On Distance Mapping from non-Euclidean Spaces to Euclidean Spaces

Wei Ren, Yoan Miche, Ian Oliver, Silke Holtmanns, Kaj-Mikael Björk, Amaury Lendasse

On Distance Mapping from non-Euclidean Spaces to Euclidean Spaces

Wei Ren[1], Yoan Miche[1], Ian Oliver[1], Silke Holtmanns[1], Kaj-Mikael Bjork[2], and Amaury Lendasse[3]

[1] Nokia Bell Labs
Karakaari 13, FI-02760 Finland
[2] Arcada University of Applied Sciences
Jan-Magnus Janssonin aukio 1, 00550 Helsinki, Finland
[3] The University of Iowa
Iowa City, IA 52242-1527, USA

Abstract. Most Machine Learning techniques traditionally rely on some forms of Euclidean Distances, computed in a Euclidean space (typically $\mathbb{R}^d$). In more general cases, data might not live in a classical Euclidean space, and it can be difficult (or impossible) to find a direct representation for it in $\mathbb{R}^d$. Therefore, distance mapping from a non-Euclidean space to a canonical Euclidean space is essentially needed. We present in this paper a possible distance-mapping algorithm, such that the behavior of the pairwise distances in the mapped Euclidean space is preserved, compared to those in the original non-Euclidean space. Experimental results of the mapping algorithm are discussed on a specific type of datasets made of timestamped GPS coordinates. The comparison of the original and mapped distances, as well as the standard errors of the mapped distributions, are discussed.

1 Introduction

Traditionally, most data mining and machine learning have relied on the classical (or variations thereof) Euclidean distance (Minkowski distance with the exponent set to 2), over data that lies in $\mathbb{R}^d$ (with $d \in \mathbb{N}_+$). One problem with this approach is that it forces the data provider to process the original, raw data, in such a way that it is in $\mathbb{R}^d$, while it might not be natural to do so. For example, encoding arbitrary attributes (such as words from a specific, finite set) using integers, creates an underlying order between the elements, which has to be carefully taken care of. It also creates a certain distance between the elements, and the various choices in the conversion process are practically unlimited and difficult to address. We therefore look here into the possibility of not converting the data to a Euclidean space, and retain the data in its original space, provided that we have a distance function between its elements. In effect, we "only" require and concern ourselves, in this paper, with metric spaces, over potentially non-Euclidean spaces. With the fact that most data mining and machine learning methods rely on Euclidean distances and their properties, we want to verify

that in such a case, the distances in a non-Euclidean metric space can behave close enough to the Euclidean distances over a Euclidean space. The reasoning behind this is that traditional Machine Learning and Data Mining algorithms might expect the distances between elements to behave in a certain manner, and respect certain properties, which we attempt to mimic with the following distance mapping approach.

A distance-mapping algorithm, in the context of this paper, is an approach that takes a set of objects as well as their pairwise distance function in the specific non-Euclidean space (so, a non-Euclidean metric space), and maps those pairwise distances to a canonical Euclidean space, in such a way that the distance distribution among the objects is approximately preserved in the mapped Euclidean space.

Distance-mapping algorithms are a useful tool in data applications, for example, data clustering and visualisations. Another good use case for distance-mapping, is mutual information [3,7,10], which is used to quantitatively measure the mutual dependence between two (or more) sets of random variables in information theory. Mutual information is most often estimated by constructing the k-nearest neighbors [7,10] graphs of the underlying data, which thus rely on the Euclidean distances. Hence, there is a strong need to re-calculate the distances over a potentially non-Euclidean space.

In the following section 2, we first introduce basic notations to describe the distance mapping approach in sections 3, 4 and 5. After a short discussion about important implementation details in section 7, we finally present and discuss results over a synthetic data set in section 8.

2 Notations

As in the data privacy literature, one traditionally defines a dataset of N records by $\boldsymbol{X} = [\boldsymbol{x_1}, ..., \boldsymbol{x_N}]^T$, the matrix of N samples (records) with d attributes $\{\boldsymbol{A^{(1)}}, ..., \boldsymbol{A^{(d)}}\}$. A record $\boldsymbol{x_l}$ is now defined as $\boldsymbol{x_l} = [a_l^{(1)}, a_l^{(2)}, ..., a_l^{(d)}]$, $a_l^{(j)} \in \mathbb{X}^{(j)}$, where $\mathbb{X}^{(j)}$ is the set of all the possible values for a certain attribute $\boldsymbol{A^{(j)}}$. Hence, we can see the vector $[a_1^{(j)}, a_2^{(j)}, ..., a_N^{(j)}]^T \in \mathbb{X}^{(j)}$ as a discrete random variable for a certain attribute over all the N samples.

Let us consider a metric space $\mathcal{X}^{(j)} = (\mathbb{X}^{(j)}, d^{(j)})$ using the set $\mathbb{X}^{(j)}$ explained above, endowed with the distance function $d^{(j)} : \mathbb{X}^{(j)} \times \mathbb{X}^{(j)} \longrightarrow \mathbb{R}_+$. Generally, $\mathcal{X}^{(j)}$ need not be an Euclidean metric space.

3 Distances over non-Euclidean spaces

Now we consider two metric spaces $\mathcal{X}^{(i)} = (\mathbb{X}^{(i)}, d^{(i)})$ and $\mathcal{X}^{(j)} = (\mathbb{X}^{(j)}, d^{(j)})$. Let us assume $\mathcal{X}^{(i)}$ to be a canonical Euclidean space with the distance function $d^{(i)}$ the Euclidean norm and $\mathbb{X}^{(i)} = \mathbb{R}^d$, while $\mathcal{X}^{(j)}$ is a non-Euclidean space endowed with a non-Euclidean distance function $d^{(j)}$.

Assume $\boldsymbol{x^{(j)}}$ and $\boldsymbol{y^{(j)}}$ are two sets of discrete independent and identically distributed (iid) random variables for a certain attribute over $\mathbb{X}^{(j)}$. The distances

within the metric space $\mathcal{X}^{(j)}$ can then be constructed by another set of random variable $\boldsymbol{z^{(j)}}$:

$$\boldsymbol{z^{(j)}} = d^{(j)}(\boldsymbol{x^{(j)}}, \boldsymbol{y^{(j)}}), \tag{1}$$

where the values of $\boldsymbol{z^{(j)}}$ are over $\mathbb{R}_+$.

We denote by $f_{\boldsymbol{z^{(j)}}}(d)$ the probability density function (PDF) of $\boldsymbol{z^{(j)}}$, which describes the pairwise distance distribution over the non-Euclidean metric space $\mathcal{X}^{(j)}$. In the same way, we define $f_{\boldsymbol{z^{(i)}}}(d)$ to be the distribution of pairwise distances over the Euclidean metric space $\mathcal{X}^{(i)}$.

We assume that there is a way to transform the distribution of the non-Euclidean distances $f_{\boldsymbol{z^{(j)}}}(d)$ to $f^{map}_{\boldsymbol{z^{(j)}}}(d)$ in such a way that $f^{map}_{\boldsymbol{z^{(j)}}}(d)$ can be as close as possible to the Euclidean distance distribution $f_{\boldsymbol{z^{(i)}}}(d)$:

$$\lim_{N\to\infty} f^{map}_{\boldsymbol{z^{(j)}}} = f_{\boldsymbol{z^{(i)}}}, \tag{2}$$

meaning that the two probability density function are equal at every point evaluated, in the limit case.

As we are using a limit number of realisations N of the random variables to estimate the distribution $f_{\boldsymbol{z^{(j)}}}(d)$, the limit over N is based on the assumption that we can "afford" to draw sufficiently large enough number N of the variables to possibly estimate $f_{\boldsymbol{z^{(j)}}}(d)$ to be close enough to $f_{\boldsymbol{z^{(i)}}}(d)$. We present the mapping approach used in this paper in the following section 4, by solving an integral equation so as to obtain equal probability masses.

4 Mapping solution

We propose to use Machine Learning (more specifically, Universal Function Approximators [4]) to map the distribution $f_{\boldsymbol{z^{(j)}}}$ of the non-Euclidean distance to the distribution $f_{\boldsymbol{z^{(i)}}}$ of the Euclidean distance, with the fact that most Machine Learning techniques are able to fit a continuous input to another different continuous output.

We then want to make it so that given a certain distance $z = d^{(j)}(x, y)$ obtained over $\mathcal{X}^{(j)}$, we calculate α such that

$$\int_0^z f_{\boldsymbol{z^{(j)}}}(t)dt = \int_0^\alpha f_{\boldsymbol{z^{(i)}}}(t)dt, \tag{3}$$

We want to obtain α values so that the probability masses of the distances in the Non-Euclidean metric space $\mathcal{X}^{(j)}$ and the Euclidean metric space $\mathcal{X}^{(i)}$ are the same. The α value is the mapped distance over $\mathcal{X}^{(i)}$. To obtain α, firstly we need to calculate the integral in the left part of Eq. 3 with the given z; secondly we need to calculate the integral in the right part of Eq. 3 as a function of α; the α value can then be solved with Eq. 3.

The following section 5 describes in practice the algorithm to achieve the distance mapping proposed.

5 Algorithm of distance mapping

To calculate the integral in the left part of Eq. 3, we first need to construct the distribution function $f_{\boldsymbol{z}^{(j)}}$ using Machine Learning for functional estimates. The algorithm for distance mapping is explained as follows:

A.1 Draw as many samples as possible from $\boldsymbol{x}^{(j)}$ and $\boldsymbol{y}^{(j)}$ (random variables over $\mathbb{X}^{(j)}$);
A.2 Compute $\boldsymbol{z}^{(j)} = d^{(j)}(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})$;
A.3 Compute the histogram of $\boldsymbol{z}^{(j)}$;
A.4 Use a Machine Learning algorithm to learn this histogram: this creates an un-normalized version of $f_{\boldsymbol{z}^{(j)}}(t)$;
A.5 Compute the integral $f_{\boldsymbol{z}^{(j)}}(t)$ over its domain to obtain the normalizing constant $\boldsymbol{C}^{(j)}$;
A.6 Normalize the estimated function from 4. with the constant $\boldsymbol{C}^{(j)}$;
A.7 This yields a functional representation $g^{(j)}(t)$ of $f_{\boldsymbol{z}^{(j)}}(t)$ that behaves as an estimate of the PDF of $\boldsymbol{z}^{(j)}$;
A.8 We can finally integrate $g^{(j)}(t)$ from 0 to z (which was the given distance value) to obtain a value we denote β:

$$\beta = \int_0^z g^{(j)}(t) \approx \int_0^z f_{\boldsymbol{z}^{(j)}}(t)dt, \tag{4}$$

and this is also done numerically;
A.9 We assume the cumulative distribution function (CDF) of the Euclidean distances $\alpha = d^{(i)}(x, y)$ to be $F_{\boldsymbol{z}^{(i)}}(\alpha)$. Solving Eq. 3 now becomes:

$$\beta = \int_0^z g^{(j)}(t) \approx F_{\boldsymbol{z}^{(i)}}(\alpha), \tag{5}$$

$$\alpha = F^{-1}_{\boldsymbol{z}^{(i)}}(\beta), \tag{6}$$

where $F^{-1}_{\boldsymbol{z}^{(i)}}(\beta)$ is the inverse of the CDF in the mapped Euclidean space $\mathcal{X}^{(i)}$. The distances are then mapped from z to α.

Note that this algorithm is independent on the nature of $\mathcal{X}^{(i)}$: at this point, $\mathcal{X}^{(i)}$ can be any metric space. In the following, we look at the two possibilities of mapping a non-Euclidean space to a Euclidean space, or to another, non-Euclidean space (for completeness sake).

So, we are presented with two possibilities:

- $\mathcal{X}^{(i)} = (\mathbb{X}^{(i)}, d^{(i)})$ is the canonical Euclidean space, i.e. $\mathbb{X}^{(i)} = \mathbb{R}$ and $d^{(i)}$ is the Euclidean distance over $\mathbb{R}$;
- $\mathcal{X}^{(i)}$ is not the canonical Euclidean space, and the set of all necessary values $\mathbb{X}^{(i)}$ does not have to be $\mathbb{R}$, while $d^{(i)}$ is the Euclidean distance.

5.1 First case

In the case of $\mathcal{X}^{(i)}$ being the canonical Euclidean space, we can find analytical expressions for $f_{\boldsymbol{z}^{(i)}}$ in Eq. 3, by making assumptions on how the variables $\boldsymbol{x}^{(i)}$ and $\boldsymbol{y}^{(i)}$ are distributed. If such assumptions are not acceptable for some reason, it is always possible to revert to the estimation approach mentioned above, or possibly solve analytically as below for other well-know distributions.

If $\boldsymbol{x}^{(i)}$ and $\boldsymbol{y}^{(i)}$ are normally distributed We assume that $\boldsymbol{x}^{(i)}$ and $\boldsymbol{y}^{(i)}$ follow a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with mean μ and variance σ^2. $\boldsymbol{x}^{(i)}$ and $\boldsymbol{y}^{(i)}$ are iid.

It is then clear that $\boldsymbol{z}^{(i)} = d(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) = |\boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i)}|$ is distributed as a folded normal distribution of mean 0 and variance $2\sigma^2$: $\boldsymbol{z}^{(i)} \sim \mathcal{N}_f(0, 2\sigma^2)$. The probability density function of $\boldsymbol{z}^{(i)}$ can then be described as:

$$f_{\boldsymbol{z}^{(i)}}(t) = \frac{1}{\sigma\sqrt{\pi}} e^{-t^2/(4\sigma^2)}, \quad \text{for } t \geq 0 \tag{7}$$

Its CDF follows that

$$F_{\boldsymbol{z}^{(i)}}(\alpha) = \int_0^{\alpha} f_{\boldsymbol{z}^{(i)}}(t)dt = \operatorname{erf}(\frac{\alpha}{2\sigma}) \tag{8}$$

Finally, as we have calculated β in Eq. 5, we can solve easily

$$\alpha = 2\sigma \operatorname{erf}^{-1}(\beta). \tag{9}$$

If $\boldsymbol{x}^{(i)}$ and $\boldsymbol{y}^{(i)}$ are uniformly distributed If we assume that $\boldsymbol{x}^{(i)}$ and $\boldsymbol{y}^{(i)}$ follow a uniform distribution $\mathcal{U}(a, b)$, with $(a \leq b)$, and are iid. The probability distribution of the distances $\boldsymbol{z}^{(i)}$ is then obtained that

$$f_{\boldsymbol{z}^{(i)}}(t) = \begin{cases} \frac{2}{b-a}\left(1 - \frac{t}{b-a}\right) & \text{if } 0 \leq t \leq b-a \\ 0 & \text{elsewhere} \end{cases} \tag{10}$$

which means that

$$F_{\boldsymbol{z}^{(i)}}(\alpha) = \int_0^{\alpha} f_{\boldsymbol{z}^{(i)}}(t)dt = -\frac{1}{(b-a)^2}\alpha^2 + \frac{2}{b-a}\alpha. \tag{11}$$

Solving as before, we finally have that

$$\alpha = (b-a)(1 - \sqrt{1-\beta}), \tag{12}$$

given the fact that the other solution is not acceptable for our case (negative result for a distance value).

If the distances $|\boldsymbol{x}^{(i)}$-$\boldsymbol{y}^{(i)}|$ are Rayleigh distributed A Rayleigh distribution often arises when the metric space is analyzed by the magnitude of the orthogonal two-dimensional vector components. If we assume the distances $\boldsymbol{z}^{(i)} = |\boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i)}|$ follow a Rayleigh distribution with the scale parameter σ that

$$f_{\boldsymbol{z}^{(i)}}(t) = \frac{t}{\sigma^2} e^{-t^2/(2\sigma^2)}, \quad \text{for } \ t \geq 0. \tag{13}$$

The cumulative density then becomes

$$F_{\boldsymbol{z}^{(i)}}(\alpha) = \int_0^{\alpha} f_{\boldsymbol{z}^{(i)}}(t)dt = 1 - e^{-\alpha^2/(2\sigma^2)}. \tag{14}$$

The mapped distances α can then be solved by

$$\alpha = \sigma\sqrt{-2\log(1-\beta)}. \tag{15}$$

Other distributions We have discussed above the most common distributions of the distances $\boldsymbol{z}^{(i)}$ in the canonical Euclidean space $\mathcal{X}^{(i)}$. Certainly, the PDF of the distances $\boldsymbol{z}^{(i)}$ can exist in other less common fashions in certain specific circumstances. The implementation of the solution is not consummate so far. However, for the datasets used in practice with this work, which consist of timestamped GPS coordinates in the form of latitudes and longitudes, the discussed typical distributions are sufficient enough to illustrate the mapped distributions in the canonical Euclidean space. We will discuss about it in section 8.

5.2 Second case

In the second case, where $\mathcal{X}^{(i)}$ is not canonical Euclidean space, we basically have to perform the same estimate of $f_{\boldsymbol{z}^{(i)}}$ and its integral from 0 up to α, as we did for $f_{\boldsymbol{z}^{(j)}}$. The result is another function $G(\alpha)$:

$$G^{(i)}(\alpha) = \int_0^{\alpha} g^{(i)}(t)dt \approx \int_0^{\alpha} f_{\boldsymbol{z}^{(i)}}(t)dt. \tag{16}$$

We then have to solve numerically

$$G^{(i)}(\alpha) = \beta \tag{17}$$

for α.

6 Using ELM to learn the functional distribution

We propose to use Extreme Learning Machines (ELM) [5,6] as the mapping tool between distance functions. The reason for choosing this specific Machine Learning technique is its excellent performance/computational time ratio among all the techniques. The model is simple and involves a minimal amount of computations. Since we are dealing with the limit problem of the number of records N

to estimate the distribution $f_{\boldsymbol{z}^{(j)}}(d)$ (in Eq. 2), the ELM model is applicable in that it can learn the mapping in reasonable time for large amounts of data, if such a need arises. ELM is a universal function approximator, which can fit any continuous function.

The ELM algorithm was originally proposed by Guang-Bin Huang *et al.* in [6], and further developed, e.g. in [12,9,8], and analysed in [2]. It uses the structure of a Single Layer Feed-forward Neural Network (SLFN) [1]. The main concept behind the ELM approach is its random initialization, instead of a computationally costly procedure of training the hidden layer. The output weights matrix is then to be found between the hidden representation of the inputs and the outputs.

It works as following: Consider a set of N district observations ($\boldsymbol{x}_i$,$\boldsymbol{y}_i$), with $\boldsymbol{x}_i \in \mathbb{R}^d$, $\boldsymbol{y}_i \in \mathbb{R}^c$, and $i = 1, ..., N$. In the case the SLFN would perfectly approximates the data, with the errors between the estimated outputs $\hat{\boldsymbol{y}}_i$ and the actual outputs $\boldsymbol{y}_i$ being zeros: $\hat{\boldsymbol{y}}_i = \boldsymbol{y}_i$. The relation between inputs, weights and outputs is then:

$$\sum_{j=1}^{n} \beta_j \phi(\boldsymbol{w}_j \boldsymbol{x}_i + b_j) = \boldsymbol{y}_i, \quad i \in [1, N], \tag{18}$$

where $\phi : \mathbb{R}^d \to \mathbb{R}^c$ is the activation function of the hidden neurons; $\boldsymbol{w}_j$ is the input weights; b_j is the biases; and β_j is the output weights. Eq. 18 can also be written compactly as:

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{Y}, \tag{19}$$

with $\boldsymbol{\beta} = (\beta_1^T, ..., \beta_n^T)^T$, and $\boldsymbol{Y} = (\boldsymbol{y}_1^T, ..., \boldsymbol{y}_n^T)^T$.

The output weights $\boldsymbol{\beta}$ can be solved from the hidden layer representation of inputs $\boldsymbol{H}$ and the actual outputs $\boldsymbol{Y}$:

$$\boldsymbol{\beta} = \boldsymbol{H}^{\dagger}\boldsymbol{Y}, \tag{20}$$

where $\boldsymbol{H}^{\dagger}$ is the Moore-Penrose generalised inverse [11] of the matrix $\boldsymbol{H}$.

The ELM training does not require iterations, so the most computationally costly part is the calculation of a pseudo-inverse of the matrix $\boldsymbol{H}$. This makes ELM an extremely fast Machine Learning method. Thus, we propose to use ELM to learn the distribution of the pairwise distances over non-Euclidean spaces $f_{\boldsymbol{z}^{(j)}}(t)$ or $F_{\boldsymbol{z}^{(j)}}(t)$.

7 Implementation improvement

When implemented the mapping solution straightforwardly as in section 5, the algorithm spends most of the CPU time on calculating the integral of $f_{\boldsymbol{z}^{(j)}}(t)$ over the distances $\boldsymbol{z}^{(j)}$ numerically as in Eq. 4. This consumes lots of computational time. This is because the number of the pairwise distances $\boldsymbol{z}^{(j)}$ is $N(N-1)/2$, which can obviously grow to a very large value when the data size N increases. Thus, we avoided the integration calculations by using machine leaning to learn

the CDF $F_{\boldsymbol{z}^{(j)}}$, instead of learning the PDF $f_{\boldsymbol{z}^{(j)}}$ in **A.4**. This yields a function representation of $F_{\boldsymbol{z}^{(j)}}(t)$ (with the normalisation constant directly from $F_{\boldsymbol{z}^{(j)}}$). β can then be obtained straight from $F_{\boldsymbol{z}^{(j)}}(z)$.

The second most CPU consuming step in this algorithm is to find the most suitable described distribution of $f_{\boldsymbol{z}^{(i)}}(t)$ or $F_{\boldsymbol{z}^{(i)}}(t)$ in the Euclidean space $\mathcal{X}^{(i)}$. To choose whether the non-Euclidean distribution should best be mapped to a Normal, Uniform, Rayleigh, or other distribution, we have to fit the $F_{\boldsymbol{z}^{(j)}}(t)$ to those well defined canonical Euclidean distances distributions and find the optimised parameters in the best suitable distribution with the least errors.

Again, if we use the pairwise distances $\boldsymbol{z}^{(j)}$ in $F_{\boldsymbol{z}^{(j)}}$ directly, the fitting computation is very heavy as we are trying to fit the data with $N(N-1)/2$ points. To make it easy, we use the functional representation of $F_{\boldsymbol{z}^{(j)}}(t)$ with the user-defined distances in the pre-defined domain, with the purpose only to find the best distribution and its parameters (the functional presentation of $F_{\boldsymbol{z}^{(i)}}(t)$). Then the mapped distance α can be obtained from Eq. 6 with the calculated β and the inverse functional representation of $F_{\boldsymbol{z}^{(i)}}(t)$.

In the following section 8, we present results over the typical data used for this work, GPS traces (latitude and longitude).

8 Experimental results

We have tested the proposed mapping algorithm by mapping the pairwise distances in the dataset of GPS coordinates in the form of latitudes and longitudes, which is shown as the trajectory in Fig. 1. Assume we have a dataset $\boldsymbol{X} = [\boldsymbol{x_1}, ..., \boldsymbol{x_N}]^T$ to depict the trajectory of one specific person, where the attributes of each record $\boldsymbol{x_i}$ explain the locations by latitude and longitude coordinates at the corresponding time $\boldsymbol{t_i}$.

Note that the metric space of the GPS coordinates $\mathcal{X}^{(\boldsymbol{gps})} = (\mathbb{X}^{(\boldsymbol{gps})}, d^{(\boldsymbol{gps})})$ is a non-Euclidean space, because the distance $d^{(\boldsymbol{gps})}$ of two GPS coordinates $(\boldsymbol{lat}, \boldsymbol{lon})$ is the shortest route between the two points on the Earth's surface, namely, a segment of a great circle.

We first explore the limit condition on the number of records N in Eq. 2, in that N needs to be sufficiently large to possibly estimate $f_{z^{(j)}}$(or $F_{z^{(j)}}$) to be close enough to $f_{z^{(i)}}$ (or $F_{z^{(i)}}$). We test on experimental datasets with various $N = 10, 30, 100, 1000$, within which each location record is randomly chosen along the introduced trajectory in Fig. 1.

Fig. 2 illustrates the comparisons of the CDF $F_{\boldsymbol{z}^{(j)}}(d)$ of the pairwise distances obtained from $\mathcal{X}^{(\boldsymbol{gps})} = (\mathbb{X}^{(\boldsymbol{gps})}, d^{(\boldsymbol{gps})})$, and the CDF $F_{\boldsymbol{z}^{(i)}}(d)$ of the mapped distances in the Euclidean space, with $N = 10, 30, 100, 1000$ for the four subplots respectively.

It is clear to see that, in this specific simple case, with small N values of 10 and 30, there exists comparable disagreements of the CDF distributions between $\mathcal{X}^{(\boldsymbol{gps})}$ and the mapped Euclidean space. Meanwhile, with the larger N values of 100 and 1000, the limit condition on N is well satisfied, as it is plain to see that the non-Euclidean GPS metric $\mathcal{X}^{(\boldsymbol{gps})}$ behaves over its non-Euclidean space,

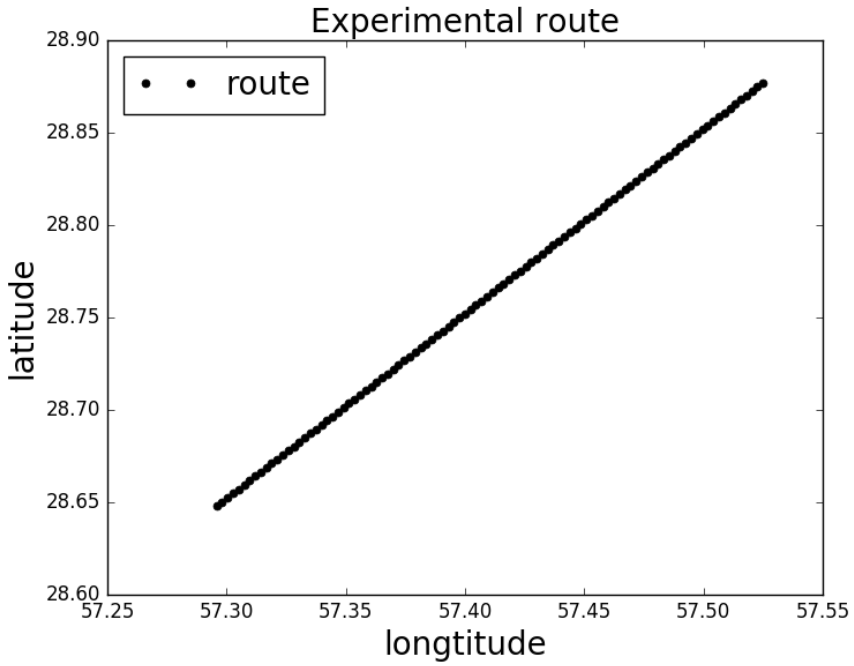

Fig. 1. Experimental dataset of locations in GPS coordinates with the form of latitudes and longitudes.

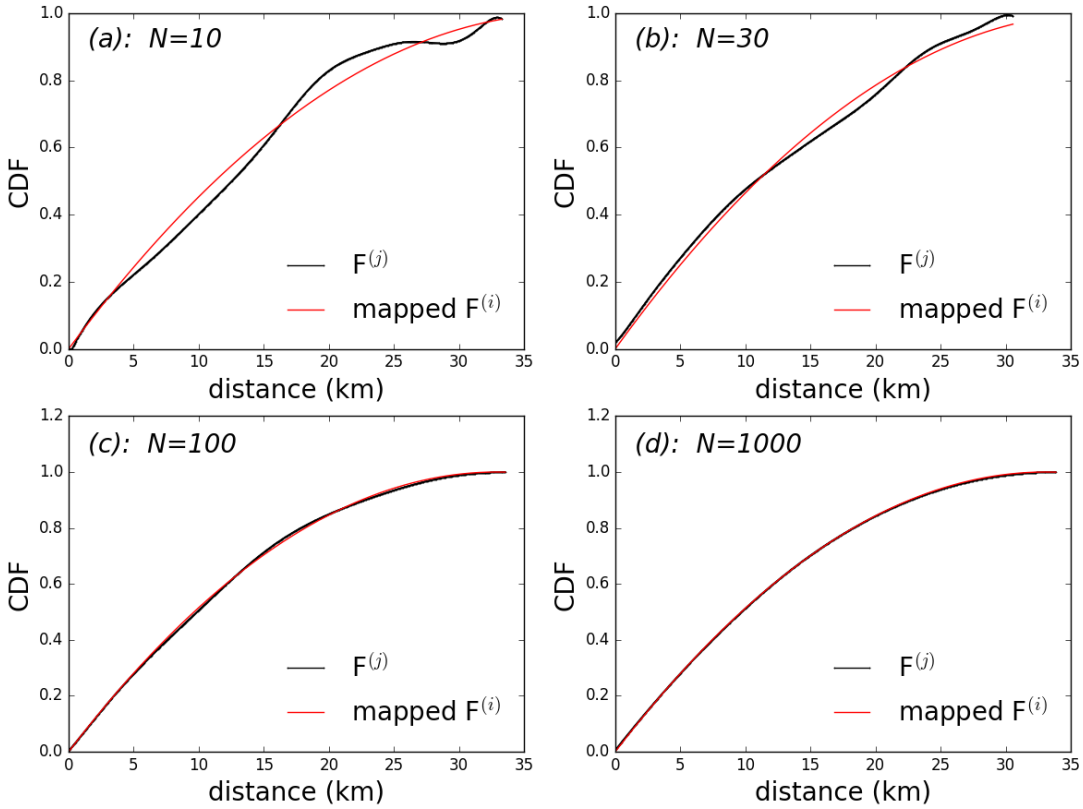

Fig. 2. The CDFs $F_{\boldsymbol{z}^{(j)}}$ of the pairwise distances in the experimental dataset, and its corresponding mapped CDFs $F_{\boldsymbol{z}^{(i)}}$, with $N = 10, 30, 100, 1000$.

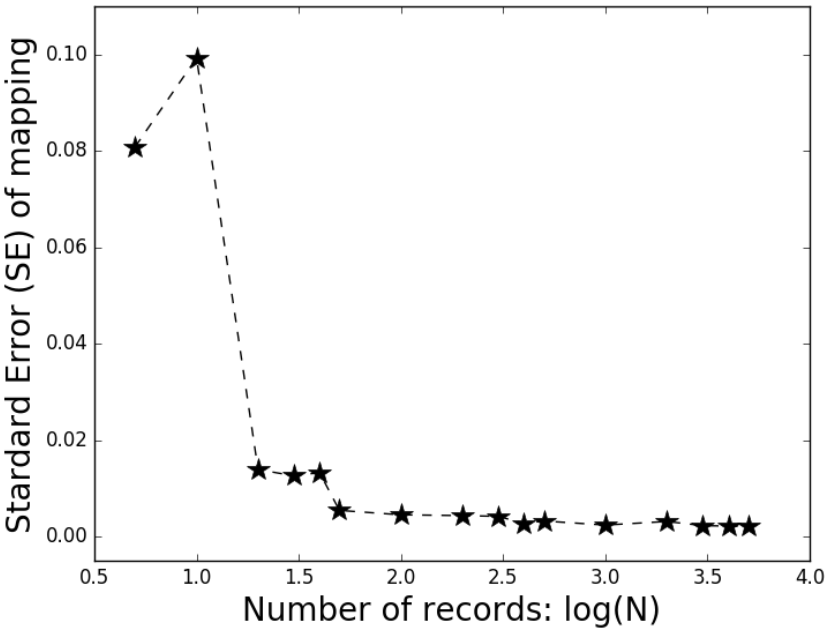

Fig. 3. The standard errors of the mapped distributions $F_{\boldsymbol{z}^{(i)}}$ VS. number of records n.

accurately close to the mapped Euclidean metric over the mapped Euclidean space.

Thus, we can see that the number of record $N = 100$ is sufficient to closely estimate the distribution $f_{\boldsymbol{z}^{(j)}}(d)$ to $f_{\boldsymbol{z}^{(i)}}(d)$ in this very simple case. The Standard Errors (SE) of the mapped distribution $f_{\boldsymbol{z}^{(i)}}$ are calculated, meanwhile selecting meticulously denser and broader N values from 5 to 5000, along the specific route. Fig. 3 shows the SEs of the mapped distribution with the dependence on N. As the latitude and longitude coordinates are linearly altering in this simple case, the distances are mapped to a uniform distribution straightforwardly. The SE of the mapped $f_{\boldsymbol{z}^{(i)}}(d)$ converges closely to 0 at very small $N \simeq 50$.

9 Conclusion

We have developed and implemented a distance-mapping algorithm, which projects a non-Euclidean space to a canonical Euclidean space, in a way that the pairwise distances distributions are approximately preserved. The mapping algorithm is based on the assumptions that both spaces are actual metric spaces, and that the number of records N is large enough to estimate the pairwise distances $f_{z^{(j)}}$(or $F_{z^{(j)}}$) to be close enough to $f_{z^{(i)}}$ (or $F_{z^{(i)}}$). We have tested our algorithm by illustrating the distance mapping of an experimental dataset of GPS coordinates. The limitation condition of N is discussed by the comparison of $F_{z^{(j)}}$ and its mapped $F_{z^{(i)}}$, using various N values. The standard errors of the mapped distance distribution $F_{z^{(i)}}$ is also analyzed with various N.

Our distance mapping algorithm is performed with the most common canonical Euclidean distance distributions. Certainly, less common distributions are needed to be implemented as well, and might require specific adjustments. More diversified experimental examples are needed for completeness.

References

1. Peter Auer, Harald Burgsteiner, and Wolfgang Maass. A learning rule for very simple universal approximators consisting of a single layer of perceptrons. *Neural networks*, 21(5):786–795, 2008.
2. Erik Cambria, Guang-Bin Huang, Liyanaarachchi Lekamalage Chamara Kasun, Hongming Zhou, Chi Man Vong, Jiarun Lin, Jianping Yin, Zhiping Cai, Qiang Liu, Kuan Li, et al. Extreme learning machines [trends & controversies]. *IEEE Intelligent Systems*, 28(6):30–59, 2013.
3. Thomas M Cover. Elements of information theory. 1991.
4. George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.
5. Guang-Bin Huang, Lei Chen, Chee Kheong Siew, et al. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans. Neural Networks*, 17(4):879–892, 2006.
6. Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.

7. Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6), 2004.
8. Yoan Miche, Antti Sorjamaa, Patrick Bas, Olli Simula, Christian Jutten, and Amaury Lendasse. Op-elm: optimally pruned extreme learning machine. *IEEE Transactions on Neural Networks*, 21(1):158–162, 2010.
9. Yoan Miche, Mark Van Heeswijk, Patrick Bas, Olli Simula, and Amaury Lendasse. Trop-elm: a double-regularized elm using lars and tikhonov regularization. *Neurocomputing*, 74(16):2413–2421, 2011.
10. Dávid Pál, Barnabás Póczos, and Csaba Szepesvári. Estimation of rényi entropy and mutual information based on generalized nearest-neighbor graphs. In *Advances in Neural Information Processing Systems*, pages 1849–1857, 2010.
11. Calyampudi Radhakrishna Rao and Sujit Kumar Mitra. Generalized inverse of matrices and its applications. 1971.
12. Mark Van Heeswijk, Yoan Miche, Erkki Oja, and Amaury Lendasse. Gpu-accelerated and parallelized elm ensembles for large-scale regression. *Neurocomputing*, 74(16):2430–2437, 2011.