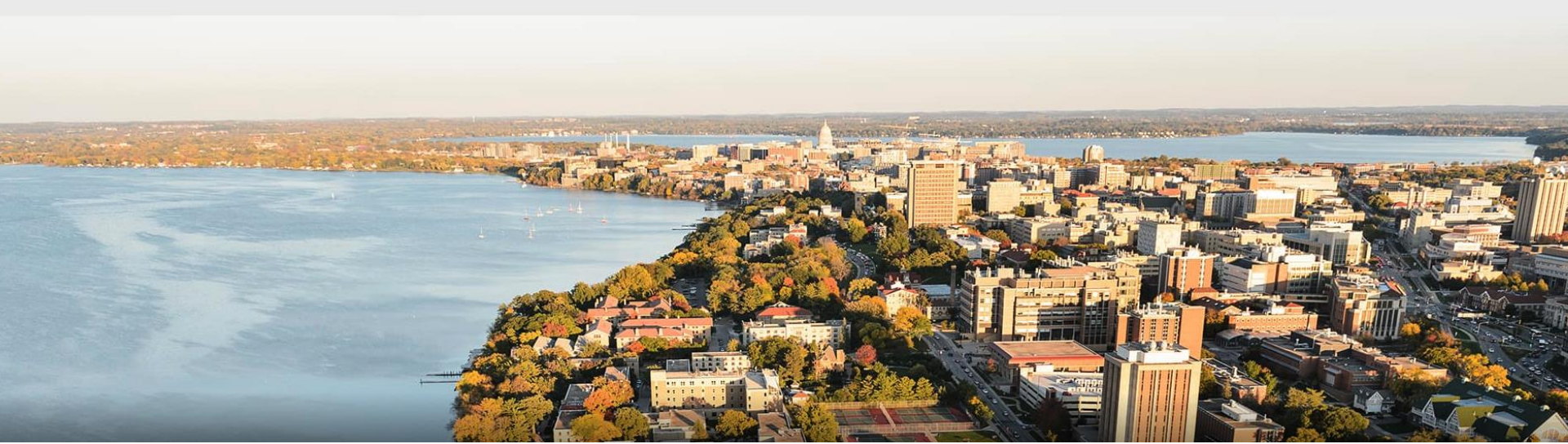


CS 540 Introduction to Artificial Intelligence
Unsupervised Learning II

Fred Sala
University of Wisconsin-Madison

Feb 23, 2021

Announcements

- **Homeworks**:
  - HW3 recap
  - HW4 released: clustering
- Class roadmap:

Thursday, Feb 18	ML Intro
Tuesday, Feb 23	**ML Unsupervised I**
Thursday, Feb 25	ML Linear Regression
Tuesday, March 2	ML: Naïve Bayes, Recap

Machine Learning

Outline

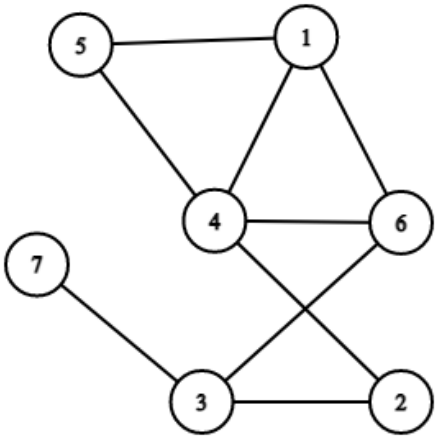
- Finish up Other Clustering Types
  - Graph-based, cuts, spectral clustering
- Unsupervised Learning: Visualization
  - t-SNE, algorithm, example, vs. PCA
- Unsupervised Learning: Density Estimation
  - Kernel density estimation: high-level intro

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**

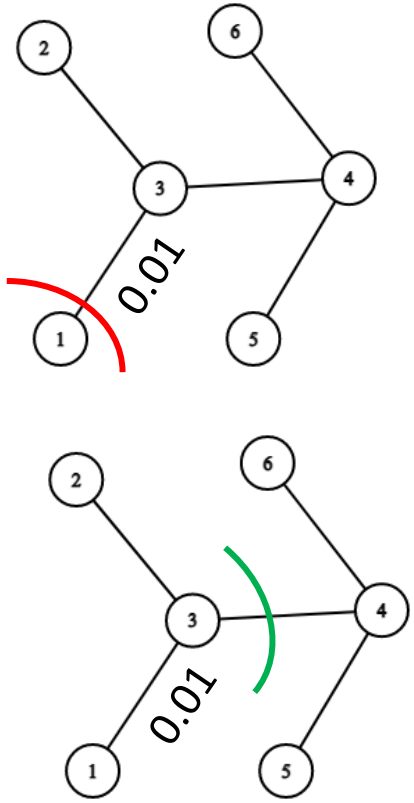
- Don’t need vectors here
  - Just edges (and maybe weights)



Graph-Based Clustering

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



Partition-Based Clustering

Want: partition V into V_1 and V_2

- Just minimizing weight isn't good… want **balance!**
- **Approaches:**

$$\mathrm{Cut}(V_1, V_2) = \frac{\mathrm{Cut}(V_1, V_2)}{|V_1|} + \frac{\mathrm{Cut}(V_1, V_2)}{|V_2|}$$

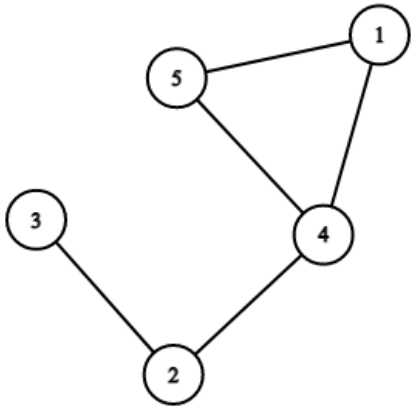
$$\mathrm{NCut}(V_1, V_2) = \frac{\mathrm{Cut}(V_1, V_2)}{\sum_{i \in V_1} d_i} + \frac{\mathrm{Cut}(V_1, V_2)}{\sum_{i \in V_2} d_i}$$

Sum of edge weights at vertex

Partition-Based Clustering

How do we compute these?

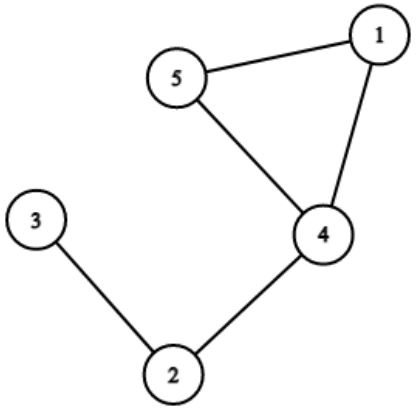
- Hard problem → heuristics
  - Greedy algorithm
  - “Spectral” approaches
- Spectral clustering approach:
  - **Adjacency** matrix



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

Partition-Based Clustering

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

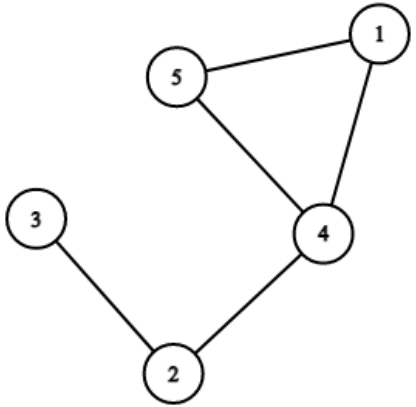


$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \quad A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Spectral Clustering

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

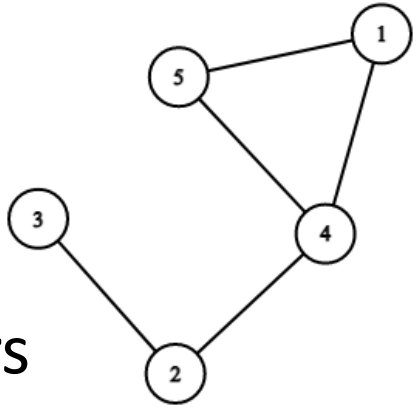
  (Important tool in graph theory)



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

Degree Matrix — Adjacency Matrix — Laplacian

Spectral Clustering



- Spectral clustering approach:
  - 1. Compute **Laplacian L = D – A**
  - 2. Compute *k* **smallest** eigenvectors
  - 3. Set *U* to be the *n* x *k* matrix with u_1, …, u_k as columns. Take the *n* rows formed as points
  - 4. Run k-means on the representations

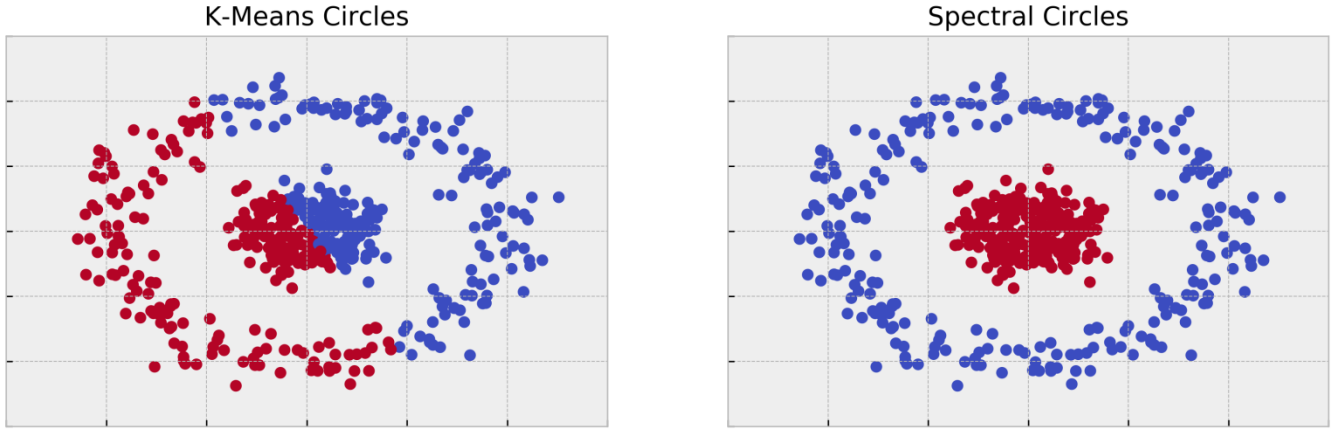
Spectral Clustering

- 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!)

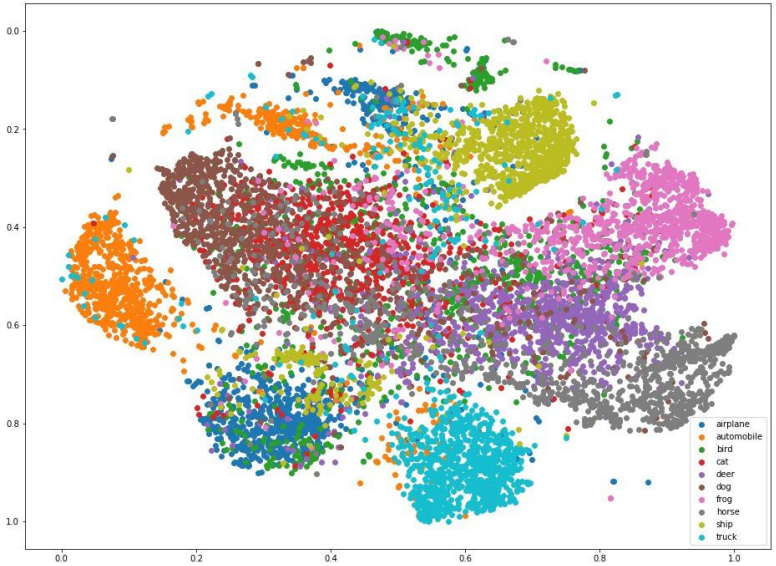


Credit: William Fleshman

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

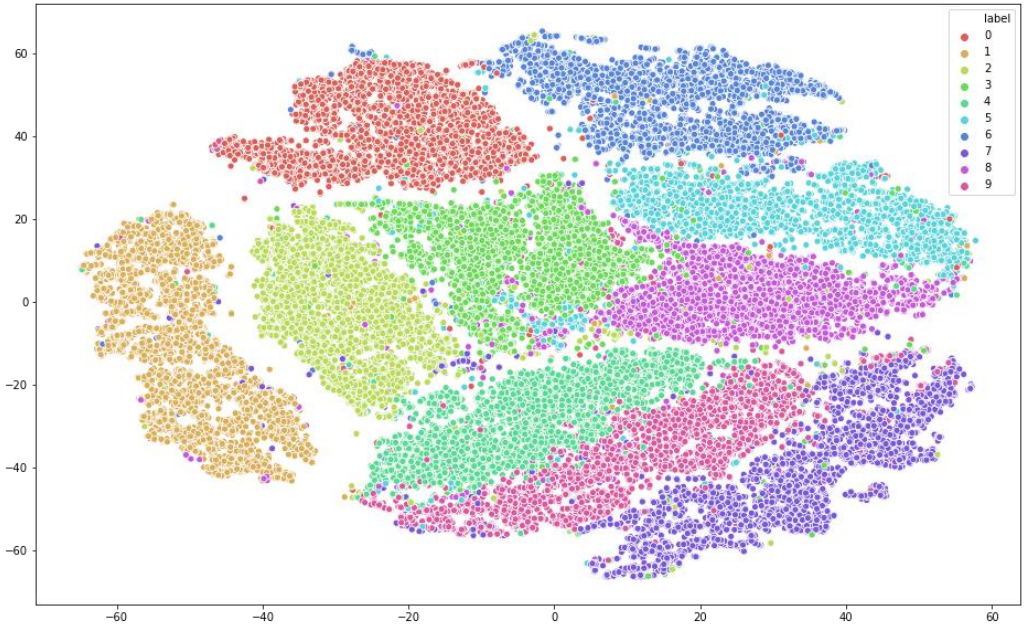
- Dimensionality reduction?

Visualization: **T-SNE**

Typical dataset: MNIST

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

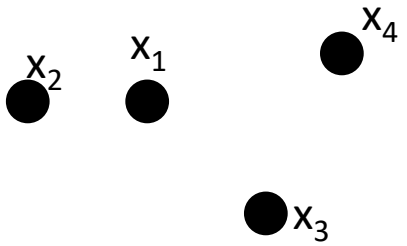
- MNIST Example
- **Input**: $x_1, x_2, \ldots, x_n$
- **Output**: 2D/3D $y_1, y_2, \ldots, y_n$



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:

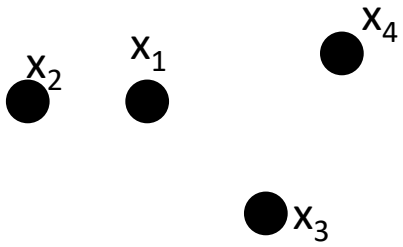
$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2/2\sigma_i^2)} \quad p_{ij} = \frac{1}{2n}(p_{j|i} + p_{i|j})$$

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

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

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!

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

KL Divergence between p and q

- **Optimizing a loss function**---we'll see more in supervised learning.

T-SNE Examples

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

T-SNE Examples

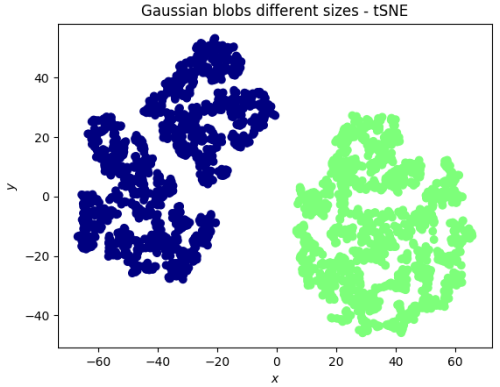
- Examples: (from Laurens van der Maaten)
- **NORB**: https://lvdmaaten.github.io/tsne/examples/norb_tsne.jpg

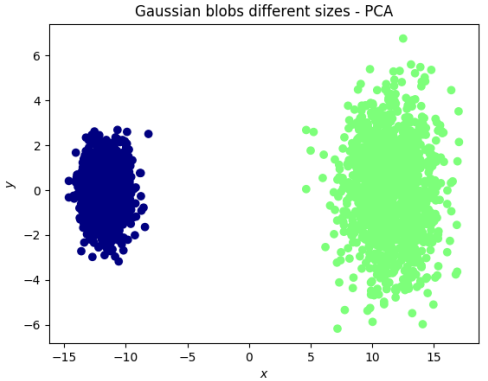
Visualization: **T-SNE**

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/





Short Intro to Density Estimation

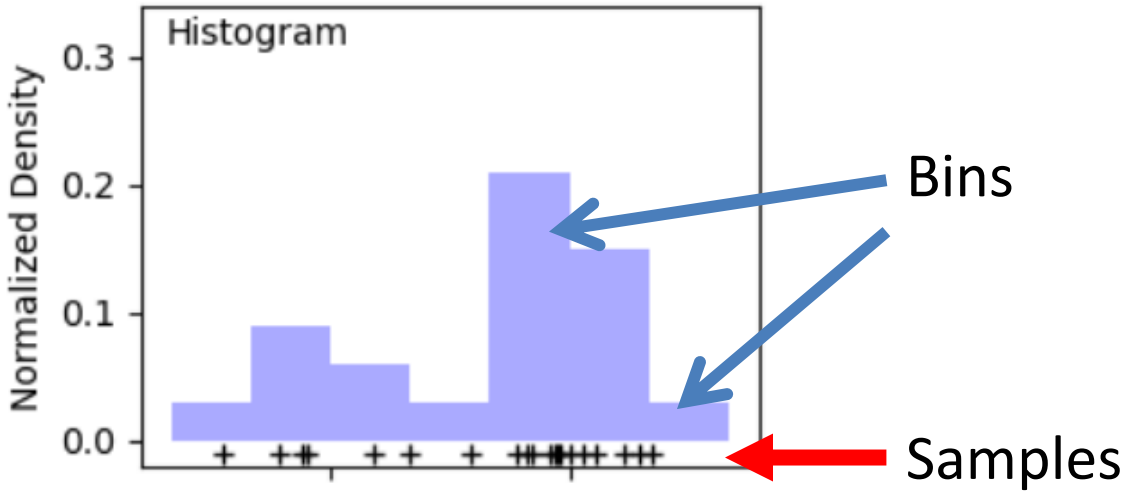
Goal: given samples x_1, …, x_n from some distribution P, estimate P.

- 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

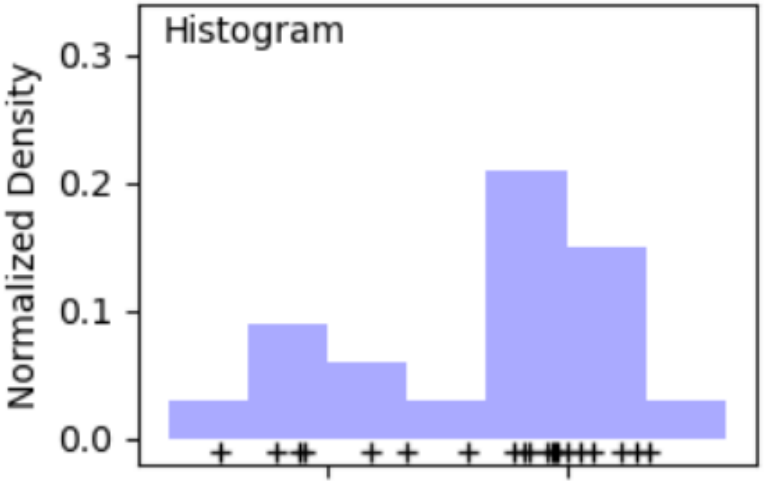
Goal: given samples x_1, …, x_n from some distribution *P*, estimate P.

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