Minimum-Distortion Embedding

Akshay Agrawal
Stanford University

July 19 2021

Research at Stanford

- A. Agrawal, R. Verschueren, S. Diamond, S.Boyd. *A rewriting system for convex optimization problems.* Journal of Control and Decision, 2018.
- A. Agrawal, et al. *TensorFlow Eager: A multi-stage, Python-embedded DSL for machine learning.* SysML, 2019.
- A. Agrawal, S. Diamond, S.Boyd. *Disciplined geometric programming.* Optimization Letters, 2019.
- A. Agrawal, S.Boyd. *Disciplined quasiconvex programming.* Optimization Letters, 2020.
- A. Agrawal, S. Barratt, S. Boyd, S.Diamond, J. Z. Kolter. *Differentiating through a cone program.* Journal of Applied and Numerical Optimization, 2019.
- A. Agrawal, B. Amos, S. Barratt, S. Boyd, S.Diamond, J. Z. Kolter. *Differentiable convex optimization layers.* NeurIPS, 2020.
- A. Agrawal, S. Boyd. *Differentiating through log-log convex programs.* Pre-print, 2020.
- A. Agrawal, S. Barratt, S. Boyd, B. Stellato. *Learning convex optimization control policies.* Learning for Dynamics and Control, 2020.
- A. Agrawal, S. Barratt, S. Boyd. *Learning convex optimization models.* IEEE Journal of Automatica Sinica, 2020.
- A. Agrawal, S. Boyd, D. Narayanan, F. Kazhamiaka, M. Zaharia. *Allocation of fungible resources via a fast, scalable price discovery method.* Pre-print, 2021.
- A. Agrawal, J. Zhang, S. Boyd. *Doubly projected gradient methods.* Draft.

This talk

A. Agrawal, A. Ali, S. Boyd. *Minimum-Distortion Embedding*. Foundations and Trends in Machine Learning, 2021.

contributions:

- a framework unifying over a century's worth of work on embedding
- a novel algorithm for approximately solving embedding problems (implemented in PyMDE)

Outline

Outline

Embedding

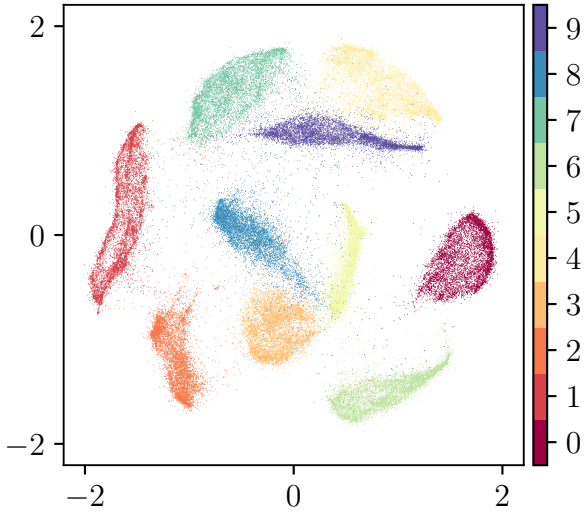
- representation of abstract items $(1, \ldots, n)$ by vectors $(x_1, \ldots, x_n \in \mathbf{R}^m)$
- used for data visualization or insight, downstream computational tasks
- should be faithful to original data in some way
  - if two items are similar, vectors should be near each other
  - if two items are dissimilar, vectors should not be near each other

  - similarity is a property of pairs of *items*, and is application dependent
  - nearness is a property of pairs of *vectors*, measured in Euclidean distance

MNIST

- handwritten digits
- 70k images
- 1.5M pairs of similar/dissimilar images
- embed into 2 dimensions

MNIST

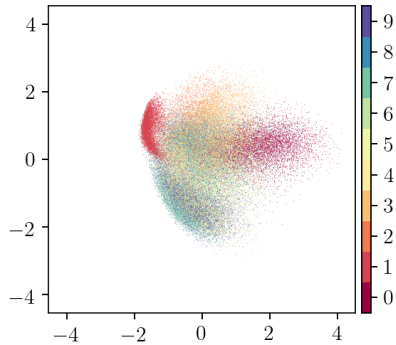


Applications

- interactive data exploration
  - counties
  - co-authorship networks
  - citation networks
  - genomes
  - single-cell mRNA transcriptomes
  - cardiac muscle movement
  - rocks and minerals
  - stock exchange orders
  - words
  - news documents
  - (PyMDE has been used for all the above)
- dimensionality reduction
- feature generation

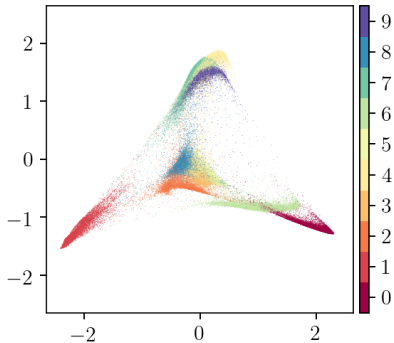
History: 1900-1960s

- early research in psychology
  - principal component analysis [Pea01]
  - multi-dimensional scaling [Ric38]
- tractable (reduce to eigenproblems)



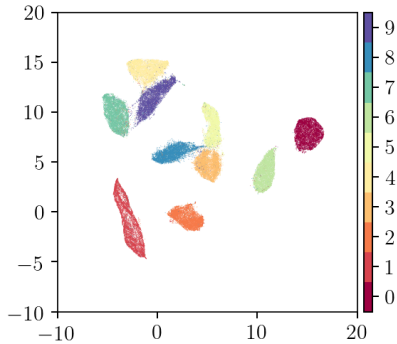
History: 1990s-2000

- dimensionality reduction methods
  - lsomap
  - locally-linear embedding
  - maximum variance unfolding
  - Laplacian embedding
- tractable (reduce to eigenproblems)



History: 2000-present

- emphasis on "solving" nonconvex problems
  - t-SNE
  - LargeVis
  - UMAP
  - neural networks
- not tractable (rely on heuristics)



This talk

- a general framework for faithful embedding
  - exactly reproduces many historical examples
  - approximately reproduces others
  - generates new kinds of embeddings
- an efficient heuristic optimization algorithm

Outline

Embedding

- an embedding of items $\mathcal{V} = \{1, \ldots, n\}$ items is a matrix

$$X \in \mathbf{R}^{n \times m}$$

- rows $x_1^T, \ldots, x_n^T$ are the embedding vectors
- m columns are features

Distortion

- quality of an embedding is a function of the distances

$$d_{ij} = \|x_i - x_j\|_2, \quad (i,j) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \quad (i < j)$$

- each pair $(i,j) \in \mathcal{E}$ has an associated distortion function $f_{ij} : \mathbf{R}_+ \to \mathbf{R}$
- distortion of the embedding for $(i,j) \in \mathcal{E}$ is

$$f_{ij}(d_{ij})$$

- examples

$$f_{ij}(d_{ij}) = w_{ij} d_{ij}^2 \qquad (w_{ij} \text{ a weight or similarity score})$$
$$f_{ij}(d_{ij}) = (d_{ij} - \delta_{ij})^2 \qquad (\delta_{ij} \text{ a deviation or original distance})$$

Minimum-distortion embedding (MDE) problem

$$\begin{array}{ll} \text{minimize} & E(X) \\ \text{subject to} & X \in \mathcal{X} \end{array}$$

- $X \in \mathbf{R}^{n \times m}$ is the variable
- $E(X)$ is the average distortion

$$E(X) = \frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} f_{ij}(d_{ij})$$

- $\mathcal{X} \subseteq \mathbf{R}^{n \times m}$ is the set of allowable embeddings

Solving MDE problems

- intractable, except in a few special cases that we will see later
- we develop an algorithm that approximately solves MDE problems
  - reliably solves tractable problems
  - reliably finds good embeddings for others

Distortion functions from weights

item pairs $(i, j) \in \mathcal{E}$ have associated weights $w_{ij} \in \mathbf{R}$

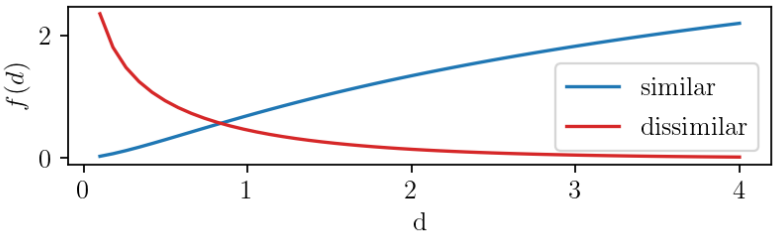
- $w_{ij} > 0$ means i and j are similar
- $w_{ij} < 0$ means i and j are dissimilar
- magnitude of weight conveys degree of (dis)similarity

simplest example is the quadratic:

$$f_{ij}(d_{ij}) = w_{ij} d_{ij}^2$$

Example

- $w_{ij} > 0$, $f_{ij}(d_{ij}) = w_{ij} \log(1 + d_{ij}^2)$
- $w_{ij} < 0$, $f_{ij}(d_{ij}) = w_{ij} \log(1 - \exp(-d_{ij}))$



- captures basic idea of faithfulness:
  - vectors for similar items should be near each other
  - vectors for dissimilar items should not be near each other

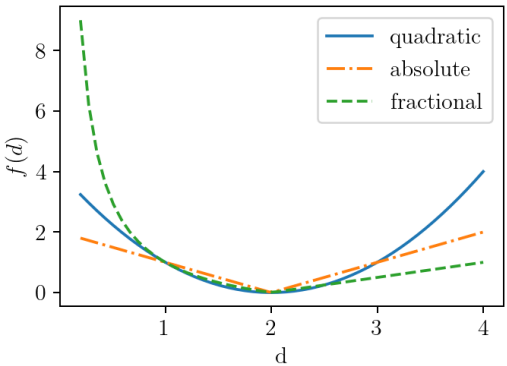
Distortion functions from deviations

item pairs $(i, j) \in \mathcal{E}$ have associated deviations $\delta_{ij} \in \mathbf{R}_+$

- distortion function minimized when $d_{ij} = \delta_{ij}$

Examples

- quadratic: $f_{ij}(d_{ij}) = (d_{ij} - \delta_{ij})^2$
- absolute: $f_{ij}(d_{ij}) = |d_{ij} - \delta_{ij}|$
- fractional: $\max\{\delta/d, d/\delta\} - 1$



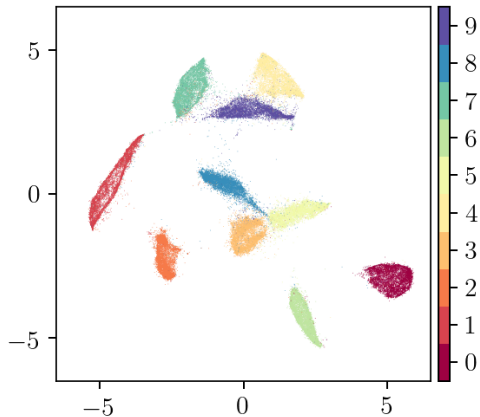
Pre-processing

- often useful to preprocess raw similarity data
- somewhat of an art, like feature engineering
- assume we have some original deviations
- to emphasize local structure
  - similar pairs from k-nearest neighbors of each item
  - distortion functions from weights
  - $w_{ij} = +1$ for neighbors, optionally $w_{ij} = -1$ for non-neighbors
- to emphasize global structure
  - compute shortest path distances
  - distortion functions that preserve them

Centering constraint

$$\mathcal{C} = \{X \mid X^T \mathbf{1} = 0\}$$

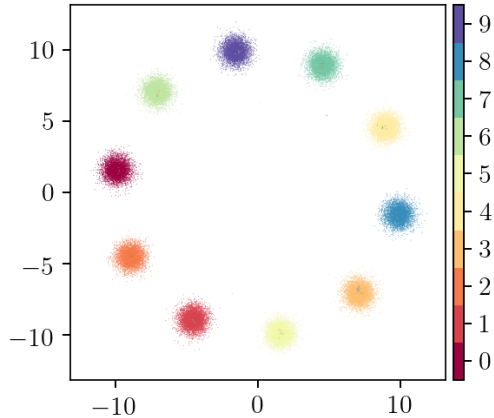
- centers the embedding vectors around the origin
- without loss of generality (distances not affected by translation)



Anchor constraint

$$\mathcal{A} = \{X \mid x_i = x_i^{\text{given}},\ i \in \mathcal{K}\}$$

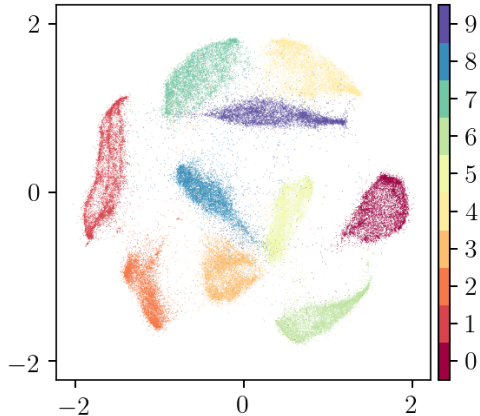
- pins embedding vectors for anchored items ($\mathcal{K}$)
- useful for incremental embedding, placement problems



Standardization constraint

$$\mathcal{S} = \{X \mid \frac{1}{n} X^T X = I,\ X^T \mathbf{1} = 0\}$$

- feature columns uncorrelated with RMS value 1
- forces vectors to spread out, but not too much



Quadratic MDE problem

- distortion functions
$$f_{ij}(d_{ij}) = w_{ij} d_{ij}^2$$
and standardization constraint $\mathcal{S} = \{X \mid \frac{1}{n} X^T X = I,\ X^T \mathbf{1} = 0\}$
- analytical solution via eigenvectors of a certain matrix
- many historical methods are special cases
  - PCA (and kernel PCA)
  - Laplacian eigenmap
  - Isomap
  - locally linear embedding
  - classical MDS
  - maximum variance unfolding

Outline

PCA

- PCA [Pea01] starts with data matrix $Y \in \mathbf{R}^{n \times p}$, with rows $y_1^T, y_2^T, \ldots, y_n^T$
- embedding: top m eigenvectors of YY^T
- equivalent to solving a quadratic MDE problem with

$$w_{ij} = y_i^T y_j, \quad \mathcal{E} = \{(i,j) \mid 1 \leq i < j \leq n\}$$

interpretation

- i and j are similar (dissimilar) if angle between y_i and y_j is acute (obtuse)
- neutral to (i,j) when y_i, y_j are orthogonal

Laplacian eigenmap

- Laplacian eigenmap [BN02] starts with data matrix $Y \in \mathbf{R}^{n \times p}$
- $(i, j) \in \mathcal{E}$ if y_i is among k-nearest neighbors of y_j
- embedding: m bottom eigenvectors of graph Laplacian (excluding $\mathbf{1}$)
- equivalent to solving a quadratic MDE problem with $w_{ij} = +1$

UMAP

- UMAP [MHM18] is a widely used dimensionality reduction method
- distortion functions from weights, with

$$f_{ij}(d_{ij}) = w_{ij} \log(1 + \alpha d^{\beta}), \quad w_{ij} > 0$$

and

$$f_{ij}(d_{ij}) = w_{ij} \log\left(\frac{d^{\gamma}}{1 + d^{\gamma}}\right) \quad w_{ij} < 0$$

(α, β, γ are hyper-parameters); unconstrained

- $\mathcal{E}$ is a union of a k-nearest neighbor graph (edges have positive weights) and random sample of its complement (edges have negative weights)

Outline

Stationarity condition

- tangent cone to $\mathcal{X}$ at X is the set

$$\mathcal{T}_X(\mathcal{X}) = \{V \in \mathbf{R}^{n\times m} \mid \mathbf{dist}(X + hV, \mathcal{X}) = o(h),\ h \to 0\}$$

  - $V \in \mathcal{T}_X(\mathcal{X})$ is called a *tangent* to $\mathcal{X}$ at X
- a direction is a feasible descent direction if

$$V \in \mathcal{T}_X(\mathcal{X}), \quad \mathbf{tr}(\nabla E(X)^T V) < 0$$

- X is stationary if the cone of feasible descent directions is empty

Stationarity condition

- projected gradient at X with constraint $\mathcal{X}$ is

$$G = \Pi_{\mathcal{T}_X}(\nabla E(X))$$

- $-G$ is the steepest feasible descent direction, *i.e.*, the solution to

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(\nabla E(X)^T V) + \frac{1}{2}\|V\|_F^2 \\ \text{subject to} & V \in \mathcal{T}_X(\mathcal{X}) \end{array}$$

- stationarity condition can be written as

$$G = 0$$

Projections

algorithm requires two projections related to the constraint set $\mathcal{X}$

- $\Pi_{\mathcal{T}_X}$, projection onto tangent cone
- $\Pi_{\mathcal{X}}$, projection onto constraints
- both are analytical (and cheap) for $\mathcal{C}$, $\mathcal{A}$, and $\mathcal{S}$

example, for standardization constraint $\mathcal{S}$:

$$\Pi_{\mathcal{T}_X}(Z) = Z - (1/n)XZ^TX, \qquad \Pi_{\mathcal{S}}(Z) = \sqrt{n}UV^T, \quad Z = U\Sigma V^T$$

takes $O(nm^2)$ time (usually $m \ll n$)

Traditional projected gradient method

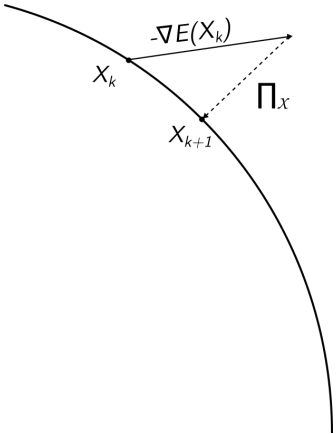
- simplest algorithm is the traditional projected gradient method
- works, but extremely slow

---

for $k = 0, \ldots$

1. *Projected gradient.* Compute gradient $\nabla E(X_k)$
2. *Line search.* Choose step length t_k
3. *Update.* $X_{k+1} := \Pi_{\mathcal{X}}(X_k - t_k \nabla E(X_k))$

Traditional projected gradient method



(Doubly) projected gradient method

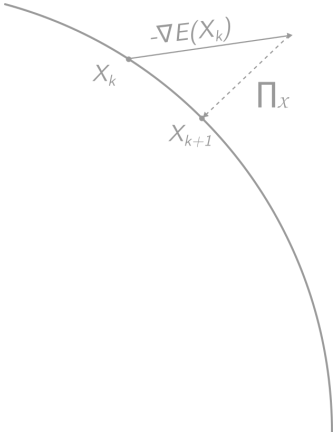
- more sophisticated is a doubly projected gradient method
- works, but slow

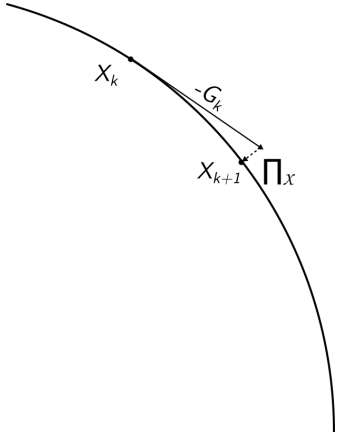
---

for $k = 0, \ldots$

1. *Projected gradient.* Compute projected gradient $G_k = \Pi_{\mathcal{T}_X}(\nabla E(X_k))$
2. *Line search.* Choose step length t_k
3. *Update.* $X_{k+1} := \Pi_{\mathcal{X}}(X_k - t_k G_k)$

(Doubly) projected gradient method





Projected L-BFGS method

- extension of L-BFGS to handle constraints
- gradients replaced with projected gradients
- appears to be new, though just combines some old ideas
- works, and extremely fast

---

for $k = 0, \ldots$

1. *Projected gradient.* Compute projected gradient $G_k = \Pi_{\mathcal{T}_{X_k}}(\nabla E(X_k))$
2. *Search direction.* Compute L-BFGS search direction V_k, using G_k
3. *Line search.* Find step length t_k
4. *Update.* $X_{k+1} := \Pi_{\mathcal{X}}(X_k + t_k V_k)$

Search direction

in iteration k, search direction $V_k \in \mathbf{R}^{n\times m}$ chosen as

$$\mathbf{vec}\, V_k = -B_k^{-1}\, \mathbf{vec}\, G_k,$$

where $B_k \in \mathbf{S}^{n\times m}_{++}$ is given by the recursion

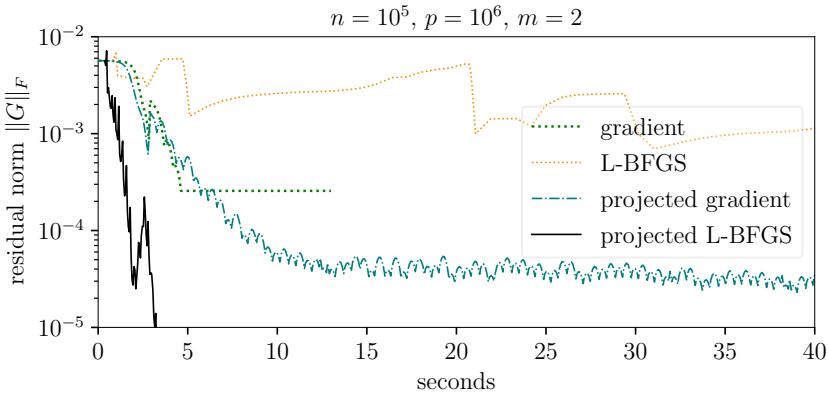
$$B_{j+1}^{-1} = \left(I - \frac{s_j y_j^T}{y_j^T s}\right) B_j^{-1} \left(I - \frac{y_j s_j^T}{y_j^T s_j}\right) + \frac{s_j s_j^T}{y_j^T s_j}, \quad j = k-1, \ldots, k-M,$$

using $B_{k-M} = \gamma_k I$ (M is the memory size), and

$$s_j = \mathbf{vec}(X_{j+1} - X_j), \qquad y_j = \mathbf{vec}(G_{j+1} - G_j), \qquad \gamma_k = \frac{y_{k-1}^T s_{k-1}}{y_{k-1}^T y_{k-1}}$$

Experiments

- quadratic MDE problem, $w_{ij} = +1$, edges chosen uniformly at random
- solved on CPU (quad core, 4GHz)



Experiments

Problem dimensions			Embedding time (s)		Objective values	
n	$\lvert\mathcal{E}\rvert$	m	CPU	GPU	$E(X_K)$	$E(X^\star)$
10^3	10^4	2	0.1	0.4	1.432	1.432
10^3	10^4	10	0.2	0.4	7.797	7.795
10^3	10^4	100	0.8	1.5	104.5	104.5
10^4	10^5	2	0.4	0.4	1.007	1.007
10^4	10^5	10	0.7	0.4	6.066	6.065
10^4	10^5	100	20.8	2.9	81.12	81.08
10^5	10^6	2	2.5	0.6	0.935	0.931
10^5	10^6	10	10.5	1.2	5.152	5.137
10^5	**10^6**	**100**	**334.7**	**15.8**	**63.61**	**63.51**

Software

solution method implemented in PyMDE

- provides library of distortion functions
- extensible
- scales to many millions of items and pairs

code: https://github.com/cvxgrp/pymde

docs: https://pymde.org

Outline

US counties

data:

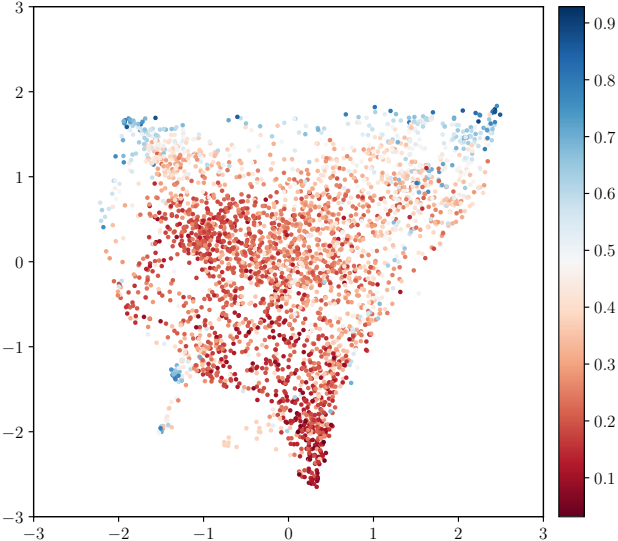
- 3,220 US counties
- represented by 34 features, from 2013-17 ACS 5-Year Estimates
  - demographics, income, employment, commute, ...

embedding:

- 73k pairs of similar and dissimilar counties
- distortion functions from weights
- embed into $\mathbf{R}^2$
- color by fraction that voted democratic in 2016 presidential election (voting data not used to make embedding)

Embedding (US counties)

- ↗: NY, CA, PA ...
- ↖: GA, MI, AL ...
- ↙: TX, NM, AZ ...



Co-authorship network

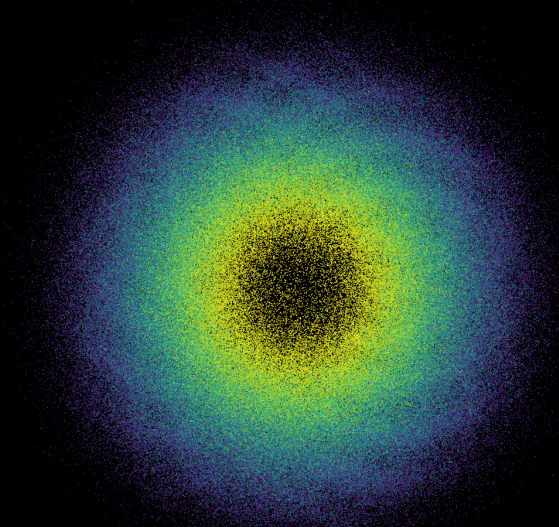
data:

- 590k authors scraped from Google Scholar
  - 16k authors with labeled academic discpline (bio, physics, EE, CS, AI)

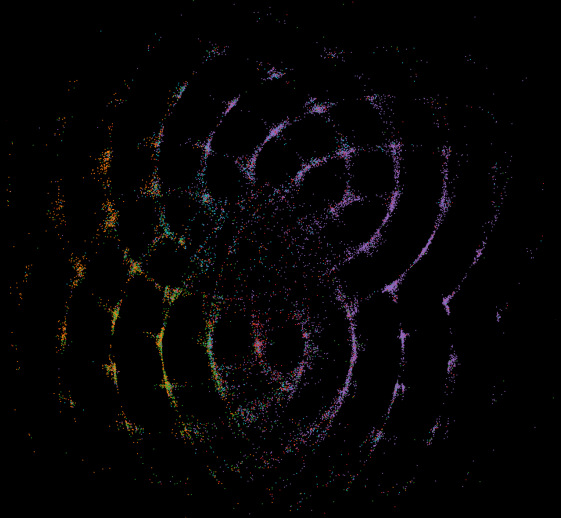
embedding:

- $\approx$ 88M pairs of authors
- distortion functions to preserve graph distance (absolute loss)
- embed into $\mathbf{R}^2$

Full network embedding (co-authorship network)



Academic disciplines (co-authorship network)



Summary

- MDE generalizes well-known embedding methods and leads to new ones
- heuristic algorithm scales to millions of items and pairs

Acknowledgements

- PhD advisor Stephen Boyd
- undergrad advisor Mehran Sahami
- undergrad research advisor Andreas Paepcke
- defense committee, Sanjay Lall, Mert Pilanci, Ashok Srivastava
- beta readers and testers of MDE, esp. Dmitry Kobak and Lawrence Saul
- lab, esp. Alnur Ali, Guillermo Angeris, Shane Barratt, Steven Diamond, Jonathan Tuck, & Junzi Zhang
- friends, too many to name
- family, Sarika, Ajay, and Akansha Agrawal
- Delenn Chin

References

[BN02] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*. 2002, pp. 585–591.

[MHM18] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv* (2018).

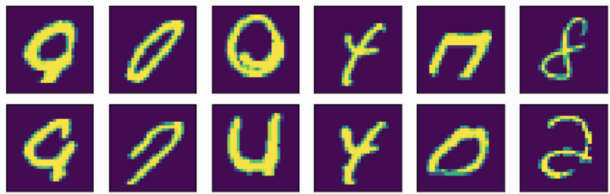
[Pea01] K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572.

[Ric38] M. Richardson. Multidimensional psychophysics. *Psychological Bulletin* 35 (1938), pp. 659–660.

[Wil+20] A. Wilk, A. Rustagi, N. Zhao, J. Roque, G. Martínez-Colón, J. McKechnie, G. Ivison, T. Ranganath, R. Vergara, T. Hollis, et al. A single-cell atlas of the peripheral immune response in patients with severe COVID-19. *Nature Medicine* (2020), pp. 1–7.

Sanity checking

- when m is 2 or 3, color by held-out attributes
- check sensitivity by removing some distortion functions & re-embedding
- manually inspect pairs with high distortion; *e.g.*, for MNIST:



- ultimately, validation depends on downstream application

Standardization constraint

- for $X \in \mathcal{S}$,

$$\sum_{1 \le i < j \le n} d_{ij}^2 = n^2 m$$

- the RMS value of the d_{ij},

$$d_{\text{nat}} = \sqrt{\frac{2nm}{n-1}},$$

  can be interpreted as the natural or typical embedding distance

Quadratic MDE problem

- equivalent to

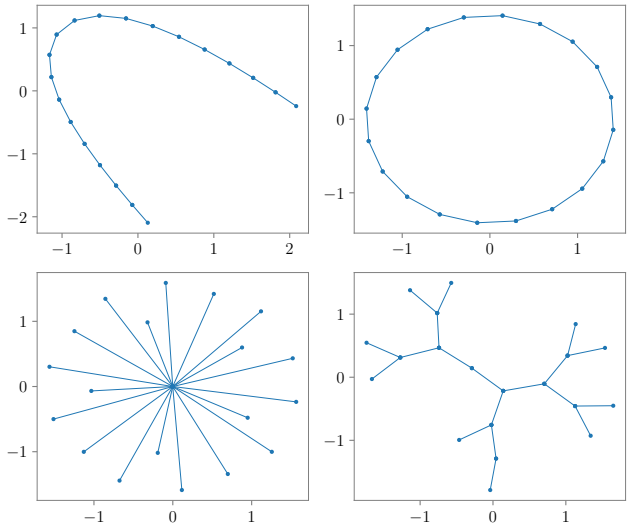
$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(X^T L X) \\ \text{subject to} & X \in \mathcal{S} \end{array}$$

  where $L \in \mathbf{S}^n$ has upper triangular entries

$$L_{ij} = \begin{cases} -w_{ij} & (i,j) \in \mathcal{E} \\ 0 & \text{otherwise,} \end{cases}$$

  and diagonal entries $L_{ii} = -\sum_{j \neq i} L_{ij}$
- solution stacks m bottom eigenvectors of L, excluding $\mathbf{1}$

Graph layout



Gradient

- index the distortion functions (and distances) in some fixed order

$$f_1, f_2, \ldots, f_p, \quad p = |\mathcal{E}|$$

- the gradient of the average distortion is

$$\nabla E(X) = (1/p)ACA^T X, \quad C = \mathbf{diag}(f_1'(d_1)/d_1, \ldots, f_p'(d_p)/d_p),$$

  where A is the incidence matrix of $\mathcal{E}$
- assumption throughout: E is differentiable
  - possibly nondifferentiable when embedding vectors are nondistinct
  - nondifferentiability does not matter in practice

Modified Wolfe conditions

find step length t_k satisfying

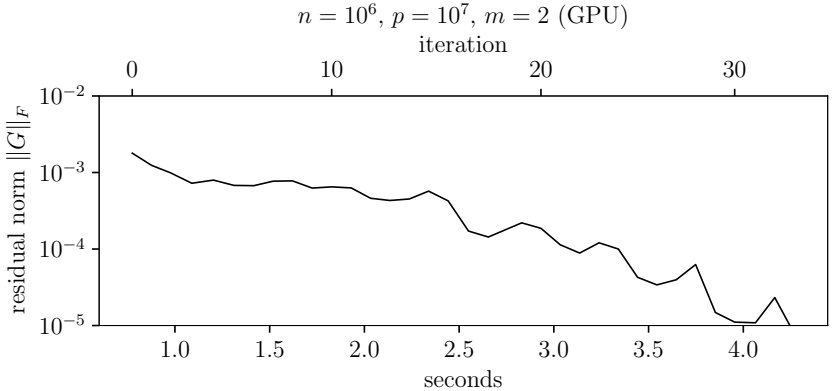
$$E(\Pi_{\mathcal{X}}(X_k + t_k V_k)) \leq E(X_k) + c_1 t_k \operatorname{\mathbf{tr}}(G_k^T V_k),$$
$$|\operatorname{\mathbf{tr}}(G_{k+1}^T V_k))| \leq c_2 |\operatorname{\mathbf{tr}}(G_k^T V_k)|,$$

where $0 < c_1 < c_2 < 1$ are constants (typically 10^{-4} and 0.9).

- first inequality is a modified sufficient decrease condition
- second inequality is a modified curvature condition

Experiments (GPU)

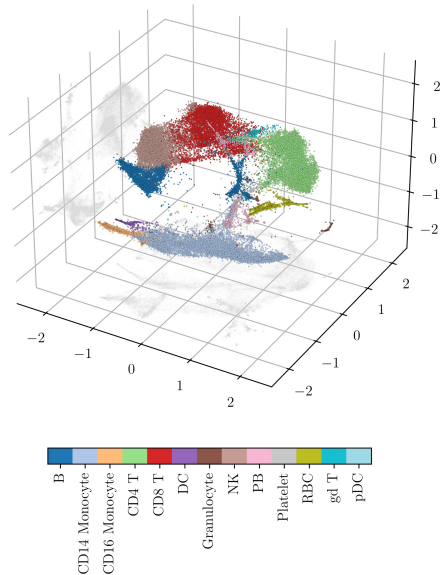
- quadratic MDE problem, w_{ij} $+1$, edges chosen uniformly at random
- solved on GPU (NVIDIA GeForce GTX 1070)



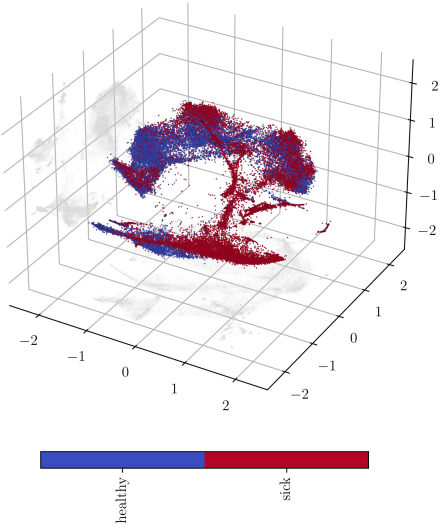
Single-cell genomics

- single-cell mRNA transcriptomes
- 7 patients with COVID-19, 6 healthy controls [Wil+20]
- 44k cells (1M pairs)
- embed into 3 dimensions
- 27s CPU, 6s GPU

Single-cell genomics



Single-cell genomics



MNIST

- handwritten digits
- 70k grayscale images
  - 28-by-28 pixels, vectors in $\mathbf{R}^{784}$

Pre-processing (MNIST)

- want pairs of similar and dissimilar items, $\mathcal{E}_{\text{sim}}$ and $\mathcal{E}_{\text{dis}}$
- Euclidean distance is a poor global metric on images, but a good local one
- take $\mathcal{E}_{\text{sim}}$ to be 15 Euclidean nearest-neighbors of images
- take $\mathcal{E}_{\text{dis}}$ to be $|\mathcal{E}_{\text{sim}}|$ randomly sampled non-neighbors
- $\mathcal{E} = \mathcal{E}_{\text{sim}} \cup \mathcal{E}_{\text{dis}}$, $|\mathcal{E}| = 1.5 \times 10^6$

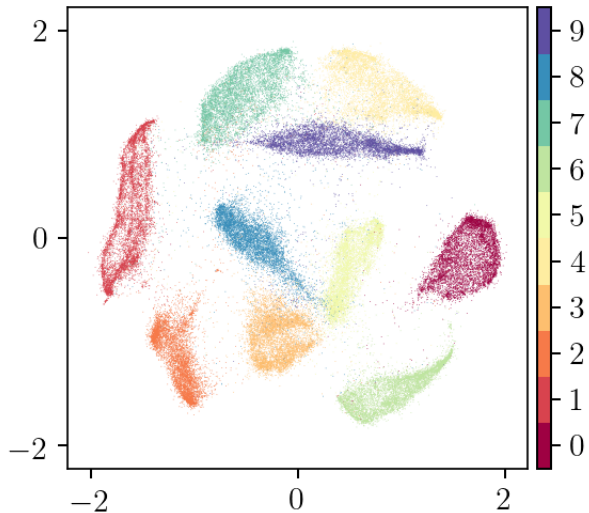
Embedding (MNIST)

- distortion functions

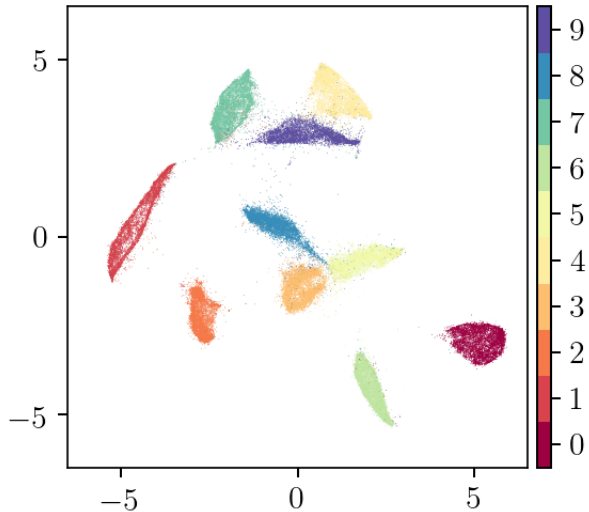
$$f_{ij}(d_{ij}) = \begin{cases} \log(1 + d_{ij}^2) & (i,j) \in \mathcal{E}_{\text{sim}} \\ -\log(1 - \exp(-d_{ij})) & (i,j) \in \mathcal{E}_{\text{dis}} \end{cases}$$

- two embeddings, one with standardization constraint, other centered
- embed into $\mathbf{R}^2$
- roughly 6s GPU, 30s CPU
- color by digit
  - digit label not used to make embedding

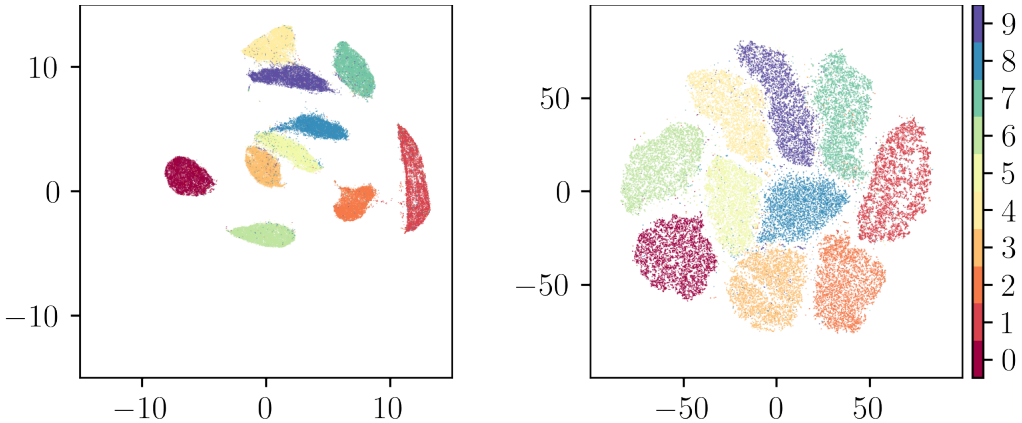
Standardized embedding (MNIST)



Centered embedding (MNIST)

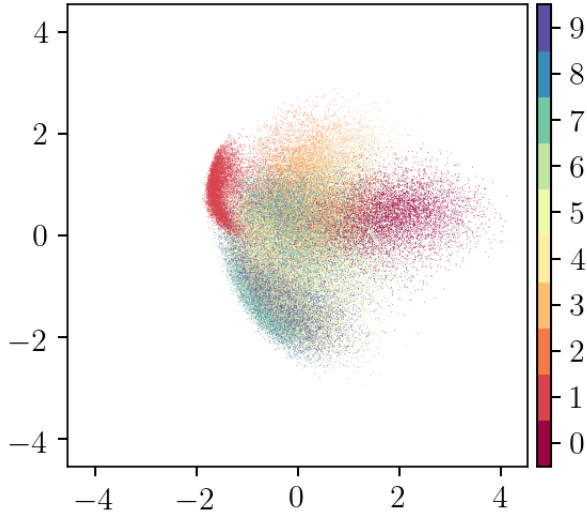


Comparison to UMAP, t-SNE (MNIST)



UMAP (left) and t-SNE (right) embeddings

PCA (MNIST)



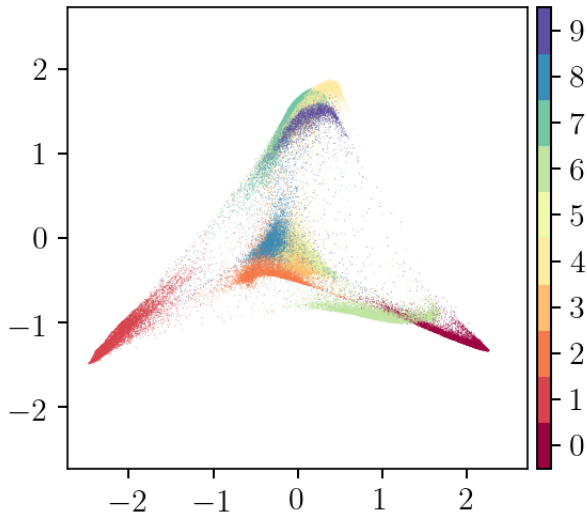
PCA (MNIST)

```
import pymde


mnist = pymde.datasets.MNIST()
X = pymde.pca(mnist.data, embedding_dim=2)
pymde.plot(X, color_by=mnist.attributes['digits'])
```

Laplacian embedding (MNIST)

- $f_{ij}(d_{ij}) = d_{ij}^2$, $(i,j) \in \mathcal{E}_{\text{sim}}$
- standardization constraint $\mathcal{S}$
- embed into $\mathbf{R}^2$
  - 1.8s GPU, 3.8s CPU
- embed into $\mathbf{R}^3$
  - 2.6s GPU, 7.5s CPU

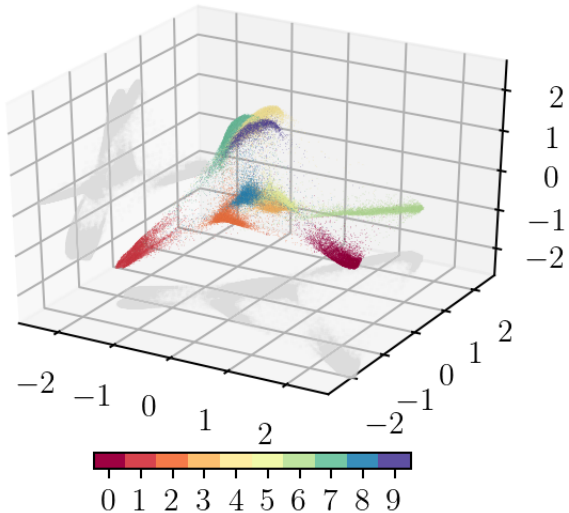
Laplacian embedding (MNIST)



Laplacian embedding (MNIST)

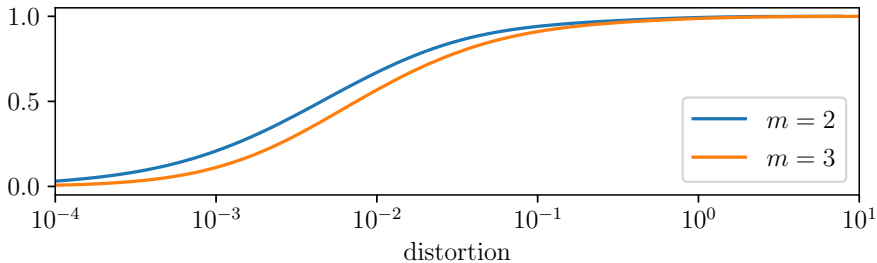
```
import pymde

mnist = pymde.datasets.MNIST()
mde = pymde.preserve_neighbors(
  mnist.data,
  embedding_dim=2,
  attractive_penalty=pymde.penalties.Quadratic,
  repulsive_penalty=None,
  constraint=pymde.Standardized())
X = mde.embed()
pymde.plot(X, color_by=mnist.attributes['digits'])
```

Laplacian embedding (MNIST)



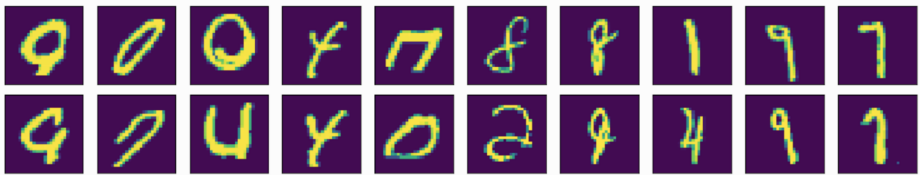
Distortions CDF (MNIST)

- sanity check: plot distortion CDFs
- most distortions small, but some very large
- plot with `mde.distortions_cdf()`



High distortion pairs (MNIST)

- sanity check: inspect pairs with highest distortion
- get pairs with `mde.high_distortion_pairs()`
- analogous to checking classification errors in machine learning
- here, images look odd or poorly paired



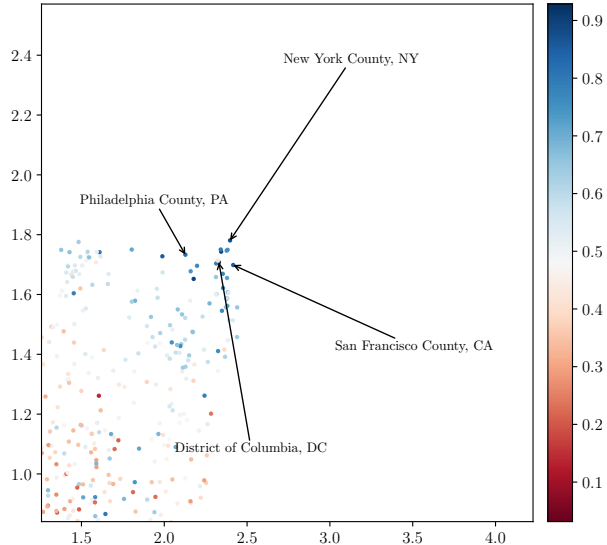
Standardized embedding (MNIST)

```
import pymde

mnist = pymde.datasets.MNIST()
mde = pymde.preserve_neighbors(
  mnist.data,
  embedding_dim=2,
  attractive_penalty=pymde.penalties.Log1p,
  repulsive_penalty=pymde.penalties.Log,
  constraint=pymde.Standardized())
X = mde.embed()
pymde.plot(X, color_by=mnist.attributes['digits'])
```

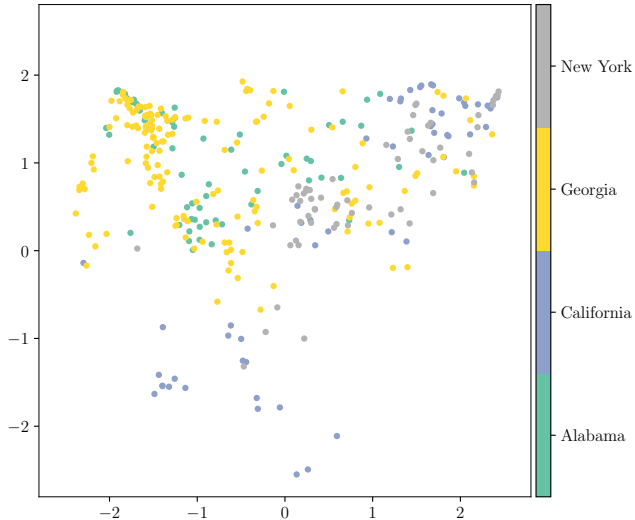
Centered embedding (MNIST)

```
import pymde

mnist = pymde.datasets.MNIST()
mde = pymde.preserve_neighbors(
  mnist.data,
  constraint=pymde.Centered())
X = mde.embed()
pymde.plot(X, color_by=mnist.attributes['digits'])
```

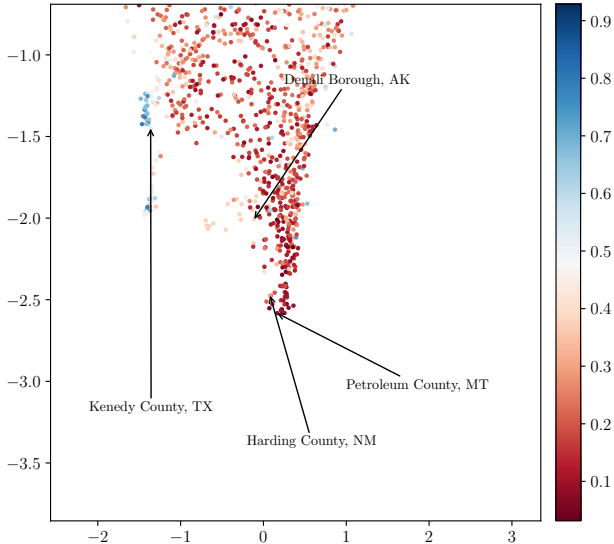
Embedding (US counties)



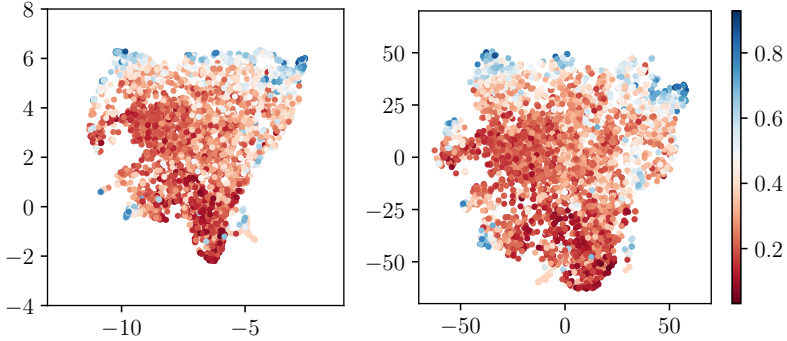
Embedding (US counties)



Embedding (US counties)



Comparison to UMAP, t-SNE (US counties)



UMAP (left) and t-SNE (right) embeddings