CS 540 Introduction to Artificial Intelligence

Unsupervised Learning II

University of Wisconsin-Madison

Fall 2022

Announcements

- **Homeworks**:
  - HW4 out, HW2 grades released tonight
  - Midterm: online
- Class roadmap:

Thursday, Oct. 6	ML Unsupervised II
Tuesday, Oct. 11	ML Linear Regression
Thursday, Oct. 13	Machine Learning: K - Nearest Neighbors & Naive Bayes
Tuesday, Oct. 18	Machine Learning: Neural Networks I (Perceptron)

Machine Learning

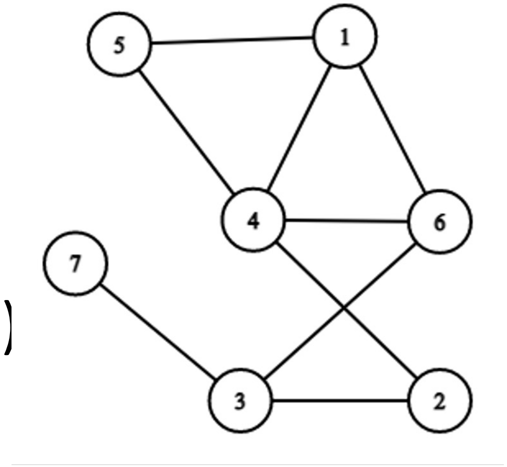
Outline

- Finish up Other 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

Other Types of Clustering

Graph-based/proximity-based

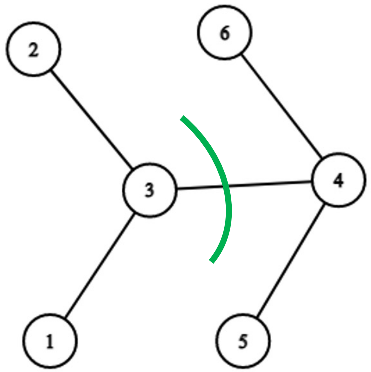
- Recall: Graph G = (V,E) has vertex set V, edge set E.
  - Edges can be weighted or unweighted
  - Encode **similarity:** $w_{ij} = \mathrm{sim}(v_i, v_j)$
- Don't need to KEEP vectors v
  - Only keep the edges (possibly weighted)



Graph-Based Clustering

Want: partition V into V_1 and V_2

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



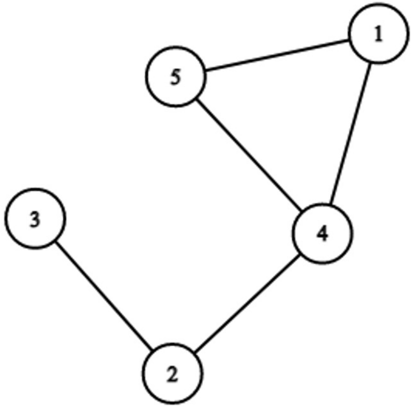
$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

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

Partition-Based Clustering

How do we compute these?

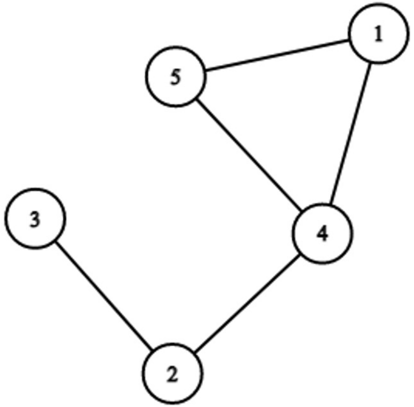
- Hard problem → heuristics
  - Greedy algorithm
  - "Spectral" approaches
- Spectral clustering approach:
  - **Adjacency** matrix



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

Partition-Based Clustering

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

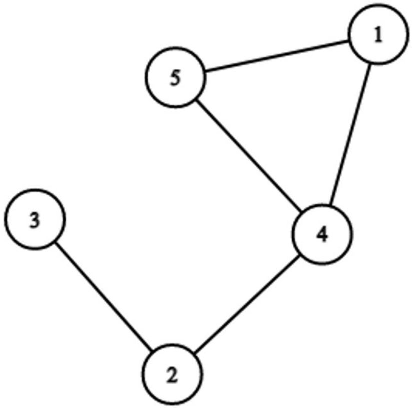


$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \qquad A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Spectral Clustering

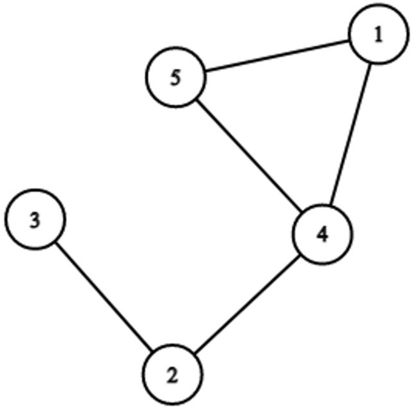
- Spectral clustering approach:
  - 1. Compute **Laplacian** L = D – A

(Important tool in graph theory)



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

Spectral Clustering

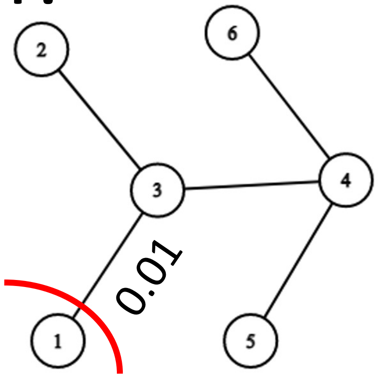


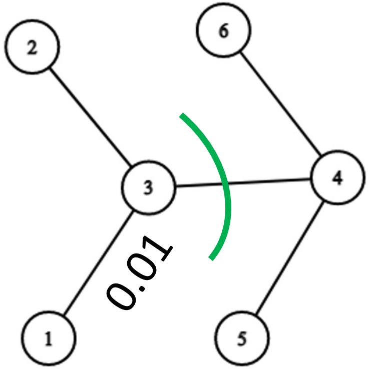
- Spectral clustering approach:
  - 1. Compute **Laplacian** $L = D - A$
  - 1a (optional): compute normalized Laplacian:
    $L = I - D^{-1/2}AD^{-1/2}$, or $L = I - D^{-1}A$
  - 2. Compute *k* **smallest** eigenvectors of L
  - 3. Set U to be the n x k matrix with $u_1, \ldots, u_k$ as columns. Take the n rows formed as points
  - 4. Run k-means on the representations

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 of 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!**

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

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

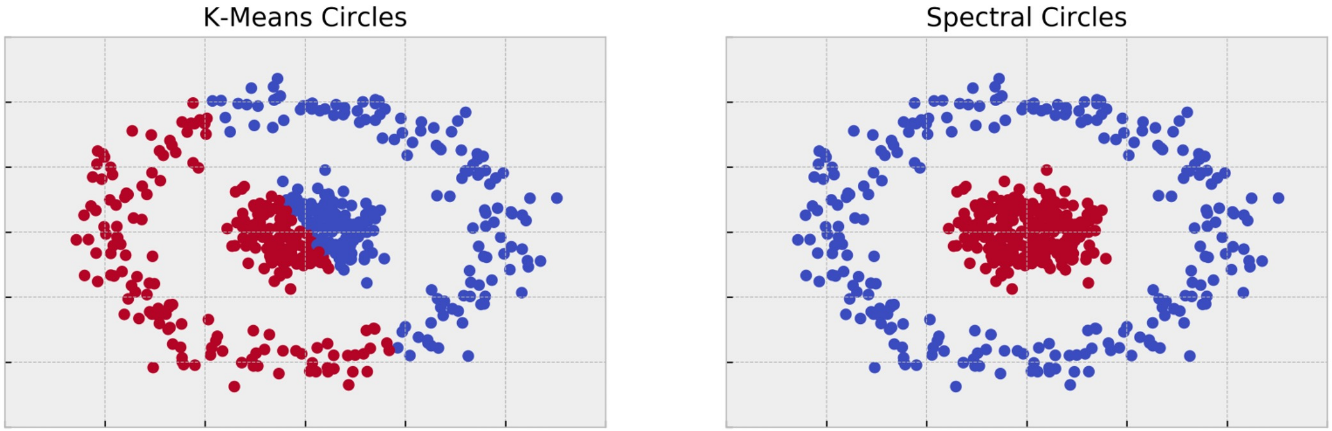
Spectral Clustering

- Compare/contrast to **PCA**:
  - Use an **eigendecomposition /** dimensionality reduction
    - But, run on Laplacian (not covariance); use smallest eigenvectors, not largest
- Intuition: Laplacian encodes structure information
  - "Lower" eigenvectors give partitioning information

Spectral Clustering

Q: Why do this?

- 1. No need for points or distances as input
- 2. Can handle intuitive separation (k-means can't!)



Credit: William Fleshman

Break & Quiz

Q 1.1: We have two datasets: a social network dataset S_1 which shows which individuals are friends with each other along with image dataset $S_{2.}$
What kind of clustering can we do? Assume we do not make additional data transformations.

- A. k-means on both S_1 and S_2
- B. graph-based on S_1 and k-means on S_2
- C. k-means on S_1 and graph-based on S_2
- D. hierarchical on S_1 and graph-based on S_2

Break & Quiz

Q 1.1: We have two datasets: a social network dataset S_1 which shows which individuals are friends with each other along with image dataset $S_{2.}$

What kind of clustering can we do? Assume we do not make additional data transformations.

- A. k-means on both S_1 and S_2
- **B. graph-based on S_1 and k-means on S_2**
- C. k-means on S_1 and graph-based on S_2
- D. hierarchical on S_1 and graph-based on S_2

Break & Quiz

Q 1.1: We have two datasets: a social network dataset S_1 which shows which individuals are friends with each other along with image dataset S_2.

What kind of clustering can we do? Assume we do not make additional data transformations.

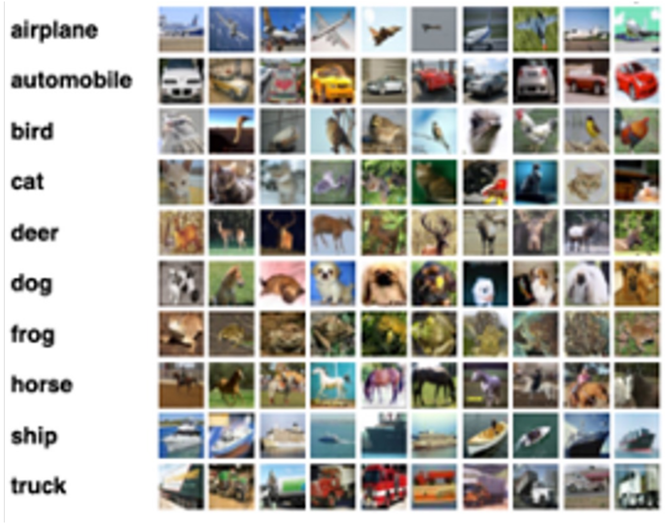
- A. k-means on both S_1 and S_2 **(No: can't do k-means on graph)**
- **B. graph-based on S_1 and k-means on S_2**
- C. k-means on S_1 and graph-based on S **(Same as A)**
- D. hierarchical on S_1 and graph-based on S_2 **(No: S_2 is not a graph)**

Break & Quiz

Q 1.2: The CIFAR-10 dataset contains 32x32 images labeled with one of 10 classes. What could we use it for?

(i) Supervised learning (ii) PCA (iii) k-means clustering

- A. Only (i)
- B. Only (ii) and (iii)
- C. Only (i) and (ii)
- D. All of them



Break & Quiz

Q 1.2: The CIFAR-10 dataset contains 32x32 images labeled with one of 10 classes. What could we use it for?

(i) Supervised learning (ii) PCA (iii) k-means clustering

- A. Only (i)
- B. Only (ii) and (iii)
- C. Only (i) and (ii)
- **D. All of them**

Break & Quiz

Q 1.2: The CIFAR-10 dataset contains 32x32 images labeled with one of 10 classes. What could we use it for?

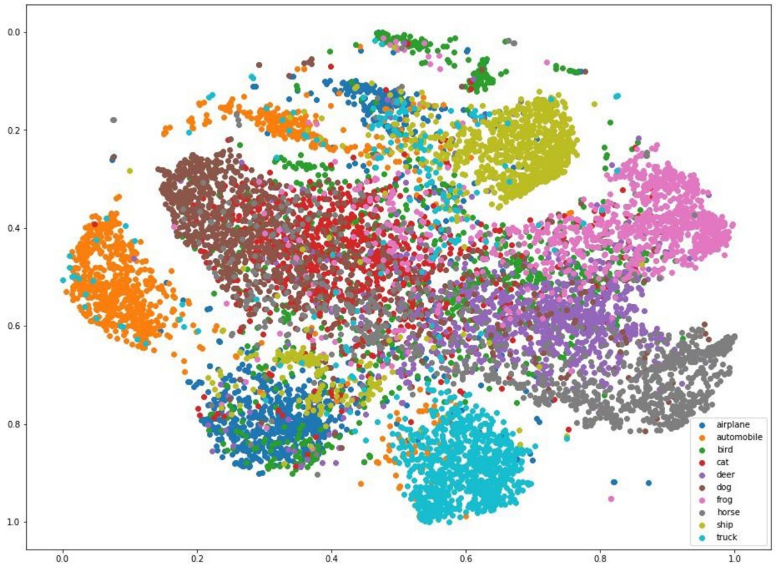
(i) Supervised learning (ii) PCA (iii) k-means clustering

- (i) **Yes: train an image classifier; have labels)**
- (ii) **Yes: run PCA on image vectors to reduce dimensionality**
- (iii) **Yes: can cluster image vectors with k-means**
- **D. All of them**

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 & Visualization

Run PCA on MNIST

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

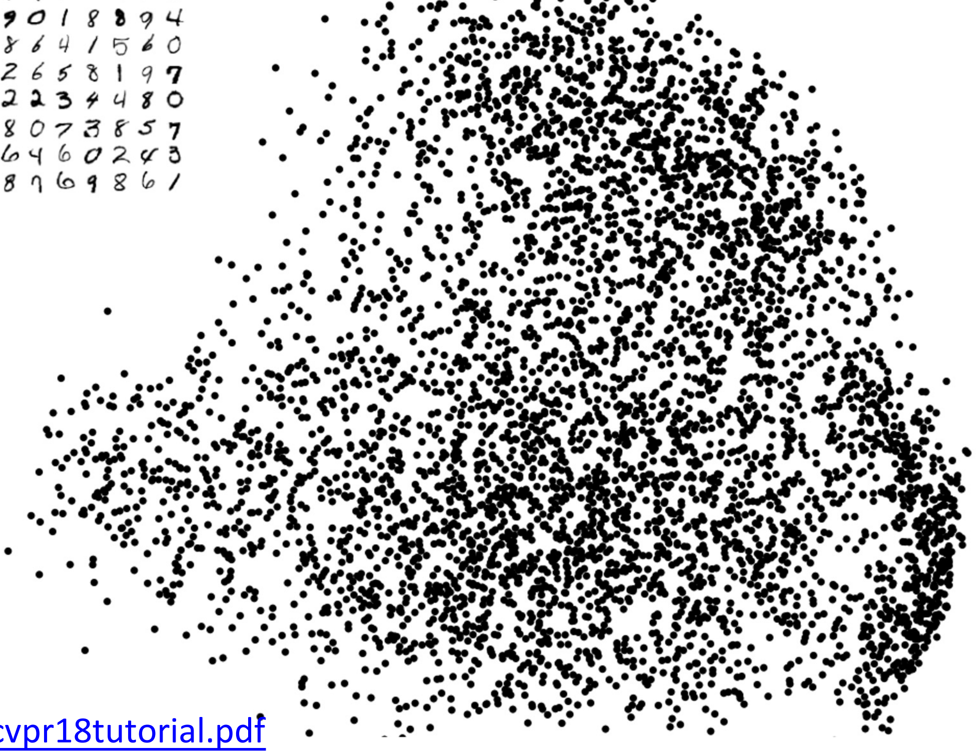


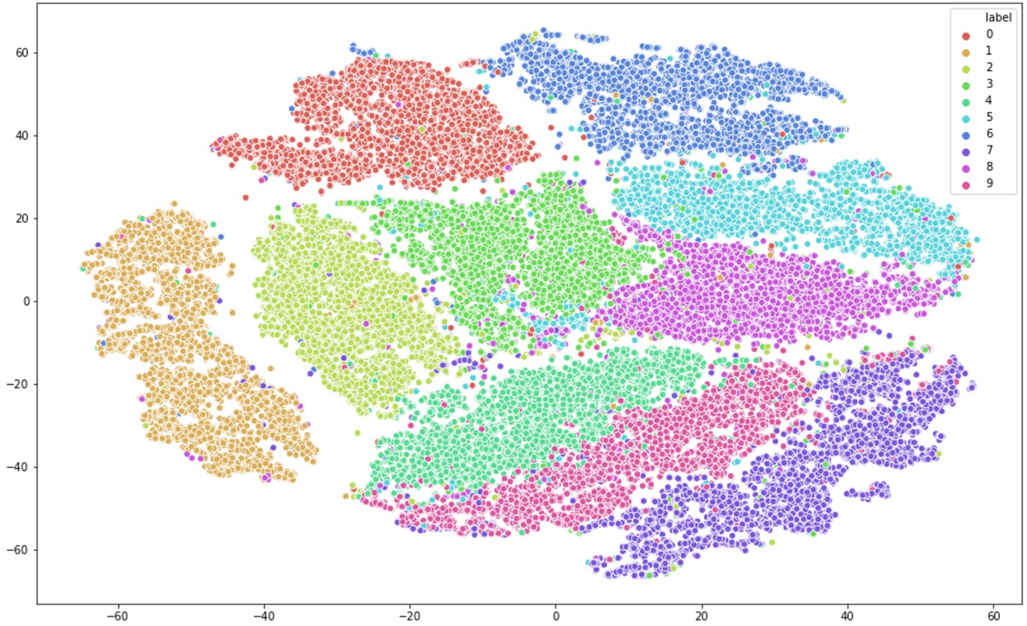
Image source:
http://deeplearning.csail.mit.edu/slide_cvpr2018/laurens_cvpr18tutorial.pdf

Visualization: **T-SNE**

Typical dataset: MNIST

- **T-SNE**: project data into just 2 dimensions
- Try to maintain structure

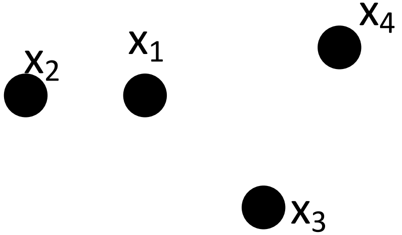
- MNIST Example
- **Input**: $x_1, x_2, \ldots, x_n$
- **Output**: 2D/3D $y_1, y_2, \ldots, y_n$



T-SNE Algorithm: Step 1

How does it work? Two steps

- **1.** Turn vectors into probability pairs
- **2**. Turn pairs back into **(lower-dim)** vectors



Step 1:

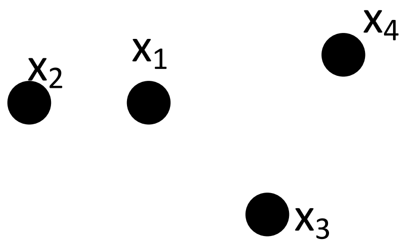
$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)} \quad p_{ij} = \frac{1}{2n}(p_{j|i} + p_{i|j})$$

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

T-SNE Algorithm: Step 2

How does it work? Two steps

- **1.** Turn vectors into probability pairs
- **2**. Turn pairs back into **(lower-dim)** vectors



Step 2: set

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

and minimize

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

KL 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)
- **Movies**: https://lvdmaaten.github.io/tsne/examples/netflix_tsne.jpg

T-SNE Examples

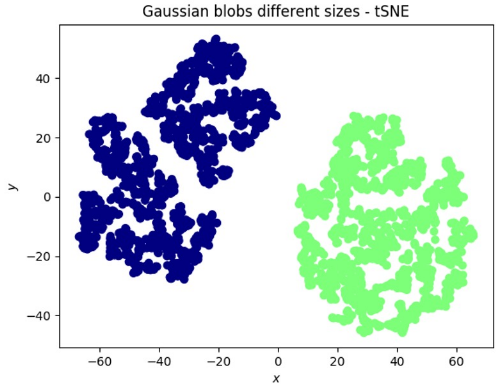
- Examples: (from Laurens van der Maaten)
- **NORB**: https://lvdmaaten.github.io/tsne/examples/norb_tsne.jpg

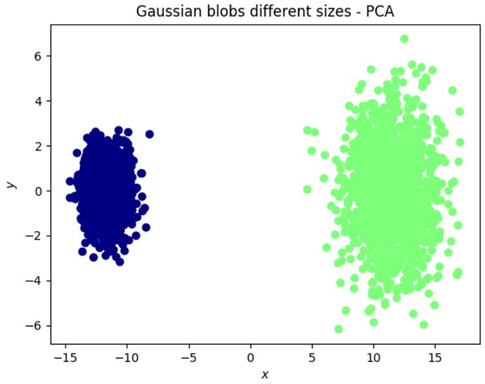
Visualization: **T-SNE**

t-SNE vs PCA?

- "Local" vs "Global"
- Lose information in t-SNE
  - not a bad thing necessarily
- Downstream use

Good resource/credit:
https://www.thekerneltrip.com/statistics/tsne-vs-pca/





Break & Quiz

Q 2.1: Can we do t-SNE on NLP (words) or graph datasets?

- A. Never
- B. Yes, after running PCA on them
- C. Yes, after mapping them into R^d (ie, embedding)
- D. Yes, after running hierarchical clustering on them

Break & Quiz

Q 2.1: Can we do t-SNE on NLP (words) or graph datasets?

- A. Never
- B. Yes, after running PCA on them
- **C. Yes, after mapping them into R^d (ie, embedding)**
- D. Yes, after running hierarchical clustering on them

Break & Quiz

Q 2.1: Can we do t-SNE on NLP (words) or graph datasets?

- A. Never **(No: too strong)**
- B. Yes, after running PCA on them **(No: can't run PCA on words or graphs directly. Need vectors)**
- **C. Yes, after mapping them into R^d (ie, embedding)**
- D. Yes, after running hierarchical clustering on them **(No: hierarchical clustering gives us a graph)**

Short Intro to Density Estimation

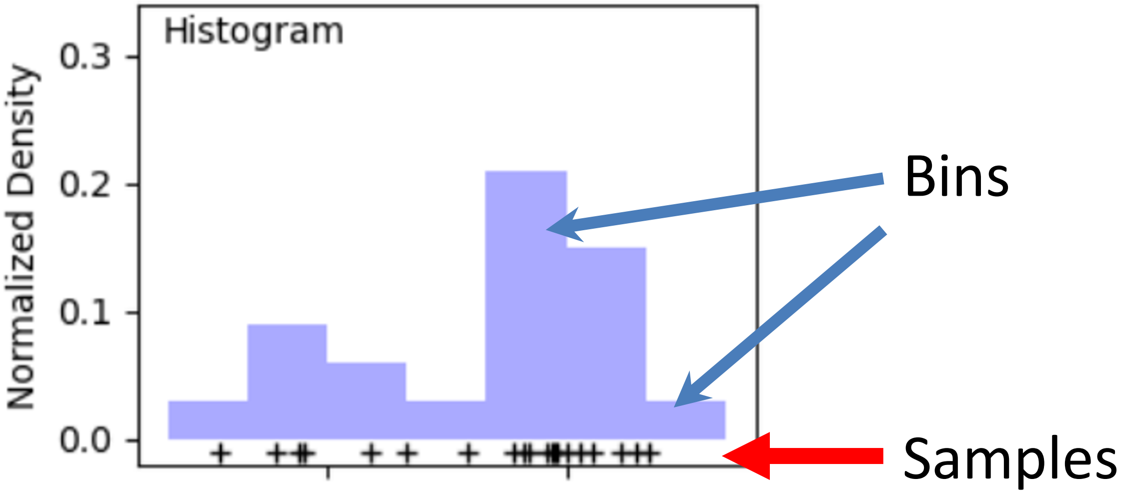
Goal: given samples $x_1, \ldots, x_n$ from some distribution P, estimate P.

- Compute statistics (mean, variance)
- Generate samples from P
- Run inference

Zach Monge

Simplest Idea: Histograms

Goal: given samples x_1, …, x_n from some distribution P, estimate P.



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

Simplest Idea: Histograms

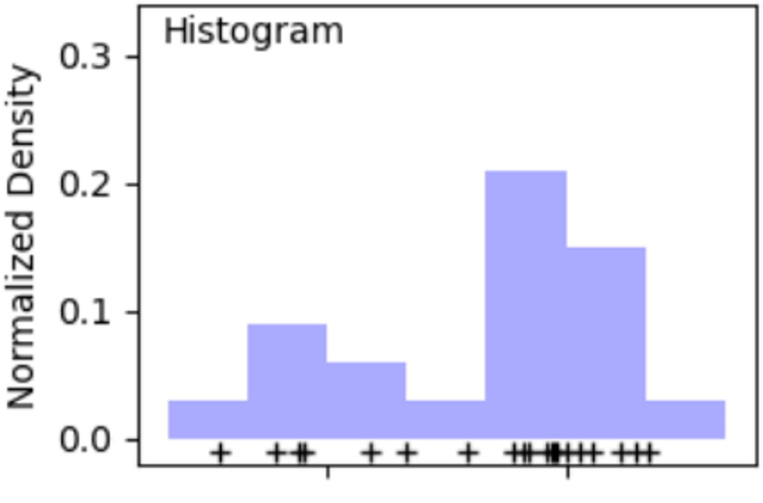
Goal: given samples $x_1, \ldots, x_n$ from some distribution P, estimate P.

Downsides:

i) High-dimensions: most bins empty

ii) Not continuous

iii) How to choose bins?



Kernel Density Estimation

Goal: given samples x_1, …, 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