Overcoming Machine Learning's Data Bottlenecks

Fred Sala

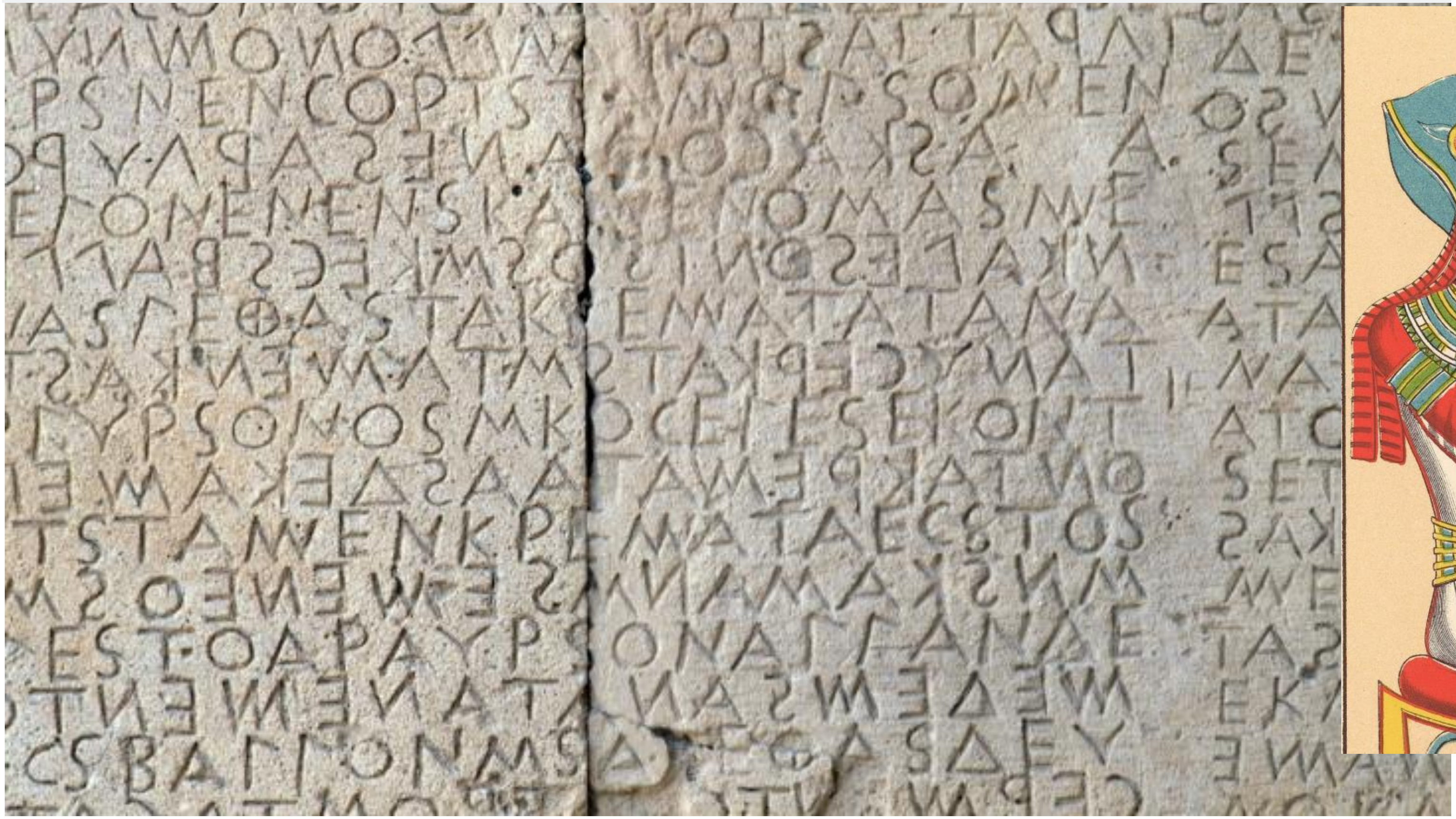
WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

ML Progress

WRITTEN IN STONE —

DeepMind’s new AI gives historians a powerful new tool to interpret the past

Google launches hieroglyphics translator powered by AI

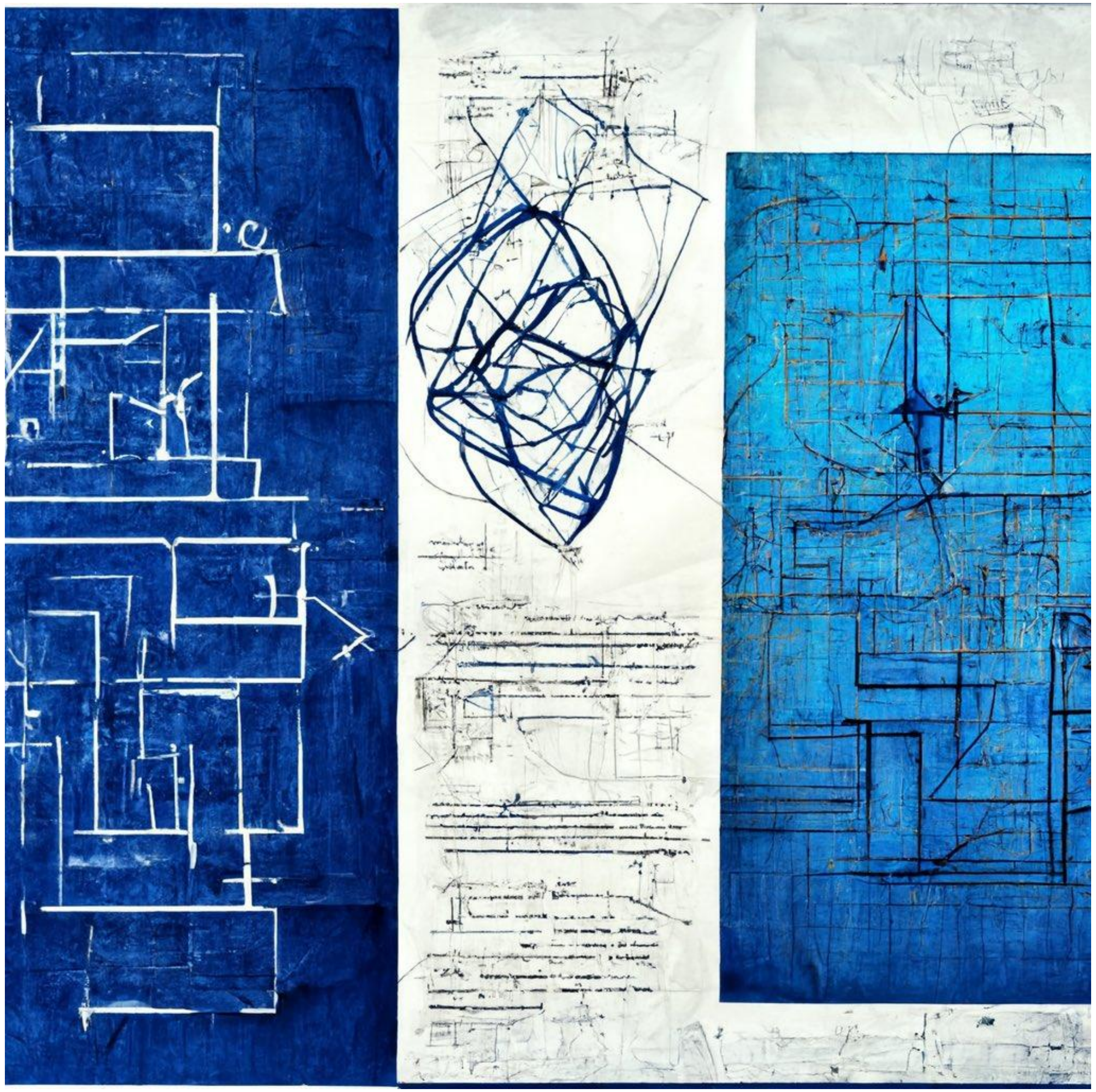


Google Translate / Life Collection

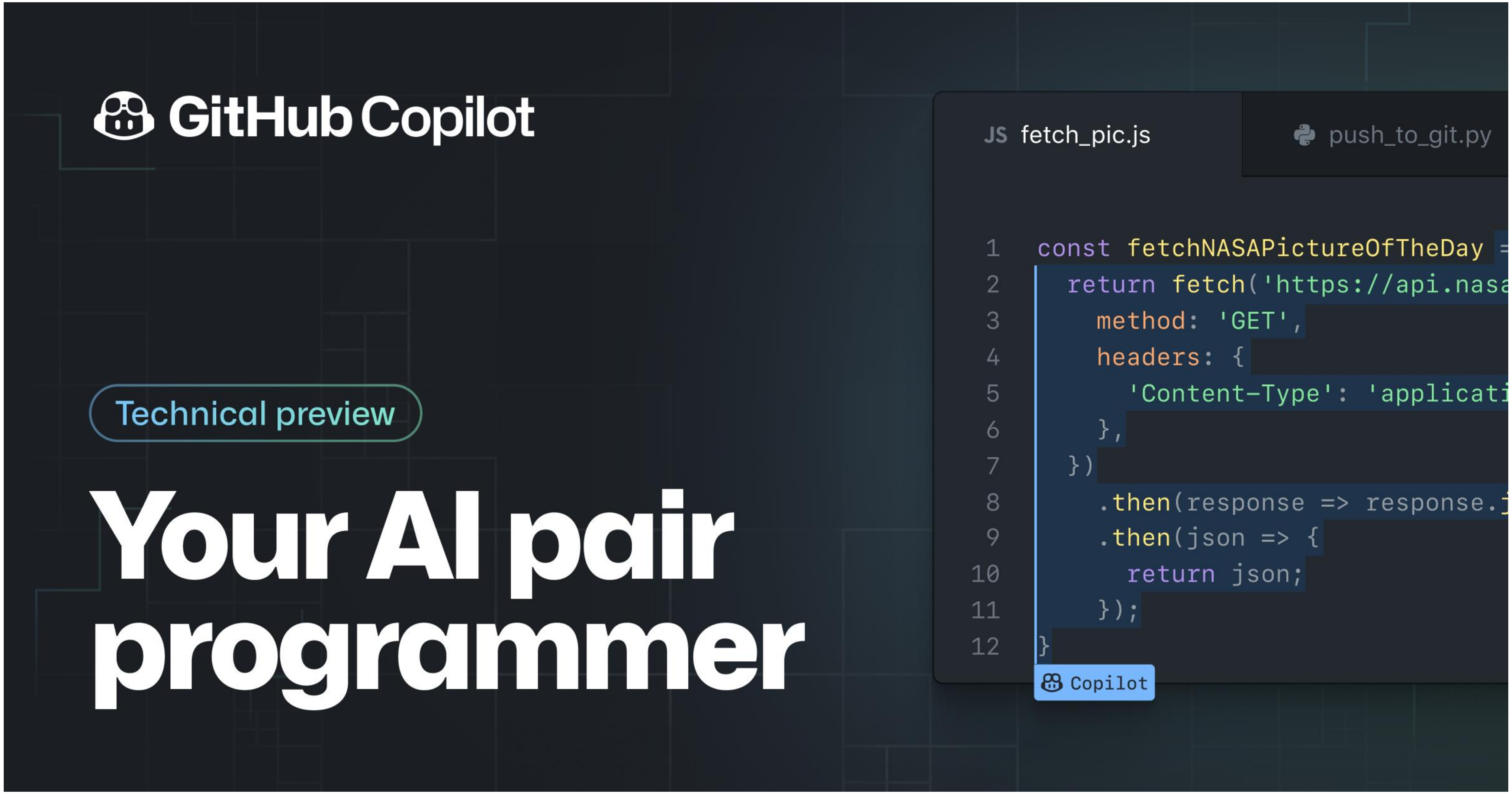
DeepMind / DEA / ARCHIVIO J. LANGE / Contributor / Getty Images

ML Progress

Parker Molloy / MidJourney



ML Progress


GitHub Copilot
Technical preview
Your AI pair programmer
JS fetch_pic.js
push_to_git.py
const fetchNASAPictureOfTheDay
return fetch('https://api.nasa
method: 'GET',
headers: {
'Content-Type': 'applicat
},
})
.then(response => response.
.then(json => {
return json;
});
}
Copilot

…and ML Promise

Results

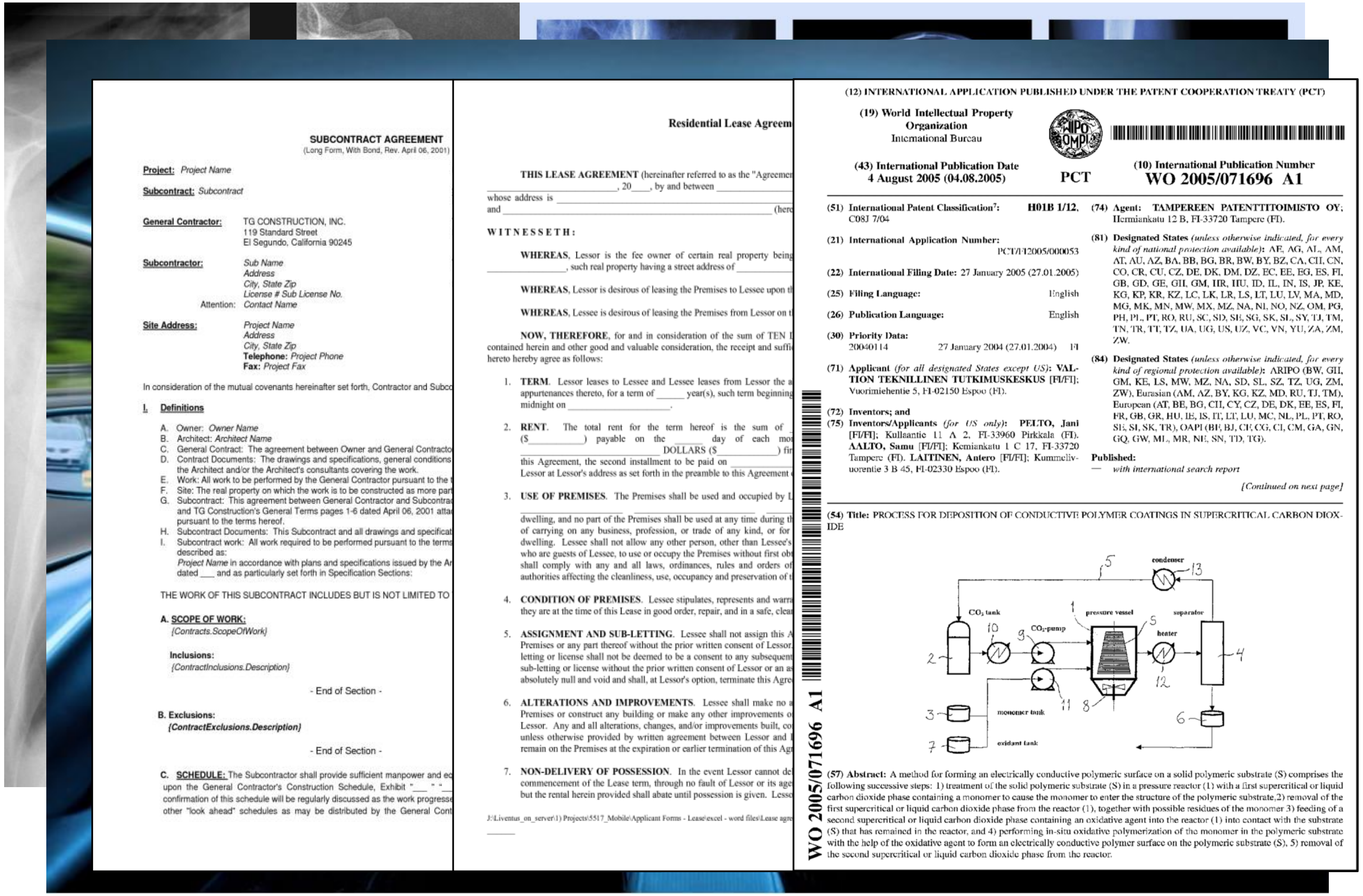
- Presence of tumor
- Bone break
- Additional conditions

ML Promise

Results:

- Traffic Jam Ahead
- Take Next Exit
- Pedestrian Detected

ML Promise

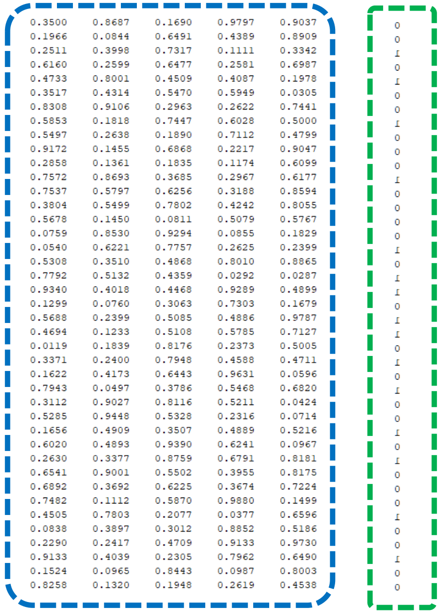


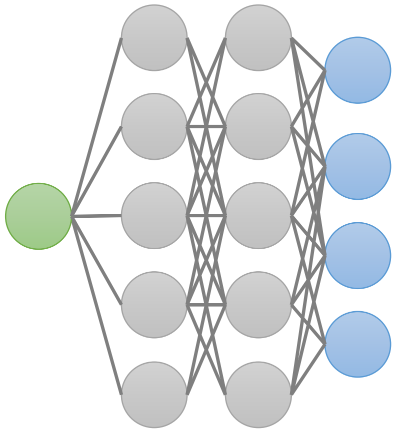
Results:

- Overlap with US10318434B2
- Related to US8847335B2

Supervised Machine Learning

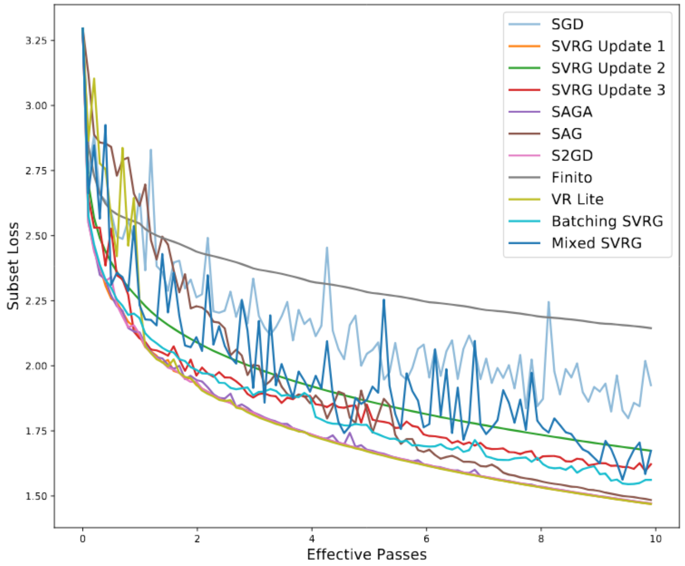
- Today’s supervised ML pipeline components:





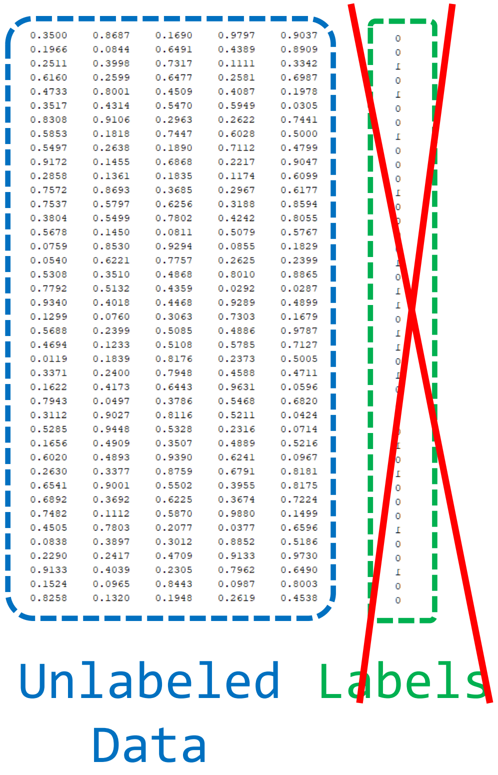
Model

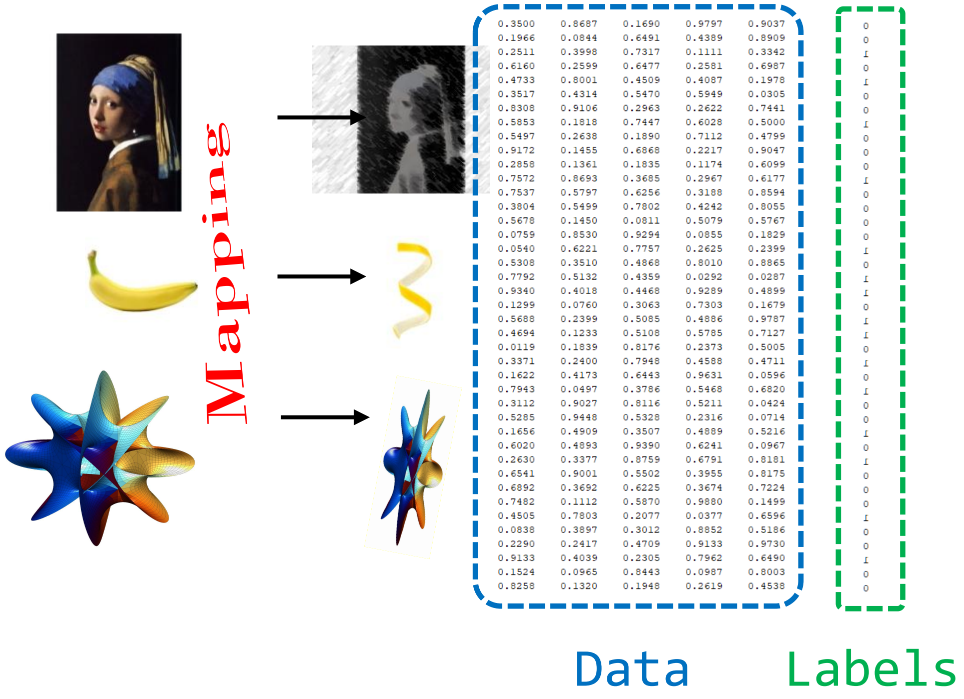
$$-\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)\right]$$



Data Labels Loss Training

Data Bottlenecks

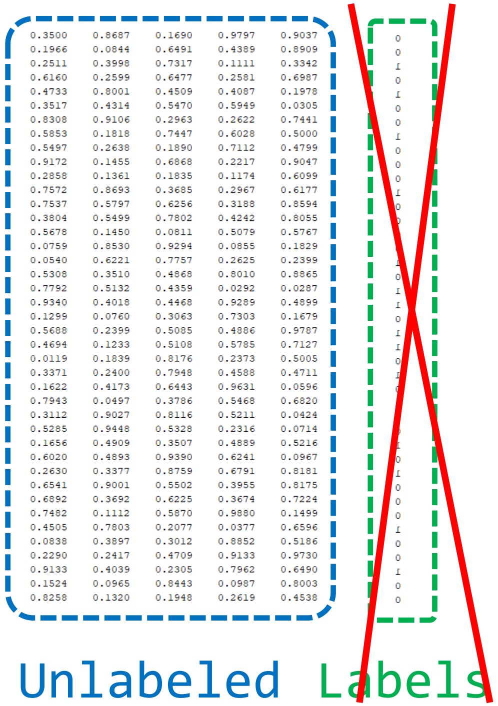




Bottleneck 1: Getting Labels

Bottleneck 2: Distortion

Data Bottlenecks



Bottleneck 1: Getting Labels

Bottleneck 2: Distortion

The Need for Labels...

Modern supervised models need **lots** of labeled data

The Need for Labels...

Modern supervised ML models need **lots** of labeled data

Tons of unlabeled data, but labeling is

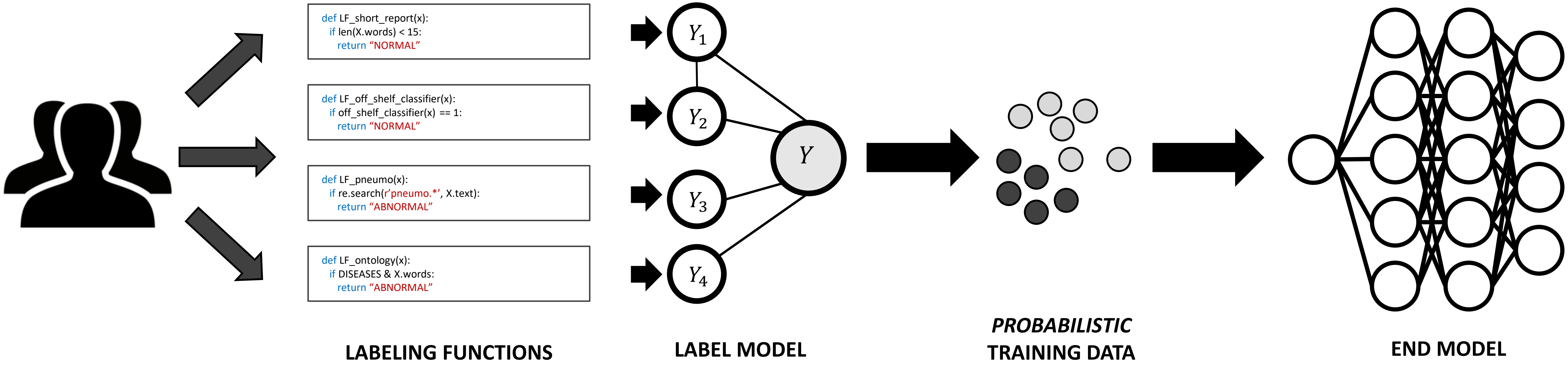
- Expensive,
- Static,
- **Slow**.

Crawford and Paglen

Weak Supervision To the Rescue

- **Weak supervision**: reduce the label bottleneck
  - Some of the users:

The Snorkel / Weak Supervision Pipeline

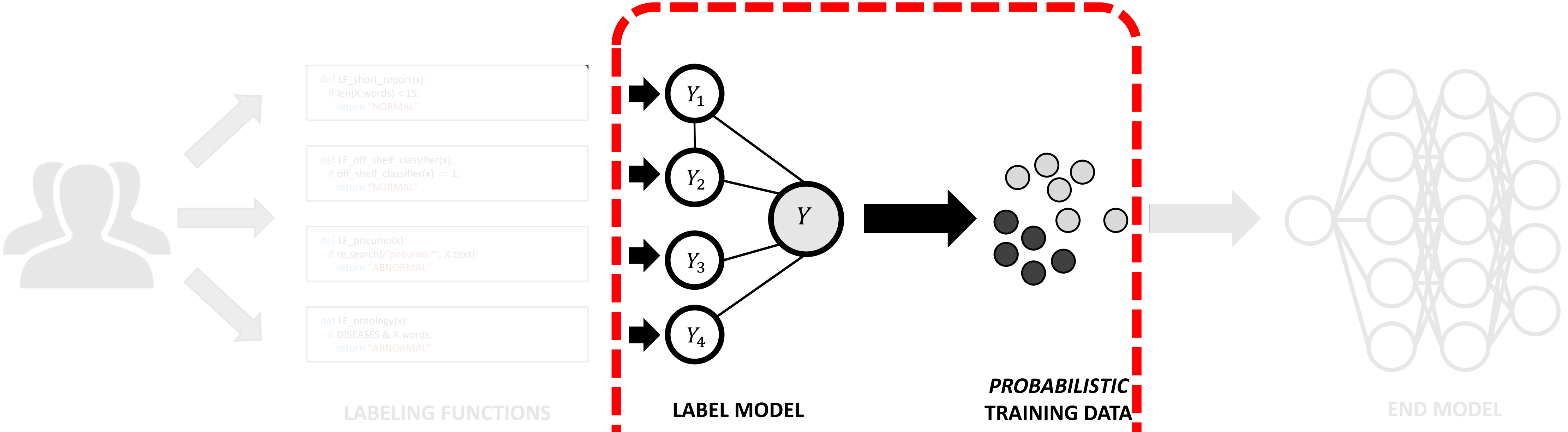


1. Users write ***labeling functions*** to create noisy labels

2. We **model** and **combine** these labels

3. The generated labels are used to **train a downstream model**

The Snorkel / Weak Supervision Pipeline



1. Users write ***labeling functions*** to create noisy labels

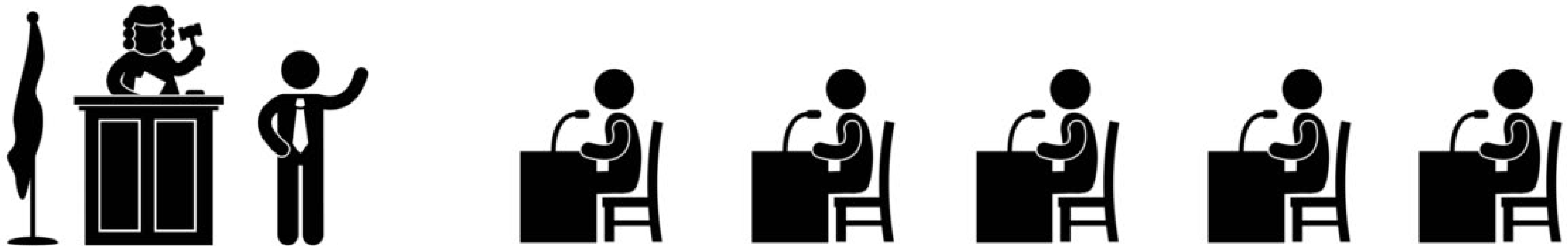
2. We **model** and **combine** these labels

3. The generated labels are used to **train a downstream model**

Intuition

Witnesses

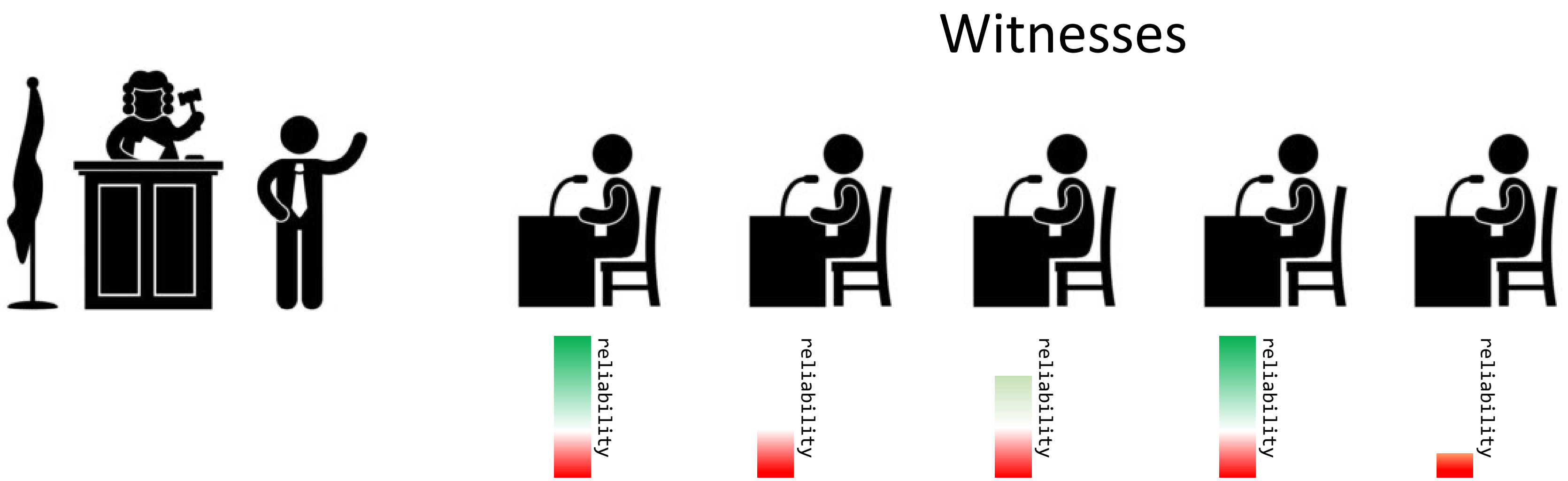
Result



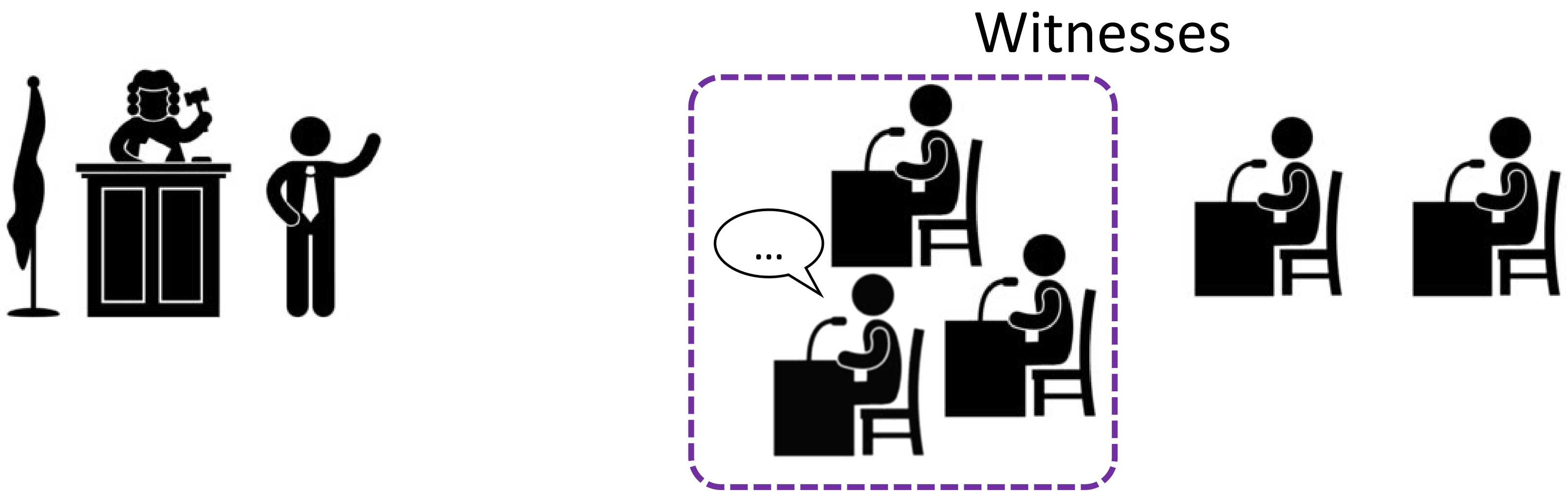
Votes

Naïve approach: **majority vote**

Improving on Majority Vote



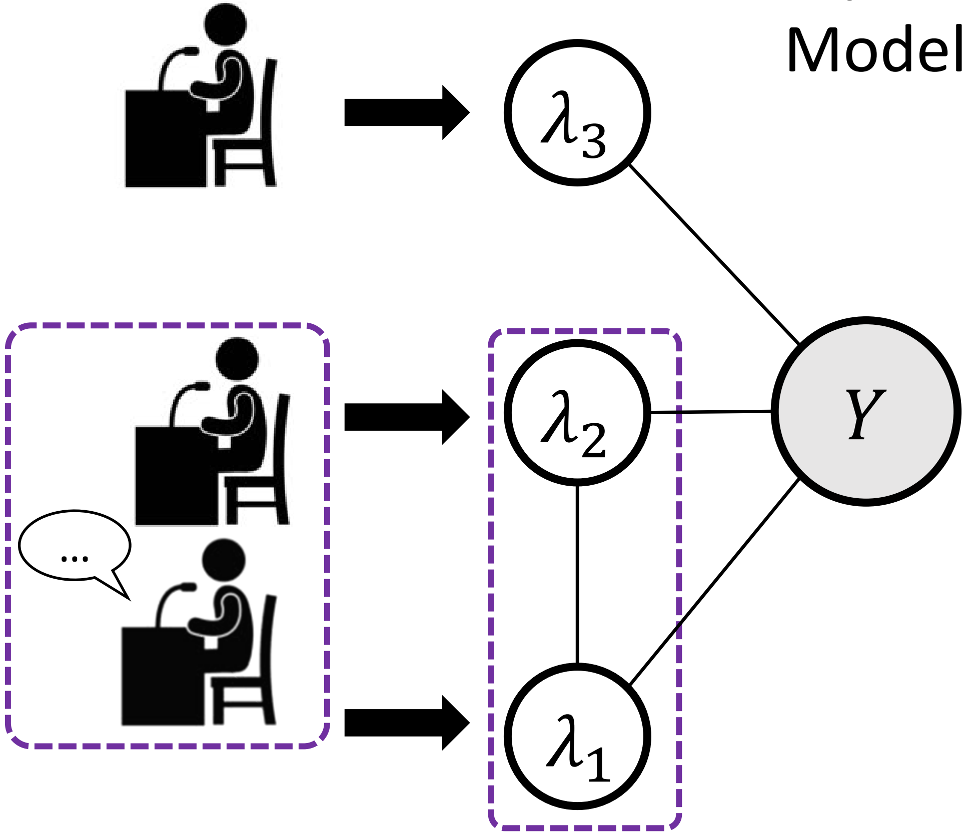
1. Incorporate **accuracies**



2. Incorporate **correlations**

Label Model

Graphical Model



Parameters

1. **Accuracies** $\mathbf{E}[\lambda_i Y]$
2. **Correlations** $\mathbf{E}[\lambda_i \lambda_j]$

If we knew parameters… could do inference $P(Y|\lambda_1, \lambda_2, \dots, \lambda_m)$

Our goal: learn parameters, **without observing Y**

How Does WS Work?

Look for latent relationships with **observables**

$$\mathbb{E}[\lambda_1 \lambda_2] = \mathbb{E}[\lambda_1 Y]\mathbb{E}[\lambda_2 Y]$$

Observable: Rate of agreement/disagreement

Accuracy Parameters: Want to estimate these

How Does WS Work?

Exploit latent relationships with **observables**

$$\begin{cases} \mathbb{E}[\lambda_1 \lambda_2] = \mathbb{E}[\lambda_1 Y]\mathbb{E}[\lambda_2 Y] \\ \mathbb{E}[\lambda_1 \lambda_3] = \mathbb{E}[\lambda_1 Y]\mathbb{E}[\lambda_3 Y] \\ \mathbb{E}[\lambda_2 \lambda_3] = \mathbb{E}[\lambda_2 Y]\mathbb{E}[\lambda_3 Y] \end{cases}$$

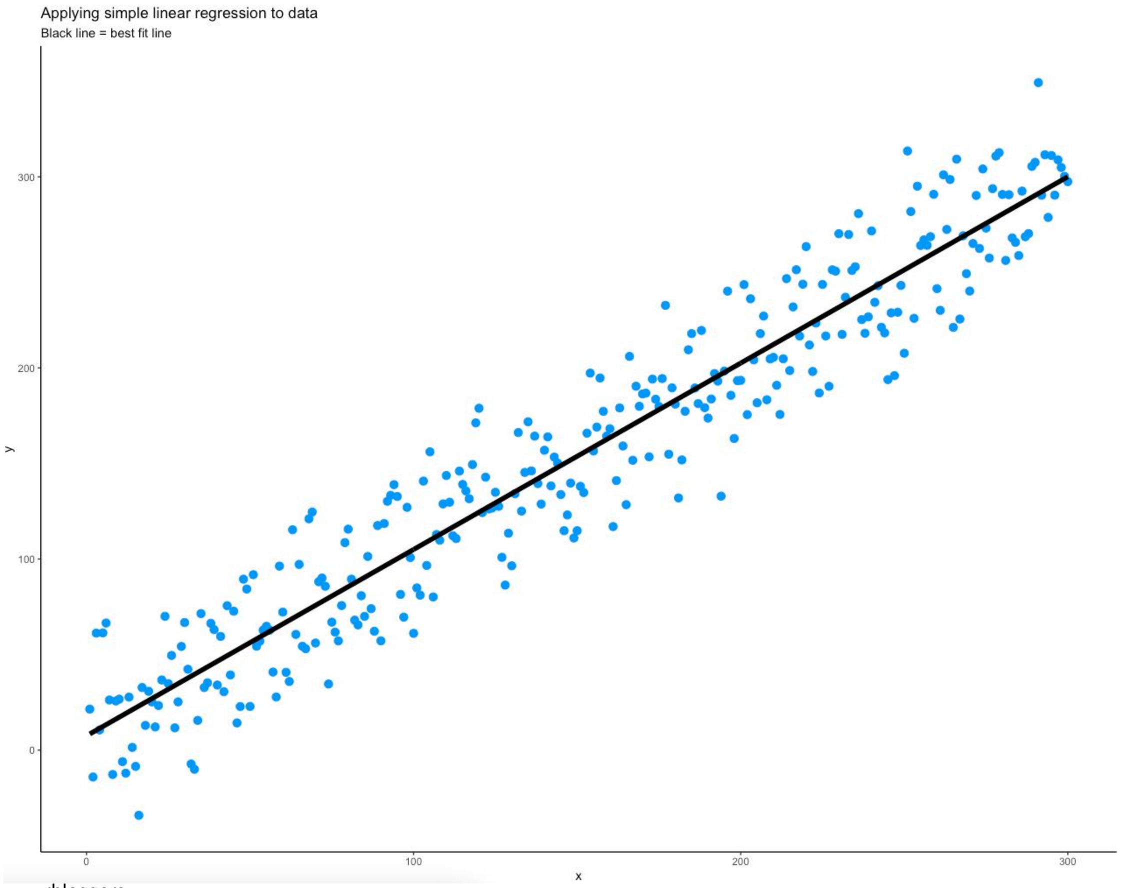
System in three accuracies, lhs are three pairwise rates

Multiply first two equations, divide by third

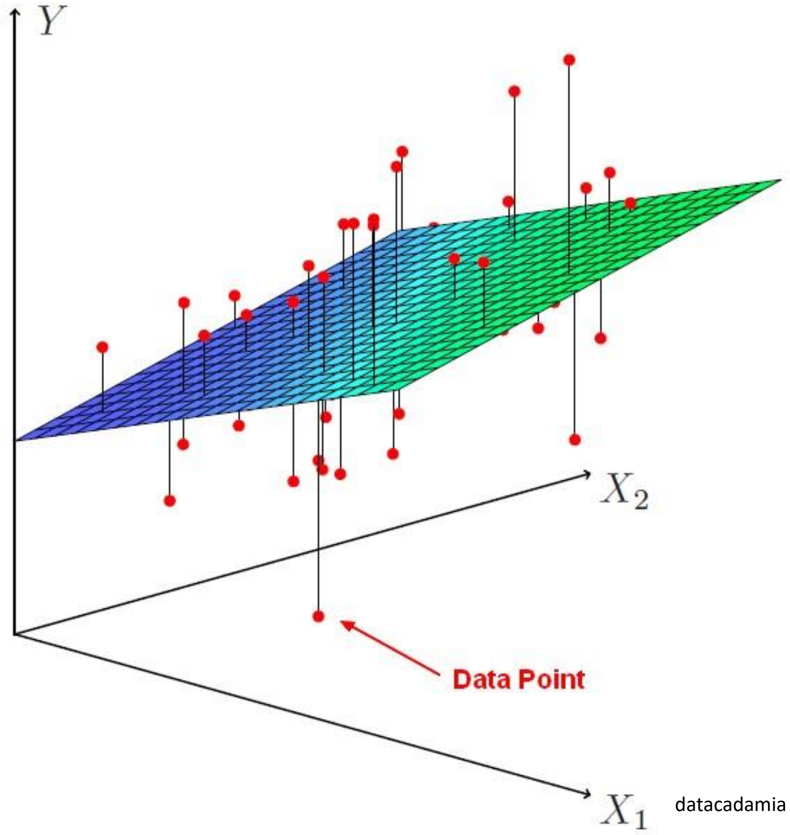
$$|\mathbb{E}[\lambda_1 Y]| = \sqrt{\frac{\mathbb{E}[\lambda_1 \lambda_2]\mathbb{E}[\lambda_1 \lambda_3]}{\mathbb{E}[\lambda_2 \lambda_3]}}$$

But ML is Far More Diverse…

- Labels can be **real-valued**



rbloggers



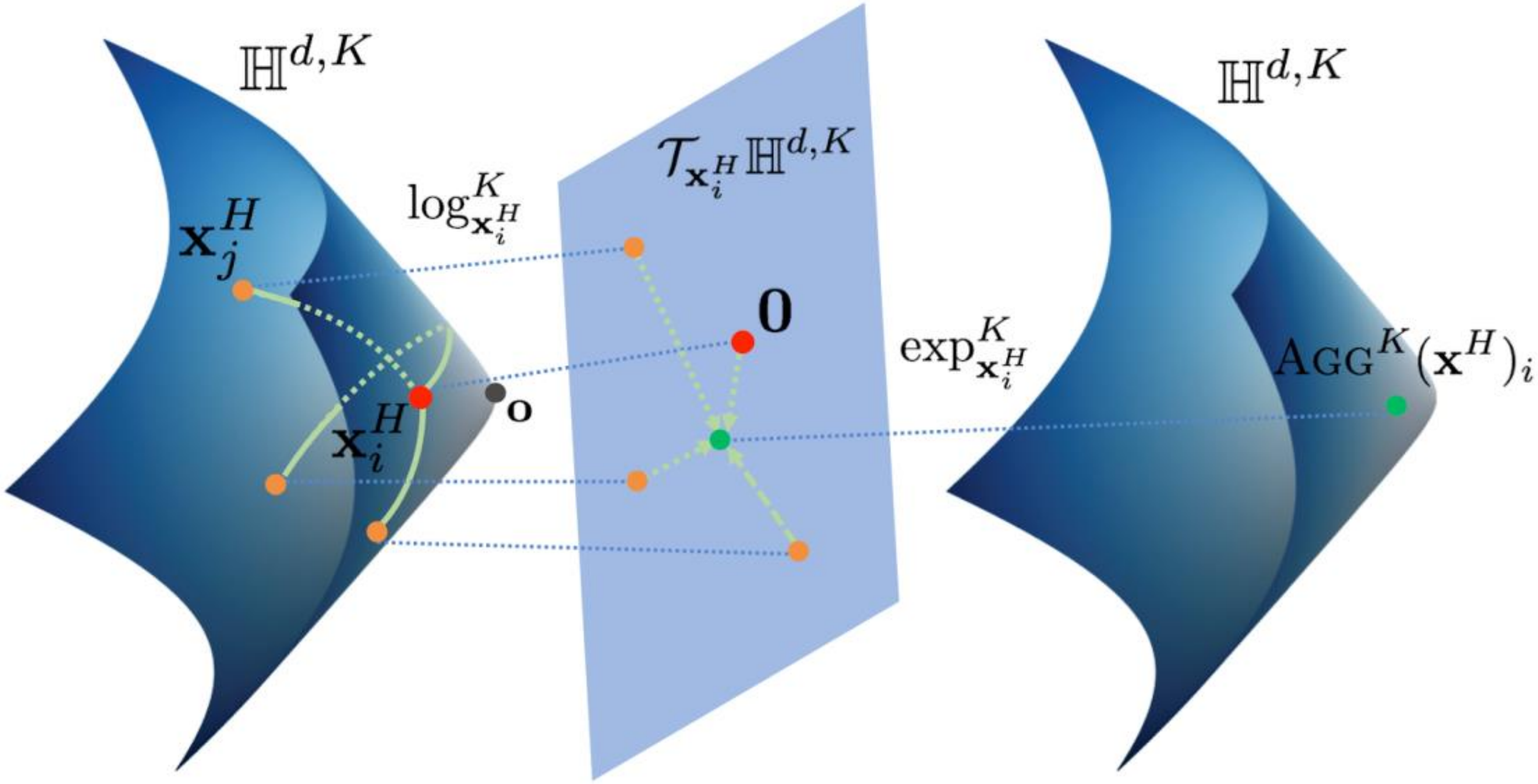
datacadamia

Labels Can Be: **Rankings**

Laughing Squid

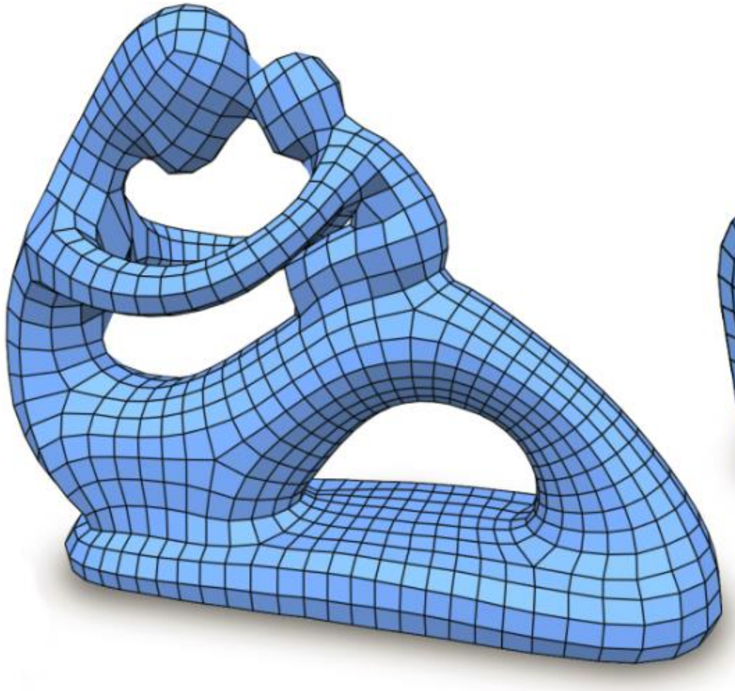
Casey Newell

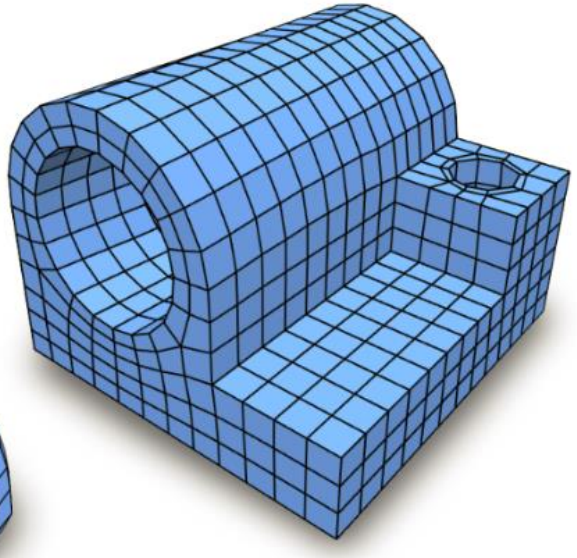
Labels Can Be: **Hyperbolic Space** Points



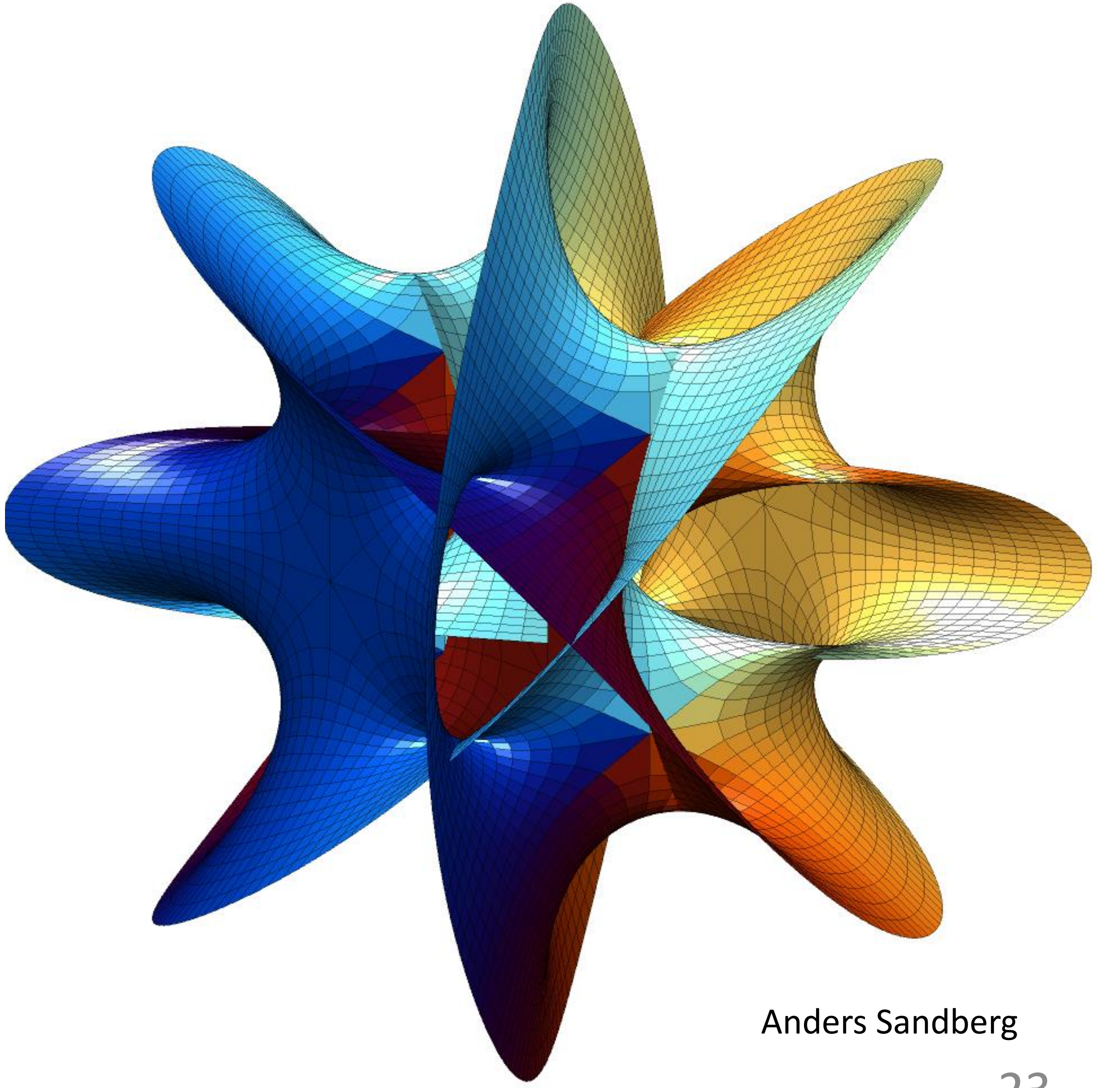
Hyperbolic Graph Convolution Networks, NeurIPS ‘19

Labels Can Be: **Manifold** Points



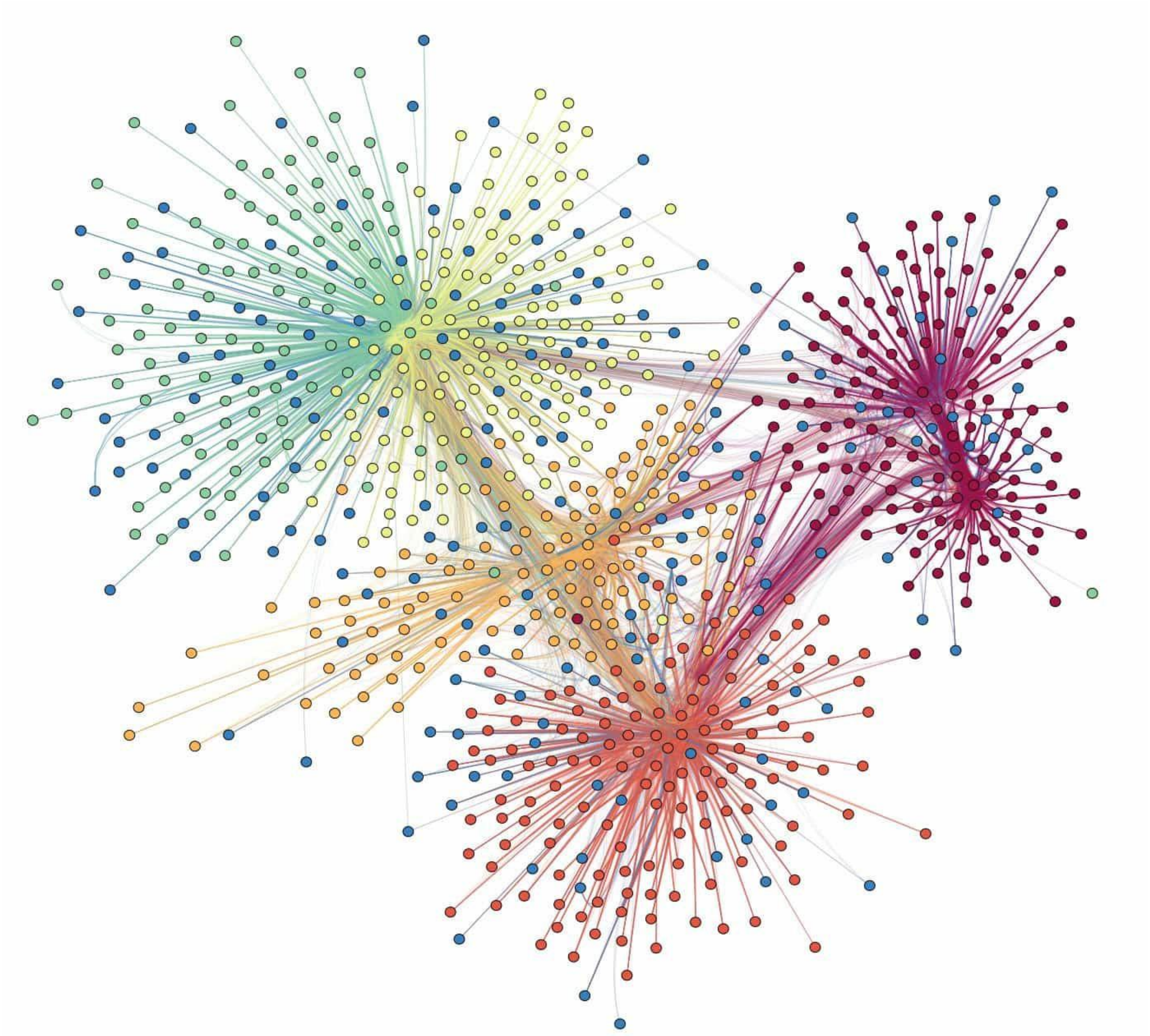


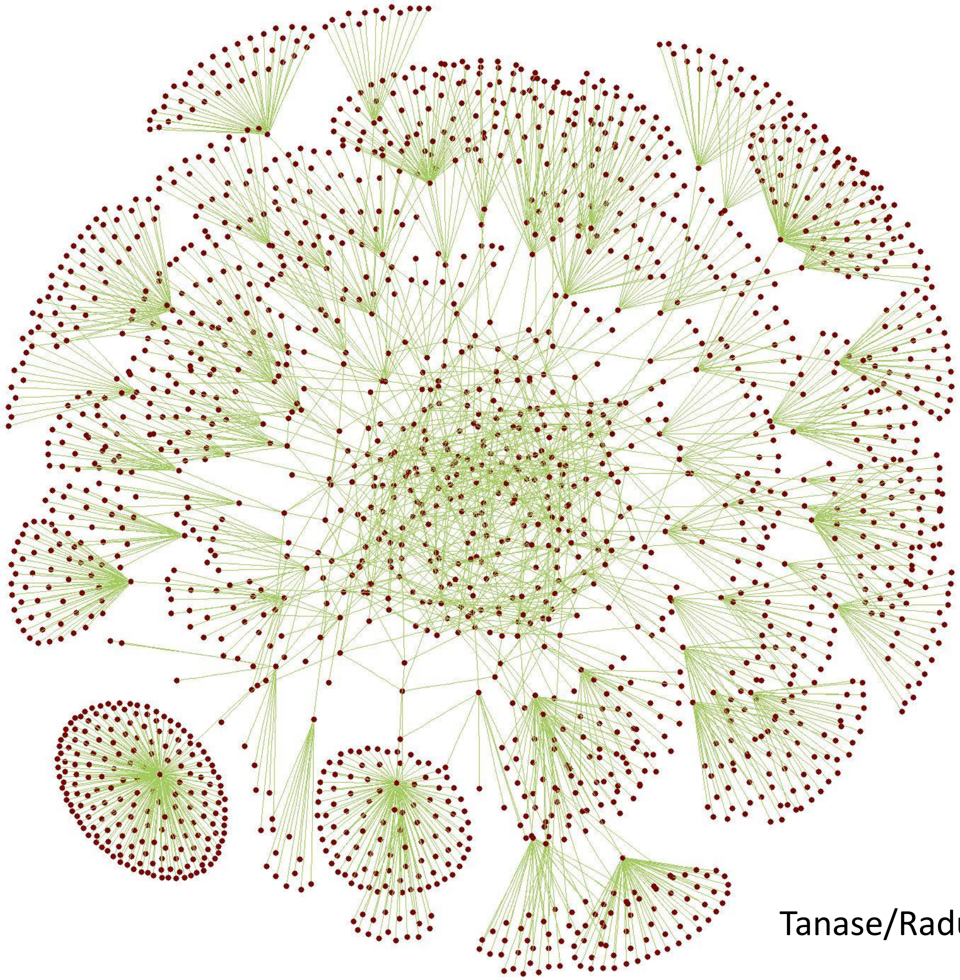
Marcel Campen



Anders Sandberg

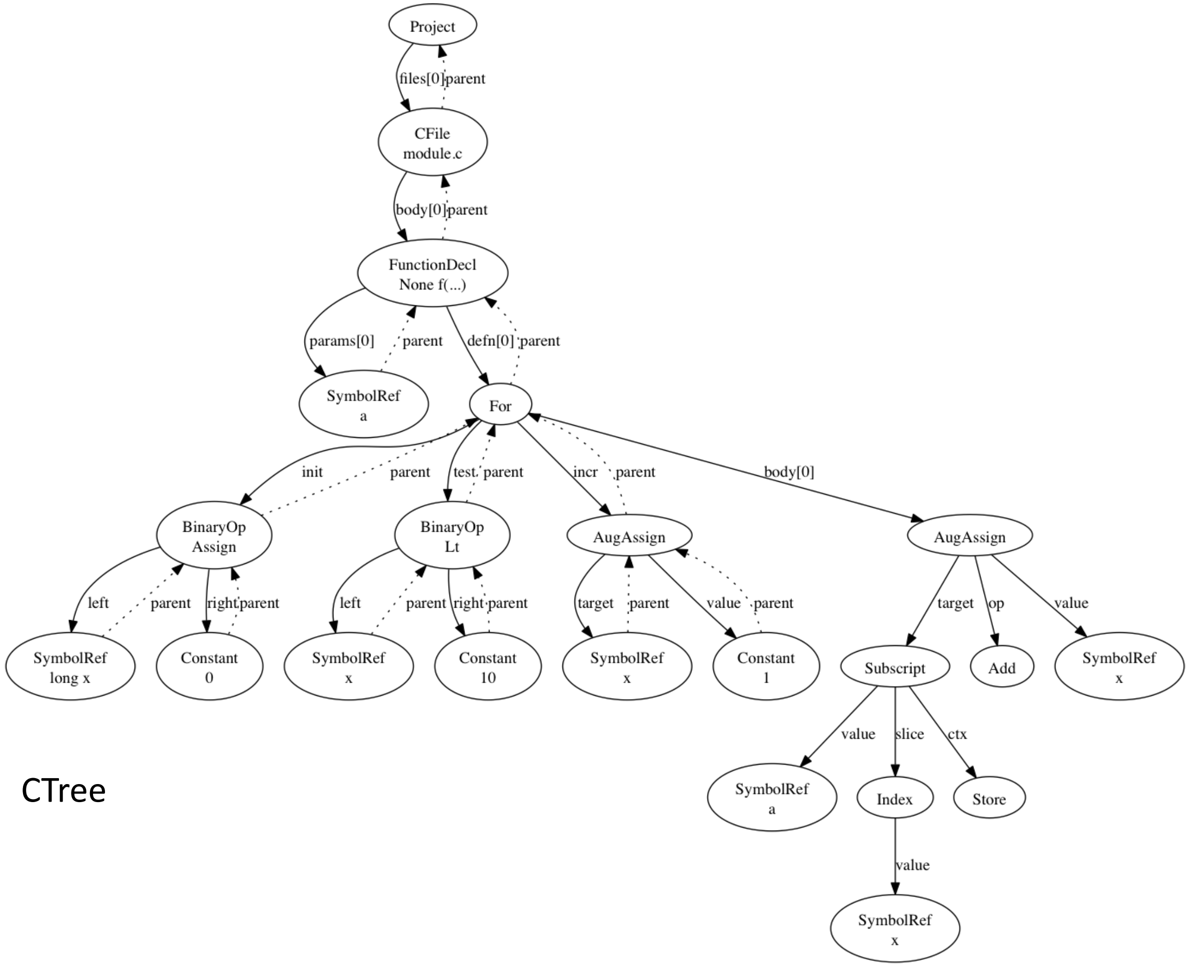
Labels Can Be: **Graphs**



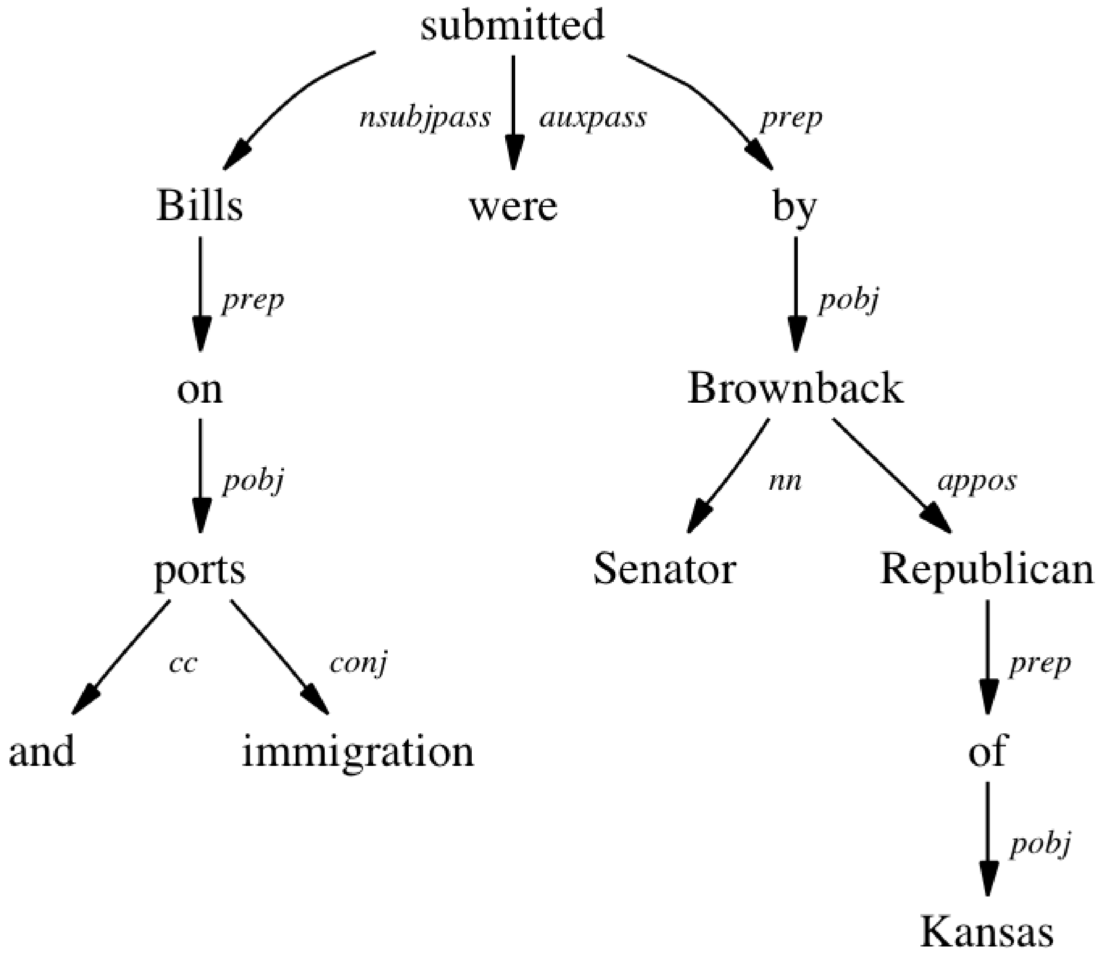


Tanase/Radu

Labels Can Be: **Trees**



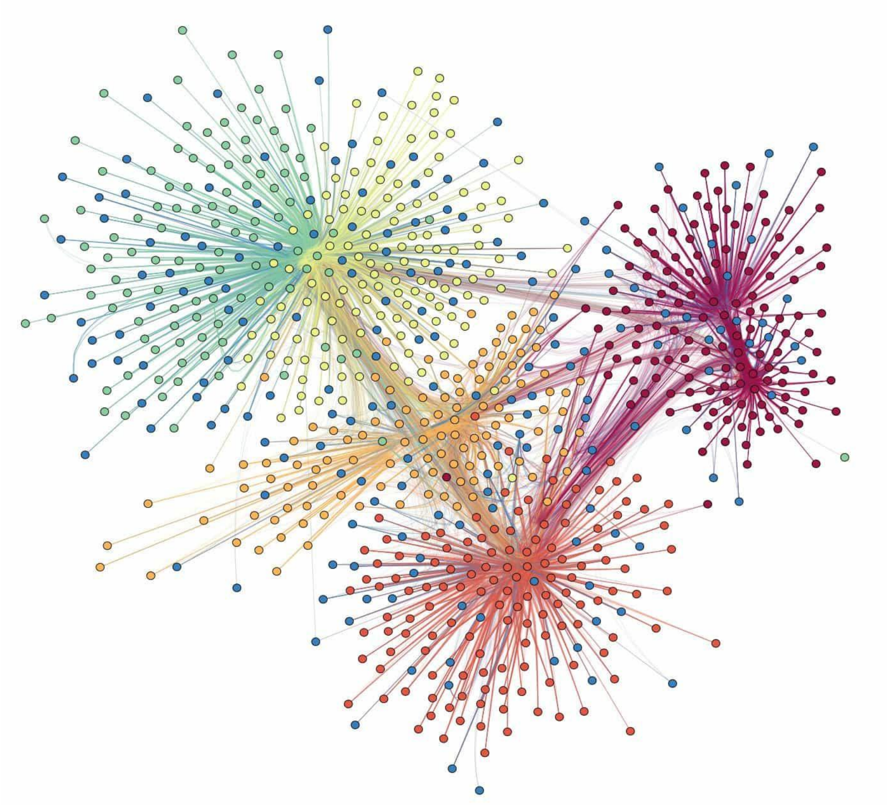
CTree

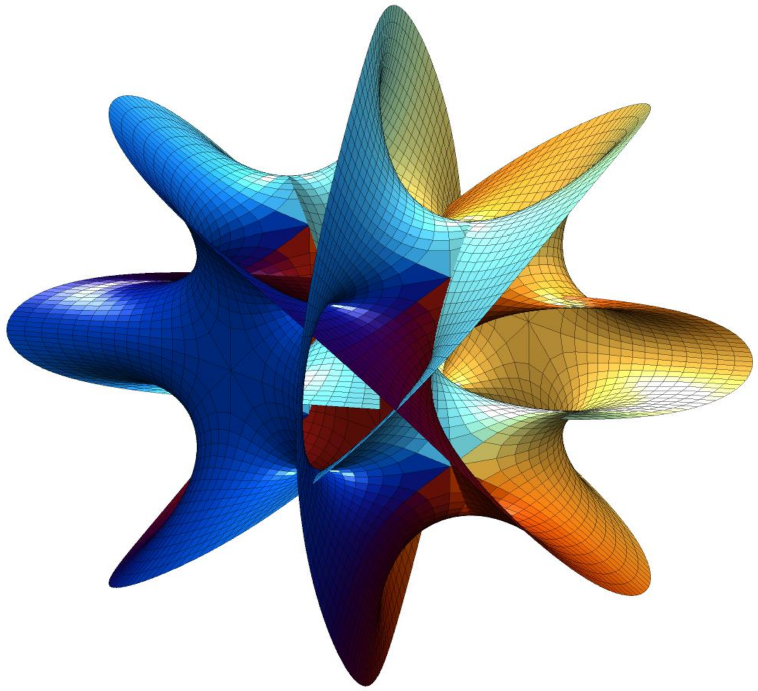


Stanford NLP

How Do We Do This for Diverse Y's?

What does this multiplication even mean? $\mathbb{E}[\lambda_1 Y]$





Another Way of Encoding **Accuracy**

Common to our labels: a way to measure **distance**

- Ranked lists: various ways to measure closeness
- Manifolds: often equipped with a distance
- Graphs: edges in common

- Space with a distance: **metric space**

$$\frac{1}{Z}\exp\left(\theta_i \underbrace{\lambda_i Y}_{\text{Binary accuracy term}}\right) \rightarrow \frac{1}{Z}\exp\left(\theta_i \underbrace{d(\lambda_i, Y)}_{\text{General accuracy term}}\right)$$

Distances Generalize Majority Vote

Before modeling accuracies… what's a **majority vote**?

- All weak outputs might be different!
- Need to use distance… the natural choice:

$$(1, 2, 3, 4, 5)$$
$$(2, 1, 3, 4, 5)$$
$$(3, 2, 1, 4, 5)$$
$$(3, 2, 4, 1, 5)$$

$$\hat{Y} = \arg\min_{y} \sum_{i=1}^{m} d(y, \lambda_i)$$

How Do We Find Accuracies?

Need more relationships between latent terms and observables…

$$\mathbb{E}[\lambda_1 \lambda_2] = \mathbb{E}[\lambda_1 Y]\mathbb{E}[\lambda_2 Y]$$

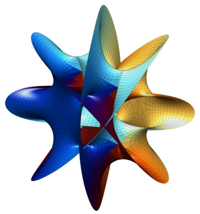
Observable: Rate of agreement

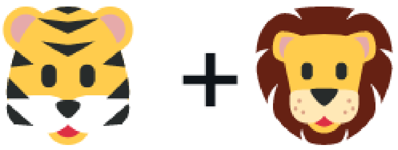
Accuracy Parameters

$$\mathbb{E}[d(\lambda_1, \lambda_2)] = \mathbb{E}[d(\lambda_1, Y)] + \mathbb{E}[d(\lambda_2, Y)]$$

Needs strong assumptions… alternatives exist

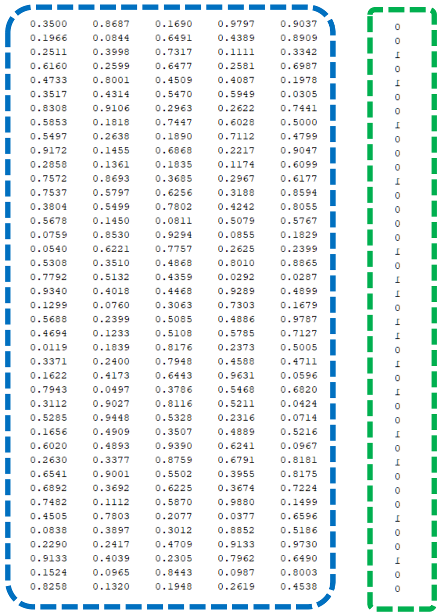
Some of Our Work





- **Snorkel MeTaL**: Training complex models with multi-task weak supervision, RHD**S**PR, **AAAI ‘19**
- Multi-Resolution Weak Supervision for Sequential Data, **S**VSFFKRXFPR, **NeurIPS ‘19**
- **FlyingSquid**: Fast and three-rious: Speeding up weak supervision with triplet methods, FC**S**HFR, **ICML ’20**
- Comparing the value of labeled and unlabeled data in method-of-moments latent variable estimation”, CCM**S**R, **AISTATS ‘21**
- Universalizing Weak Supervision, SLVR**S**, **ICLR ’22**
- **Liger**: “Shoring Up the Foundations: Fusing Model Embeddings and Weak Supervision”, CFAZ**S**FR, ‘22.

Data Bottlenecks



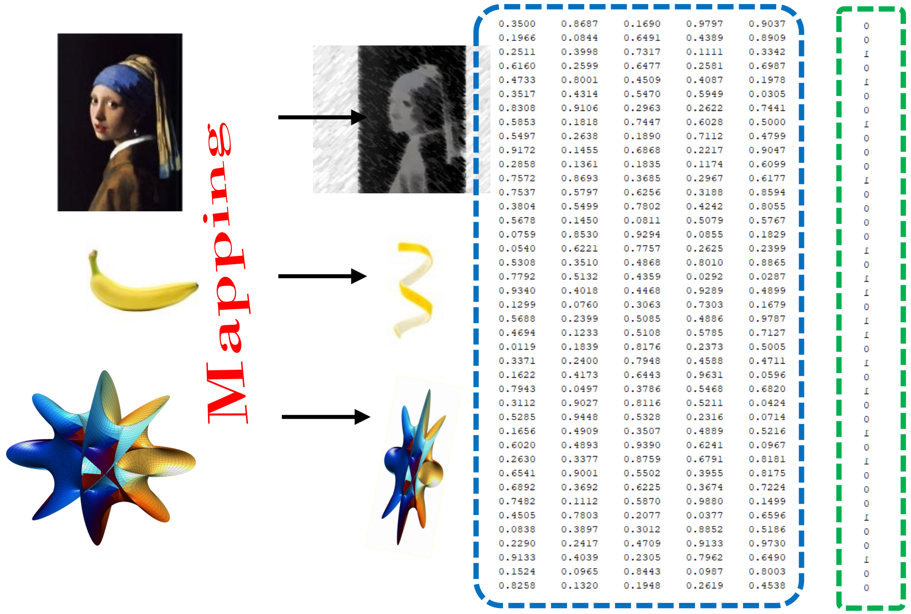
Unlabeled Data Labels

Bottleneck 1: Getting Labels **Bottleneck 2: Distortion**

Data Bottlenecks

Unlabeled Data   Labels

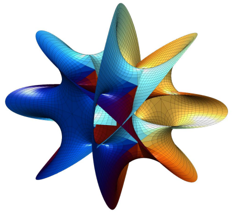
Bottleneck 1: Getting Labels



Data   Labels

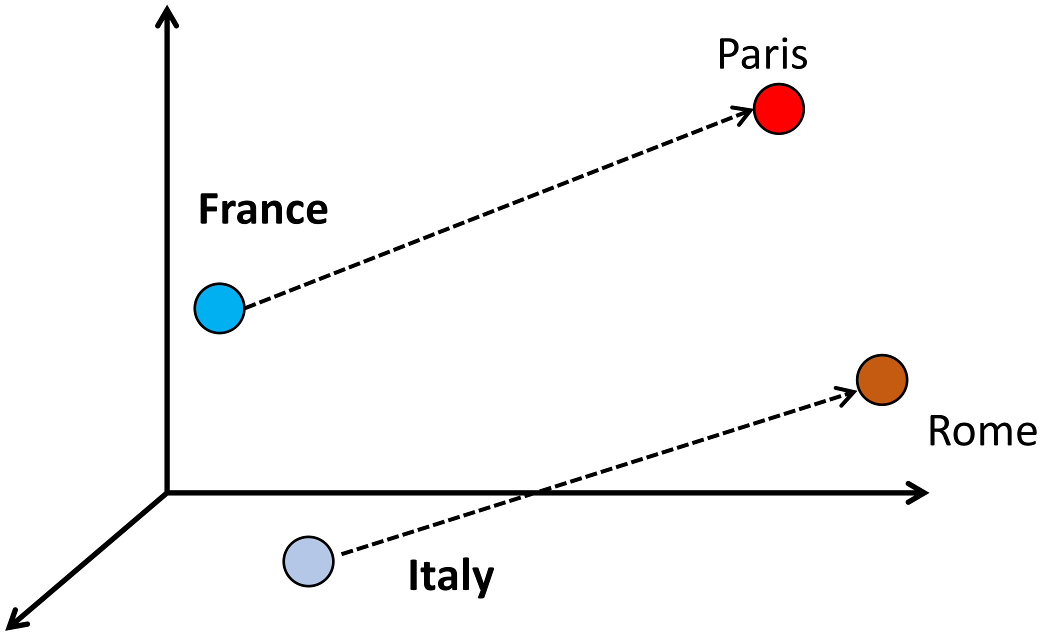
Bottleneck 2: Distortion

Embeddings



Mapping

0.3500	0.8687	0.1690	0.9797	0.9037
0.1966	0.0844	0.6491	0.4389	0.8909
0.2511	0.3998	0.7317	0.1111	0.3342
0.6160	0.2599	0.6477	0.2581	0.6987
0.4733	0.8001	0.4509	0.4087	0.1978
0.3517	0.4314	0.5470	0.5949	0.0305
0.8308	0.9106	0.2963	0.2622	0.7441
0.5853	0.1818	0.7447	0.6028	0.5000
0.5497	0.2638	0.1890	0.7112	0.4799
0.9172	0.1455	0.6868	0.2217	0.9047
0.2858	0.1361	0.1835	0.1174	0.6099
0.7572	0.8693	0.3685	0.2967	0.6177
0.7537	0.5797	0.6256	0.3188	0.8594
0.3804	0.5499	0.7802	0.4242	0.8055
0.5678	0.1450	0.0811	0.5079	0.5767
0.0759	0.8530	0.9294	0.0855	0.1829
0.0540	0.6221	0.7757	0.2625	0.2399
0.5308	0.3510	0.4868	0.8010	0.8865
0.7792	0.5132	0.4359	0.0292	0.0287
0.9340	0.4018	0.4468	0.9289	0.4899
0.1299	0.0760	0.3063	0.7303	0.1679
0.5688	0.2399	0.5085	0.4886	0.9787
0.4694	0.1233	0.5108	0.5785	0.7127
0.0119	0.1839	0.8176	0.2373	0.5005
0.3371	0.2400	0.7948	0.4588	0.4711
0.1622	0.4173	0.6443	0.9631	0.0596
0.7943	0.0497	0.3786	0.5468	0.6820
0.3112	0.9027	0.8116	0.5211	0.0424
0.5285	0.9448	0.5328	0.2316	0.0714
0.1656	0.4909	0.3507	0.4889	0.5216
0.6020	0.4893	0.9390	0.6241	0.0967
0.2630	0.3377	0.8759	0.6791	0.8181
0.6541	0.9001	0.5502	0.3955	0.8175
0.6892	0.3692	0.6225	0.3674	0.7224
0.7482	0.1112	0.5870	0.9880	0.1499
0.4505	0.7803	0.2077	0.0377	0.6596
0.0838	0.3897	0.3012	0.8852	0.5186
0.2290	0.2417	0.4709	0.9133	0.9730
0.9133	0.4039	0.2305	0.7962	0.6490
0.1524	0.0965	0.8443	0.0987	0.8003
0.8258	0.1320	0.1948	0.2619	0.4538



Continuous representations that

preserve structure & relationships

Preserving Relationships

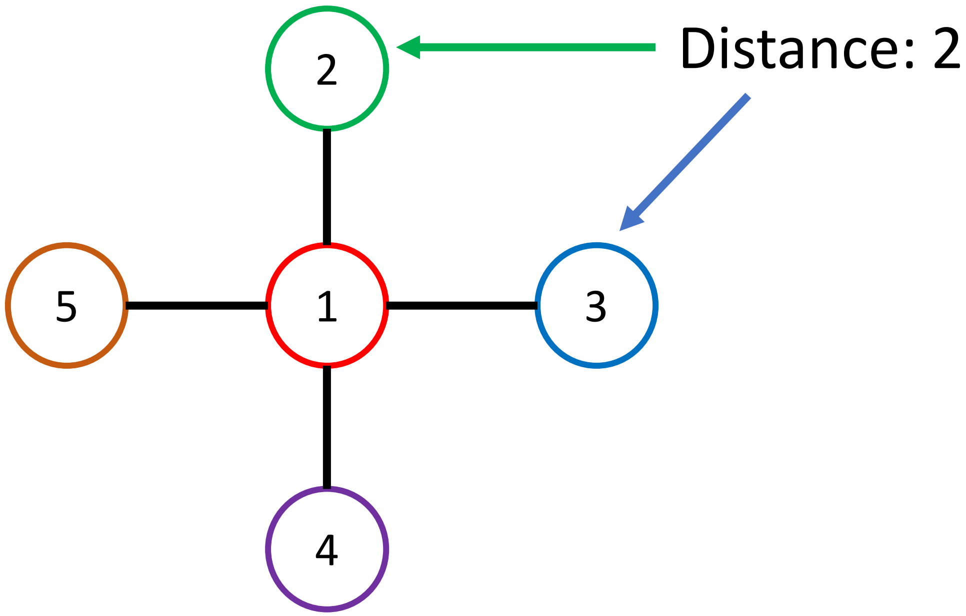
Encode relationships into a **graph**

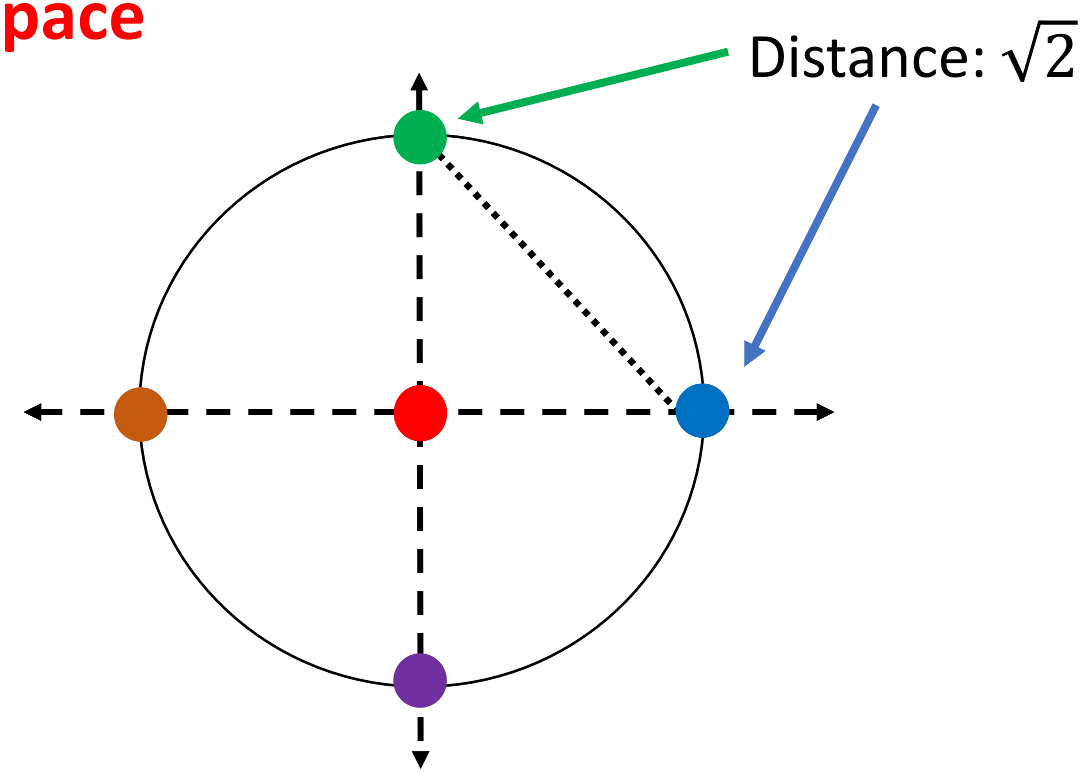
- **Hierarchical relationships:** trees
  - Ex: Artists -> Albums -> Songs
- Embed into **Euclidean space**?
  - Results in **distortion**

→ Non-Euclidean Embeddings!

Choice of Embedding Space Matters!

- **Q:** Do trees embed well in **Euclidean space**

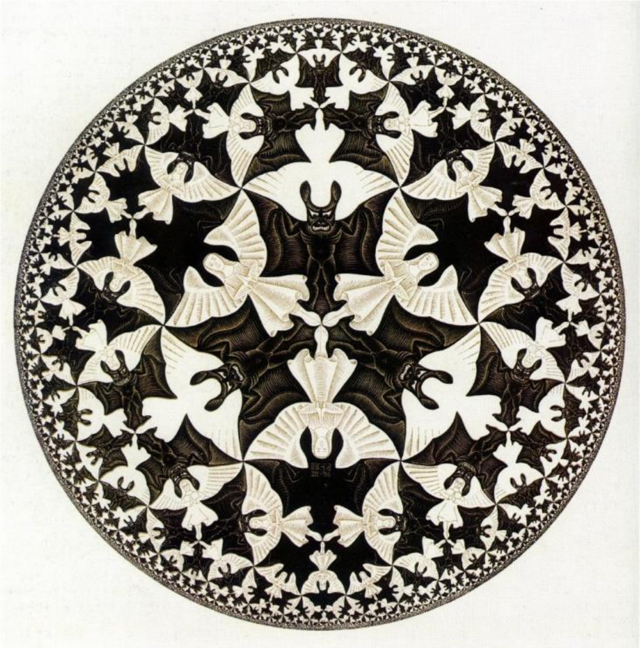


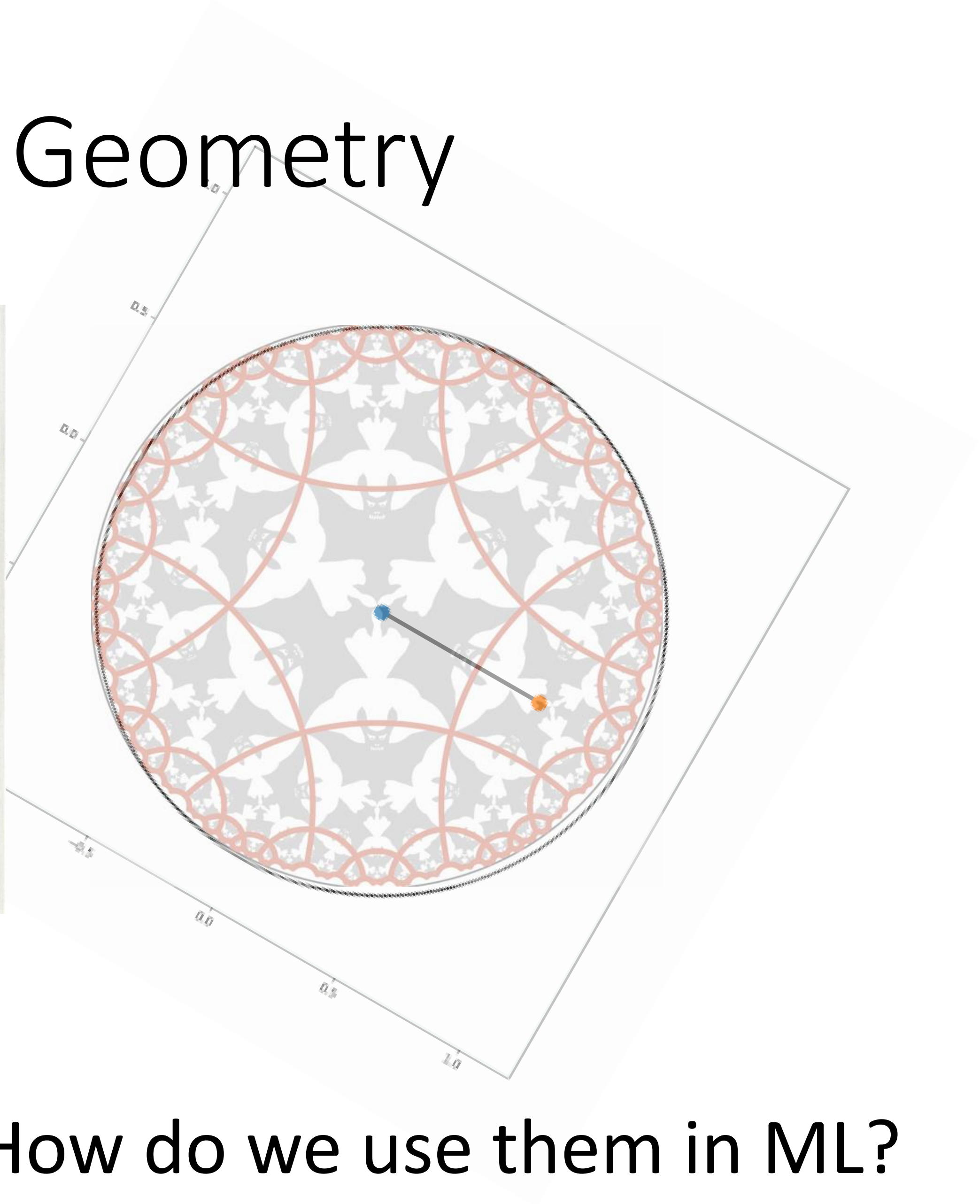


Distortion! $= \dfrac{\text{Biggest Stretch}}{\text{Worst Compression}}$

Euclidean space distorts hierarchical relationships.

Hyperbolic Geometry





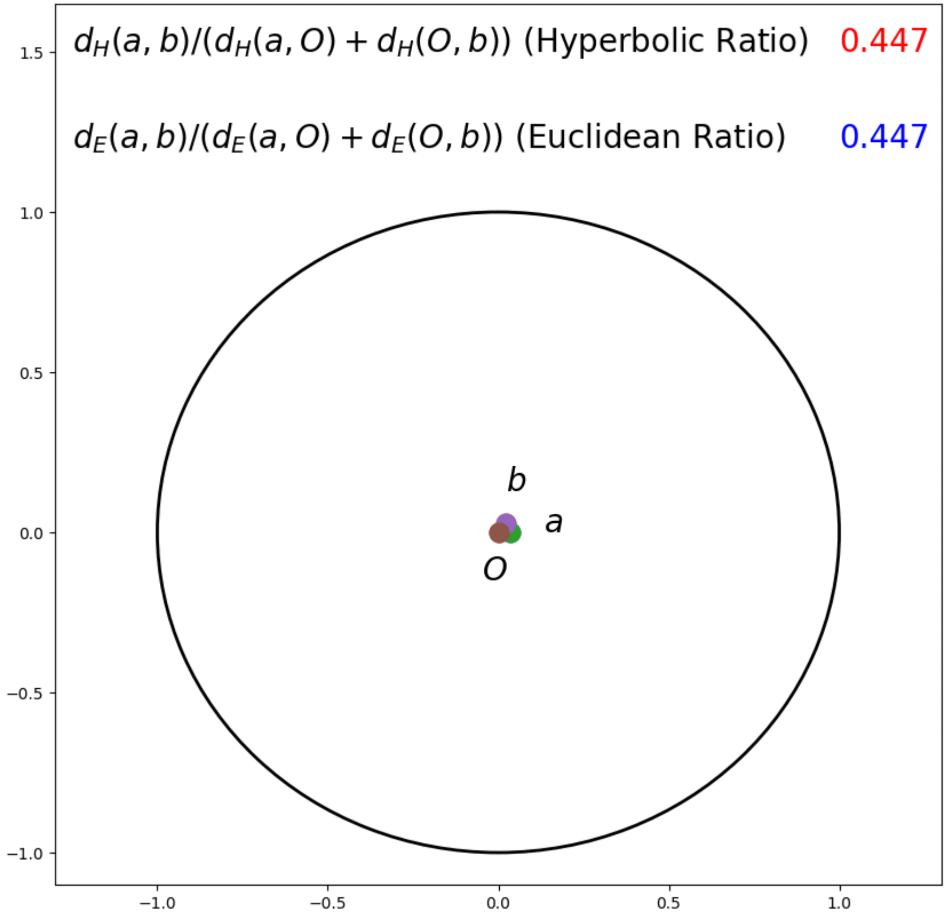
What are these spaces? How do we use them in ML?

Models, Distances, and Trees

- Poincaré **model** of hyperbolic space
- Connection to **tree distance**:
  - Hyperbolic distance:

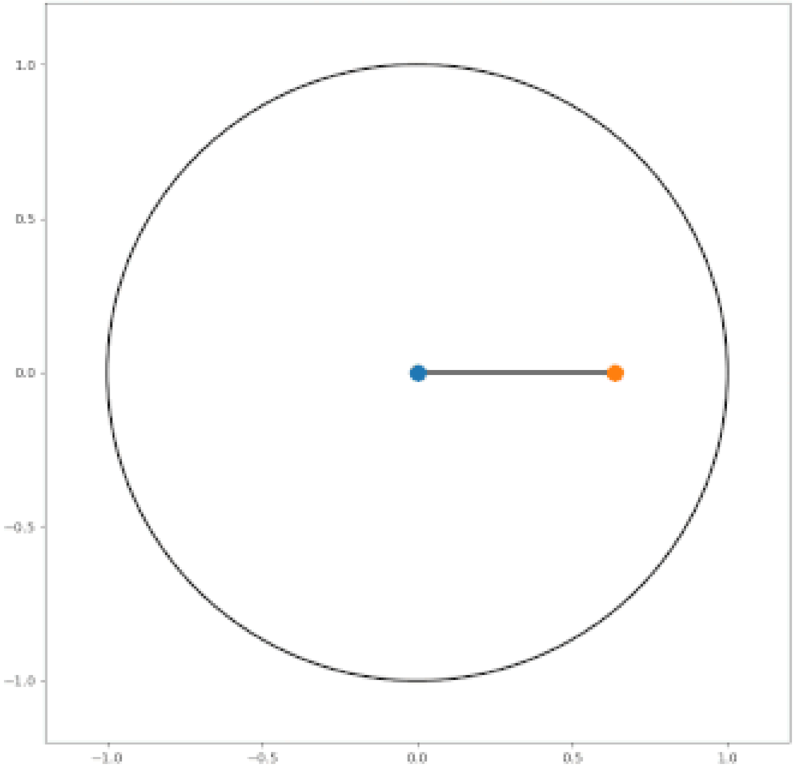
$$d_H(x,y) = \operatorname{acosh}\left(1 + 2\frac{\|x-y\|^2}{(1-\|x\|^2)(1-\|y\|^2)}\right)$$

- Hyperbolics naturally represent trees!



Embedding Trees: New Constructions

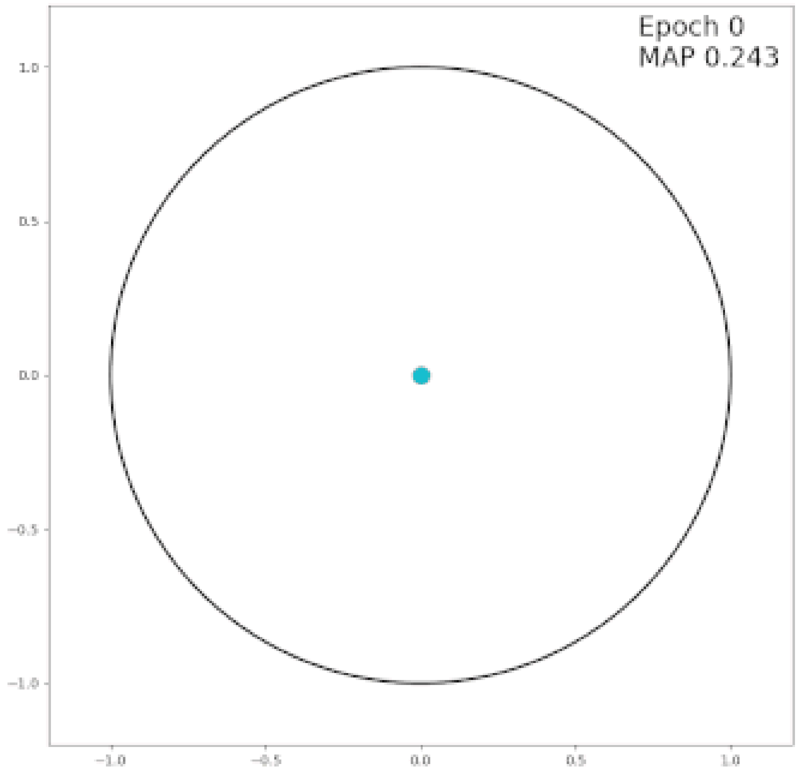
- Powerful tool for **embedding trees**
- **Arbitrarily low distortion!**
- At each node: place children into disjoint **subcones**
- Single global scaling factor.

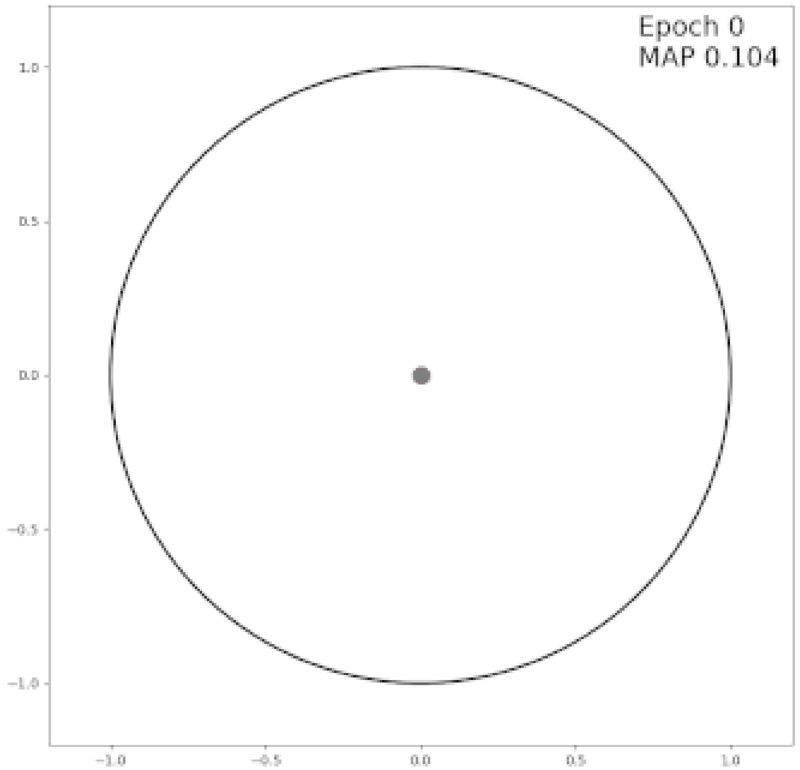


Non-Euclidean embeddings: **scales matter!**

Optimization Model

- Examples: 20 node **cycle** and ternary **tree**





Loss:

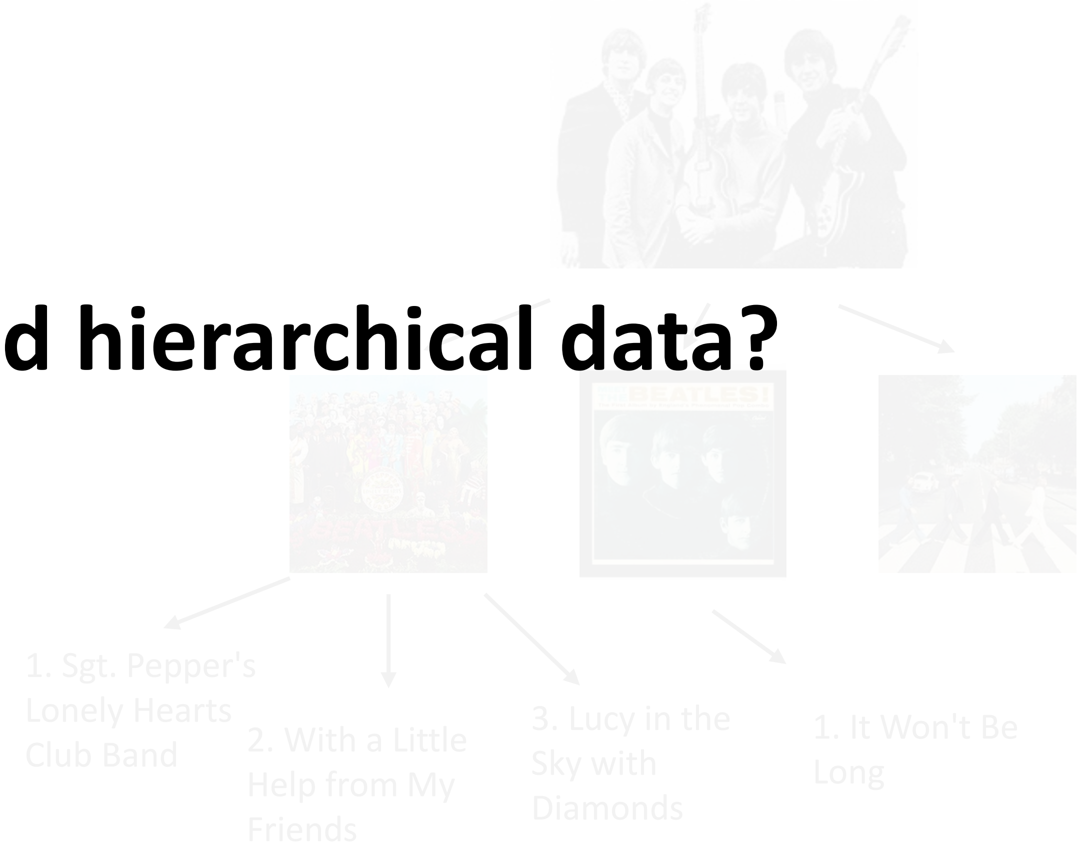
$$\sum_{1 \le i < j \le n} \left| \left(\frac{d_{\mathcal{H}}(p_i, p_j)}{d_G(x_i, x_j)} \right)^2 - 1 \right|$$

Optimizer:
Riemannian
SGD

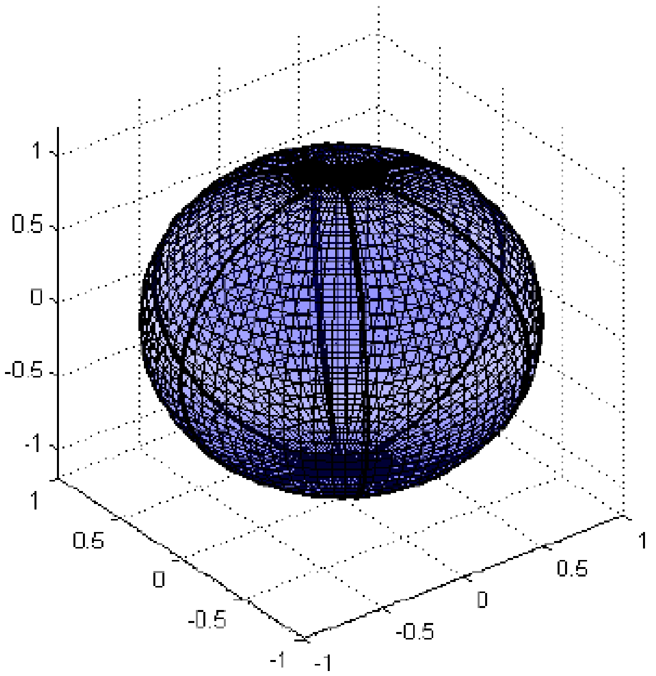
- **Hierarchical relationships:** trees
  - Ex: Artists -> Albums -> Songs

How do we go beyond hierarchical data?

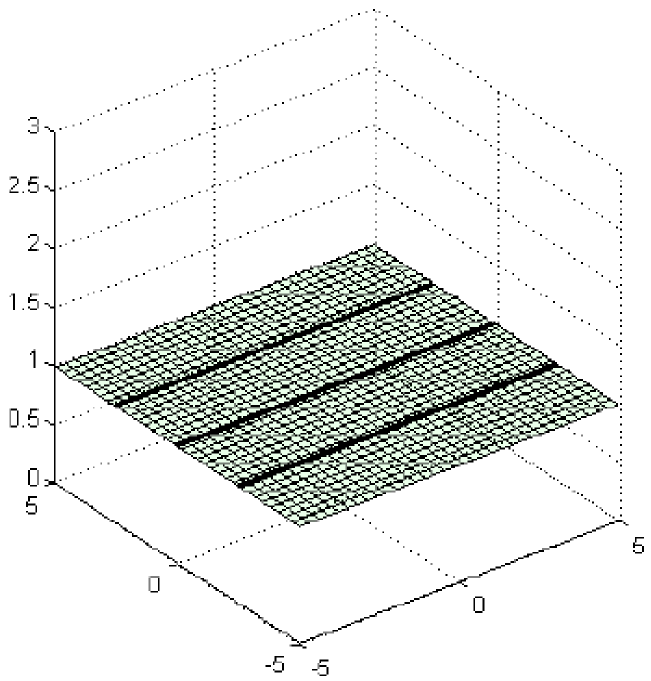
- Embed into **hyperbolic space**!
  - Low **distortion**
  - **+ Guarantees**



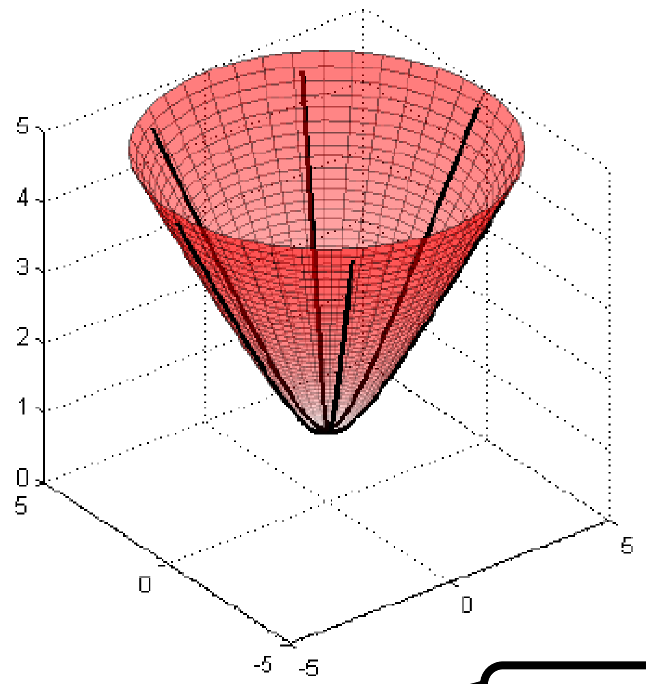
Model Spaces



K = +1



K = 0



K = -1

curvature

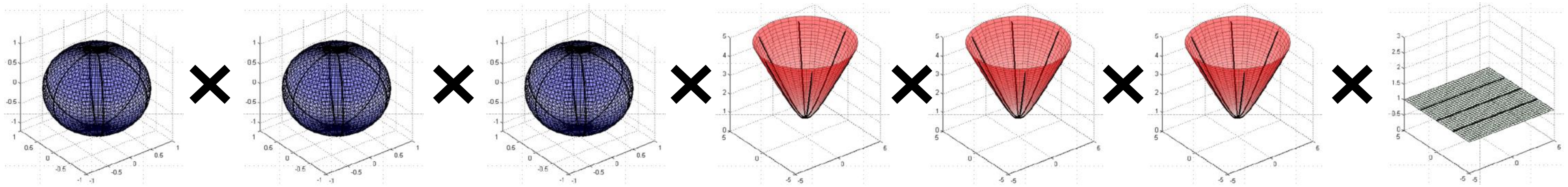
Spherical **Euclidean** **Hyperbolic**

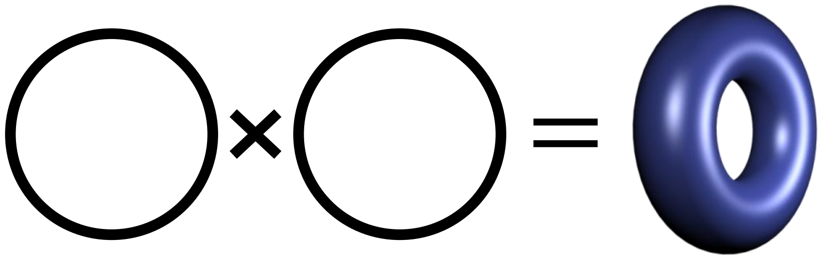
Problem: How do we **combine these**?

Simple Answer: Take Products

Product manifold

$$\mathcal{P} = \mathbb{S}^{s_1} \times \mathbb{S}^{s_2} \times \cdots \times \mathbb{S}^{s_m} \times \mathbb{H}^{h_1} \times \mathbb{H}^{h_2} \times \cdots \times \mathbb{H}^{h_n} \times \mathbb{E}^{e},$$

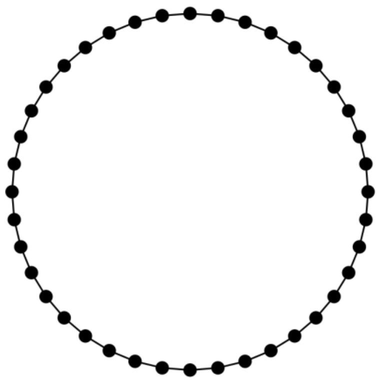


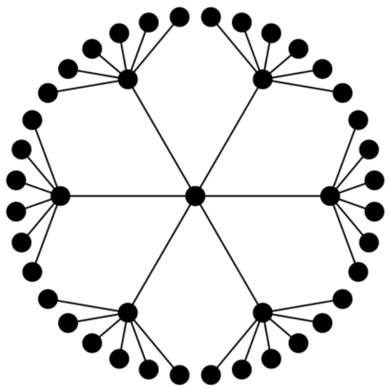


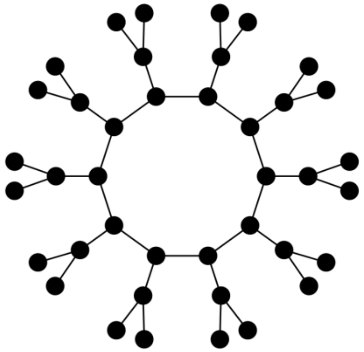
- Distances decompose:
- Easy optimization

$$d^2_{\mathcal{P}}(x, y) = \sum_{i=1}^{k} d_i^2(x_i, y_i)$$

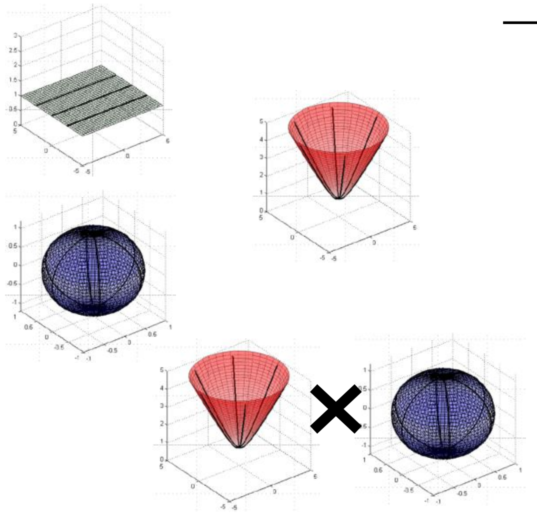
Results



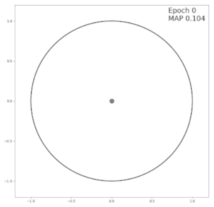


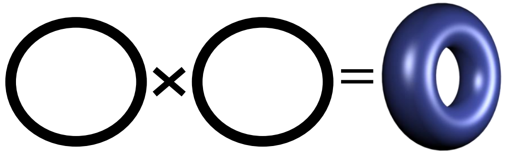


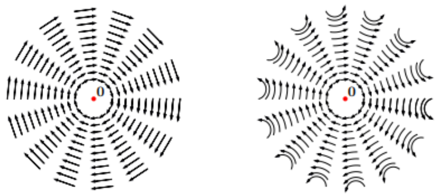
	Cycle (|V| = 40, |E| = 40)	**Tree** (|V| = 40, |E| = 39)	**Ring of Trees** (|V| = 40, |E| = 40)
$(\mathbf{E}^3)^1$	0.106	0.148	0.099
$(\mathbf{H}^3)^1$	0.164	**0.032**	0.080
$(\mathbf{S}^3)^1$	**0.001**	0.161	0.111
$(\mathbf{H}^3)^1 \times (\mathbf{S}^3)^1$	0.111	0.054	**0.062**



Some of Our Work

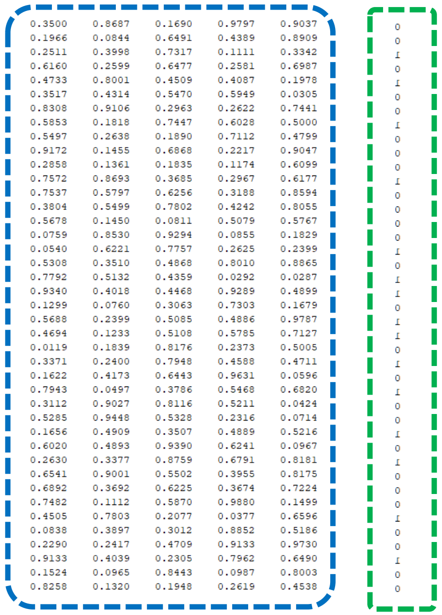


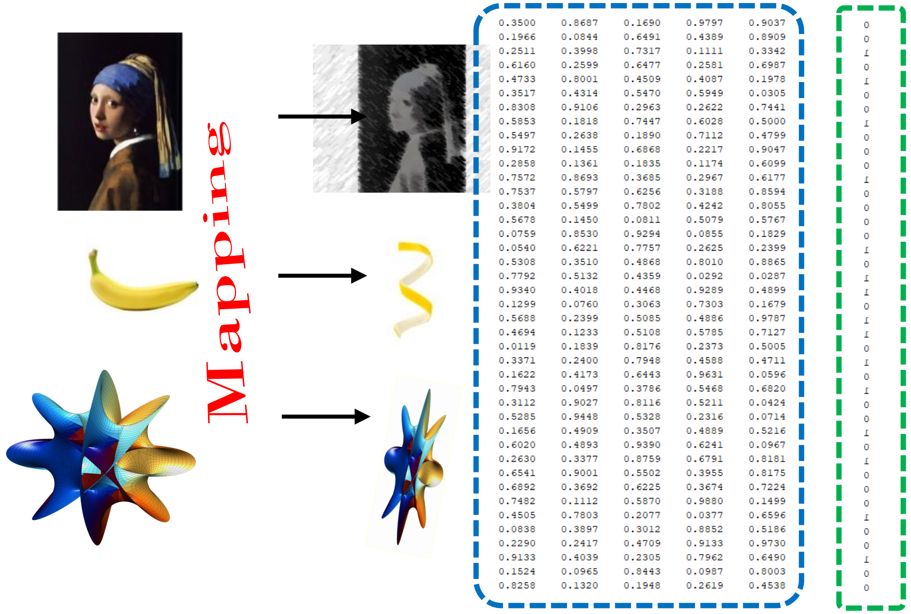




- Representation tradeoffs for hyperbolic embeddings, RHD**S**PR, **ICML'18**
- Learning mixed-curvature representations in product spaces, G**S**GR, **ICLR '19**
- Hyperbolic graph convolutional neural networks, CYRL, **NeurIPS '19**
- Low-dimensional knowledge graph embeddings via hyperbolic rotations, CW**S**R, **NeurIPS GRL '19**
- Low-Dimensional Hyperbolic Knowledge Graph Embeddings", CWJ**S**RR, **ACL '20**

Data Bottlenecks





Bottleneck 1: Getting Labels

Bottleneck 2: Distortion

Thank you!

Joint Work With:

Nicholas Roberts, Changho Shin, Winfred Li, Harit Vishwakarma, Dyah Adila, Aws Albarghouthi, Ben Boecking, Chris Ré, Chris De Sa, Alex Ratner, Albert Gu, Paroma Varma, Jared Dunnmon, Ines Chami, Beliz Gunel, Dan Fu, Mayee Chen

https://pages.cs.wisc.edu/~fredsala/

fredsala@cs.wisc.edu