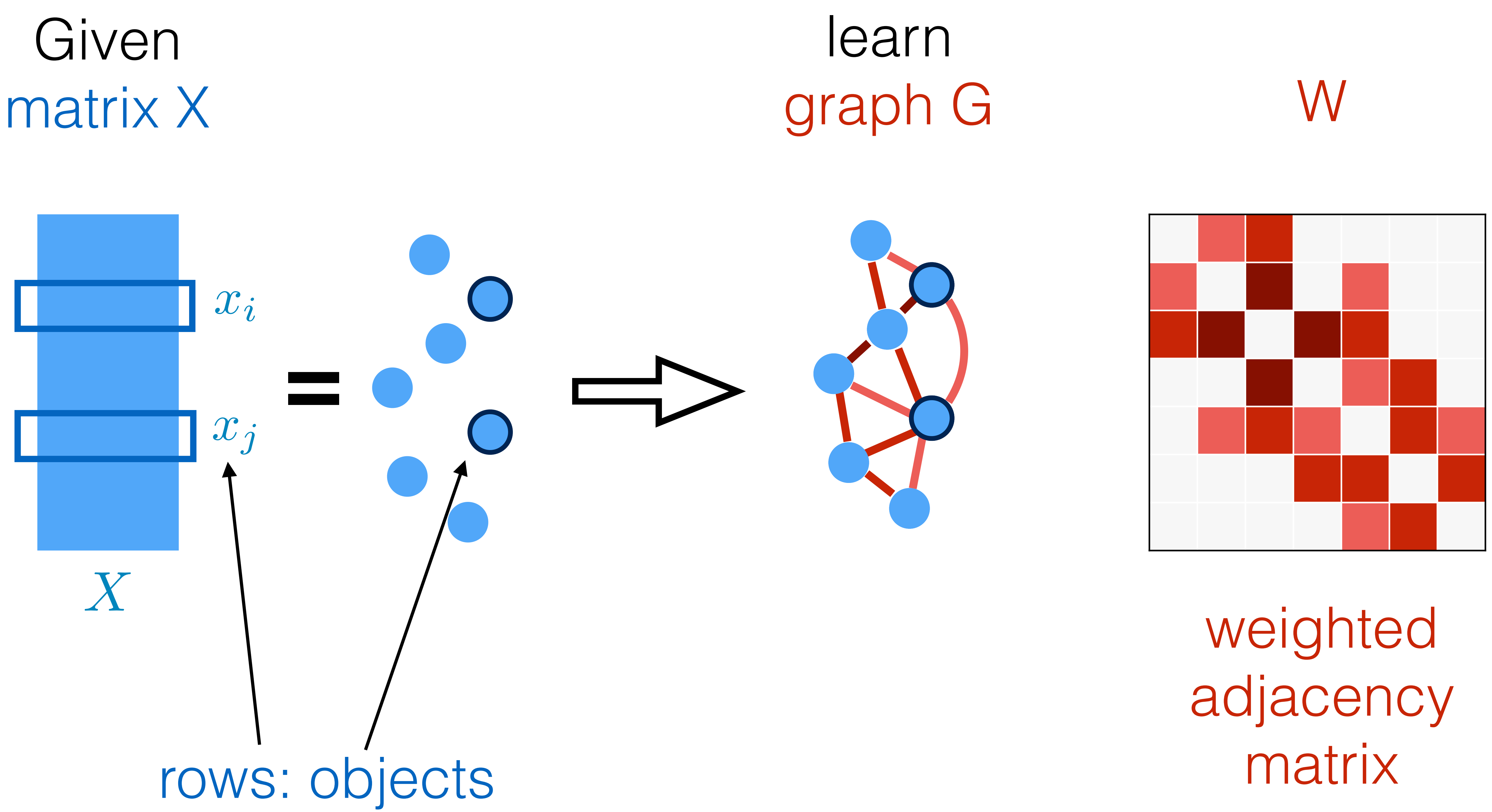
Large scale graph learning from smooth signals

Kalofolias Vassilis
Nathanael Perraudin

13 November 2019

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Graph learning



Dimensionality - manifolds

Interesting problems:

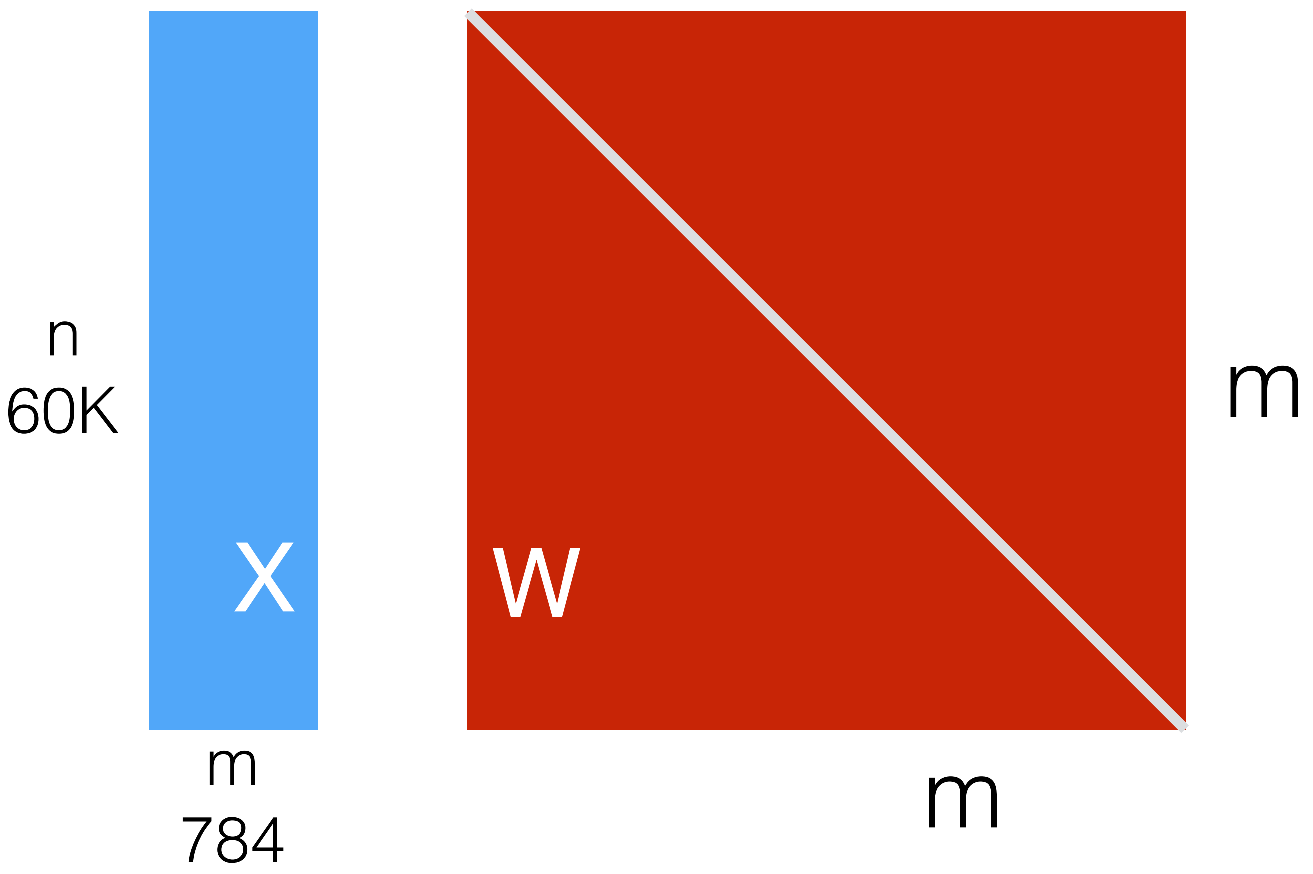
(# nodes) ≫ (# features)

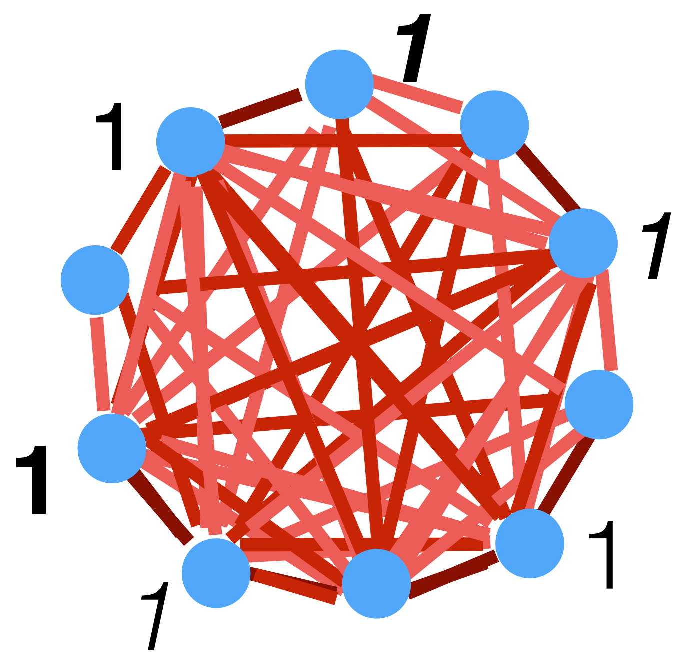
Example: MNIST

No structure?

⇒ Full W

✘ Ill-posed



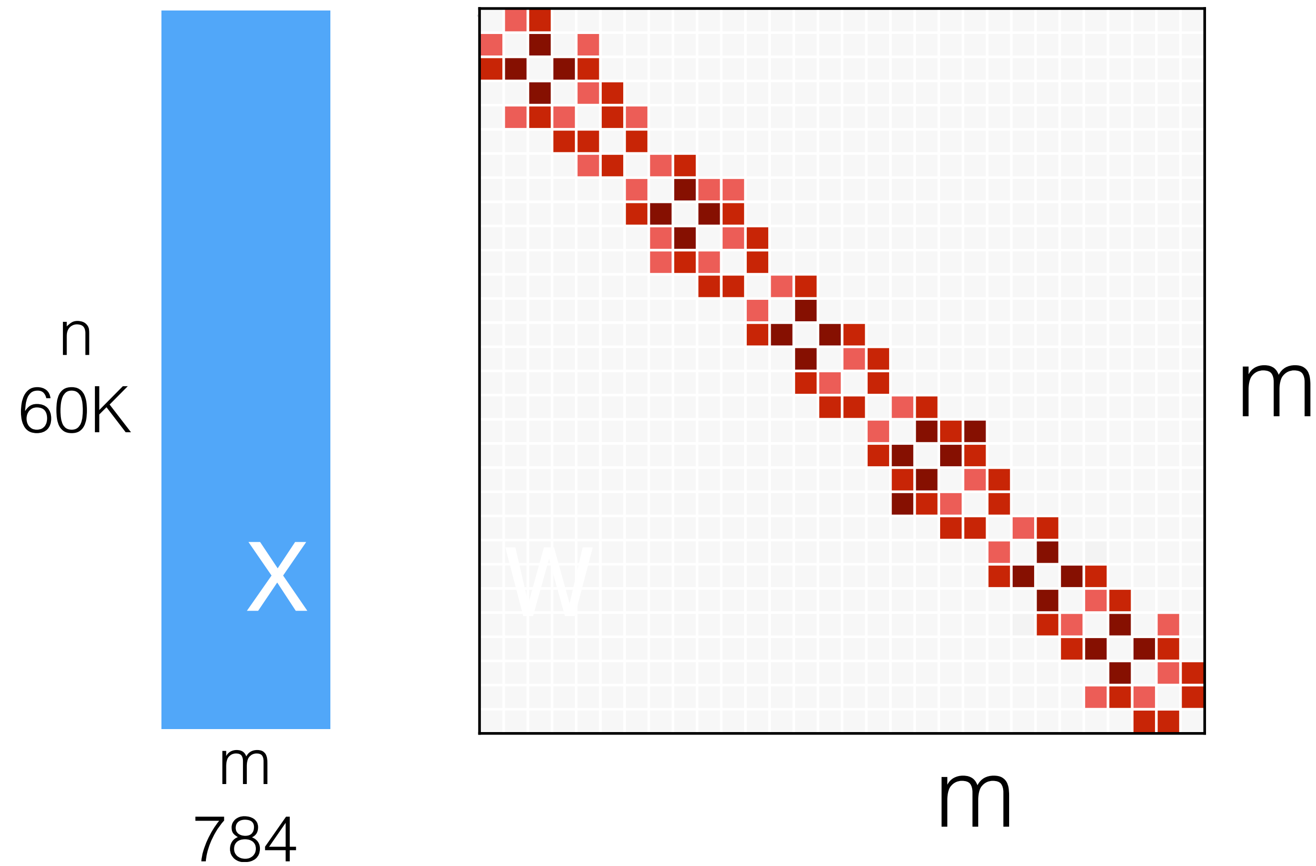


Dimensionality - manifolds

Interesting problems:

(# nodes) ≫ (# features)

Example: MNIST



No structure?

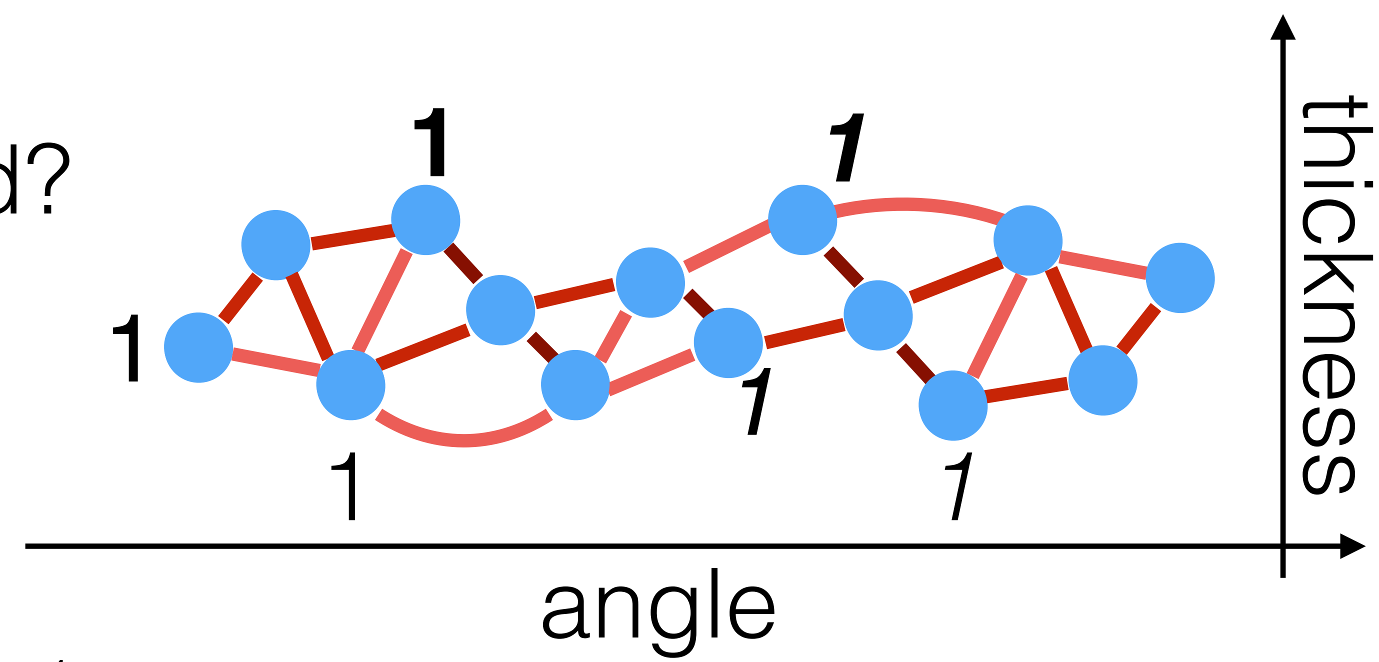
⇒ Full W

✘ Ill-posed

Low-dimensional manifold?

⇒ Local dependencies

✔ Sparse W



Smoothness

Data is **smooth** on graph

Data lives on a low-dimensional manifold

Dirichlet energy is small:

$$\|\nabla_G X\|_F^2 = \mathrm{tr}\left(X^\top L X\right) = \frac{1}{2}\sum_{i,j} W_{i,j}\|x_i - x_j\|_2^2$$

L: $\nabla_G^\top \nabla_G$

$D - W$

$$Z_{ij} = \|x_i - x_j\|^2$$

$$= \frac{1}{2}\|W \circ Z\|_{1,1}$$

data smoothness ⇨ graph sparsity

Graphs from smooth signals

$$\min \ \mathrm{tr} \ \| LX \|_F^2 \quad \text{s.t} \quad W\mathbf{1} \geq \mathbf{1}$$

[Daitch etal 2009]

$$\min \ \mathrm{tr} \ \| LX \|_F^2 \quad \text{s.t} \quad \mathbf{1}^\top \max\left(\mathbf{0}, W\mathbf{1}\right)^2 \leq \alpha n$$

[Daitch etal 2009]

$$\min \ \mathrm{tr}\left(X^\top L X\right) - \log|L + \alpha I| + \beta \|W\|_{1,1}$$

[Lake & Tenenbaum 2010]

$$\min \ \mathrm{tr}\left(X^\top L X\right) + \alpha \|L\|_F^2 \quad \text{s.t.} \quad \mathrm{tr}(L) = n$$

[Hu etal 2015, Dong etal 2016]

$$\min \ \mathrm{tr}\left(X^\top L X\right) - \alpha \mathbf{1}^\top \log(W\mathbf{1}) + \frac{\beta}{2}\|W\|_F^2$$

[Kalofolias 2016]

The log-degrees model

$$\min_{W \in \mathcal{W}_m} \|W \circ Z\|_{1,1} - \alpha \mathbf{1}^\top \log(W\mathbf{1}) + \frac{\beta}{2}\|W\|_F^2$$

- First algorithm of $\mathcal{O}(n^2)$
- Best results among “scalable” models

Goal: scale it further!

How to scale it?

1. Reduce the number of variables from $\mathcal{O}(n^2)$
2. Eliminate grid search: automatic parameter selection

How to scale it?

1. **Reduce the number of variables from** $\mathcal{O}(n^2)$
2. Eliminate grid search: automatic parameter selection

Optimization

Objective can be split in 3 functions:

$$\min_{W \in \mathcal{W}_m} \|W \circ Z\|_{1,1} - \alpha \mathbf{1}^\top \log(W\mathbf{1}) + \frac{\beta}{2}\|W\|_F^2$$

Sketch of algorithm:

Approximately minimize each function

1. Shrink edges according to distance $\mathcal{O}(n^2)$
2. Enhance edges of badly connected nodes $\mathcal{O}(n^2)$
3. Shrink large edges $\mathcal{O}(n^2)$

Optimization

Objective can be split in 3 functions:

$$\min_{W \in \mathcal{W}_m} \|M \circ W \circ Z\|_{1,1} - \alpha \mathbf{1}^\top \log((M \circ W)\mathbf{1}) + \frac{\beta}{2}\|M \circ W\|_F^2$$

1. Shrink edges according to distance $\mathcal{O}\left(\left|\mathcal{E}^{\text{allowed}}\right|\right)$
2. Enhance edges of badly connected nodes $\mathcal{O}\left(\left|\mathcal{E}^{\text{allowed}}\right|\right)$
3. Shrink large edges $\mathcal{O}\left(\left|\mathcal{E}^{\text{allowed}}\right|\right)$

Reducing allowed edge set

How do we choose a restricted edge set?

- Prior: structure imposed by application
  e.g. geometric constraints

What if no structure known?

- Approximate Nearest Neighbours (ANN)

Using ANN to reduce cost

“I want a graph with 10 edges per node on average” $k = 10$

↓

Compute approximate 30 NN graph (binary) $\mathcal{O}\left(n \log(n) m\right)$

↓ $|\mathcal{E}^{\text{allowed}}| \approx 3kn = 30n$

Learn weights for allowed edges $\mathcal{O}\left(nk\right)$

↓

Some of them are deleted! (W_{ij}=0)

↓

Final 10 NN graph **Cost?**

How to scale it?

1. Reduce the number of variables from $\mathcal{O}(n^2)$
2. **Eliminate grid search: automatic parameter selection**

Change of parameters

"I want a graph with 20 edges per node on average"

Grid search? ✘

$$W^*(Z, \alpha, \beta) = \arg\min_{W \in \mathcal{W}_m} \|W \circ Z\|_{1,1} - \alpha \mathbf{1}^\top \log(W\mathbf{1}) + \frac{\beta}{2}\|W\|_F^2$$

$$= \delta \arg\min_{W \in \mathcal{W}_m} \|W \circ \theta Z\|_{1,1} - \mathbf{1}^\top \log(W\mathbf{1}) + \frac{1}{2}\|W\|_F^2$$

$\delta = \sqrt{\frac{\alpha}{\beta}}$ $\theta = \sqrt{\frac{1}{\alpha\beta}}$

$$= \delta W^*(\theta Z, 1, 1)$$

$$\leq \delta$$

Only θ changes sparsity

δ only changes the scale

Sparsity of one node

So δ is not important. How do we find θ?

$$\min_{W \in \mathcal{W}_m} \|W \circ \theta Z\|_{1,1} - \mathbf{1}^\top \log(W\mathbf{1}) + \frac{1}{2}\|W\|_F^2$$

Take **1** node:

$$\min_{w \geq 0} \theta w^\top z - \log(w^\top \mathbf{1}) + \frac{1}{2}\|w\|_2^2.$$

ignore symmetricity

1 column of W

Analyse role of θ on simpler problem!

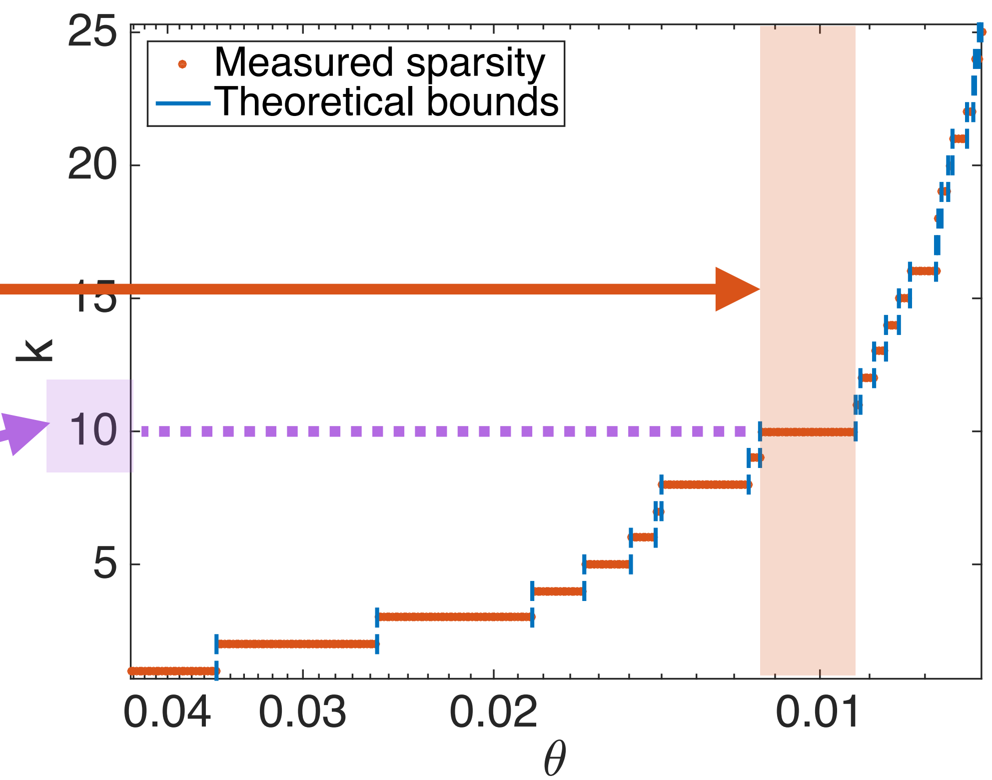
Sparsity of one node

Theorem:

By setting θ in the range $\left(\frac{1}{\sqrt{kz_{k+1}^2 - b_k z_{k+1}}}, \frac{1}{\sqrt{kz_k^2 - b_k z_k}}\right]$, w^* has exactly k non-zero elements.

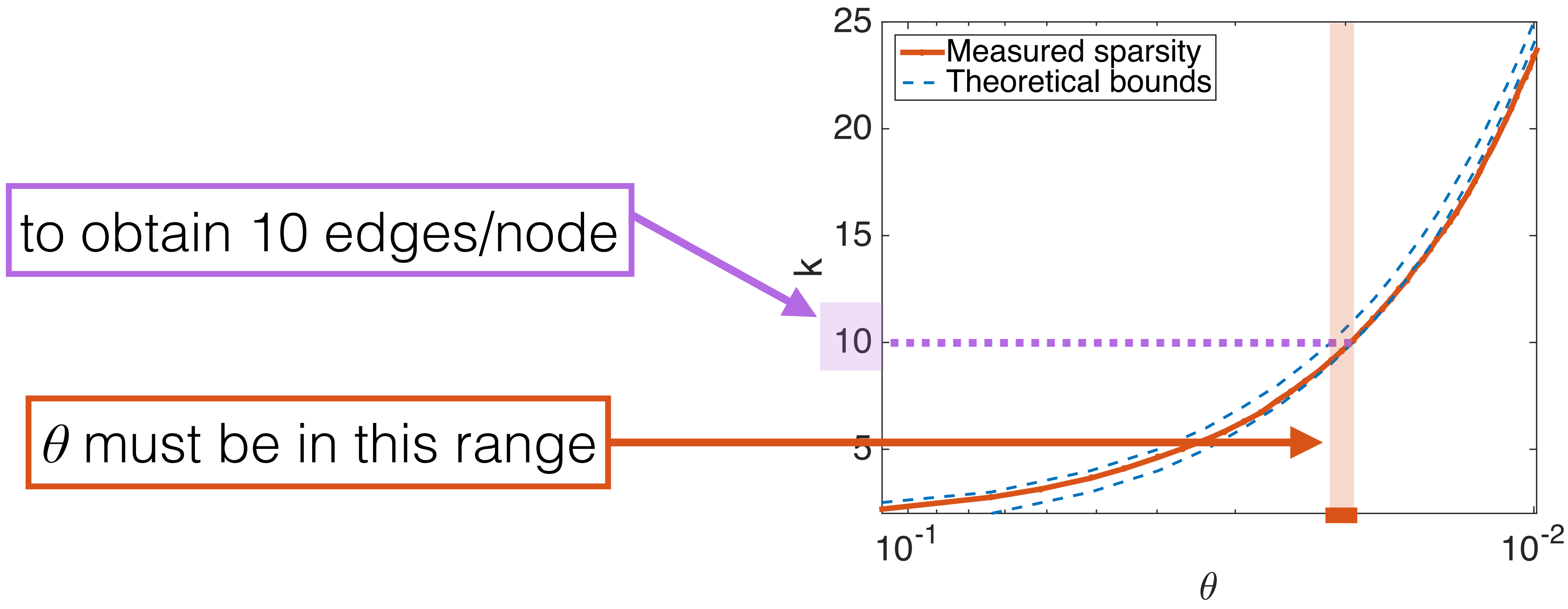
If θ in this range

we obtain 10 non zeros

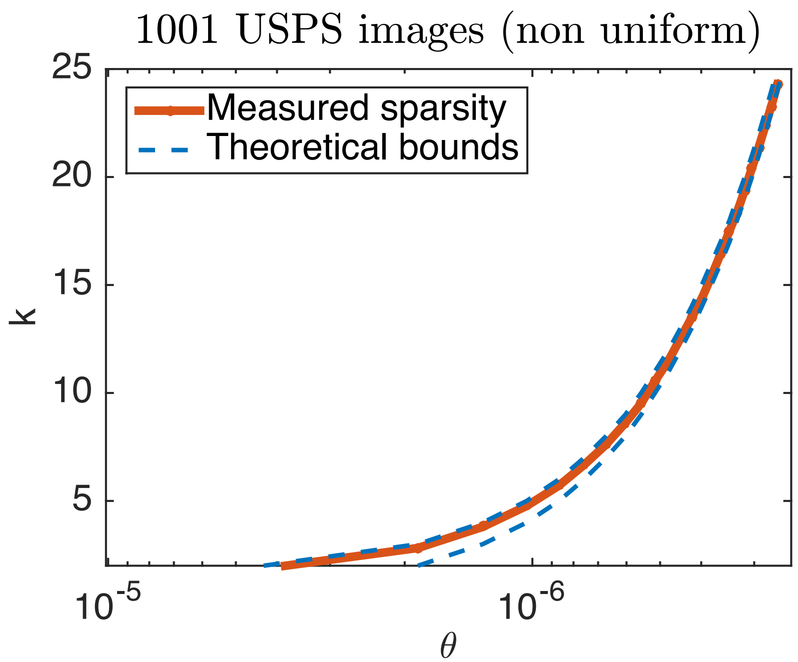


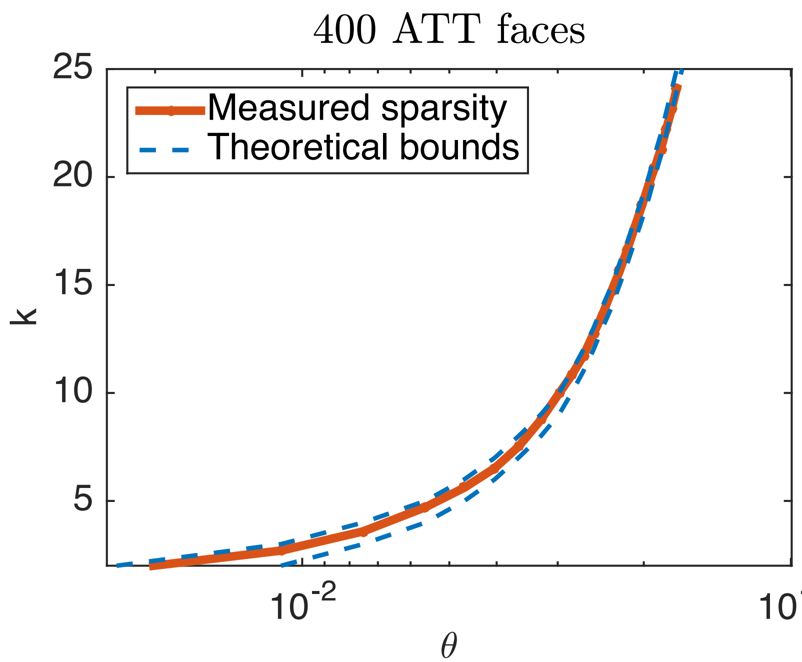
Sparsity of entire graph

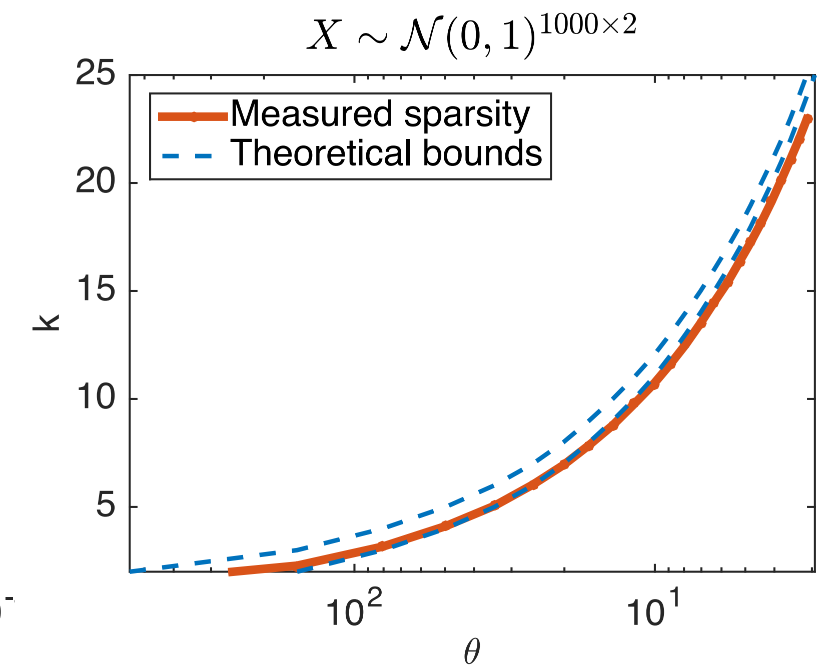
Use **average** over all nodes



More examples

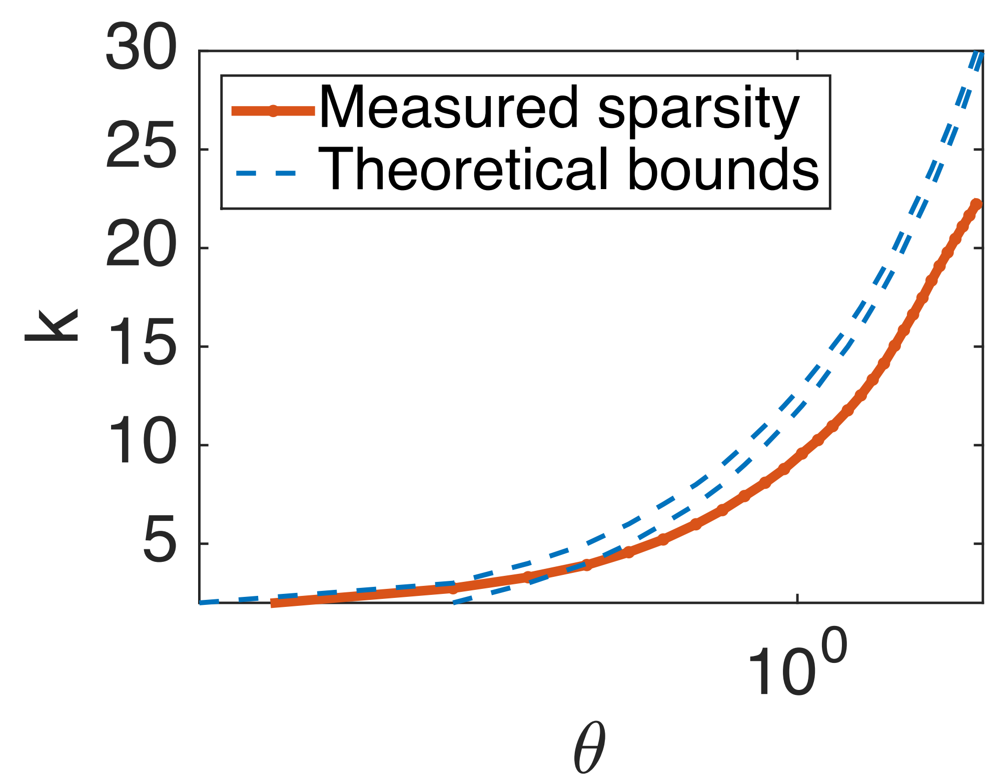






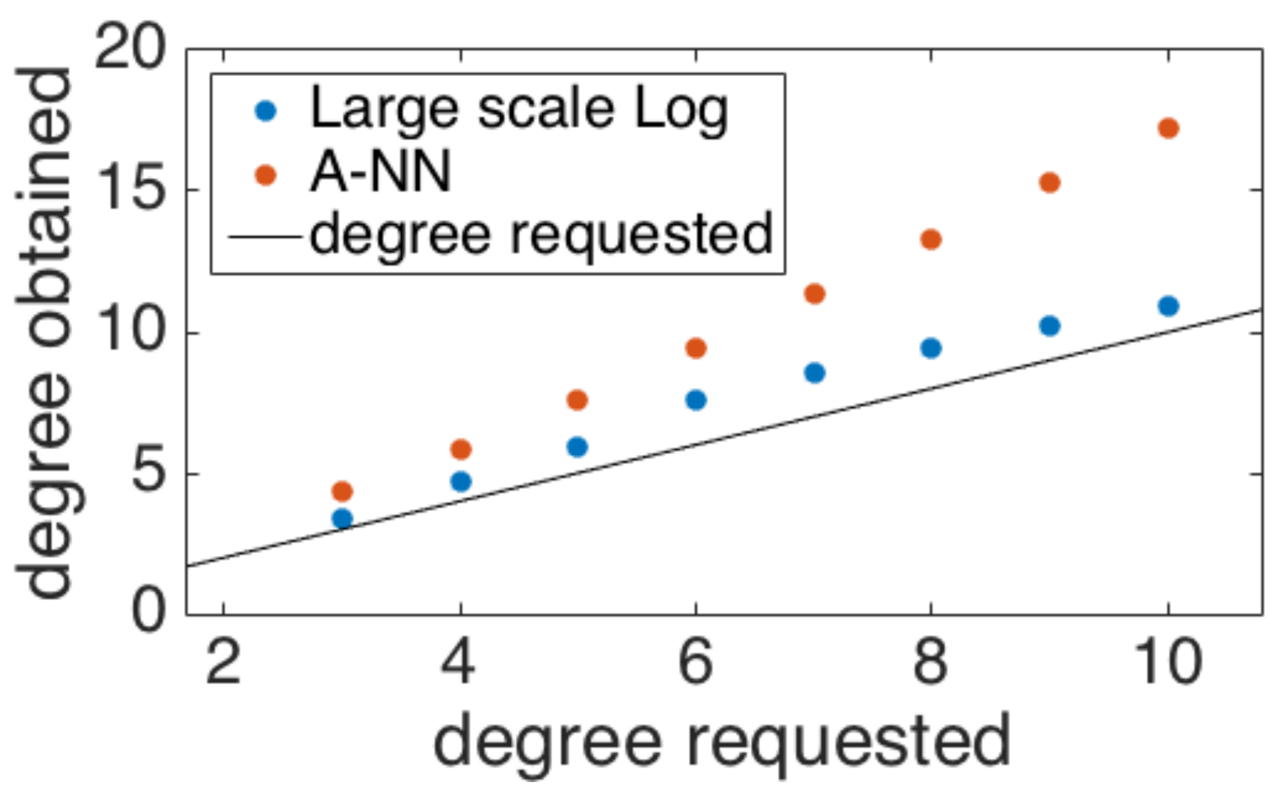
"Failing" case:

- COIL20, n = 1440



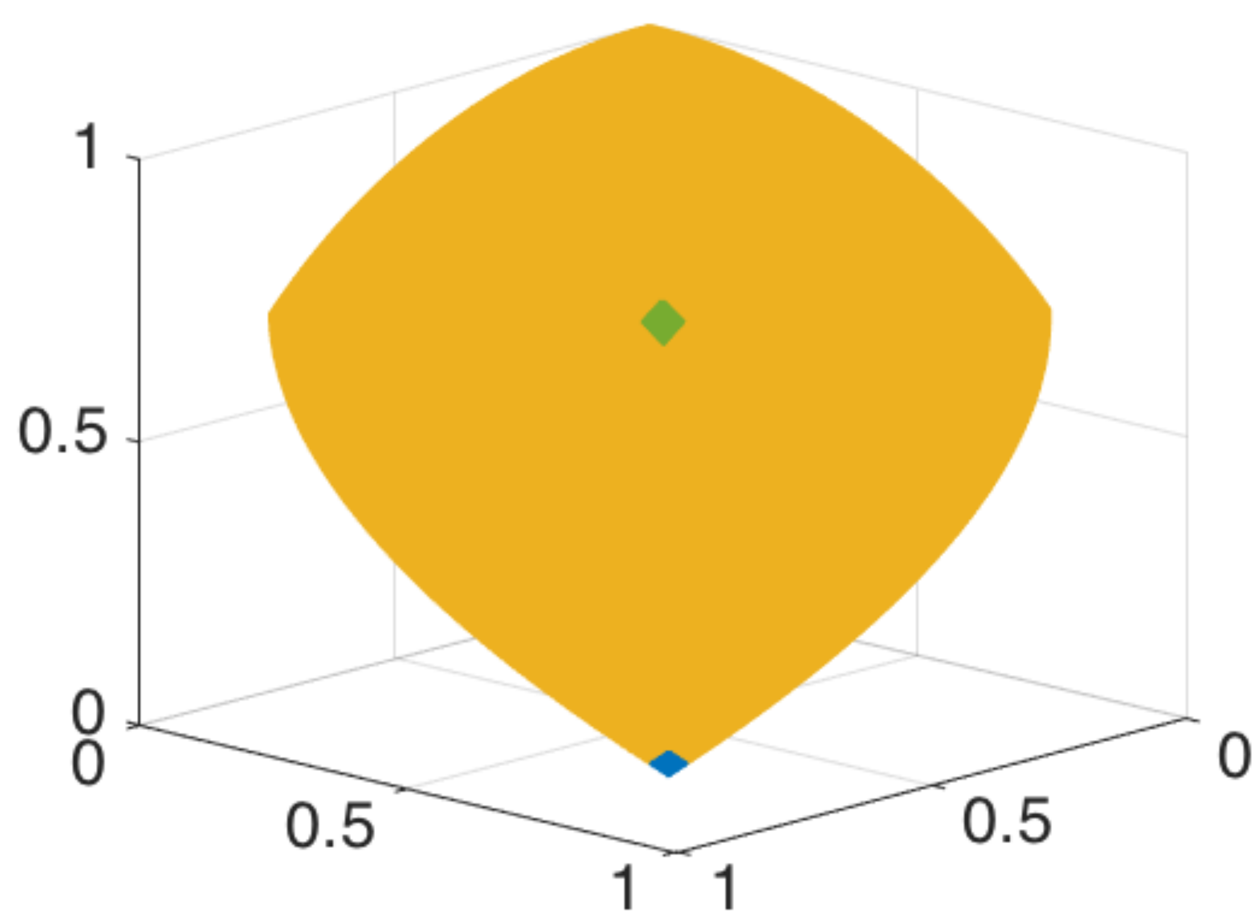
Obtained degrees

"spherical" data (n = 260K)



Manifold recovery

"spherical" data (n = 260K, m = 2K)

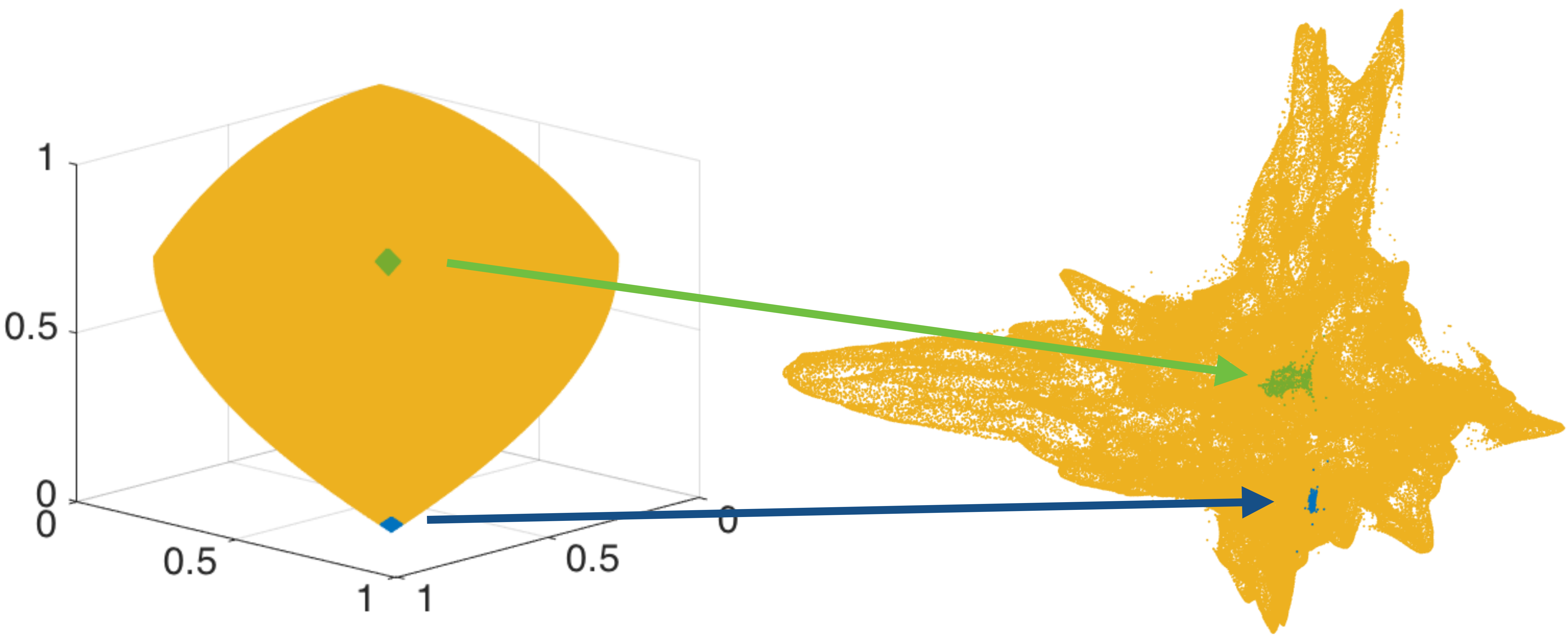


Manifold recovery

“spherical” data (n = 260K, m = 2K)

Original 2-D manifold

Recovered with ANN

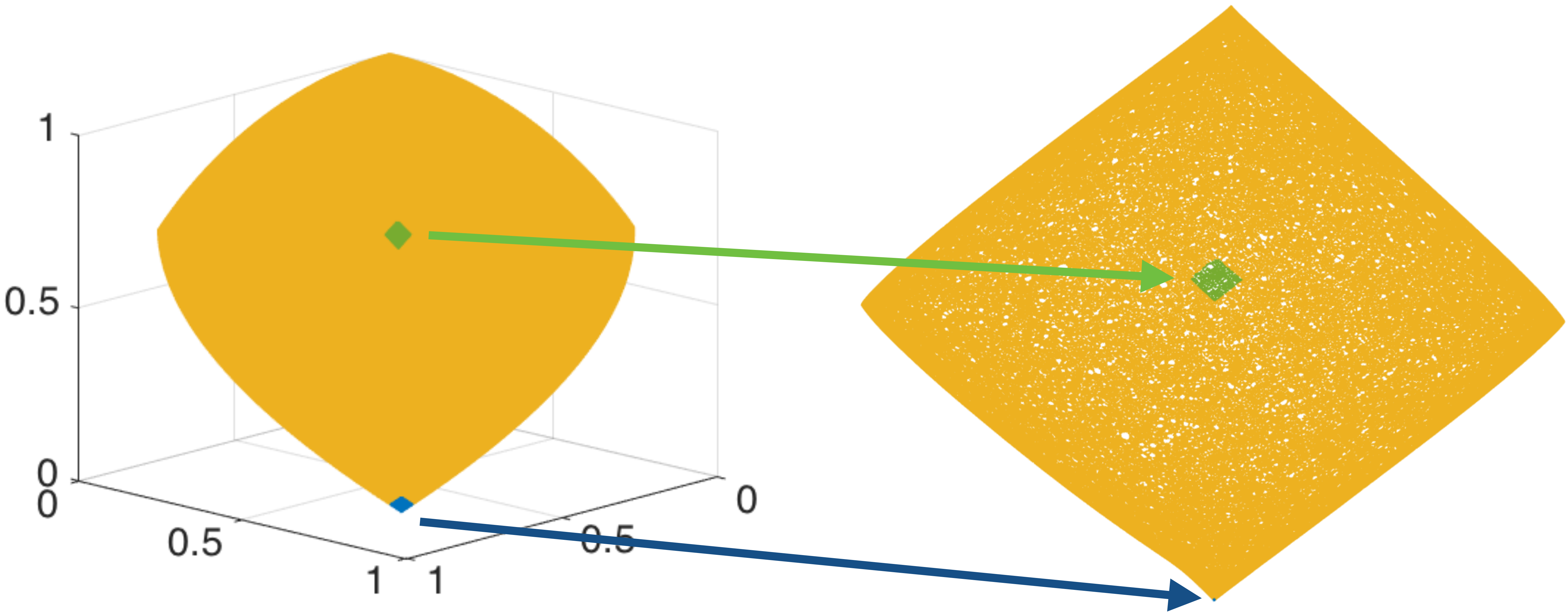


Manifold recovery

“spherical” data (n = 260K)

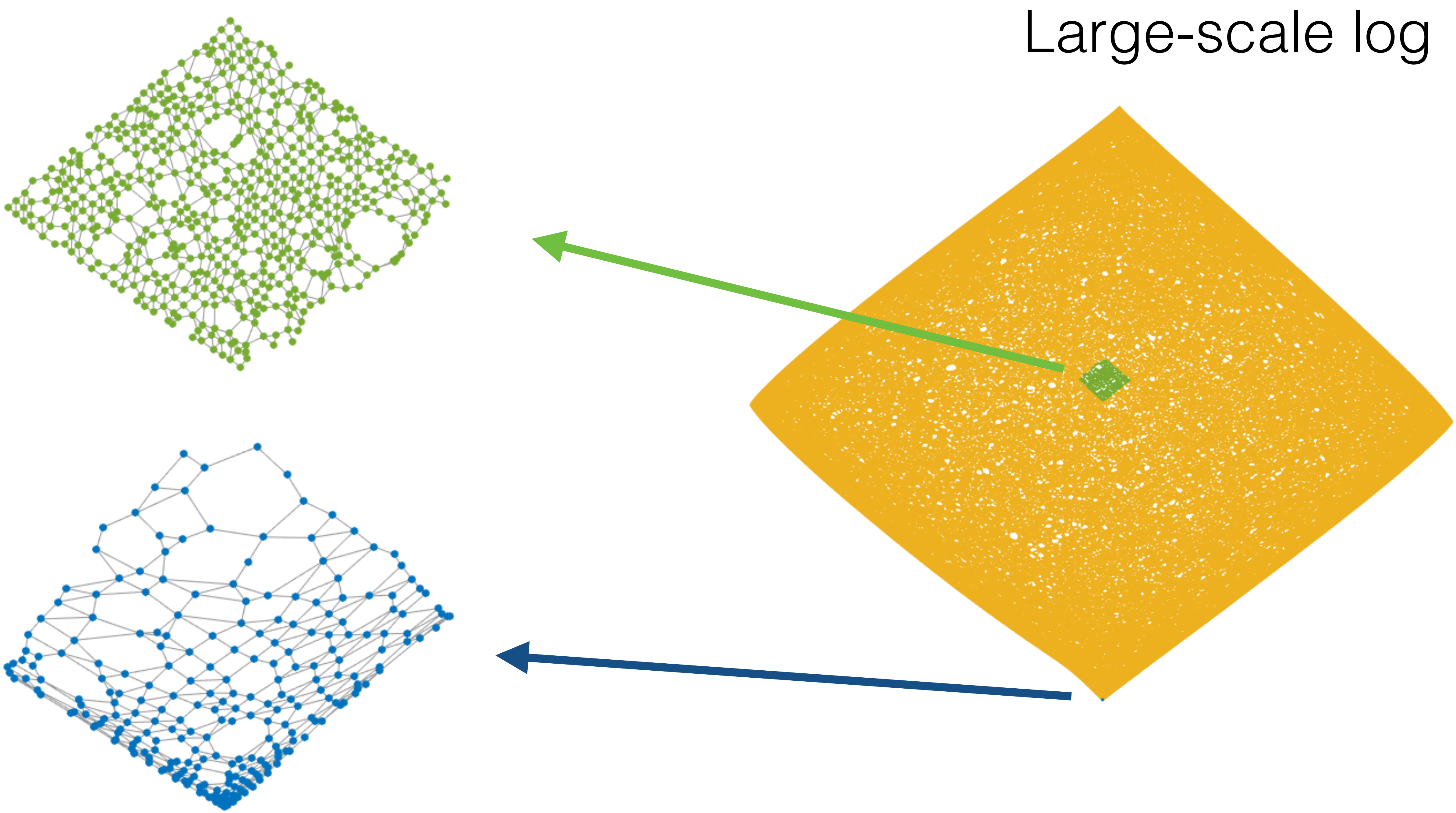
Original 2-D manifold

Large-scale log



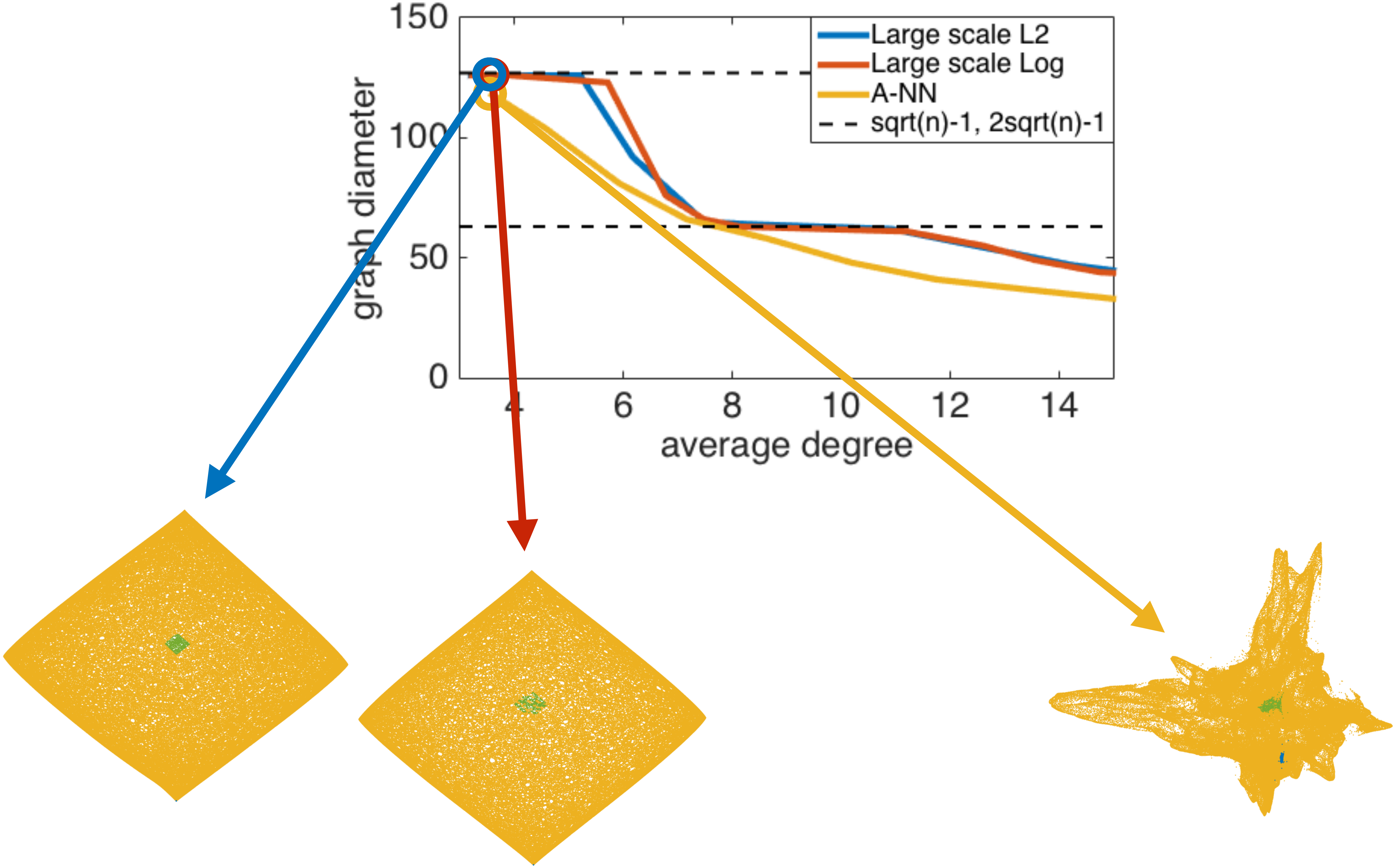
Manifold recovery

"spherical" data (n = 260K)



Manifold recovery

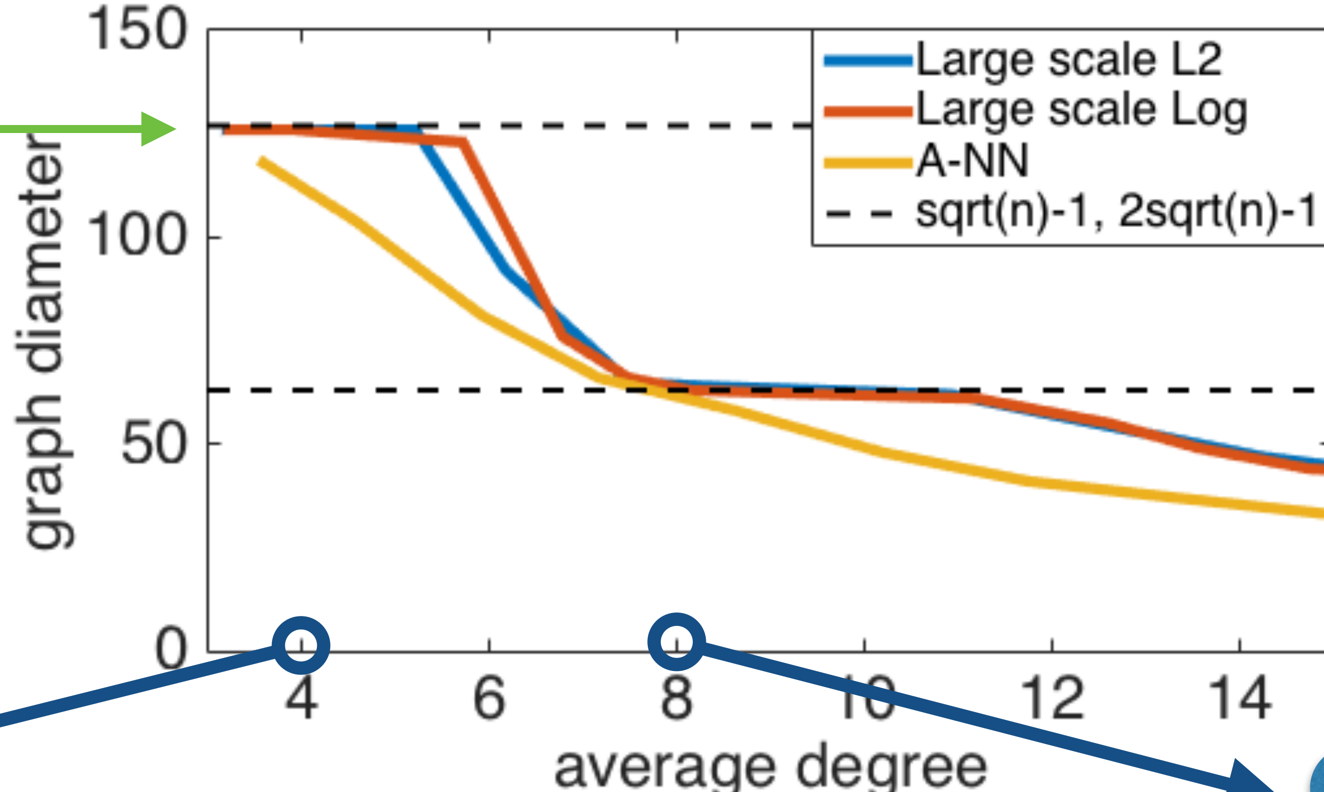
"spherical" data (n = 4K, m = 2K)



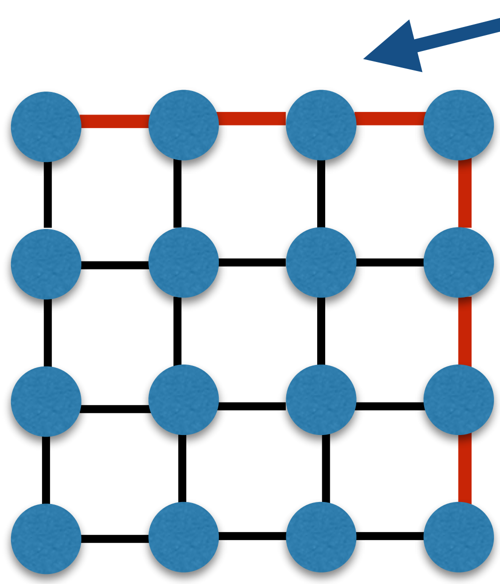
Manifold recovery

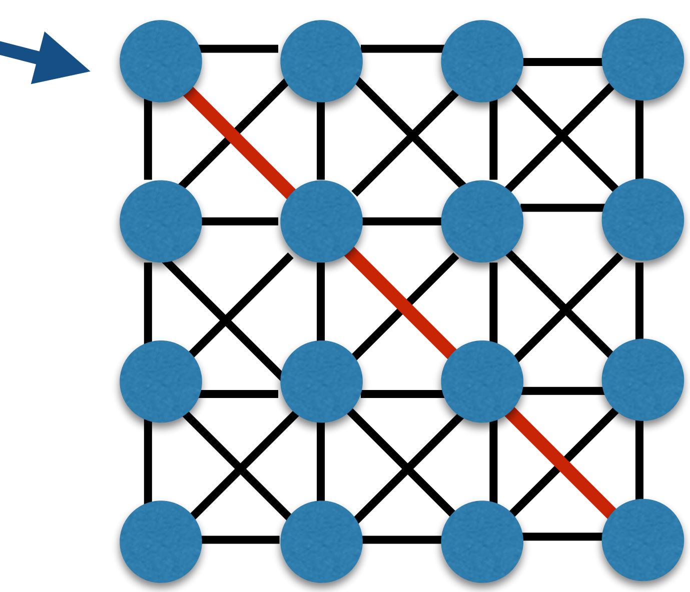
"spherical" data (n = 4K, m = 2K)

diameter = $2\left(\sqrt{n} - 1\right)$



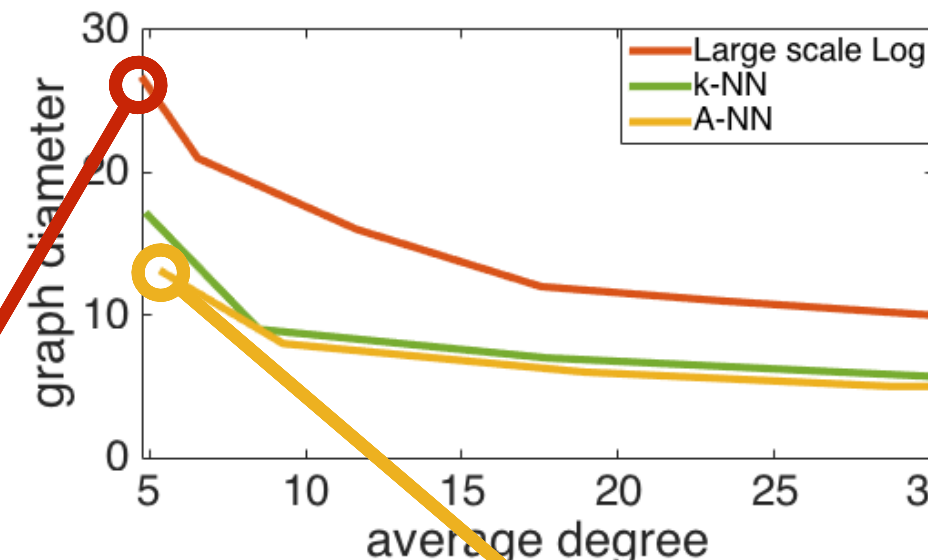
diameter $= \sqrt{n} - 1$





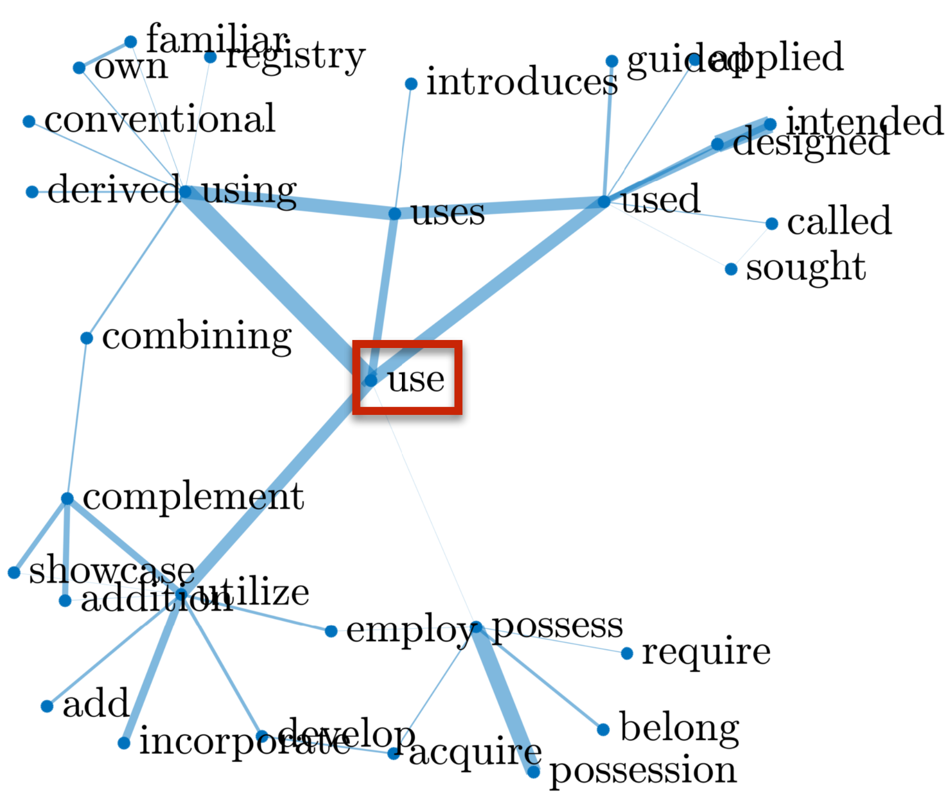
Manifold recovery

Word2vec (n = 10K, m = 300)

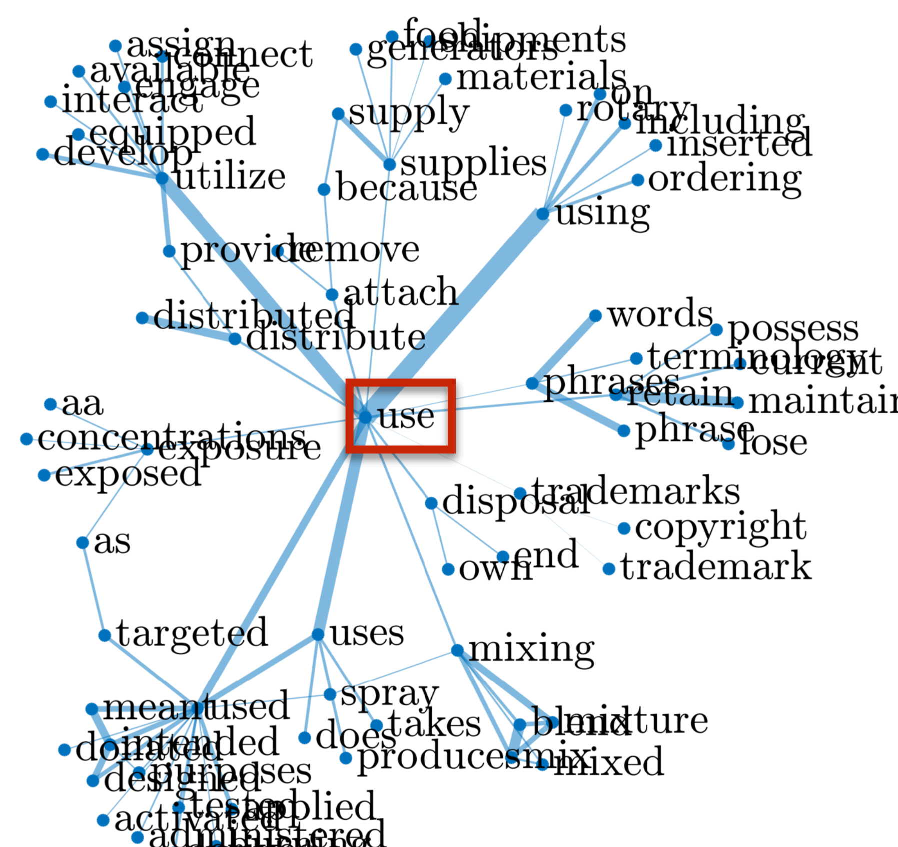


2-hops subgraph from term “use"

29 words



69 words

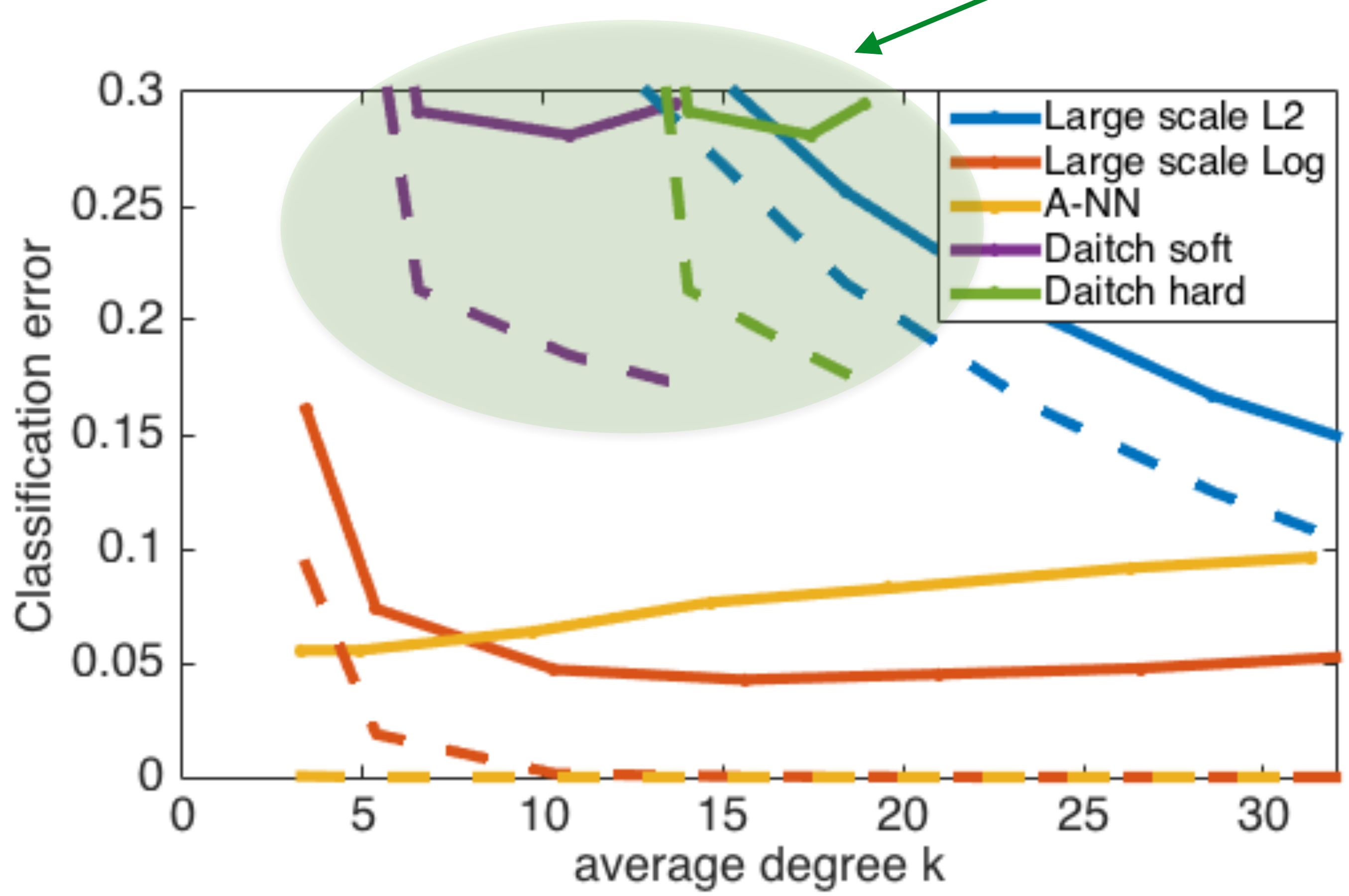


Label propagation

MNIST (n = 60K)

Label propagation (1% known labels)

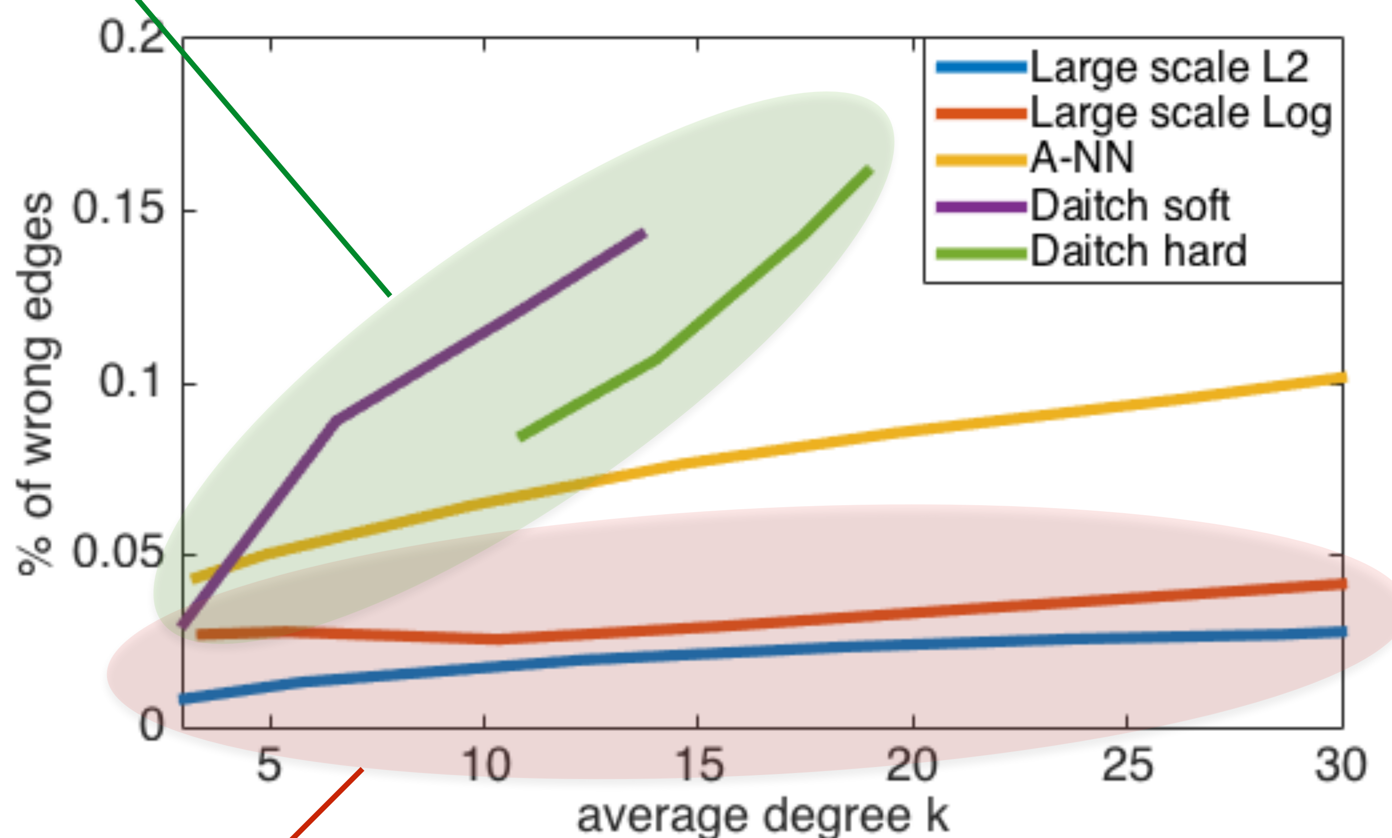
$$\| LX \|_F^2$$



Edge accuracy

MNIST (n = 60K)

$\| LX \|_F^2$



$\operatorname{tr}\left(X^\top L X\right) = \| WZ \|_{1,1}$

Summary

1. Good manifold recovery

2. Scalable!
   - ✔ $\mathcal{O}(nk)$ per iteration $\ll \mathcal{O}(n^2)$
   - ✔ $\mathcal{O}(n\log(n)m)$ (one time)

3. No need for parameter tuning
   - ✔ Automatic parameter selection for desired sparsity

Code: Matlab & Python (GSP box, pyGSP)

Thank you!