Approximating Metric Magnitude of Point Sets

Rayna Andreeva[1], **James Ward**[1], **Primoz Skraba**[2], **Jie Gao**[3], **Rik Sarkar**[1]

[1]School of Informatics, University of Edinburgh
[2]Queen Mary University of London
[3]Rutgers University
r.andreeva@sms.ed.ac.uk, rsarkar@inf.ed.ac.uk

Abstract

Metric magnitude of a point cloud is a measure of its "size." It has been adapted to various mathematical contexts and recent work suggests that it can enhance machine learning and optimization algorithms. But its usability is limited due to the computational cost when the dataset is large or when the computation must be carried out repeatedly (e.g. in model training). In this paper, we study the magnitude computation problem, and show efficient ways of approximating it. We show that it can be cast as a convex optimization problem, but not as a submodular optimization. The paper describes two new algorithms – an iterative approximation algorithm that converges fast and is accurate in practice, and a subset selection method that makes the computation even faster. It has previously been proposed that the magnitude of model sequences generated during stochastic gradient descent is correlated to the generalization gap. Extension of this result using our more scalable algorithms shows that longer sequences bear higher correlations. We also describe new applications of magnitude in machine learning – as an effective regularizer for neural network training, and as a novel clustering criterion.

Code — https://github.com/rorondre/approx_magnitude
Extended version — https://arxiv.org/pdf/2409.04411

1 Introduction

Magnitude is a relatively new isometric invariant of metric spaces. It was introduced to characterize ecology and biodiversity data, and was initially defined as an Euler Characteristic of certain finite categories (Leinster 2008). Similar to quantities such as the cardinality of a point set, the dimension of vector spaces and Euler characteristic of topological spaces, Magnitude can be seen as measuring the "effective size" of mathematical objects. See Figure 1 for an intuition of Magnitude of Euclidean points. It has been defined, adapted and studied in many different contexts such as topology, finite metric spaces, compact metric spaces, graphs, and machine learning (Leinster 2013; Leinster and Willerton 2013; Barceló and Carbery 2018; Leinster 2019; Leinster and Shulman 2021; Kaneta and Yoshinaga 2021; Giusti and Menara 2024).

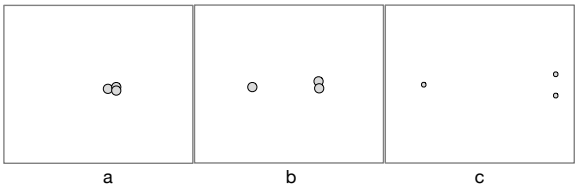


Figure 1: Consider the magnitude function of a 3-point space, visualized above at different scales. (a) For a small value of the scale parameter (e.g. $t = 0.0001$), all the three points are very close to each other and appear as a single unit. This space has magnitude close to 1. (b) At $t = 0.01$ the distance between the two points on the right is still small and they are clustered together, and the third point is farther away. This space has Magnitude close to 2 (c) When t is large, all the three points are distinct and far apart, and Magnitude is 3.

In machine learning and data sciences, the magnitude of a point cloud can provide useful information about the structure of data. It has recently been applied to study the boundary of a metric space (Bunch et al. 2020), edge detection for images (Adamer et al. 2024), diversity (Limbeck et al. 2024) and dimension (Andreeva et al. 2023) of sets of points in Euclidean space, with applications in data analysis as well as generalization of models (Andreeva et al. 2024). Wider applications of magnitude are limited by the computation cost. For a set of n points, the standard method of computing Magnitude requires inverting an $n \times n$ matrix. The best known lower bound for matrix multiplication and inversion is $\Omega(n^2 \log n)$ (Raz 2002); the commonly used Strassen's algorithm (Strassen 1969) has complexity $O(n^{2.81})$[1]. By definition, Magnitude computation requires consideration of all pairs of input points, making it expensive for large datasets.

Our contributions. In this paper, we take the approach that for many scenarios in data sciences, an approximate yet fast estimate of magnitude is useful, particularly in real-world

[1]Faster algorithms for matrix inversion exist, such as the Coppersmith-Winograd algorithm (Coppersmith and Winograd 1990) with running time $O(n^{2.376})$ and optimized CW-like algorithms with the best running time $O(n^{2.371552})$ (Williams et al. 2024).

modern applications where datasets and models are large and noisy and often require repeated computation.

Given a point set $X \subset \mathbb{R}^D$, we first show (Section 3.1) that computing the magnitude $\mathrm{Mag}(X)$ can formulated as finding the minimum of a convex function, and so can be approximated using suitable gradient descent methods. We then define a new algorithm that iteratively updates a set of weights called the Magnitude weighting to converge to the true answer (Section 3.2). This method converges quickly and is faster than matrix inversion or gradient descent.

While avoiding inversion, both these methods need $n \times n$ matrices to store and use all pairs of similarities between points. To improve upon this setup, we take an approach of selecting a smaller subset $S \subset X$ of representative points so that $\mathrm{Mag}(S)$ approximates $\mathrm{Mag}(X)$. We first prove that Magnitude is not a submodular function, that is, if we successively add points to S, $\mathrm{Mag}(S)$ does not satisfy the relevant diminishing returns property. In fact, for arbitrarily high dimension D, the increase in $\mathrm{Mag}(S)$ can be arbitrarily large with the addition of a single point. Though in the special case of $D = 1$, $\mathrm{Mag}(S)$ is in fact submodular, and the standard greedy algorithm (Nemhauser, Wolsey, and Fisher 1978) for submodular maximization can guarantee an approximation of $(1 - 1/e)$ (Section 3.3). In practice, the greedy algorithm is found to produce accurate approximations on all empirical datasets – both real-world ones and synthetic ones. This algorithm adds points to S one by one; in each step it iterates over all remaining points to find the one that maximizes $\mathrm{Mag}(S)$. These magnitude computations are faster due to the smaller size of S, but the costs add up as they are repeated over X.

Section 3.4 describes an approach to speed up the approximations further. It uses properties of Magnitude such as monotonicity and growth with scale, to develop a selection method – called Discrete centers – that does not require repeated computation of Magnitude. It is particularly useful for computing the *Magnitude Function* – which is magnitude as a function of scale, and useful in dimension computation (Andreeva et al. 2023). This method can also easily adapt to dynamic datasets where points are added or removed. Faster estimates of magnitude allows new applications of Magnitude in machine learning. Section 4 describes the use of Magnitude as a regularizer for neural network, and a clustering algorithm similar to density based clustering methods, using Magnitude as a clustering criterion.

Experiments in Section 5 show that the approximation methods are fast and accurate. Iterative Normalization outperforms inversion for larger dataset sizes and converges fast; for the subset selection algorithms, Discrete centers approximates the Greedy Maximization approach empirically at a fraction of the computational cost. The more scalable computation allows us to produce new results in the topic of generalization, where we extend prior work on computing topological complexities based on magnitude (Andreeva et al. 2024) to a larger number of training trajectories, extending from 5×10^3 due to computational limitations to 10^4, and observe that the correlation coefficients average Granulated Kendall ($\mathbf{\Psi}$) and Kendall tau (τ) improve significantly with the increased number of trajectories. The new regularization and clustering methods based on Magnitude are also shown to be effective in practice. Related work and discussion can be found in Sec. 6.

2 Technical Background

This section introduces the definitions needed for the rest of the paper.

2.1 Metric Magnitude

For a finite metric space (X, d) with distance function d, we define the similarity matrix $\zeta_{ij} = e^{-d_{ij}}$ for $i, j \in X$. The metric magnitude $\mathrm{Mag}(X, d)$ is defined (Leinster 2013) in terms of a *weighting* as follows.

Definition 1 (Weighting w)**.** *A weighting of (X, d) is a function $w : X \to \mathbb{R}$ such that $\forall i \in X, \sum_{j \in X} \zeta_{ij} w(j) = 1$.*

We refer to the $w(i)$ as the magnitude weight of i, and interchangeably write it as w_i.

Definition 2 (Metric Magnitude $\mathrm{Mag}(X, d)$)**.** *The magnitude of (X, d) is defined as $\mathrm{Mag}(X, d) = \sum_{i \in X} w(i)$, where w is a weighting of (X, d).*

The existence of a suitable w, and therefore magnitude of (X, d) is not guaranteed in general, but it exists for finite point clouds $X \subset \mathbb{R}^D$ with the ambient metric. In practice, metric magnitude is often computed by inverting the similarity matrix ζ and summing all the elements:

$$\mathrm{Mag}(X, d) = \sum_{ij} (\zeta^{-1})_{ij}. \tag{1}$$

Observe that when X is a finite subset of $\mathbb{R}^D$, then ζ is a symmetric positive definite matrix, and the inverse exists (Leinster 2013).

Magnitude is best illustrated when considering a few sample spaces with a small number of points. For example, with a single point a, ζ_X is a 1×1 matrix with $\zeta_X^{-1} = 1$ and $\mathrm{Mag}(X) = 1$. (When distance measure d is understood, such as in R^D, we often omit it in the notation.)

Example 1. *Consider the space of two points. Let $X = \{a, b\}$ be a finite metric space where $d(a, b) = d$. Then*

$$\zeta_X = \begin{bmatrix} 1 & e^{-d} \\ e^{-d} & 1 \end{bmatrix},$$

so that $\zeta_X^{-1} = \frac{1}{1 - e^{-2d}} \begin{bmatrix} 1 & -e^{-d} \\ -e^{-d} & 1 \end{bmatrix}$, and therefore

$$\mathrm{Mag}(X) = \frac{2 - 2e^{-d}}{1 - e^{-2d}} = \frac{2}{1 + e^{-d}}.$$

More information about a metric space can be obtained by looking at its rescaled counterparts. The resulting representation is richer, and is called the magnitude function, which we describe next.

2.2 Scaling and the Magnitude Function

For each value of a parameter $t \in \mathbb{R}^+$, we consider the space where the distances between points are scaled by t, often written as tX.

Definition 3 (Scaling and tX)**.** *Let (X, d) be a finite metric space. We define (tX, d_t) to be the metric space with the same points as X and the metric $d_t(x, y) = t \cdot d(x, y)$.*

Definition 4 (Magnitude function)**.** *The magnitude function of a finite metric space (X, d) is the function $t \mapsto \mathrm{Mag}(tX)$, which is defined for all $t \in (0, \infty)$.*

This concept is best illustrated by Figure 1.

Magnitude Function is important in computing magnitude dimension, which is determined by growth rate of $\mathrm{Mag}(tX)$ with respect to t. It is a quantity similar to fractal dimension and useful in predicting generalization of models computed via gradient descent (Andreeva et al. 2023).

2.3 Submodular Functions and Maximization Algorithm

The notion of submodularity is inspired by diminishing returns observed in many real world problems.

Definition 5 (Submodular Function)**.** *Given a set V, a function $f : 2^V \to \mathbb{R}$ is a submodular set function if:*

$$\forall S, T \subseteq V, f(S) + f(T) \geq f(S \cup T) + f(S \cap T).$$

The definition implies that marginal utility of items or subsets are smaller when they are added to larger sets. An example is with sensor or security camera coverage, where the marginal utility of a new camera is smaller then its own coverage area as its coverage overlaps with existing cameras.

The submodular maximization problem consists of finding a subset $S \subset V$ of a fixed size k that maximizes the function f. It shows up in various areas of machine learning, such as active learning, sensing, summarization, feature selection and many others. See (Krause and Golovin 2014) for a survey. The maximization problem is NP-hard, but can be approximated to within a factor of $1 - 1/e$ using a greedy algorithm (Nemhauser, Wolsey, and Fisher 1978).

3 Approximation Algorithms

We first examine algorithms that start with an arbitrary vector of weights for all points, and then iteratively adjusts them to approximate a Magnitude weighting. Then we describe methods that further improve efficiency by selecting a small subset of points that have magnitude close to that of X.

3.1 Convex Optimization Formulation and Gradient Descent

The problem of finding weights w can be formulated as a convex optimization using the squared loss:

$$\min_w \sum_i \left(\sum_j \zeta_{ij} w_j - 1 \right)^2 \tag{2}$$

This loss function is based on weighting (Definition 1), and reflects the error with respect to an ideal weighting where for each i, $\sum_j \zeta_{ij} w_j$ is 1.

This is a strongly convex optimization problem and can be addressed using methods suitable for such optimization, including gradient descent or stochastic gradient descent.

Algorithm 1: Iterative normalization algorithm for the approximation of magnitude

Input: The set of points X

Initialise $w_i = 1$ for all $i \in X$
while not converged **do**
  Compute $G(i) = \sum_j \zeta_{ij} w_j$ for all $i \in X$
  Update $w_i = w_i / G(i)$ for all $i \in X$
end while

3.2 Iterative Normalization Algorithm

In this section we present a different algorithm that we call the Iterative Normalization Algorithm. It starts with a weight vector of all ones. Then for every point i, it computes the sum $G(i) = \sum_j \zeta_{ij} w_j$. For a proper magnitude weighting every $G(i)$ should be 1, thus the algorithm simulates dividing by $G(i)$ to normalize the row to 1, and saves $w_i \leftarrow w_i / G(i)$. It does this in parallel for all rows (points).

Observe that compared to matrix inversion, which has a complexity of $O(n^{2.371552})$, the iterative normalization uses $O(n^2)$ per iteration, and achieves usable accuracy in relatively few iterations. In this problem, unlike usual optimization problems, we in fact know the optimum value of the loss for each point, and as a result we can use this approach of pushing the parameters toward this minimum value.

A caveat is that this algorithm produces a weighting that consists of positive weights. While individual magnitude weights can in principle be negative, Magnitude of a point cloud is always positive and in our experiments, the algorithm always finds a weighting whose sum converges toward the true magnitude. In this context, note that positive weights have been found to be relevant in predicting generalization of neural networks. See (Andreeva et al. 2024).

3.3 Approximation via Greedy Subset Selection

To approximate more efficiently, we can attempt to identify a subset S of points that approximate the magnitude of X. Magnitude increases monotonically with addition of points to S (Leinster 2013), which suggests approximation via algorithms that greedily add points to X similar to Nemhauser's submodular maximization (Nemhauser, Wolsey, and Fisher 1978). However, magnitude of a point set is not quite submodular, and thus the approximation guarantees do not carry over.

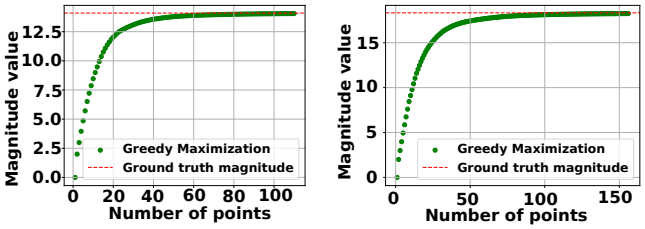
The non-submodularity can be seen in the following counterexample. Let $e_1, ..., e_D$ be the standard basis vectors of $\mathbb{R}^D$, so $e_1 = (1, 0, ..., 0)$ etc. Let $t > 0$ be a real number and $te_1, ..., te_D$ be the scaled basis vectors of $\mathbb{R}^D$, so $te_1 = (t, 0, 0..., 0)$ etc. Consider $X = \{te_1, -te_1, ..., te_D, -te_D\}$, with the usual metric. Thus X consists of the points on the axes of $\mathbb{R}^D$ that are a distance of t away from the origin. For a numerical example: when $t = 5$ and $D = 500$, we get $\mathrm{Mag}(X \cup \{0\}) - \mathrm{Mag}(X) \approx 7.18$. Thus, while the magnitude of a single point (origin) is 1 by itself, adding it to X produces an increase far greater than 1. This gap increases with increasing D.

This construction can be generalised to higher dimensions D, and behaves as follows in the limit:

Algorithm 2: Greedy algorithm for the computation of original magnitude

Input: The set of points S
Parameter: Tolerance k
Output: The approximated total magnitude and the maximising set S'

Initialise S' to be the empty set
Add a random element s_1 from S to S'.
while $\mathrm{Mag}(S') < (1-k) \times \mathrm{Mag}(S' \setminus s_i|)$ (The previous computation of magnitude is within the tolerance parameter) **do**
  Find the element s_i in $S \backslash S'$, maximising $\mathrm{Mag}(S' \cup s_i)$
  Add s_i to S'
end while
return $S', \mathrm{Mag}(S')$



(a) (b)

Figure 2: **Greedy algorithm approximates magnitude with small number of points.** Plot (a) shows magnitude approximation of a Gaussian blobs, 3 centers, with 500 points. Plot (b) shows Gaussian blobs with 3 clusters and 10^4 points.

Theorem 2. *Let $X = \{te_1, -te_1, ..., te_D, -te_D\}$ be a set of points in $\mathbb{R}^D$ as described above. Then in the limit:*

$$\lim_{D \to \infty} (\mathrm{Mag}(X \cup \{0\}) - \mathrm{Mag}(X)) = \frac{(e^t - e^{t\sqrt{2}})^2}{e^{2t} - e^{t\sqrt{2}}}.$$

Greedy set selection algorithm While the theorem above implies that submodularity does not hold in general for magnitude, our experiments suggest that in practice, Nemhauser's algorithm (Nemhauser, Wolsey, and Fisher 1978) adapted to Magnitude achieves approximation rapidly. A version of this idea can be seen in Algorithm 2.

In certrain restricted cases, submodularity can be shown:

Theorem 3. $\mathrm{Mag}(X)$ *is submodular when* $X \subset \mathbb{R}$.

Thus, when $X \subset \mathbb{R}$, the greedy approximation of $(1 - 1/e)$ holds.

3.4 Discrete Center Hierarchy Algorithm

The computational cost of the greedy algorithm arises from the need to repeatedly compute magnitude at each greedy step to examine $\Omega(n)$ points and compute magnitude each time to find the next point to add. To avoid this cost, we propose a faster approximation method.

In addition to being monotonically increasing with addition of points in X, the $\mathrm{Mag}(tX)$ also grows with t, and at the limit $\lim_{t \to \infty} \mathrm{Mag}(tX) = \#X$, where $\#X$ is the number of points in X (Leinster 2013). Therefore, point sets with larger distances between the points will have larger magnitude. Thus an iterative subset selection algorithm that prefers well-separated points is likely to increase the estimate faster toward the true magnitude.

Algorithm 3: Discrete Center Hierarchy construction

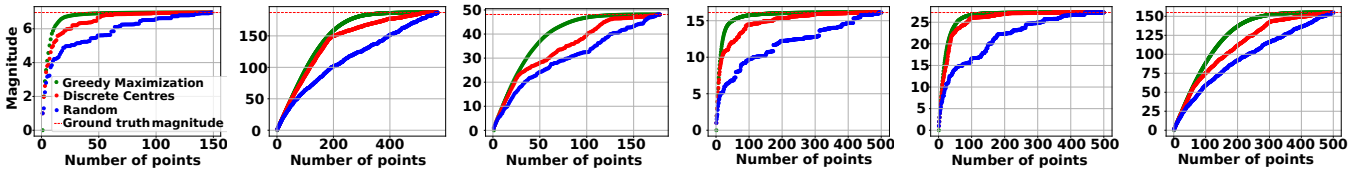
Input: (X, d).
$S_0 = X$
$S_i \leftarrow \emptyset$ for $i = 1, 2, ...$
for $i = 1, 2, ...$ **do**
  Select $S_i \subseteq S_{i-1}$ where S_i is a minimal independent covering set of S_{i-1} of radius 2^{i-1}
end for

This effect is achieved using Algorithm 3, which creates a hierarchy of discrete centers and uses them to successively approximate magnitude. The hierarchy is constructed as a sequence $S_0, S_1, S_2, \ldots$ of independent covering sets. Given a set S, a subset s is a minimal independent covering set of radius r, if it satisfies the following properties: (1): for every $x \in S$, there exists $y \in s$ such that $d(x, y) \leq r$ (2): $\forall x, y \in s, d(x, y) > r$ and (3) s is minimal with respect to these properties, that is, removing any point from s will violate the first property. With this in mind, we can construct the hierarchy as follows:

The hierarchy will have a height of at most $h = \log_2(\max_{x,y \in X} d(x, y))$, that is, log of the diameter of X. This hierarchy is used to successively approximate magnitude by traversing it from top to bottom. That is, starting from $s = \emptyset$, we first add points in S_h to s, followed by those in S_{h-1}, S_{h-2} etc, with $\mathrm{Mag}(s)$ increasing towards $\mathrm{Mag}(X)$.

Observe that when computing Magnitude function (Definition 4) which requires computation for several values of t, this same sequence can be used for approximations at all the scales. Experiments described later show that a small number of points in this sequence (from the top few levels) suffice to get a good approximation of magnitude.

Incremental updates to the hierarchy. This hierarchy can be efficiently updated to be consistent with addition or removal of points. When a new point q is added, we traverse top down in the hierarchy searching for the center in S_i within distance 2^{i-1} to q. When such a center does not exist, we insert q to be a center in S_j for all $j \leq i$. With a data structure that keeps all centers of S_i within distance $c \cdot 2^{i-1}$ for some constant c, we could implement efficient 'point location' such that insertion takes time proportional to the number of levels in the hierarchy. If a point q is deleted from the hierarchy, we need to delete q from bottom up. At each level i if there are centers of S_{i-1} within distance 2^{i-1} from q, some of them will be selectively 'promoted' to S_i to restore the property. For more detailed description of a similar geometric hierarchy, see (Gao, Guibas, and Nguyen 2006).



(a) (b) (c) (d) (e) (f)

Figure 3: **Discrete centers are close to Greedy Maximization at a fraction of the computational cost and better than random.** In plot (a) we have the Iris dataset, in plot (b) the Breast cancer dataset, in plot (c) the Wine dataset. In the remaining plots, we see subsamples of size 500 for popular image datasets: (d) MNIST, (e) CIFAR10 and (f) CIFAR100.

Algorithm 4: Magnitude Clusterer

Let X be a set of points (scaled so the average pairwise distance is 1) and $t \geq 0$ be some threshold.
Initialise $R = X \setminus \{a\}$ and $C = \{\{a\}\}$ for some random point a.
while $R \neq \varnothing$ **do**
  Initialise **best increase** = ∞ and **best point** = $\varnothing$, **best cluster** = $\varnothing$.
  for $b \in R$, $c \in C$ **do**
    Set **increase** = $\mathrm{Mag}(c \cup \{b\}) - \mathrm{Mag}(c)$
    if increase $<$ **best increase then**
      best increase = increase
      best point = b
      best cluster = c
    end if
  end for
  if increase $<$ t **then**
    Replace $c \in C$ with $c \cup \{$ **best point** $\}$.
  else
    Add $\{$ **best point** $\}$ to C.
  end if
  Remove **best point** from R.
end while
return C

4 Applications in Machine Learning

Here we describe the use of magnitude in two novel applications: as a regularization strategy for neural networks and for clustering.

4.1 Neural Network Regularization

Large neural network weights can be an indicator of overfitting to noise in the training data. Methods like weight decay add a term to the model's loss function to penalise large weights. We use Magnitude of the weights as a regulariser term. If the weight parameters are given by a vector p, where each $p_i \in \mathbb{R}$, then the magnitude of this metric space $(p, \mathbb{R})$ with the ambient metric of $\mathbb{R}$ is submodular (Theorem 3) with guaranteed approximation of $1 - 1/e$.

Specifically, we use the following algorithm to estimate magnitude. First select 1000 randomly chosen weights of the network, and then add the network weights with the smallest and largest values. As the set of the smallest and largest weights is the set of two weights with the largest possible magnitude, these points will be returned by the initial execution of the Greedy Maximization algorithm which, as magnitude is submodular on the real line, has a theoretical guarantee of performance. Then select a random subset of the remaining weights.

4.2 Clustering

Inspired by the greedy approximation algorithm for submodular set functions, we propose a novel magnitude-based clustering algorithm. The key idea behind this algorithm is that, given a pre-defined set of clusters, if a new point belongs to one of those clusters then its inclusion in the cluster should not cause the magnitude of the cluster to increase significantly. Thus the algorithm works as follows: In every round, the algorithm tries to find a point b that is coherent with an existing cluster c, where coherence is measured as the change in magnitude of c being below some threshold t when adding b to c. If no such point-cluster pair can be found, then the algorithm initializes b as a new cluster. The details are in Algorithm 4.

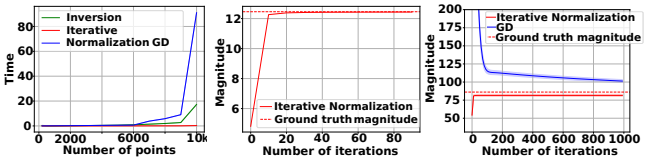
Good thresholds can be found by carrying out magnitude clustering over a range of threshold values and monitoring the number of clusters. The cluster counts that persist over a range of threshold values are likely to be represent natural clusterings of the data. Selecting the most persistent count is natural way to determine clustering without any other parameter.

5 Experiments

Experiments ran on a NVIDIA 2080Ti GPU with 11GB RAM and Intel Xeon Silver 4114 CPU. We use `PyTorch`'s GPU implementation for matrix inverison. The Gradient descent experiments used a learning rate of 0.005 and momentum of 0.9. Further comparisons with GD can be found in the extended version.

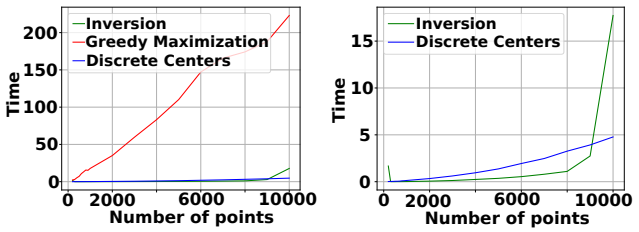
5.1 Accuracy and Computation Cost Comparison

Iterative algorithms In Figure 4 we see a comparison of the iterative algorithms on points sampled from $\mathcal{N}(0, 1)$ in $\mathbb{R}^2$ with 10^4 points. We observe that the Iterative Normalization algorithm is faster than Inversion and Gradient descent in plot (a) and it only needs a few iterations (less than 20) to converge as seen in plot (b), while Gradient descent takes a longer number of iterations. In plot (c) we see the convergence performance over 100 different runs, and again we



(a) (b) (c)

Figure 4: **Iterative algorithms comparison** Comparison of Inversion, Iterative Normalization and GD (a) Mean and standard deviation over 10 different runs, with 50 iterations of both iterative algorithms. (b) Number of iterations for convergence of Iterative Normalization for a randomly generated sample of 10000 points. (c) Iterative Normalization vs GD. Iterative Normalization converges fast, GD takes a longer number of iterations. 100 runs. Comparison on larger point sets can be found in the extended version.



(a) (b)

Figure 5: **Subset selection algorithms comparison** (a) Time taken for Inversion, Greedy Maximization and Discrete Centers to execute. (b) zoom on the performance of Inversion and Discrete Centers, and note that Discrete Centers performs better as the number of points increases. Comparison on larger datasets can be found in the extended version.

note that Iterative Normalization converges fast, while GD requires a larger number of iterations.

Subset selection algorithms Figure 5, shows a comparison of the subset selection algorithms on a randomly generated dataset with 10^4 points sampled from $\mathcal{N}(0,1)$ in $\mathbb{R}^2$.

Figure 3 shows the performance of the subset selection algorithms for a number of `scikit-learn` datasets (Iris, Breast Cancer, Wine) and for subsamples of MNIST, CIFAR10 (Krizhevsky, Nair, and Hinton 2014) and CIFAR100 (Krizhevsky 2009). We note that the Greedy Mazimization performs the best, but Discrete Centers produces a very similar hierarchy of points. Selecting points at Random does not lead to an improvement in a sense that you need to approach the cardinality of the set to get a good enough approximation of magnitude.

5.2 Applications in ML

Training trajectories and generalization It has been shown that Magnitude and a quantity derived from Magnitude called Positive magnitude (PMag), consisting of positive weights) are important in bounds of worst case generalization error. The method relies on computing a trajectory by taking n steps of mini-batch gradient descent after convergence, and computing the Magnitude of corresponding point set on the loss landscape. See (Andreeva et al. 2024) for details.

Metric	ψ_{lr}	ψ_{bs}	$\mathbf{\Psi}$	τ
Mag_{5000}	0.68	0.62	0.65	0.64
Mag_{7000}	0.71	0.77	0.74	0.69
Mag_{10000}	**0.75**	**0.82**	**0.79**	**0.74**
PMag_{5000}	0.91	0.67	0.79	0.85
PMag_{7000}	0.93	0.73	0.83	0.88
PMag_{10000}	**0.97**	**0.79**	**0.88**	**0.90**

Table 1: Generalization gap correlation improvement using an increasing number of points. τ is Kendall tau.

The experiments up to now have been limited to training trajectories of at most size 5×10^3 due to computational limitations. Our faster approximation methods can allow us to verify the results on larger trajectories.

We denote by Mag_{n} and PMag_{n} the relevant quantities trajectories of length n. Sizes upto 5000 have been considered in the original paper. We extend to sizes of 7000 and 10000. We use ViT (Touvron et al. 2021) on CIFAR10, ADAM optimizer (Kingma and Ba 2017), and perform the experiment over a grid of 6 different learning rates and 6 batch sizes, where the learning rate is in the range $[10^{-5}, 10^{-3}]$, and the batch size is between $[8, 256]$ resulting in 36 different experimental settings.

The results can be found in Table 1, showing a number of correlation coefficients relevant for generalization (Jiang et al. 2020) between generalization gap and Mag_{n} and PMag_{n} for $n = \{5000, 7000, 10000\}$. We use the granulated Kendall's coefficients (ψ_{lr} and ψ_{bs} are the granulated Kendall coefficient for the learning rate and for batch size respectively, and $\mathbf{\Psi}$ is the Average Kendall coefficient, which is the average of ψ_{lr} and ψ_{bs} (Jiang et al. 2020), which are more relevant than the classical Kendall's coefficient for capturing causal relationships.

We observe that all correlation coefficients improve with the increase of trajectory size. In particular, the Kendall tau coefficient and the Average Granulated Kendall coefficient increases by 0.14 for Mag_{10000} compared to Mag_{5000}, and by 0.09 for PMag_{10000}. Similarly, Kendall tau improves by 0.10 for Mag_{10000} and 0.05 for PMag_{10000}. This is an interesting result which needs to be investigated further for more models and datasets. Further visualisation results can be seen in Figure 6, where we see how the proposed quantities change with the generalization gap, and when more trajectories are considered.

Neural Network Regularization Utilising the magnitude approximation described in Section 4.1, we train five neural networks each with two fully connected hidden layers on the MNIST dataset for 2000 epochs, using cross entropy loss on MNIST. We train the models with a scalar multiple of the magnitude of the weights as a penalty term. One of the networks we train (with a regularization constant of 0) corresponds to an unregularised model. We then evaluate

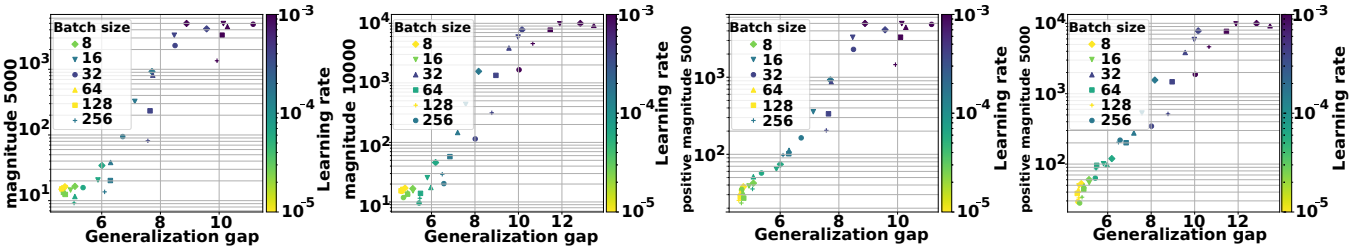


Figure 6: **Extended complexity measures vs. the generalization gap** We compare the original topological complexity measure Mag_{5000} (a) and PMag_{5000} (c) with the extended complexity measures Mag_{10000} (b) and PMag_{10000} (d) for a ViT trained on CIFAR10.

λ	Train. Loss	Test Loss	Gap	Magnitude
0	0.0021	0.0757	0.0736	1.5810
0.1	0.0041	0.0641	0.0600	1.1567
0.2	0.0061	0.0607	0.0546	1.1293
0.5	0.0103	0.0602	0.0499	1.0842
1	0.0167	0.0631	0.0464	1.0668

Table 2: Magnitude and performance of Neural Networks after training to minimise Training Loss(weights) + λMag(weights).

the differences in magnitude as well as train and test loss for each model. Our results are shown in Table 2. We first observe that as expected, adding a magnitude-based penalty term causes the network's magnitude to decrease. More interestingly magnitude regularization causes the neural network to perform better. This increase in performance occurs both in terms of test loss and generalization error, with the unregularised model recording both the largest test loss and generalization gap. It is also interesting to note that the generalization appears to increase consistently with the strength of regularization, whereas test loss appears to have an optimal strength of regularization at $\lambda = 0.5$.

Clustering The results of using the Clustering algorithms described in the previous section are presented in Figure 7. We note that our algorithm performs well, providing a better clustering than Agglomerative, k-means and DBSCAN.

6 Related work

The literature relevant to magnitude and machine learning has already been discussed in previous sections. We have studied closely the relation of magnitude to generalization (Andreeva et al. 2024, 2023). In other works diversity of latent representations have been recently explored in Limbeck et al. (2024). Magnitude based clustering has been suggested in O'Mally (2023). The algorithms proposed make use of a similar quantity called alpha magnitude (O'Malley, Kalisnik, and Otter 2023), but unlike ours, requires multiple parameters as input. Recent developments such as Magnitude for graphs (Leinster 2019) and relation between Magnitude and entropy (Chen and Vigneaux 2023) are also likely to be of interest in machine learning – as is the interpretation

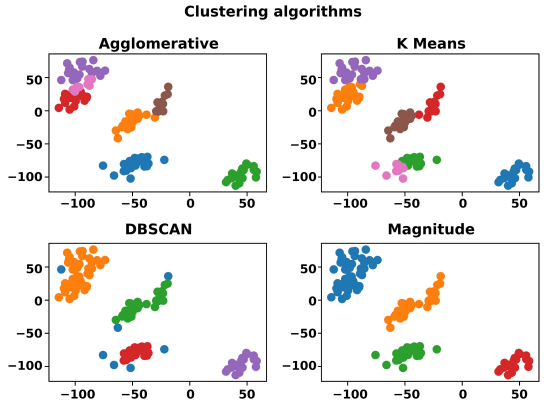


Figure 7: Results of applying the Magnitude clustering algorithm to an artificial dataset. We can see that the magnitude-based algorithm manages to find natural clusters and is able to determine a suitable number of clusters, whereas K-means and hierarchical clustering (Ward clustering in `scikit-learn`) need the user to determine this. A main difference between DBSCAN and the magnitude algorithm is in the treatment of outliers.

of magnitude as a dual of the Reproducing Kernel Hilbert Space (Meckes 2015).

The computational problem can be seen as solving the linear system $\zeta w = \mathbb{1}$, with $\mathbb{1}$ as a vector of all 1. Notice that our matrix ζ is symmetric positive definite but also a dense matrix. The iterative normalization algorithm we proposed bears resemblance to basis pursuit and other algorithms in compressive sensing (Foucart and Rauhut 2013), but the relation is not yet clear. Furthermore, obtaining approximation bounds relates to the 1-norm, making the extensive body of work on 2-norm approximation not applicable to this setting.

7 Conclusion

The fast and scalable methods for approximating metric magnitude should help greater application and exploration of magnitude in improving machine learning and our understanding of it. The novel applications to deep learning and clustering also can be explored further.

Acknowledgements

R.A. is supported by the United Kingdom Research and Innovation (grant EP/S02431X/1), UKRI Centre for Doctoral Training in Biomedical AI at the University of Edinburgh, School of Informatics and Helmholtz Visiting Researcher Grant. J.G. would like to thank funding support through NSF IIS-2229876, DMS-2220271, DMS-2311064, CCF-2208663, CCF-2118953. P.S. would like to acknowledge funding support from the ESPRC hub An "Erlangen Programme" for AI (grant EP/Y028872/1).

References

Adamer, M. F.; De Brouwer, E.; O'Bray, L.; and Rieck, B. 2024. The magnitude vector of images. *Journal of Applied and Computational Topology*, 1–27.

Andreeva, R.; Dupuis, B.; Sarkar, R.; Birdal, T.; and Simsekli, U. 2024. Topological Generalization Bounds for Discrete-Time Stochastic Optimization Algorithms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Andreeva, R.; Limbeck, K.; Rieck, B.; and Sarkar, R. 2023. Metric Space Magnitude and Generalisation in Neural Networks. In *Proceedings of 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML)*, volume 221 of *Proceedings of Machine Learning Research*, 242–253. PMLR.

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

Bunch, E.; Dickinson, D.; Kline, J.; and Fung, G. 2020. Weighting vectors for machine learning: numerical harmonic analysis applied to boundary detection. In *TDA & Beyond*.

Chen, S.; and Vigneaux, J. P. 2023. Categorical magnitude and entropy. In *International Conference on Geometric Science of Information*, 278–287. Springer.

Coppersmith, D.; and Winograd, S. 1990. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3): 251–280. Computational algebraic complexity editorial.

Foucart, S.; and Rauhut, H. 2013. *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Basel. ISBN 0817649476.

Gao, J.; Guibas, L. J.; and Nguyen, A. 2006. Deformable spanners and applications. *Comput. Geom.*, 35(1-2): 2–19.

Giusti, C.; and Menara, G. 2024. Eulerian magnitude homology: subgraph structure and random graphs. *arXiv preprint arXiv:2403.09248*.

Jiang, Y.; Neyshabur, B.; Mobahi, H.; Krishnan, D.; and Bengio, S. 2020. Fantastic Generalization Measures and Where to Find Them. In *International Conference on Learning Representations*.

Kaneta, R.; and Yoshinaga, M. 2021. Magnitude homology of metric spaces and order complexes. *Bulletin of the London Mathematical Society*, 53(3): 893–905.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Krause, A.; and Golovin, D. 2014. Submodular function maximization. *Tractability*, 3(71-104): 3.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images.

Krizhevsky, A.; Nair, V.; and Hinton, G. E. 2014. The Cifar-10 Dataset.

Leinster, T. 2008. The Euler characteristic of a category. *Documenta Mathematica*, 13: 21–49.

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

Leinster, T. 2019. The magnitude of a graph. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 166, 247–264. Cambridge University Press.

Leinster, T.; and Shulman, M. 2021. Magnitude homology of enriched categories and metric spaces. *Algebraic & Geometric Topology*, 21(5): 2175–2221.

Leinster, T.; and Willerton, S. 2013. On the asymptotic magnitude of subsets of Euclidean space. *Geometriae Dedicata*, 164: 287–310.

Limbeck, K.; Andreeva, R.; Sarkar, R.; and Rieck, B. 2024. Metric Space Magnitude for Evaluating the Diversity of Latent Representations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming*, 14: 265–294.

O'Malley, M.; Kalisnik, S.; and Otter, N. 2023. Alpha magnitude. *Journal of Pure and Applied Algebra*, 227(11): 107396.

O'Mally, M. 2023. *Magnitude, Alpha Magnitude, and Applications*. Ph.D. thesis, Wesleyan University.

Raz, R. 2002. On the complexity of matrix product. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 144–151.

Strassen, V. 1969. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4): 354–356.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, 10347–10357. PMLR.

Williams, V. V.; Xu, Y.; Xu, Z.; and Zhou, R. 2024. New Bounds for Matrix Multiplication: from Alpha to Omega. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 3792–3835.