



Article

Bregman–Hausdorff Divergence: Strengthening the Connections Between Computational Geometry and Machine Learning

Tuyen Pham ^{1,*} , Hana Dal Poz Kouřimská ² and Hubert Wagner ^{1,*}¹ Department of Mathematics, University of Florida, Gainesville, FL 32611, USA² Applied Geometry and Topology, University of Potsdam, Am Neuen Palais 10, 14469 Potsdam, Germany; hana.dal.poz.kourimska@uni-potsdam.de

* Correspondence: tuyen.pham@ufl.edu (T.P.); hwagner@ufl.edu (H.W.)

Abstract: The purpose of this paper is twofold. On a technical side, we propose an extension of the Hausdorff distance from metric spaces to spaces equipped with asymmetric distance measures. Specifically, we focus on extending it to the family of Bregman divergences, which includes the popular Kullback–Leibler divergence (also known as relative entropy). The resulting dissimilarity measure is called a Bregman–Hausdorff divergence and compares two collections of vectors—without assuming any pairing or alignment between their elements. We propose new algorithms for computing Bregman–Hausdorff divergences based on a recently developed Kd-tree data structure for nearest neighbor search with respect to Bregman divergences. The algorithms are surprisingly efficient even for large inputs with hundreds of dimensions. As a benchmark, we use the new divergence to compare two collections of probabilistic predictions produced by different machine learning models trained using the relative entropy loss. In addition to the introduction of this technical concept, we provide a survey. It outlines the basics of Bregman geometry, and motivated the Kullback–Leibler divergence using concepts from information theory. We also describe computational geometric algorithms that have been extended to this geometry, focusing on algorithms relevant for machine learning.

Keywords: computational geometry; machine learning; information theory; Shannon’s entropy; relative entropy; Bregman divergence; non-Euclidean geometry; Bregman geometry



Academic Editor: Andreas Holzinger

Received: 12 March 2025

Revised: 7 May 2025

Accepted: 13 May 2025

Published: 26 May 2025

Citation: Pham, T.; Dal Poz Kouřimská, H.; Wagner, H. Bregman–Hausdorff Divergence: Strengthening the Connections Between Computational Geometry and Machine Learning. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 48. <https://doi.org/10.3390/make7020048>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Various kinds of data—from sounds to images to text corpora—are routinely represented as finite sets of vectors. These vectors can be processed using a wide range of algorithms, often based on linear algebra. The intermediate representations as well as final outcomes of the data are, similarly, sets of vectors.

Conveniently, the above setup allows for an intuitive geometric interpretation. Indeed, it is usual to equip the vector space $\mathbb{R}^d$ in which such representations live with the Euclidean metric. Geometric objects in the geometry induced by this metric, such as the distance itself, balls and their intersections, bisectors, etc., help explain such algorithms in intuitive terms.

In recent years, however, other notions of distances have started to play an important role. One popular distance is the Kullback–Leibler divergence, often referred to as the *relative entropy*. A form of this divergence is the *cross-entropy* loss, commonly used for training deep learning models, in particular.

Compared to the once popular *mean squared error* loss (based on the Euclidean metric) the cross-entropy loss (based on the Kullback–Leibler divergence) provides significantly better performance. While the Kullback–Leibler divergence is often viewed as a distance

between probability vectors, it lacks standard features of a metric. In particular, it is typically non-symmetric and never satisfies the triangle inequality. As such, its behavior can be less intuitive.

It may therefore be surprising that the Kullback–Leibler divergence induces a well-behaved geometry. Moreover, there exists an infinite family of distance measures, so-called *Bregman divergences*, that induce similar geometries. The aforementioned Kullback–Leibler divergence is one of its most prominent members, along with the squared Euclidean distance.

There is a significant overlap between algorithms in machine learning and computational geometry. Nevertheless, computational geometry tends to focus on the Euclidean distance (and other metric distances). In contrast, the non-metric aspects of the Kullback–Leibler divergence (and other Bregman divergences) prevent computational geometry algorithms from working—at least at the first glance.

It turns out that several popular algorithms can be extended to the Bregman setting—despite the lack of symmetry and triangle inequality, which are often deemed crucial. While this is an ongoing direction, there have been efforts to extend popular algorithms to operate within this framework.

In the first part of the paper, we offer a geometric perspective on Bregman divergences. We hope this perspective will streamline further development and analysis of algorithms at the intersection of machine learning and computational geometry; in particular, in the context of data measured using relative entropy, such as probabilistic predictions returned by a classifier trained using cross-entropy.

In the second part, we develop a crucial geometric tool in the context of Bregman geometry. The idea is simple: where a Bregman divergence provides a comparison between two vectors, we propose a natural way of comparing two *sets* of vectors. This idea is analogous to the Hausdorff distance between two sets and we therefore call it a *Bregman–Hausdorff divergence*. Notably, this measurement does not rely on any pairing or alignment between elements of the sets, and the sets may differ in size. This contrasts with the computation of a classifier’s loss during training, where each prediction is compared with the corresponding correct label.

Interestingly, the lack of symmetry characteristic to Bregman divergences allows for several different definitions—we select three, guided by the geometric interpretation of the original Hausdorff distance. Additionally, we propose first algorithms for computing these new divergences. These algorithms are enabled by recent developments in Bregman nearest-neighbor search, and we experimentally show they are efficient in practice.

Our contribution extends the arsenal of tools capable of handling data living in Bregman geometries. One crucial example of such data is the set of probabilistic predictions of modern classifiers trained with the cross-entropy loss.

Paper outline. In the first part of this paper, we introduce concepts from information theory and a geometric interpretation for the relative entropy (Section 2). This interpretation connects the relative entropy to a larger family of distance measures known as Bregman divergences. After a brief introduction to this family and the geometry its members induce (Section 3), we explain why the asymmetry is a beneficial property in context for machine learning, and highlight computational tools that have been extended to this setting (Section 4).

The second part of this paper introduces three new measurements based on Bregman divergences. We provide definitions as well as interpretations in the context of comparing sets of discrete probability distributions (Section 5). We then provide efficient algorithms for these measurements (Section 6). In Section 7, we experimentally show that the new measurements can be efficiently computed in practical situations. In particular, we combine

the theory and tools from the two previous sections to provide quantitative ways to analyze and compare machine learning models. Section 8 concludes the paper.

2. Information Theory and Relative Entropy

We begin by highlighting certain concepts from information theory, with the goal of providing an interpretation of the relative entropy. We will use this interpretation to develop intuition for the geometry induced by the relative entropy in Section 2. (More details on information theory can be found in [1,2].) In particular, we emphasize the inherent asymmetry of the relative entropy, which will inform our decision to focus on asymmetric versions of the Hausdorff distance later. We also provide a geometric interpretation of relative entropy. This interpretation is shared among all Bregman divergences, and will be our focus afterwards.

Setup. We first set up a running example to guide us through each definition in this section. Let E_1, E_2, E_3 , and E_4 be events occurring with probabilities $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$, and $\frac{1}{8}$, respectively.

More generally, for d events we encode these probabilities as a probability vector or in other words a discrete probability distribution. Geometrically, the space of all such vectors is the $(d - 1)$ -dimensional open probability simplex contained in $\mathbb{R}^d$, namely

$$\Delta^{d-1} = \{x \in \mathbb{R}^d : \sum_{i=1}^d x_i = 1, 0 < x_i < 1\}.$$

Going back to our example, we now plan to transmit information on sequences of observed events. To this end, we first encode each event as a finite sequence of bits, called a *codeword*. We aim to minimize the expected length of a codeword with the restriction that sequences of codewords be uniquely decodable.

Consider Table 1, in which p_i is the probability of event E_i . The three rightmost columns provide three different codes.

Table 1. Example codes for a discrete probability distribution on four events.

	Probability	Code ₁	Code ₂	Code ₃
E_1	$1/2$	00	0	0
E_2	$1/4$	10	1	10
E_3	$1/8$	01	01	110
E_4	$1/8$	11	11	111

Given a code and a discrete probability distribution p , we can compute the expected code length for the transmission of information about the events. Specifically, the expected length, $\mathbb{E}[\ell_i]$, for each code i is as follows:

$$\begin{aligned}\mathbb{E}[\ell_1] &= 2 \times \frac{1}{2} + 2 \times \frac{1}{4} + 2 \times \frac{1}{8} + 2 \times \frac{1}{8} = 2, \\ \mathbb{E}[\ell_2] &= 1 \times \frac{1}{2} + 1 \times \frac{1}{4} + 2 \times \frac{1}{8} + 2 \times \frac{1}{8} = \frac{5}{4}, \\ \mathbb{E}[\ell_3] &= 1 \times \frac{1}{2} + 2 \times \frac{1}{4} + 3 \times \frac{1}{8} + 3 \times \frac{1}{8} = \frac{7}{4}.\end{aligned}$$

While Code₁ may be the most straightforward way to encode these four events, we can find a more optimized code. Although Code₂ has a shorter expected code length, it is not decodable. Indeed, the sequence 0111 can describe both $E_1E_2E_4$ and $E_3E_2E_2$. In contrast, Code₃ is decodable and has a shorter expected code length than Code₁.

Shannon's entropy. In his seminal paper [3], Shannon introduced a formula to compute the lower bound for the expected length of a code for a discrete probability distribution. Specifically, given a discrete probability distribution p , the Shannon's entropy is defined as

$$H(p) = \sum_i p_i \log_2 \frac{1}{p_i},$$

with $p_i \log_2 \frac{1}{p_i} = 0$ for $p_i = 0$.

Returning to our example: for the discrete probability distribution p from Table 1, $H(p) = \frac{7}{4} = \mathbb{E}[\ell_3]$. Thus, Code₃ in Table 1 is the optimal way to encode events $E_1, \dots, E_4$.

Cross-entropy. Suppose we (erroneously) assume that the probability of the events is given by a probability distribution q , while in reality it is p . How inefficient is the code optimized for q compared to the code optimized for the true distribution p ?

In this situation, we would assign longer codewords to less probable events, as measured by q . However, to compute the expected code length we must use the true probabilities of events, given by p .

The cross-entropy is an extension of Shannon's entropy that provides a lower bound on the length of such codes:

$$H(p, q) = \sum_i p_i \log_2 \frac{1}{q_i}.$$

In other words, **the cross-entropy gives the lower bound for the expected code length for events with probabilities represented by p , assuming they occur with probabilities represented by q .**

Given distributions p and q , their cross-entropy has a geometric interpretation. It is the approximation of $H(p)$ by the best affine approximation of H centered at q . Indeed, we can write the cross-entropy as

$$H(p, q) = H(q) + \langle \nabla H(q), p - q \rangle,$$

where $\langle \cdot, \cdot \rangle$ is the standard dot product.

Relative entropy. Relative entropy is the difference between cross-entropy and entropy, $H(p, q) - H(p)$. It therefore measures the expected *loss of coding efficiency* incurred by using the 'approximate' probability q instead of the 'true' probability p .

The relative entropy is often viewed as a distance measure between two discrete probability distributions. However, unlike proper metric distances, it is *generally not symmetric and does not satisfy the triangle inequality*. Given $p, q \in \Delta^{d-1} \subset \mathbb{R}^d$, we write

$$\begin{aligned} D_{KL}(p||q) &= H(p, q) - H(p) \\ &= \sum_{i=1}^d p_i \log_2 \frac{1}{q_i} - \sum_{i=1}^d p_i \log_2 \frac{1}{p_i} \\ &= \sum_{i=1}^d p_i \log_2 \frac{p_i}{q_i} \end{aligned}$$

to denote the relative entropy. We provide a further explanation of this notation at the end of this section, and expand on it in Section 3.

Usage. In machine learning models, relative entropy is often used as a loss function. Let us consider a multiclass classification task, with $X \subset \mathbb{R}^d$ being a data set and Y the collection of correct labels encoded as probability vectors. For a model M dependent on a parameter vector θ and $x \in X$, we write $M(x; \theta)$ for the probability vectors, which are

compared to the correct labels in Y . Then, minimizing $\sum_{(x,y) \in X \times Y} D_{KL}(y \| M(x; \theta))$ can be interpreted as penalizing predictions that are poor approximations of the true distribution. Relative entropy has also been used in the training of Variational Autoencoders [4]. Outside of machine learning, relative entropy has seen use in statistical physics [5] and information geometry [6].

Asymmetry. The asymmetry of relative entropy has a tangible interpretation. Let $p = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$ as in Table 1, and choose $q = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0)$. If we attempt to approximate p using q , then $D_{KL}(p \| q) = +\infty$. This value reflects the fact that E_4 is an impossible event assuming q , and therefore no code was prepared for it (along with all other events that occur with zero probability). Hence, there exists no codeword of finite length that we could use to encode this event.

If, on the other hand, we approximate q using p , then $D_{KL}(q \| p) \approx 0.415$. This value reflects the fact that while E_4 cannot occur, the coding efficiency is decreased by accounting for a codeword that is never used. (We remark that this asymmetry also occurs when the outputs of relative entropy are finite in both directions.)

Recall that $H(p)$ is Shannon's entropy of p , while the cross-entropy, $H(p, q)$, is an approximation of $H(p)$ based on an affine approximation at $H(q)$. Thus, the relative entropy can be interpreted as the vertical distance between the graph of this affine approximation and the graph of $H(p)$. See Figure 1 for illustration.

This geometric construction of relative entropy can be generalized. Indeed, the family of distance measures arising from this type of construction is known as Bregman divergences [7]. Like relative entropy, these distance measures generally lack symmetry and do not fulfill triangle inequality. The relative entropy is a member of this family and is often referred to as the Kullback–Leibler (KL) divergence in this context. We will introduce this family next.

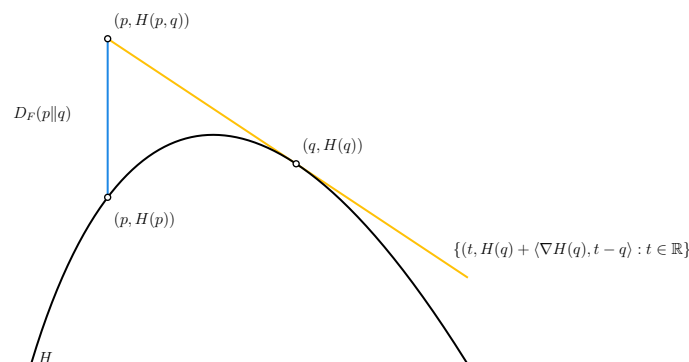


Figure 1. A visualization of the relative entropy.

3. Background on Bregman Geometry

In this section, we introduce Bregman divergences [7], portray the KL divergence as an important instance, and give an overview of Bregman geometry. We will put emphasis on the geometry induced by the KL divergence.

Bregman divergences—definition and basic properties. Each Bregman divergence is generated by a function of Legendre type. Given an open convex set $\Omega \subseteq \mathbb{R}^d$, a function $F : \Omega \rightarrow \mathbb{R}$ is of **Legendre type** [8,9] if F is as follows:

1. differentiable;
2. strictly convex;
3. the boundary $\partial\Omega$ of Ω is nonempty, then $\lim_{x \rightarrow \partial\Omega} \|\nabla F(x)\| = \infty$.

The last condition is often omitted, but it enables a correct application of the *Legendre transformation*. It is a useful tool coming from convex optimization [9], which we will review later.

Given a function F of Legendre type, the **Bregman divergence** [7] generated by F is the function

$$D_F : \Omega \times \Omega \rightarrow [0, \infty], \quad D_F(x||y) = F(x) - (F(y) + \langle \nabla F(y), x - y \rangle).$$

In other words, the divergence in the **direction** from x to y is the difference between $F(x)$ and the best affine approximation of F at y , evaluated at x . We illustrate this in Figure 2. The construction also mirrors the geometric interpretation of the *KL divergence* from Figure 1.

Just like metrics, Bregman divergences are always non-negative: in fact, $D_F(x||y) \geq 0$, with equality if and only if $x = y$. Unlike metrics, however, they are generally not symmetric, and do not satisfy the triangle inequality. To emphasize the lack of these two properties, we refrain from referring to the output of $D_F(x||y)$ as a ‘distance’. Similarly, to emphasize the lack of symmetry, it is customary to write $D_F(x||y)$ and not simply $D_F(x, y)$ for the divergence computed from x to y .

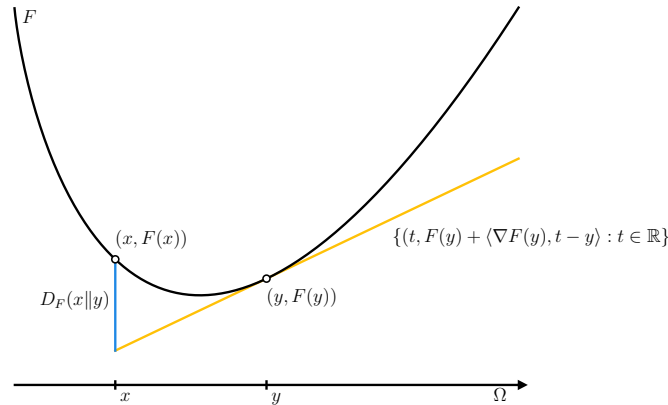


Figure 2. Visualization of a Bregman divergence formula for a one-dimensional domain.

Relative entropy as a Bregman divergence. Given the function $F : \mathbb{R}_+^d \rightarrow \mathbb{R}$, $F(x) = -E(x) = \sum_i x_i \log_2 x_i$, we have $\nabla F(x) = \frac{1}{\log(2)}(1 + \log x_1, \dots, 1 + \log x_d)$. Substituting in the formula for the Bregman divergence, we obtain

$$\begin{aligned} D_{KL}(x||y) &= D_{-E}(x||y) \\ &= \sum_{i=1}^d x_i \log_2 x_i - \sum_{i=1}^d y_i \log_2 y_i - \sum_{i=1}^d \frac{1}{\log(2)}(1 + \log y_i)(x_i - y_i) \\ &= \sum_{i=1}^d x_i \log_2 x_i - x_i \log_2 y_i + \frac{y_i - x_i}{\log(2)}. \end{aligned}$$

This divergence is often called a **generalized Kullback–Leibler divergence**. Restricting it to the probability simplex, Δ^{d-1} , we obtain a divergence that coincides with the relative entropy defined in the previous section. Following the statistics and information theory literature, we will refer to it as the Kullback–Leibler divergence and denote it by D_{KL} . One can easily check that the generator of this divergence is indeed a function of Legendre type, also when restricted to the probability simplex.

Other Bregman divergences. Among other examples of Bregman divergences, the most prominent one is the **squared Euclidean distance (SE)**, generated by the square of the Euclidean norm:

$$\mathbb{R}^d \rightarrow \mathbb{R}, \quad x \mapsto \sum_{i=1}^d x_i^2,$$

$$D_{SE}(x\|y) = \sum_{i=1}^d (x_i - y_i)^2.$$

We remark that restricting the domain to any bounded subset of $\mathbb{R}^d$ violates Condition 3 for a Legendre-type function. Consequently, under such a restriction the square of the Euclidean distance does not fulfill the above definition of a Bregman divergence.

The **Itakura–Saito (IS) divergence** [10], generated by the function

$$\mathbb{R}_+^d \rightarrow \mathbb{R}, \quad x \mapsto -\sum_{i=1}^d \log x_i,$$

$$D_{IS}(x\|y) = \sum_{i=1}^d \frac{x_i}{y_i} - \log \frac{x_i}{y_i} - 1,$$

has seen success as a loss function in machine learning models analyzing speech and sound data [11].

All of the above divergences are often classified as **decomposable Bregman divergences**, since each of them decomposes as a sum of 1-dimensional divergences.

The **Mahalanobis divergence** [12] is commonly used in statistics as a distance notion between probability distributions [13]. It is generated by the convex quadratic form

$$\begin{aligned} \mathbb{R}^d &\rightarrow \mathbb{R}, & x &\mapsto x^\top Q x, \\ \mathbb{R}^d \times \mathbb{R}^d &\rightarrow [0, \infty], & (x, y) &\mapsto (x - y)^\top Q (x - y), \end{aligned}$$

where Q is the inverse of a variance–covariance matrix. Unlike the other examples, it is generally not a decomposable divergence. The Mahalanobis divergence has also seen success in machine learning—examples include supervised clustering [14] and the classification of hyperspectral images [15].

Bregman Geometry. Similarly to a metric, each Bregman divergence induces a geometry. We provide a brief overview of some key objects and features of Bregman geometries, and provide their information–theoretic interpretations in the case of the KL divergence.

A fundamental object in geometry is the ball. Due to the asymmetry of Bregman divergences, one can define two types of Bregman balls [16]. The **primal Bregman ball** of radius $r \geq 0$ centered at the point $q \in \Omega$, is defined as

$$B_F(q; r) = \{y \in \Omega : D_F(q\|y) \leq r\},$$

and is the collection of those points whose divergence *from* q does not exceed r . See Figure 3 for various illustrations. Primal Bregman balls have a particularly nice geometric interpretation: given a light source at point $(q, F(q) - r)$, the primal ball $B_F(q; r)$ is the *illuminated* part of the graph of F , projected vertically onto Ω . We illustrate this in Figure 4, on the left.

The **dual Bregman ball** of radius $r \geq 0$ centered at the point $q \in \Omega$, is defined as

$$B'_F(q; r) = \{y \in \Omega : D_F(y\|q) \leq r\},$$

and is the collection of points whose divergence *to* q does not exceed r . The dual ball has a geometric interpretation similar to the primal ball. We first shift the tangent plane of the graph of F at $(q, F(q))$ up by r . The dual Bregman ball $B'_F(q; r)$ is the portion of the graph of F below this plane, projected vertically onto Ω . We illustrate this in Figure 4, on the right.

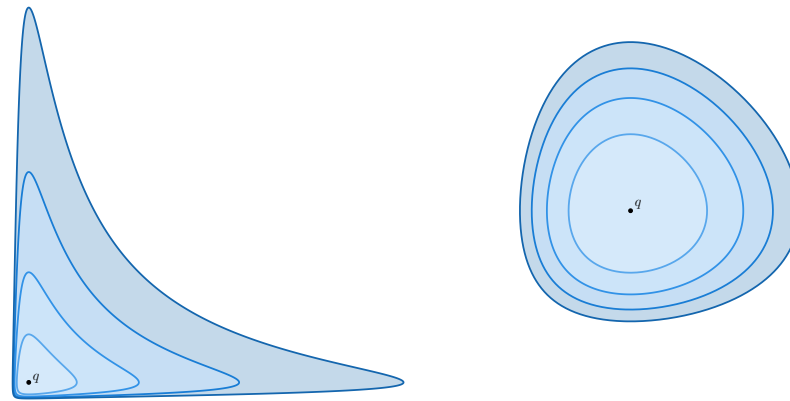


Figure 3. Left: concentric primal Itakura–Saito balls. Right: concentric primal generalized Kullback–Leibler balls.

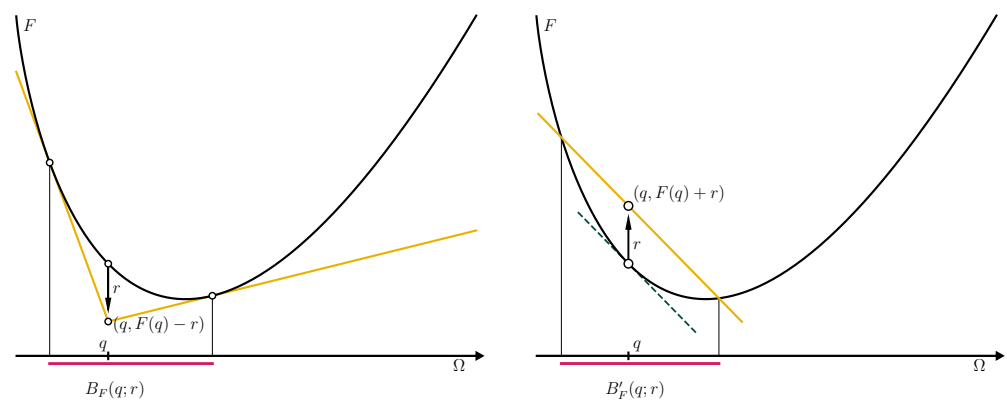


Figure 4. Geometric interpretation of primal (left) and dual (right) Bregman balls in dimension one.

As seen in Figure 3, and observed in [17], primal Bregman balls can be non-convex when viewed as a subset of Euclidean space. In contrast, dual balls are always convex since $D_F(x||y)$ is convex in x .

It may be tempting to consider the two geometries induced by a Bregman divergence to and from a point, separately. However, there is a strong connection between the two, given by the Legendre transformation [9].

Legendre transform. The **Legendre transform** F^* of a function F of Legendre type is defined on a conjugate domain

$$\Omega^* = \{\nabla F(x) \mid x \in \Omega\}$$

as

$$F^*(x^*) = \sup_{x \in \Omega} (\langle x, x^* \rangle - F(x)).$$

This construction induces a canonical map

$$\Omega \rightarrow \Omega^*, \quad x \mapsto x^* = \nabla F(x).$$

It turns out that F^* is a convex function of Legendre type, and thus it also generates a Bregman divergence, D_{F^*} . This divergence satisfies

$$D_{F^*}(p^*, q^*) = D_F(q, p),$$

implying that the function mentioned above maps primal balls in Ω to dual balls in Ω^* [17].

Chernoff point. Another connection between the two geometries is given by the **Chernoff point** [18]. Following [19], we say a set $X \subset \Omega$ has an **enclosing sphere**, if there exists a dual Bregman ball containing X . The enclosing sphere is then the boundary of this ball.

Moreover, every finite set $X \subset \Omega$ has a unique smallest enclosing Bregman sphere [16,19,20]. The center of this smallest enclosing sphere is known as the Chernoff point for X .

We are interested in a simple situation in which the set X consists of two points only, $X = \{p, q\}$. In this case, the Chernoff point minimizes the divergence $D_F(p\|c)$ subject to $D_F(p\|c) = D_F(q\|c)$; one can view it as lying midway between p and q with respect to the chosen Bregman divergence. Indeed, in the squared Euclidean case, it is the usual midpoint (arithmetic mean), namely $\frac{p+q}{2}$. However, for other Bregman divergences this point will generally not be the midpoint, but some other point on the segment joining the two points. We also remark that the midpoint does play a special role for all Bregman divergences. We elaborate on this when discussing the Bregman k-means clustering algorithm. Additionally, the popular Jensen–Shannon divergence, and its variants, rely on the midpoint.

The Chernoff point can be visualized: for each point $p \in X$, consider the primal ball $B_F(p; r)$ growing about p , parameterized by $r \geq 0$. Then, the Chernoff point is the point where all the balls $B_F(p; r)$ intersect for the first time. In the case when $X = \{p, q\}$, let us denote this radius by r^* , and the Chernoff point by c . Then, c lies on the boundary of both $B_F(p; r^*)$ and $B_F(q; r^*)$, and p and q lie on the boundary of $B'_F(c; r^*)$. A visualization of this interaction can be seen in Figure 5.

We refer to works by Nielsen [18,21] for more information on Chernoff points and their applications.

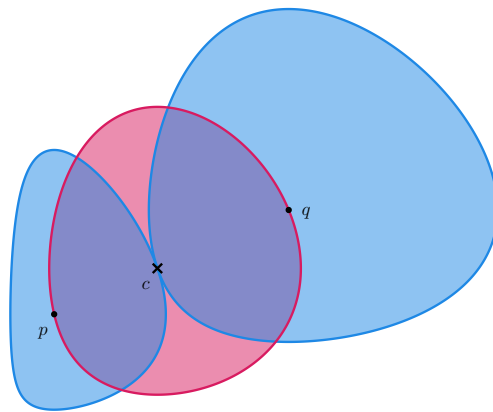


Figure 5. In blue (light): Primal KL balls with centers x and y intersect at the Chernoff point c . In magenta (dark): A dual KL ball of radius $D_{KL}(x\|c)$ is drawn about c .

Information-theoretic interpretation of the geometric objects stemming from the KL divergence. In the language of information theory, the primal and dual KL balls have the following interpretation: Let $p \in \Delta^{d-1}$ be a probability vector, and $r \geq 0$ a radius, expressed in bits. Note that, since the KL divergence measures the *expected* efficiency loss in bits, the radius r can indeed be of any (non-negative) real value, and is not limited to being an integer. The primal ball $B_{KL}(p; r)$ contains all probability vectors q that can approximate p with an expected loss of at most r bits. In contrast, the dual ball $B'_{KL}(p; r)$ contains all probability vectors q that can be approximated by p with an expected loss of at most r bits. Consequently, the Chernoff point for two probability vectors p and q is the

vector that approximates *both* p and q with the least loss of expected efficiency (as usual counted in bits).

4. Algorithms in Bregman Geometry

The development of algorithms for Bregman divergences is relatively young. We survey computational geometry algorithms that were adapted to the Bregman setting from the common Euclidean and metric settings, and highlight the necessary modifications.

k -means clustering. The first algorithm to be adapted was the k -means clustering algorithm. It was originally proposed by Lloyd [22] in 1957, and worked with the Euclidean distance. It was extended to arbitrary Bregman divergences by Banerjee and collaborators [23] in 2005.

In Euclidean space, the k -means clustering partitions a data set into k clusters. Each cluster is identified by a unique point, called a cluster centroid. Here, the centroid is chosen as the arithmetic mean of all the data points in the cluster. This choice is motivated by the fact that the mean is the unique point that minimizes the sum of the squares of the Euclidean distances (i.e., the variance) to all data points in a given cluster.

Surprisingly, the k -means algorithm still works when the square of the Euclidean distance is replaced with a Bregman divergence computed from data points. Indeed, Banerjee and collaborators show that for $X = \{x_i\}_{i=1,2,\dots,d} \subset \Omega$, the sum $\sum_{i=1}^d D_F(x_i \| p)$ is uniquely minimized by point p being the arithmetic mean of X —independently of the choice of a Bregman divergence. Consequently, apart from the requirement of computing the divergence towards the centroid, the original algorithm works in the Bregman setting without modification. This remarkable fact was the first hint that other geometric algorithms may work in the Bregman setting. (See Figure 6 for a comparison of k -means clustering in different geometries.)

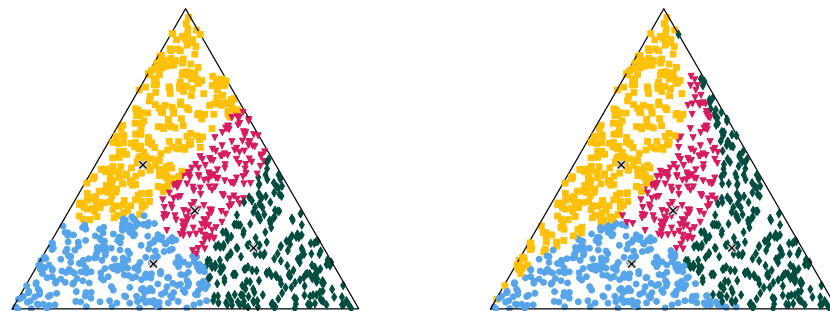


Figure 6. Clustering assignments for k -means with four centroids on Δ^2 . **Left** is with squared Euclidean distance, **right** is with KL divergence. Centroids are denoted by $\times$'s.

Voronoi diagrams. Voronoi diagrams were first formally defined for two- and three-dimensional Euclidean space by Dirichlet [24] in 1850, and for general Euclidean spaces by Voronoi [25,26] in 1908. Given a finite set $S \subset \mathbb{R}^d$ of points, called **sites**, we partition the space $\mathbb{R}^d$ into **Voronoi cells**, such that the Voronoi cell of $s \in S$ contains all points in the space for which s is the closest site. More formally, $\{x \in \mathbb{R}^d : d(x, s) \leq d(x, y) \forall y \in S\}$.

This definition for Voronoi diagrams was extended to the Bregman setting by Boissonnat, Nielsen, and Nock [17] in 2007. In the Euclidean space, Voronoi cells are convex polytopes (possibly unbounded). However, when another distance measure is used, the shape of these cells can change drastically. See Figure 7 for an illustration of Voronoi diagrams for various distance measures.

In the Euclidean case, the bisector of a pair of sites is a hyperplane, and the Voronoi cells arise as intersections of the resulting half-spaces. One method of computing these cells is by Chazelle's half-space intersection algorithm [27]. In the Bregman version, Boissonnat, Nielsen, and Nock show that calculating the divergence to the sites yields Bregman bisectors

that are also hyperplanes. Chazelle’s algorithm can therefore be used to compute the Bregman Voronoi diagram. However, when computing the divergence *from* a site, the Bregman bisectors are more general hypersurfaces. Consequently, the Bregman Voronoi cells may have curved faces. As a side note, the Legendre transform can be used to handle this harder scenario by mapping the input to the conjugate space, performing the computations there using hyperplanes, and mapping the result back.

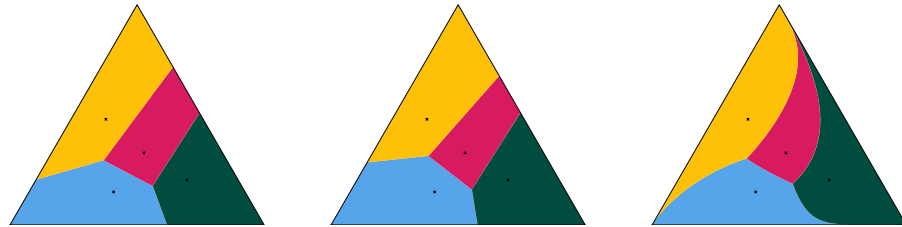


Figure 7. Voronoi diagrams on Δ^2 fixed sites S , each marked with x . The **left** is computed with the Euclidean distance. The **middle** and **right** images compute nearest site with respect to the KL divergence, the **middle** to each site and the **right** from each site.

k -Nearest neighbor search. In a metric space (M, d) , given a finite subset $S \subset M$ and a query $q \in M$, a k -nearest neighbor search finds the k elements of S closest to q . A common strategy to answer these queries is to spatially decompose the set S . This decomposition is often encoded as a tree data structure. This type of data structures includes ball trees [28], vantage point trees [29], and Kd-trees [30]. Collectively, these are referred to as metric trees—although many of them can be extended to the non-metric setting of Bregman divergences.

In the Bregman setting, due to the asymmetry, we consider two types of nearest neighbor search. Namely, $\arg \min_{s \in S} D_F(q||s)$ and $\arg \min_{s \in S} D_F(s||q)$. Queries are performed by searching subtrees while updating a nearest neighbor candidate. Subtrees may be ignored (pruned) to accelerate the search if certain conditions, which depend on the type of tree, are met.

In 2008, Cayton extended the concept of ball trees from metric spaces to the Bregman setting, and created software for Bregman ball trees [31]. The software works for the squared Euclidean distance and the KL divergence, with experimental support for the IS divergence. Cayton’s Bregman ball trees are constructed with the help the aforementioned Bregman k -means clustering. The pruning decision involves a projection of a candidate point onto the surface of the Bregman ball. This is performed via a bisection search.

Nielsen, Piro, and Barlaud improved on Cayton’s ball tree algorithm [32]. In particular, they improved the construction by altering the initial points for the Bregman k -means algorithm, and introduced a branching factor to allow for more splits at each internal node. They also implemented a priority-based traversal instead of the standard recursive ball tree traversal. The same authors also adapted another data structure called a **vantage point tree** in 2009 [32,33]. Specifically, they replaced the metric ball with a dual Bregman ball, and in the pruning decision, they performed a bisection search to check intersections of Bregman balls.

Kd-trees were introduced by Bentley [30] in 1975, and extended to Bregman divergences by Pham and Wagner [34] in 2025. Unlike ball trees and vantage point trees, the construction of the Bregman Kd-tree is independent of the Bregman divergence. Indeed, the choice of divergence can be deferred to the time of performing each query. The query algorithm, surprisingly, is the same as in the Euclidean case. For decomposable Bregman divergences in particular, it allows each pruning decision to be made in effectively $O(1)$ time, independently on the dimension of the data.

Cayton’s Bregman ball trees algorithm is specialized for the KL divergence, with experimental implementation for the IS divergence. Nielsen, Piro, and Barlaud’s implementations show results for the KL and IS divergences. However, for both Bregman ball trees and vantage point trees, adding implementations for new divergences is nontrivial. In contrast, although Kd-trees work for a subfamily of Bregman divergences, a further extension of the implementation to new Bregman divergences is straightforward.

Apart from the data structures and algorithms described above, other exact searches have been extended to work in the Bregman setting. These include R-trees and VA-files (extended by Zhang and collaborators [35] in 2009), and BrePartition (introduced by Song, Gu, Zhang, and Yu [36] in 2020). The listed algorithms provide exact nearest neighbor queries in the Bregman setting [37]. We remark that there exist other nearest neighbor algorithms that work in the Bregman setting [37,38]. However, these algorithms focus on approximations, without guarantees to find the exact nearest neighbors.

Computational topology. Topological concepts, such as homology groups, have been imported into computational geometry. One key concept is **persistent homology** [39], a stable geometric–topological descriptor of data, including point cloud data. It is the basis of the field called **topological data analysis** [40].

Building on geometric results of Boissonnat, Nielsen, and Nock [17], Edelsbrunner and Wagner [41] extended concepts of computational topology to the Bregman setting in 2017. In particular, basic concepts that allow for computation of persistent homology were extended to the Bregman setting. These include generalizations of the alpha and Čech constructions [42]. One key result is the proof of contractibility of nonempty intersection of Bregman balls. Intuitively speaking, this result ensures that these constructions correctly capture the topology of data. More recent work focuses on implementation and experimental aspects [43].

The development of computational tools for Bregman divergences motivates the extension of other geometric concepts from metric spaces to the Bregman setting. In the following sections we concentrate on one of the most commonly used—the *Hausdorff distance*. In short, we aim to compare *two sets of vectors* embedded in a Bregman geometry.

In Section 5, we recall the definition of the Hausdorff distance and some of its properties. We then introduce the extension to the Bregman setting. Here, we offer two separate variants: the Bregman–Hausdorff and Chernoff–Bregman–Hausdorff divergence. Finally, we offer an interpretation of both divergences based on the KL divergence, through the lens of information theory. In Sections 6 and 7, we demonstrate how nearest neighbor algorithms can be used to compute the two divergences.

We hope that extending the basic concept to the Bregman setting may also open the door for further development. This would not be unprecedented: as mentioned above, the development of Bregman k -means enabled the development of efficient Bregman ball trees and Bregman vantage point trees.

5. Bregman–Hausdorff Divergence

Hausdorff distance is a very natural concept: introduced by Hausdorff [44] in 1914, it has since become the standard distance measure for comparing sets of points, used ubiquitously across multiple fields of mathematics.

Recently, Hausdorff distance has also been used in applications. Indeed, it is a natural choice whenever two shapes need to be compared. For example, in computer vision, Hausdorff distance has been implemented as a measurement for the largest segmentation error in image segmentation [45,46] and to compare 3D shapes, such as meshes [47,48].

We start this section by providing the definition of the Hausdorff distance in a metric space. We then extend this concept to the Bregman setting. The inherent asymmetry of

Bregman divergences leads to a number of distinct definitions, which would all coincide in the metric setting. In devising the definitions, we are guided by the geometric and information-theoretical considerations. We elaborate on situations in which each of these new definitions finds a natural application.

Hausdorff distance in metric spaces. Given two sets P and Q in a metric space (M, d) , the **one-sided Hausdorff distance** from P to Q is defined as

$$d(P, Q) = \sup_{p \in P} \inf_{q \in Q} d(p, q).$$

Similarly to the Bregman divergence, this measurement is not symmetric: $d(P, Q) \neq d(Q, P)$. The **Hausdorff distance** between P and Q is the symmetrization of the two one-sided Hausdorff distances, and is given by the maximum,

$$H_d(P, Q) = \max\{d(P, Q), d(Q, P)\}.$$

Equivalently, the one-sided Hausdorff distance can be defined using a so-called **thickening**. A thickening—sometimes also called an offset—of the set Q of size r consists of all those points in the ambient space M , whose distance to Q is at most r . In other words, it is the union of all balls of radius r centered at a point in Q .

The one-sided Hausdorff distance from the set P to Q is the radius of the smallest thickening of Q that contains P :

$$d(P, Q) = \inf \left\{ r \geq 0 : P \subseteq \bigcup_{q \in Q} B_d(q; r) \right\},$$

where $B_d(x; r)$ is the ball of radius r (with respect to the metric d) centered at x . (We illustrate the two one-sided Hausdorff distances in Figure 8.) The Hausdorff distance between P and Q is—as before—the maximum of the two radii, $d(P, Q)$ and $d(Q, P)$. It is a well-known fact that the Hausdorff distance defines a *metric* on the collection of closed, bounded, and nonempty subsets of the metric space (M, d) .

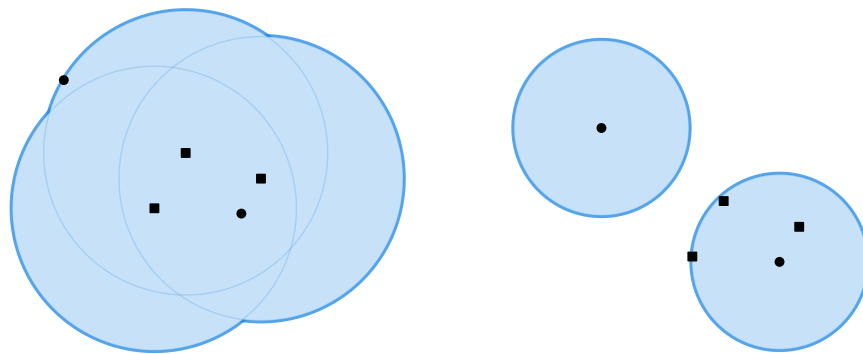


Figure 8. One-sided Hausdorff distances with respect to the Euclidean metric (d_{Euc}) between the sets P (black points) and Q (black squares). The **left** image highlights $d_{Euc}(P, Q)$ and the **right** image highlights $d_{Euc}(Q, P)$. Clearly, $d_{Euc}(P, Q) > d_{Euc}(Q, P)$.

Bregman–Hausdorff divergence. To extend the notion of the one-sided Hausdorff distance to the Bregman setting, we use the geometric perspective of thickenings to help us select viable definitions.

Let F be a function of Legendre type, defined on the domain Ω , and let P and Q be two nonempty subsets of Ω . The **primal** (resp., **dual**) **thickening** of Q of size $r \geq 0$ is the union of primal (resp., dual) balls of radius r , centered at the points in Q , with respect to

the divergence D_F . We define the **primal** (resp., **dual**) **(one-sided) Bregman–Hausdorff divergence** from P to Q , with respect to the divergence D_F , as

$$H_{D_F}(P\|Q) = \inf\{r \geq 0 : P \subseteq \bigcup_{q \in Q} B_F(q; r)\},$$

$$H'_{D_F}(P\|Q) = \inf\{r \geq 0 : P \subseteq \bigcup_{q \in Q} B'_F(q; r)\},$$

respectively. See Figure 9 for a visualization. We will refer to this new measurement as a divergence, rather than calling it a ‘one-sided distance’, to emphasize that it is generated by a Bregman divergence, and does not satisfy the triangle inequality.

Similarly to the one-sided Hausdorff distances, we have equivalent expressions for both the primal and the dual Bregman–Hausdorff divergence:

$$H_{D_F}(P\|Q) = \sup_{p \in P} \inf_{q \in Q} D_F(p\|q),$$

$$H'_{D_F}(P\|Q) = \sup_{p \in P} \inf_{q \in Q} D_F(q\|p).$$

These expressions will be useful for computations. It is worth noting that the asymmetry of Bregman divergences allows for more definitions, which however, deviate from the natural geometric interpretation of the original Hausdorff distance.

Furthermore, it would be possible to define symmetrized (primal and dual) Bregman–Hausdorff divergence as the maximum of the two variants. However, we refrain from symmetrizing it this way, for the same reason Bregman divergences are typically not symmetrized. Namely, each of the above definitions has a natural interpretation and applications. However, we will introduce a third variant, which will be naturally symmetric.

For popular divergences with established names and abbreviations, such as the KL and IS divergences, we shorthand the Bregman–Hausdorff divergences for these divergences to H_{KL} for the KL –Hausdorff divergence, and H_{IS} for the IS –Hausdorff divergence.

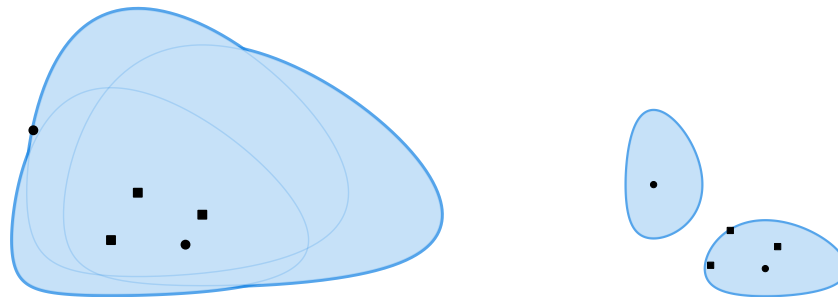


Figure 9. The **left** portion of the image visualizes the primal Bregman–Hausdorff divergence from P to Q , and the **right** visualizes the Bregman–Hausdorff divergence from Q to P . The set P consists of black points, the set Q consists of black squares. Shaded regions are the primal thickening of Q (on the **left**) and Q (on the **right**). One sees that the primal thickening of Q has to be of a large radius in order to contain the far left point of P . In contrast, the primal thickening of P can contain Q at a smaller radius. Thus, the difference between $H_{KL}(P\|Q)$ and $H_{KL}(Q\|P)$ can be used to see that Q clusters about P , but not vice versa.

The proposed Bregman–Hausdorff divergences can be used to compare the probabilistic predictions of a machine learning model with a reference model, as we showcase on the example of the KL divergence in the next paragraph.

Interpreting the Bregman–Hausdorff divergences with respect to the KL divergence. We can now extend the interpretation of the KL divergence presented in Section 2 to the

Bregman–Hausdorff divergences based on this divergence. This new case involves not only pairs of probability vectors, but pairs of *collections* of such vectors.

Let P and Q be nonempty collections of probability vectors in Δ^d . If we form a primal KL ball $B_{KL}(q; r)$, with a fixed radius $r \geq 0$, around every point $q \in Q$, then the region covered by these balls will contain all probability vectors that can approximate a vector in Q with an expected loss of at most r bits. Now, if the set P is contained in this region, then r is an upper bound on how inefficient the approximation of probabilities in Q is for some vector $p \in P$. Thus, by taking the infimum over all radii such that P is contained within the primal KL balls around Q , we can compute how efficient the approximation can be. The infimum is precisely the primal Bregman–Hausdorff divergence from P to Q , $H_{KL}(P\|Q)$; it measures the maximum expected efficiency loss (in bits) if P is used to reasonably approximate Q . In other words, for any probability vector $p \in P$, there exists a vector $q \in Q$, which p approximates with an expected efficiency loss of at most $H_{KL}(P\|Q)$ bits.

In contrast, the dual Bregman–Hausdorff divergence $H'_{KL}(P\|Q)$ measures the minimum radius for which the dual thickening of Q covers P . Each dual ball $B'_{KL}(q; r)$ contains all probability vectors that q approximates with an expected efficiency loss of at most r bits. Thus, when $r = H'_{KL}(P\|Q)$, the union of the dual Bregman balls captures the maximum expected number of bits lost for any probability vector $p \in P$ to be reasonably approximated by a probability vector in Q .

Hence, $H_{KL}(P\|Q)$ and $H'_{KL}(Q\|P)$ measure how P and Q can approximate each other. Specifically, $H_{KL}(P\|Q)$ is the maximum loss of expected bits if any vector $p \in P$ is used to approximate some vector $q \in Q$. On the other hand, for $H'_{KL}(Q\|P)$, every vector $q \in Q$ is approximated by some probability vector in P , but not every q will be used as an approximator. For $H_{KL}(P\|Q) = r$, every point in P is contained in some $B_{KL}(q; r)$, so every p is approximating the center of a ball it is contained in.

We can use the Bregman–Hausdorff divergence in the assessment of performance of machine learning models. Indeed, let M_1 and M_2 be two different classification models trained using the KL divergence loss (or equivalently the cross-entropy loss). We also let X, Z be two data sets, and denote the probabilistic predictions made by the models as $P = \{M_1(x; \theta_1)\}_{x \in X}$ and $Q = \{M_2(z; \theta_2)\}_{z \in Z}$. Then, $H_{KL}(P\|Q)$ quantifies a divergence from the set of predictions made by Q towards the set of predictions made by P .

We stress that this measurement *does not rely on any explicit pairing* between the outputs of two models (i.e., there is no obvious bijection between X and Z). Although the data sets X and Z are not explicitly paired, we can still make a reasonable numerical measurement between the two. This is the case when, for example, $M_1 = M_2$, and X and Z are training and test data, respectively. In this case, the values $H_{KL}(P\|Q)$ and $H'_{KL}(Q\|P)$ can be used to gauge the generalization power of a model. Importantly, this measurement is consistent with the loss function used to train the model.

Chernoff–Bregman–Hausdorff distance. We propose one more natural distance measurement. As its name suggests, the Chernoff–Bregman–Hausdorff distance is based on the notion of the Chernoff point. As before, we let F be a function of Legendre type, defined on the domain Ω , and let P and Q be two nonempty subsets of Ω .

For each pair of points $(p, q) \in P \times Q$, write $c_{p,q}$ for the Chernoff point of the set $\{p, q\}$, and write $C = \{c_{p,q} : (p, q) \in P \times Q\}$ for the collection of all the Chernoff points. Then, the primal (resp., dual) Chernoff–Bregman–Hausdorff distance is the smallest size of the primal (resp., dual) thickening of C that contains the union $P \cup Q$. To be more concrete, we define the **primal Chernoff–Bregman–Hausdorff distance** between P and Q as

$$CH_{D_F}(P, Q) = \inf\{r \geq 0 : P \cup Q \subseteq \bigcup_{c \in C} B'_F(c; r)\},$$

and the **dual Chernoff–Bregman–Hausdorff distance** between P and Q as

$$CH'_{D_F}(P, Q) = \inf\{r \geq 0 : P \cup Q \subseteq \bigcup_{c \in C} B_F(c; r)\}.$$

We emphasize that each divergence is named after the type of the Bregman ball that *grows about the pair* $(p, q) \in P \times Q$, and *not* the balls growing about the Chernoff points. We visualize the primal and dual Chernoff–Bregman–Hausdorff distances in Figure 10. Unlike the Bregman–Hausdorff divergences, the Chernoff–Bregman–Hausdorff distance is symmetric.

If fact, the set C can be viewed as the ‘average’ of P and Q with respect to the chosen divergence at the level of sets. In contrast to symmetrizing the Bregman–Hausdorff divergence by taking the average $(H_{D_F}(P||Q) + H_{D_F}(Q||P))/2$, the Chernoff–Bregman–Hausdorff distance avoids mixing directions of divergence computations. In particular, in the context of the KL divergence, the corresponding Chernoff–Bregman–Hausdorff distance inherits the information–theoretical interpretation, as we will see shortly. In the case of the squared Euclidean distance, C contains the usual arithmetic average $\frac{p+q}{2}$ for each pair $(p, q) \in P \times Q$.

Again, for the KL and IS divergences, we shorten the notation to CH_{KL} and CH_{IS} , respectively.

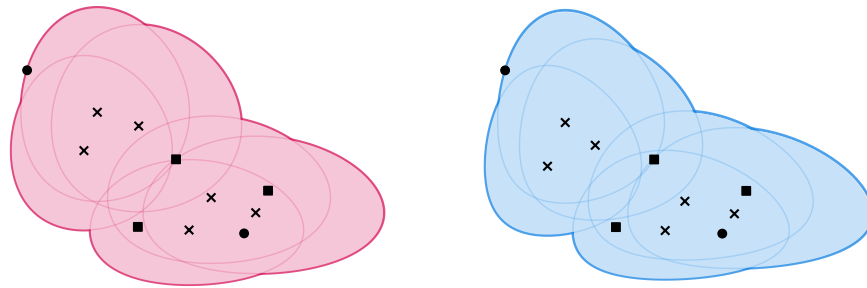


Figure 10. On the **left**, we visualize the primal Chernoff–Bregman–Hausdorff distance between P and Q . On the **right**, we visualize the dual Chernoff–Bregman–Hausdorff distance between P and Q . The set P consists of black points, the set Q of black squares. Chernoff points are marked as $\times$. The **left** is a *dual* thickening of the Chernoff points while the **right** is a *primal* thickening.

Interpreting the Chernoff–Bregman–Hausdorff distance for the KL divergence. As the primal Chernoff–Bregman–Hausdorff distance is defined by taking the infimum of the radius of a dual Bregman ball about the Chernoff points, CH_{KL} gives the least number of expected bits lost using C to approximate both P and Q . Similarly, the dual Chernoff–Bregman–Hausdorff distance, where we center primal balls about each Chernoff point, gives the least number of expected bits lost to approximate C by either P or Q .

Returning to machine learning, for models $P = \{M_1(x; \theta)\}_{x \in X}$ and $Q = \{M_2(z; \theta')\}_{z \in Z}$, C can be viewed as the collection of ‘average’ probability distributions [20] for every $(p, q) \in P \times Q$. Therefore, $CH_{KL}(P, Q)$ measures the maximum expected loss of coding efficiency (in bits) when attempting to reasonably approximate both P and Q using C . In this sense, C contains ‘joint approximators’ for pairs from $P \times Q$. Similarly, $CH'_{KL}(P, Q)$ measures the maximum expected loss when using P and Q to approximate the set C . While the Bregman–Hausdorff divergence is applicable if either P or Q is a reference set of vectors, the Chernoff–Bregman–Hausdorff distance is a natural choice when P and Q play the same role.

6. Algorithms for Bregman–Hausdorff Divergences

In this section, we present first algorithms for the Bregman–Hausdorff divergences. The algorithms rely on the data structure for nearest neighbor search, which we outline next. For the remainder of this section, we let F be a decomposable function of Legendre type, defined on the domain Ω , and let P and Q be two finite, nonempty subsets of Ω .

Bregman Kd-trees. We use the Bregman Kd-tree structure and search algorithm mentioned in Section 4. In [34], Pham and Wagner experimentally show that Kd-trees for Bregman divergences are efficient in a range of practical situations. In particular, they perform better than Cayton’s Bregman ball trees, and alleviate issues with certain other implementations.

The aforementioned Bregman Kd-tree implementation works for decomposable Bregman divergences [49,50], including the squared Euclidean distance, and KL and IS divergences. One additional benefit of this algorithm is its ability to compute the nearest neighbor in either direction.

Computing the Bregman–Hausdorff divergences. We first provide an algorithm for computing the Bregman–Hausdorff divergences between P and Q .

To compute the Bregman–Hausdorff divergence $H_F(P\|Q)$ from P to Q , we can use a version of the Kd-tree data structure for decomposable Bregman divergences [34]. We first construct a Kd-tree to represent P . Then, for each $q \in Q$, we search for $\rho^* = \arg \min_{p \in P} D_F(p\|q)$, maintaining value of the largest divergence, $D_F(q\|\rho^*)$. We denote this maintained value as *haus* in the three algorithms presented.

In Algorithm 1, we can replace line 2 and line 4 with any nearest neighbor structure and search. As the Bregman Kd-tree can be queried for divergences computed in both directions, this algorithm also works for computing the dual Bregman–Hausdorff divergence. For a proximity search algorithm with time complexity $O(\mathcal{C}(n, d))$, this algorithm runs in $O(m \cdot \mathcal{C}(n, d))$ time.

Algorithm 1 Primal Bregman–Hausdorff divergence algorithm (basic version)

Require: Point clouds P and Q of size n, m ; decomposable Bregman divergence D_F .

Ensure: Bregman–Hausdorff divergence $H_{D_F}(P\|Q)$

```

1: haus  $\leftarrow 0$ 
2: KdTree  $\leftarrow$  build Kd-tree( $P$ )
3: for  $q \in Q$  do
4:    $nn = \text{KdTree.query}(q, D_F)$   $\triangleright$  Find  $\arg \min_{p \in P} D_F(p\|q)$ 
5:    $nn\_div = D_F(nn, q)$ 
6:    $haus = \max(nn\_div, haus)$ 
7: return haus

```

To approximate the Bregman–Hausdorff divergence, one can use a $(1 + \epsilon)$ -nearest neighbor algorithm.

We can accelerate Algorithm 1 by adding an early query termination during the Kd-tree search. The the best of our knowledge, this technique was introduced to approximate the one-sided Hausdorff distance between 3D meshes [48]. We adjust the Kd tree query in line 4 as follows. During each query, candidates for $\arg \min_{p \in P} D_F(p\|q_i)$ are found. For any candidate $\rho \in P$, if $D_F(q_i\|\rho) \leq haus$, then we terminate the query. We denote this adjusted search as the *shell_query* method on line 4 in Algorithm 2. The *shell* method returns the nearest neighbor if the search completes and *Null* otherwise. An illustration of the above considerations can be seen in Figure 11.

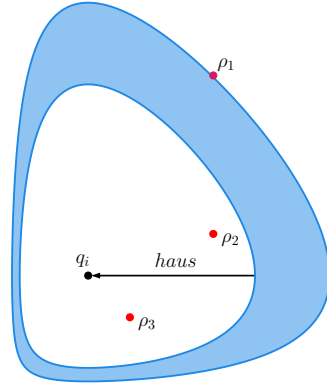


Figure 11. A shrinking shell around a query, q_i , whose inner radius is defined by $haus$ and outer radius is defined by $d(\rho_1 \| q_i)$. The search will terminate at ρ_2 , since $D_F(\rho_2 \| q) < haus$, instead of returning ρ_3 as the true nearest neighbor.

In the worst case scenario, the shell variant will have the same complexity as Algorithm 1, but we will see in Section 7 that reducing the number of complete searches provides significant speed-ups, even in high dimensions.

Algorithm 2 Primal Bregman–Hausdorff divergence shell algorithm (improved)

Require: Point clouds P and Q of size n, m , respectively; decomposable Bregman divergence D_F .

Ensure: Bregman–Hausdorff divergence $H_{D_F}(P \| Q)$

```

1:  $haus \leftarrow 0$ 
2:  $KdTree \leftarrow \text{build Kd-tree}(P)$ 
3: for  $q \in Q$  do
4:    $nn = KdTree.shell\_query(q, D_F, haus)$ 
5:   if  $nn \neq \text{Null}$  then
6:      $haus \leftarrow D_F(nn, q)$ 
7: return  $haus$ 
```

We will experimentally compare these implementations with a naive algorithm, where the Kd-tree search in line 4 of Algorithm 1 is replaced by a linear search.

Computing the Chernoff–Bregman–Hausdorff distance. We also provide an algorithm for the Chernoff–Bregman–Hausdorff distance. To compute the distance, we first determine the set of Chernoff points: $C = \{c_{p,q} : (p, q) \in P \times Q\}$.

To approximate the Chernoff point $c_{p,q}$ for a pair $(p, q) \in P \times Q$, we perform a bisection search along the line segment connecting p and q : $\alpha p + (1 - \alpha)q$ with parameter $\alpha \in [0, 1]$. This search was proposed by Nielsen [20]. We assume it runs in time $O(\beta(\epsilon)d)$ to be ϵ close to the true Chernoff point, with d being the dimension of the ambient space in which the domain Ω lies. Thus this component of the algorithm runs in $O(\beta(\epsilon)d m n)$ time. Letting $O(\mathcal{C}(n + m, d))$ be the time complexity of the chosen proximity search algorithm, the search runs in $O(nm \cdot \mathcal{C}(n + m, d))$ time. This gives us a total running time of $O(mn(\beta(\epsilon)d + \mathcal{C}(n + m, d)))$.

As with the Bregman–Hausdorff divergence, Kd-trees may be replaced by other exact Bregman nearest neighbor structure and algorithms; an $(1 + \epsilon)$ approximation may be used as well.

Computing the dual Chernoff–Bregman–Hausdorff distance requires mapping the input to the Legendre conjugate space. This adds a preprocessing step, but does not affect the complexity of the algorithm.

7. Experiments

In this section, we compute Bregman–Hausdorff divergences between various data sets. We work both with practical and synthetic data sets, and provide computation times. Generally, we use Algorithms 1 and 2 with domain $\Omega = \Delta^{d-1}$, where d depends on the chosen data set. We will run experiments using the KL and IS divergences, as well as the squared Euclidean distance. We will focus on Algorithms 1 and 2 because, unlike Algorithm 3, they promise to be efficient in practice. The computed values are exact up to machine precision, since the Bregman Kd-trees we use provide exact nearest neighbors.

Algorithm 3 Primal Chernoff–Bregman–Hausdorff divergence algorithm

Require: Point clouds P and Q of size n, m ; decomposable Bregman divergence D_F .

Ensure: Chernoff–Bregman–Hausdorff divergence $CH_{D_F}(P\|Q)$

```

1:  $haus \leftarrow 0$ 
2:  $C = \text{empty array of size } nm \times \dim(\Omega)$ 
3:  $i = 0$ 
4: for  $q \in Q$  do
5:   for  $p \in P$  do
6:      $C[i] = \text{Chernoff}(p, q)$  ▷ Compute the Chernoff points.
7:      $i = i + 1$ 
8:  $KdTree \leftarrow \text{build Kd-tree}(C)$ 
9: for  $a \in Q \cup P$  do
10:   $nn = KdTree.query(a, D_F)$  ▷ Find  $\arg \min_{c \in C} D_F(a\|c)$ 
11:   $nn\_div = D_F(nn, q)$ 
12:   $haus = \max(nn\_div, haus)$ 
13: return  $haus$ 

```

Compiler and hardware. Software was compiled with Clang 14.0.3. The experiments were performed on a single core of a 3.5 GHz ARM64-based CPU with 4 MB L2 cache using 32 GB RAM.

Data sets. We use predictions from machine learning models and synthetic data sets. We train two neural networks, M_1 and M_2 , on a classification task using CIFAR100, which has 50,000 training images and 10,000 test images. Specifically, we perform transfer learning using EfficientNetB0 [51] pretrained on ImageNet as a backbone. The first model, M_1 , is trained with fine-tuning and the second model, M_2 , is trained without. Both models are trained using the KL divergence as the loss function. The models, M_1 and M_2 , achieve 80.22%, and 71.74% test accuracy, respectively. From each model, we produce two sets of predictions: (trn_i, tst_i) , for $i \in \{1, 2\}$. The synthetic data sets are drawn uniformly from the open simplex in dimension 50, 100, and 250. The target data set, P , has 100,000 sample points; the query data set, Q , has 20,000.

Bregman–Hausdorff computations. One concrete motivation for this measurement was the need to quantify how well one set of predictions approximates another. We set this up as a computation of the Bregman–Hausdorff divergence. We are especially interested in comparing the probabilistic predictions arising from two models of different quality, as well as the training and test data.

Additionally, we aim to investigate to what extent the choice of the underlying Bregman divergence affects the resulting Bregman–Hausdorff divergence. Of course, in practice, one would use an appropriate divergence. For example, if the data represent probability vectors, the KL divergence, and by extension the KL –Hausdorff divergence is a natural choice. However, in practice the (squared) Euclidean distance and the related Hausdorff distance is often used, mostly due to limitations of computational tools. The following experiments confirm that the choice of divergence has a significant effect on the outputs of the corresponding Bregman–Hausdorff divergence.

Using the relative entropy loss. In Table 2, we compare the Bregman–Hausdorff divergence between the four sets of predictions. Divergences change by row, data sets by column, and the units for the KL divergence are in bits. We choose a subset of the possible pairings to highlight the asymmetry and some key interpretations, and also for brevity.

From the row containing the values of H_{KL} in Table 2, the lowest value is $H_{KL}(tst_1||trn_1)$. This value shows that tst_1 predicts trn_1 with a maximum expected loss of 1.764 bits. In particular, $H_{KL}(tst_1||trn_1) < H_{KL}(trn_1||tst_1)$ —the relation we would expect from M_1 's predictions on a train and test data sets. In contrast, $H_{SE}(tst_1||trn_1) = 0.371$ is the largest value in its row. In particular, $H_{SE}(tst_1||trn_1) > H_{SE}(trn_1||tst_1)$, which is the reverse of how we expect the soft predictions from the training data and test data to behave. On the surface, this would indicate that trn_1 is a poor predictor of tst_1 . However, as the models were trained to minimize the KL divergence, and because of the difference between the Euclidean geometry and the Bregman geometry induced by the KL divergence, values from H_{SE} do not carry the information–theoretical interpretation. This shows the importance of analyzing these models using the proposed KL –Hausdorff divergence rather than the standard one-sided Hausdorff distance.

We see a similar effect for the computation of H_{IS} and H'_{IS} ; the results suggest that the geometry induced by the IS divergence is drastically different from the geometry induced by the KL divergence, and thus the computations lack an obvious interpretation in this context. However, for comparing speech and sound data (as opposed to probabilistic predictions) the IS –Hausdorff divergence would be the preferred choice. In this case, the measurement would inherit an interpretation from the IS divergence [11]. So, while in practice it is clear that the KL –Hausdorff divergence is the only reasonable choice, we provide the table to demonstrate that our algorithms can handle various Bregman divergences and that the outputs are highly sensitive to the chosen divergence.

Table 2. Bregman–Hausdorff divergences between outputs of two classification models trained using the relative entropy loss (which corresponds to the KL divergence). The values for H_{KL} and H'_{KL} are measured in bits.

	$(tst_1 trn_1)$	$(trn_1 tst_1)$	$(trn_1 tst_2)$	$(trn_1 trn_2)$	$(trn_2 trn_1)$
H_{KL}	1.765b	2.215b	2.044b	2.236b	2.237b
H'_{KL}	3.797b	4.541b	4.509b	4.343b	4.033b
H_{IS}	32,496.887	9,822,345.381	1,739,646,377.745	14,801,113.426	584,772.398
H'_{IS}	3147.685	2987.831	1998.378	2360.0485	1309.6230
H_{SE}	0.371	0.296	0.243	0.234	0.271

Using the mean squared error loss. In this case, we train M_1 and M_2 by minimizing the mean squared error (MSE) loss, which corresponds to the squared Euclidean distance. In this case, the training accuracy of M_1 is 73.97% while the training accuracy of M_2 is 69.18%. For numerical stability, we decrease the learning rate for M_2 for this loss function. We stress that, like before, these models output classification predictions, interpreted as vectors in $\mathbb{R}^{100}$.

Similarly, we produce two sets of data per model, labeled as (tst_i^{SE}, trn_i^{SE}) for $i \in \{1, 2\}$. We can compute the Bregman–Hausdorff divergences for the same pairs of sets as in Table 2 and record them in Table 3.

In contrast to minimizing with respect to the KL divergence, all KL –Hausdorff divergence values from SE distance minimized data sets are smaller. This tells us that tst_i^{SE} and trn_i^{SE} are closely clustered together using the KL divergence geometry more so than tst_i and trn_i from above. In contrast, H_{SE} is larger when comparing the training data predictions, trn_1^{SE} and trn_2^{SE} , between the two models, showing that the two resulting prediction sets are less clustered when changing the minimization function.

Table 3. Bregman–Hausdorff divergences when M_1 and M_2 are trained to minimize the mean squared error loss (corresponding to the squared Euclidean distance).

	$(\text{tst}_1^{SE} \parallel \text{trn}_1^{SE})$	$(\text{trn}_1^{SE} \parallel \text{tst}_1^{SE})$	$(\text{trn}_1^{SE} \parallel \text{tst}_2^{SE})$	$(\text{trn}_1^{SE} \parallel \text{trn}_2^{SE})$	$(\text{trn}_2^{SE} \parallel \text{trn}_1^{SE})$
H_{KL}	1.888b	1.557b	1.938b	1.979b	1.888b
H'_{KL}	2.719b	1.904b	2.965b	2.625b	2.527b
H_{IS}	891,154.1	12,281.8	251,969.8	12,875,882.0	2,985,641,328,333.4
H'_{IS}	961.2	552.6	551.8	819.2	2080.3
H_{SE}	0.326	0.212	0.291	0.330	0.330

Timings. Finally, we compare the computation speeds of the algorithms using the Bregman Kd-tree (Algorithms 1 and 2) and a version that uses a linear search. The results can be seen in Table 4. The left columns are computation times for two sets of predictions. For the Kd-tree algorithm without the shell acceleration, we see significant speed-ups in the left two columns, while the right three columns have a similar time between the two searches. As predictions from classification models tend to cluster near the vertices of the simplex and randomly generated points have more spread across the simplex, we see that the distribution of points heavily influences the speed of computations. Similarly, the run time increases as the dimension increases. This is expected, as the performance of Kd-trees is known to degrade in higher dimensions [52,53].

Perhaps surprisingly, when we apply early termination via the shell method, we see that we maintain considerable speed-ups even in high dimensions. While the worst case complexity of the shell method is still equivalent to the Kd-tree search, we see up to $1000\times$ speed-up even in dimension 250.

These speed-ups show that the Bregman–Hausdorff divergence is an efficient tool for the comparison of the outputs of machine learning models—as well as in other situations that require the comparison between two sets of vectors.

Table 4. Computation times for the KL– and IS–Hausdorff divergences using the Kd-tree search algorithms and a linear search. Left two columns use data sets from the predictions of M_1 and M_2 in dimension 100; the three rightmost columns use synthetic data sets using a set of 20,000 sampled points as the query into 100,000 points. Speed-up compares the Kd-shell computation times and linear search computation times.

		$(\text{trn}_1 \parallel \text{tst}_2)$	$(\text{trn}_2 \parallel \text{trn}_1)$	Dim 10	Dim 50	Dim 100	Dim 250
KL	Kd-shell	0.47 s	1.63 s	0.02 s	0.78 s	1.36 s	2.19 s
	Kd-tree	1.74 s	6.79 s	2.43 s	481.15 s	999.26 s	2413.94 s
	Linear	284.30 s	1417.32 s	112.84 s	563.17 s	1113.63 s	2753.32 s
	Speed-up	598.54×	866.86×	4906.26×	720.16×	818.24×	1254.36×
IS	Kd-shell	0.21 s	1.01 s	0.01 s	0.66 s	1.48 s	3.92 s
	Kd-tree	7.09 s	23.92 s	3.85 s	412.38 s	738.37 s	1645.400 s
	Linear	279.70 s	1391.39 s	115.12 s	574.26 s	1129.01 s	2803.430 s
	Speed-up	1338.29×	1374.89×	6395.77×	862.25×	759.25×	714.98×

Alternative implementations. There are alternative approaches to Bregman nearest neighbor search, but they come with certain limitations. To implement computations for Bregman–Hausdorff divergences using Cayton’s Bregman ball trees [31], significant changes to the source code would be required. However, as shown in [34], Bregman Kd-trees are a preferred choice as they are more efficient in practice. Implementations of ball trees and VP trees (due to Nielsen, Piro, and Barlaud) are not usable on modern systems due to severe compilation issues. Our comparison is therefore limited to our Kd-tree algorithms and the linear search.

8. Conclusions

Modern machine learning models commonly rely on optimizing the Kullback–Leibler divergence. As a result, the resulting vectors, such as the probabilistic predictions of a classifier, would naturally be measured using this divergence. However, many standard tools that could be used to analyze such data are limited to the Euclidean distance.

One geometric tool that found applications in various fields in the Euclidean context is the Hausdorff distance. It serves as a distance between two collections of points—without requiring any pairing or alignment between the inputs. While potentially useful also in machine learning, the standard Hausdorff distance is limited to the metric setting—as opposed to the Kullback–Leibler divergence and other Bregman divergences.

This paper is an attempt to bridge the gap between the necessity of using non-metric distances—and the familiarity with such concepts and the paucity of computational tools supporting them. Specifically, we outlined the field of Bregman geometry, including descriptions of existing tools that can be used in the above context. Importantly, we extended a popular geometric tool to this new setup: We defined several variants of Bregman–Hausdorff divergences, which allow one to compare two collections of vectors according to a chosen Bregman divergence. We highlighted situations in which each of these variants can be meaningfully used. In particular, when the underlying divergence is the Kullback–Leibler divergence, we explained that its Hausdorff counterpart inherits a clear information–theoretical interpretation.

To make these theoretical considerations practical, we also designed novel algorithms for computing the newly defined Bregman–Hausdorff divergences. To ensure efficiency, we applied the Bregman version of the Kd-tree data structure for exact nearest neighbor search, which we recently developed. We exploited the special structure of the Bregman–Hausdorff computations to better leverage the Kd-tree data structure. We implemented these three algorithms.

We benchmarked our implementations in scenarios arising from a machine learning setup. Our experiments show that our most optimized algorithm performs well in this scenario. Specifically, it achieves several orders of magnitude speed-up compared to more basic algorithms. This is surprising, given that the Kd-trees the algorithm uses are known to perform poorly in such high-dimensional scenarios. In fact, our straightforward application of Kd-trees scales significantly worse with dimension. Understanding this speed-up theoretically is an interesting future direction.

Overall, we hope that the Bregman–Hausdorff divergence will find applications. In this paper, we focused on the theoretical setup, interpretations of the new measurement, as well as efficiency aspects. More broadly, we hope that geometric tools supporting Bregman divergences—and not only the Euclidean distance or other metrics—will eventually become more popular in machine learning.

Author Contributions: Conceptualization, T.P., H.D.P.K. and H.W.; methodology, T.P., H.D.P.K. and H.W.; software, T.P.; validation, T.P., H.D.P.K. and H.W.; formal analysis, T.P., H.D.P.K. and H.W.; investigation, T.P., H.D.P.K. and H.W.; writing—original draft preparation, T.P. and H.W.; writing—review and editing, T.P., H.D.P.K. and H.W.; visualization, T.P.; supervision, H.W.; funding acquisition, H.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the 2022 Google Research Scholar program awarded for project “Algorithms for Topological Analysis of Neural Networks”. Hana Dal Poz Kouřimská was supported by the DFG Project No. 524578210.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Gray, R. *Entropy and Information Theory*; Springer: New York, NY, USA, 2013.
- Cover, T.; Thomas, J. *Elements of Information Theory*; Wiley: Hoboken, NJ, USA, 2012.
- Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [\[CrossRef\]](#)
- Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. *arXiv* **2022**, arXiv:1312.6114.
- Pathria, R.K.; Beale, P.D. *Statistical Mechanics*, 3rd ed.; Academic Press: Boston, MA, USA, 2011.
- Amari, S.I. *Information Geometry and Its Applications*; Springer: Tokyo, Japan, 2016. [\[CrossRef\]](#)
- Bregman, L.M. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Comput. Math. Math. Phys.* **1967**, *7*, 200–217. [\[CrossRef\]](#)
- Bauschke, H.H.; Borwein, J.M. Legendre functions and the method of random Bregman projections. *J. Convex Anal.* **1997**, *4*, 27–67.
- Rockafellar, R.T. *Convex Analysis*; Princeton University Press: Princeton, NJ, USA, 1970. [\[CrossRef\]](#)
- Do, M.N.; Vetterli, M. Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance. *IEEE Trans. Image Process.* **2002**, *11*, 146–158. [\[CrossRef\]](#) [\[PubMed\]](#)
- Itakura, F. Analysis synthesis telephony based on the maximum likelihood method. In Proceedings of the 6th International Congress on Acoustics, Tokyo, Japan, 21–28 August 1968.
- Mahalanobis, P.C. On the generalised distance in statistics. *Proc. Natl. Inst. Sci. India* **1936**, *2*, 49–55.
- Sapatinas, T. Discriminant Analysis and Statistical Pattern Recognition. *J. R. Stat. Soc. Ser. Stat. Soc.* **2005**, *168*, 635–636. [\[CrossRef\]](#)
- Law, M.T.; Yu, Y.; Cord, M.; Xing, E.P. Closed-Form Training of Mahalanobis Distance for Supervised Clustering. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 3909–3917. [\[CrossRef\]](#)
- Li, L.; Sun, C.; Lin, L.; Li, J.; Jiang, S. A dual-layer supervised Mahalanobis kernel for the classification of hyperspectral images. *Neurocomputing* **2016**, *214*, 430–444. [\[CrossRef\]](#)
- Nock, R.; Nielsen, F. Fitting the Smallest Enclosing Bregman Ball. In *Machine Learning: ECML 2005*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 649–656. [\[CrossRef\]](#)
- Boissonnat, J.D.; Nielsen, F.; Nock, R. Bregman Voronoi diagrams. *Discret. Comput. Geom.* **2010**, *44*, 281–307. [\[CrossRef\]](#)
- Nielsen, F. Chernoff information of exponential families. *arXiv* **2011**, arXiv:1102.2684.
- Edelsbrunner, H.; Virk, Ž.; Wagner, H. Smallest Enclosing Spheres and Chernoff Points in Bregman Geometry. In Proceedings of the 34th International Symposium on Computational Geometry, Budapest, Hungary, 11–14 June 2018.
- Nielsen, F. An Information-Geometric Characterization of Chernoff Information. *IEEE Signal Process. Lett.* **2013**, *20*, 269–272. [\[CrossRef\]](#)
- Nielsen, F. Revisiting Chernoff Information with Likelihood Ratio Exponential Families. *Entropy* **2022**, *24*, 1400. [\[CrossRef\]](#)
- Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **1982**, *28*, 129–137. [\[CrossRef\]](#)
- Banerjee, A.; Merugu, S.; Dhillon, I.S.; Ghosh, J. Clustering with Bregman Divergences. *J. Mach. Learn. Res.* **2005**, *6*, 1705–1749.
- Dirichlet, G.L. Über die Reduction der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. *J. Pure Appl. Math. Crelles J.* **1850**, *1850*, 209–227. [\[CrossRef\]](#)
- Voronoi, G. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire. Sur quelques propriétés des formes quadratiques positives parfaites. *J. Pure Appl. Math. Crelles J.* **1908**, *1908*, 97–102. [\[CrossRef\]](#)
- Voronoi, G. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les paralléloèdres primitifs. *J. Pure Appl. Math. Crelles J.* **1908**, *1908*, 198–287. [\[CrossRef\]](#)
- Chazelle, B. An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.* **1993**, *10*, 377–409. [\[CrossRef\]](#)
- Omohundro, S.M. Five Balltree Construction Algorithms. 1989. Available online: https://steveomohundro.com/wp-content/uploads/2009/03/omohundro89_five_balltree_construction_algorithms.pdf (accessed on 12 May 2025).
- Yianilos, P.N. Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. In Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). Society for Industrial and Applied Mathematics, Austin, TX, USA, 25–27 January 1993; pp. 311–321.
- Bentley, J.L. Multidimensional binary search trees used for associative searching. *Commun. ACM* **1975**, *18*, 509–517. [\[CrossRef\]](#)
- Cayton, L. Fast Nearest Neighbor Retrieval for Bregman Divergences. In Proceedings of the 25th International Conference on Machine Learning, ICML '08, New York, NY, USA, 5–9 July 2008; pp. 112–119. [\[CrossRef\]](#)
- Nielsen, F.; Piro, P.; Barlaud, M. Tailored Bregman ball trees for effective nearest neighbors. In Proceedings of the 25th European Workshop on Computational Geometry (EuroCG), Brussels, Belgium, 16–18 March 2009; pp. 29–32.
- Nielsen, F.; Piro, P.; Barlaud, M. Bregman Vantage Point Trees for Efficient Nearest Neighbor Queries. In Proceedings of the 2009 IEEE International Conference on Multimedia and Expo, (ICME), New York, NY, USA, 28 June–3 July 2009; pp. 878–881. [\[CrossRef\]](#)
- Pham, T.; Wagner, H. Fast Kd-trees for the Kullback–Leibler Divergence and other Decomposable Bregman Divergences. In Proceedings of the 19th Algorithms and Data Structures Symposium (WADS 2025), Toronto, ON, Canada, 11–15 August 2025.

35. Zhang, Z.; Ooi, B.C.; Parthasarathy, S.; Tung, A.K.H. Similarity Search on Bregman Divergence: Towards Non-Metric Indexing. *Proc. VLDB Endow.* **2009**, *2*, 13–24. [\[CrossRef\]](#)
36. Song, Y.; Gu, Y.; Zhang, R.; Yu, G. BrePartition: Optimized High-Dimensional kNN Search with Bregman Distances. *arXiv* **2020**, arXiv:2006.00227. [\[CrossRef\]](#)
37. Boytsov, L.; Naidan, B. Engineering Efficient and Effective Non-metric Space Library. In Proceedings of the Similarity Search and Applications—6th International Conference, SISAP 2013, A Coruña, Spain, 2–4 October 2013; Brisaboa, N.R., Pedreira, O., Zezula, P., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8199, pp. 280–293. [\[CrossRef\]](#)
38. Malkov, Y.A.; Yashunin, D.A. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 824–836. [\[CrossRef\]](#) [\[PubMed\]](#)
39. Edelsbrunner, H.; Letscher, D.; Zomorodian, A. Topological persistence and simplification. In Proceedings of the 41st Annual Symposium on Foundations of Computer Science, Redondo Beach, CA, USA, 12–14 November 2000; IEEE: Piscataway, NJ, USA, 2000; pp. 454–463.
40. Carlsson, G. Topology and data. *Bull. Amer. Math. Soc.* **2009**, *46*, 255–308. [\[CrossRef\]](#)
41. Edelsbrunner, H.; Wagner, H. Topological Data Analysis with Bregman Divergences. In Proceedings of the 33th International Symposium on Computational Geometry (SoCG), Brisbane, Australia, 4–7 July 2017; pp. 67–86. [\[CrossRef\]](#)
42. Edelsbrunner, H.; Harer, J. *Computational Topology: An Introduction*; American Mathematical Society: Providence, RI, USA, 2010.
43. Edelsbrunner, H.; Ölsböck, K.; Wagner, H. Understanding Higher-Order Interactions in Information Space. *Entropy* **2024**, *26*, 637. [\[CrossRef\]](#)
44. Hausdorff, F. *Grundzüge der Mengenlehre*; Göschens Lehrbücherei/Gruppe I: Reine und Angewandte Mathematik Series; Von Veit: Berlin, Germany, 1914.
45. Huttenlocher, D.P.; Klanderman, G.A.; Rucklidge, W. Comparing Images Using the Hausdorff Distance. *IEEE Trans. Pattern Anal. Mach. Intell.* **1993**, *15*, 850–863. [\[CrossRef\]](#)
46. Karimi, D.; Salcudean, S.E. Reducing the Hausdorff Distance in Medical Image Segmentation with Convolutional Neural Networks. *arXiv* **2019**, arXiv:1904.10030. [\[CrossRef\]](#)
47. Cignoni, P.; Rocchini, C.; Scopigno, R. Metro: Measuring Error on Simplified Surfaces. *Comput. Graph. Forum* **1998**, *17*, 167–174. [\[CrossRef\]](#)
48. Guthe, M.; Borodin, P.; Klein, R. Fast and Accurate Hausdorff Distance Calculation between Meshes. *J. WSCG* **2005**, *13*, 41–48.
49. Zhang, J. Divergence Function, Duality, and Convex Analysis. *Neural Comput.* **2004**, *16*, 159–195.
50. Nielsen, F.; Nock, R. The Dual Voronoi Diagrams with Respect to Representational Bregman Divergences. In Proceedings of the 2009 Sixth International Symposium on Voronoi Diagrams, Copenhagen, Denmark, 23–26 June 2009; pp. 71–78. [\[CrossRef\]](#)
51. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; PMLR: Breckenridge, CO, USA, 2019; pp. 6105–6114.
52. Marimont, R.B.; Shapiro, M.B. Nearest Neighbour Searches and the Curse of Dimensionality. *IMA J. Appl. Math.* **1979**, *24*, 59–70. [\[CrossRef\]](#)
53. Chávez, E.; Navarro, G.; Baeza-Yates, R.; Marroquín, J.L. Searching in metric spaces. *ACM Comput. Surv.* **2001**, *33*, 273–321. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.