



CS 540 Introduction to Artificial Intelligence
Unsupervised Learning II

University of Wisconsin-Madison
Fall 2026 Sections 1 & 2

Announcements

- **HW3** due on **Friday Oct 2nd at 11:59 PM**
- Class roadmap:

ML Unsupervised II

ML Linear Regression

Machine Learning: K - Nearest
Neighbors & Naive Bayes

Machine Learning: Neural Networks I
(Perceptron)

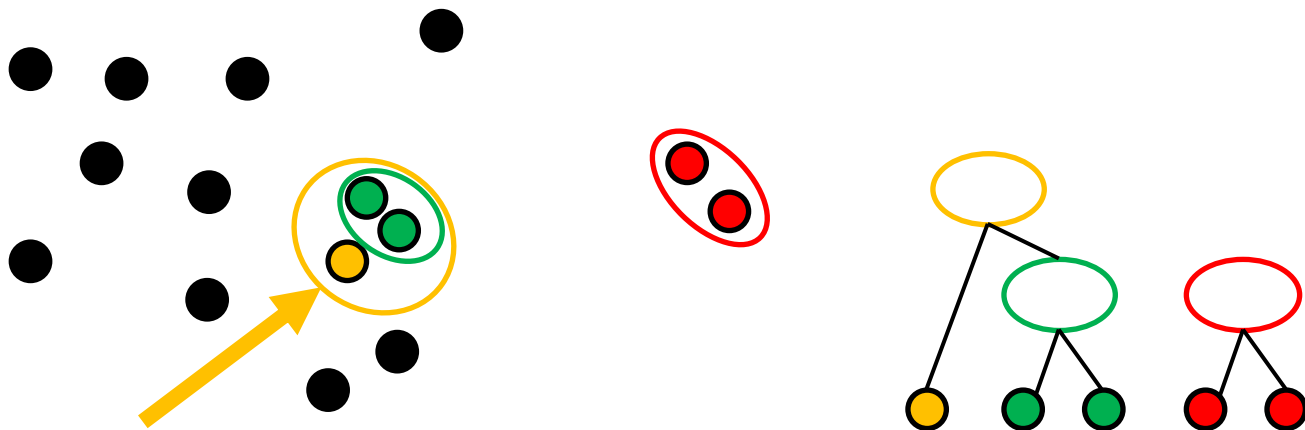
Machine Learning

Outline

- Review Agglomerative Hierarchical Clustering and k-Means
- Other Partition-based clustering Types
 - Graph-based, cuts, spectral clustering
- Unsupervised Learning: Visualization
 - t-SNE, algorithm, example, vs. PCA
- Unsupervised Learning: Density Estimation
 - Kernel density estimation: high-level intro

Review: Hierarchical Agglomerative Clustering

Get pair of clusters that are closest and merge



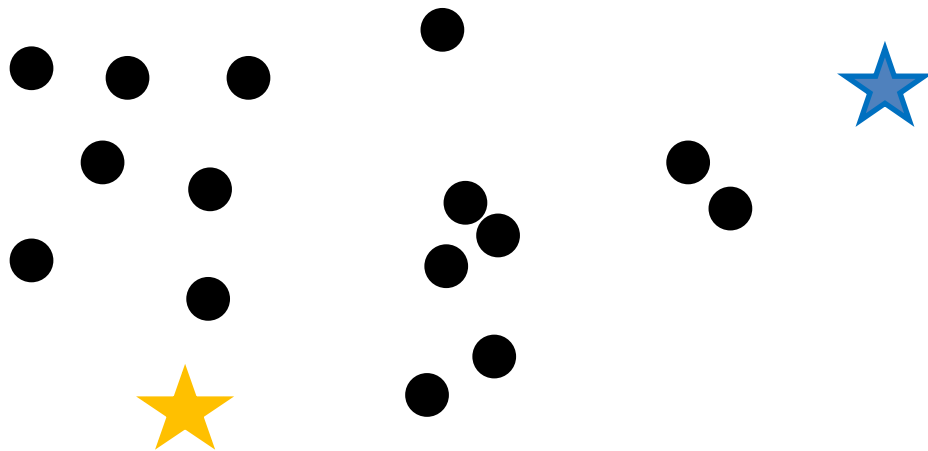
Review: Merging Criteria

Merge: use closest clusters. Define closest?

- Single-linkage $d(A, B) = \min_{x_1 \in A, x_2 \in B} d(x_1, x_2)$
- Complete-linkage $d(A, B) = \max_{x_1 \in A, x_2 \in B} d(x_1, x_2)$
- Average-linkage $d(A, B) = \frac{1}{|A||B|} \sum_{x_1 \in A, x_2 \in B} d(x_1, x_2)$

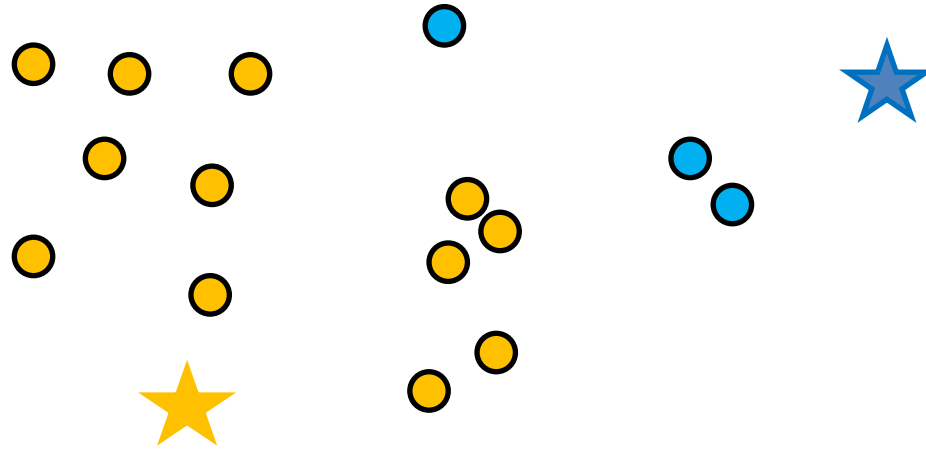
Review: K-Means Clustering

- Steps: **1.** Randomly pick k cluster centers



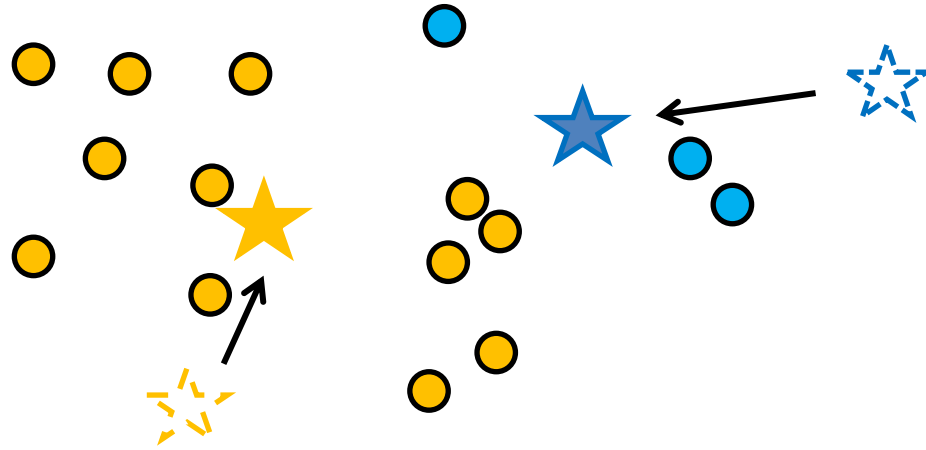
Review: K-Means Clustering

- **2.** Find closest center for each point



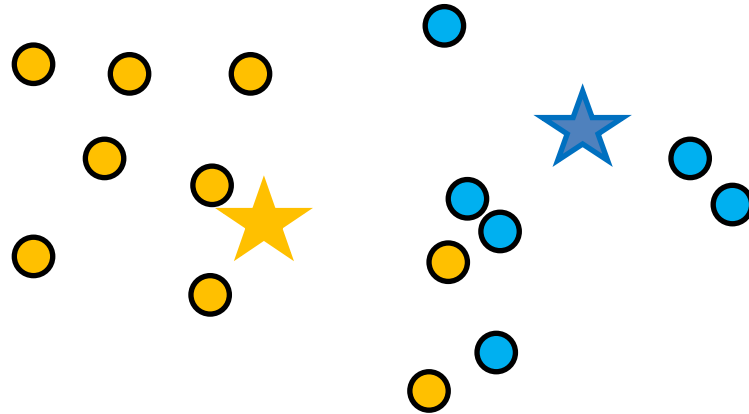
Review: K-Means Clustering

- **3.** Update cluster centers by computing centroids



Review: K-Means Clustering

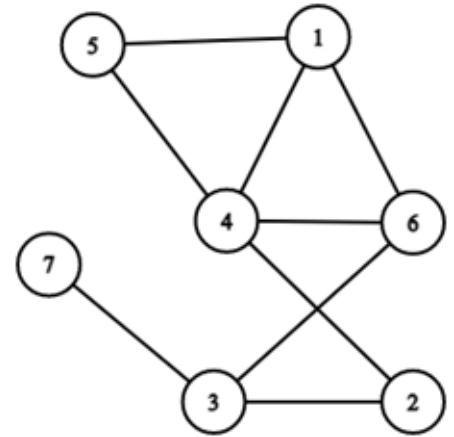
- Repeat Steps 2 & 3 until convergence



Other Types of Clustering

Graph-based/proximity-based

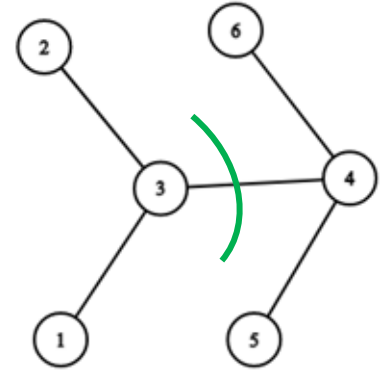
- Recall: Graph $G = (V, E)$ has vertex set V , edge set E .
 - Edges can be weighted or unweighted
 - Encode **similarity**: $w_{ij} = \text{sim}(v_i, v_j)$
- Don't need to KEEP vectors v
 - Only keep the edges (possibly weighted)



Graph-Based Clustering

Want: partition V into V_1 and V_2

- Implies a graph “cut”
- One idea: minimize the **weight** of the cut



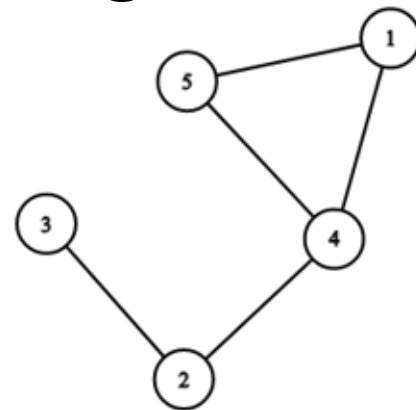
$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

$$\text{cut}(A_1, \dots, A_k) := \frac{1}{2} \sum_{i=1}^k W(A_i, \bar{A}_i).$$

Partition-Based Clustering

How do we compute these?

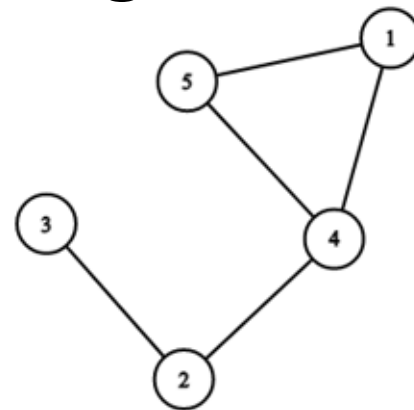
- Hard problem → heuristics
 - Greedy algorithm
 - “Spectral” approaches
- Spectral clustering approach:
 - **Adjacency** matrix



$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Partition-Based Clustering

- Spectral clustering approach:
 - **Adjacency** matrix
 - **Degree** matrix



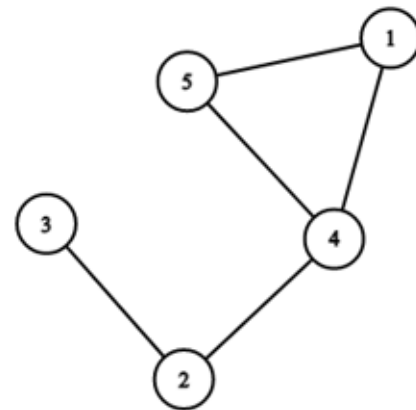
$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Spectral Clustering

- Spectral clustering approach:
 - 1. Compute **Graph Laplacian**

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

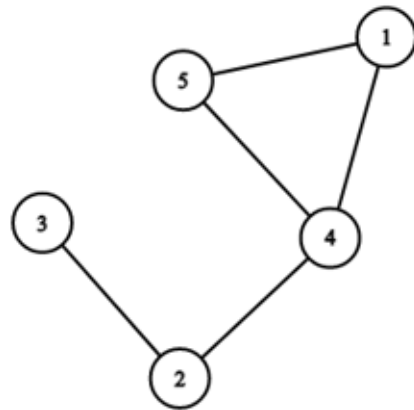


$$\mathbf{L} = \underbrace{\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}}_{\text{Degree Matrix}} - \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{Adjacency Matrix}} = \underbrace{\begin{bmatrix} 2 & 0 & 0 & -1 & -1 \\ 0 & 2 & -1 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 3 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}}_{\text{Laplacian}}$$

Spectral Clustering

- Spectral clustering approach:
 - 1. Compute **Laplacian** $\mathbf{L} = \mathbf{D} - \mathbf{A}$
 - 1a (optional): compute normalized Laplacian:
 $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$, or $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1} \mathbf{A}$
 - 2. Sort eigenvalues in descending order and compute the **bottom** - k eigenvectors of $\mathbf{L}$:

$\mathbb{R}^n$

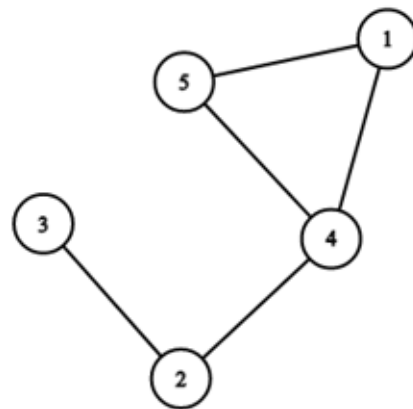


$\mathbf{u}_1, \mathbf{u}_2,$

Spectral Clustering

- 3. Set U to be the $n \times k$ matrix with $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k$ as columns.
- 4. Take the n rows formed as points

$$U = \begin{array}{c} \mathbf{u}_1 \quad \dots \quad \mathbf{u}_k \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \begin{array}{l} x_1 \\ x_2 \\ \dots \\ \dots \\ x_n \end{array} \end{array}$$

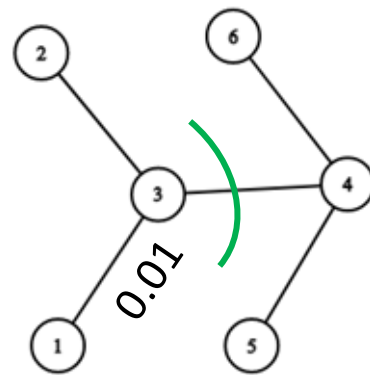
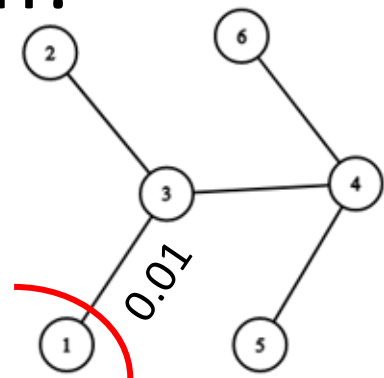


- 5. Run k-means on the representations $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$

Why normalized Laplacian?

Want: partition V into V_1 and V_2

- Implies a graph “cut”
- One idea: minimize the **weight** of the cut
 - Downside: might just cut off one node
 - Need: “**balanced**” cut



Why normalized Laplacian?

Want: partition V into V_1 and V_2

- Just minimizing weight is not always a good idea.
- We want **balance**!

$$\text{Ncut}(A_1, \dots, A_k) := \frac{1}{2} \sum_{i=1}^k \frac{W(A_i, \bar{A}_i)}{\text{vol}(A_i)}$$

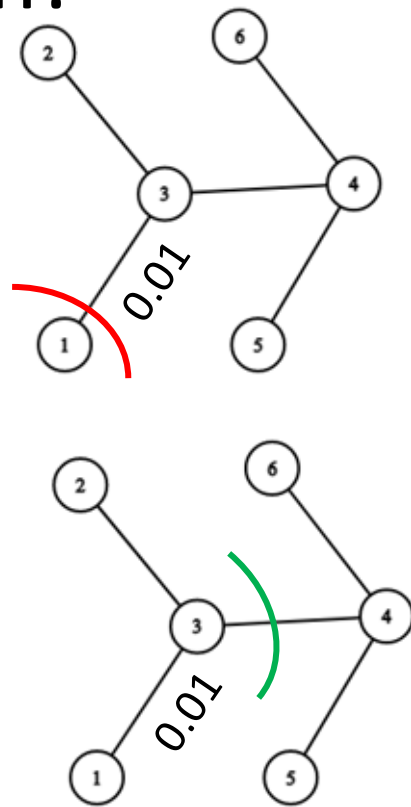
$$\text{vol}(A) = \sum_{i \in A} \text{degree}(i)$$

Why normalized Laplacian?

Assuming all other edges have a weight of 1.

$$\begin{aligned} Ncut(A_1, A_2) \\ = \frac{1}{2} \left(\frac{0.01}{0.01} + \frac{0.01}{(1 + 2.01 + 3 + 1 + 1)} \right) \approx 0.501 \end{aligned}$$

$$\begin{aligned} Ncut(A_3, A_4) \\ = \frac{1}{2} \left(\frac{1}{(0.01 + 1 + 2.01)} + \frac{1}{(3 + 1 + 1)} \right) \approx 0.265 \end{aligned}$$

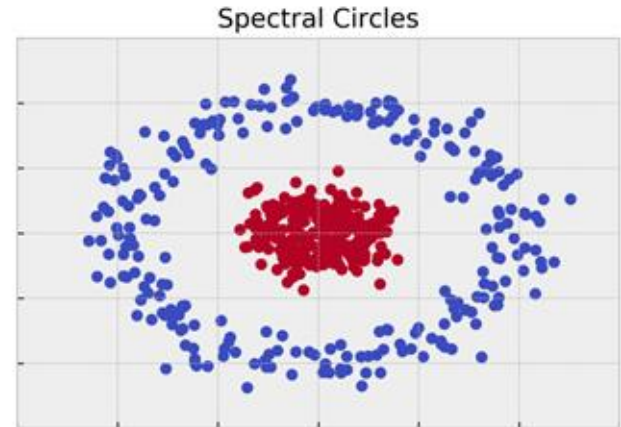
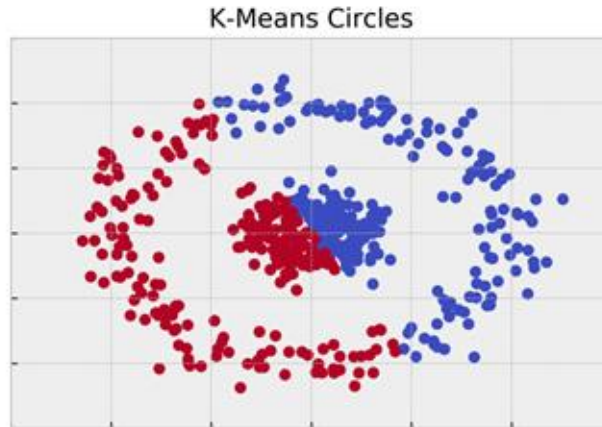


Spectral Clustering

- Compare/contrast to **PCA**:
 - Use an **eigendecomposition** / dimensionality reduction
 - Run on Laplacian (not covariance)
 - Use **smallest-k** eigenvectors, not largest
- Intuition: Laplacian encodes structure information
 - “Lower” eigenvectors give partitioning information

Spectral Clustering

- Can handle clusters of the type below (Why?)
- Intuition: Laplacian encodes structure information
 - “Lower” eigenvectors give partitioning information

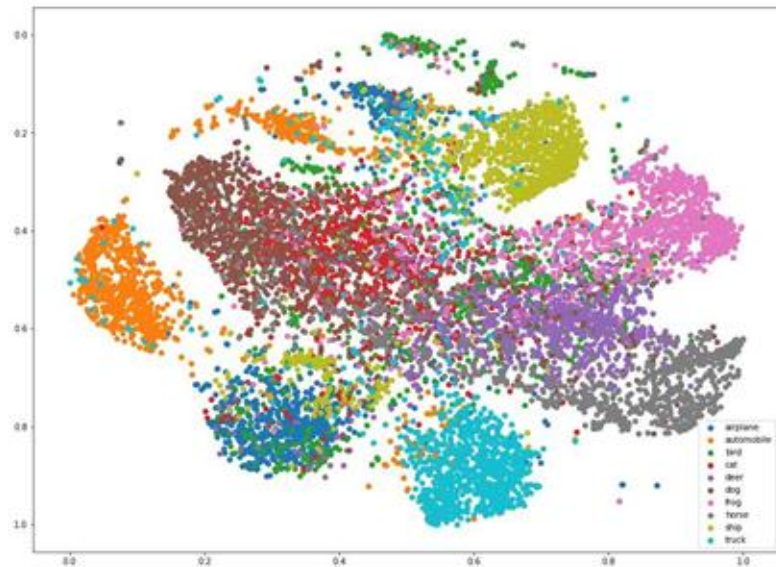


Credit: William Fleshman

Unsupervised Learning Beyond Clustering

Data analysis, dimensionality reduction, etc

- Already talked about PCA
- Note: PCA can be used for visualization, but not specifically designed for it
- Some algorithms **specifically** for visualization



Philip Slingerland

Dimensionality Reduction & Visualization

Typical dataset: MNIST

- Handwritten digits 0-9
 - 60,000 images (small by ML standards)
 - 28×28 pixel (784 dimensions)
 - Standard for image experiments

- Dimensionality reduction?



Dimensionality Reduction vs Visualization

Run PCA on MNIST

- PCA is a linear mapping,
(can be restrictive)



3	6	8	1	7	9	6	6	4	1
6	7	5	7	8	6	3	4	8	5
2	1	7	9	7	1	2	3	4	5
4	8	1	9	0	1	8	8	9	4
7	6	1	8	6	4	1	5	6	0
7	5	9	2	6	5	8	1	9	7
2	2	2	2	3	4	4	8	0	
0	2	3	8	0	7	3	8	5	7
0	1	4	6	4	6	0	2	4	3
7	1	2	8	7	6	9	8	6	1

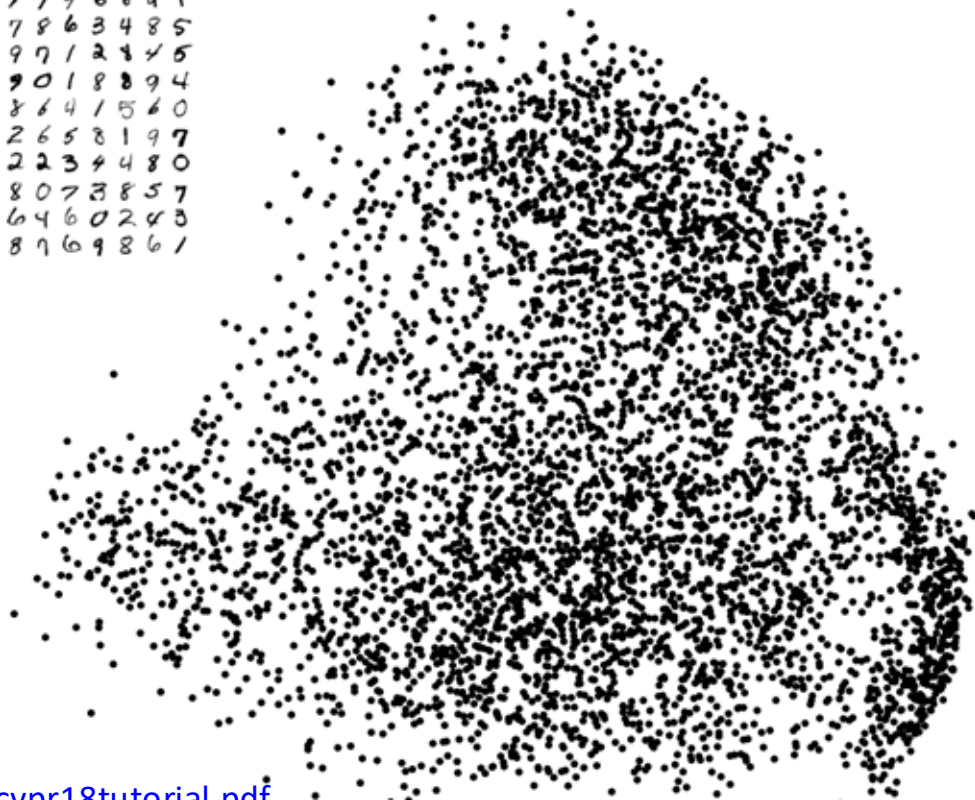


Image source:

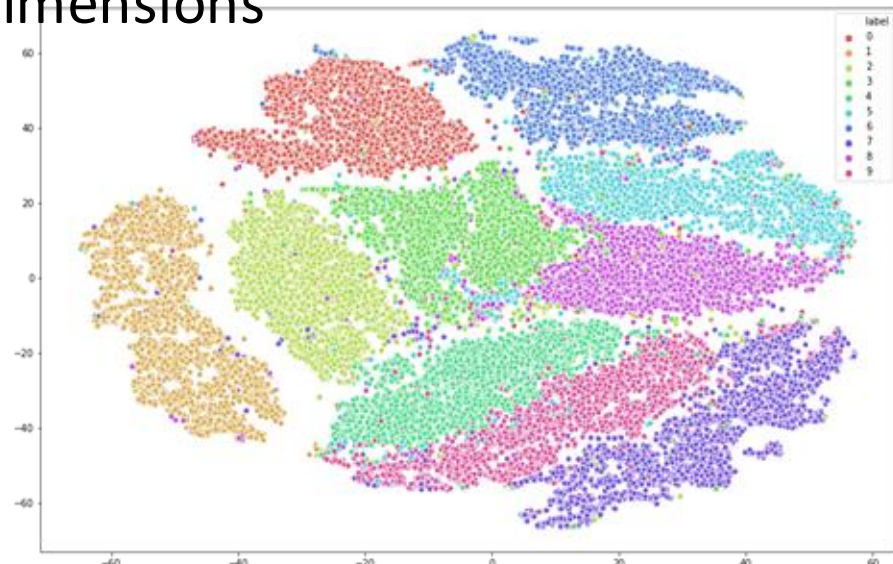
http://deeplearning.csail.mit.edu/slide_cvpr2018/laurens_cvpr18tutorial.pdf

Visualization: T-SNE

T-SNE: T-distributed Stochastic Neighbor Embedding

Typical dataset: MNIST

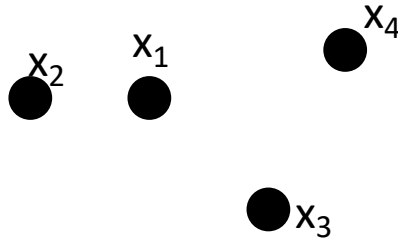
- **T-SNE:** project data into just 2 dimensions
- Try to maintain structure
- **Input:** $x_1, x_2, \dots, x_n$
- **Output:** 2D/3D $y_1, y_2, \dots, y_n$



T-SNE Algorithm

How does it work? Two steps

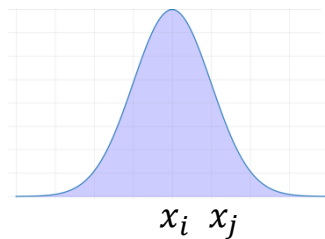
- **1.** Turn vectors into probability pairs
- **2.** Turn pairs back into ***lower dimensional*** vectors



T-SNE Algorithm: Step 1

Step 1 High-Dimensional Space:

We calculate the similarity of point x_j to x_i using a Gaussian distribution.

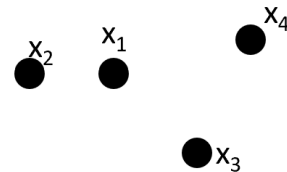


$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)} \quad p_{ij} = \frac{1}{2n} (p_{j|i} + p_{i|j})$$

Intuition: probability that x_i would pick x_j as its neighbor under a Gaussian probability.

Key Concept: We care about *probabilities* of being neighbors, not the absolute distance.

T-SNE Algorithm: Step 2



Step 2 Low-Dimensional Space:

- We place points randomly in 2D.
- We use the Student t-distribution to measure similarity q_{ij} here.

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq \ell} (1 + \|y_k - y_\ell\|^2)^{-1}}$$

and minimize

$$\sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$



KL (Kullback Leibler)
Divergence between
p and q

T-SNE Algorithm: Step 2

More on step 2:

- We have two distributions p, q . p is fixed
- q is a function of the y_i which we move around
- Move y_i around until the KL divergence is small
 - So we have a good representation!
- **Optimizing a loss function**---we'll see more in supervised learning.

$$\sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$



KL Divergence
between p and q

T-SNE Examples

- Examples: (from Laurens van der Maaten)

<https://lvdmaaten.github.io/tsne/>

Short Intro to Density Estimation

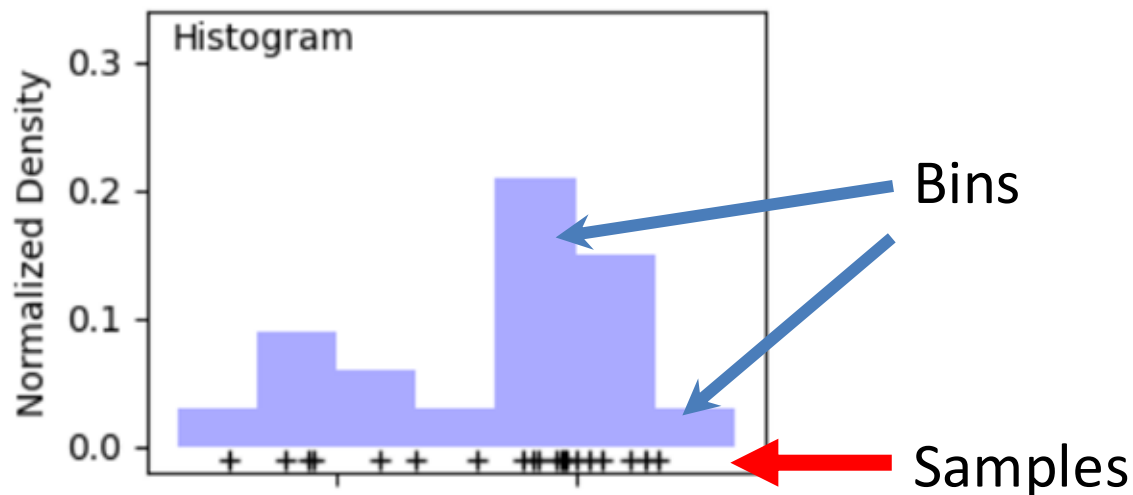
Goal: given samples $x_1, \dots, x_n$ from some distribution P , estimate P .



Zach Monge

Simplest Idea: Histograms

Goal: given samples $x_1, \dots, x_n$ from some distribution P , estimate P .



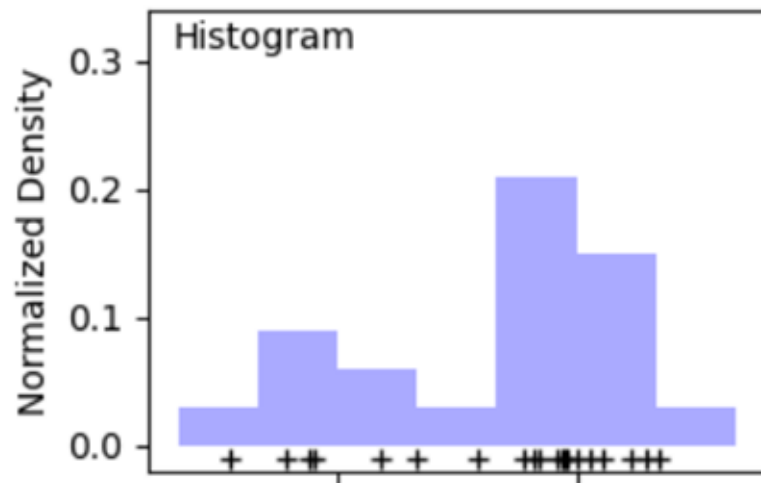
Define bins; count # of samples in each bin, normalize

Simplest Idea: Histograms

Goal: given samples $x_1, \dots, x_n$ from some distribution P , estimate P .

Downsides:

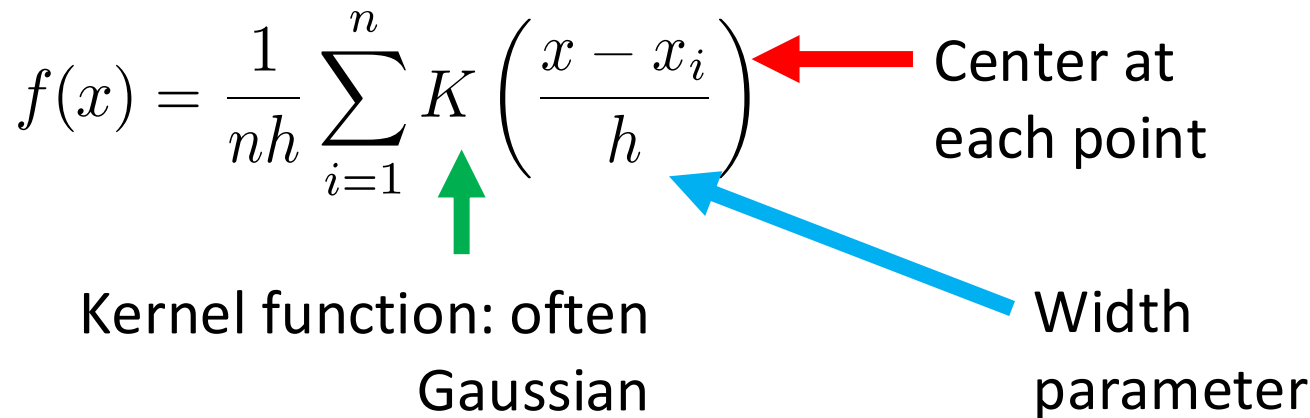
- i) High-dimensions: most bins empty
- ii) Not continuous



Kernel Density Estimation

Goal: given samples $x_1, \dots, x_n$ from some distribution P , estimate P .

Idea: represent density as combination of “kernels”

$$f(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$


Center at each point

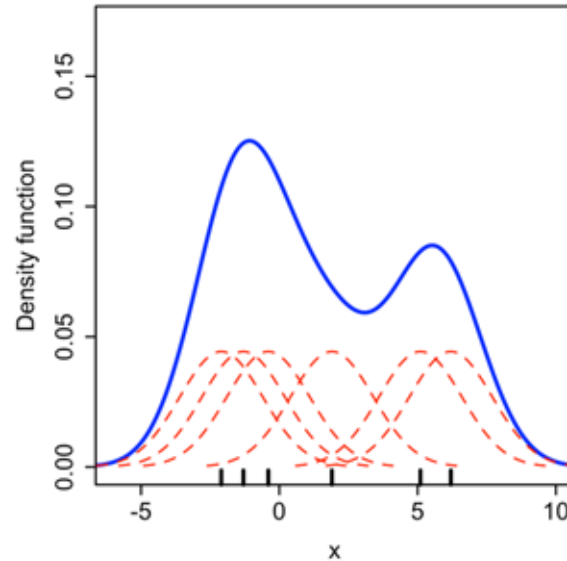
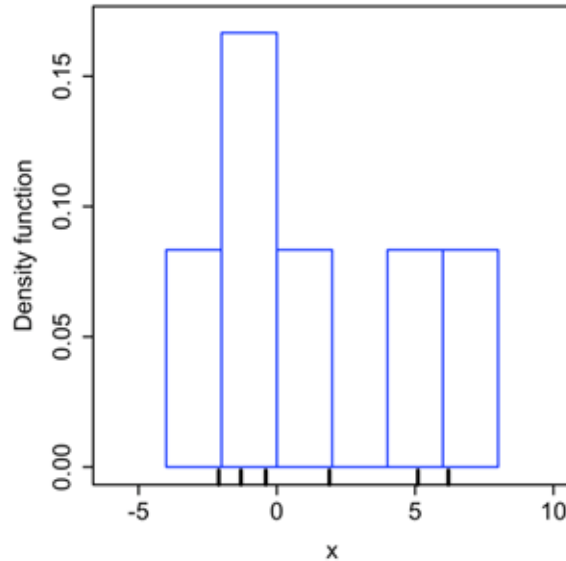
Kernel function: often Gaussian

Width parameter

Kernel Density Estimation

Idea: represent density as combination of kernels

- “Smooth” out the histogram



Suggested reading

- A Tutorial on Spectral Clustering, Ulrike von Luxburg
https://people.csail.mit.edu/dsontag/courses/ml14/notes/Luxburg07_tutorial_spectral_clustering.pdf
- Textbook: Artificial Intelligence: A Modern Approach (4th edition). Stuart Russell and Peter Norvig. Pearson, 2020.
 - Section 19.9 Exploratory data analysis and visualization
 - Section 20.2 Introduction