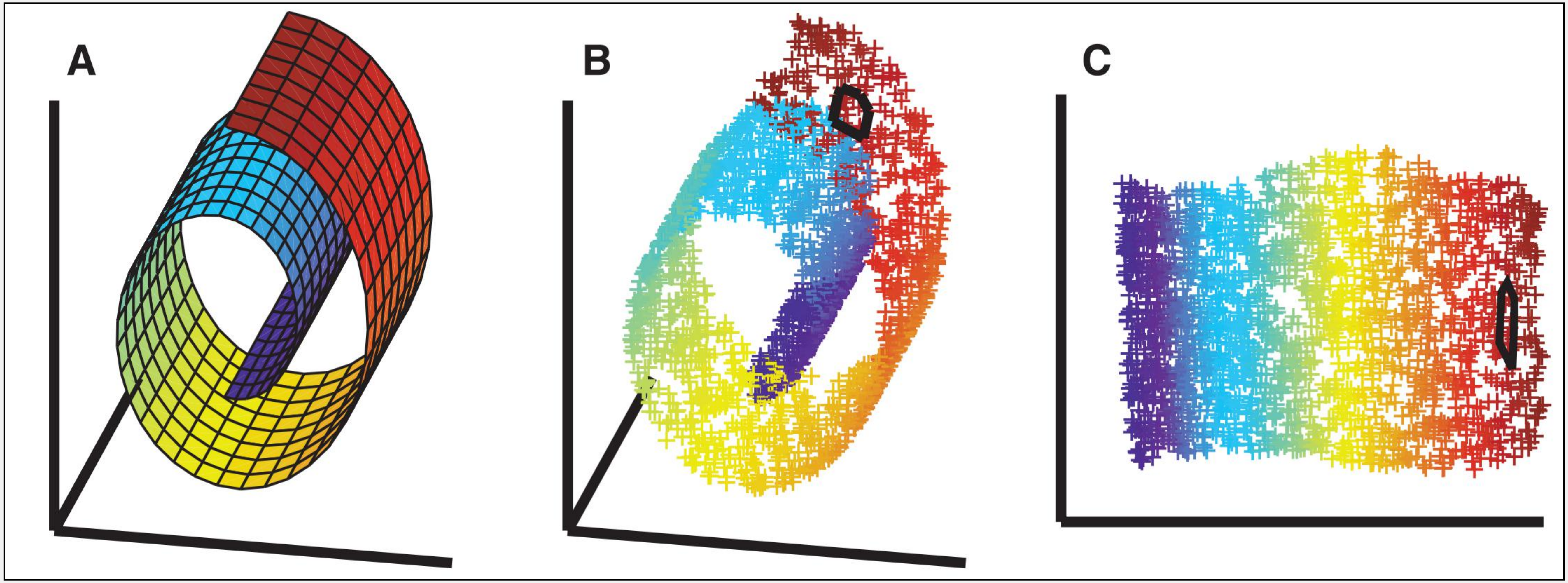
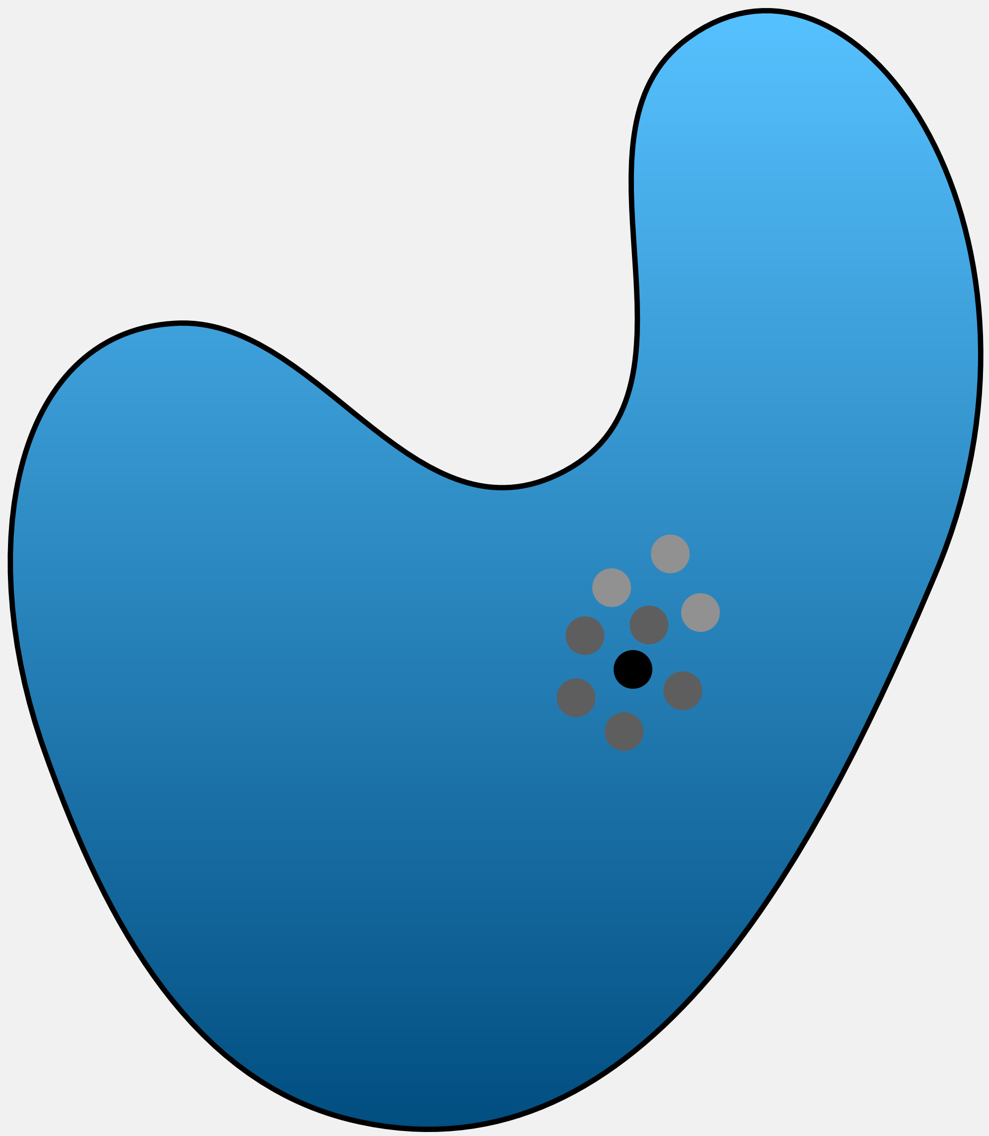
# Nonlinear Dimensionality Reduction

- Data cannot be described as living on a linear subspace, but rather, a manifold:

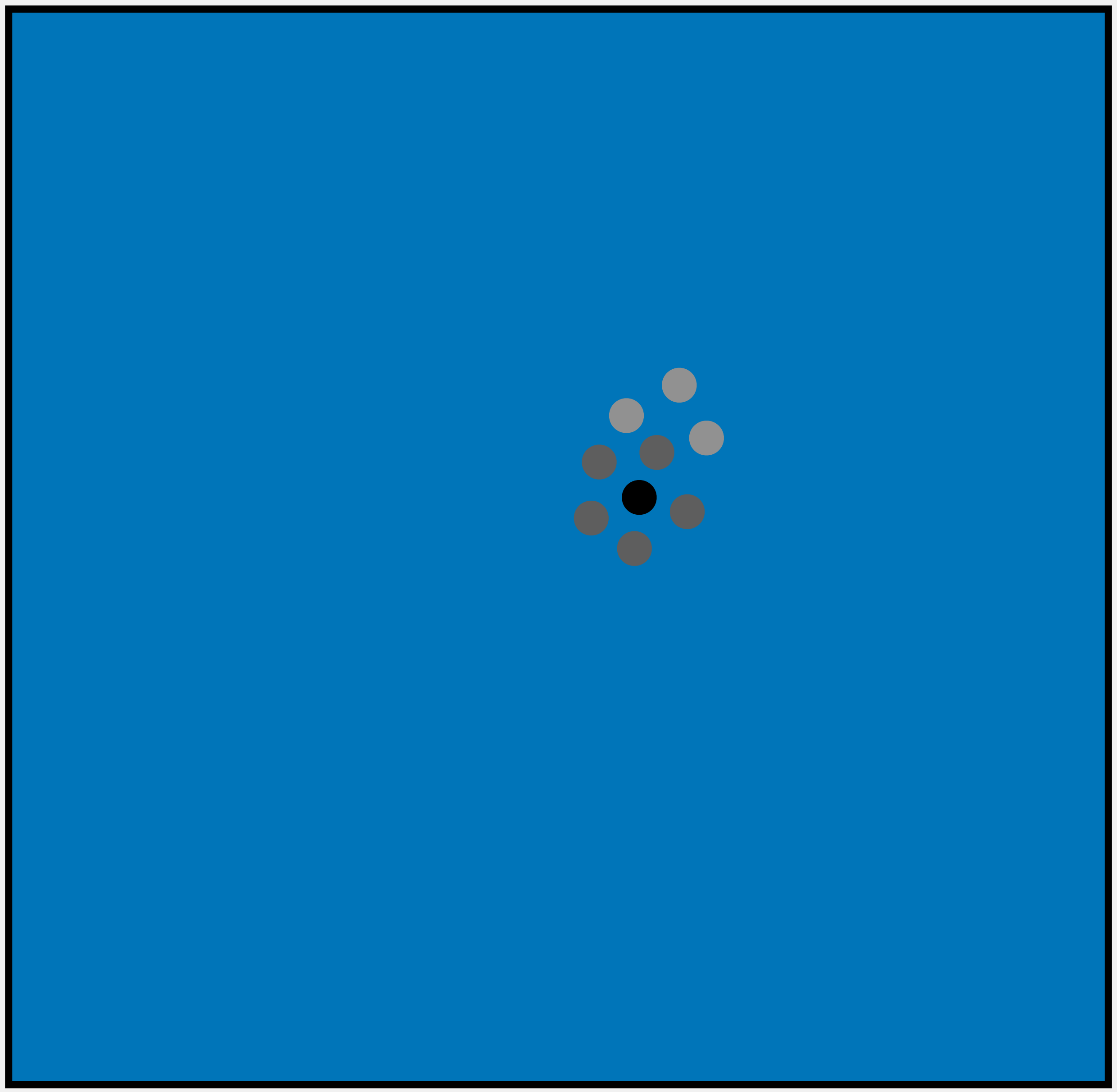



[Roweis et al. 2000]

# Intuition: Nonlinear Dimensionality Reduction

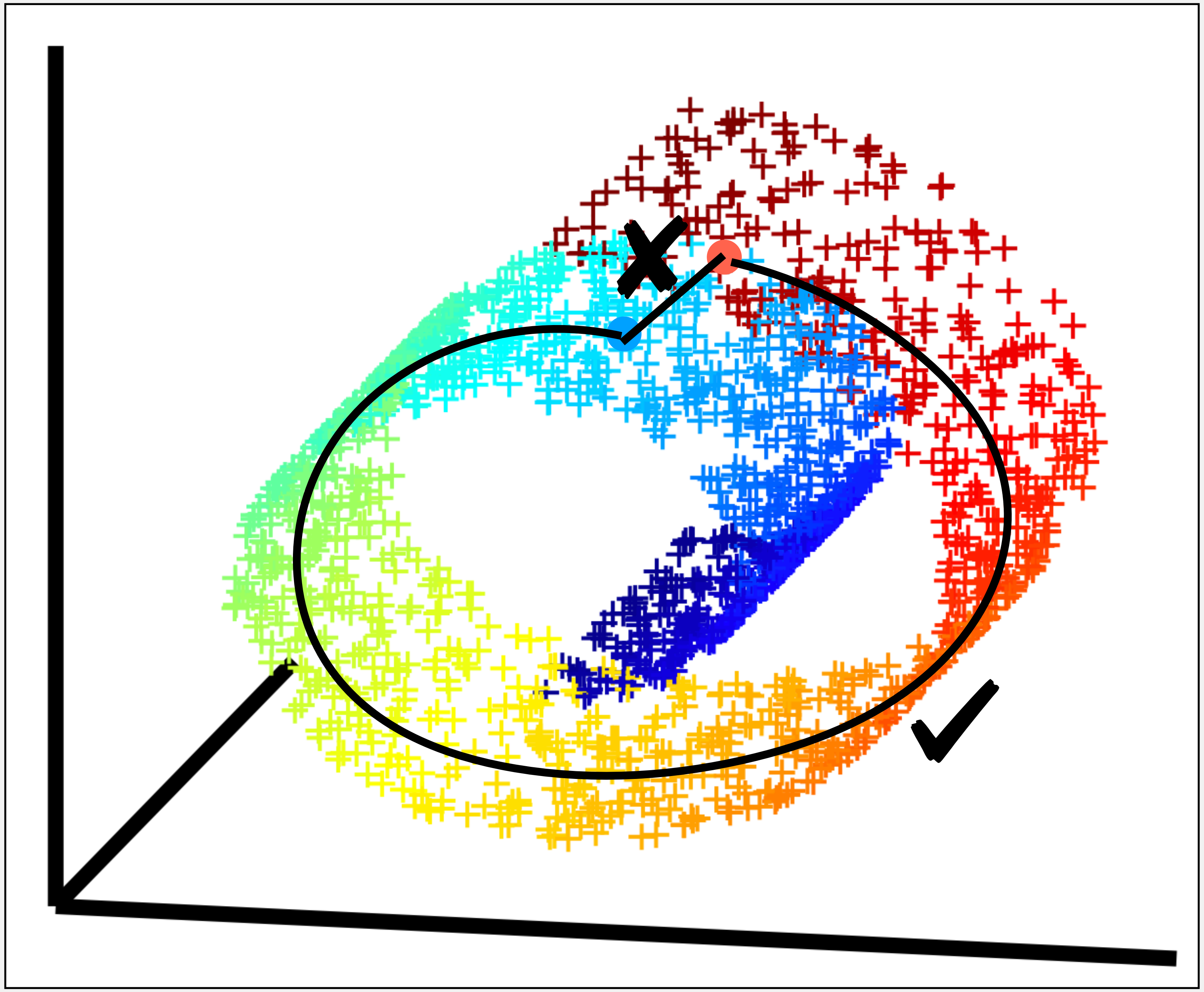


Surface in 3D



2D Space

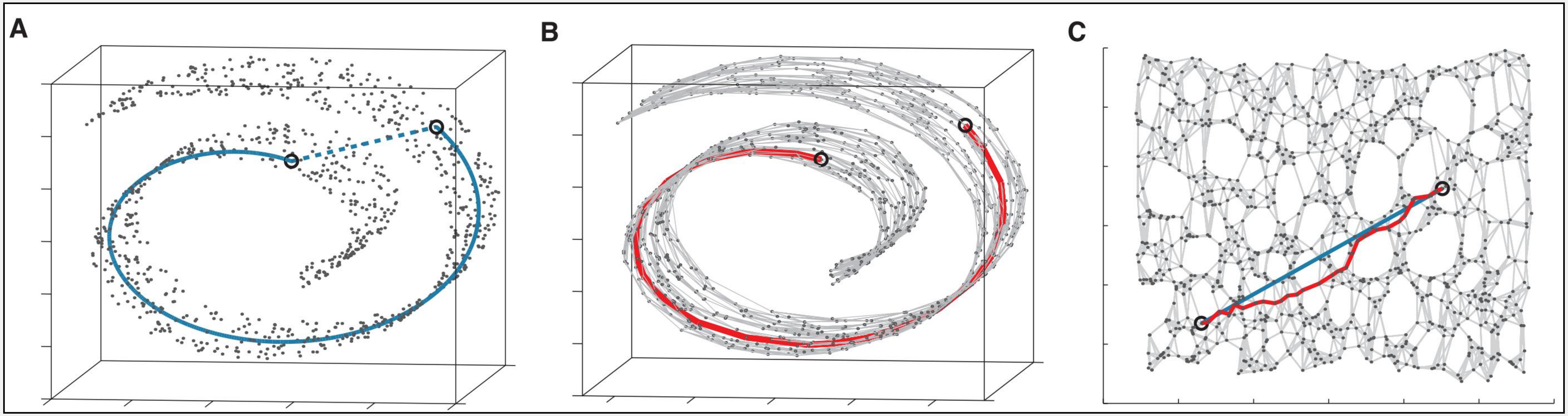
# Geodesic Distances, Euclidean Distances



# Isomap

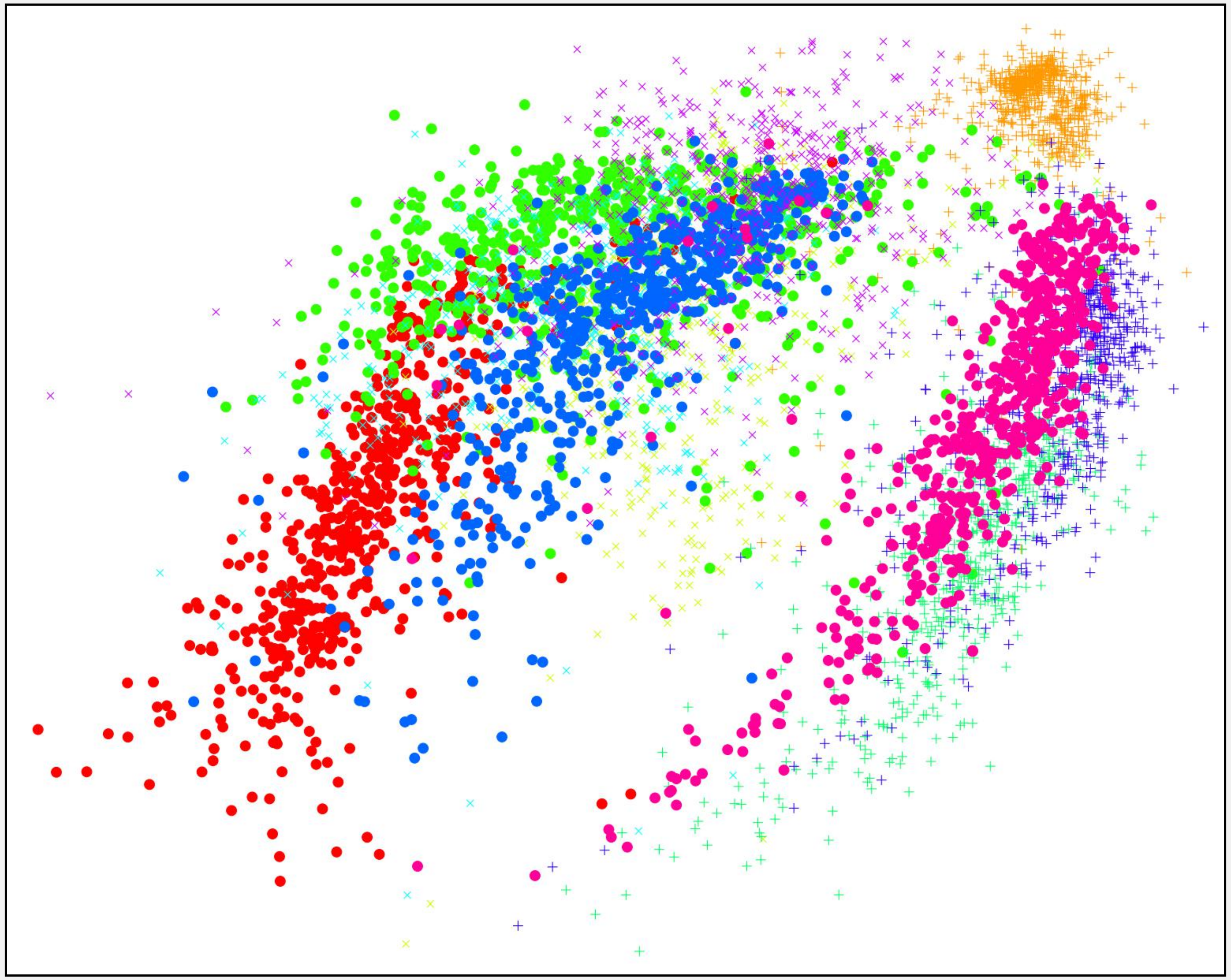
**[Tenenbaum et al. 2000]**

- Multidimensional scaling, where the distance matrix is built from geodesic distances



# Limitations with Nonlinear Methods

- We have little control over the placement of points in 2D. Often results in crowding.



# t-distributed Stochastic Neighbor Embedding

**[der Maaten & Hinton 2008]**

- Objective:

$$C(P,Q) = KL(P||Q) = \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} p_{ij} \ln\left(\frac{p_{ij}}{q_{ij}}\right)$$

- Interpretation: treat each point's relationship to all other points as a probability distribution; minimize the difference between probability distributions of HD and 2D data.

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$$

**Extremely common part of visual analytics techniques**

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma_i)}{\sum_{k \neq i}^{n} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/\sigma_i)}$$

# SNE

- So we've defined the p's. What about the q's?

- Stochastic Neighbor Embedding: we use Gaussian kernels.

$$q_{ij} = \frac{\exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2)}{\sum_{k=1}^{n} \exp(-\|\mathbf{z}_i - \mathbf{z}_k\|^2)}$$

- Optimization: we perform gradient descent on our cost function.

- Most relevant bit: the gradient itself, defined for a single point:

$$\frac{\partial C}{\partial \mathbf{z}_i} = 4 \sum_{j=1}^{n} (p_{ij} - q_{ij})(\mathbf{z}_i - \mathbf{z}_j)$$

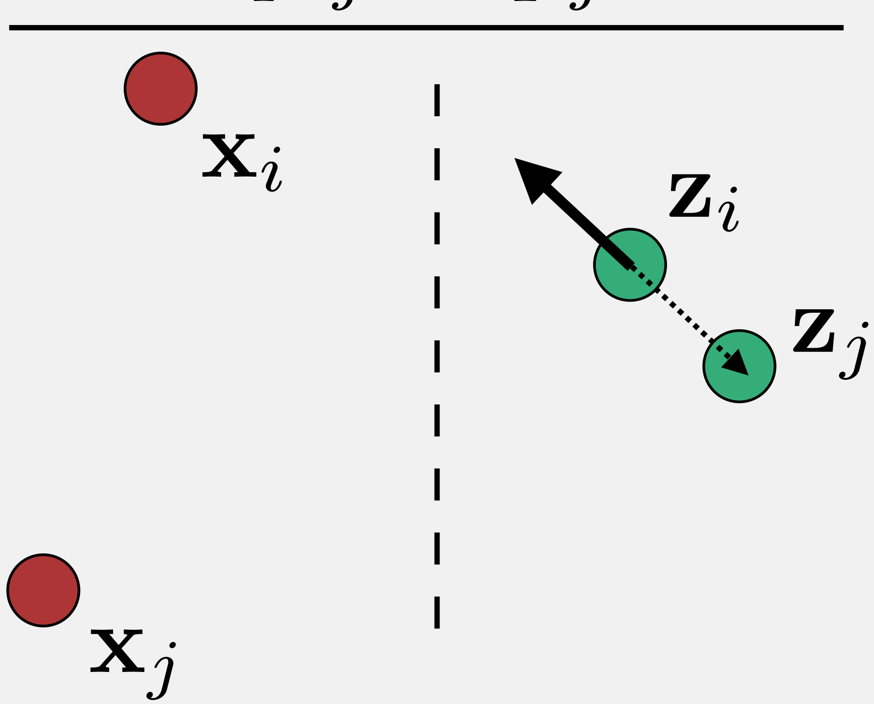
# SNE Optimization

- At each iteration, for each point, we subtract off the gradient:

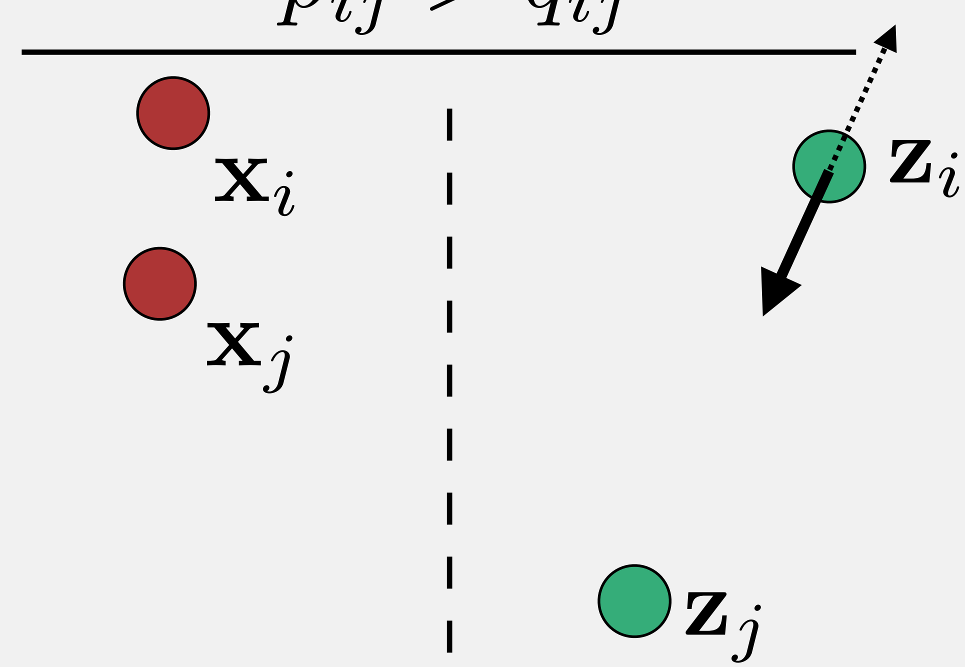
$$\mathbf{z}_i^t = \mathbf{z}_i^{t-1} - \eta \frac{\partial C}{\partial \mathbf{z}_i^t}$$

$$\frac{\partial C_j}{\partial \mathbf{z}_i} = (p_{ij} - q_{ij})(\mathbf{z}_i - \mathbf{z}_j)$$

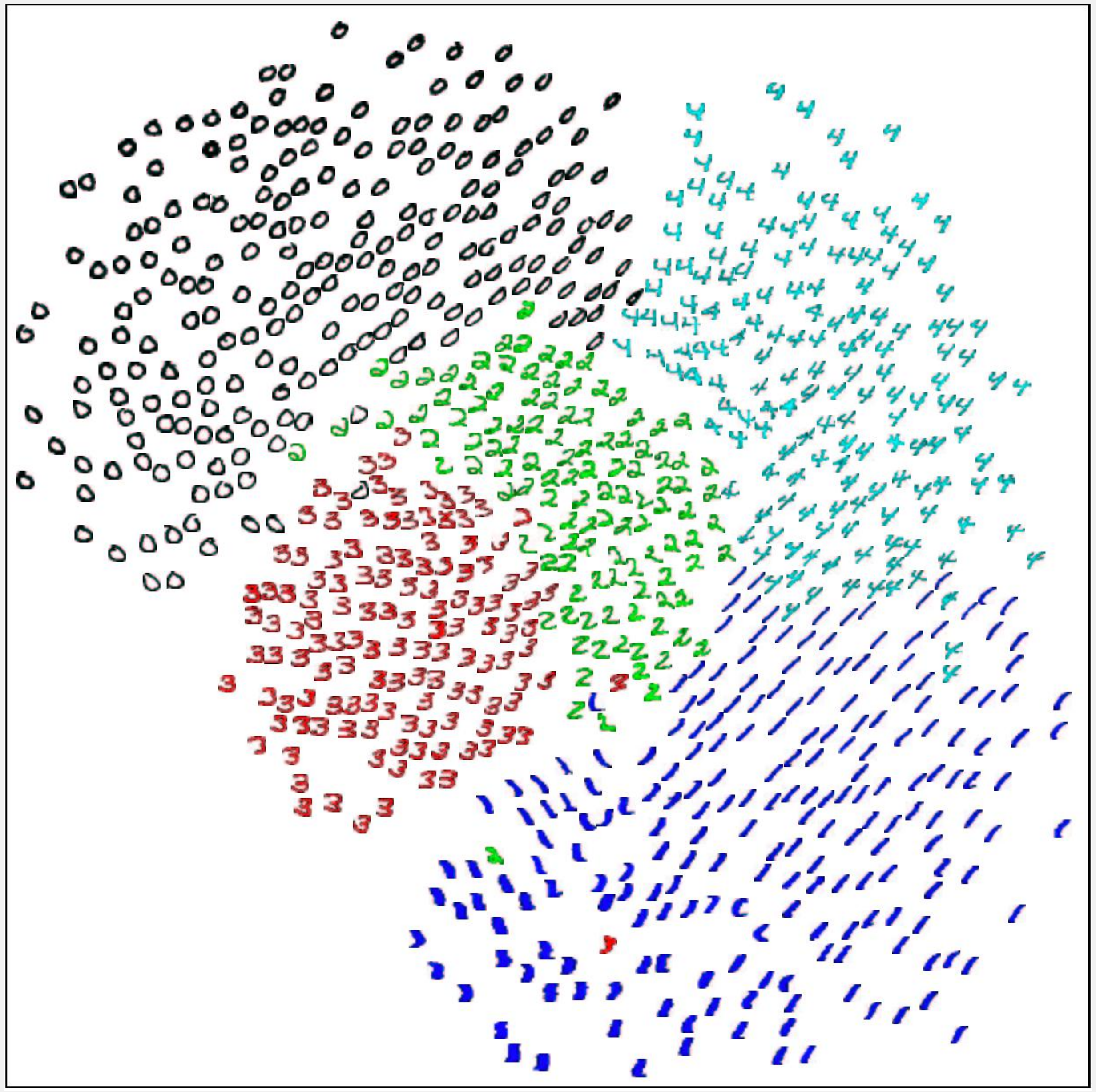
$p_{ij} < q_{ij}$




$p_{ij} > q_{ij}$



# SNE Result



- Ok solution, but the crowding problem still exists.
- How else can we define the q's?

# tSNE

- Student t-distribution: heavier tail than Gaussian, 2D points can more easily spread out:

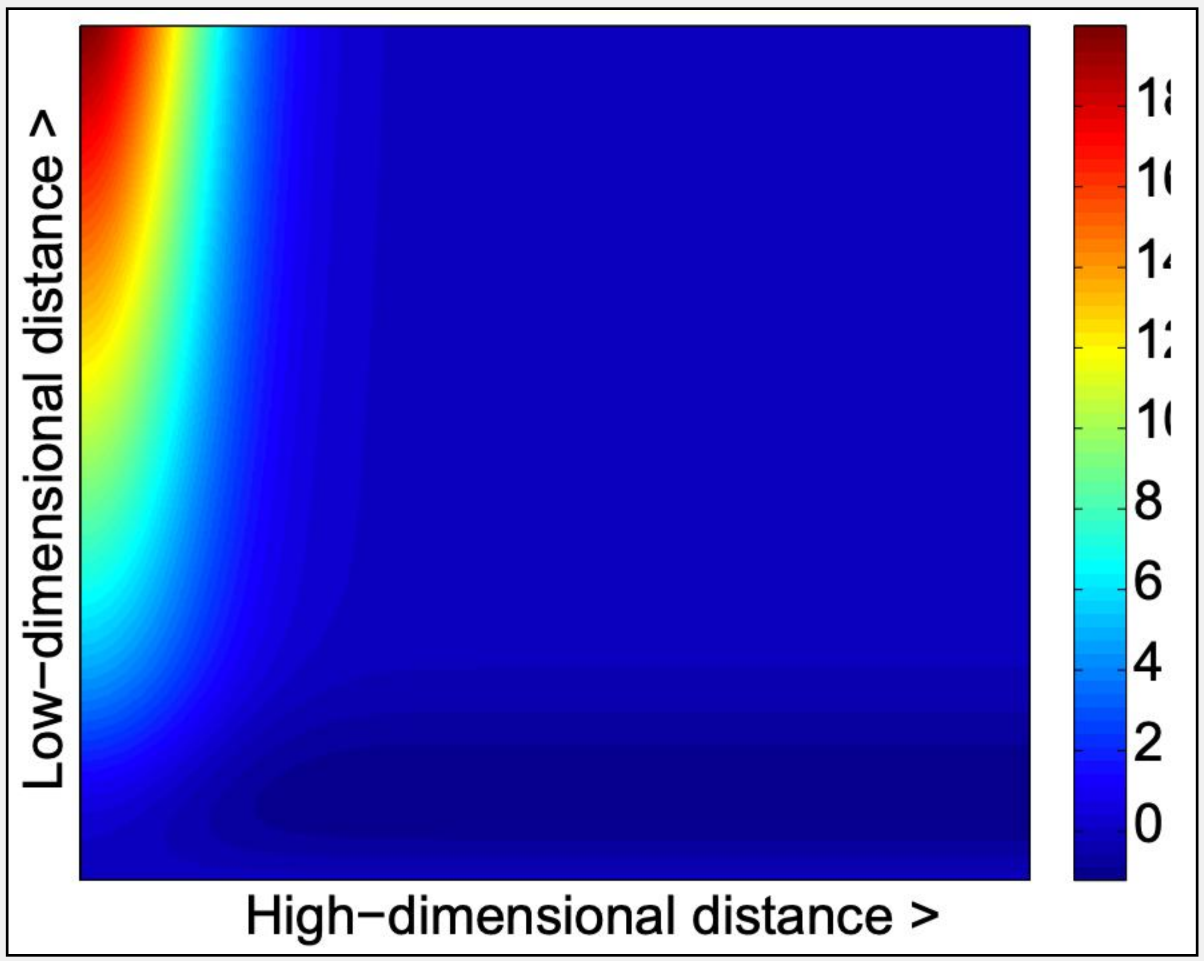
$$q_{ij} = \frac{(1 + \|\mathbf{z}_i - \mathbf{z}_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|\mathbf{z}_k - \mathbf{z}_i\|^2)^{-1}}$$

- Resulting gradients:

$$\frac{\partial C}{\partial \mathbf{z}_i} = 4 \sum_{j=1}^{n} (\underbrace{p_{ij} - q_{ij}}_{\boxed{\textbf{mismatched tails!}}})(\mathbf{z}_i - \mathbf{z}_j)(1 + \|\mathbf{z}_i - \mathbf{z}_j\|^2)^{-1}$$

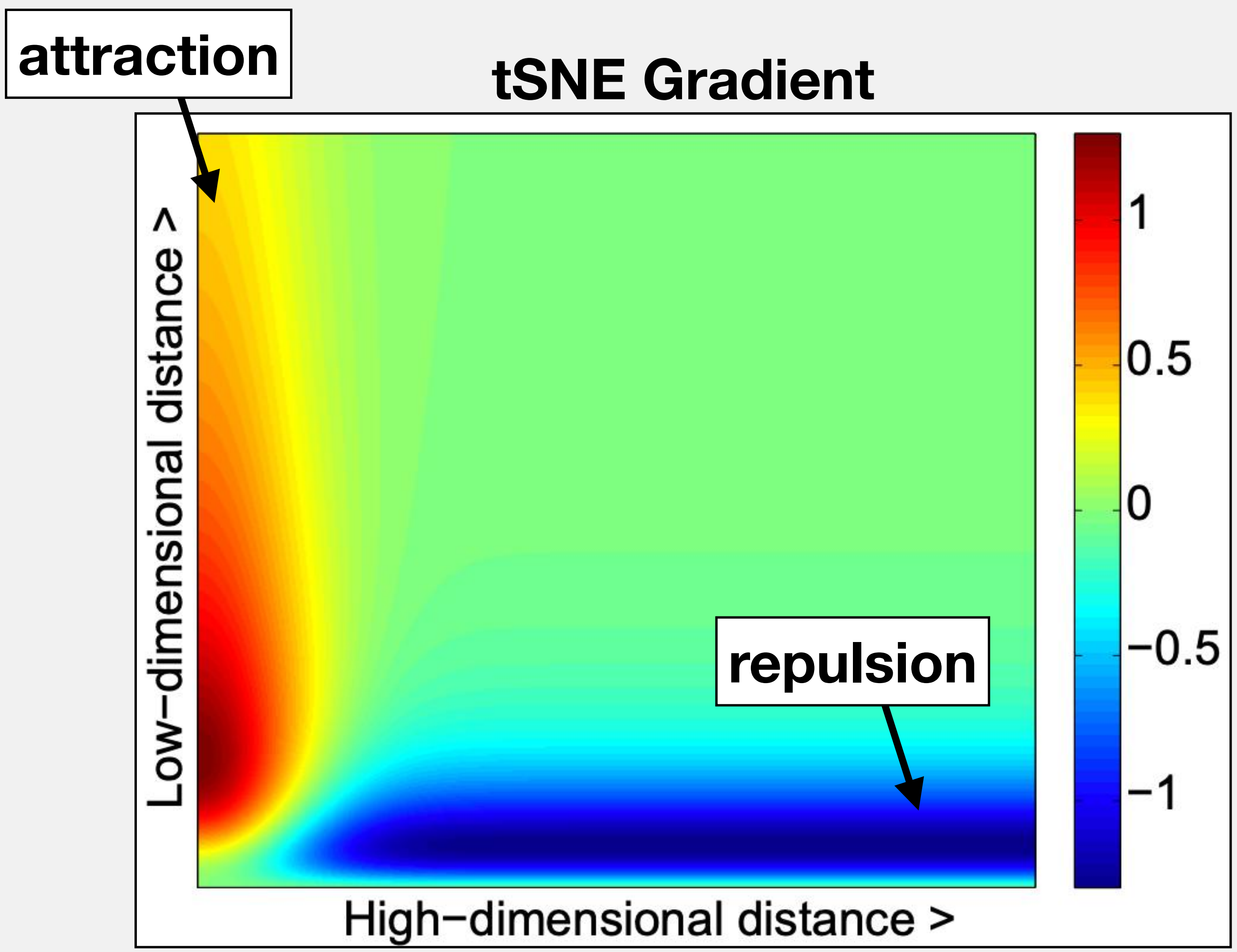
# Gradient Visualizations

SNE Gradient




$$\frac{\partial C_j}{\partial \mathbf{z}_i} = \underline{(p_{ij} - q_{ij})}(\mathbf{z}_i - \mathbf{z}_j)$$

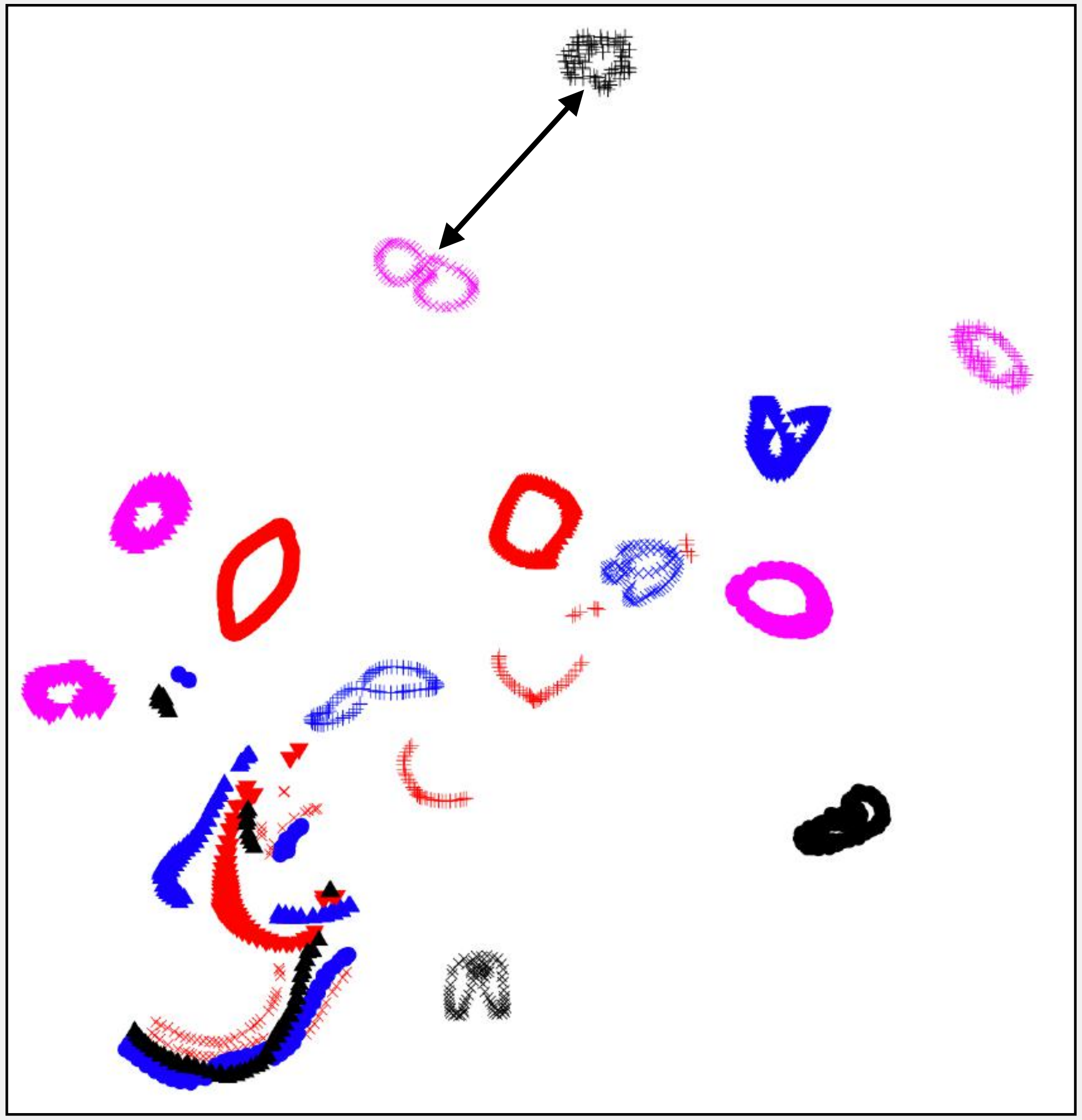
tSNE Gradient



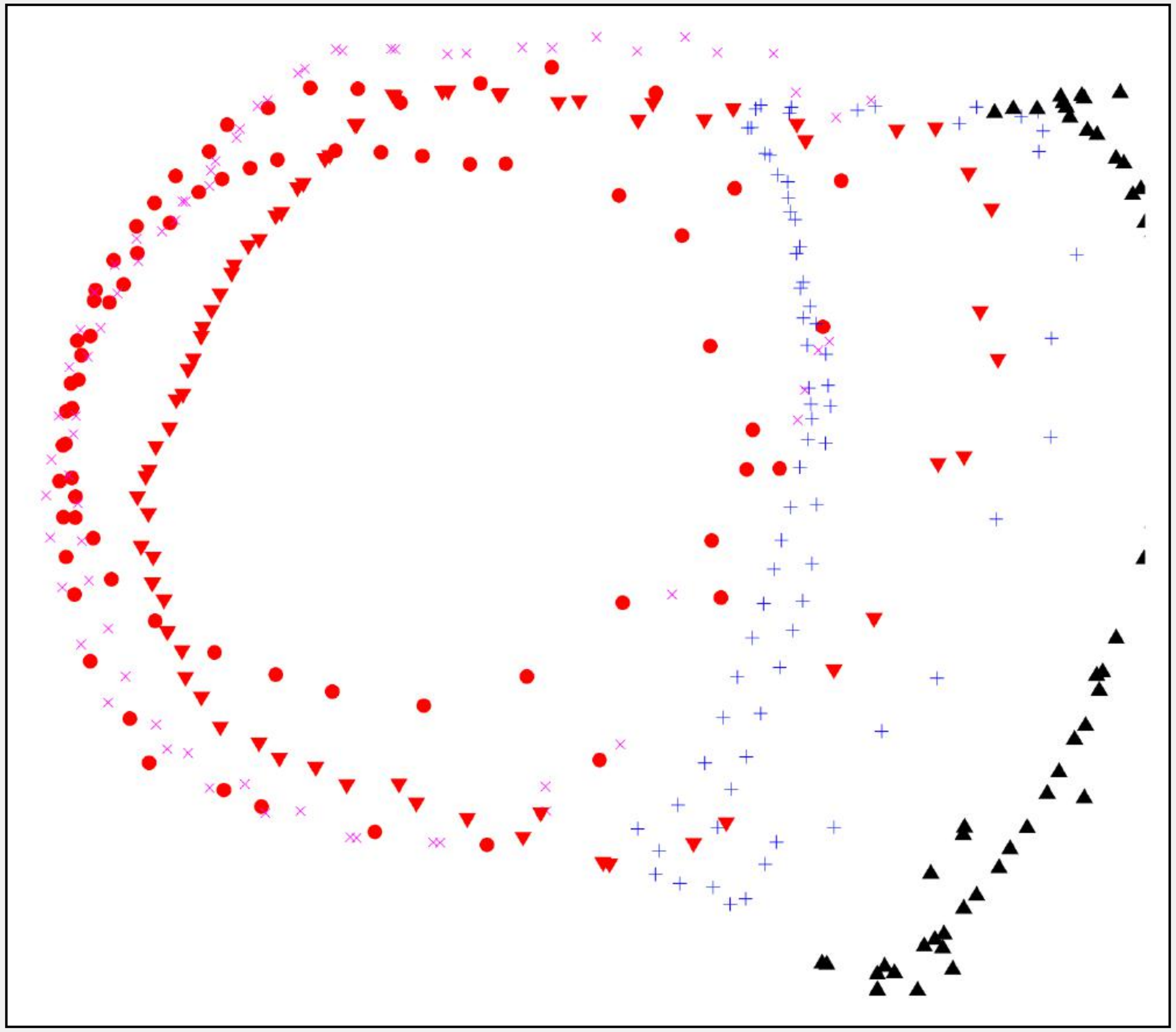

$$\frac{\partial C_j}{\partial \mathbf{z}_i} = \underline{(p_{ij} - q_{ij})}(\mathbf{z}_i - \mathbf{z}_j)\underline{(1 + \|\mathbf{z}_i - \mathbf{z}_j\|^2)^{-1}}$$

# tSNE Comparisons

**tSNE**

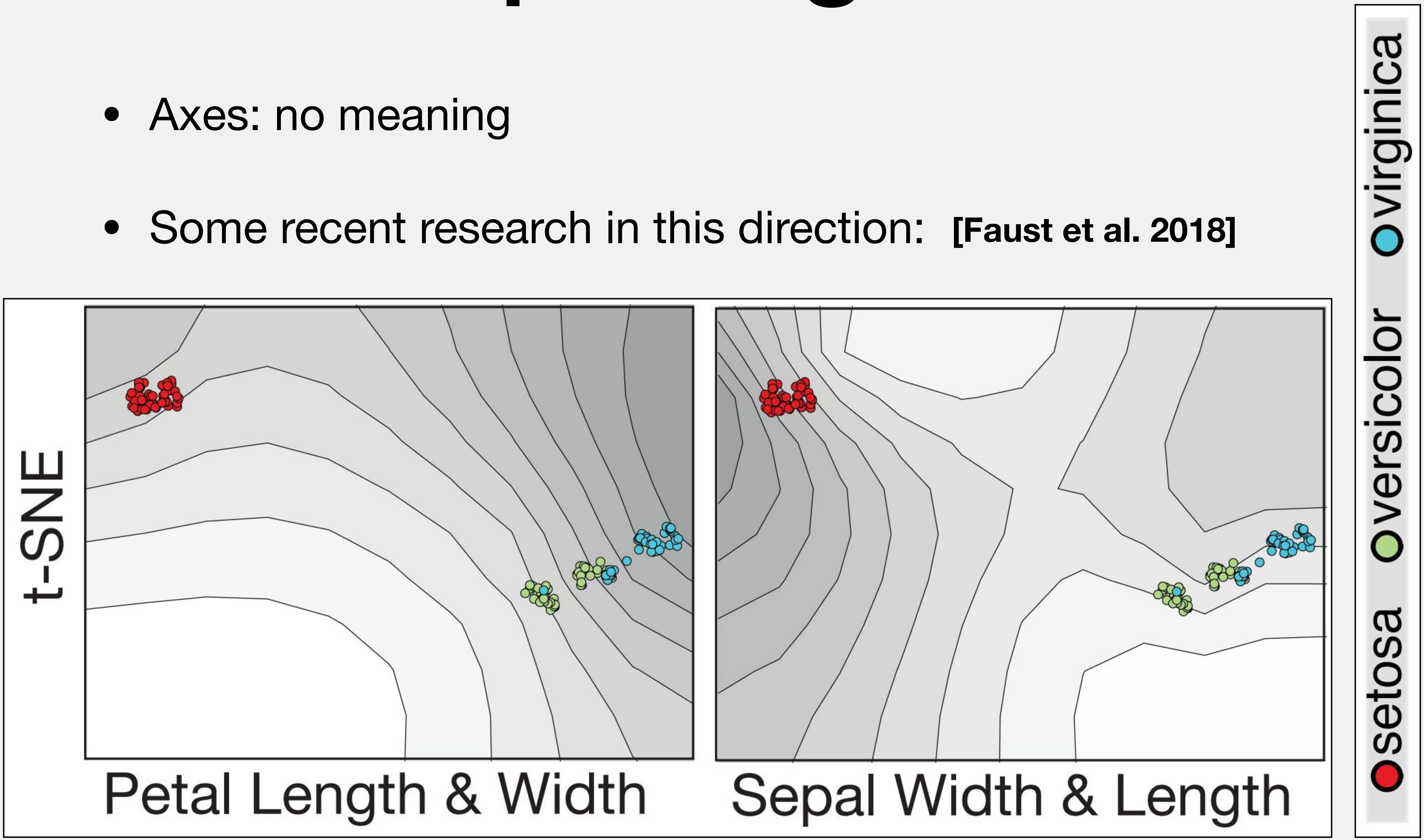


**Isomap**

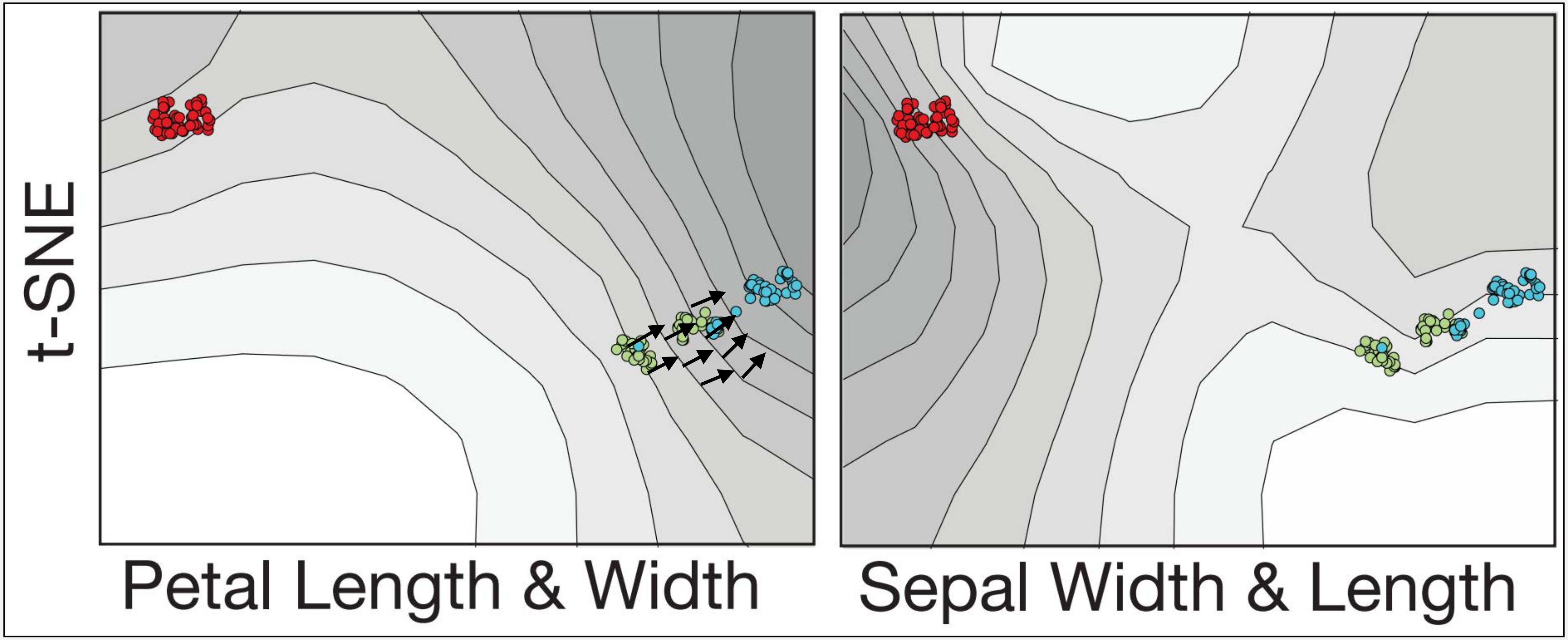


# Interpreting tSNE

- Axes: no meaning
- Some recent research in this direction: **[Faust et al. 2018]**



# Recovering Axes




$\mathbf{v}_i \in \mathbb{R}^2$ **direction of largest change to the projection, with respect to perturbation**

$$\|\nabla f(\mathbf{p}_i) - \mathbf{v}_i\|_2^2 \longrightarrow \mathbf{\Delta f} = \nabla \cdot \mathbf{v}$$

# Interactivity in tSNE

- Vanilla tSNE is slooooowwww: $O(dn^2 + tn^2)$

- One culprit: computing Gaussians of high-dimensional data points

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$$

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma_i)}{\sum_{k\neq i}^{n} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/\sigma_i)}$$

- Another issue: the gradient step involves summing over all points, also quadratic

# Efficiently Evaluating Gaussians

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma_i)}{\sum_{k\neq i}^{n} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/\sigma_i)}$$

- For a given point, this quantity is *really, really small* for most other points - *depending on the bandwidth*.

- Bandwidth: defined based on perplexity:

$$\underbrace{\mu}_{\textbf{user-specified parameter}} = perp(\mathbf{p}_i) = 2^{\underbrace{-\sum_j p_{j|i} \log_2 p_{j|i}}_{\textbf{entropy}}}$$

- If set small enough, leads to **tiny probabilities at most points** - set to zero in practice

- If small, similarity distribution sharply peaked at a couple points

- If large, similarity distribution more spread out

# Accelerating Gradient Step

**[der Maaten 2013]**

- Recall the gradient:

$$\frac{\partial C}{\partial \mathbf{z}_i} = 4\sum_{j=1}^{n}(p_{ij} - q_{ij})(\mathbf{z}_i - \mathbf{z}_j)(1 + \|\mathbf{z}_i - \mathbf{z}_j\|^2)^{-1}$$

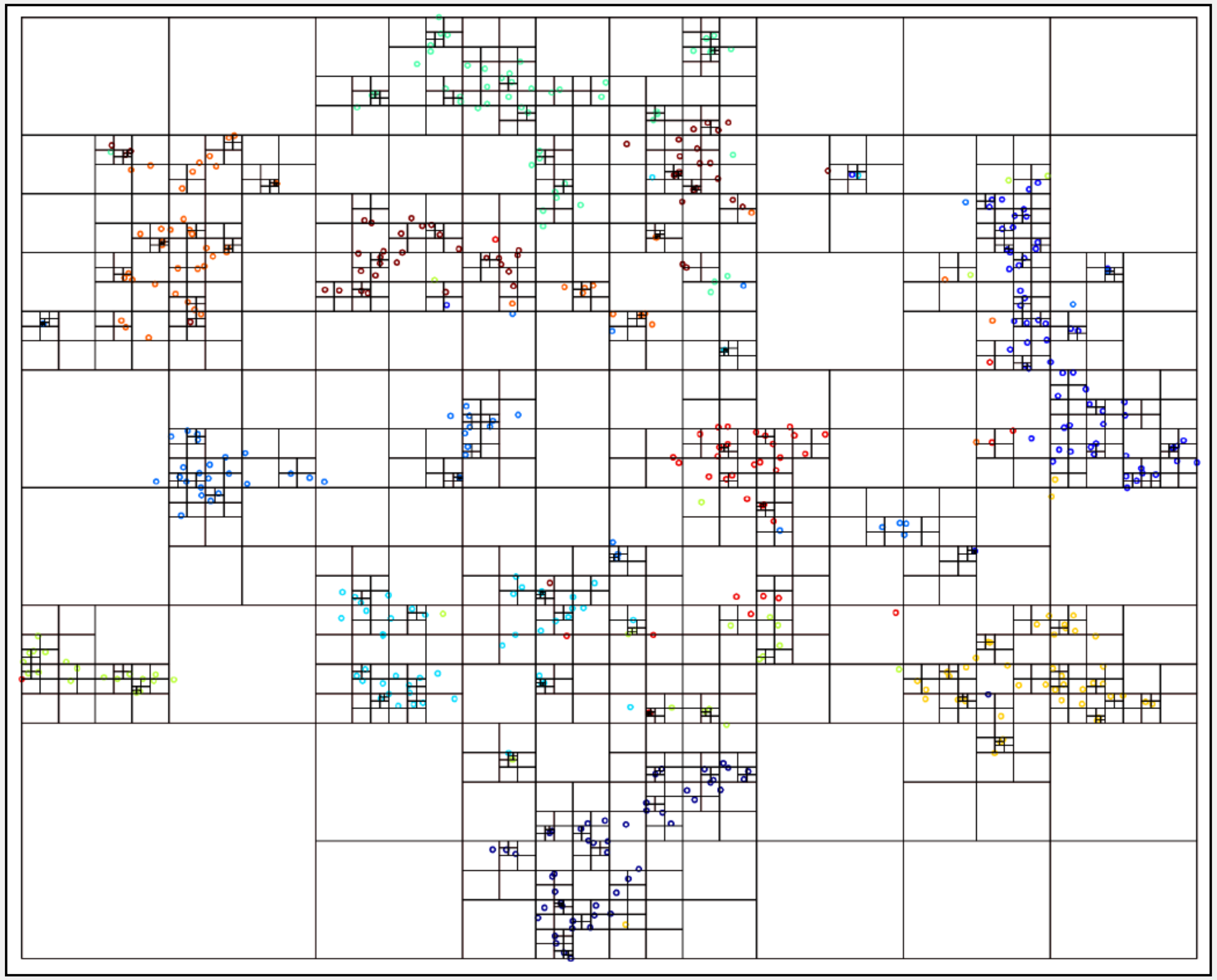
- We may reorganize the gradient as follows:

$$\frac{\partial C}{\partial \mathbf{z}_i} = 4\left(\underbrace{\overbrace{\sum_{j\neq i} p_{ij}q_{ij}Z(\mathbf{z}_i - \mathbf{z}_j)}^{\text{attractive forces}}}_{\text{sparse}} - \underbrace{\overbrace{\sum_{j\neq i} q_{ij}^2 Z(\mathbf{z}_i - \mathbf{z}_j)}^{\text{repulsive forces}}}_{\text{not sparse}}\right) \qquad Z = 4\sum_{k\neq l}(1 + \|\mathbf{z}_k - \mathbf{z}_l\|)^{-1}$$

# Accelerating Gradient Step

**[der Maaten 2013]**

- Barnes-Hut [1986] approximation:



- Algorithm (for a given point):
  - traverse tree top-down
  - if distance to cell center is far, relative to its size, then use an approximate repulsion force, with respect to center

# Still too slow!

- Barnes-Hut tSNE can still take minutes for large (n > 10,000) datasets.
- **Progressive visual analytics**:
  - Need ability to display partial results
  - User steers computation from current view
  - Need to update model, refine computation
  - Repeat

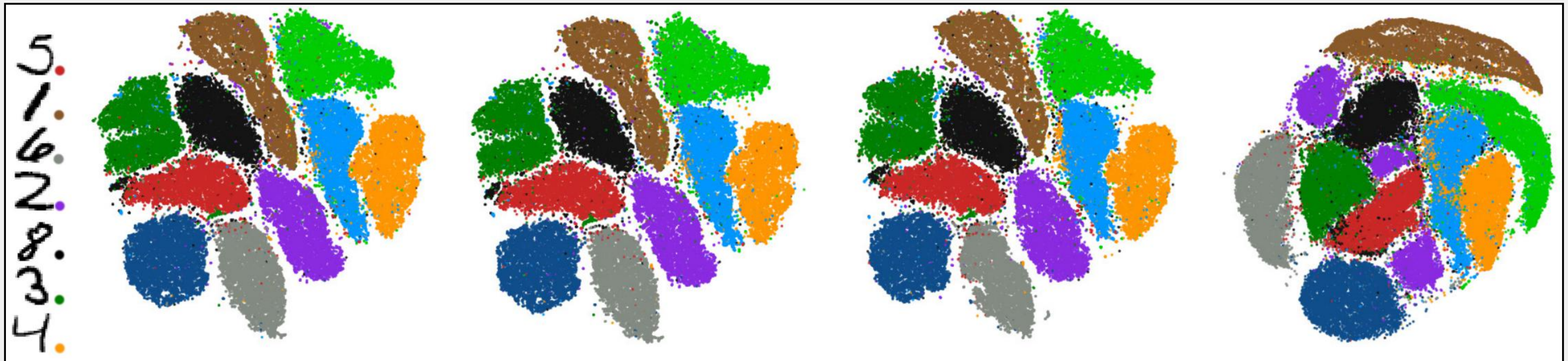
# Approximate tSNE

**[Pezzotti et al. 2016]**

- Replace exact nearest neighbor queries with **approximate queries**.

- Associate a precision with each point, defined as ratio of approximate neighborhood to actual neighborhood:

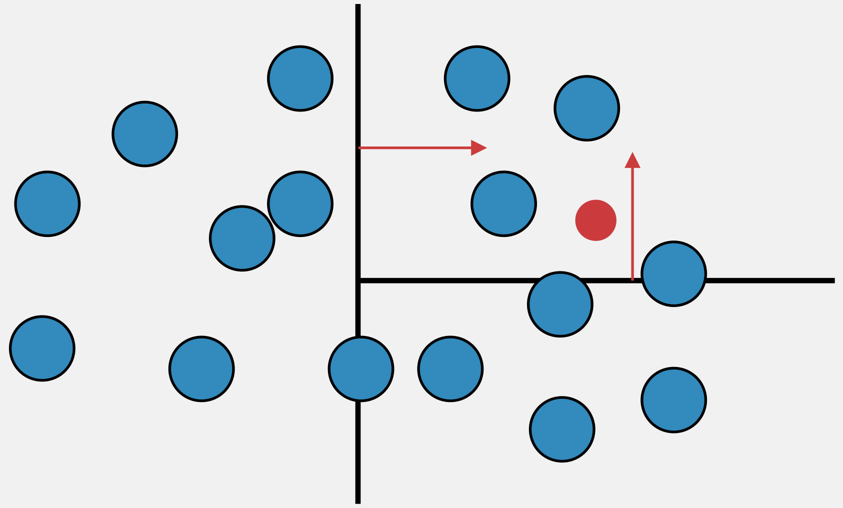
$$\rho_i = \frac{|\mathcal{N}_i^A \cap \mathcal{N}_i|}{|\mathcal{N}_i|} \qquad\qquad \rho = \frac{1}{n}\sum_{i=1}^{n}\rho_i$$

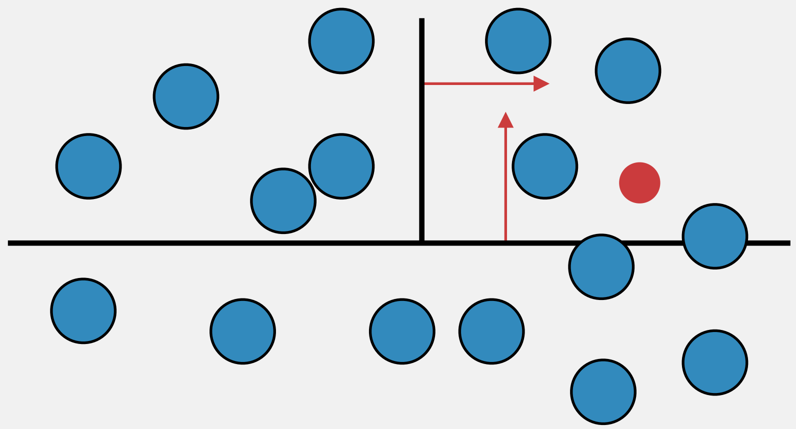



(a) (b) BH-SNE - Time: 3191.8 s (c) $\rho = 0.34$ - Time: 30.1 s (d) $\rho = 0.23$ - Time: 20.4 s (e) $\rho = 0.07$ - Time: 13.0 s

# Randomized kd-tree search

- Precision determines approximation quality of nearest-neighbor search using randomized kd-tree construction:

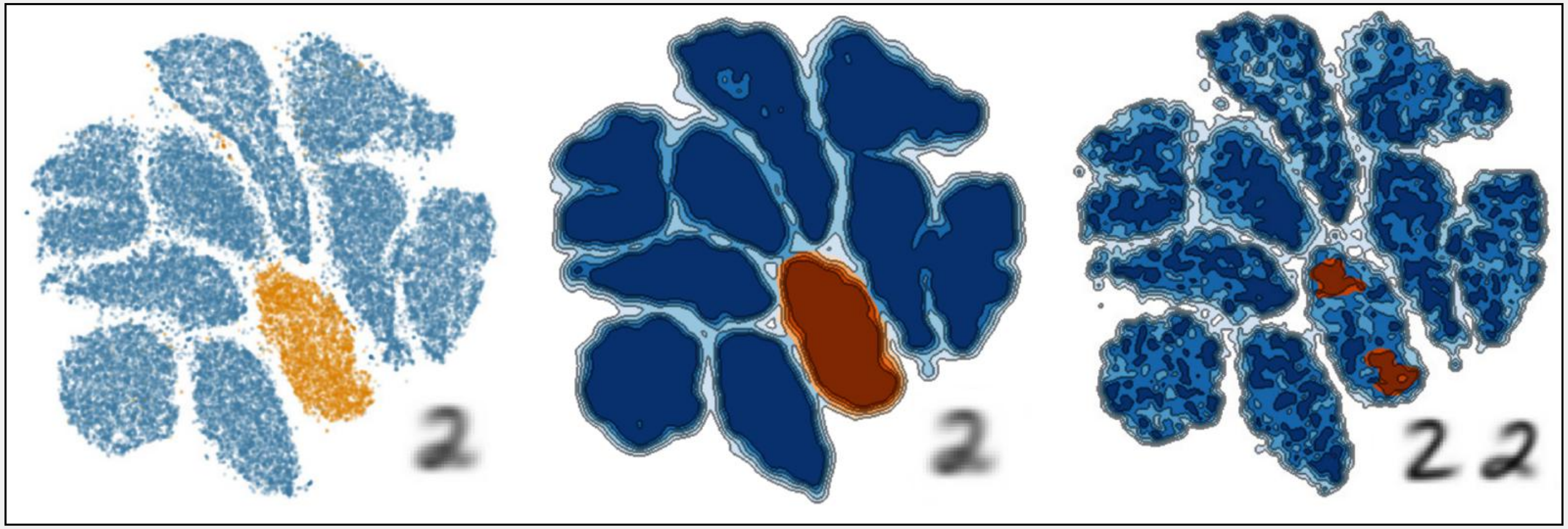




- Precision controls: number of trees, number of splits, number of leaf nodes to traverse when using the set of trees.

# Density-Based Visualization

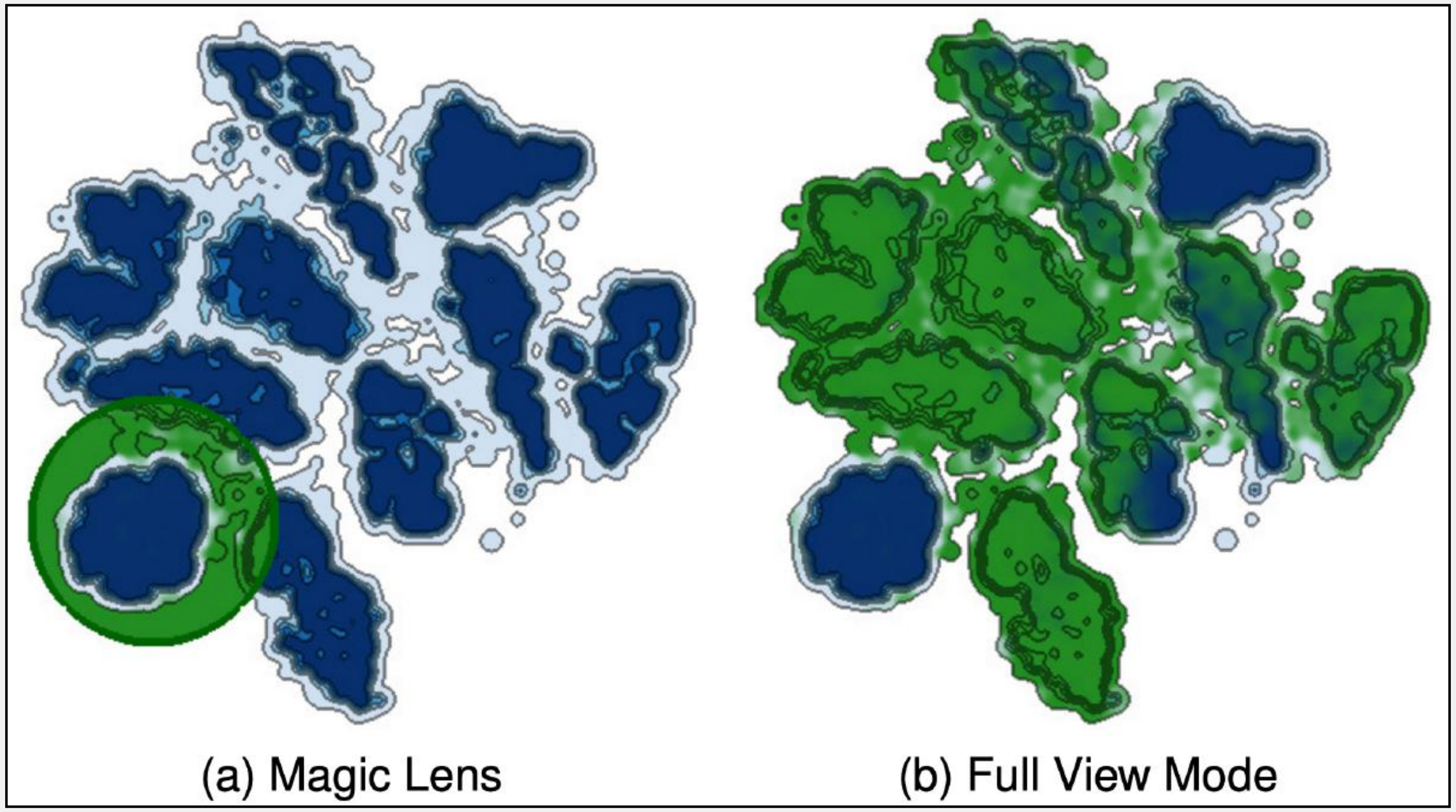
- For large datasets, scatterplots lead to clutter
- Color each point based on kernel density estimation:




$$f(\mathbf{z}, h) = \frac{1}{n} \sum_{i=1}^{n} G(\|\mathbf{z} - \mathbf{z}_i\|, h) \quad G(x, h) = e^{-\frac{x^2}{h^2}}$$

# Steering the Precision

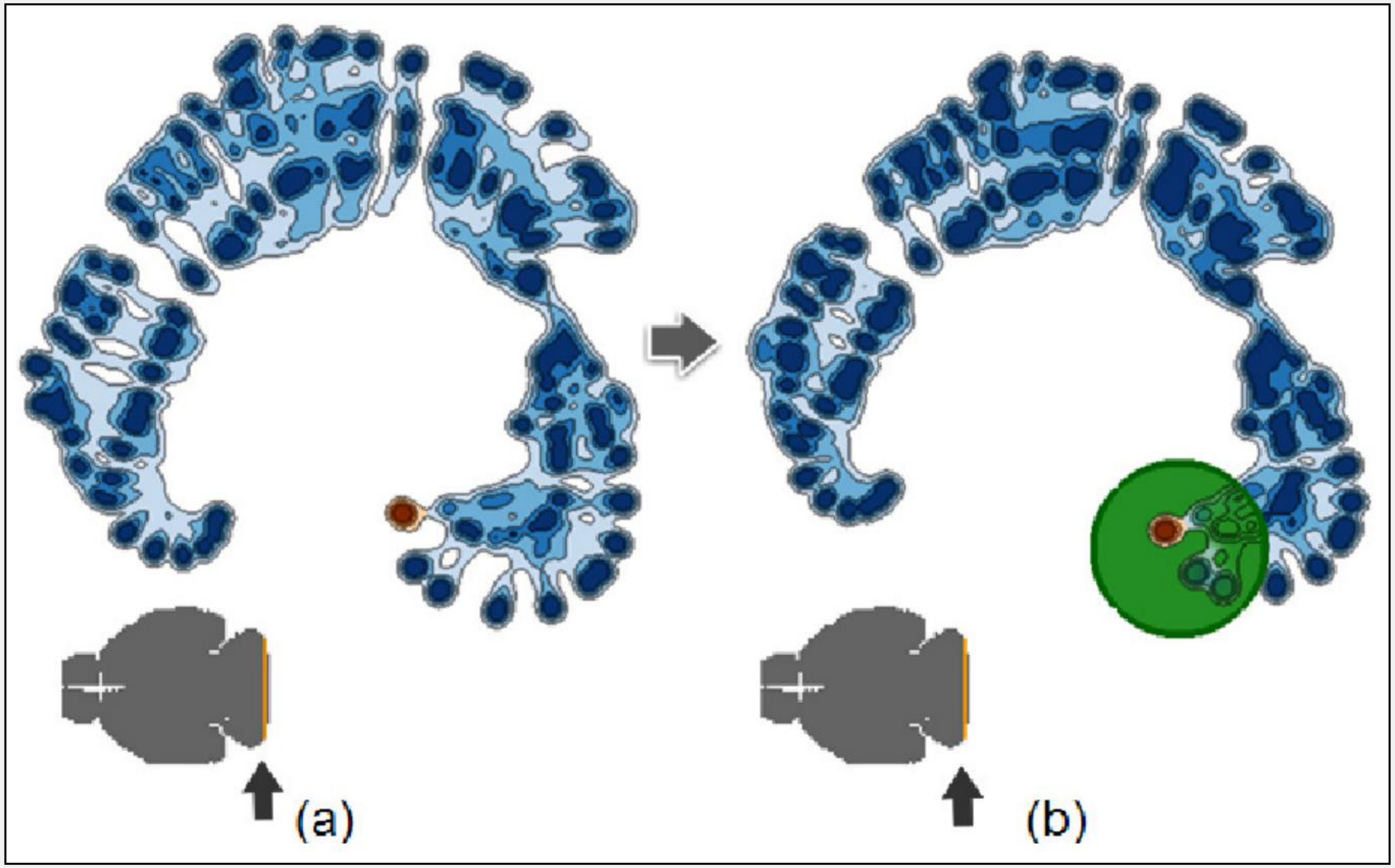
- Users can brush a set of points to prompt **exact neighborhood computations**.
- Subsequently visualize the set of points based on precision of points thus far in computation:



# Case Study

- Gene expression in mouse brain:
  - Volumetric dataset comprised of 61,164 voxels.
  - Each voxel associated with a 4,345-dimensional vector of expressions of different genes
  - If we perform tSNE on just the genes, ignoring spatial positions, can we detect salient structures? E.g. different brain regions?

# Mouse brain with A-tSNE



# Mouse brain with A-tSNE