Overlaying Spaces and Practical Applicability of Complex Geometries

Kirill Shevkunov
Yandex
Moscow Institute of Physics and Technology
Moscow, Russia
shevkunov.ks@phystech.edu

Liudmila Prokhorenkova
Yandex
Moscow Institute of Physics and Technology
Higher School of Economics
Moscow, Russia
ostroumova-la@yandex.ru

Abstract

Recently, non-Euclidean spaces became popular for embedding structured data. Following hyperbolic and spherical spaces, more general *product spaces* have been proposed. However, searching for the best configuration of a product space is a resource-intensive procedure, which reduces the practical applicability of the idea. We introduce a novel concept of *overlaying spaces* that does not have the problem of configuration search and outperforms the competitors in structured data embedding tasks when the aim is to preserve all distances. For local loss functions (e.g., for ranking losses), the *dot-product* similarity, which is often overlooked in graph embedding literature since it cannot be converted to a metric, outperforms all metric spaces. We discuss advantages of the dot product over proper metric spaces.

1 Introduction

Building vector representations of various objects is one of the central tasks of machine learning. Word embeddings such as Glove [18] and Word2Vec [15] are widely used in natural language processing, a similar Prod2Vec [7] approach is used in recommendation systems. There are many algorithms proposed for graph embeddings, e.g., Node2Vec [8] and DeepWalk [19]. Recommendation systems often construct embeddings of a bipartite graph that describes interactions between users and items. Such embeddings can be constructed via matrix factorization techniques such as ALS [10].

For a long time, embeddings were considered exclusively in $\mathbb{R}^n$, but hyperbolic space was shown to be more suitable for graph and word representations due to the underlying hierarchical structure [16, 17, 22]. Going beyond spaces of constant curvature, a recent study [9] proposes *product spaces*, which combine several copies of Euclidean, spherical and hyperbolic spaces. While these spaces demonstrate promising results, the optimal signature (types of combined spaces and their dimensions) has to be chosen via brute force, which may not be acceptable in large-scale applications.

In this paper, we propose a more general metric space called *overlaying space* together with an optimization algorithm that trains signature *simultaneously* with embedding allowing to avoid brute-forcing. We provide extensive empirical evaluation to see whether complex non-Euclidean spaces are useful in practice. For this purpose, we first consider the graph reconstruction task with both distortion loss and a more realistic ranking loss. We also apply the proposed methods to train embeddings via DSSM [11] to compare the spaces in information retrieval and recommendation tasks. We conclude that the proposed overlaying space outperforms the competitors in the graph reconstruction task with distortion loss, i.e., when the aim is to embed data preserving the distances. On the other hand, when ranking losses are optimized and if the dimensionality is sufficiently large, the best results are achieved with the dot-product similarity. Dot products are often overlooked in graph embedding literature since they cannot be converted to a metric. Our experiments show that despite this shortcoming, dot products provide good-quality embeddings. We try to explain this and discuss the advantages of the dot-product similarity compared to metric spaces.

2 Background and related work

2.1 Embeddings and loss functions

For a graph $G = (V, E)$, an embedding is a mapping $f : V \to U$, where U is a metric space equipped with a distance $d_U : U \times U \to \mathbb{R}_+$.[1] On the graph one can consider a shortest path distance $d_G : V \times V \to \mathbb{R}_+$. In the graph reconstruction task, it is expected that a good embedding preserves the original graph distances: $d_G(v, u) \approx d_U(f(v), f(u))$. The most commonly used evaluation mertic is *distortion*, which averages relative errors of distance reconstruction over all pairs of nodes:

$$D_{avg} = \frac{2}{|V|(|V|-1)} \sum_{(v,u)\in V^2, v\neq u} \frac{|d_U(v,u) - d_G(f(v), f(u))|}{d_G(v,u)} \,. \tag{1}$$

While commonly used in graph reconstruction, distortion is not the best choice for many practical applications. For example, in recommendation tasks, one usually deals with a partially observed graph (some positive and negative element pairs), so a huge graph distance between nodes in the observed part does not necessarily mean that the nodes are not connected by a short path in the full graph. Also, often only the order of the nearest elements is important while predicting distances to faraway objects is not critical. In such cases, it is more reasonable to consider a local ranking metric, e.g., the mean average precision (mAP) that measures the relative closeness of the relevant (adjacent) nodes compared to the others:[2]

$$\begin{gathered} \mathrm{mAP} = \frac{1}{|V|} \sum_{v \in V} \mathrm{AP}(v) = \frac{1}{V} \sum_{v \in V} \frac{1}{\deg(v)} \sum_{u \in N_v} \frac{|N_v \cap R_v(u)|}{|R_v(u)|} \,, \\ R_v(u) = \{w \in V | d_U\big(f(v), f(w)\big) \leq d_U\big(f(v), f(u)\big)\} \,, \;\; N_v = \{w \in V | (v,w) \in E\} \,. \end{gathered} \tag{2}$$

Note that mAP cannot be directly optimized since it is not differentiable. In our experiments, we use the following probabilistic loss function as a proxy:[3]

$$L_{proxy} = - \sum_{(v,u)\in E} \log \mathrm{P}((v,u) \in E) = - \sum_{(v,u) \in E} \log \frac{\exp(-d_U(f(v), f(u)))}{\sum\limits_{w \in V} \exp(-d_U(f(v), f(w)))} \,. \tag{3}$$

Note that when substituting $d_U(x, y) = c - f(x)^T f(y)$ (assuming that $f(x) \in \mathbb{R}^n$, so the dot product is defined), we get the standard word2vec loss function.

2.2 Spaces, distances and similarities

In the previous section, we assumed that $d_U : U \times U \to \mathbb{R}_+$ is an arbitrary distance. In this section, we discuss particular choices often assumed in the literature. For many years, Euclidean space was the primary choice for structured data embeddings [6]. For two points $x, y \in \mathbb{R}^d$, Euclidean distance is defined as

$$d_E(x,y) = \left(\sum_{i=1}^{d} (x_i - y_i)^2 \right)^{1/2} \,.$$

Spherical spaces were also found to be suitable for some applications [14, 20, 26]. Indeed, in practice, vector representations are often normalized, so cosine similarity between vectors is a natural way to measure their similarity. This naturally corresponds to a spherical space $S_d = \{x \in \mathbb{R}^{d+1} : \|x\|_2^2 = 1\}$ equipped with a spherical distance:

$$d_S(x, y) = \arccos(x^T y) \,.$$

[1]Note that any discrete metric space correspond to a weighted graph, so graph terminology is not restrictive.

[2]For mAP, the relevance labels are assumed to be binary (unweighted graphs). If a graph is weighted, then we say that N_v consists of the closest element to v (or several closest elements if the distances to them are equal).

[3]We have also experimented with other way to convert distance to probability, see the supplementary materials for more details.

In recent years, hyperbolic spaces also started to gain popularity. Hyperbolic embeddings have shown their superiority over Euclidean ones in a number of tasks, such as graph reconstruction and word embedding [16, 17, 21, 22]. To represent the points, early approaches were based on the Poincare model of the hyperbolic space [16], but later it has been shown that the hyperboloid (Lorentz) model may lead to more stable results [17]. In this work, we also adopt the hyperboloid model: $H_d = \{x \in \mathbb{R}^{d+1} | \langle x, x \rangle_h = 1, x_1 > 0\}$ and the distance is defined as

$$d_H = \operatorname{arccosh}(\langle x, y \rangle_h) \,, \quad \text{where} \quad \langle x, y \rangle_h := x_1 y_1 - \sum_{i=2}^{d+1} x_i y_i \,. \tag{4}$$

Going even further, a recent paper [9] proposes more complex *product spaces* than combine several copies of Euclidean, spherical, and hyperbolic spaces. Namely, the overall dimension d is split into k parts (smaller dimensions): $d = \sum_{i=1}^{k} d_i$, $d_i > 0$. Each part is associated with a space $D_i \in \{E_{d_i}, S_{d_i}, H_{d_i}\}$ and scale coefficient $w_i \in \mathbb{R}_+$. Varying scale coefficients corresponds to changing curvature of hyperbolic/spherical space, while in Euclidean space this coefficient is not used ($w_i = 1$). Then, the distance in the product space is defined as:

$$d_P(x, y) = \sqrt{\sum_{i=1}^{k} w_i d_{D_i}(x[t_{i-1} + 1 : t_i], y[t_{i-1} + 1 : t_i])^2} \,,$$

where $t_0 = 0$, $t_i = t_{i-1} + d_i$, and $x[s : e]$ is a vector $(x_s, \ldots, x_e) \in \mathbb{R}^{e-s+1}$. If $k = 1$, we get a standard Euclidean, spherical or hyperbolical space. In [9], it is proposed to simultaneously learn an embedding and scale coefficients w_i. However, choosing the optimal signature (how to split d into d_i and which types of spaces to choose) is a challenge. A heuristics proposed in [9] allows to guess types of spaces if d_i's are given. If $d_1 = d_2 = 5$, this heuristics agrees well with the experiments on three considered datasets. Generalizability of this idea to other datasets and configurations is unclear. In addition, it cannot be applied if a dataset is partially observed (e.g., there are several known positive-negative pairs), i.e., graph distances cannot be computed. Hence, in practice it is more reliable to choose a signature via the brute force which can be inapplicable on large datasets.

Another way to measure objects' similarity, which is usually overlooked in embedding literature but is often used in practical applications, is via the dot product of vectors $x^T y$. In this paper, we stress that the dot-product similarity has some advantages over other spaces. In particular, it allows us to easily differentiate between more popular and less popular items (the vector norm can be considered as a measure of popularity). This feature is usually attributed to hyperbolic spaces, but it better agrees with the dot-product similarity. The main shortcoming of the dot product is the fact that it does not correspond to a metric, however, it may be used to predict similarity or dissimilarity between objects, which is often sufficient in practice, and in some cases is able to preserve the distances.

2.3 Optimization

Gradient optimization in Euclidean space is straightforward, while for spherical or hyperbolic embeddings, we have to additionally control that points belong to a surface. In previous works, Riemann-SGD was used to solve this problem [2]. In short, it projects Euclidean gradients on the tangent space at a point, and then uses a so-called exponential map to move the point along the surface according to the gradient projection. For product spaces, a generalization of exponential map has been proposed [5, 23].

In [25], the authors compare RSGD with the retraction technique (points are moved along the gradients in the ambient space and are projected onto the surface after each update). From their experiment, the retraction technique requires from 2% to 46% more iterations, depending on the learning rate. However, the exponential update step takes longer, hence the advantage of RSGD in terms of computation time depends on the specific implementation.

3 Overlaying spaces

In this section, we propose a new concept of *overlaying spaces*. This concept generalizes product spaces and also allows us to make signature (types of combined spaces) trainable. Our main idea is to divide the embedding vector into several *intersecting* (unlike product spaces) segments, each segment corresponds to its own space. Then, instead of discrete signature brute-forcing, we optimize the weights of the signature elements.

Importantly, we allow the same coordinates of an embedding vector to define distances in spaces of different geometry. For this purpose, we need to map a vector $x \in \mathbb{R}^d$ (for any $d \geq 2$) to a point in Euclidean, hyperbolic and spherical space. Let us denote this mapping by M. Obviously, for Euclidean space, we may take $M_E(x) = x$. For hyperbolic and spherical spaces, we set

$$M_S(x) = \frac{x}{|x|} \in S_{d-1}; M_H(x) = \left(\sqrt{1 + \sum_{i=2}^{d} x_i^2}, x_1, \ldots, x_d\right) \in H_d. \tag{5}$$

Note that a d-dimensional vector x is mapped into Euclidean and hyperbolic spaces of dimension d and into a spherical space of dimension $d-1$. While it is possible to parameterize points in S_d by d-dimensional vectors, the most straightforward mapping usually used in practice is the one in (5).[4]

Now we are ready to define an overlaying space. Consider two vectors $x, y \in \mathbb{R}^d$. Let $p_1, \ldots, p_k$ denote some subsets of coordinates, i.e., $p_i \subset \{1, \ldots, d\}$. We assume that together the subsets cover all coordinates, i.e., $\cup_{i=1}^k p_i = \{1, \ldots, d\}$. By $x[p_i]$ we denote a subvector of x induced by p_i. Let $D_i \in \{E, S, H\}$. We define $d_i(x,y) = d_{D_i}\big(M_{D_i}(x[p_i]), M_{D_i}(y[p_i])\big)$ and aggregate these distances with arbitrary positive weights $w_1 \ldots w_k \in \mathbb{R}_+$:

$$\begin{aligned} d_O^{l0}(x,y) &= \max\big(w_1 d_1(x,y), \ldots, w_k d_k(x,y)\big)\,, \\ d_O^{l1}(x,y) &= \sum_{i=1}^k w_i d_i(x,y)\,,\ d_O^{l2}(x,y) = \left(\sum_{i=1}^k w_i d_i^2(x,y)\right)^{1/2}\,. \end{aligned} \tag{6}$$

The obtained space equipped with distance d_O^{l0}, d_O^{l1}, or d_O^{l2} we call an overlaying space. It is defined by p_i, D_i, and w_i. Note that it is sufficient to assume that spherical and hyperbolic spaces have curvatures 1 and -1, respectively, since changing curvature is equivalent to changing scale which is captured by w_i. The following statement follows from the definition above and from the fact that d_E, d_S, and d_H are distances.

Statement 1 *If $\cup_{i=1}^k p_i = \{1, \ldots, d\}$ and $w_1 \ldots w_k \in \mathbb{R}_+$, then d_O^{l0}, d_O^{l1}, d_O^{l2} are distances on $\mathbb{R}^d \times \mathbb{R}^d$, i.e., they satisfy the metric axioms.*

It is easy to see that overlaying spaces generalize product spaces. Indeed, if we assume $p_i \cap p_j = \emptyset$ for all $i \neq j$, then an overlaying space reduces to a product space. However, the fact that we allow $p_i \cap p_j \neq \emptyset$ gives a larger expressive power for the same dimension d.

4 Optimization in overlaying spaces

4.1 Universal signature

Overlaying spaces defined in the previous section are flexible and allow capturing various geometries. However, similarly to product spaces, they need a signature (p_i and D_i) to be chosen in advance. In this section, we show that a universal signature can be chosen, so no brute force is needed to choose the best signature for a particular dataset.

Let $t \geq 0$ denote the depth (complexity) of the signature for a d-dimensional embedding. Each layer l, $0 \leq l \leq t$, of the signature consists of 2^l subsets of coordinates:

$$p_i^l = \left\{\left[d(i-1)/2^l\right] + 1, \ldots, \left[di/2^l\right]\right\}\,,\quad 1 \leq i \leq 2^l\,.$$

Each p_i^l is associated with Euclidean, spherical and hyperbolic spaces simultaneously. The corresponding weights are denoted by $w_i^{l,E}, w_i^{l,S}, w_i^{l,H}$. Then, the distance is computed according to (6), see Figure 1 for an illustration of the procedure (for $d = 10$ and $t = 1$). Informally, we first consider the original vectors x, y and compute Euclidean, spherical and hyperbolic distances between them.

[4]For instance, [9] uses $d+1$ dimensional vectors for storing points in both S_d and H_d.

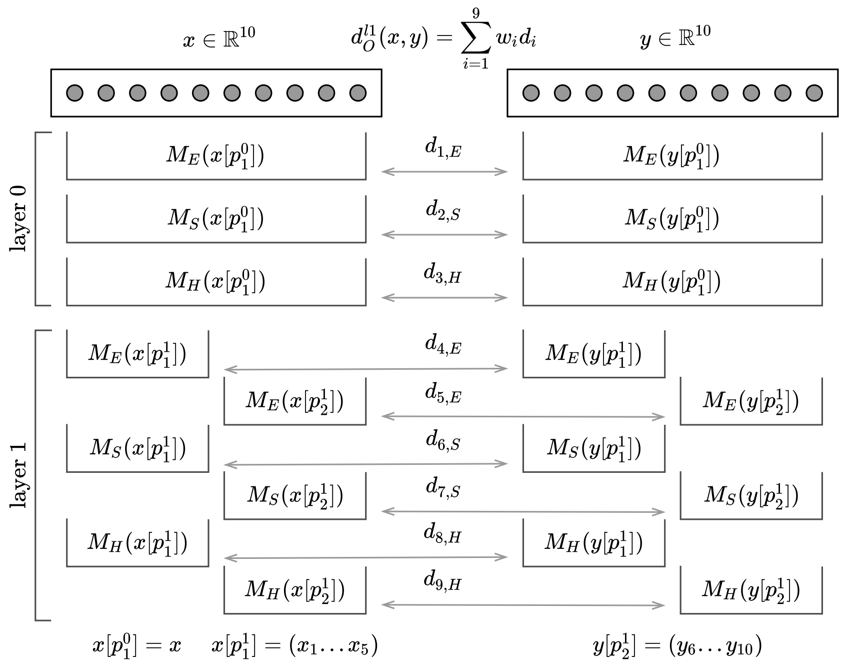

Figure 1: Example for $d = 10$, $t = 1$ and $l1$ (sum) aggregation

Then, we split the vectors into two halves and for each half we also compute all three distances, etc. Finally, all the obtained distances are averaged with the weights coefficient according to (6). Note that we have $3(2^{t+1} - 1)$ different weights in our structure in general, but with $l2$-aggregation this value may be reduced to $2(2^{t+1} - 1) + 2^t$ since for the Euclidean space the distances between sub-vectors at the upper layers can be split into terms corresponding to smaller sub-vectors, so we essentially need only the last layer with 2^t terms. Recall that in product spaces the weights correspond to curvatures of the combined spaces. In our case, they also play another important role: weights allow us to balance between different spaces. Indeed, for each subset of coordinates, we simultaneously compute the distance between the points assuming each of the combined spaces. Varying the weights, we can increase or decrease the contribution of a particular space to the distance. As a result, our signature allows us to learn the optimal signature, which does not have to be a product space since all weights can be non-zero.

4.2 Optimization

In this section, we describe how we embed into the overlaying space. Although Riemann-SGD (see Section 2.3) is a good solution from the theoretical point of view, in practice, due to errors in storing and processing real numbers, it may cause some problems. A point that we assume to lie on a surface (sphere or hyperboloid) does not numerically lie on it usually. Due to the accumulation of numerical errors, with each iteration of RSGD, the point may move away from the surface. Therefore, in practice, after each step, all embeddings are explicitly projected onto the surface, which may slow down the algorithm. Moreover, RSGD is not applicable if one needs to process the output of a neural network, which cannot be required to belong to a given surface (e.g., to satisfy $\langle x, x \rangle_h = 1 \Leftrightarrow x \in H_{n-1}$). As a result, before finding the hyperbolic distance between two outputs of a neural network in Siamese [3] setup, one first needs to somehow map them to a hyperboloid.

Instead of RSGD, we store the embedding vectors in Euclidean space and calculate distances between them using the mappings (5) to the corresponding surfaces. Thus, we are able to evaluate the distances between the outputs of neural networks and also use conventional optimizers. To optimize embeddings, we first map Euclidean vectors into the corresponding spaces, calculate distances and loss function, and then backpropagate through projection functions. To improve the convergence, we use Adam [12] instead of the standard SGD. Applying this to product spaces, we achieve the results

Table 1: Datasets for graph reconstruction

	UCSA312	CS PhDs	Power	Facebook	WLA6
Nodes	312	1025	4941	4039	3227
Edges	48516 (weighted)	1043	6594	88234	3604

similar to the original paper [9] (see Table 1 of the supplementary materials), where RSGD was used with the learning rate brute-forcing, custom learning rate for curvature coefficients, and other tricks.

5 Experiments

5.1 Graph reconstruction

To compare with previous research, we start with the graph reconstruction task with distortion loss (1). The goal is to embed all nodes of a given graph into a d-dimensional space approximating the pairwise graph distances between the nodes. Similarly to [9], we use the USCA312 dataset of distances between North American cities [4] (weighted complete graph), graph of computer science Ph.D. advisor-advisee relationships [1], a power grid distribution network with backbone structure [24], and a dense social network from Facebook [13]. We also created a new dataset, obtained by launching the breadth-first search on the Wikipedia category graph, starting from the "Linear Algebra" category with search depth limited to 6. Further, we refer to this dataset as WLA6 [5] and we expect it to be well described by a hyperbolic geometry due to its hierarchical structure.

We compare all spaces discussed in the paper: standard Euclidean, hyperbolic and spherical spaces (with trainable curvature); product spaces with all signatures from [9]; the proposed overlaying space; and also two dot-product-based distances. For the overalying space, we take $t = 0$ which gives a weighted combination of Euclidean, hyperbolic and spherical distances, and $t = 1$ where one more layer is added (see Figure 1). For $t = 1$ we compare $l1$ and $l2$ aggregations. For the dot-product-based distances, we consider $d(x, y) = c - x^T y$ and $d(x, y) = c \exp(-x^T y)$ with trainable parameter $c \in \mathbb{R}$. While these functions are not distances (do not satisfy the metric axioms), we add them to analyze whether they are still able to approximate graph distances. Similarly to [9], we fix the dimension $d = 10$. However, for a fair comparison, we fix the number of *stored values* for each embedding. In our case, this means that dimension of a spherical space is smaller by 1 (S_9 or $S_4 \times S_4$), since for the each spherical space we store one additional value (see (5)).[6] All models are trained to minimize distortion (1). The code of our experiments supplements the submission [7].

The results are shown in Table 8. It can be clearly seen that the overlaying spaces outperform other metric spaces, and the best overlaying space (among considered) is the one with $l1$ aggregation and complexity $t = 1$. Interestingly, the performance of such overlaying space is often better than for the *best* product space. Recall that we also added to the comparison the dot-similarity-based functions $c -$ dot and $ce^{-\text{dot}}$. These functions are not proper distances, hence their performance is highly unstable for this task: for example, for UCSA312 dataset the obtained distortion is orders of magnitude worse than the best one. However, on some datasets (Facebook and WLA6) the performance is quite good and for Facebook $ce^{-\text{dot}}$ has much better performance than all other solutions. We conclude that for the graph reconstruction with distortion loss the dot products are worth trying, but their performance is very unstable, in contrast to overlaying spaces that show good and stable results on all datasets.

As discussed in Section 2.1, in many practical applications, only the order of the nearest neighbors matters. In this case, it is more reasonable to use mAP (2). In previous work [9], mAP was also reported but the models were trained to minimize distortion. In our experiments, we observed that distortion optimization weakly correlates with mAP optimization. Hence, we minimize the proxy-loss defined in equation (7). The results are shown in Table 9 and the obtained values for mAP are indeed much better than the ones obtained with distortion optimization [9], i.e., it is important to use

[5]The dataset will is publicly available.

[6]In the supplementary materials we evaluate spherical spaces without this reduction to compare with [9].

[7]https://github.com/shevkunov/Overlaying-Spaces-and-Practical-Applicability-of-Complex-Geometries

[8]Differences between the best and the second results on UCSA312 and WLA6 are not statistically significant.

Table 2: Graph reconstruction with distortion loss, top three results are highlighted.[8]

Signature	UCSA312	CS PhDs	Power	Facebook	WLA6
E_{10}	**0.00318**	0.0475	0.0408	0.0487	0.0530
H_{10}	0.01114	0.0443	0.0348	0.0483	0.0279
S_9	0.00986	0.0524	0.0481	0.0597	0.0666
$H_5^2 \equiv H_5 \times H_5$	0.00573	0.0345	**0.0255**	0.0372	0.0279
$S_4^2 \equiv S_4 \times S_4$	0.00753	0.0543	0.0505	0.0633	0.0727
$H_5 \times S_4$	0.00652	0.0346	**0.0255**	0.0336	0.0308
H_2^5	0.00592	**0.0344**	0.0273	0.0439	0.0356
S_1^5	0.00758	0.0761	0.0716	0.0990	0.1231
$H_2^2 \times E_2 \times S_1^2$	0.00383	0.0395	0.0335	0.0577	0.0474
$c-\text{dot}$	0.04005	0.0412	0.0461	**0.0236**	0.0296
$ce^{-\text{dot}}$	0.08306	0.0424	0.0505	**0.0192**	**0.0270**
$O_{l1}, t=0$	**0.00356**	0.0368	0.0281	0.0458	0.0286
$O_{l1}, t=1$	**0.00330**	**0.0300**	**0.0231**	0.0371	**0.0272**
$O_{l2}, t=1$	0.00530	**0.0328**	**0.0246**	**0.0324**	**0.0278**

Table 3: Graph reconstruction with mAP ranking loss, top three results are highlighted

Signature	UCSA312	CS PhDs	Power	Facebook	WLA6
E_{10}	0.9290	0.9487	0.9380	0.7876	0.7199
H_{10}	0.9173	0.9399	0.9385	0.7997	0.9617
S_9	0.9271	0.9586	0.9481	0.7795	0.7200
H_5^2	0.9247	0.9481	0.9415	0.8084	0.9682
S_4^2	0.9178	0.9613	0.9517	0.7706	0.7109
$H_5 \times S_4$	0.9274	0.9647	0.9524	0.8005	**0.9770**
H_2^5	0.9364	0.9671	0.9508	0.7979	0.8597
S_1^5	0.9311	0.9013	0.8101	0.7132	0.4957
$H_2^2 \times E_2 \times S_1^2$	0.9343	0.9504	0.9397	0.7690	0.5876
$c-\text{dot}$	**1**	**1**	**0.9983**	**0.8745**	**0.9990**
$O_{l1}, t=0$	**0.9522**	0.9879	0.9728	0.8093	0.6759
$O_{l1}, t=1$	**0.9522**	**0.9904**	**0.9762**	**0.8185**	0.9598
$O_{l2}, t=1$	**0.9522**	**0.9938**	**0.9907**	**0.8326**	**0.9694**

an appropriate loss function. According to Table 9, among the metric spaces, the best results are achieved with the overlaying spaces (especially for $l2$-aggregation with $t=1$). However, in contrast to distortion loss, ranking based on the dot-product clearly outperforms all metric spaces. This result is important from a practical point of view: there is no need to use complex geometries if the goal is to preserve the local neighborhood.

5.2 DSSM with custom distances

From a practical point of view, it is much more interesting to analyze whether an embedding is able to generalize to unseen examples. For instance, an embedding can be made via a neural network based on objects' characteristics, such as text descriptions or images. In this section, we analyze whether it is reasonable to use complex geometries in such a scenario. For this purpose, we trained a classic DSSM model[9] [11] on a private Wikipedia search dataset consisting of 491044 pairs (search query, relevant page), examples are given in Table 4. All queries are divided into train, validation, and test sets and for each signature the optimal iteration was selected on validation.

Table 5 compares all models for two embedding sizes: short of length 10 and "industrial size" of length in 256. For short embeddings, we see that a product space based on spherical geometry is

[9]We changed dense layers sizes in order to achieve required embedding length and used more complex text tokenization with char bigrams, trigrams and words, instead of just char trigrams.

Table 4: Search query examples

Query	**Web site**
Kris Wallace	en.wikipedia.org/wiki/Chris_Wallace
1980: Mitsubishi produces one million cars...	en.wikipedia.org/wiki/Mitsubishi_Motors
code napoleon	en.wikipedia.org/wiki/Napoleonic_Code

Table 5: DSSM results, top three results are highlighted

Signature	**Test mAP**
E_{10}	0.4459
H_{10}	0.4047
S_9	**0.4687**
H_5^2	0.4492
S_4^2	**0.4720**
$H_5 \times S_4$	0.3109
H_2^5	0.3681
S_1^5	0.3877
$H_2^2 \times E_2 \times S_1^2$	0.3264
$c-$dot	0.4194
$O_{l1}, t=0$	**0.4562**
$O_{l1}, t=1$	0.4498
$O_{l2}, t=1$	0.4456

Signature	**Test mAP**
E_{256}	**0.717**
H_{256}	0.412 [10]
S_{255}	0.588
H_{128}^2	0.547
S_{127}^2	0.662
$H_{128} \times S_{127}$	0.501
$H_{61}^4 \times H_{62}$	0.621
$S_{60}^4 \times S_{61}$	**0.701**
$c-$dot	**0.738**
$O_{l1}, t=0$	0.677
$O_{l1}, t=1$	0.662

indeed useful. However, in large dimensions, the best results are achieved with the standard dot product, questioning the utility of complex geometries in industrial applications.

5.3 A bipartite graph reconstruction

In Section 2.2 we already briefly discussed the advantages of dot products over metric spaces. Let us illustrate this intuition and show that dot-products are indeed better suitable for data with a few objects being more popular than the other ones. For this purpose, we perform graph reconstruction on a synthetic bipartite graph with two sets of size 20 and 700 with 5% edge probability (isolated nodes were removed and the remaining graph is connected). Clearly, in the obtained graph there are a few popular nodes and many nodes of small degrees. Table 11 compares the performance of the best metric space with the dot-product performance. As we can see, this experiment confirms our assumption that specific graphs are poorly embedded in metric spaces. In the supplementary materials, we show the results for all other metric spaces and also discuss why dot products are suitable for certain data structures and can outperform other spaces in practical applications.

Table 6: Bipartite graph reconstruction

	mAP	**distortion**
best metric space (type)	0.82 ($O_{l1}, t=0$)	0.082 ($O_{l1}, t=1$)
$c-$dot	**0.86**	**0.079**

6 Conclusion

In this paper, we proposed overlaying spaces that have better or comparable performance relative to the best product space in the graph reconstruction task, but do not require signature brute-forcing. Improvements are observed for both global distortion and local mAP loss functions. However, the conventional dot-product outperforms all considered methods in graph reconstruction task for mAP loss. In DSSM setup with large embeddings, it also outperforms all methods. This clearly shows the

[10]The gap between E_{256} and H_{256} may seem suspicious, but in Table 5 of [9] a similar pattern is observed.

limitations of hyperbolical, product, and overlaying spaces and the necessity of comparison with the dot product in addition to Euclidean and spherical distances when exploring different spaces for vector representations. On the other hand, custom spaces are useful in DSSM setup for low-dimensional vector representations. This can be useful if there is a need to store very large embedding databases, for example in recommendation systems.

We have to pay attention that some of our conclusions can potentially be biased towards particular datasets considered and may not hold for datasets of different nature. In particular, in DSSM-based analysis we considered a particular web search dataset and for other datasets the impact of the use of complex geometries can be different.

Overlaying spaces proposed in the current paper are metric spaces and can be used in methods based on distances between the elements. However, more complex operations, e.g., algebraic operations over elements in an overlaying space, are questionable. In this case, one may still use the proposed idea and search for the optimal product space signature through overlaying space training with additional regularization. This question has not been considered in this paper and is a subject of a separate study.

Finally, it is important to stress that while vector spaces and dot-product similarities are often used in practice, research papers usually compare new complex geometries with the Euclidean space. This may cause confusion and a false impression that complex geometries improve over widely used systems. Our results show that a comparison with the standard dot-product similarity is necessary for research articles of this kind.

References

[1] Phillip Bonacich. 2008. Book Review: W. de Nooy, A. Mrvar, and V. Batagelj Exploratory Social Network Analysis With Pajek. (2004). *Sociological Methods & Research - SOCIOL METHOD RES* 36 (05 2008), 563–564. https://doi.org/10.1177/0049124107306674

[2] Silvere Bonnabel. 2013. Stochastic Gradient Descent on Riemannian Manifolds. *IEEE Trans. Automat. Control* 58 (2013), 2217–2229.

[3] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*. 737–744.

[4] John Burkardt. 2011. Cities – City Distance Datasets. https://people.sc.fsu.edu/~jburkardt/datasets/cities/cities.html

[5] Frederick Arthur Ficken. 1939. The Riemannian and affine differential geometry of product-spaces. (1939), 892–913.

[6] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94. https://doi.org/10.1016/j.knosys.2018.03.022

[7] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1809–1818.

[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. *CoRR* abs/1607.00653 (2016). arXiv:1607.00653 http://arxiv.org/abs/1607.00653

[9] Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. 2019. Learning mixed-curvature representations in product spaces. *International Conference on Learning Representations (ICLR)* (2019).

[10] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *IEEE International Conference on Data Mining (ICDM 2008)*. 263–272. http://yifanhu.net/PUB/cf.pdf

[11] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search using Clickthrough Data. ACM International Conference on Information and Knowledge Management (CIKM).

[12] Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (12 2014).

[13] Jure Leskovec and Julian J. Mcauley. 2012. Learning to Discover Social Circles in Ego Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 539–547. http://papers.nips.cc/paper/4532-learning-to-discover-social-circles-in-ego-networks.pdf

[14] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. 2017. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 212–220.

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781

[16] Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*. 6338–6347. http://papers.nips.cc/paper/7213-poincare-embeddings-for-learning-hierarchical-representations.pdf

[17] Maximillian Nickel and Douwe Kiela. 2018. Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry. In *International Conference on Machine Learning*. 3776–3785. https://arxiv.org/abs/1806.03417

[18] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. *CoRR* abs/1403.6652 (2014). arXiv:1403.6652 http://arxiv.org/abs/1403.6652

[20] Gang Qian, Shamik Sural, Yuelong Gu, and Sakti Pramanik. 2004. Similarity between Euclidean and Cosine Angle Distance for Nearest Neighbor Queries. In *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC '04)*. Association for Computing Machinery, New York, NY, USA, 1232–1237. https://doi.org/10.1145/967900.968151

[21] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Re. 2018. Representation Tradeoffs for Hyperbolic Embeddings. In *International Conference on Machine Learning*. 4457–4466.

[22] Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. 2018. Poincar\'e GloVe: Hyperbolic Word Embeddings. *arXiv preprint arXiv:1810.06546* (2018).

[23] Pavan K Turaga and Anuj Srivastava. 2016. *Riemannian Computing in Computer Vision.* Springer.

[24] Steven H Watts, Duncan J./Strogatz. 1998. *Collective Dynamics of Small- World Networks. Nature*. 393:440 – 442. https://doi.org/10.1007/978-3-658-21742-6_130

[25] Benjamin Wilson and Matthias Leimeister. 2018. Gradient descent in hyperbolic space. *arXiv preprint arXiv:1805.08207* (2018).

[26] Richard C Wilson, Edwin R Hancock, Elżbieta Pekalska, and Robert PW Duin. 2014. Spherical and hyperbolic embeddings of data. *IEEE transactions on pattern analysis and machine intelligence* 36, 11 (2014), 2255–2269.

Supplementary materials

A Experimental setup

A.1 Training details

All models discussed in Section 5.1 were trained with 2000 iterations. If more than one learning rate was used for a certain dataset (due to problems with the convergence of individual models), all the spaces were evaluated for all learning rates and the best result was reported for each space. For distortion, the learning rate was 0.1 for all datasets except UCSA312 (Cities), where we had 0.1 and 0.01. For mAP, the learning rate 0.1 was used for all datasets except UCSA312 and CSPhDs, where we had 0.01 and 0.05 for both datasets.

For the experiments in Section 5.2, we used 5000 iterations for short embeddings and 1000 for long ones (long embeddings converged faster). Hard-negative mining was not used for DSSM training. Instead, large batches of 4096 random training examples (almost 1% of the entire dataset) were used. During the learning process, only the training queries and documents were used. For evaluation, the nearest website was searched among all documents. The training part was 90% of the dataset, and the quality discrepancy between validation and test sets was quite small. Our code[11] supplements the submission.

A.2 WLA6 dataset details

As described in the main text, this dataset is obtained by running the breadth-first search algorithm on the category graph [12] of the English-language Wikipedia, starting from the vertex (category) “Linear algebra” and limited to the depth 6 (Wikipedia Linear Algebra 6). We provide this graph along with the texts (names) of the vertices (categories). The resulting graph is very close to being a tree, although there are some cycles. Predictably, hyperbolic space gives a significant profit for this graph, while using product spaces gives almost no additional advantage. The purpose of using this dataset is to check our conclusions on data other than those used in [9] and to evaluate overlaying spaces on data where product spaces do not provide quality gains.

B Additional experimental results

B.1 Our implementation of product spaces vs original one

Table 7 compares our implementation with the results reported in [9]. It should be noted that we have significantly different algorithms with differing numbers of iterations.

The optimal values of distortion obtained with our algorithm (except the UCSA312 dataset) are comparable and usually better than the ones reported in [9]. On the UCSA312 the obtained distortion is orders of magnitude better, what can be caused by the proper choice of the learning rate (in our experiments on this dataset, this choice significantly affected the results). These results indicate that our solution is a good starting point to compare different spaces and similarities.

For MAP, we optimize the proxy-loss, in contrast to the canonical implementation, where both metrics were specified for models trained with distortion. Obviously, for our approach, the results are more stable: we do not have such a large spread of values for different spaces. We noticed that optimizing mAP directly leads to significant improvements.

[11] https://github.com/shevkunov/Overlaying-Spaces-and-Practical-Applicability-of-Complex-Geometries

[12] https://en.wikipedia.org/wiki/Special:CategoryTree - in fact it is not a tree now: category "Matrix theory" have subcategory "Matrices" with subcategory "Matrix theory", for example

Table 7: Graph reconstruction original vs our

	UCSA312		**CS PhDs**		**Power**		**Facebook**	
	Canon.	Our	Canon.	Our	Canon.	Our	Canon.	Our
				Distortion				
E_{10}	0.0735	0.0032	0.0543	0.0475	0.0917	0.0408	0.0653	0.0487
H_{10}	0.0932	0.0111	0.0502	0.0443	0.0388	0.0348	0.0596	0.0483
S_{10}	0.0598	0.0095	0.0569	0.0503	0.0500	0.0450	0.0661	0.0540
$H_5 \times H_5$	0.0756	0.0057	0.0382	0.0345	0.0365	0.0255	0.0430	0.0372
$S_5 \times S_5$	0.0593	0.0079	0.0579	0.0492	0.0471	0.0433	0.0658	0.0511
$H_5 \times S_5$	0.0622	0.0068	0.0509	0.0337	0.0323	0.0249	0.0402	0.0318
H_2^5	0.0687	0.0059	0.0357	0.0344	0.0396	0.0273	0.0525	0.0439
S_2^5	0.0638	0.0072	0.0570	0.0460	0.0483	0.0418	0.0631	0.0489
$H_2^2 \times E_2 \times S_2^2$	0.0765	0.0044	0.0391	0.0345	0.0380	0.0299	0.0474	0.0406
				mAP				
E_{10}		0.9290	0.8691	0.9487	0.8860	0.9380	0.5801	0.7876
H_{10}		0.9173	0.9310	0.9399	0.8442	0.9385	0.7824	0.7997
S_{10},		0.9254	0.8329	0.9578	0.7952	0.9436	0.5562	0.7868
$H_5 \times H_5$		0.9247	0.9628	0.9481	0.8605	0.9415	0.7742	0.8084
$S_5 \times S_5$		0.9231	0.7940	0.9662	0.8059	0.9466	0.5728	0.7891
$H_5 \times S_5$		0.9316	0.9141	0.9654	0.8850	0.9467	0.7414	0.8087
H_2^5		0.9364	0.9694	0.9671	0.8739	0.9508	0.7519	0.7979
S_2^5		0.9281	0.8334	0.9714	0.8818	0.9521	0.5808	0.7915
$H_2^2 \times E_2 \times S_2^2$		0.9391	0.8672	0.9611	0.8152	0.9486	0.5951	0.7970

B.2 Graph reconstruction

In Tables 2 and 3 of the main text, we reduced the dimensionality of spherical spaces since we fixed the number of stored values for each space. Here, in Tables 8 and 9, we present the extended results, where we fix the mathematical dimension of product spaces, similarly to [9]. Taking into account the statistical significance estimated for five restarts of the algorithm with different random initialization, the results are similar to ones reported in the main text.

Table 8: Graph reconstruction with distortion loss, top results are highlighted.

Signature	**UCSA312**	**CS PhDs**	**Power**	**Facebook**	**WLA6**
E_{10}	**0.00318**	0.0475	0.0408	0.0487	0.0530
H_{10}	0.01114	0.0443	0.0348	0.0483	**0.0279**
S_{10}	0.00951	0.0503	0.0450	0.0540	0.0589
$H_5^2 \equiv H_5 \times H_5$	0.00573	0.0345	0.0255	0.0372	**0.0279**
$S_5 \times S_5 \equiv S_5^2$	0.00792	0.0492	0.0433	0.0511	0.0585
$H_5 \times S_5$	0.00681	**0.0337**	**0.0249**	**0.0318**	0.0296
H_2^5	0.00592	0.0344	0.0273	0.0439	0.0356
S_2^5	0.00720	0.0460	0.0418	0.0489	0.0549
$H_2^2 \times E_2 \times S_2^2$	0.00436	0.0345	0.0299	0.0406	0.0405
$c-\text{dot}$	0.04005	0.0412	0.0461	**0.0236**	0.0296
$ce^{-\text{dot}}$	0.08306	0.0424	0.0505	**0.0192**	**0.0270**
$O_{l1}, t=0$	**0.00356**	0.0368	0.0281	0.0458	0.0286
$O_{l1}, t=1$	**0.00330**	**0.0300**	**0.0231**	0.0371	**0.0272**
$O_{l2}, t=1$	0.00530	**0.0328**	**0.0246**	**0.0324**	**0.0278**

Table 9: Graph reconstruction with mAP ranking loss, top results are highlighted

Signature	**UCSA312**	**CS PhDs**	**Power**	**Facebook**	**WLA6**
E_{10}	0.9290	0.9487	0.9380	0.7876	0.7199
H_{10}	0.9173	0.9399	0.9385	0.7997	0.9617
S_{10}	0.9254	0.9578	0.9436	0.7868	0.7287
H_5^2	0.9247	0.9481	0.9415	0.8084	0.9682
S_5^2	0.9231	0.9662	0.9466	0.7891	0.7353
$H_5 \times S_5$	0.9316	0.9654	0.9467	0.8087	**0.9779**
H_2^5	0.9364	0.9671	0.9508	0.7979	0.8597
S_2^5	0.9281	0.9714	0.9521	0.7915	0.7346
$H_2^2 \times E_2 \times S_2^2$	0.9391	0.9611	0.9486	0.7970	0.6796
$c-$ dot	**1**	**1**	**0.9983**	**0.8745**	**0.9990**
$O_{l1}, t=0$	**0.9522**	0.9879	0.9728	0.8093	0.6759
$O_{l1}, t=1$	**0.9522**	**0.9904**	**0.9762**	**0.8185**	0.9598
$O_{l2}, t=1$	**0.9522**	**0.9938**	**0.9907**	**0.8326**	**0.9694**

B.3 Other ways to convert distances to probabilities

For the proxy-loss, we additionally experimented with other ways of converting distances to probabilities. Let us write L_{proxy} in the general form:

$$L_{proxy} = -\sum_{(v,u)\in E} \log \mathrm{P}((v,u) \in E) = -\sum_{(v,u)\in E} \log \frac{t\big(d_U(f(v), f(u))\big)}{\sum_{w\in V} t\big(d_U(f(v), f(w))\big)}, \tag{7}$$

where $t(d)$ is a function that decreases with distance d. We compare the following alternatives for $t(d)$:

$$\begin{aligned} t_1(d) &= \exp(-d), \\ t_2(d) &= \exp\left(\frac{1}{\min(d, d_0)}\right), \\ t_3(d) &= \frac{1}{\min(d, d_0)}, \end{aligned}$$

where d_0 is a small constant.

Recall that t_1 was used in the main text and it seems to be the most natural choice [13]. Table 10 compares the options and shows that the best results are indeed achieved with t_1.

B.4 Synthetic experiment with bipartite graph

In Table 11, we extend the results presented in Table 6 of the main text. We report distortion and mAP and the corresponding models were trained with distortion and proxy losses, respectively, similar to the experiments in Section 5.1. For each space, learning rates 0.1, 0.05, 0.01, 0.001 were used and the best result was selected. We had 2000 and 1000 iterations for distortion and mAP, respectively. Figure 2 shows the graph visualization.

[13] Note that this is the softmax over inverted distances

Table 10: Different proxy-loss comparison (mAP)

	UCSA312			CS PhD		
$P \sim$	e^{-d}	$e^{1/d}$	$1/d$	e^{-d}	$e^{1/d}$	$1/d$
E_{10}	0.929	0.911	0.899	0.949	0.956	0.831
H_{10}	0.917	0.807	0.885	0.940	0.749	0.764
S_9	0.927	0.801	0.829	0.959	0.583	0.684
S_{10}	0.925	0.797	0.838	0.958	0.572	0.689
H_5^2	0.925	0.890	0.883	0.948	0.976	0.723
S_4^2	0.918	0.821	0.864	0.961	0.733	0.751
S_5^2	0.923	0.802	0.858	0.966	0.748	0.775
$H_5 \times S_4$	0.927	0.835	0.853	0.965	0.781	0.724
$H_5 \times S_5$	0.932	0.838	0.865	0.965	0.804	0.721
H_2^5	0.936	0.896	0.903	0.967	0.998	0.823
S_1^5	0.931	0.850	0.851	0.901	0.863	0.826
S_2^5	0.928	0.856	0.871	0.971	0.876	0.881
$H_2^2 \times E_2 \times S_1^2$	0.934	0.887	0.820	0.950	0.891	0.751
$H_2^2 \times E_2 \times S_2^2$	0.939	0.872	0.865	0.961	0.884	0.689
$O_{l1}, t = 0$	0.952	0.933	0.872	0.988	0.961	0.762
$O_{l1}, t = 1$	0.952	0.947	0.877	0.990	0.963	0.815
$O_{l2}, t = 1$	0.952	0.939	0.880	0.994	0.979	0.810
$c -$ dot	1	1	0.777	1	0.999	0.917

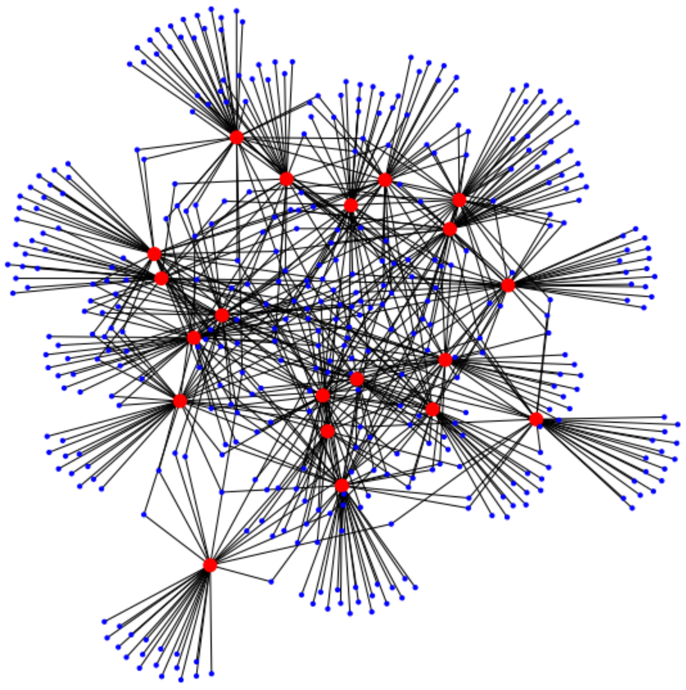

Figure 2: Graph visualization, big (red) nodes from the smaller part

Table 11: Bipartite graph reconstruction

	mAP	**distortion**
E_{10}	0.777	0.094
H_{10}	0.794	0.095
S_9	0.689	0.100
H_5^2	0.799	0.090
S_4^2	0.522	0.107
$H_5 \times S_4$	0.787	0.094
H_2^5	0.761	0.086
S_1^5	0.334	0.148
$H_2^2 \times E_2 \times S_1^2$	0.482	0.098
$O_{l1}, t = 0$	0.824	0.094
$O_{l1}, t = 1$	0.803	0.082
$O_{l2}, t = 1$	0.814	0.092
best metric space	0.824	0.082
$c -$ dot	**0.863**	**0.079**

C Discussion on advantages of dot products

In this section, we discuss the advantages of the dot product and give an intuition regarding particular structures that are better embedded using this similarity measure.

The most straightforward advantage of the dot product is that it allows us to easily differentiate between popular and unpopular items. This property is usually attributed to the hyperbolic space when it is compared with spherical and Euclidean ones. However, the concept of popularity can be much easier expressed with the dot-product similarity. Popularity often affects the structure of real-world data, from social networks to recommendation systems. Assume that there are two items with similar properties/topic, but with different quality/popularity. Then, given a query with the same topic (the direction in the vector space), it is better to recommend the more popular item (with larger vector norm). This scenario can be visualized with the following graph structure. Assume that we have an arbitrary graph G, which has a standard core-periphery structure. Now we add two elements to this graph: the element u is not very popular, it is connected only to several core elements of G; the element v is popular and it is connected to all elements of G. Such a situation is easily modeled with the dot-product similarity: the vectors u and v have the same direction (corresponding to the core elements of G), but different norms; as a result, they have different numbers of neighbors.[14] In other spaces, this situation is harder to model: u and v are connected to the same core elements of G, so they have to be close to each other and hence have similar neighborhoods.

Also, let us discuss why dot products are well suitable for modeling structures similar to the bipartite graph used in our synthetic experiment. Assume that we have a small set of popular nodes V and a large set of less popular nodes U. On U we may have an arbitrary structure, but we want all nodes in V to be not connected to each other and connected to all nodes from U. If $|V|$ is small enough (less than the dimension of the space), then we can easily get several popular items located far away from each other: we can take them to be co-directional to the basis vectors and with large norms. Then, they all will have pairwise dot products equal to 0. The elements of U can be chosen in the positive orthant of the space. They can be easily made connected to all elements of U (if norms of the elements in U are large enough). This intuition led to our synthetic experiment, which demonstrated that the dot-product similarity indeed allows us to capture bipartite structures.

[14]The dot-product similarity can be converted to a graph, e.g., in the following way: if the similarity between two nodes is higher than some threshold, then two nodes are connected.