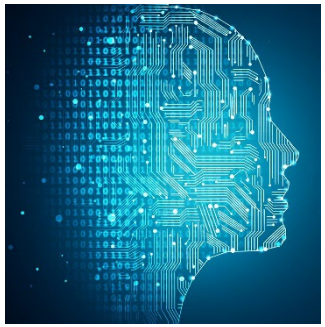
Machine Learning

Manifold Learning

Kevin Moon (kevin.moon@usu.edu)

STAT/CS 5810/6655



Outline

1. Curse of dimensionality
2. The manifold assumption
3. MDS
4. ISOMAP
5. Diffusion Maps
6. MAGIC

Curse of dimensionality

- Data analysis becomes more difficult (statistically and computationally) as the dimension increases
- **Example**: a classification problem where

$$\boldsymbol{X}|Y = 1 \sim \mathcal{N}(\mu_1, I), \qquad \mu_1 = [1,0,\dots,0]^T$$
$$\boldsymbol{X}|Y = -1 \sim \mathcal{N}(\mu_{-1}, I), \qquad \mu_{-1} = [-1,0,\dots,0]^T$$

- Only the first feature is relevant for classification
- As $d \to \infty$, the distance between two random points in the same class has the *same distribution* as the distance between two random points in opposite classes

Curse of dimensionality

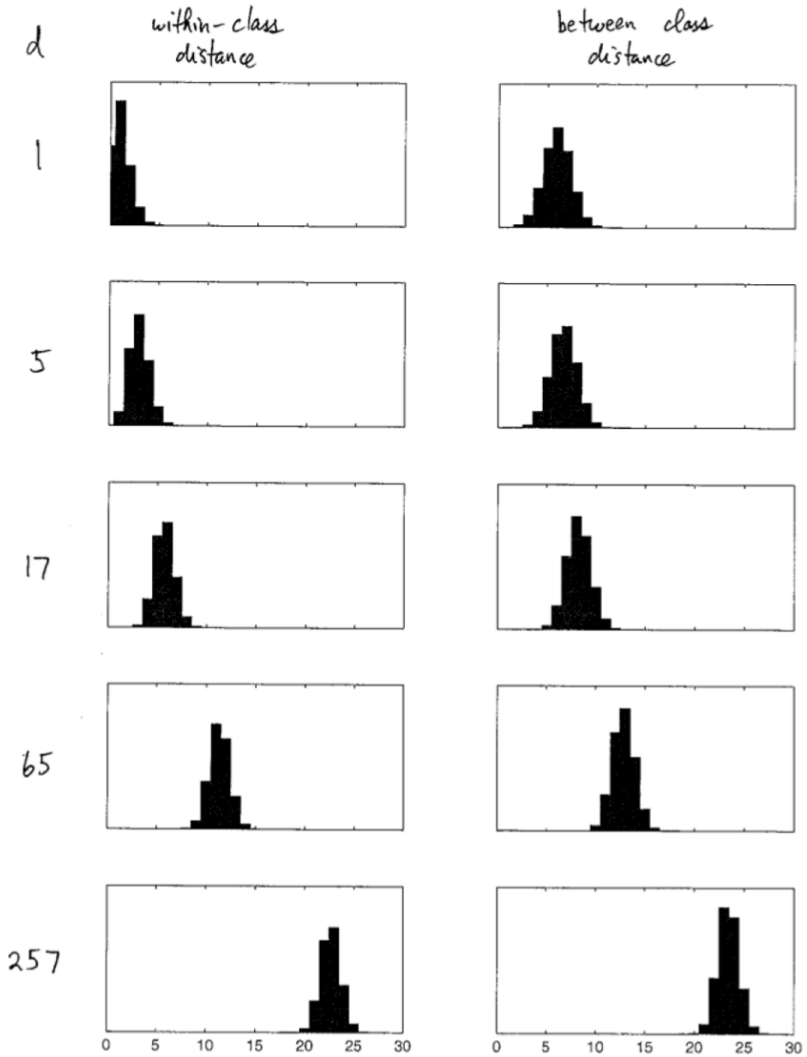


Figure 1: "Within Class" and "Between Class" distances as the dimension increases .

Curse of dimensionality: more intuition

- Simple classification idea: divide input space into cells
- Define classification rule based on training points within a cell
  - Similar to k-nearest neighbors
- What about cells without any training points?
- We could just take the result of the nearest cells, but this could be inaccurate…

Goodfellow et al., 2016

Curse of dimensionality: more intuition

- The number of possible configurations of the input space increases exponentially as the dimension increases
- Thus the number of empty cells increases as the dimension increases

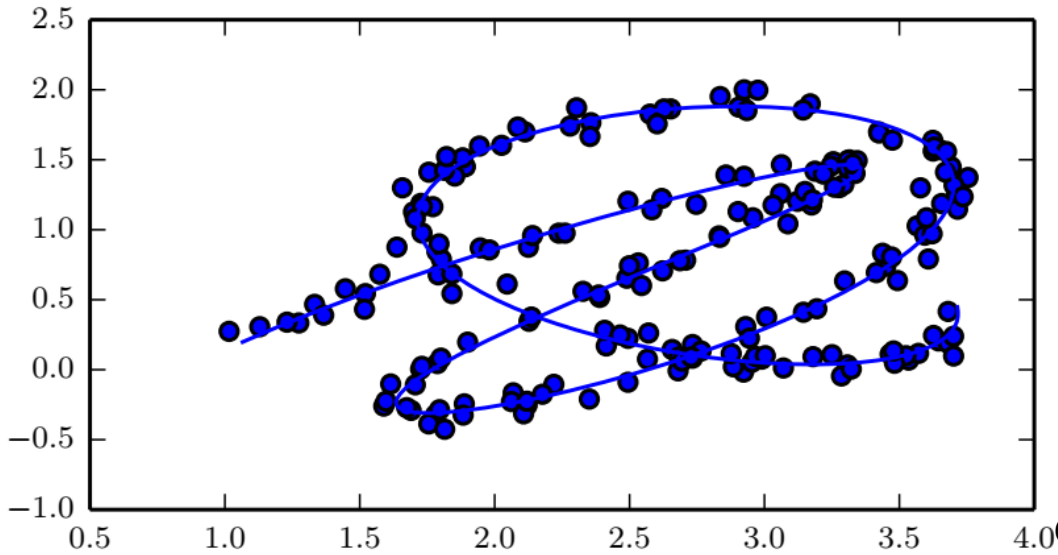
Goodfellow et al., 2016

Representation learning

- Find the “best” representation of the data
  - I.e., find a representation of the data that preserves as much information as possible while obeying some penalty or constraint that simplifies the representation
- How do we define simple?
  - Low-dimensional (mitigates curse of dimensionality)
  - Sparse
  - Independent components
- Above criteria aren’t necessarily mutually exclusive
  - E.g., low-dimensional representations often have independent or weakly dependent components

Manifold learning

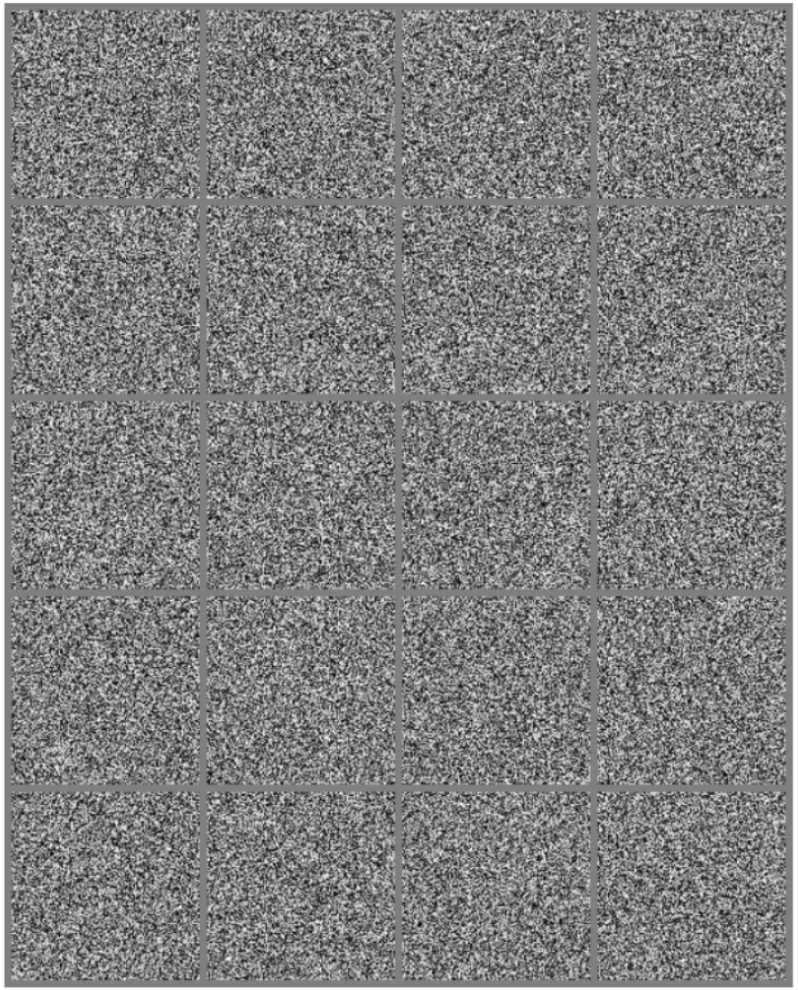
- Informally, a manifold in machine learning loosely means a connected set of points in high dimensions that can be approximated well by a small number of dimensions
  - Sometimes referred to as the **manifold assumption**
  - Simplifies the learning problem greatly and helps with the curse of dimensionality
- Is this a good assumption in machine learning?



Goodfellow et al., 2016

The manifold assumption in practice

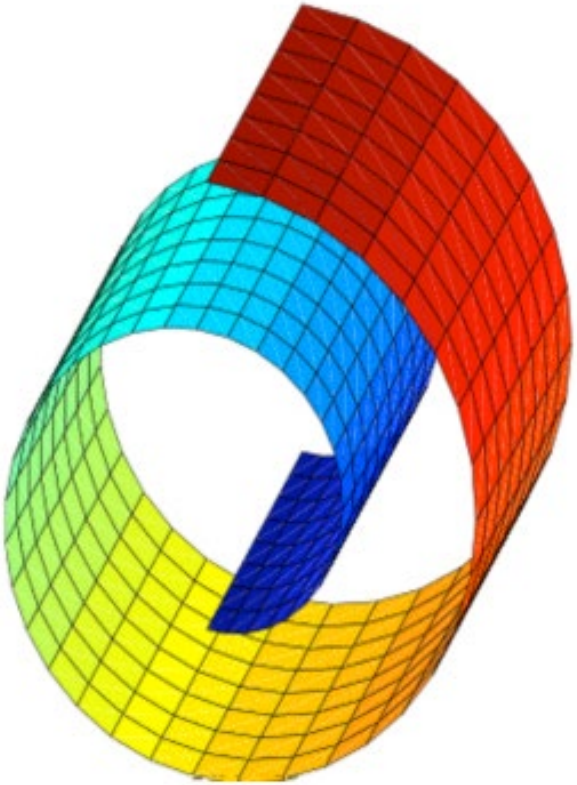
- The probability distribution over images, text strings, and sounds that occur in real life is highly concentrated
- **Example**: randomly generated images (right) rarely (never?) look like real objects
- **Example**: picking random letters unlikely to give real text
- Is this sufficient for the manifold assumption?
- No, also need to establish that the examples we encounter are connected to each other by other examples



The manifold assumption in practice

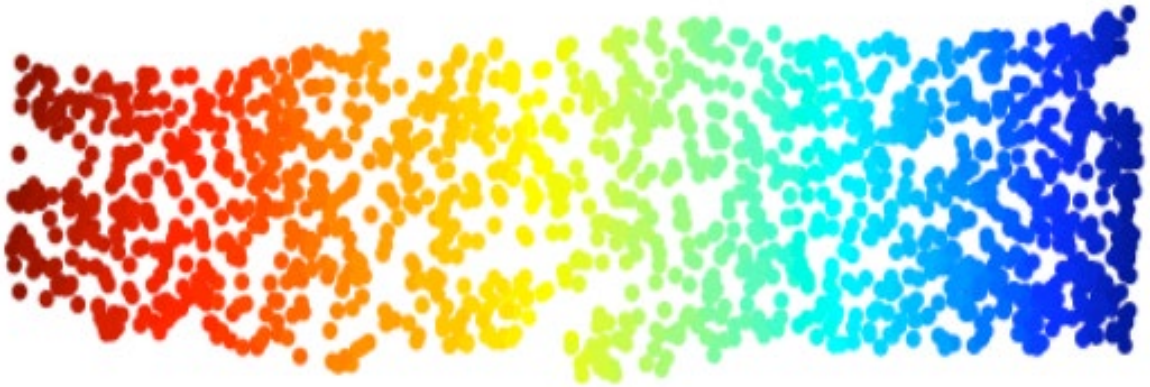
- In images, we can think of many possible transformations that trace out a manifold
  - Dim or brighten the lights, gradually move or rotate objects, etc.
- Multiple manifolds may be involved (e.g. human manifolds and cat manifolds)

Manifold learning



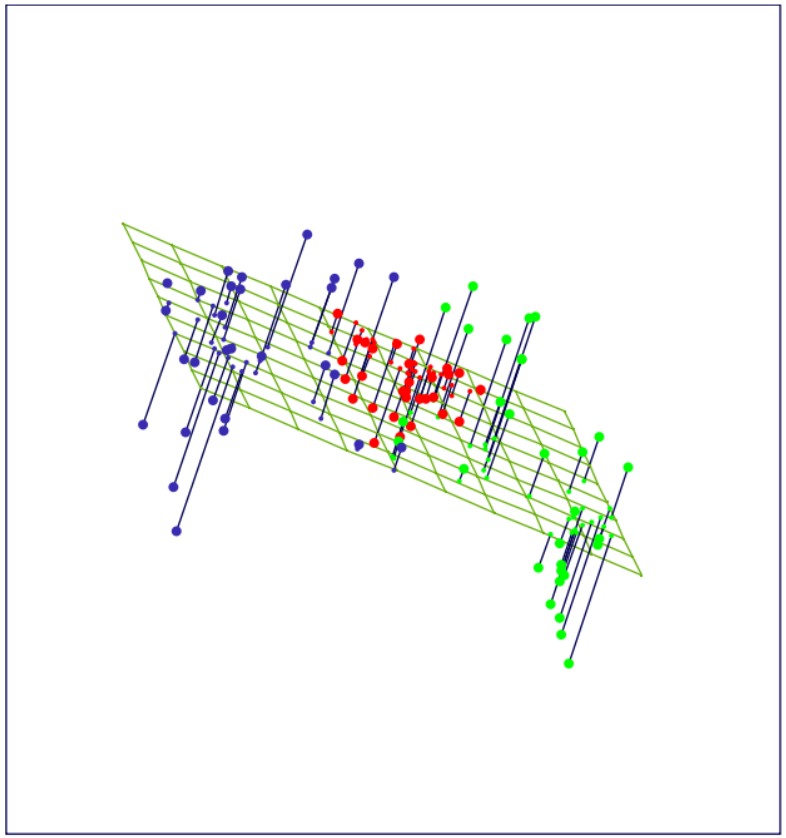
Manifold learning

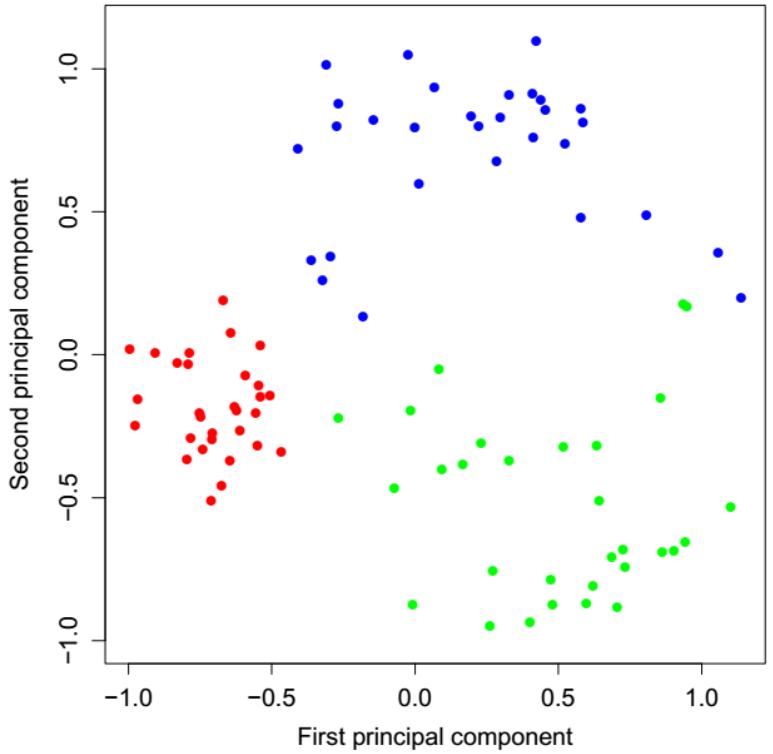
Manifold learning



PCA

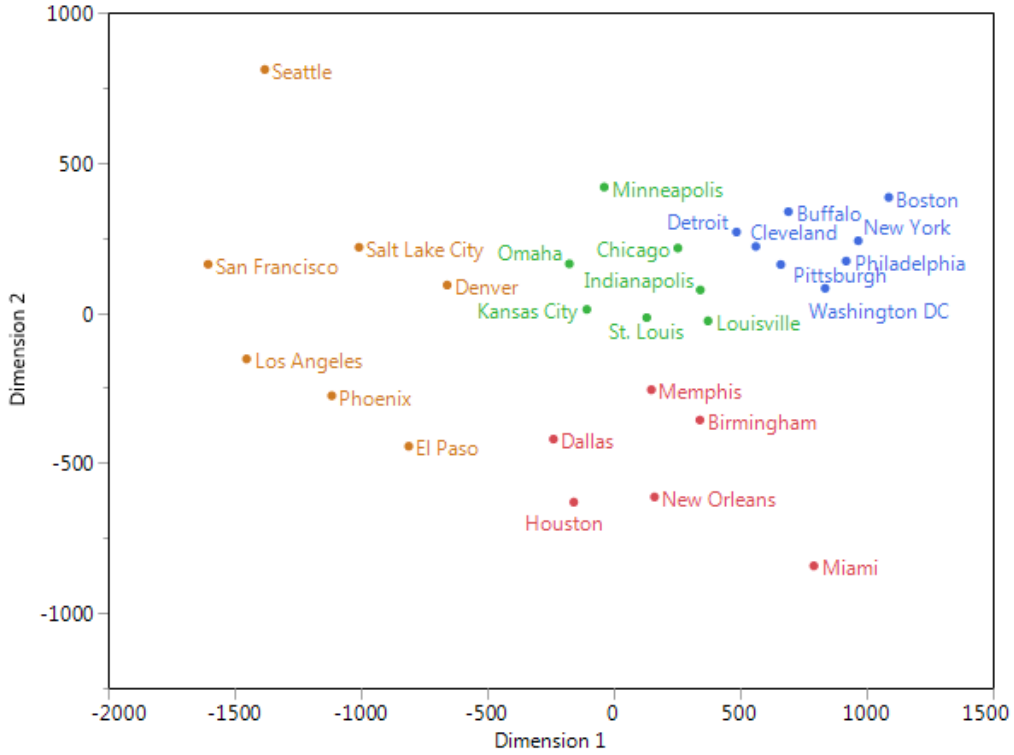
- PCA is a manifold learning method
- PCA assumes linear manifolds
  - Very restrictive as many datasets are not linear





MDS Review

- MDS is a class of dimensionality reduction methods
- **Goal**: minimize the "difference" between corresponding distances in the high and low dimensional spaces
- **Example**: projecting cities on a 3D Earth to a 2D map



MDS Review

- Different measures of "difference" give different algorithms
- Let $D^{(2)}$ be a matrix of squared pairwise distances between n points in the high-dimensional space
- Let $J = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$
  - $I =$ identity matrix, $\mathbf{1} = n$-dimensional vector of ones
- Set $B = -\frac{1}{2}JD^{(2)}J$
- Classical MDS (CMDS) minimizes the following:

$$Strain(\hat{\boldsymbol{x}}_1, \dots, \hat{\boldsymbol{x}}_n) = \sqrt{\frac{\sum_{i,j}\left(B_{ij} - \langle \hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_j \rangle\right)^2}{\sum_{i,j} B_{ij}^2}}$$

  - The $\hat{\boldsymbol{x}}_i$ are the low-dimensional coordinates
  - Can be solved using an eigendecomposition of B

Metric MDS Review

- Let D be a matrix of pairwise distances between n points in the high-dimensional space
- Metric MDS has a different "stress" function to minimize:

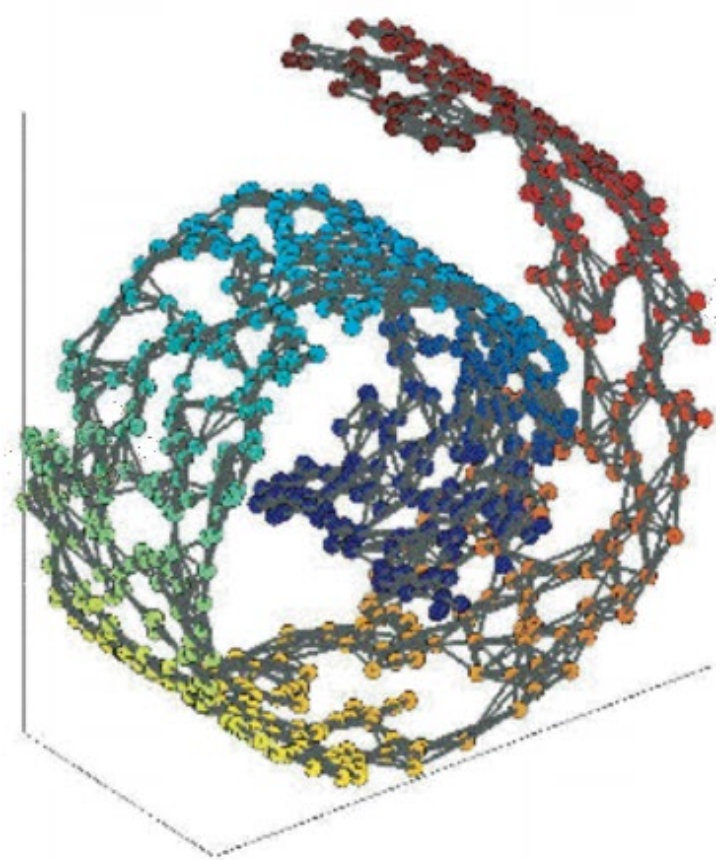
$$Stress(\hat{\boldsymbol{x}}_1, \dots, \hat{\boldsymbol{x}}_n) = \sqrt{\frac{\sum_{i,j}\left(D_{ij} - \left\|\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j\right\|\right)^2}{\sum_{i,j} D_{ij}^2}}$$

- Solving this requires an iterative approach (like gradient descent)
- Generally gives better visualization than CMDS for PHATE
- Nonmetric MDS also exists
  - Input "distances" do not need to be a true distance

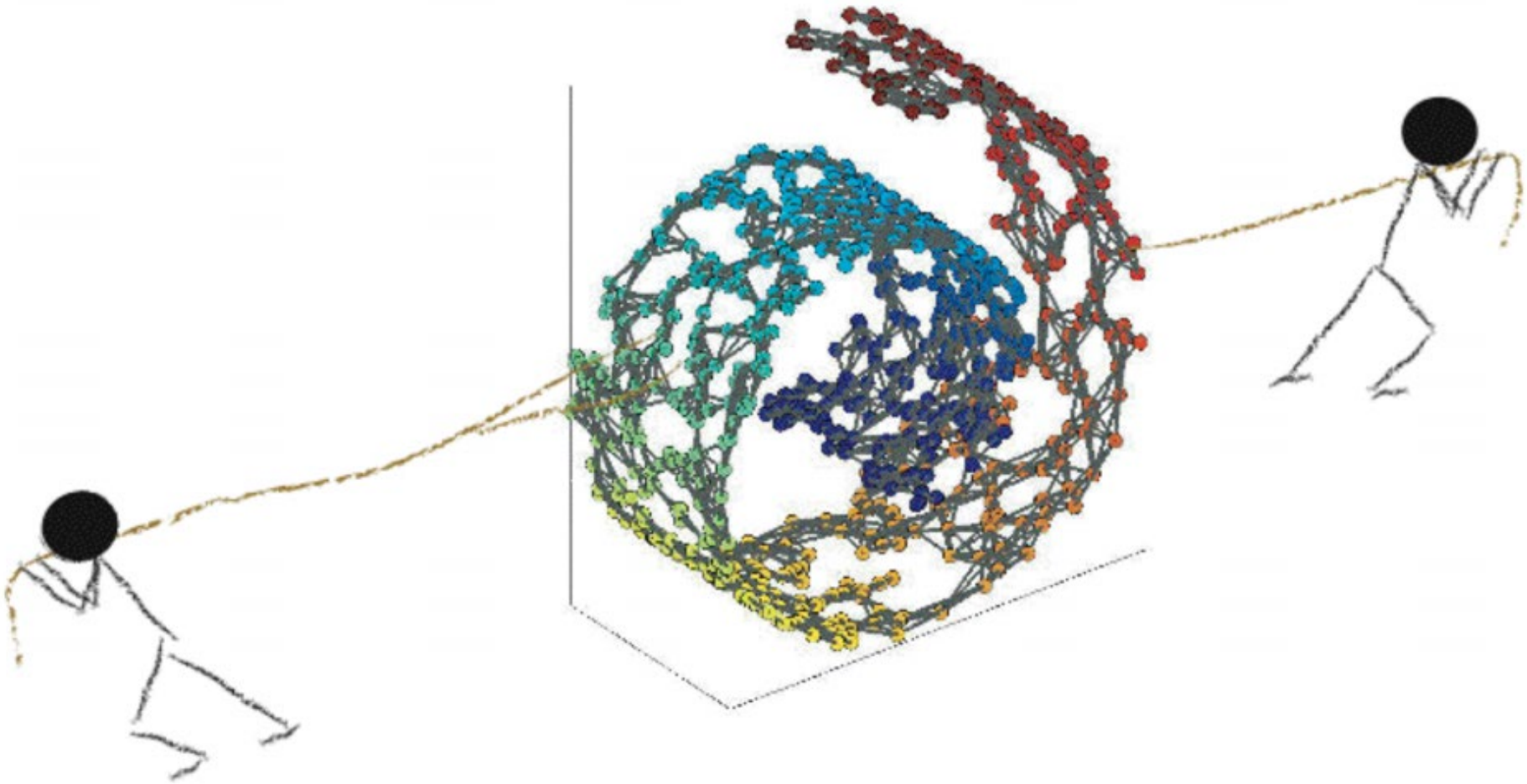
Manifold Learning

- Large Euclidean distances often uninformative
  - Exit the manifold
  - Also likely to suffer from the curse of dimensionality
- Thus MDS on Euclidean distances often isn't great
- **Solution**: input "better" distances
- We'll construct distances from a graph

Manifold learning



Manifold learning



Similarity Graphs

- Unlabeled data to be analyzed: $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$
- Assume relationships between data points are captured entirely by a *similarity graph*
- Similarity graphs are defined by an *affinity matrix*

$$W = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nn} \end{bmatrix}$$

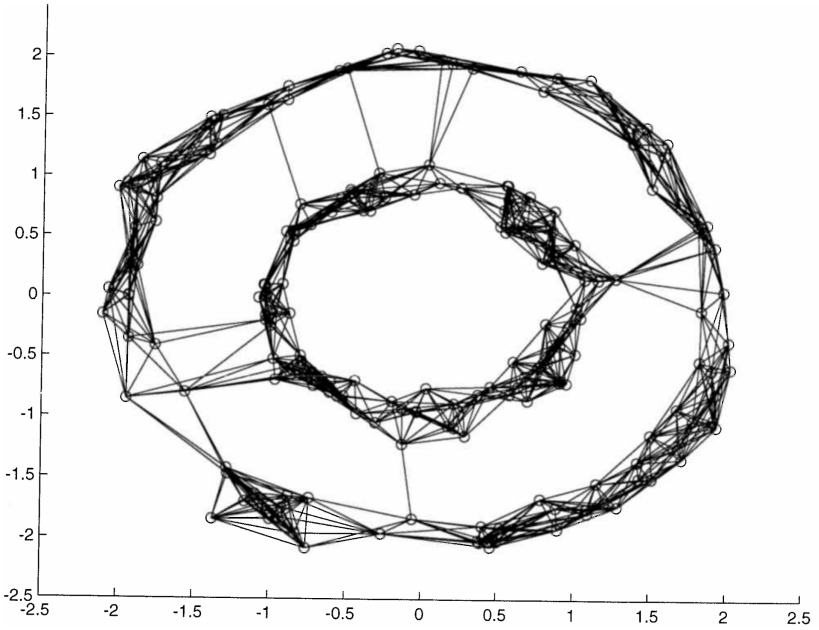
- Weights are nonnegative, symmetric:

$$w_{ij} \geq 0, \qquad w_{ij} = w_{ji}$$

- Associated graph: Nodes i and j are connected iff $w_{ij} > 0$
  - w_{ij} are the edge weights

Similarity Graphs

- Similarity graphs are defined by two things
- **Graph structure** (which edges are connected)
  - Examples
    - k-nn graph
    - ϵ —ball graph
    - Complete graph



11-nn graph

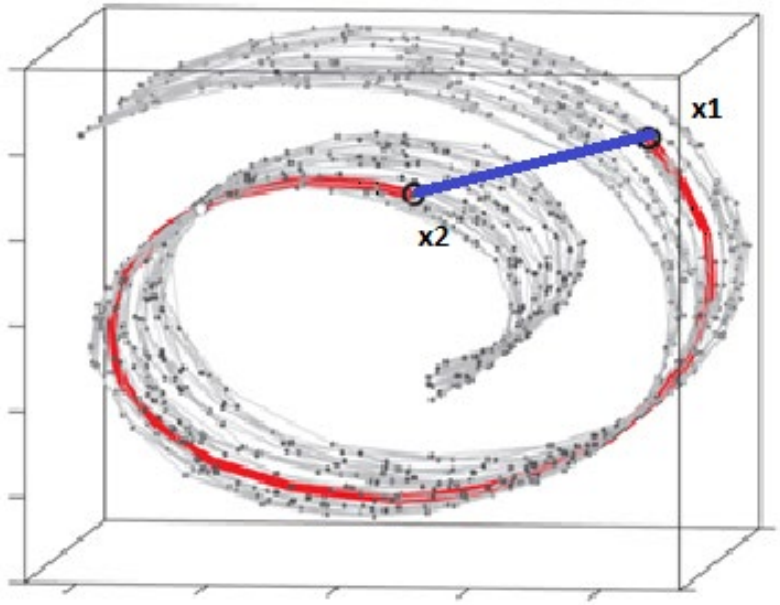
- **Weights**
  - Examples

$$w_{ij} = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right)$$

$$w_{ij} = \begin{cases} 1 & \text{if } \exists \text{ an edge between nodes } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

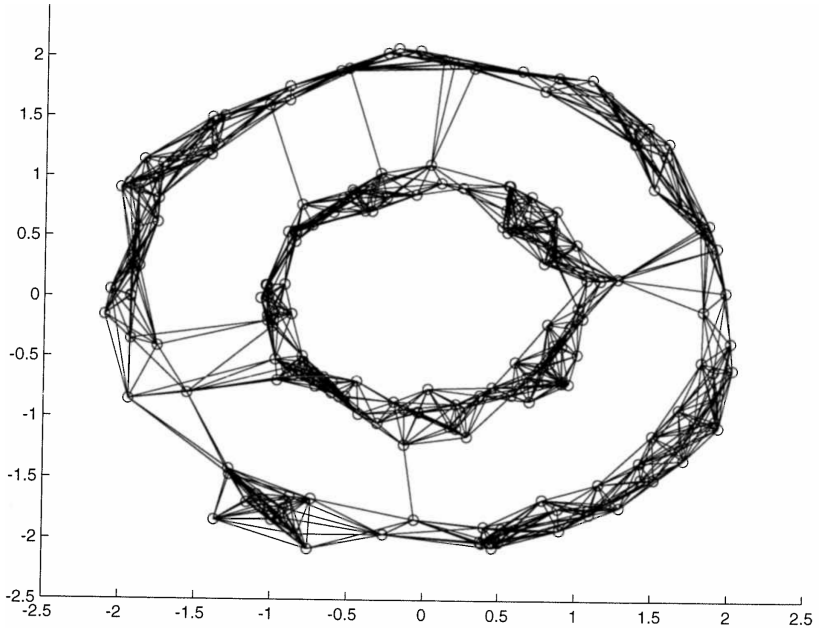
ISOMAP

1. Construct a neighborhood graph on the data
   - E.g. k-nn or ϵ-ball graph
2. Compute the shortest path along the graph between all pairs of points
   - The length of these paths is the geodesic distance
3. Embed the distances into low-dimensions using MDS



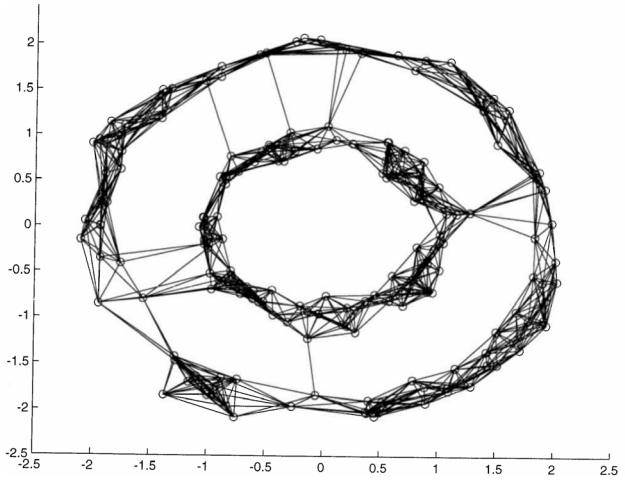
ISOMAP

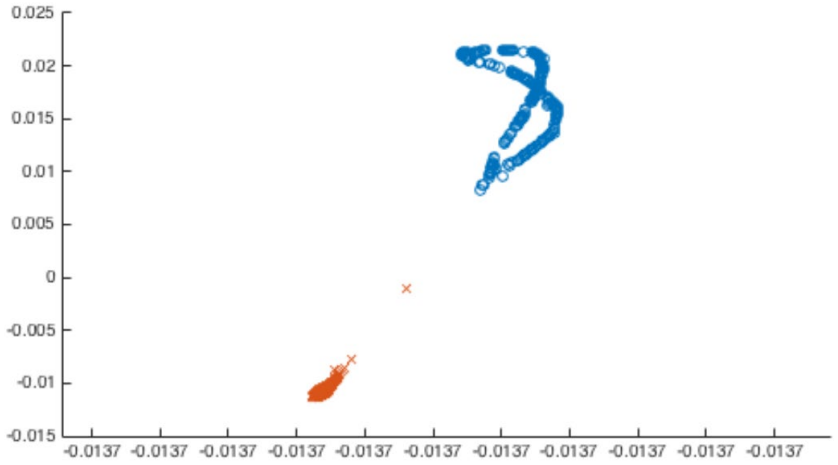
- **Weakness**: ISOMAP is sensitive to spurious connections
  - Noise or too large of k may lead to exits from the manifold



Laplacian Eigenmaps

- Direct embedding of the graph structure
- Based on an eigendecomposition of the (normalized) graph Laplacian
- Works for clustering, but may not preserve local structure well

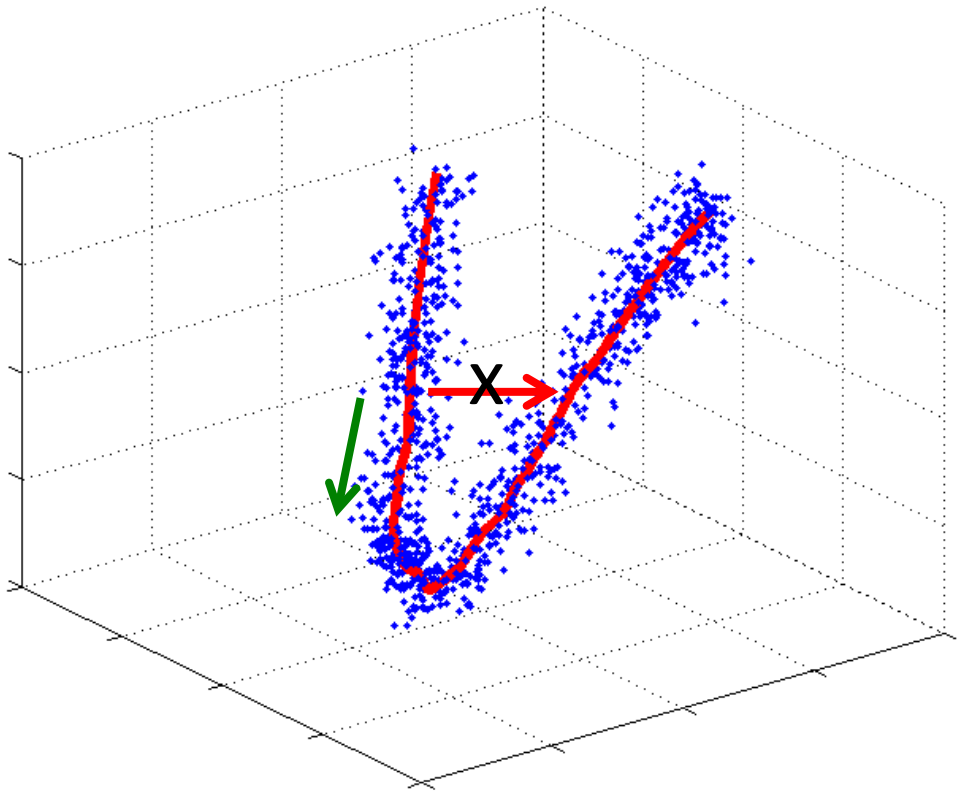




Diffusion Maps

More robust than ISOMAP

Capturing local neighborhoods



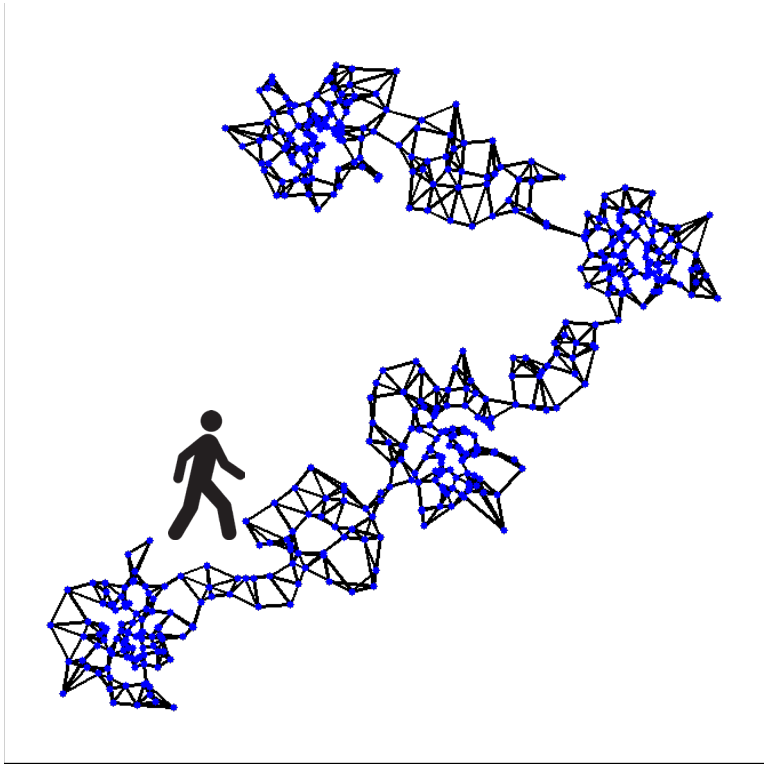
- Large Euclidean distance gives wrong global structure for visualizing
- Small Euclidean distances ok
  - Encodes local structure

From distances to affinities

Affinity: a measure of similarity between points

- Step 1: Calculate all pairwise Euclidean distances
- Step 2: Convert the distances to pairwise affinities using a kernel function
  - E.g. the Gaussian kernel: $k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{\sigma^2}\right)$
  - Kernel function must be near zero for large distances and nonzero for small distances
- I.e. create a similarity graph
- Assume we have data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$

Diffusion Operator



- Big Euclidean steps bad, likely to exit structure
- Small steps good, likely to stay within structure
- Shortest path (ISOMAP) is sensitive to outliers and spurious connections
- **Solution**: consider all possible paths via random walks
- Can do this computationally cheaply using diffusion

Diffusion Operator

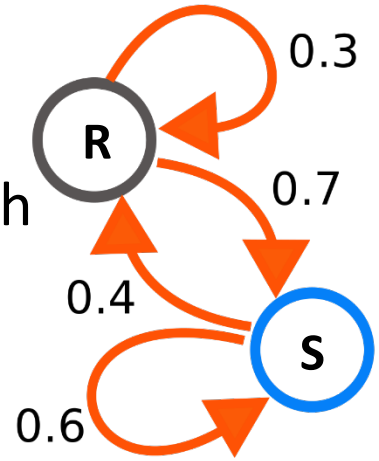
- Denote $K \in \mathbb{R}^{n \times n}$ as the pairwise affinity/kernel matrix
- Construct the degree matrix D
  - Sum the entries of each row and place along the diagonal
- Normalize K by D to obtain the diffusion operator P

$$P = D^{-1}K$$

- Properties of P:
  - A normalized Graph Laplacian
  - All entries are nonnegative
  - Not symmetric
  - Row stochastic (each row sums to 1)
  - Forms a Markov transition matrix

Markov Processes

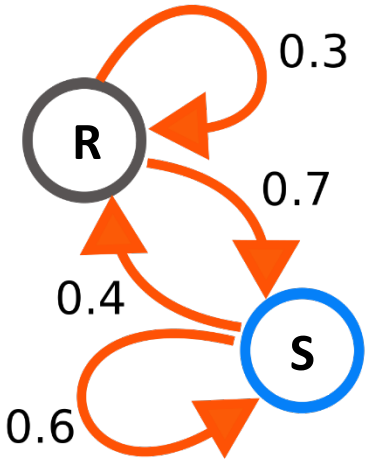
- Used to model a system's state over time
- Consists of a set of possible states and transition probabilities
- **Example**: suppose the weather is either sunny or rainy
  - Two states: sunny (S) and rainy (R)
  - If states are independent, then we flip a coin each day
  - Weather is usually NOT independent
  - Suppose we live in an area where it's uncommon for it to rain for multiple days
  - Model this with a Markov chain
  - Transition probabilities indicate how "sticky" each state is



Markov Processes

- **Example**: suppose the weather is either sunny or rainy
  - Two states: sunny (S) and rainy (R)
  - Probability transition matrix used to summarize the dynamics

$$P = \begin{matrix} \\ R \\ S \end{matrix} \begin{matrix} \begin{matrix} R & S \end{matrix} \\ \begin{bmatrix} 0.3 & 0.7 \\ 0.4 & 0.6 \end{bmatrix} \end{matrix}$$



  - P gives the probability of transitioning from one state to any other state in a single time step
  - Probability of transitioning from one state to any other in t time steps is obtained from P^t
- Markov assumption: future states depend only on the present state and are independent of past states
  - All the necessary "memory" is contained in the present state

Group Exercise

1. Give an example of a process where the Markov assumption may be valid
2. Give an example of a process where the Markov assumption may NOT be valid

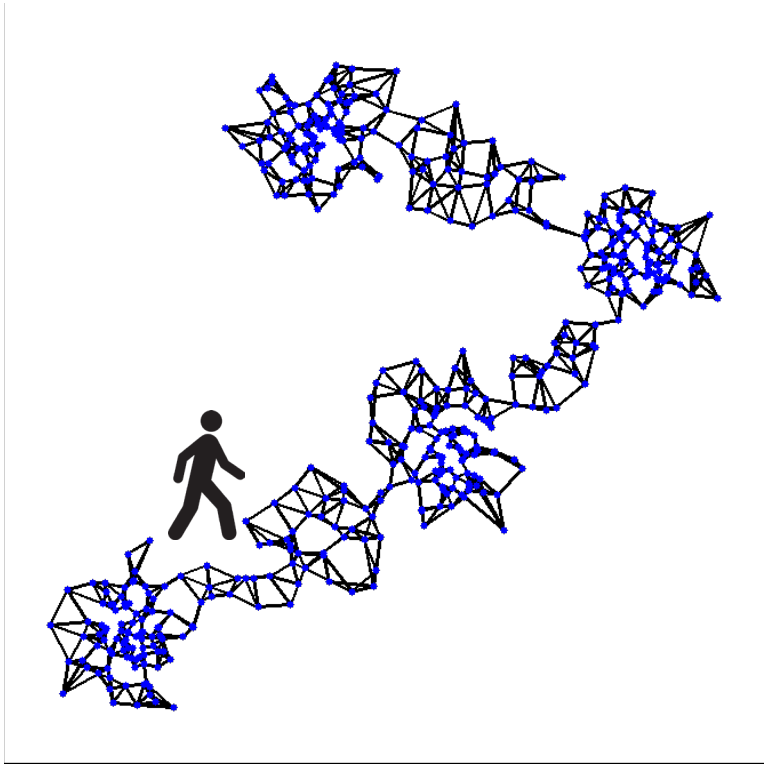
Stationary Distribution

- Consider a probability distribution π as a row vector
  - Nonnegative elements that sum to 1
- π is a *stationary distribution* of the Markov chain if

$$\pi P = \pi$$

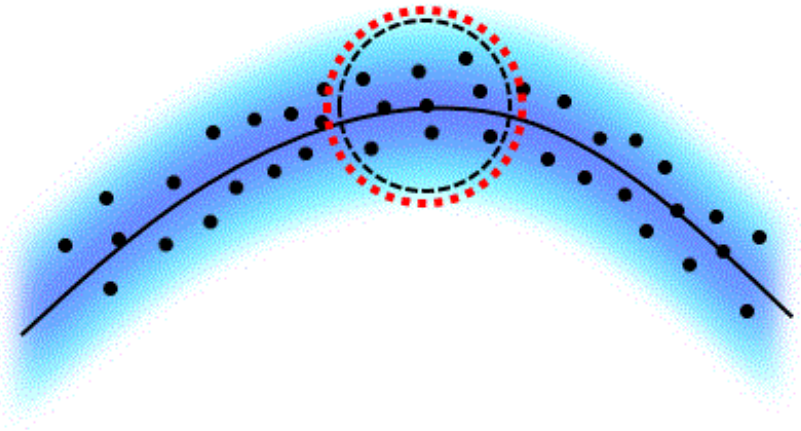
  - π is unchanged by P
- A stationary distribution represents the steady state conditions of the Markov chain
  - The probability you end up in each corresponding state after a long time (subject to initial conditions)
- **Note**: π is an eigenvector of P with eigenvalue of 1
  - Can use the eigenvalue equation to find π

Diffusion Denoises and Recovers Global Structure



- Big Euclidean steps bad, likely to exit structure
- Small steps good, likely to stay within structure
- To learn global structure via small local steps we use a random walk (diffusion)
  - Normalize affinity matrix to create Markov transition matrix (the diffusion operator) P (Coifman & Lafon, *ACHA*, 2006)
  - Power P by time step t

Diffusion Denoises and Recovers Global Structure



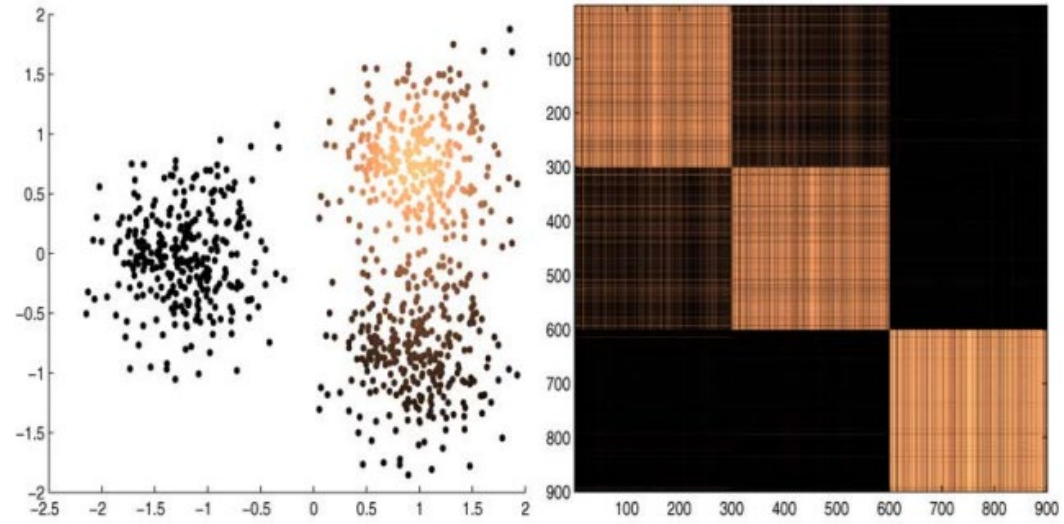
The time scale t

- **Remark**: all eigenvalues of P are between 1 and -1
- Recall that powering a matrix is equivalent to powering the eigenvalues:

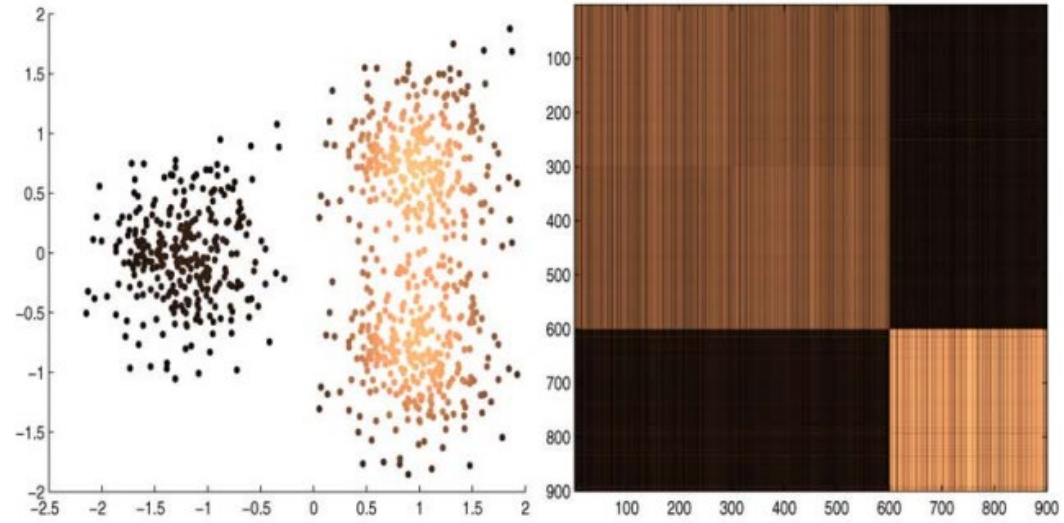
$$P^t = U\Lambda^t U^{-1}$$

- Only the stationary distribution eigenvalues remain after large t
- Thus too large of t can oversmooth the data
- But too small of t doesn't learn the global structure
- We'll see more on this when we talk about visualization and PHATE

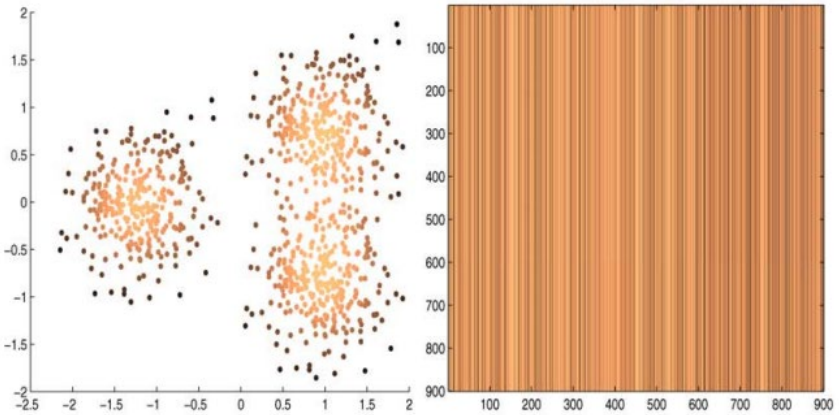
The time scale t



(a) $t = 8$



(b) $t = 64$



(c) $t = 1024$

Diffusion distances

- Denote P_{ij}^t as the ij-th entry of P^t
- The diffusion distances:

$$D_t(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 = \sum_{m=1}^{n} \frac{\left(P_{im}^t - P_{jm}^t\right)^2}{\pi_m}$$

  - Can obtain $\pi_m = \frac{D_{mm}}{\sum_{i=1}^n D_{ii}}$
  - This is a weighted ℓ^2 distance between the columns of P^t
- **Properties**:
  - Distances are smaller if points are highly connected
  - Considers all paths of length t (robust to noise paths)
  - Good from an ML perspective (an ensemble approach)

Diffusion Maps

- Let $\boldsymbol{\psi}_\ell$ be the ℓth right eigenvector of P w/ eigenvalue λ_ℓ
  - I.e. $P\boldsymbol{\psi}_\ell = \lambda_\ell \boldsymbol{\psi}_\ell$
- A diffusion map gives the $\mathbb{R}^k$ embedding of the ith data point:

$$\Psi_t(\boldsymbol{x}_i) = \begin{bmatrix} \lambda_1^t \boldsymbol{\psi}_1(i) \\ \lambda_2^t \boldsymbol{\psi}_2(i) \\ \vdots \\ \lambda_k^t \boldsymbol{\psi}_k(i) \end{bmatrix}$$

- For sufficiently large k,

$$\left\| \Psi_t(\boldsymbol{x}_i) - \Psi_t(\boldsymbol{x}_j) \right\| \approx D_t(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- Thus the mapping Ψ_t reorganizes points according to their pairwise diffusion distances
- **Note**: we assumed the graph is fully connected and that ψ_0 is a vector of ones with $\lambda_0 = 1$

Different diffusion processes

We can generalize the diffusion maps further

1. Define $K^{(\alpha)} = D^{-\alpha} K D^{-\alpha}$
2. Compute the degree matrix $D^{(\alpha)}$ of $K^{(\alpha)}$
3. Apply graph Laplacian normalization: $P = \left(D^{(\alpha)}\right)^{-1} K^{(\alpha)}$
4. Obtain new coordinates from eigendecomposition of P^t

Assuming isotropic (e.g. Gaussian) edge weights:

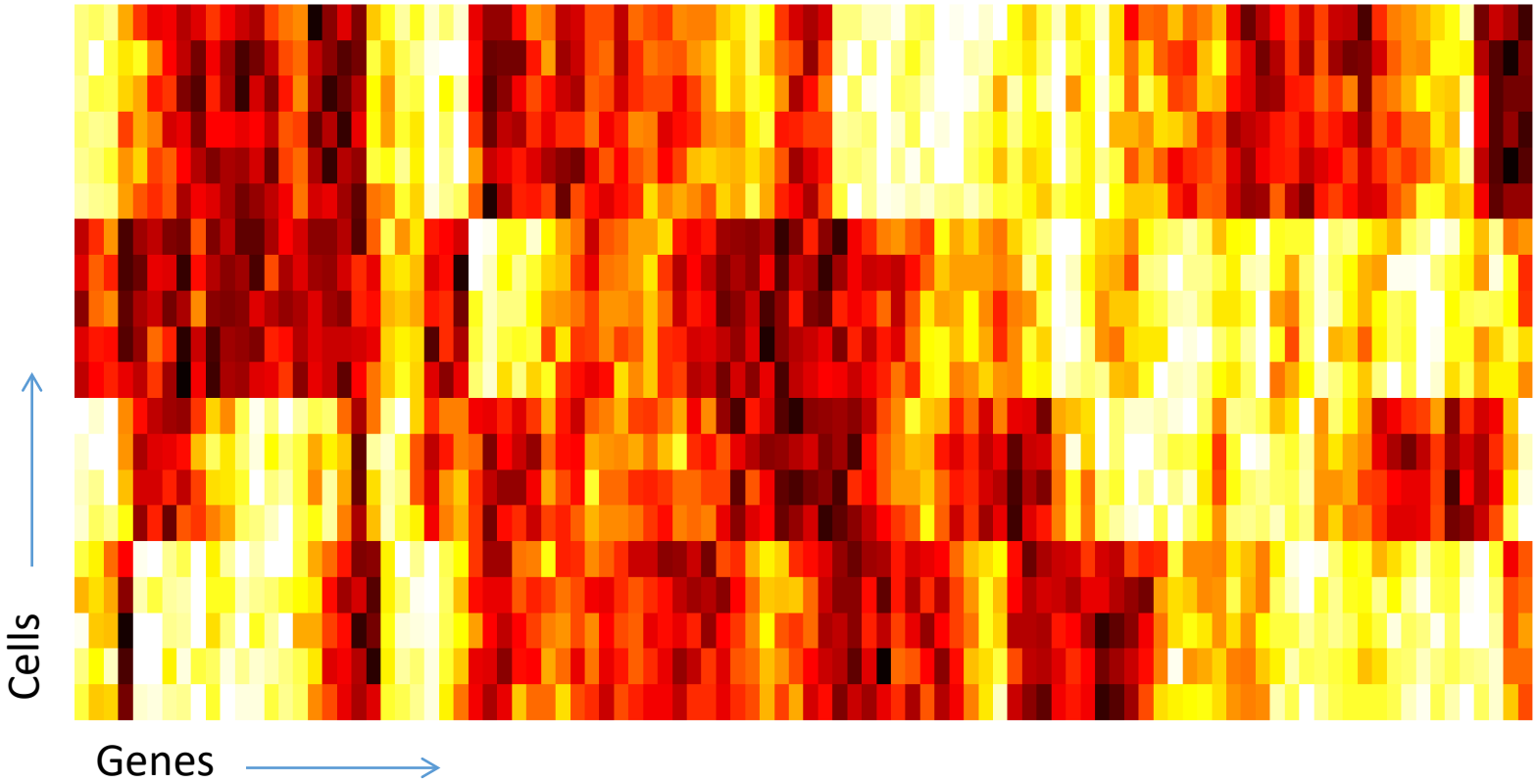
- $\alpha = 1$ approximates the Laplace-Beltrami operator
  - We recover the Riemannian geometry regardless of density
- $\alpha = 0$ gives the classical normalized graph Laplacian
  - Influence of the density is high
- $\alpha = \frac{1}{2}$ approximates the Fokker-Planck diffusion process

Diffusion Maps Comments

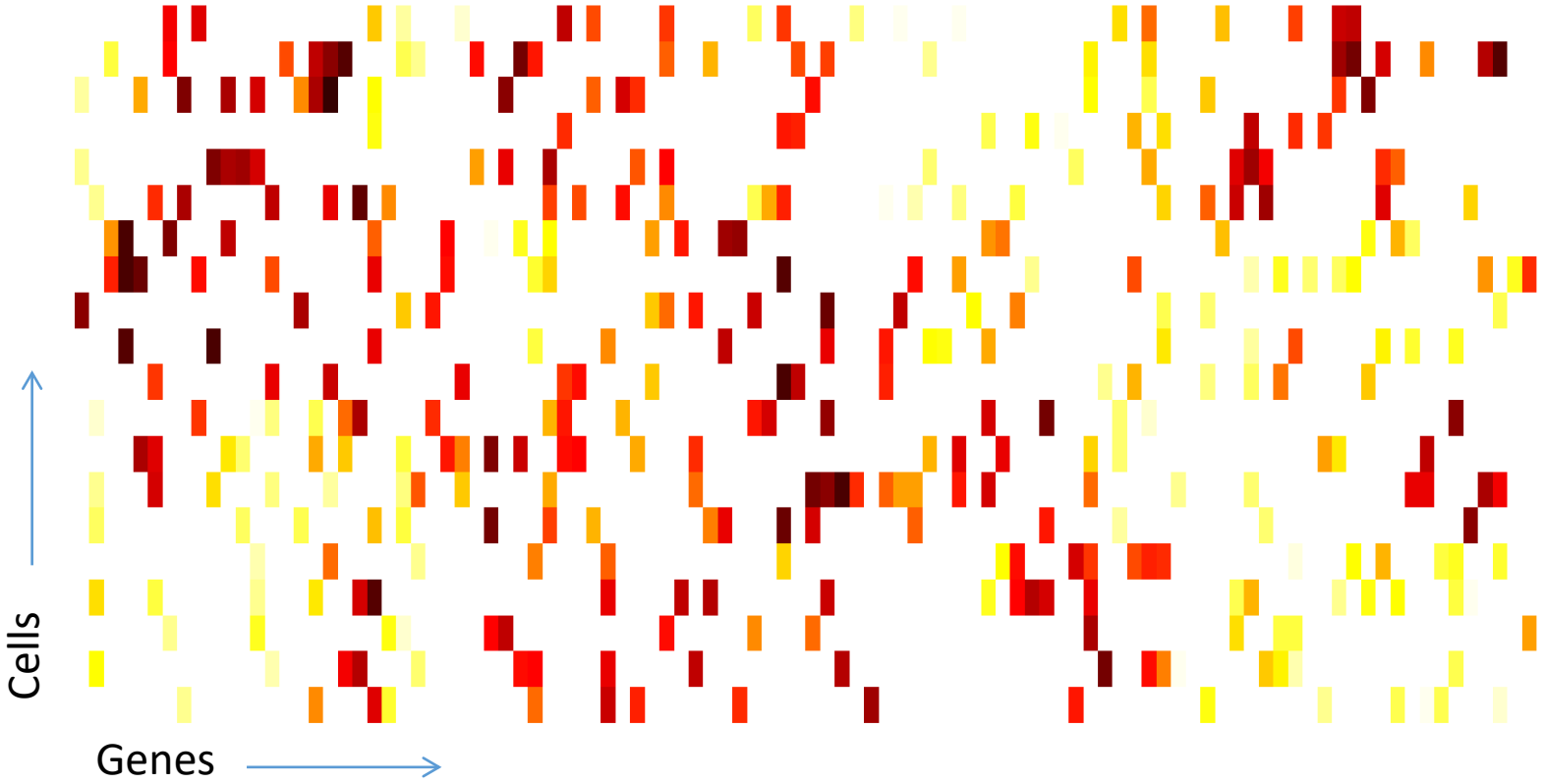
- When using a Gaussian kernel with bandwidth σ^2, n must grow faster than $(\sigma^2)^{-\frac{d}{4}-\frac{1}{2}}$ to guarantee convergence to the manifold
- When the data are noisy and when using a Gaussian kernel with bandwidth σ^2, σ must remain larger than the noise size to guarantee a good approximation
- The parameter t controls the scale of the global relationships learned
  - Larger t looks at larger scales
  - Haghverdi et al, (2016) look at a scale free version to create "diffusion pseudotime"

Data Imputation with MAGIC

scRNA-seq data: High Dimensional but Noisy

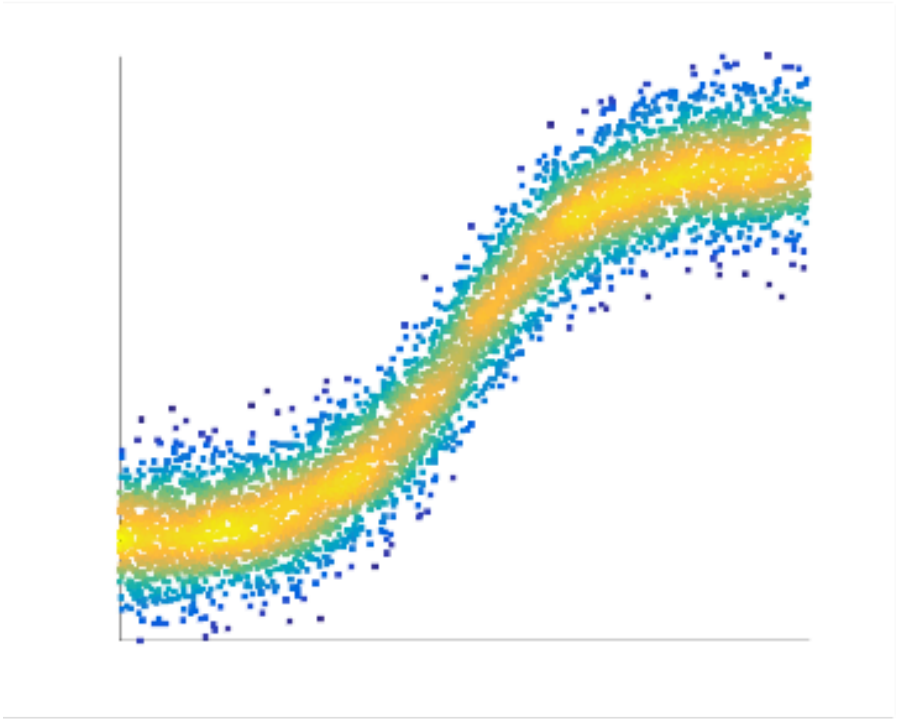


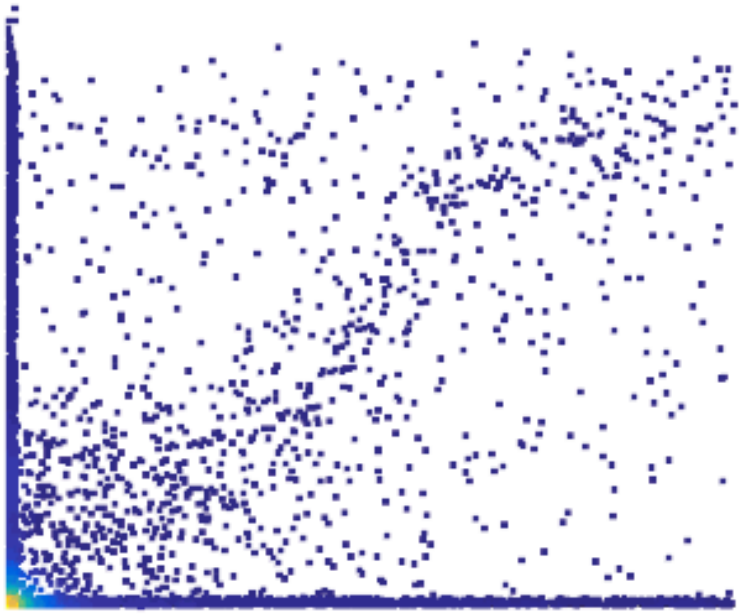
Dropout in scRNA-seq data



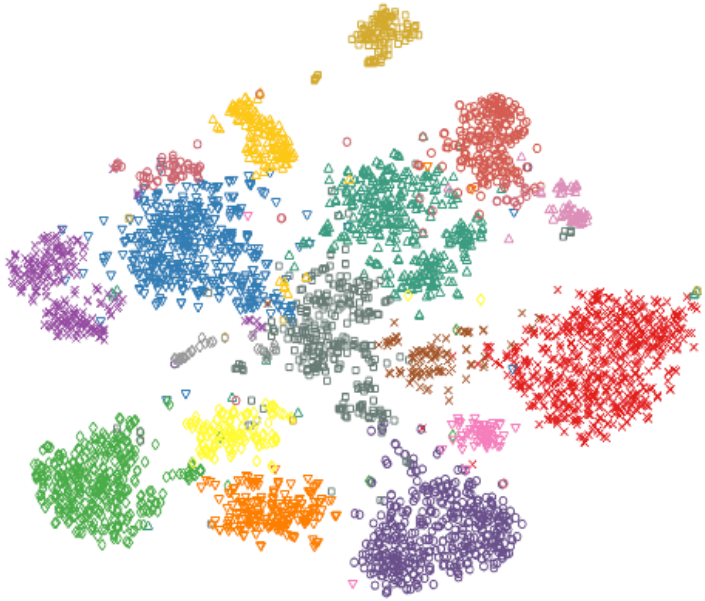
- Obscuring biological structure
- Dropout creates sparse data
- Hard to learn gene interactions

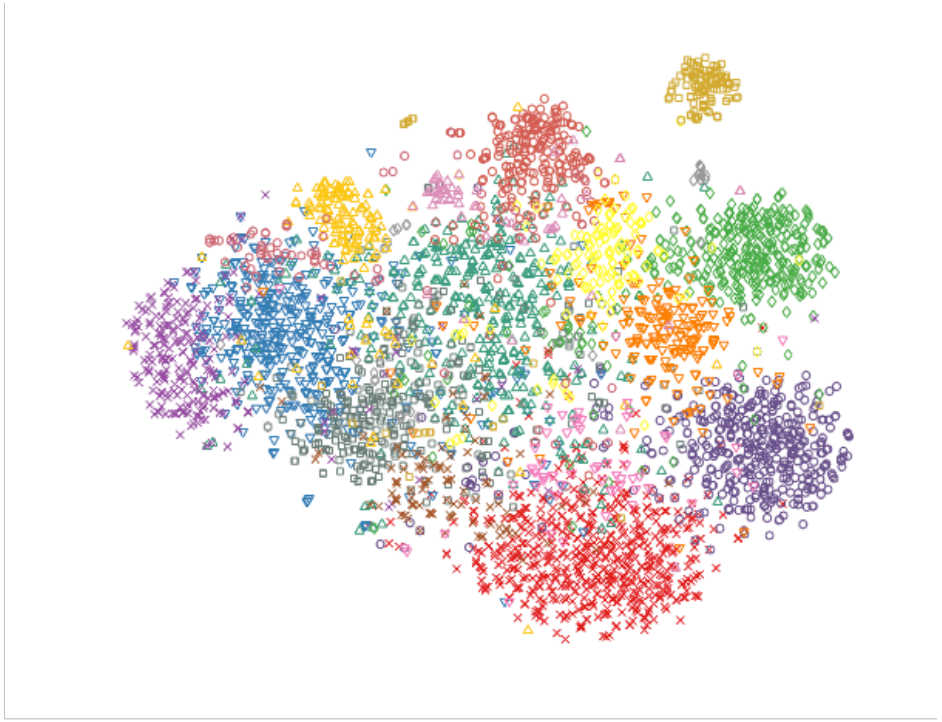
Dropout destroys gene-gene relationships





Dropout destroys clusters





Imputation in other settings: Matrix Completion

- Goal: fill in the missing entries of a partially observed matrix M
- **Example**: Netflix movie-ratings matrix
  - Movies along one dimension, users along the other
  - Can we predict the unobserved rating of a movie by a given user?
    - Used to recommend movies
- **Low rank matrix completion**: find the lowest rank matrix X which matches the matrix M in the set E of the observed entries

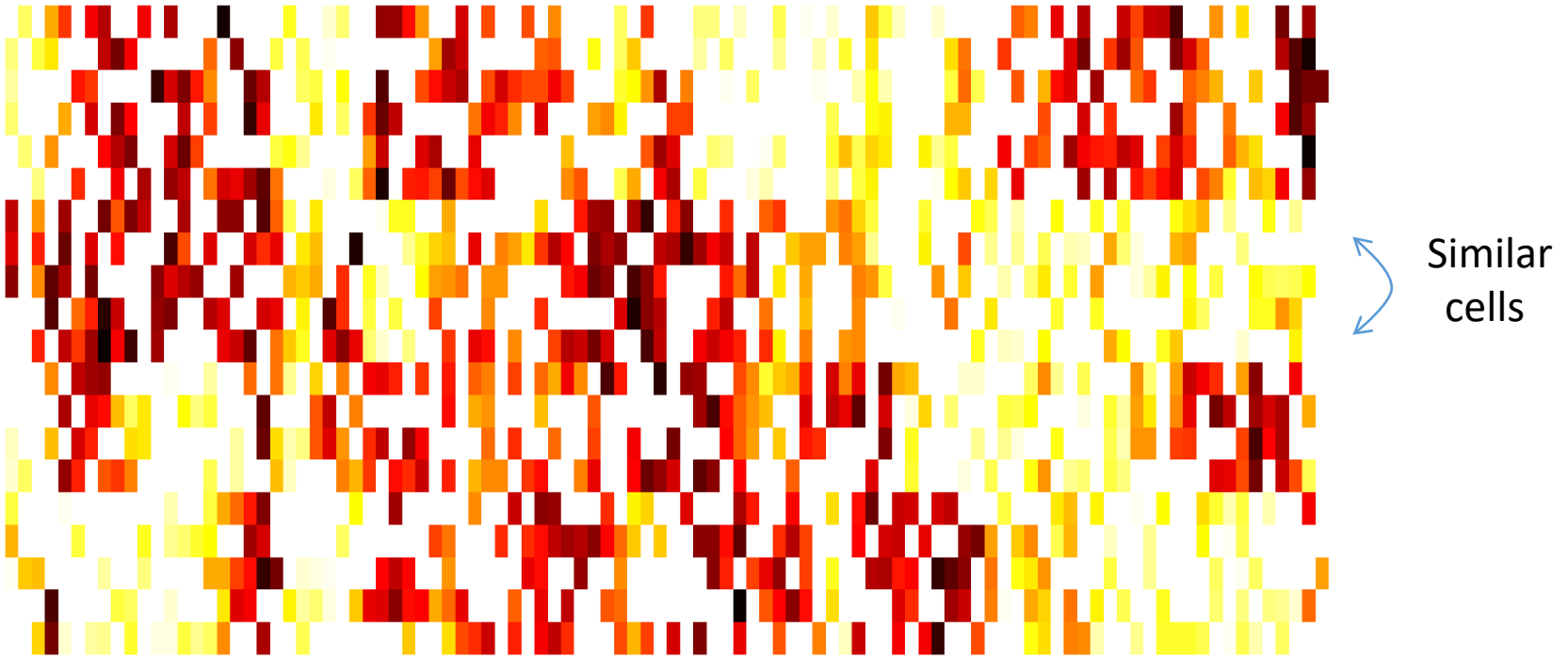
$$\min_{X} \quad rank(X)$$
$$s.t. \quad X_{ij} = M_{ij} \quad \forall i,j \in E$$

Low rank matrix completion

$$\min_{X} \quad rank(X)$$
$$s.t. \quad X_{ij} = M_{ij} \quad \forall i,j \in E$$

- **Assumptions**:
  - Uniform sampling of observed entries
  - Lower bound on number of observed entries
  - Incoherence (singular vectors of M are not too sparse)
- **Problem**: this optimization problem is NP-hard
  - Can get around this by approximating the optimization problem with a convex relaxation (e.g. replace the non-convex functions with convex functions that are upper bounds)
- **Another problem**: this is a linear method
  - Most biological processes are nonlinear

MAGIC: Impute by learning from similar cells



- Biologically similar cells will have similar transcriptomes
- Cells “exchange” information

How MAGIC works

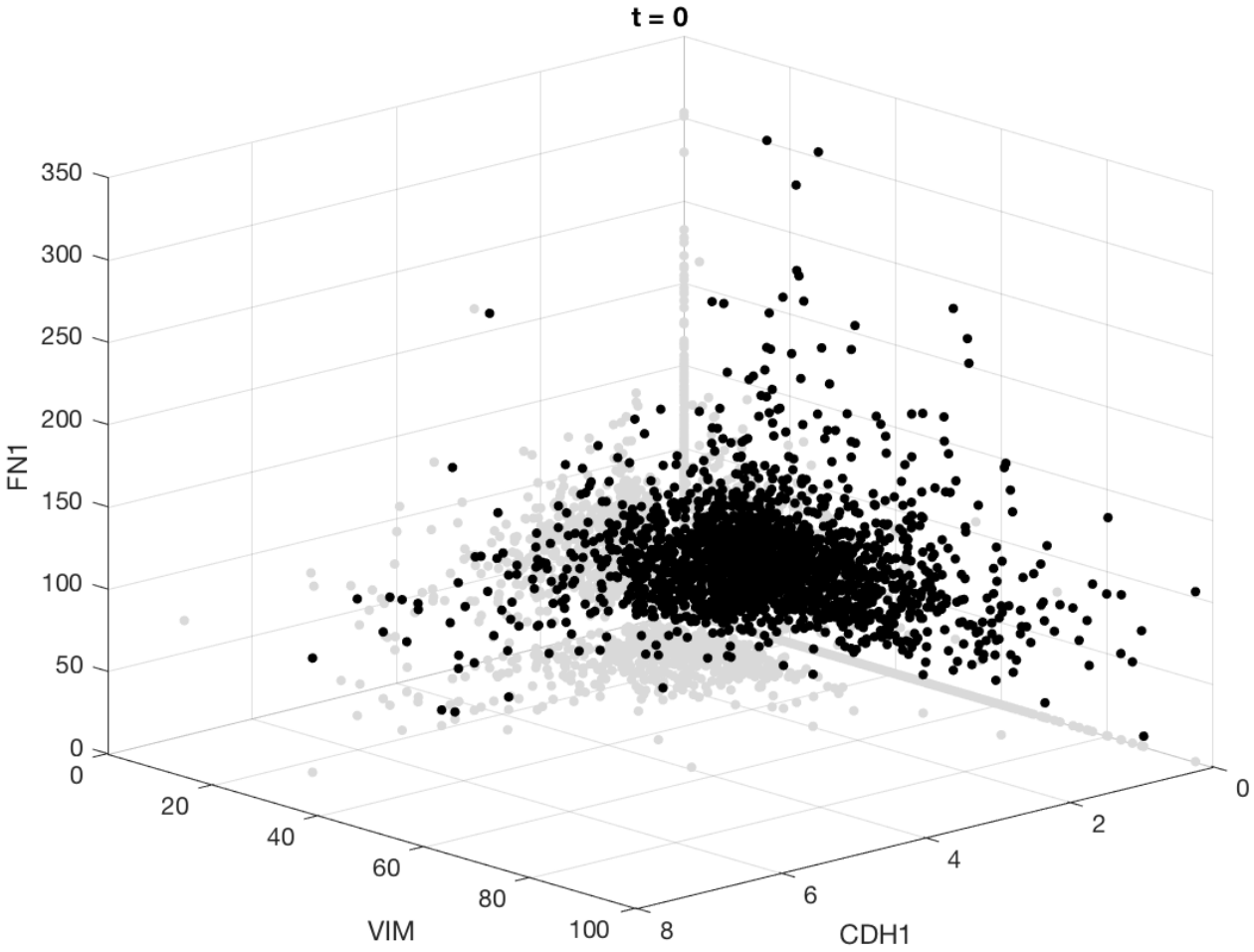
- Compute the diffusion operator P from the scRNA-seq data matrix X
  - Each row of X is a cell and each column of X is a gene
- Diffuse the operator with time step t
- Multiply the data by the diffused operator:

$$X_{MAGIC} = P^t X$$

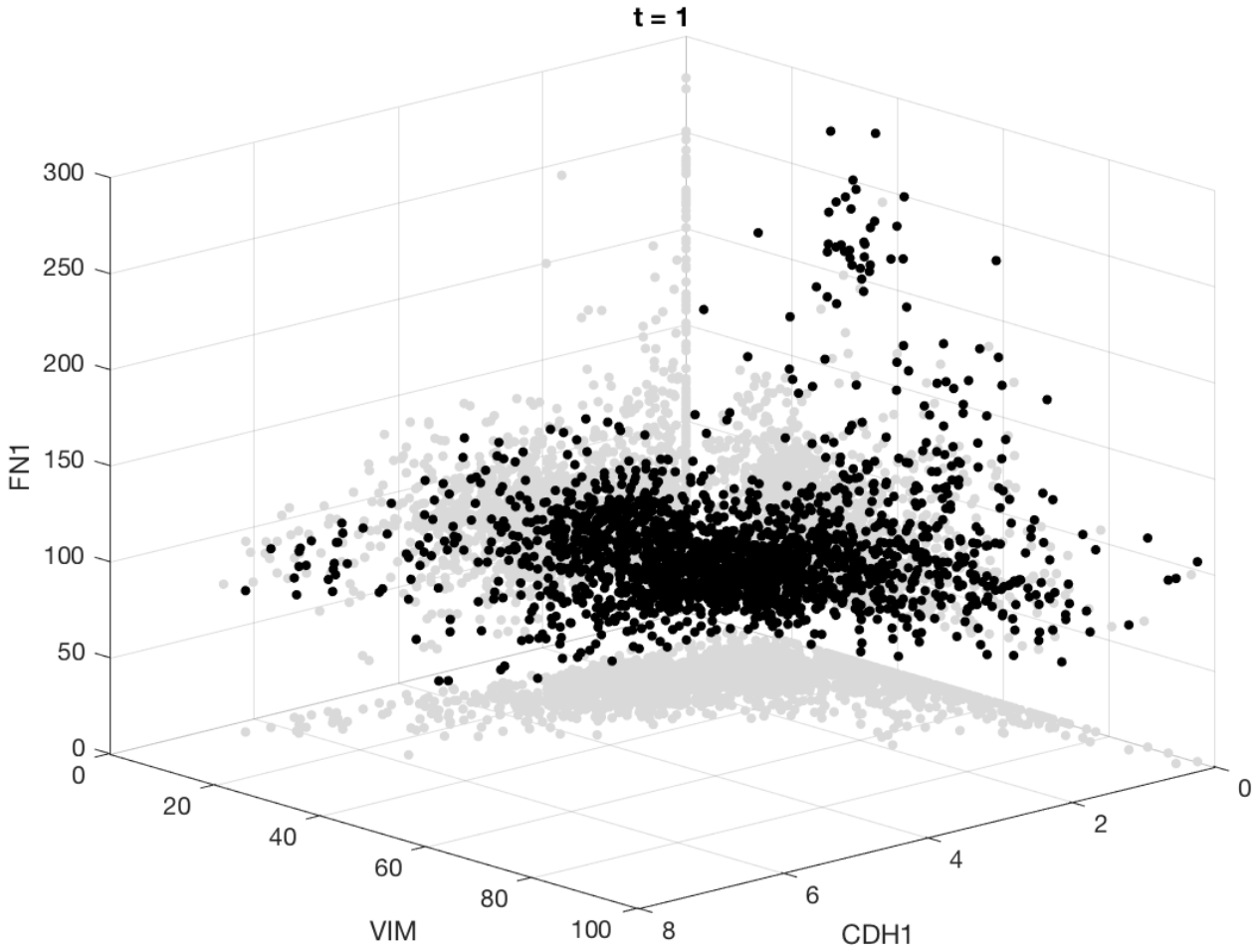
  - Simultaneously imputes and denoises the data

- Published at *Cell* (van Dijk et al., 2018)

Example

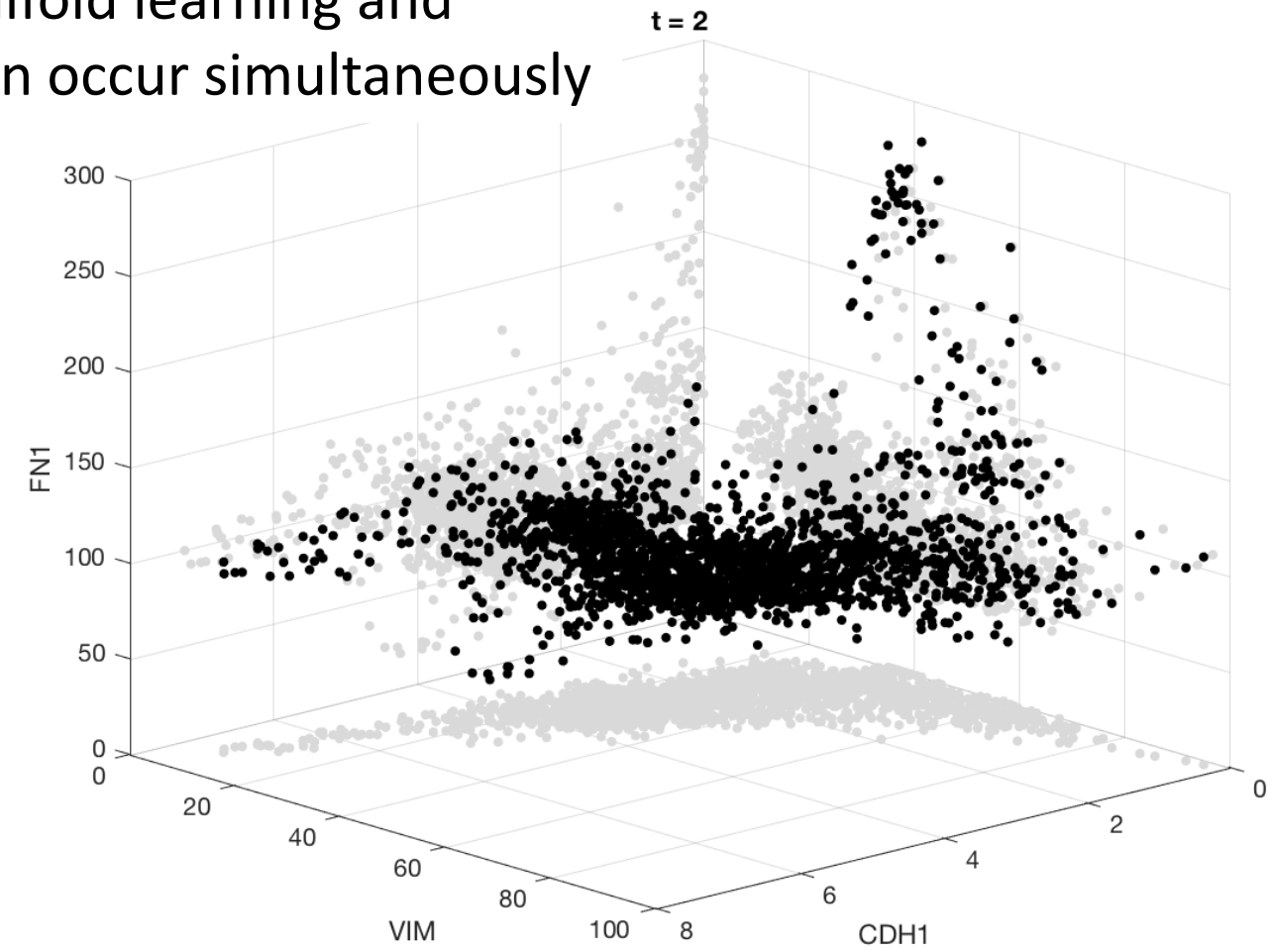

t = 0
FN1
350
300
250
200
150
100
50
0
0
20
40
60
80
100
VIM
8
6
4
2
0
CDH1

Example

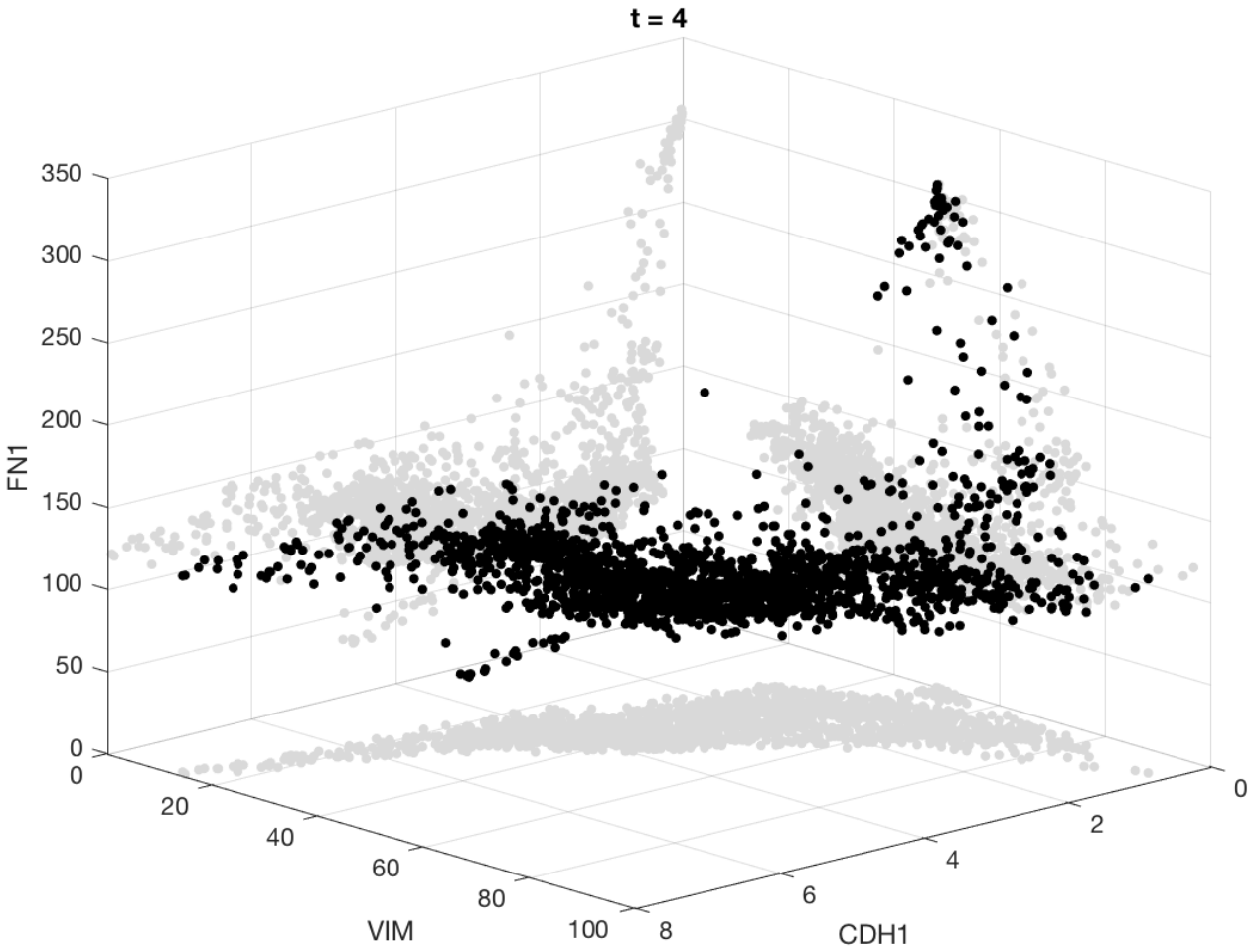


Example

Manifold learning and imputation occur simultaneously

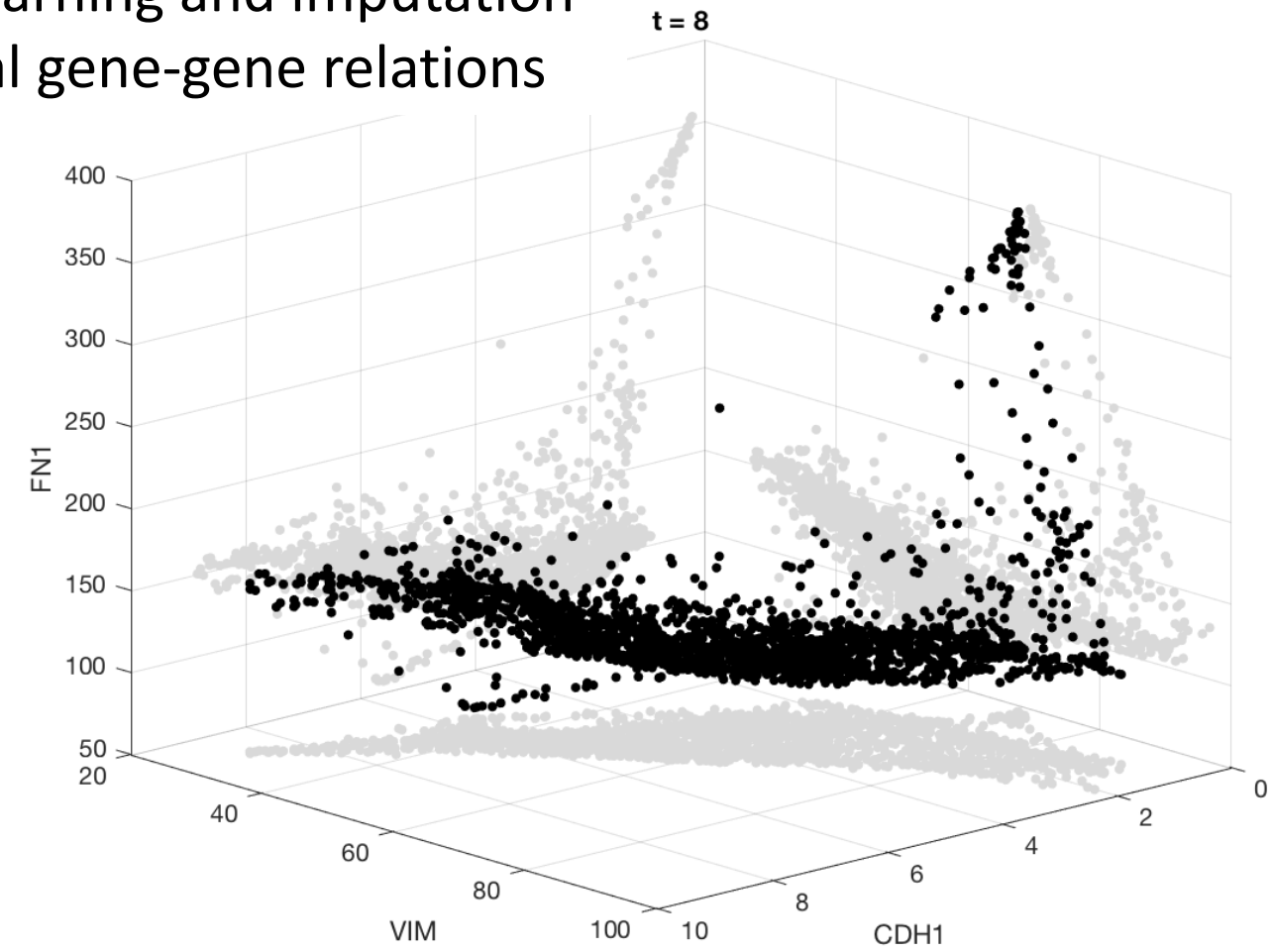


Example


t = 4
FN1
350
300
250
200
150
100
50
0
0
20
40
60
80
100
VIM
8
6
4
2
0
CDH1

Example

Manifold learning and imputation help reveal gene-gene relations



Validate MAGIC by artificial dropout

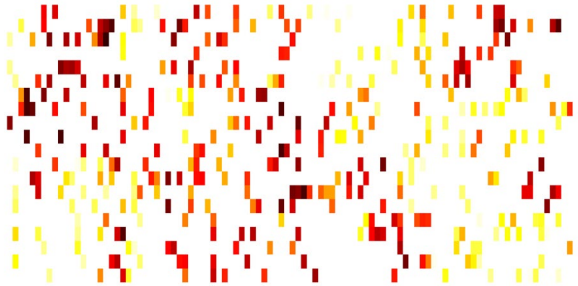
- Test case where we know "ground truth" that we can artificially dropout and recover
- Bulk data that lies on a manifold
- Simulate drop out by uniformly randomly reducing molecule count per matrix entry
- MAGIC recovers 60% in the presence of 80% dropout

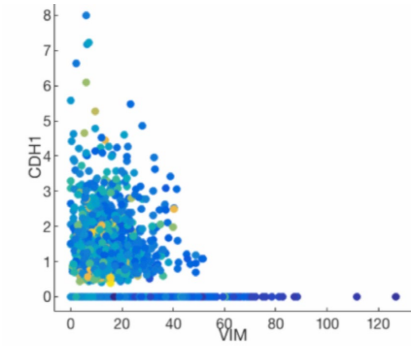
- MAGIC was also used to discover new gene-gene interactions in breast cancer tissue
  - The EMT continuum

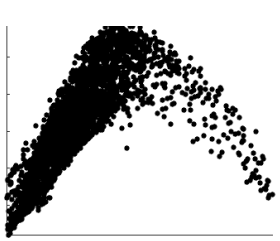
- See the *Cell* paper for details

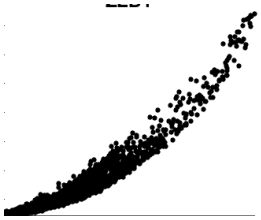
MAGIC Summary

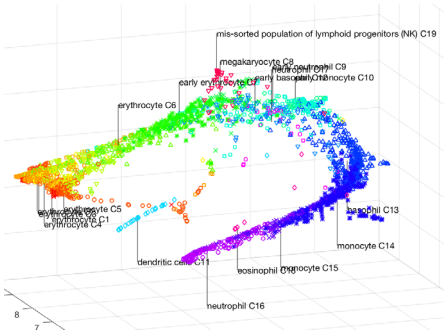
- scRNA-seq data is sparse
- MAGIC recovers structure:
  - Data diffusion process
  - Imputation from robust neighbors
- MAGIC reveals biology:
  - EMT continuum











Final comments on Diffusion methods

- Diffusion-based methods are very useful for learning the underlying structure of the data
  - There is a lot of good theory backing them up
  - Worth digging into if you're interested in manifold learning techniques

Further reading

- MAGIC
  - Paper: https://doi.org/10.1016/j.cell.2018.05.061
  - Code: https://github.com/KrishnaswamyLab/MAGIC
- Diffusion Maps
  - https://en.wikipedia.org/wiki/Diffusion_map
  - https://doi.org/10.1016/j.acha.2006.04.006
  - Diffusion pseudotime: https://www.nature.com/articles/nmeth.3971
- Matrix Completion
  - https://en.wikipedia.org/wiki/Matrix_completion