



# CS 540 Introduction to Artificial Intelligence

## **Unsupervised Learning II**

Yudong Chen  
University of Wisconsin-Madison

**Oct 7, 2021**

# Announcements

- **Homework:**
  - HW4 due Tuesday
- **Class roadmap:**

Thursday, Sep 30	ML Intro
Tuesday, Oct 5	ML Unsupervised I
Thursday, Oct 7	ML Unsupervised II
Tuesday, Oct 12	ML Linear Regression
Thursday, Oct 14	ML: KNN, Naïve Bayes

Machine Learning

# HW 3 Recap

## First three methods

```
def load_and_center_dataset(filename):  
    x = np.load(filename)  
    mu = np.mean(x, axis=0)  
    return x - mu
```

← Centering

```
def get_covariance(dataset):  
    n = len(dataset)  
    return np.dot(np.transpose(dataset), dataset) / (n - 1)
```

← Matrix multiplication  $X^T X$

```
def get_eig(S, m):  
    w, v = eig(S, eigvals=(len(S) - m, len(S) - 1))  
    return np.diag(w[::-1]), np.fliplr(v)
```

$$S = \frac{1}{n-1} \sum_{i=1}^n x_i x_i^T = \frac{1}{n-1} X^T X$$

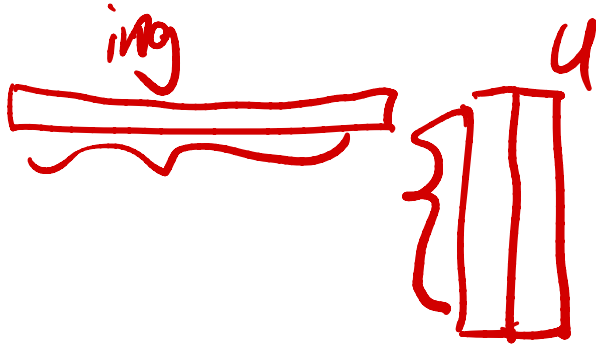
# HW 3 Recap

## Projecting and displaying the image

```
def project_image(img, U):  
    alpha = np.dot(img, U)  
    xapprox = np.dot(U, alpha)  
    return xapprox
```

← New representation for the image

← Reconstructed image

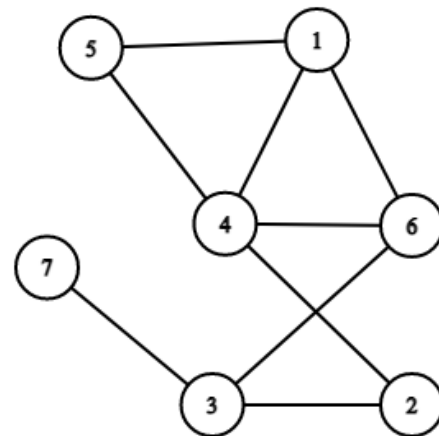


# Outline

- Recap: graph clustering, cuts, spectral method
- Unsupervised Learning: Visualization
  - t-SNE, algorithm, example, vs. PCA
- Unsupervised Learning: Density Estimation
  - Kernel density estimation: high-level intro

# Graph/proximity-based Clustering

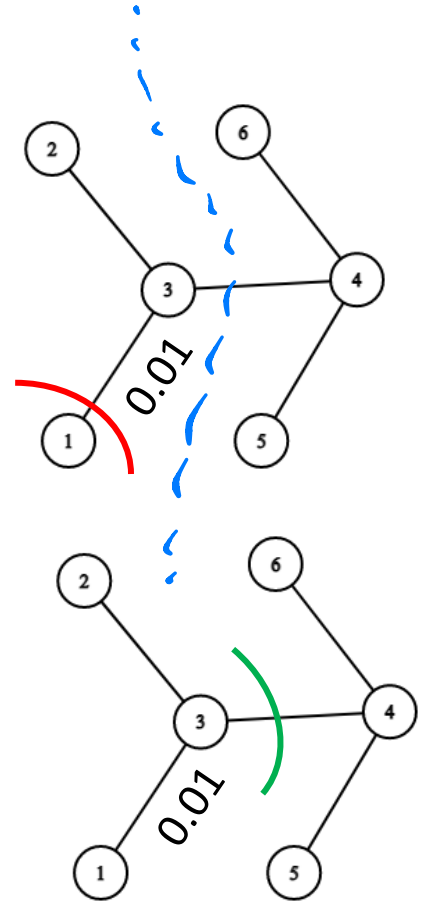
- Recall: Graph  $G = (V, E)$  has vertex set  $V$ , edge set  $E$ .
  - Edges can be weighted or unweighted
  - Encode **similarity**
- Don't need vectors here
  - Just edges (and maybe weights)



# Minimum Cuts

**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



# Minimizing Cuts

**Want:** partition  $V$  into  $V_1$  and  $V_2$

- Just minimizing weight isn't good... want **balance!**
- **Better approach:** minimize one of the following

$$\text{CCut}(V_1, V_2) = \frac{\text{Cut}(V_1, V_2)}{|V_1|} + \frac{\text{Cut}(V_1, V_2)}{|V_2|} \quad \leftarrow \text{\#vertices in } V_2$$

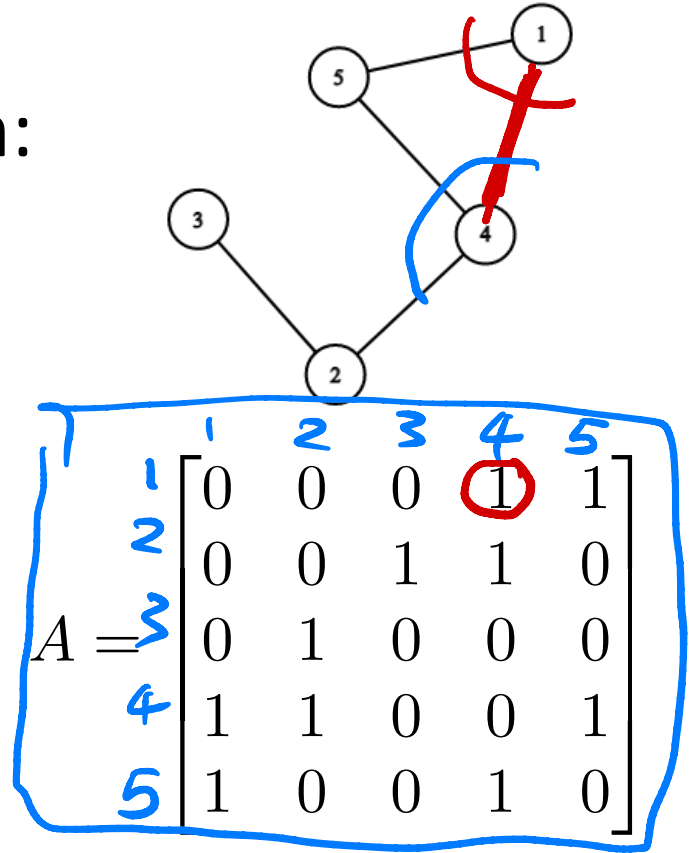
$$\text{NCut}(V_1, V_2) = \frac{\text{Cut}(V_1, V_2)}{\sum_{i \in V_1} d_i} + \frac{\text{Cut}(V_1, V_2)}{\sum_{i \in V_2} d_i} \quad \leftarrow \text{total \#edges at vertices in } V_2$$



# Spectral Clustering

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

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



# Spectral Clustering

- Spectral clustering approach:
  - 1. Compute **Laplacian**  $L = D - A$   
(Important tool in graph theory)

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

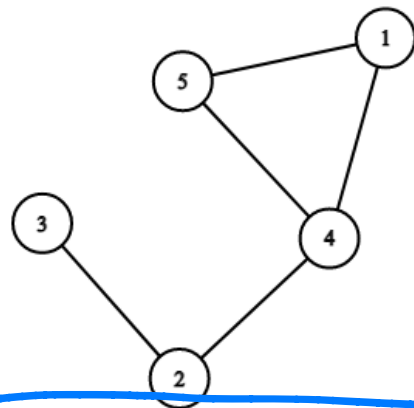
**Degree Matrix**

$$- \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

**Adjacency Matrix**

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

**Laplacian**



# Spectral Clustering

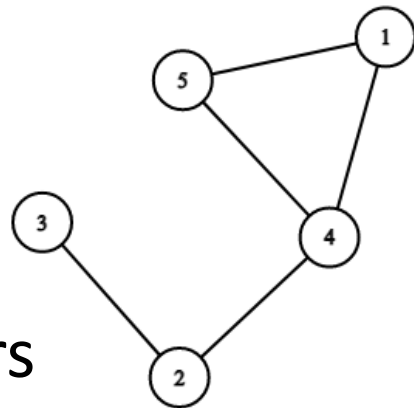
- Spectral clustering approach:

- 1. Compute **Laplacian**  $L = D - A$

- 2. Compute  $k$  **smallest** eigenvectors

- 3. Set  $U$  to be the  $n \times k$  matrix with  $u_1, \dots, u_k$  as columns. Treat  $n$  rows as  $n$  points/vectors in  $\mathbb{R}^k$

- 4. Run k-means on these  $n$  points/vectors



$u =$

# Spectral Clustering

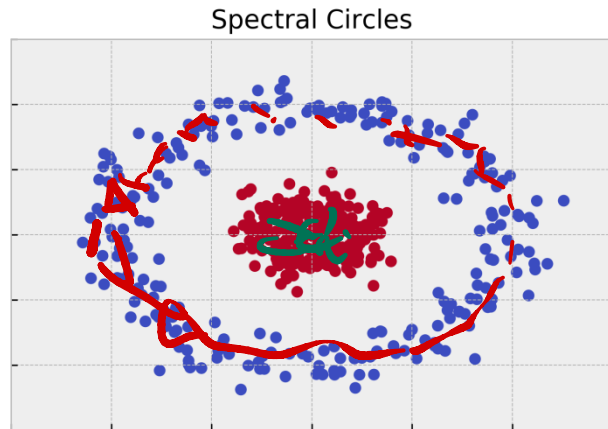
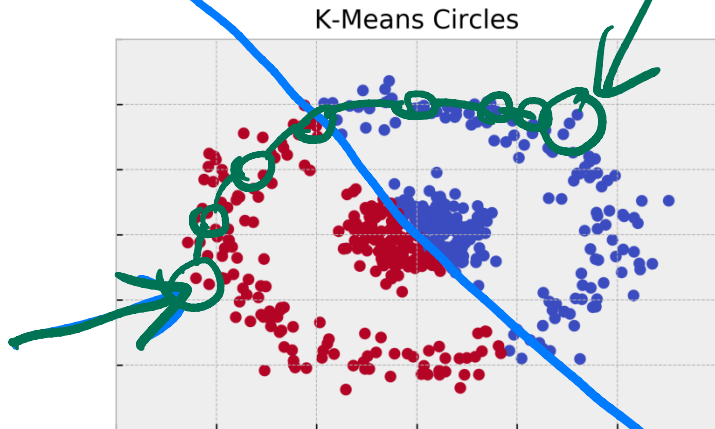
- Compare/contrast to **PCA**:
  - Use an **eigendecomposition** / dimensionality reduction
    - But, run on Laplacian (not covariance); use smallest eigenvectors, not largest
- Intuition: Laplacian encodes structure information
  - “Lower” eigenvectors give partitioning information

# Spectral Clustering

edge b/w  $i, j$  if  $d(x_i, x_j) \leq \underline{0.01}$   
edge b/w  $i, j$  if  $x_i$  is nearest neighbor of  $x_j$

Q: Why do this?

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



Credit: William Fleishman

# Break & Quiz

**Q 1.1:** We have two datasets: a social network dataset  $S_1$  which shows which individuals are friends with each other, and an 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, and an 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, and an 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?

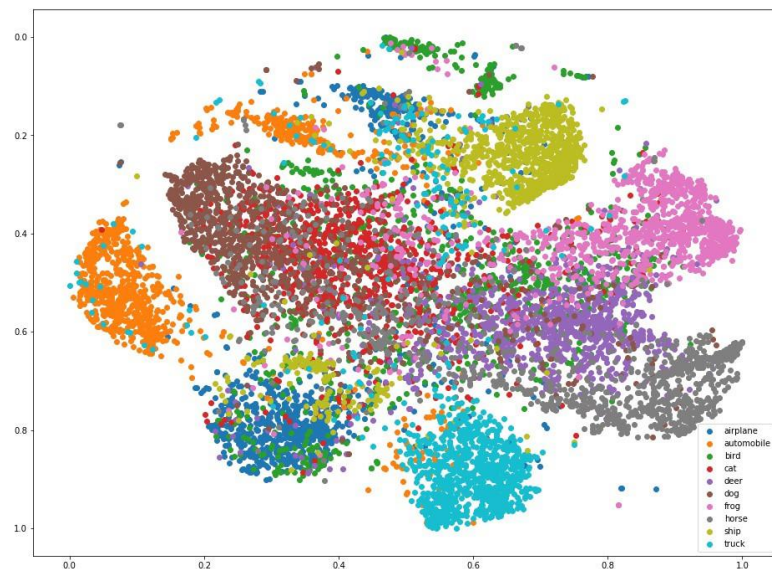
(i) Supervised learning (ii) PCA (iii) k-means clustering

- (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

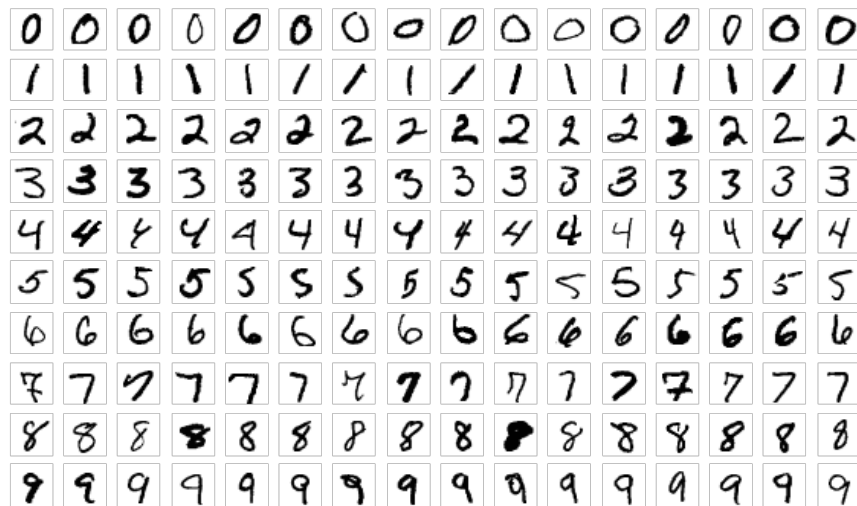


Philip Slingerland

# 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?

# Visualization: T-SNE

Typical dataset: MNIST

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

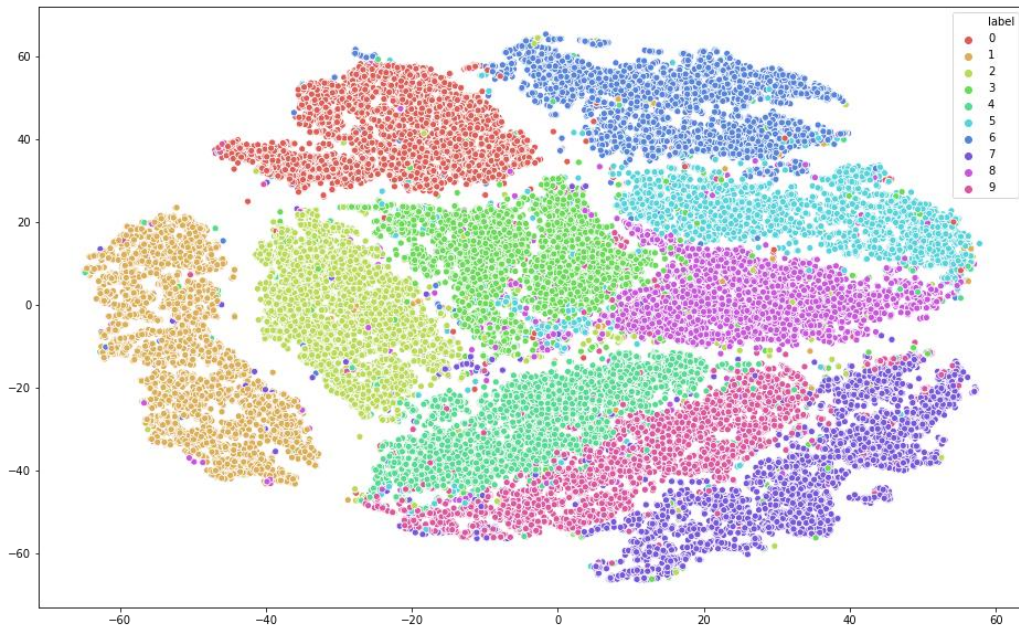
- MNIST Example

- **Input:**  $x_1, x_2, \dots, x_n$

- **Output:** 2D/3D  $y_1, y_2, \dots, y_n$

$\in \mathbb{R}^{786}$

$\in \mathbb{R}^2 \mathbb{R}^3$



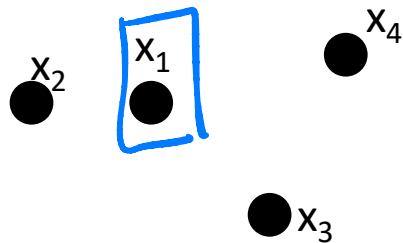
$x_i \rightarrow p_{ij} \rightarrow y_i \in \mathbb{R}^2$   
 $\in \mathbb{R}^{786}$

# T-SNE Algorithm: Step 1

$i=1$

How does it work? Two steps

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



Step 1:

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

$e^{-\|x_i - x_j\|^2 / 2\sigma_i^2}$   
*big*  $p_{2|1}$      $p_{3|1}$

$$p_{ij} = \frac{1}{2n} (p_{j|i} + p_{i|j})$$

*small*  $p_{4|1}$

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

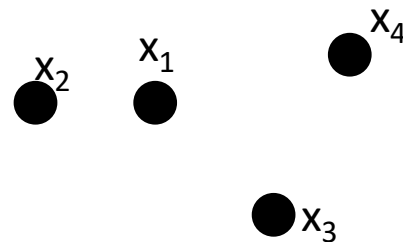
$p_{ij}$ 's

# T-SNE Algorithm: Step 2

$$y_i \in \mathbb{R}^2 \quad \mathbb{R}^3$$

How does it work? Two steps

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



Step 2: set

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

and minimize

$$\sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

$\log 1 = 0$



KL Divergence between p and q

$\approx$  closeness b/w p, q.

$$p_{j|i} \neq q_{j|i}$$



# T-SNE Algorithm: Step 2

More on step 2:

- We have two distributions  $p, q$ .  $p$  is fixed  $\{y_i\}$
- $q$  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.

$$\min_{\{y_i\}} \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$



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](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](https://lvdmaaten.github.io/tsne/examples/norb_tsne.jpg)



$(x_i, y_i), i=1, \dots, n \xrightarrow{\text{PCA}} (\bar{x}_i, \bar{y}_i) \xrightarrow{\text{NB}} \text{classifier.}$

## Visualization: T-SNE

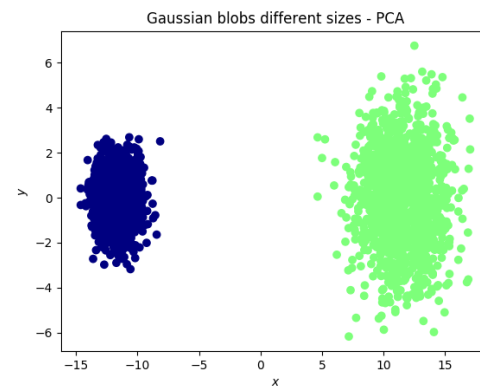
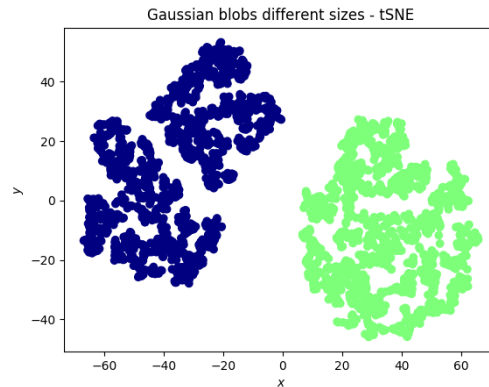
$(x_i, y_i), i=1, \dots, n \left\{ \begin{array}{l} C_1 \quad f_1 \\ C_2 \quad f_2 \end{array} \right.$

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  $\mathbb{R}^d$  (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 tree**)

UL { dim reduction (visualization)  
clustering:  
density estimation (generative modeling)

# Short Intro to Density Estimation

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

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



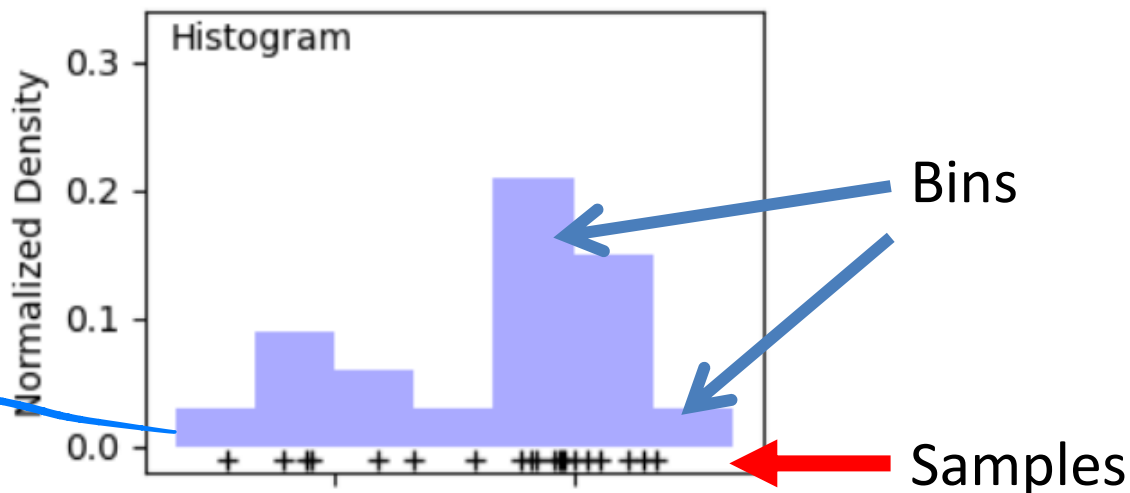
Zach Monge



height of bar = # pts in bin / total # pts.

## Simplest Idea: Histograms

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



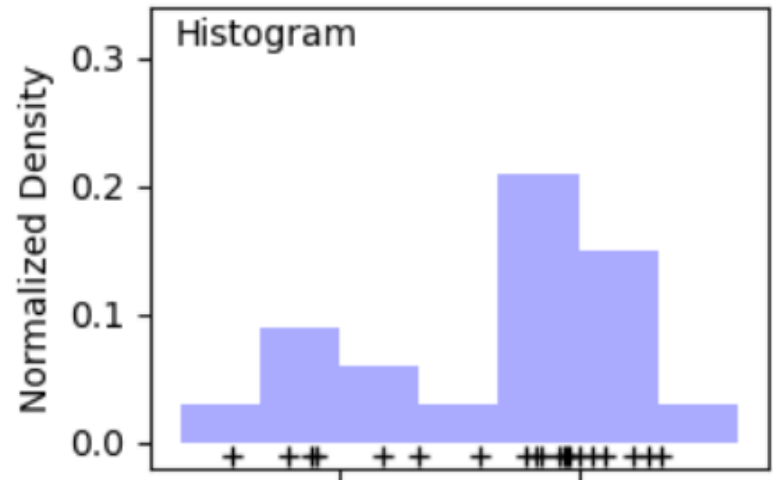
Define bins; count # of samples in each bin, normalize

# 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
- iii) How to choose bins?



# Kernel Density Estimation

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

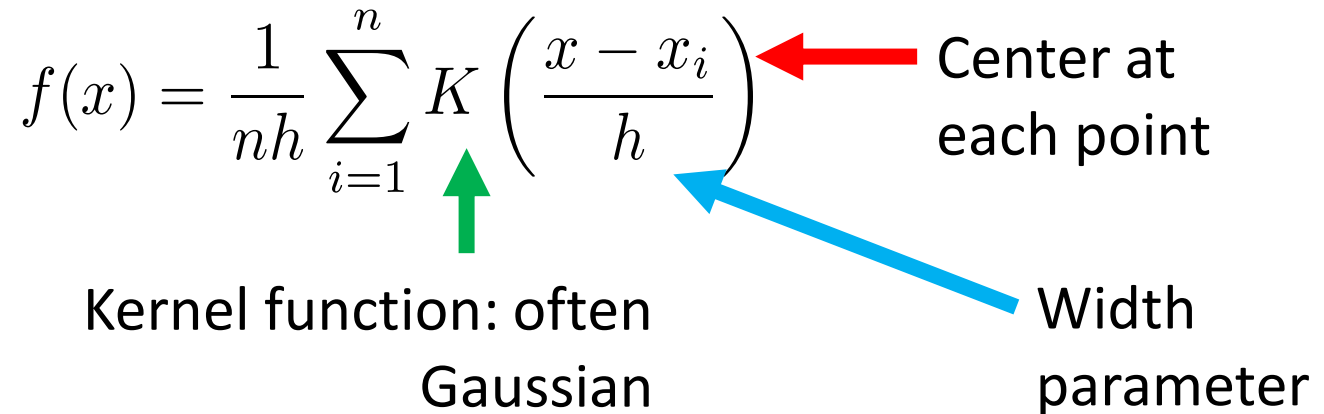
**Idea:** represent density as combination of “kernels”

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

Center at each point

Kernel function: often Gaussian

Width parameter

The diagram shows the kernel density estimation formula with three colored arrows pointing to specific parts of the equation. A red arrow points from the text 'Center at each point' to the  $x_i$  term in the denominator of the kernel function. A green arrow points from the text 'Kernel function: often Gaussian' to the  $K$  term. A blue arrow points from the text 'Width parameter' to the  $h$  term in the denominator of the kernel function.

# Kernel Density Estimation

**Idea:** represent density as combination of kernels

- “Smooth” out the histogram

