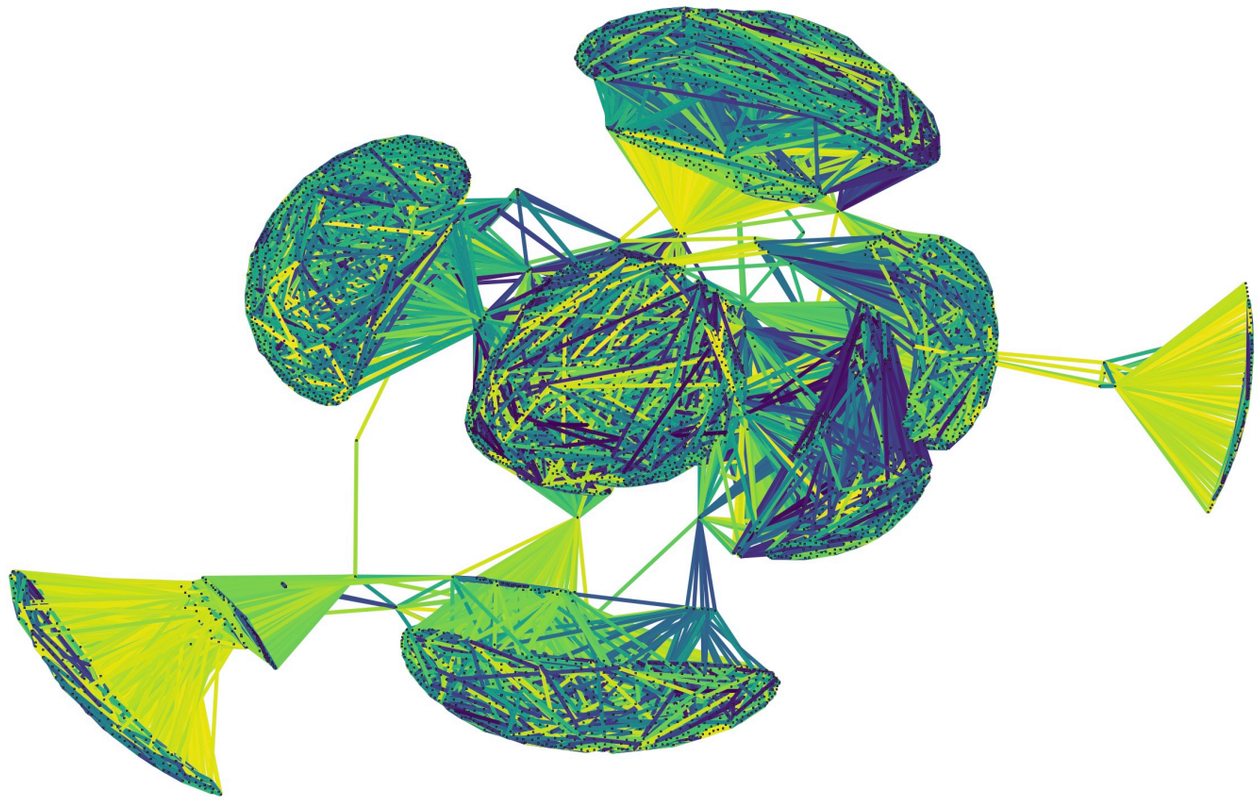


Graph Embeddings in Symmetric Spaces

Beatrice Pozzetti
Heidelberg University
(joint with F. Lopez, M. Strube, S. Trettel, A. Wienhard)

GaML-July 11, 2022

STRUCTURES
CLUSTER OF
EXCELLENCE

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Pictures Courtesy of Gye-Seon Lee, Jos Leys, Federico Lopez, Steve Trettel

Graph Embeddings in Symmetric Spaces

Plan for today:

- What are symmetric spaces?
- Graph embeddings in Siegel spaces
  [LPTSW, Symmetric Spaces for Graph Embeddings: A Finsler-Riemannian Approach. (ICML 2021)]
- Gyrocalculus in SPD
  [LPTSW, Vector-valued Distance and Gyrocalculus on the Space of Symmetric Positive Definite Matrices, (NeurIPS 2021)]

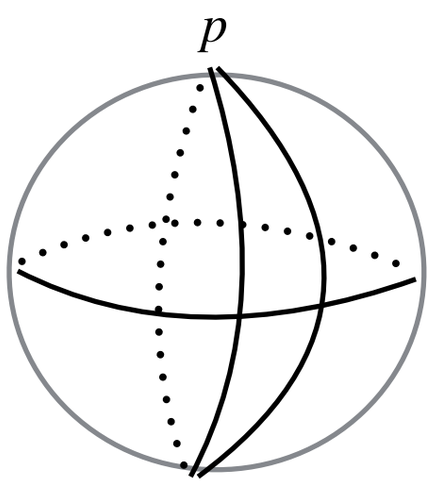
(joint with Federico Lopez, Michael Strube, Steve Trettel, Anna Wienhard)

1) What are symmetric spaces?

Definition: A **symmetric space** is a Riemannian manifold M where, for every point p, the geodesic involution $\sigma_p : M \to M$ is an isometry

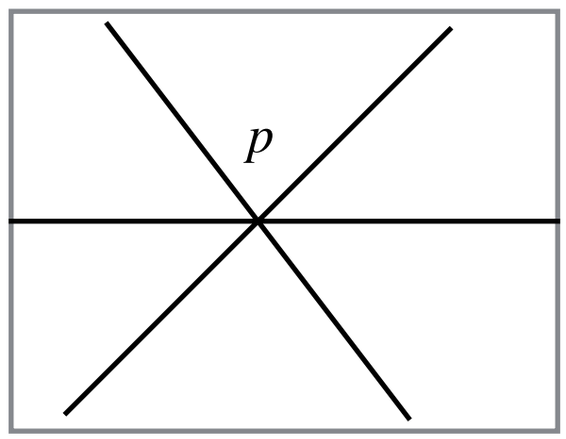
Examples:

$\mathbb{S}^2$



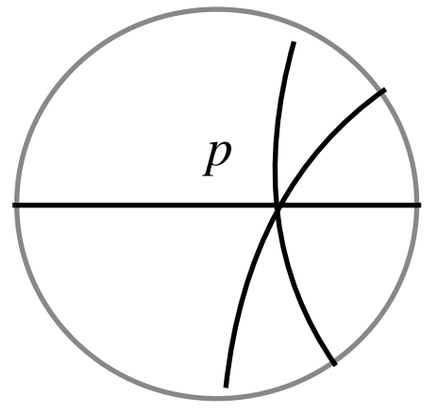
Sphere

$\mathbb{E}^2$



Euclidean space

$\mathbb{H}^2$

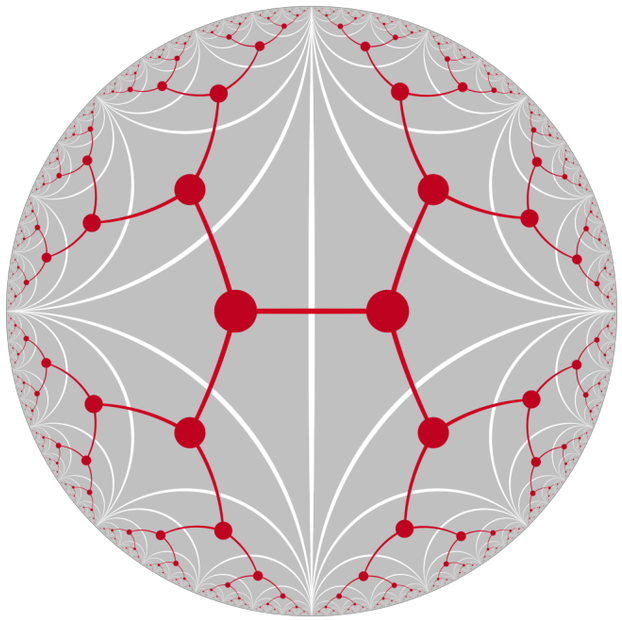


Hyperbolic space

Negative versus nonpositive curvature

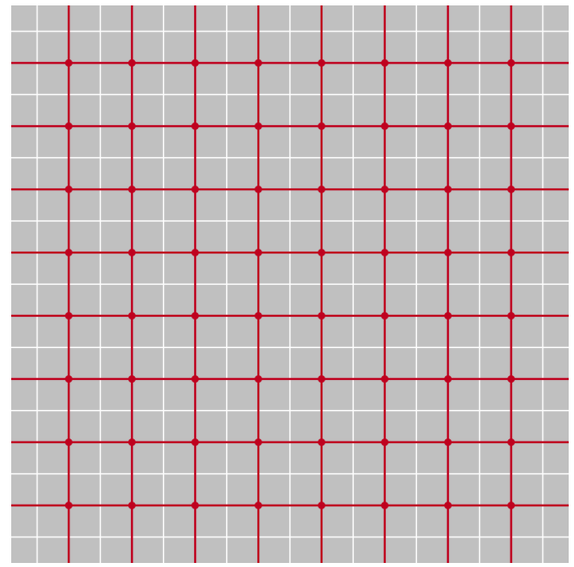
Hyperbolic geometry

looks like a tree - exponential growth

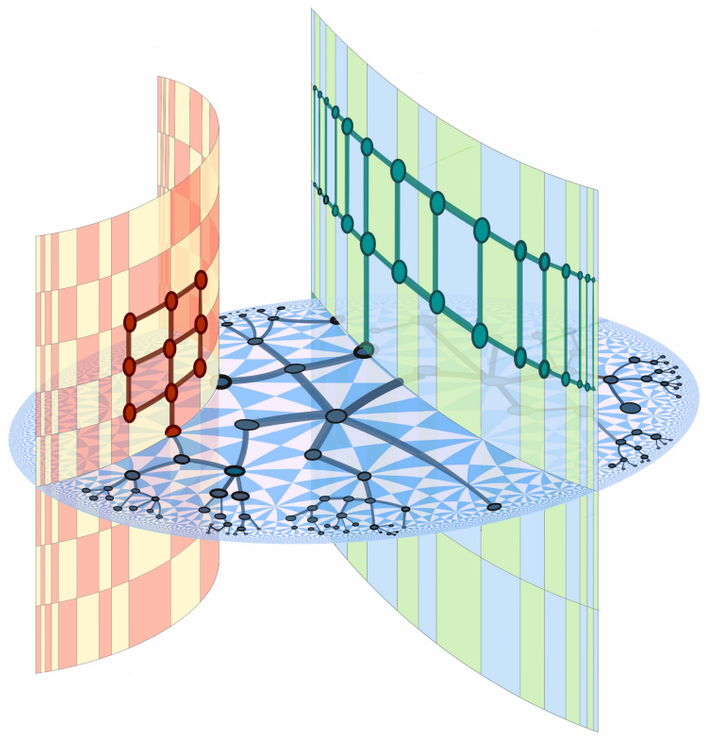


Euclidean geometry

looks like a grid - polynomial growth



Higher rank symmetric space



Rich geometry but at the same time computationally tractable

Classification

Theory: Every symmetric space can be written as $\mathbf{G/K}$ where $\mathbf{G}$ is a semisimple Lie group and $\mathbf{K}$ is a maximal compact subgroup

Type	Non-compact
AI	$\mathrm{SL}(n,\mathbb{R})/\mathrm{SO}(n,\mathbb{R})$
A	$\mathrm{SL}(n,\mathbb{C})/\mathrm{SU}(2)$
BDI	$\mathrm{SO}(p,q)/\mathrm{SO}(p)\times\mathrm{SO}(q)$
AIII	$\mathrm{SU}(p,q)/\mathrm{SU}(p)\times\mathrm{SU}(q)$
CI	$\mathrm{Sp}(2n,\mathbb{R})/\mathrm{U}(n)$
DIII	$\mathrm{SO}^*(2n)/\mathrm{U}(n)$
CII	$\mathrm{Sp}(p,q)/\mathrm{Sp}(p)\times\mathrm{Sp}(q)$
AII	$\mathrm{SL}(n,\mathbb{H})/\mathrm{Sp}(n)$
D	$\mathrm{SO}(2n,\mathbb{C})/\mathrm{SO}(2n)$
B	$\mathrm{SO}(2n+1,\mathbb{C})/\mathrm{SO}(2n+1)$
C	$\mathrm{Sp}(n,\mathbb{C})/\mathrm{Sp}(n)$

In each case we can compute concrete formulas for

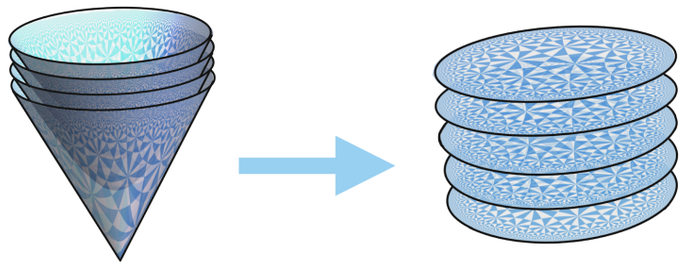
- Distance
- Gradient
- Exponential/logarithm map
- Isometries
- Riemannian metric
- Geodesics
- ….

SPD_n: Symmetric positive definite matrices $\mathrm{GL}_n(\mathbb{R})/\mathrm{SO}_n(\mathbb{R})$

- Points are positive definite real symmetric **n × n** matrices
- SPD_n: Riemannian manifold of non-positive curvature of **n(n + 1)/2** dimensions
  - n-dimensional Euclidean subspaces
  - n−1 dimensional hyperbolic subspaces
- They **combine hyperbolic and Euclidean geometry** thus they can accommodate:
  - Hierarchical structures -> hyperbolic subspaces
  - Flat structures -> Euclidean subspaces

$$\begin{pmatrix} 2 & -1 \\ -1 & 3 \end{pmatrix} \in \mathrm{SPD}_2$$

$$^tvXv > 0 \quad \forall v \in \mathbb{R}^n$$
$$\iff \lambda_i(X) > 0 \quad \forall i$$



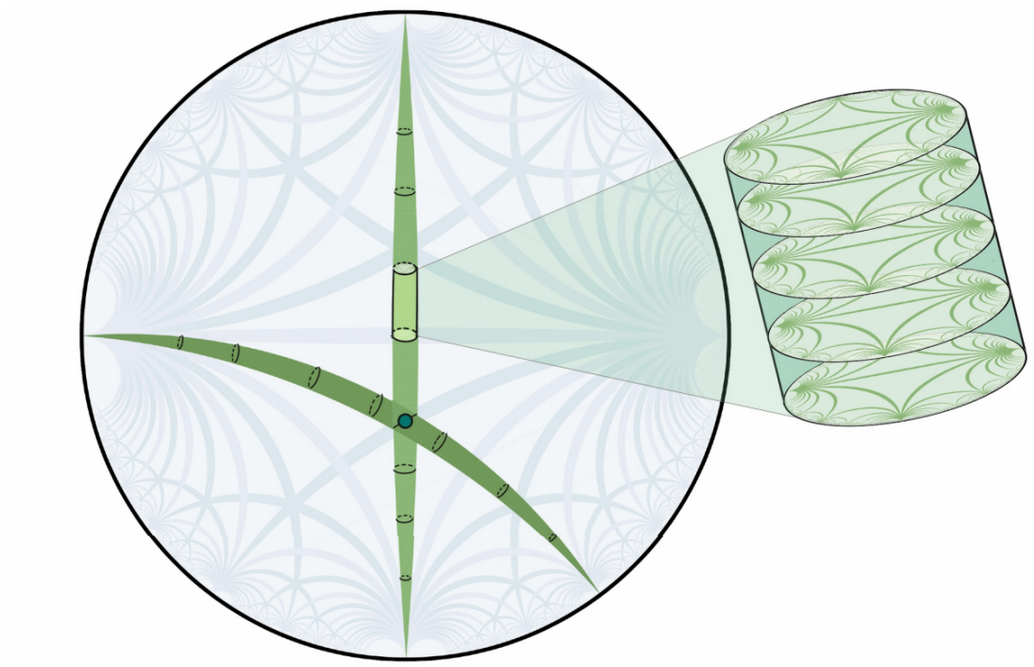
$$\mathrm{SPD}_2 \cong \mathbb{H}^2 \times \mathbb{R}$$

- SPDs have been used for:
  - Pedestrian detection
  - Action and face recognition
  - Image classification
  - Visual tracking
  - Medical image analysis

Siegel spaces

$$\mathscr{B}_n := \{Z \in \mathrm{Sym}(n, \mathbb{C}) \mid \mathrm{I_n} - \bar{Z}Z > 0\} = \mathrm{Sp}(2n, \mathbb{R})/\mathrm{U}(n)$$

rank of $\mathscr{B}_n$ = n dim($\mathscr{B}_n$) = n(n+1)

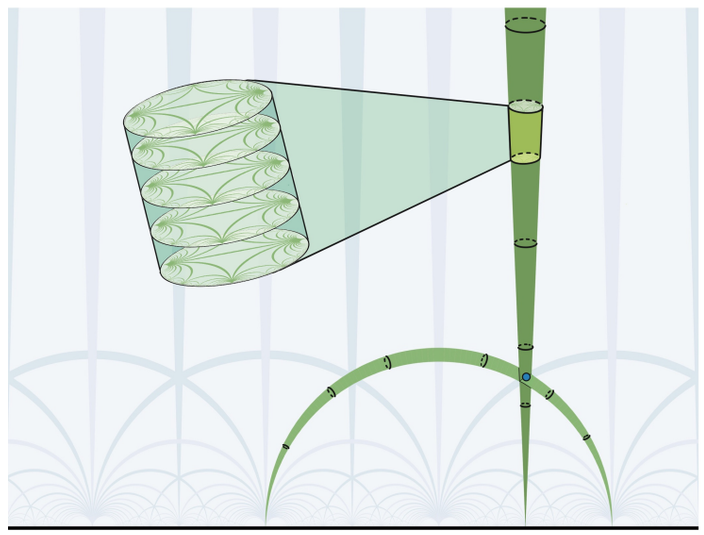


Why Siegel spaces $\mathscr{B}_n$? First of all $\mathscr{B}_1 = \mathbb{H}^2$

- concrete and tractable matrix models of Poincaré disk and upper half plane.
- rich submanifold structure.
- gradient has simple form, it is a matrix rescaling of a Euclidean gradient

$$\mathcal{S}_n := \{Z \in \mathrm{Sym}(n, \mathbb{C}) \mid \mathfrak{I}(Z) > 0\}$$

$$\begin{pmatrix} 2i & -1+i \\ -1+i & -3+i \end{pmatrix} \in \mathcal{S}_n$$



2) Graphs embeddings

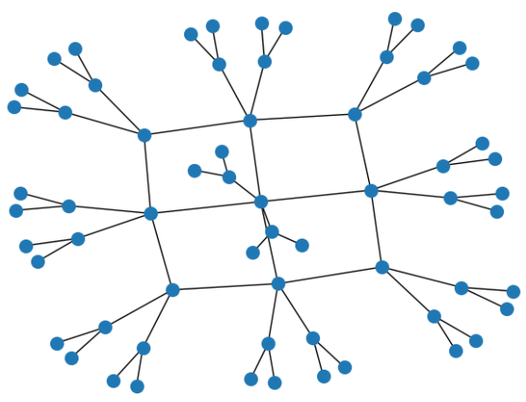
Why Graphs?

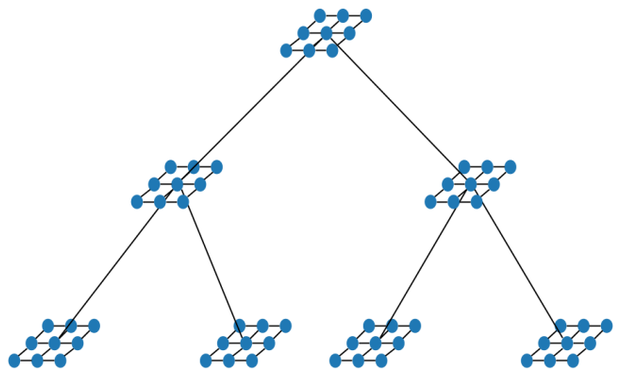
Graphs are ubiquitous, they are a convenient way to represent data.
They are used in many machine learning applications.

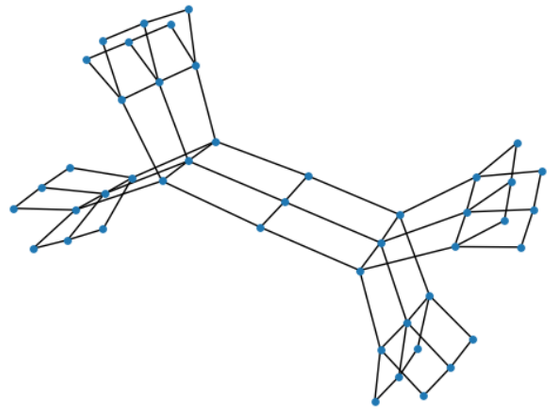
Important tool: Embedding of graphs into manifolds
Use ambient structure to infer properties or structure of the graph

Classical: embeddings into Euclidean spaces, but ….

… complex networks have inherent features of negative curvature, and many graphs have mixed features



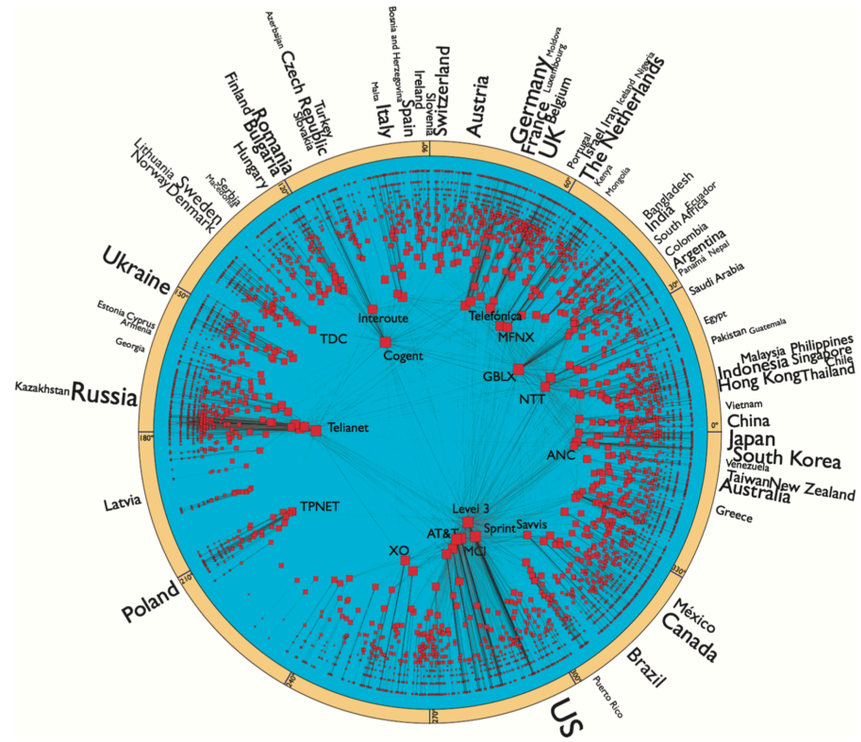




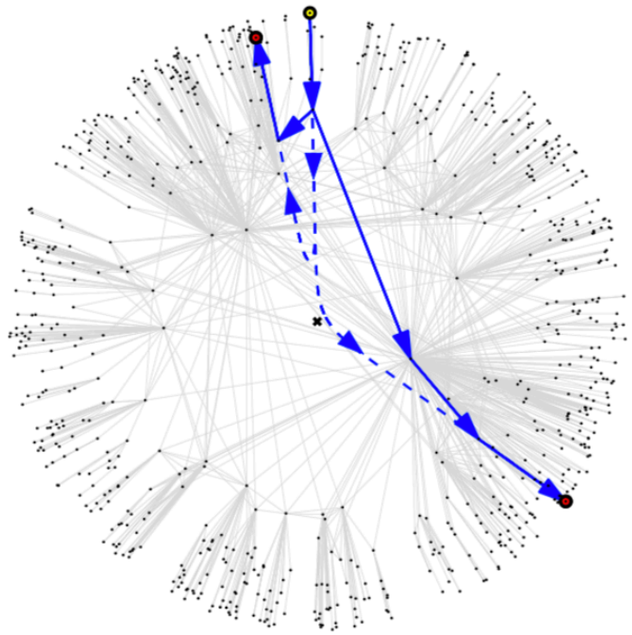
In recent years, hyperbolic spaces, products of constant curvature space, Grassmann manifolds, space of positive definite matrices have been proposed as suitable targets.

Hyperbolic geometry in machine learning

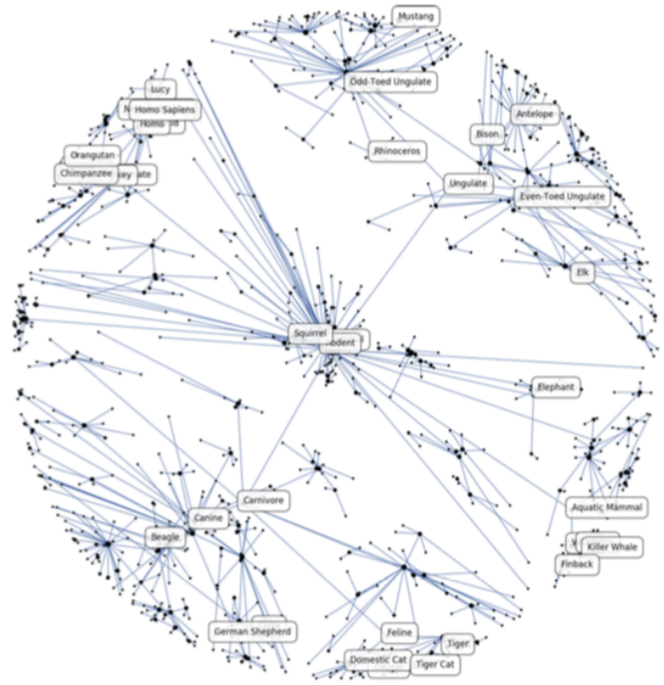
Krioukov et al., Hyperbolic Geometry of Complex Networks, 2010



Boguna et al., Sustaining the Internet with hyperbolic mapping, 2010



Nickel and Kiela, Poincaré Embeddings for Learning Hierarchical Representations, 2017



We described a general framework to learn graph embeddings in symmetric spaces, and implemented it in Siegel spaces, using additional geometric properties of symmetric spaces

Finsler metrics

X carries many Finsler metrics with same symmetry group as the Riemannian metric. Finsler metrics are different, and more graph like

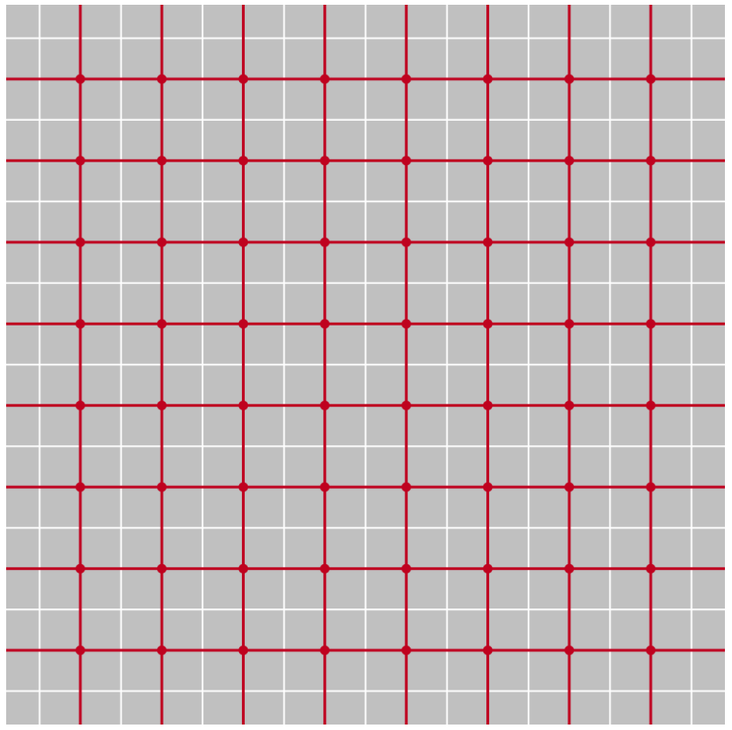
Example: grid in $\mathbb{R}^2$

Riemannian metric $||x||_2 = \sqrt{x_1^2 + x_2^2}$

ℓ_1- metric $||x||_1 = x_1 + x_2$

distortion with respect to the Riemannian metric is at least $\sqrt{2}$

no distortion with respect to the ℓ_1- metric



But Finsler metrics are not differentiable and not convex, so not so good for optimization

Use Finsler loss functions in a Riemannian optimization scheme

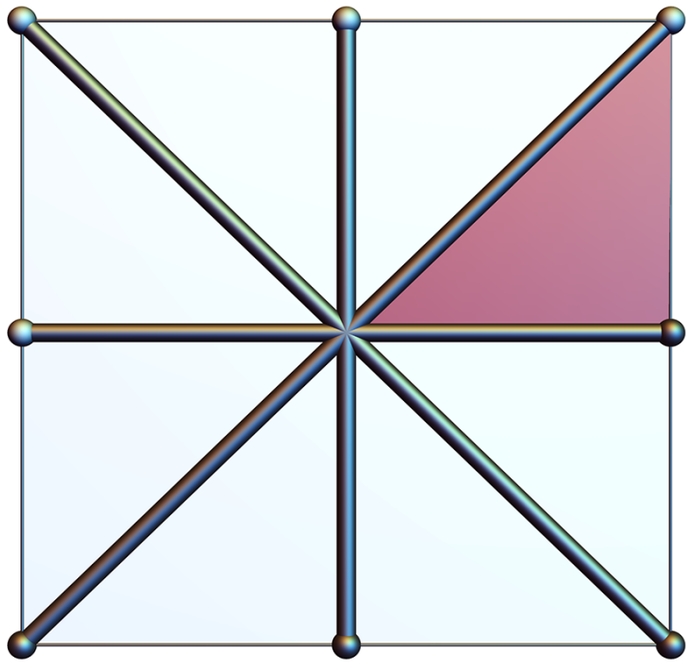
Vector Valued Distance

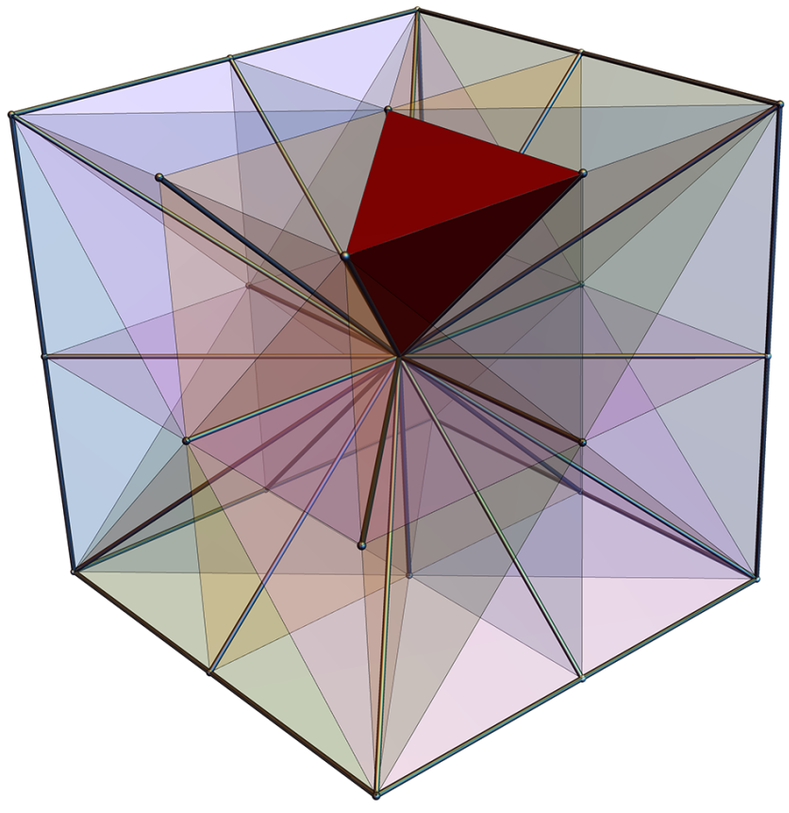
The invariant of two points in a symmetric space is not the distance, but a vector in a Euclidean space of dimension the rank

Example:

Given A,B two positive definite symmetric matrices

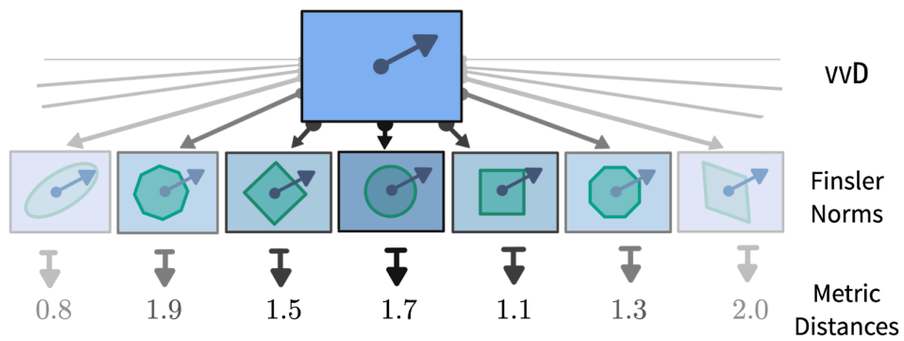
We can always find a basis in which A is the identity matrix and B is a diagonal matrix with real eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$





Advantages of VVD

- By taking different norms on the VVD we can compute **several metrics**
  - Riemannian distance
  - Finsler distances
  - Generalize previous metrics



- VVD provides **much more information** than just the distance
  - Read the regularity of the geodesics joining two points
  - **Visualize and analyze** high-dimensional embeddings

Embedding pipeline

Initial embedding: put your graph X in some way

Optimize your embedding: Riemannian gradient descent with respect to some loss function, e.g.

$$\mathscr{L}(x) = \sum_{1 \le i \le j \le n} \left| \left(\frac{d_{\mathscr{X}}(x_i, x_j)}{d_G(X_i, X_j)} \right)^2 - 1 \right|$$

After many iterations process stabilizes: this gives you your embedding

Then use the embedding, perform tasks, analyze the graph

Apply this procedure when the target is a symmetric space, using the full strength of the Finsler and vector valued distances

Measuring the quality of the embeddings

- **Distortion:** Global metric
  - Checks that distances in the spaces resemble distances in the graph

$$\text{distortion}(a,b) = \frac{|d_{\mathcal{P}}(f(a), f(b)) - d_G(a,b)|}{d_G(a,b)}$$

- **Average precision:** local metric
  - Checks that “neighbors” (closest points) in the space are neighbors in the graph

$$\text{mAP}(f) = \frac{1}{|V|} \sum_{a \in V} \frac{1}{\deg(\text{a})} \sum_{i=1}^{|\mathcal{N}_a|} \frac{|\mathcal{N}_a \cap R_{a,b_i}|}{|R_{a,b_i}|}$$

- Model Baselines
  - Euclidean, Hyperbolic
  - Cartesian product thereof
  - SPD

Implementation

Initialize embedding by randomly choosing points close to iI_n

Use Riemannian gradient descent with respect to the loss function

$$\mathscr{L}(x) = \sum_{1 \leq i \leq j \leq n} \left| \left(\frac{d_{\mathcal{S}_n}(x_i, x_j)}{d_G(X_i, X_j)} \right)^2 - 1 \right|$$

Algorithm 1 Computing Distances

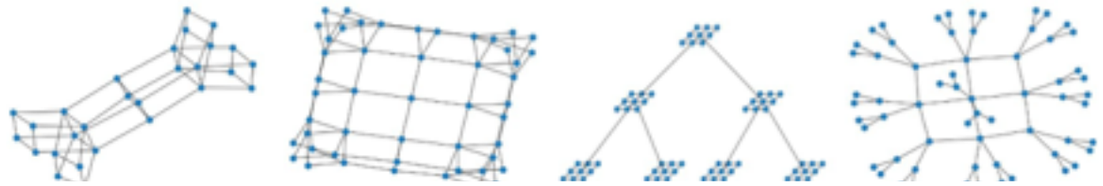
1: Given two points $Z_1, Z_2 \in \mathcal{S}_n$:
2: Define $Z_3 = \sqrt{\Im(Z_1)}^{-1}(Z_2 - \Re(Z_1))\sqrt{\Im(Z_1)}^{-1} \in \mathcal{S}_n$
3: Define $W = (Z_3 - i\mathrm{Id})(Z_3 + i\mathrm{Id})^{-1} \in \mathcal{B}_n$
4: Use the Takagi factorization to write $W = KDK^*$ for D real diagonal, and K unitary.
5: Define $v_i = \log \frac{1+d_i}{1-d_i}$ for d_i the diagonal entries of D.
6: Order the v_i so that $v_1 \geq v_2 \geq \cdots \geq 0$. The **Vector-valued Distance** is $\mathrm{vDist}(Z_1, Z_2) = (v_1, v_2, \ldots, v_n)$.
7: The **Riemannian distance** is $d^R(Z_1, Z_2) := \sqrt{\sum_{i=1}^n v_i^2}$.
8: The **Finsler distance inducing the ℓ^1-metric** is $d^{F1}(Z_1, Z_2) := \sum_{i=1}^n v_i$.
9: The **Finsler distance inducing the ℓ^∞-metric** is $d^{F\infty}(Z_1, Z_2) := \max\{v_i\} = v_1$.

Evaluate with respect to distortion and mean average precision.

Apply it in wo downstream tasks: recommender systems and node classification

Siegel spaces have better representation capabilities, without any a priori knowledge of the internal structure of the graph.

Synthetic Graphs



Symmetric Spaces for Graph Embeddings

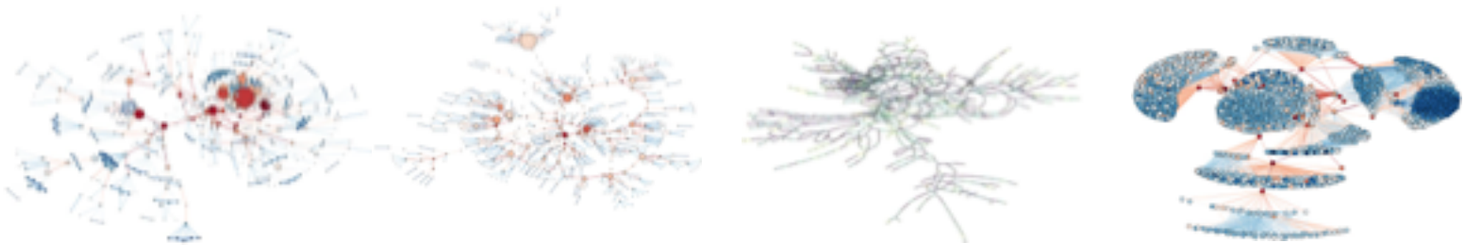
$(\lvert V\rvert, \lvert E\rvert)$	4D Grid (625, 2000)		Tree (364, 363)		Tree × Grid (496, 1224)		Tree × Tree (225, 420)		Tree ◇ Grids (775, 1270)		Grid ◇ Trees (775, 790)	
	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP
$\mathbb{E}^{20}$	11.24±0.00	**100.00**	3.92±0.04	42.30	9.81±0.00	83.32	9.78±0.00	96.03	3.86±0.02	34.21	4.28±0.04	27.50
$\mathbb{H}^{20}$	25.23±0.05	63.74	**0.54±0.02**	**100.00**	17.21±0.21	83.16	20.59±0.11	75.67	14.56±0.27	44.14	14.62±0.13	30.28
$\mathbb{E}^{10} \times \mathbb{H}^{10}$	11.24±0.00	**100.00**	1.19±0.04	**100.00**	9.20±0.01	**100.00**	9.30±0.04	98.03	2.15±0.05	58.23	2.03±0.01	**97.88**
$\mathbb{H}^{10} \times \mathbb{H}^{10}$	18.74±0.01	78.47	0.65±0.02	**100.00**	13.02±0.91	88.01	8.61±0.03	97.63	1.08±0.06	77.20	2.80±0.65	84.88
SPD_6	11.24±0.00	**100.00**	1.79±0.02	55.92	9.23±0.01	99.73	8.83±0.01	98.49	1.56±0.02	62.31	1.83±0.00	72.17
$\mathcal{S}_4^R$	11.27±0.01	**100.00**	1.35±0.02	78.53	9.13±0.01	99.92	8.68±0.02	98.03	1.45±0.09	72.49	1.54±0.08	76.66
$\mathcal{S}_4^{F\infty}$	5.92±0.06	99.61	1.23±0.28	99.56	4.81±0.55	99.28	3.31±0.06	99.95	10.88±0.19	63.52	10.48±0.21	72.53
$\mathcal{S}_4^{F_1}$	**0.01±0.00**	**100.00**	0.76±0.02	91.57	**0.81±0.08**	**100.00**	**1.08±0.16**	**100.00**	**1.03±0.00**	**78.71**	**0.84±0.06**	80.52
$\mathcal{B}_4^R$	11.28±0.01	**100.00**	1.27±0.05	74.77	9.24±0.13	99.22	8.74±0.09	98.12	2.88±0.32	72.55	2.76±0.11	96.29
$\mathcal{B}_4^{F\infty}$	7.32±0.16	97.92	1.51±0.13	99.73	8.70±0.87	96.40	4.26±0.26	99.70	6.55±1.77	73.80	7.15±0.85	90.51
$\mathcal{B}_4^{F_1}$	0.39±0.02	**100.00**	0.77±0.02	94.64	0.90±0.08	**100.00**	1.28±0.16	**100.00**	1.09±0.03	76.55	0.99±0.01	81.82

Table 1. Results for synthetic datasets. Lower D_{avg} is better. Higher mAP is better. Metrics are given as percentage.

On pure graphs the Riemannian metric performs on par with constant curvature baselines and products.

The Finsler ℓ_1-metric significantly outperforms other baselines in all graphs, except trees — almost zero distortion for the 4D grid.

Real world data sets



Symmetric Spaces for Graph Embeddings

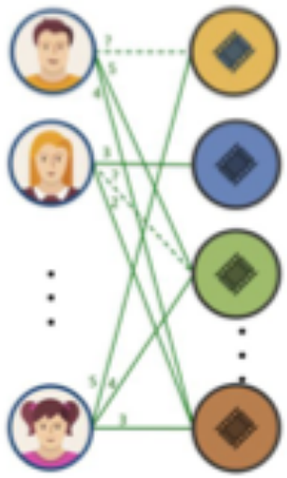
$(\lvert V\rvert, \lvert E\rvert)$	USCA312 (312, 48516)	BIO-DISEASOME (516, 1188)		CSPHD (1025, 1043)		EUROROAD (1039, 1305)		FACEBOOK (4039, 88234)	
	D_{avg}	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP
$\mathbb{E}^{20}$	**0.18±0.01**	3.83±0.01	76.31	4.04±0.01	47.37	4.50±0.00	87.70	3.16±0.01	32.21
$\mathbb{H}^{20}$	2.39±0.02	6.83±0.08	91.26	22.42±0.23	60.24	43.56±0.44	54.25	3.72±0.00	44.85
$\mathbb{E}^{10} \times \mathbb{H}^{10}$	**0.18±0.00**	2.52±0.02	91.99	3.06±0.02	73.25	4.24±0.02	89.93	2.80±0.01	34.26
$\mathbb{H}^{10} \times \mathbb{H}^{10}$	0.47±0.18	2.57±0.05	**95.00**	7.02±1.07	**79.22**	23.30±1.62	75.07	2.51±0.00	36.39
SPD_6	0.21±0.02	2.54±0.00	82.66	2.92±0.11	57.88	19.54±0.99	92.38	2.92±0.05	33.73
$\mathcal{S}_4^R$	0.28±0.03	2.40±0.02	87.01	4.30±0.18	59.95	29.21±0.91	84.92	3.07±0.04	30.98
$\mathcal{S}_4^{F\infty}$	0.57±0.08	2.78±0.49	93.95	27.27±1.00	59.45	46.82±1.02	72.03	**1.90±0.11**	**45.58**
$\mathcal{S}_4^{F_1}$	**0.18±0.02**	1.55±0.04	90.42	**1.50±0.03**	64.11	**3.79±0.07**	**94.63**	2.37±0.07	35.23
$\mathcal{B}_4^R$	0.24±0.07	2.69±0.10	89.11	28.65±3.39	62.66	53.45±2.65	48.75	3.58±0.10	30.35
$\mathcal{B}_4^{F\infty}$	0.21±0.04	4.58±0.63	90.36	26.32±6.16	54.94	52.69±2.28	48.75	2.18±0.18	39.15
$\mathcal{B}_4^{F_1}$	**0.18±0.07**	**1.54±0.02**	90.41	2.96±0.91	67.58	21.98±0.62	91.63	5.05±0.03	39.87

Table 2. Results for real-world datasets. Lower D_{avg} is better. Higher mAP is better. Metrics are given as percentage.

Finsler ℓ_1-metric outperforms the other baselines.

Strong reconstruction capabilities of the Siegel space for real world data.

Recommender systems



You have a bipartite graph (users and items) with edges for each interaction.
You try to recommend items to users according to the distance/similarity between their embeddings.

	ML-1M		ML-100K		LASTFM		MEETUP	
	HR@10	nDG	HR@10	nDG	HR@10	nDG	HR@10	nDG
$\mathbb{E}^{20}$	46.9±0.6	22.7	54.6±1.0	28.7	55.4±0.3	24.6	69.8±0.4	46.4
$\mathbb{H}^{20}$	46.0±0.5	23.0	53.4±1.0	28.2	54.8±0.5	24.9	71.8±0.5	48.5
$\mathbb{E}^{10} \times \mathbb{H}^{10}$	52.0±0.7	27.4	53.1±1.3	27.9	45.5±0.9	18.9	70.7±0.2	47.5
$\mathbb{H}^{10} \times \mathbb{H}^{10}$	46.7±0.6	23.0	54.8±0.9	29.1	55.0±0.9	24.6	71.7±0.1	48.8
$\mathcal{SPD}_6$	45.8±1.0	22.1	53.3±1.4	28.0	55.4±0.2	25.3	70.1±0.6	46.5
$\mathcal{S}_4^R$	**53.8±0.3**	**27.7**	**55.7±0.9**	28.6	53.1±0.5	24.8	65.8±1.2	43.4
$\mathcal{S}_4^{F_\infty}$	45.9±0.9	22.7	52.5±0.3	27.5	53.8±1.7	32.5	69.0±0.5	46.4
$\mathcal{S}_4^{F_1}$	52.9±0.6	27.2	55.6±1.3	**29.4**	**61.1±1.2**	**38.0**	**74.9±0.1**	**52.8**

Riemannian and Finsler ℓ_1-metric outperform the baselines.

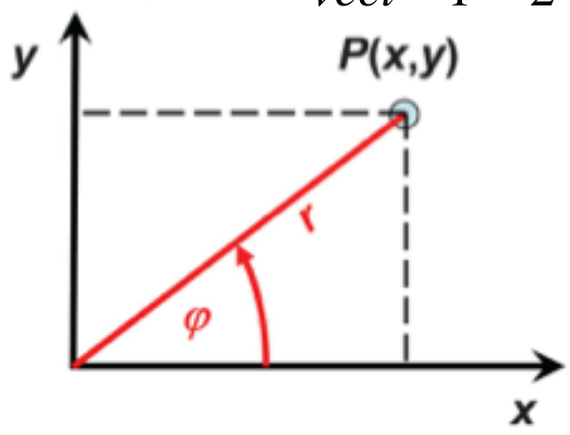
Analyze structure in the graph

Use the vector valued distance to get a coloring of the graph

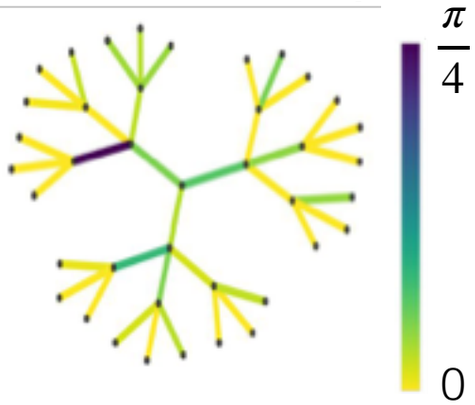
Embed into $\mathcal{S}_2$, for every edge (Z_i, Z_j) have the $d_{vect}(Z_i, Z_j) = (v_1, v_2)$ – take $\frac{v_1}{v_2}$

Take edge (z_1, z_2)

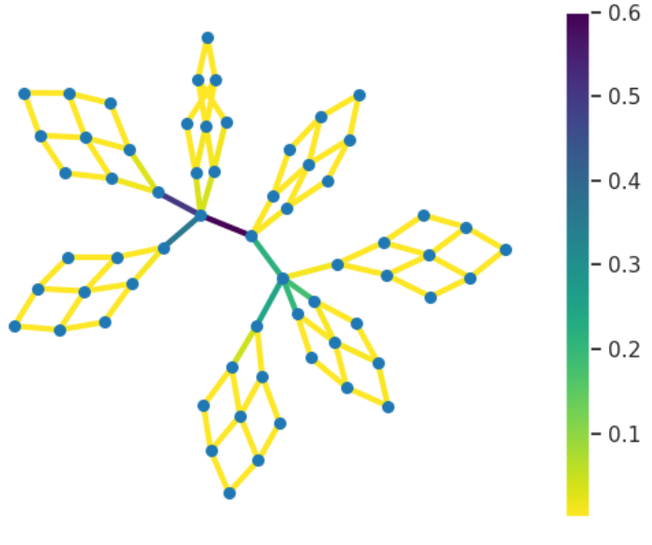
Compute $d_{vect}(z_1, z_2)$

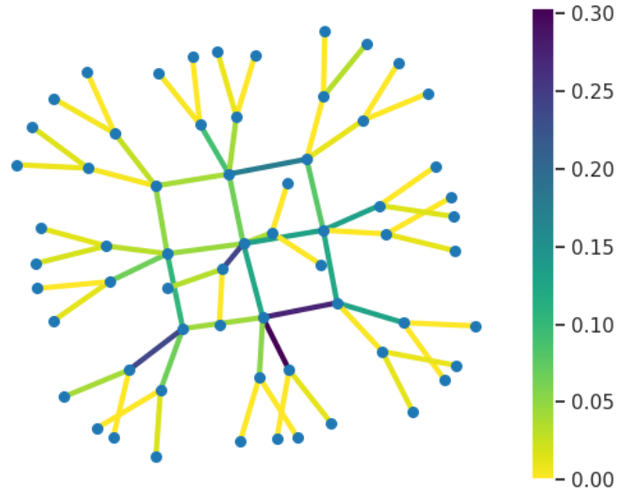


Color edge by φ



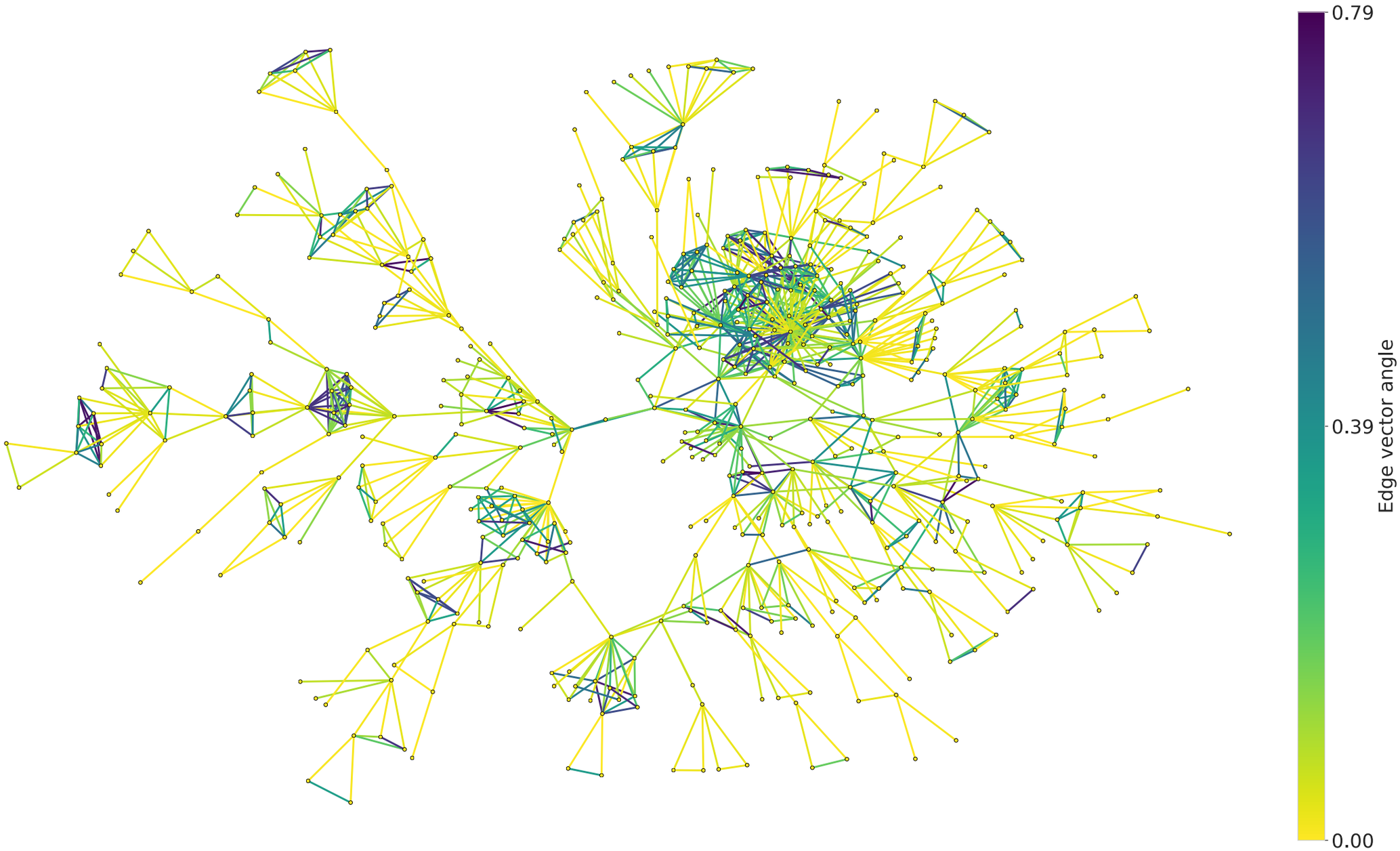
The edge coloring distinguishes different internal structures (grid-like vs. tree-like)



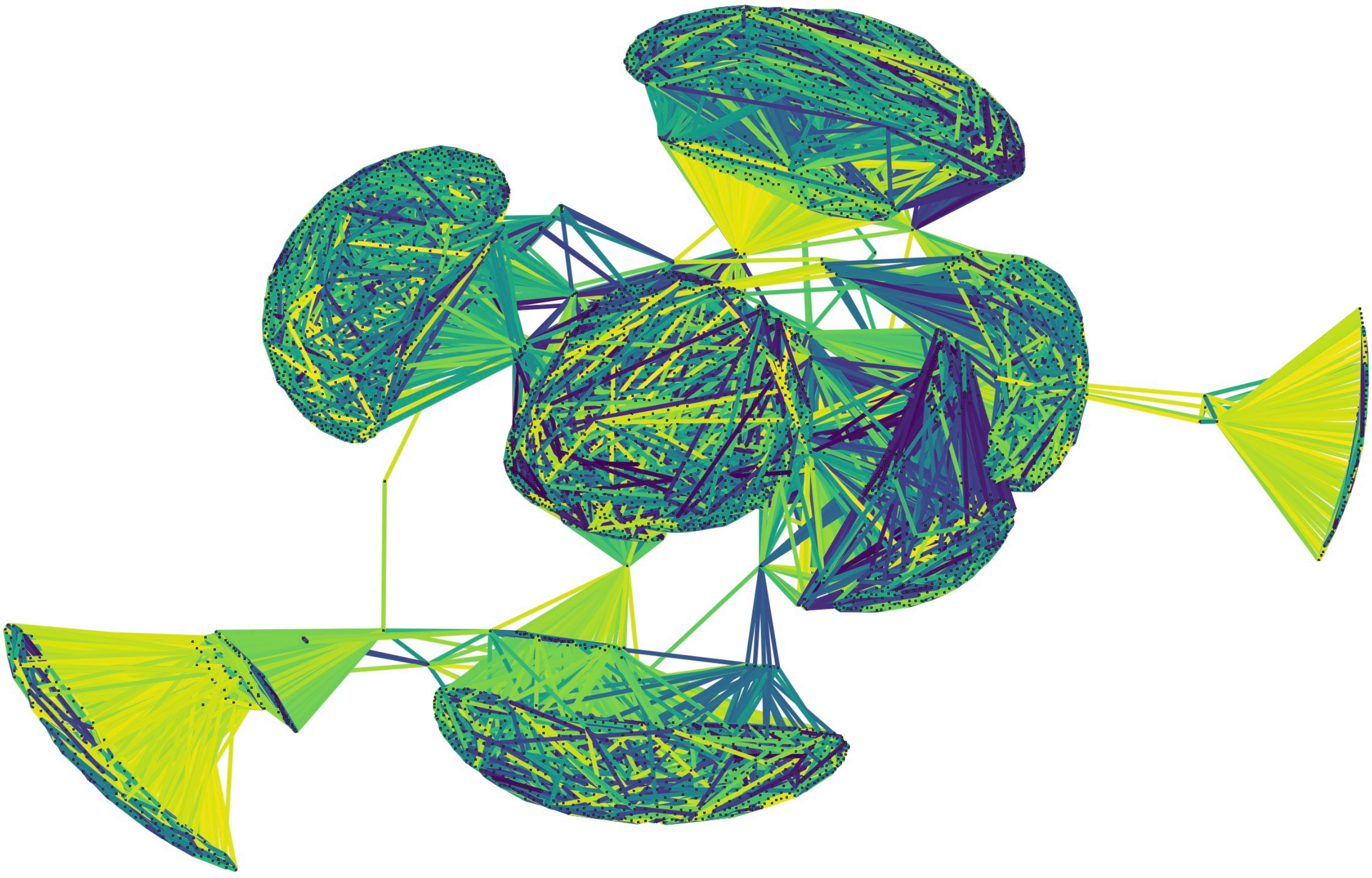


Real world data sets

Also in real world data sets (here biodiseasome) the edge coloring distinguishes the tree like structure from the well-connected parts of the graph



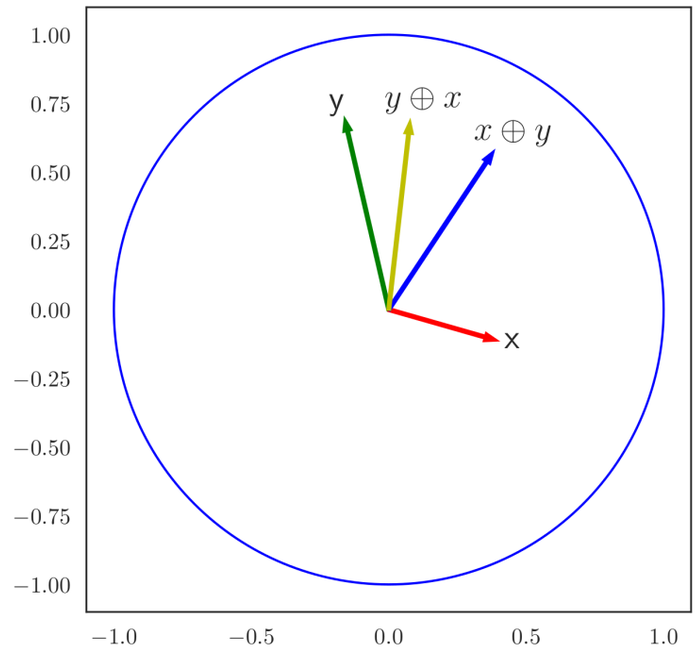
The Facebook graph

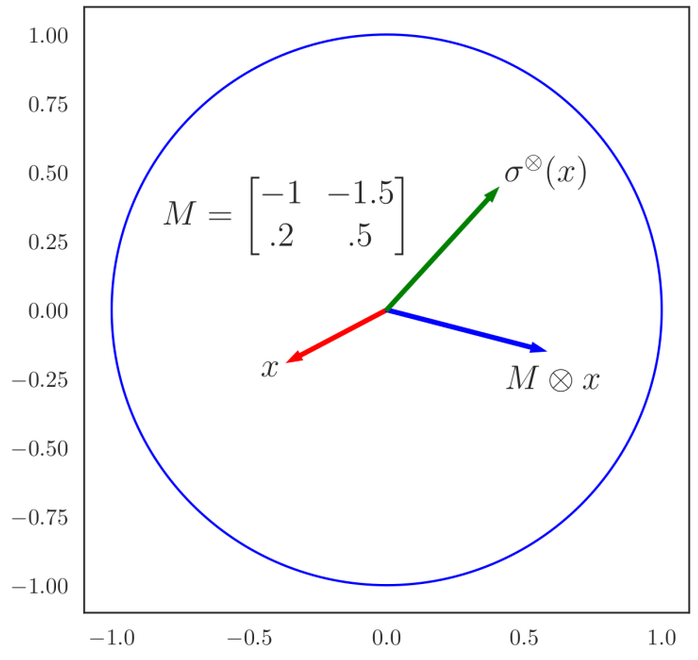


3) Gyrocalculus

- Gyrocalculus is an algebraic formalism to **translate Euclidean operations** to other spaces in a geometrically meaningful way

- Successful applications with Hyperbolic geometry [Ganea et al, 2018]
  - Addition
  - Matrix-vector multiplication
  - Pointwise non-linearities



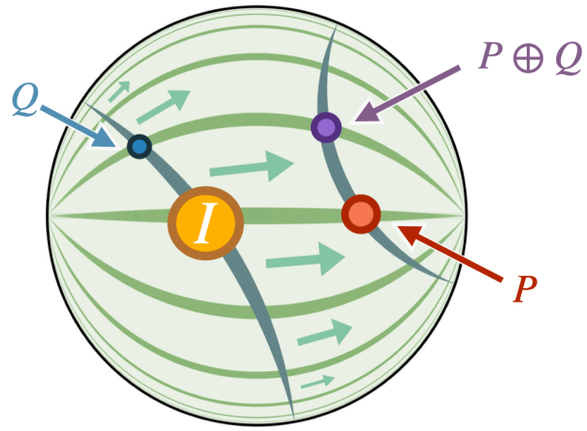


Gyrocalculus on SPD

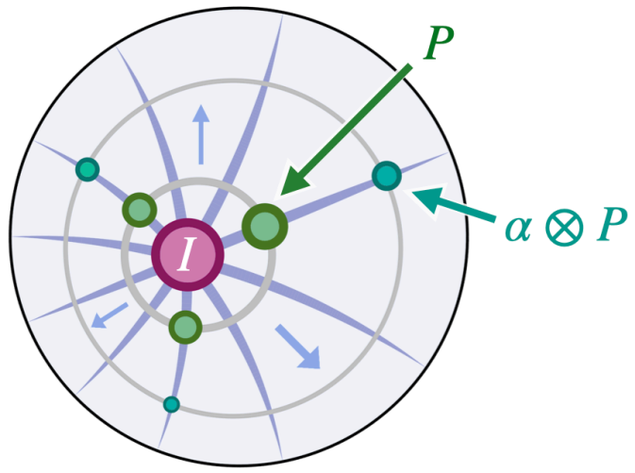
- Addition / subtraction

$$P \oplus Q = \sqrt{P} Q \sqrt{P}$$

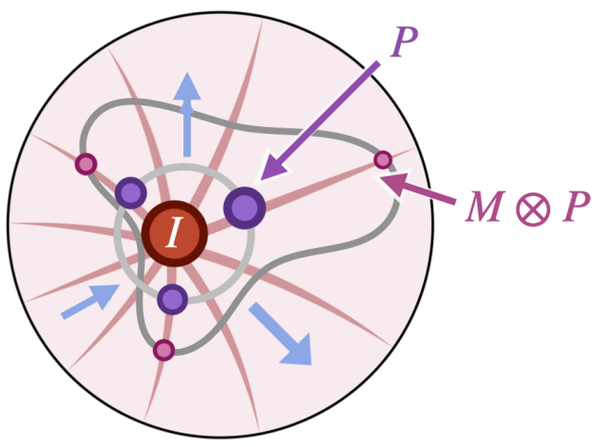
$$\ominus P = P^{-1}$$



- Scalar multiplication $\alpha \otimes P = P^{\alpha} = \exp(\alpha \log(P))$



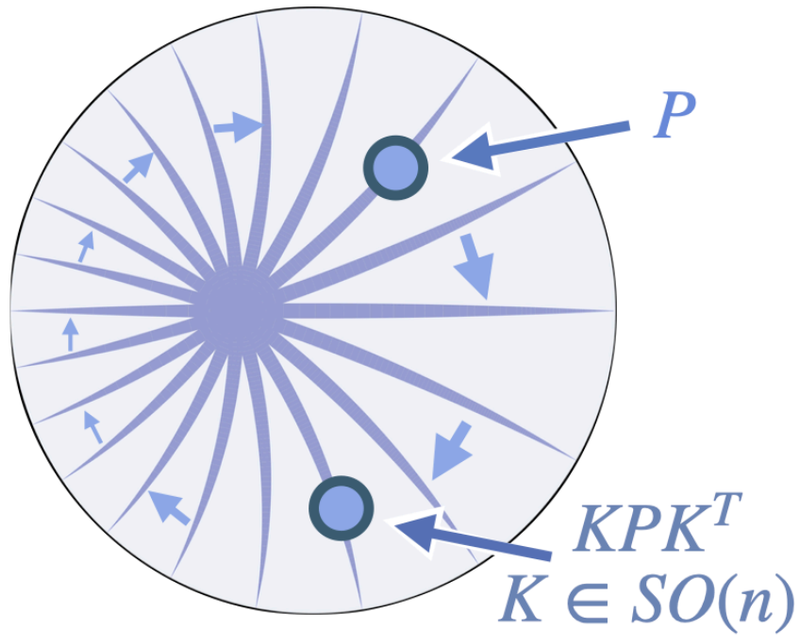
- Matrix scaling $A \otimes P = \exp(A \odot \log(P))$



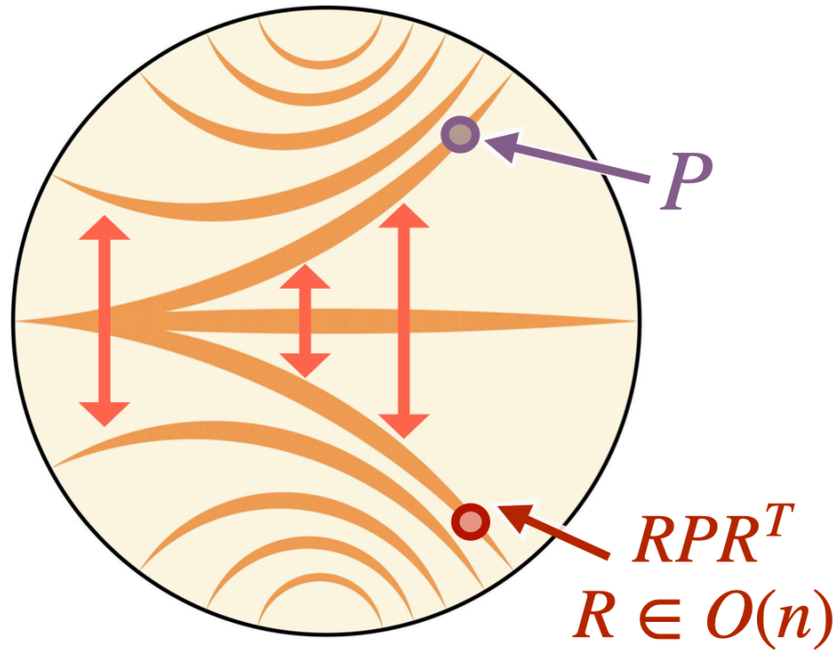
Isometries

- We also provide a way to learn rotations and reflections in SPD

- Rotations $$\mathrm{Rot}(\vec{\theta}) = \prod_{i<j} R^{+}_{ij}(\theta_{ij})$$



- Reflections $$\mathrm{Refl}(\vec{\theta}) = \prod_{i<j} R^{-}_{ij}(\theta_{ij})$$



Knowledge graph completion

we aim to complete the knowledge graph by predicting the missing entities or relationships

- Operations
  - Scaling

$$\phi(h,r,t) = -d((\mathbf{M}_r \otimes \mathbf{H}) \oplus \mathbf{R}, \mathbf{T})^2 + b_h + b_t$$

  Rotations
  - Reflections

$$\phi(h,r,t) = -d((\mathbf{M}_r \odot \mathbf{H}) \oplus \mathbf{R}, \mathbf{T})^2 + b_h + b_t$$

- Distances
  - Riemannian
  - Finsler One
- Baselines: scaling, rotations and reflections
  - Euclidean space
  - Hyperbolic spaces
  - Complex space

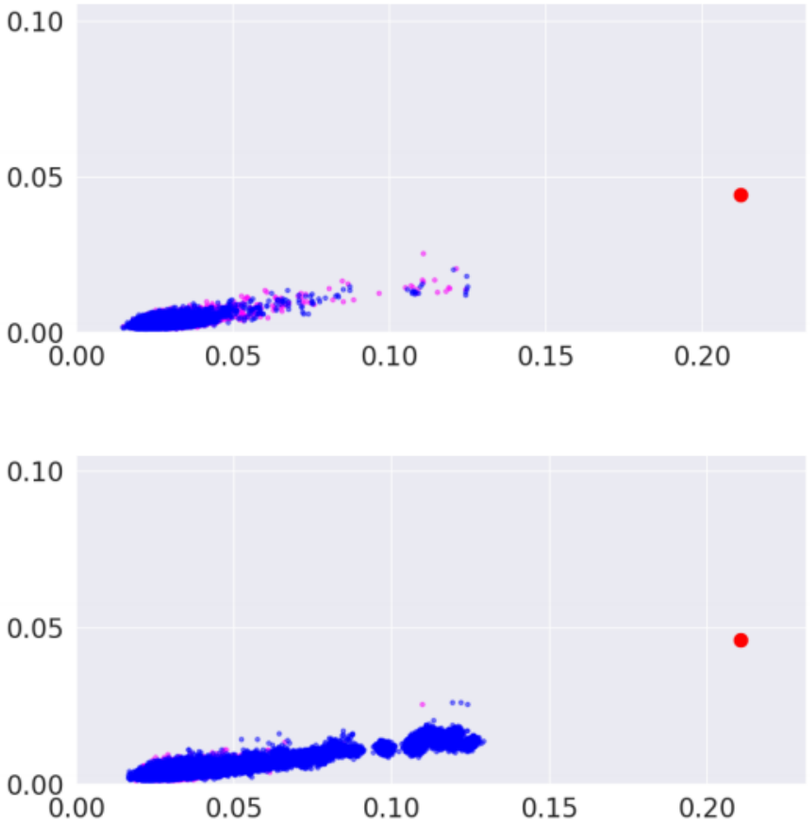
Results for Knowledge graph completion

Operation	Model	WN18RR				FB15k-237			
		MRR	HR@1	HR@3	HR@10	MRR	HR@1	HR@3	HR@10
Scaling	MuRE	47.5	43.6	48.7	55.4	33.6	24.5	37.0	52.1
	MuRP	48.1	**44.0**	49.5	56.6	33.5	24.3	36.7	51.8
	$\text{SPD}^{R}_{\text{Sca}}$	48.1	43.1	50.1	57.6	**34.5**	**25.1**	**38.0**	**53.5**
	$\text{SPD}^{F_1}_{\text{Sca}}$	**48.4**	42.6	**51.0**	**59.0**	32.9	23.6	36.3	51.5
Rotations	RotC	47.6	42.8	49.2	57.1	33.8	24.1	37.5	53.3
	RotE	49.4	44.6	51.2	58.5	**34.6**	**25.1**	**38.1**	**53.8**
	RotH	**49.6**	**44.9**	**51.4**	**58.6**	34.4	24.6	38.0	53.5
	$\text{SPD}^{R}_{\text{Rot}}$	46.2	39.7	49.6	57.8	32.9	23.6	36.3	51.6
	$\text{SPD}^{F_1}_{\text{Rot}}$	40.9	30.5	48.2	57.3	32.1	22.9	35.4	50.5
Reflections	RefE	47.3	43.0	48.5	56.1	**35.1**	**25.6**	**39.0**	**54.1**
	RefH	46.1	40.4	48.5	56.8	34.6	25.2	38.3	53.6
	$\text{SPD}^{R}_{\text{Ref}}$	48.3	44.0	49.7	56.7	32.5	23.4	35.6	51.0
	$\text{SPD}^{F_1}_{\text{Ref}}$	**48.7**	**44.3**	**50.1**	**57.4**	31.6	22.5	34.6	50.0

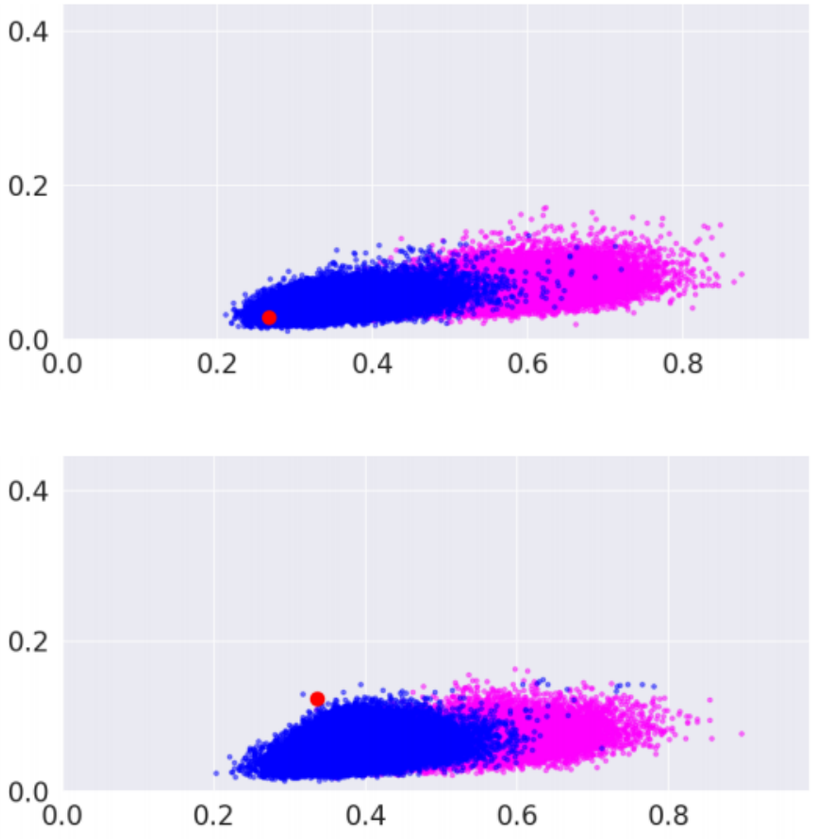
- SPD scalling outperforms Euclidean and hyperbolic models
- Competitive results for rotations and reflections with **significantly less dimensions**
- In many cases **Finsler one metric** outperforms the Riemannian distance

Visualisation of embeddings

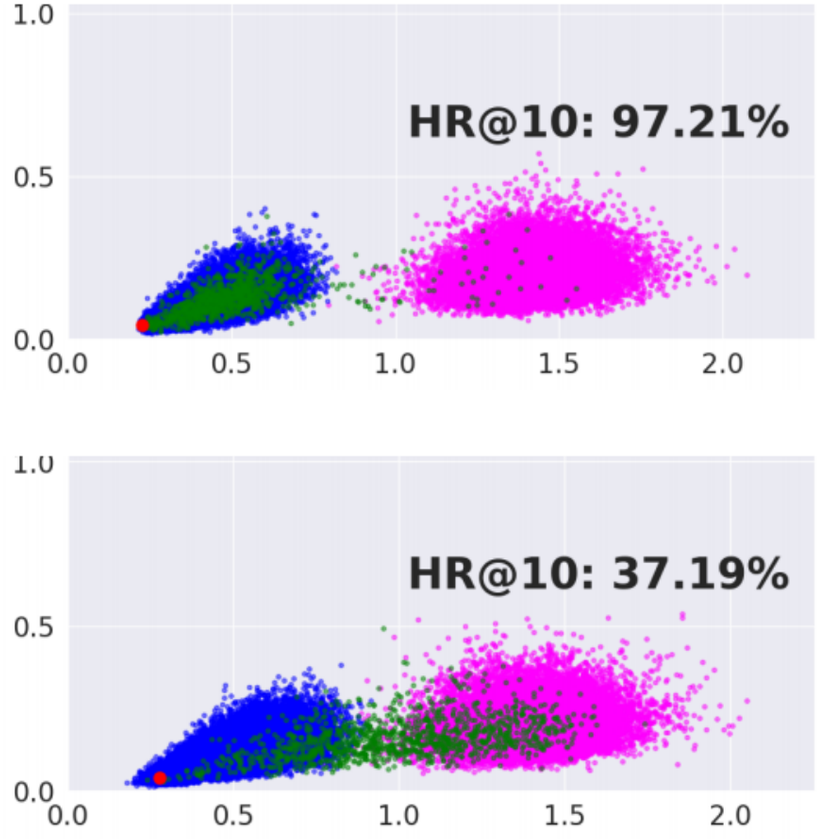
5 Epochs



50 Epochs



3000 Epochs



- **Train**, **negative** and **validation** triples for WN18RR relationships
- Position of **validation** triples directly correlates with performance

Neural linear layers on SPD

- Linear layer: feature transformation + bias addition
  - Scaling
  - Rotation
  - Reflection

$$y = Wx + b$$

$$y = W \otimes x \oplus b$$

- Experiments on Question Answering
  - Training word embeddings on SPD

- Model
  - Model question/answer as word embeddings summation followed by a linear transformation

$$\mathbf{Q} = T(\bigoplus_{i=1}^{n} t_i^q) \oplus B$$

  - Rank question-answer similarity

$$\text{sim}(q, a) = -w_f d(\mathbf{Q}, \mathbf{A}) + w_b$$

Results for Question Answering

Model	TrecQA MRR	TrecQA H@1	WikiQA MRR	WikiQA H@1
Euclidean	55.9±2.0	41.0±2.0	43.4±0.3	22.4±1.1
Hyperbolic	58.0±1.3	39.3±2.0	44.0±0.4	22.8±0.6
$\text{SPD}^{R}_{\text{Sca}}$	55.4±0.1	37.1±0.1	**45.5±0.5**	24.4±1.1
$\text{SPD}^{F_1}_{\text{Sca}}$	57.1±0.7	38.6±0.2	44.8±0.5	24.0±0.6
$\text{SPD}^{R}_{\text{Rot}}$	58.7±1.5	41.4±2.9	44.6±0.6	23.6±0.6
$\text{SPD}^{F_1}_{\text{Rot}}$	58.1±0.5	**43.6±1.0**	43.7±0.4	23.8±0.8
$\text{SPD}^{R}_{\text{Ref}}$	57.3±0.3	40.7±1.1	43.9±0.7	23.4±2.0
$\text{SPD}^{F_1}_{\text{Ref}}$	**59.6±0.5**	42.1±1.0	44.7±1.2	**25.0±2.5**

- SPD models **outperform baselines** in all cases
- **Embeddings in SPD manifolds** is effective for downstream tasks
- We showcase how to build **linear layers on SPD**

Summary

- Geometric Deep Learning arose as a field in machine learning, with a lot of interest in concepts from geometry, Lie groups etc.

- **Symmetric spaces** are a great tool because their geometry is richer than Euclidean or hyperbolic geometry, but at the same time they are algorithmically tractable.

- The combination of **Finsler** and Riemannian **metrics** gives you better performance for graph embeddings as well as in concrete tasks.

- The **vector-valued distance** function in a symmetric space gives you a completely new tool to analyze your graph embedding.

- **Gyrocalculus** can be adapted to obtain arithmetic operations that respect the geometry, and neural network architectures

- There are still many tools and ideas from geometry, topology and geometric group theory that are not yet applied….