{R}^n$ be a finite set and recall that the weighting vector of $X \subset \mathbb{R}^n$ is defined as $w_X := \zeta_X^{-1}\mathbb{1}$. This vector is related to the magnitude of X through $\mathrm{Mag}(X) = \mathbb{1}'\zeta_X^{-1}\mathbb{1} = \mathbb{1}'w_X$. Note that while $X \subset \mathbb{R}^n$, the vector $w_X \subset \mathbb{R}^{|X|}$, i.e., the dimension of the weighting vector is a function of the size of X, and not of the dimension n. Also note that the entries of w_X may be indexed in a canonical way by $x \in X$. We call $w_X(x)$, the weight of x.

Since all operations involved in evaluating w_X are continuous, the weighting vector of a small perturbation of X will approximate the weighting vector of X itself. More precisely, let $X_\epsilon := \{x + \epsilon\eta_x : x \in X, \|\eta_x\| \leqslant 1\}$. Then for all $x \in X$, one has $\lim_{\epsilon\to 0} w_{X_\epsilon}(x + \epsilon\eta_x) = w_X(x)$. Since the similarity matrix ζ_X is positive definite, its inverse exists and is also positive definite. Thus, $\mathbb{1}'\zeta_X^{-1}\mathbb{1} = \mathbb{1}'w_X > 0$. Although the average value of the entries of w_X is guaranteed to be positive, it may happen that $w_X(x) < 0$ holds for some $x \in X$. It is currently unknown what, if any, significance to ascribe to the sign of $w_X(x)$.

Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be an affine isometry, i.e., $f(x) = Qx + q$ for some $Q \in \mathbb{R}^{m\times n}$ and $q \in \mathbb{R}^m$, and f satisfies $\|z - w\| = \|f(z) - f(w)\|$ for all $z, w \in \mathbb{R}^n$. Since f is an isometry, the similarity matrices of X and $f(X)$ agree (i.e., $\zeta_X = \zeta_{f(X)}$), and as a result, $w_X(x) = w_{f(X)}(f(x))$. By the Mazur-Ulam theorem, all surjective isometries between normed spaces are necessarily affine. Thus, we have that the weighting vector is invariant under transformation by any surjective isometry. More concretely, in Figure 1, the weightings that are displayed are independent of the location and the orientation of the sets.

When $t > 0$ is large, the scaled space tX has the property that all points in it are far from each other. Thus, one has $\lim_{t\to\infty} \zeta_{tX} = \lim_{t\to\infty} \zeta_{tX}^{-1} = I$. By Theorem 3, the magnitude function satisfies $\lim_{t\to\infty} \mathrm{Mag}(tX) = |X|$. Combining these observations, we find that for all $x \in X$, $\lim_{t\to\infty} w_{tX}(tx) = 1$. The closely-related concept of a *scattered space* appears in prior work (Definition 2.1.2, (Leinster, 2013)), where under conditions far more general than considered here, it is shown that scattered spaces have well-defined magnitudes, and hence, weighting vectors.

Conversely, when $t > 0$ is small, each entry of the similarity matrix ζ_{tX} is close to 1. In particular, the limiting matrix is the rank-1 matrix $\mathbb{1}\mathbb{1}'$, which does not have an inverse. However, by Theorem 4, one has $\lim_{t\to 0^+} \mathbb{1}' w_{tX}(x) = 1$. Empirically, when $t > 0$ is very small, we find that the weights of "interior points" of the global space of X are small, while the "extreme" points of X—especially points that live nearest the convex hull of X—are significantly larger.

Finally, we consider weighting vectors of $X \subset \mathbb{R}^n$ that is neither too scattered nor especially concentrated about the origin. As one example, let $X \subset \mathbb{R}^n$ be a regular convex polytope. By a symmetry argument, for all vertices $x, y \in X$ one has $w_X(x) = w_X(y)$. Thus, modulo a normalizing constant, the weighting vector is completely specified. Next, consider Figure 1, which displays weightings of two sets that do not have any special symmetry: $X_0 \subset \mathbb{R}^2$ which consists of points supported in the union of four disjoint sets, and $X_1 \subset \mathbb{R}^3$ which lives on an embedding of the Möbius strip. Both sets were generated using a uniform random sampling process. In these renderings, every $x \in X_i$ has its weight, $w_{X_i}(x)$, conveyed both by the marker size and by color, where $i = 0, 1$. It is clear from these figures that points within the relative interior of some component have low weight, while points in close proximity to some boundary tend to have larger weight. It is this empirical observation that leads to the utility of weightings in applications.

We close this section by observing that magnitude, and by extension, weighting vectors, are well-defined on a very general class of sets, including sets that are not necessarily subsets of $\mathbb{R}^n$. It is therefore possible to extend the notion that connects a point's weight and its proximity to a boundary to *any* space that has a weighting vector.

1.3. Related work, paper structure

An early reference to the concept of magnitude occurs in (Solow & Polasky, 1994), where it was introduced as a way to measure biological diversity. However, the mathematical motivations were not divulged in this paper. Two decades later, Leinster (Leinster, 2013) placed the magnitude of a metric space within a formal mathematical framework using category theory. This highly abstract perspective lead to the current era, where it is being explored through many different lenses, including functional analysis (Meckes, 2013; Barceló & Carbery, 2018), harmonic analysis (Meckes, 2015) and homology theory (Leinster & Shulman, 2017), where it has been shown to be equivalent to an Euler characteristic. Much of the prior emphasis has been on a set's magnitude, and this focus has overshadowed the potential utility of the weighting vector.

Recently, topological data analysis has emerged as an approach to the problem of describing the shape of high-dimensional data (Edelsbrunner et al., 2002; Scopigno et al., 2004; Zomorodian & Carlsson, 2004). One particularly popular topic within this field is persistent homology (Edelsbrunner et al., 2002). Recent efforts have realized magnitude as the Euler characteristic of a homology theory, called magnitude homology (Leinster & Shulman, 2017). It has also been shown that there is a direct relationship between magnitude homology and persistent homology (Otter, 2018); however, the current paper is the first known application of magnitude directly to machine learning.

We now describe the remaining sections of this paper. Section 2 presents practical techniques for working with, and computing, the magnitude and weighting vectors of a discrete set. Section 3 introduces three algorithms that leverage the weighting vector in some essential way. The algorithms perform classification, active learning, and outlier detection. Section 4 presents results. We end with concluding remarks in Section 5.

2. 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.

2.1. Inclusion-Exclusion for Weight and Magnitude

In this section we investigate how to calculate the weighting vector for a set $Z = X \cup Y$ that is the union of two sets. Here X, Y, and Z are all finite subsets of $\mathbb{R}^n$. To approach

this, first we investigate the case when X and Y are disjoint. Then we will look at the case when $Y \subset X$, and show how to calculate either w_X or w_Y when one knows the other. Finally we will arrive at a corrected version of the inclusion-exclusion principle for magnitude, as well as the weighting vector.

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

Definition 5. Let $M := \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ 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

$$\zeta_X = \begin{bmatrix} \zeta_Y & \zeta_{Y,\bar{Y}} \\ \zeta_{Y,\bar{Y}}^T & \zeta_{\bar{Y}} \end{bmatrix}, \tag{1}$$

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 1 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

$$w_X\big|_Y = (\zeta_X/\zeta_{\bar{Y}})^{-1}(\mathbb{1}_Y - \zeta_{Y,\bar{Y}} w_{\bar{Y}}) \tag{2}$$

$$w_X\big|_{\bar{Y}} = (\zeta_X/\zeta_Y)^{-1}(\mathbb{1}_{\bar{Y}} - \zeta_{Y,\bar{Y}}^T w_Y), \tag{3}$$

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 2 and 3 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 6. For a block matrix $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$,

with A invertible, define

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

Now recall that for a block matrix M as in Definition 6, we have that

$$M^{-1} = \begin{bmatrix} A^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \rho_{MA}. \tag{4}$$

Definition 7. For $Y \subseteq X \subset \mathbb{R}^n$ finite sets, assume ζ_X is in block matrix format as in Equation 1. 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 8. *For finite sets $Y \subset X \subset \mathbb{R}^n$, let P_{XY} be a permutation matrix such that*

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

Then we have

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

Proof. This follows by setting $M = P_{XY}\zeta_X P_{XY}$, employing Equation 4, 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 a corrected inclusion-exclusion principle for weight vectors as well as magnitude.

Theorem 9. *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 8 to each subset considered, e.g.

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

2.2. 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 8. 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. This will be used in the sequel to perform more efficient inference of the machine learning classifier.

3. Algorithms

In this section we give details on how one may use weighting vectors and magnitude for a number of typical machine learning tasks.

3.1. Magnitude 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 $\mathbb{R}^n$, $x_i \in X \subset \mathbb{R}^n$, $i \in \{1, 2, ..., m\}$. 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 $l(x') \in L$.

Classification is fundamentally a task of finding or defining boundaries, thus because the weight vector can serve as a boundary detector, it is a natural fit for the task. In a classification task, we are working with finite sets of points X, so the term "boundary" is not well-defined in the mathematical sense. This prompts 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). Two points regarding our convention are worth mentioning. First, for convex sets, Definition 2 ensures that our convention matches with intuition on finite subsets that are sampled sufficiently densely, as the points with small weight all lie near the interior. Second, we can't distinguish between a point near the boundary of a set and one on the exterior of a set, as both will have relatively high weight. However, as discussed below, using our convention of interior points will be sufficient for use in a classification setting.

The weight of a point, and therefore our notion of interior points of a finite set captures global information, as it depends on all other points in the set. By changing other points in the set X, but leaving x_i fixed, it's weight w_i changes; the difference in the weight of a point relative to changes in the set is the key part of the classification algorithm.

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 $\{x'\} \cup X_i$. If weight vectors and inverse matrices for each X_i are computed in advance and cached, by the discussion in 2.2, each w_i' only requires matrix multiplication of order $|X_i|$ and inverting a 1×1 matrix. Intuitively, if w_i' has a low value, it likely is an interior point of X_i and therefore $l(x') = L_i$ 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 the classes are well-balanced and have similar underlying distributions, using the original metric space for each class is appropriate as the values of the w_i' will be similarly scaled and directly comparable. When the classes are imbalanced or have different underlying distributions, that assumption may not be appropriate, as the values of w_i' will not necessarily be comparable. We overcome this potential limitation by introducing a parameter t_i for each L_i that is used to scale distances when calculating w_i', that is we perform all operations related to L_i in the metric space $(t_i X, t_i d)$. Optimal t_i can be tuned during training for example using grid search and cross-validation. For simplicity and readability, we omit t from the basic version in algorithm 1.

We can further ensure consistency between the w_i' for different i by introducing 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. Taking w_j^i to be the weight of $x_j \in X_i$, some examples of possible functions are absolute value, $\text{SCALE}_i(w_i', \{w_j^i \mid x_j \in X_i\}) = |w_i'|$, and percentile $\text{SCALE}_i(w_i', \{w_j^i \mid x_j \in X_i\}) = \frac{\left|\{w_j^i | w_j^i <= w_i'\}\right|}{|X_i|}$.

We select a class label using a function DECIDE which operates on the w_i' after they have been scaled using SCALE_i. Letting $\text{S}_i(w_i')$ denote $\text{SCALE}_i(w_i', \{w_j^i \mid x_j \in X_i\})$, an example of DECIDE is argmax, $\text{SCALE}(\text{S}_1(w_1'), \text{S}_2(w_2'), ..., \text{S}_k(w_k')) = i$ where $\text{S}_i(w_i') > \text{S}_j(w_j')$ for all $j \neq i$ 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.

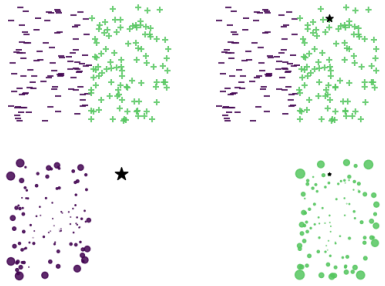

Figure 2. Upper left: Training data X, with $L_0 = -$ and $L_1 = +$. Upper right: $X \cup x'$, with a star denoting x'. Lower left: $\{x'\} \cup X_0$, with $w'_0 = 0.517$, and the sizes of markers indicate weight. Lower right: $\{x'\} \cup X_1$, with $w'_1 = 0.026$, and the sizes of markers indicate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

Algorithm 1 Classification via weighting vector

input Data set X, $L = \{L_1, L_2, ..., L_k\}$ labels, function DECIDE : $\mathbb{R}^k \rightarrow \{1, 2, ..., k\}$, function SCALE$_i$: $(\mathbb{R}, \mathbb{R}^{|X_i|}) \rightarrow \mathbb{R}$ 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 = SCALE$_i(w'_i, W_{X_i})$
    p.append(w)
  end for
  let j = DECIDE(p)
output L_j

3.2. Magnitude for active learning

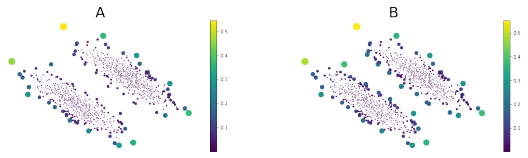

Figure 3. Magnitude for active learning example. A is weight of whole data set. B is weight of each class separately.

Next, we will describe how we can use magnitude and the weight vector to define a query strategy for an active learning algorithm. As stated in (Settles, 2009), "The key idea behind active learning is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns."

Approaches that minimize the number of human feedback needed to train machine learning models have sparked renewed interested due to the cost of labeling and the the fact that recent deep-learning-based approaches need handle large amounts of training data to achieve optimal performance. 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 (submit to an oracle for labeling) 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 below.

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 $w_{\tilde{X}_i}$
    $Q_{\min,i} = \arg\min_{\mathcal{U}} |w_{\tilde{X}_i}|$ for $i = 0, 1$
    $Q_{\max,i} = \arg\max_{\mathcal{U}} |w_{\tilde{X}_i}|$ for $i = 0, 1$
    $\mathcal{Y}_\mathcal{Q}$=query_labels($Q_{\min,0}$,$Q_{\max,0}$,$Q_{\min,1}$,$Q_{\max,1}$)
    $\mathcal{L} = \mathcal{L} \cup \{Q_{\min,0}, Q_{\max,0}, Q_{\min,1}, Q_{\max,1}\}$
    $\mathcal{Y}_\mathcal{L} = \mathcal{Y}_\mathcal{L} \cup \mathcal{Y}_\mathcal{Q}$
    $\mathcal{U} = X - \mathcal{L}$;
    $f = \mathbf{train\_classifier}(\mathcal{L}, \mathcal{Y}_\mathcal{L})$
  end while
output f

Where $|w_{\tilde{X}_i}|$ denotes all components of the vector $w_{\tilde{X}_i}$ in absolute value. We present some numerical experiments in Section 4.

3.3. Magnitude for Outlier Detection

In this section we give an algorithm that uses the values of the weight vector of a set to find outliers in a dataset. As we have seen, the weight vector serves as a boundary detector for a data set. But if the boundary is not well defined because there are outlier data points, we can use the weight to mark points as outliers. 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 $\gamma_{Xx} := \mathbb{1}^T \rho_{X \cup \{x\} X} \mathbb{1} = \text{Mag}(X \cup \{x\}) - \text{Mag}(X)$, we can see if adding x increased the magnitude substantially, thereby greatly extending the "border" of X. By Lemma 3.1.3 in (Leinster, 2013) we have that $0 \leqslant \gamma_{Xx}$.

Care must be taken, however; both the points on the boundary of the data set, and the outlier points will have high weight relative to the interior of the data set. Thus we collect all the points in X whose weight is below a threshold (here we take median weight plus 1.5 times standard deviation), and denote this subset as X_{in}, the *inliers*. The points of X not in X_{in} we call *outlier candidates*, and denote as X_{out}. Next, for each $x \in X_{out}$ with γ_{Xx} less than a user-defined threshold $0 \leqslant \tau$, we move from X_{out} to X_{in}. Then we have our final decomposition of the data set into inliers and outliers: $X = X_{in} \cup X_{out}$. We record this algorithm in Algorithm 3.

In Figure 4 we have the results of this algorithm using synthetic data. Inlier data was generated from two Gaussian distributions, and outliers were drawn from a uniform distribution.

Remark. It can be noted that the `NULL` class prediction algorithm described in Section 3.1 can be viewed as a type of online outlier detection algorithm. If the same paradigm is used when there is a single class, we obtain an outlier detection algorithm that is trained on data that only contains inliers.

Algorithm 3 Outlier detection via magnitude

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

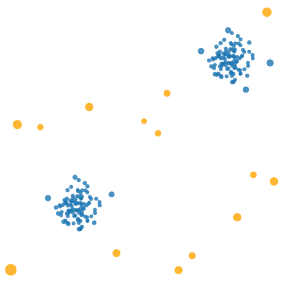

Figure 4. Outlier detection for synthetic data, $\tau = 0.2$

4. Results

4.1. 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 (Pedregosa et al., 2011) for all parameters in the other algorithms. Table 4.1 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.

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 4.1.

4.2. Active learning Experiments

In order to assess the effectiveness of the weighting-vector-based active learning (AL) algorithm proposed in Section 3.2, we compared Algorithm 3.2 to the simplest but highly effective and most commonly used query AL framework: uncertainty sampling (Lewis & Gale, 1994). In this framework, the AL algorithm queries the instances for which it

Table 1.

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

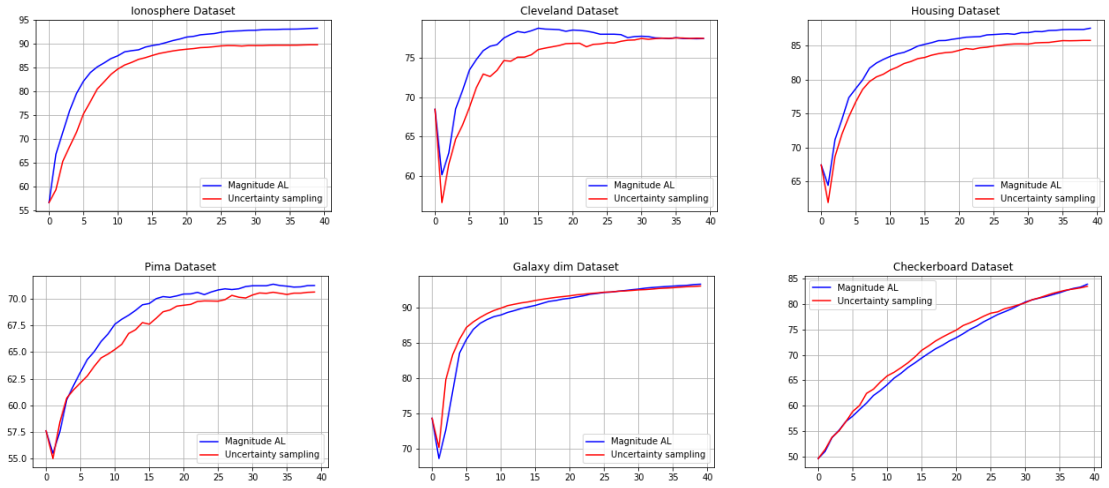

Figure 5. Active learning results comparing the weighting vector query strategy vs the uncertainty sampling strategy. Average over 100 runs

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

Table 2. Confusion matrix for classifier with `NULL` class.

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 (Cristianini & Shawe-Taylor, 2000) (also refer as to LS-SVM (Suykens & Vandewalle, 1999) or proximal SVM (Fung & Mangasarian, 2001)). 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 3.2 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-inspired algorithm chooses 4 points per iterations so we picked the four more uncertain points for the uncertainty sampling algorithm to be fair.

Figure 4.2 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.

5. 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 mag-

nitude 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.

References

Barceló, J. and Carbery, A. On the magnitudes of compact sets in Euclidean spaces. *American Journal of Mathematics*, 140(2):449–494, 2018. doi: 10.1353/ajm.2018.0011. URL https://muse.jhu.edu/article/688522.

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

Edelsbrunner, H., Letscher, D., and Zomorodian, A. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, Nov 2002. ISSN 1432-0444. doi: 10.1007/s00454-002-2885-2.

Folland, G. *Real analysis: modern techniques and their applications*. Pure and applied mathematics. Wiley, 1999. ISBN 9780471317166.

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

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

Leinster, T. and Shulman, M. Magnitude homology of enriched categories and metric spaces, 2017. URL https://arxiv.org/abs/1711.00802.

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

Meckes, M. Positive definite metric spaces. *Positivity*, 17: 733–757, Sept 2013. doi: 10.1007/s11117-012-0202-8.

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

Otter, N. Magnitude meets persistence. Homology theories for filtered simplicial sets, 2018. URL https://arxiv.org/abs/1807.01540.

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

Scopigno, R., Zorin, D., Carlsson, G., Zomorodian, A., Collins, A., and Guibas, L. Persistence barcodes for shapes, 2004.

Settles, B. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

Solow, A. R. and Polasky, S. Measuring biological diversity. *Environmental and Ecological Statistics*, 1(2):95–103, Jun 1994. ISSN 1573-3009. doi: 10.1007/BF02426650.

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

Zomorodian, A. and Carlsson, G. Computing persistent homology. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, pp. 347–356, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138857. doi: 10.1145/997817.997870.