Unsupervised

Lecture Notes

- ISL: 12.1, 12.2
- Relevant recordings from ISL:
  - 12.1 Principal components analysis https://youtu.be/kpuQqOzQXfM?si=jdTOzBq77rxkRfQo
  - 12.2. Higher order principle components https://youtu.be/O30nHhyBiAs?si=WjYxUOHEudLvHujh
- Statquest: https://www.youtube.com/c/joshstarmer
- Your friendly AI (have a conversation)

Introduction to Unsupervised Learning

Definition and Characteristics

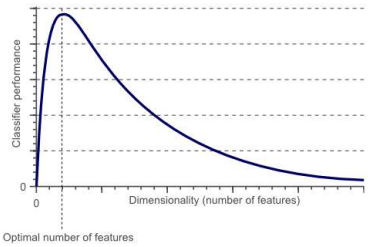
- Learning from unlabeled data $X = \{x_1, x_2, ..., x_n\}$
- Goal: Discover hidden patterns, structures, or relationships in data

Comparison with Other Learning Paradigms

- Supervised learning: $f : X \to Y$ (labeled data)
- Reinforcement learning: Agent-environment interaction
- Self-supervised learning: Creating labels from the data itself

Key Challenges

- No ground truth for direct evaluation (good or bad representation)
- Determining the optimal number of clusters/components
- Dealing with high-dimensional data ()



- $d(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$ each new dimensions adds new value= more distance

---

Types of Unsupervised Learning Tasks

1. Clustering
2. Dimension reduction (think: creating of linear vectors/models)
3. Association Rule Learning
4. Density learning, with our without given form
5. Generative models, then dimension reduction on the weights.

Clustering

- Definition: Grouping similar data points into clusters $C = \{C_1, C_2, ..., C_k\}$
- Types of clustering algorithms (Linda in BERN01 has done this very deeply:
  1. Exclusive (e.g., K-means, K-medoids) (more later)
  2. Overlapping (e.g., Fuzzy C-means, scikit-fuzzy)
     1. Each datapoint can belong to multiple clusters
     2. Membership values (percentages) $\sum membership = 1$
  3. Hierarchical
  4. Probabilistic

K-means Clustering

- Objective function: Minimize within-cluster sum of squares
  $J = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$
- Algorithm steps:
  1. Initialize k centroids randomly
  2. Assign points to nearest centroid
  3. Update centroids

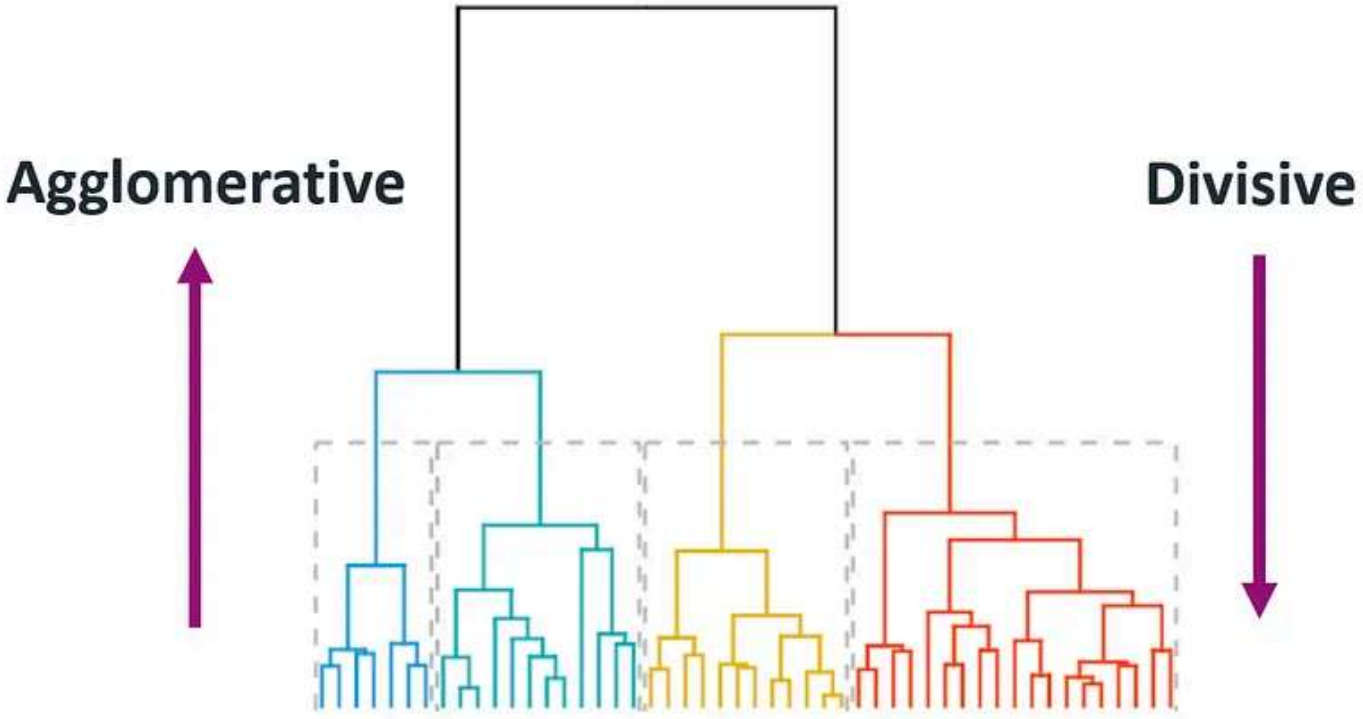
4. Repeat steps 2-3 until convergence

- Variants: K-means++, Mini-batch K-means
- Complexity: O(nkdi) where n = number of points, k = clusters, d = dimensions, i = iterations

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/
Dentrimer for checking how many clusters

Hierarchical (e.g., Agglomerative iterative merge, Divisive)

or the Art to go from each datapoint as its own cluster to through the steps of sorting and finding where sorting gets boring



https://www.linkedin.com/pulse/hierarchical-clustering-comprehensive-guide-understanding-massimo-re-r73gf

---

Linkage criteria:

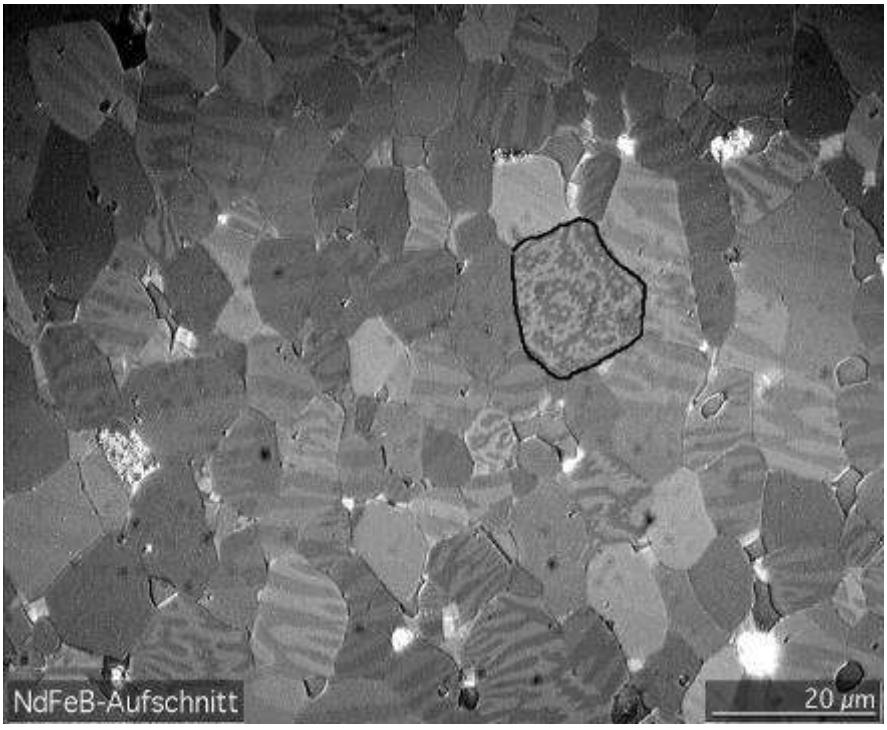
1. Single linkage, nearest neighbour (any member to any member) long structures:
$$d(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$$
2. Complete linkage(creates compact cluster, least sensitive to outliers):
$$d(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y)$$
3. Average linkage:
$$d(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{y \in C_j} d(x, y)$$

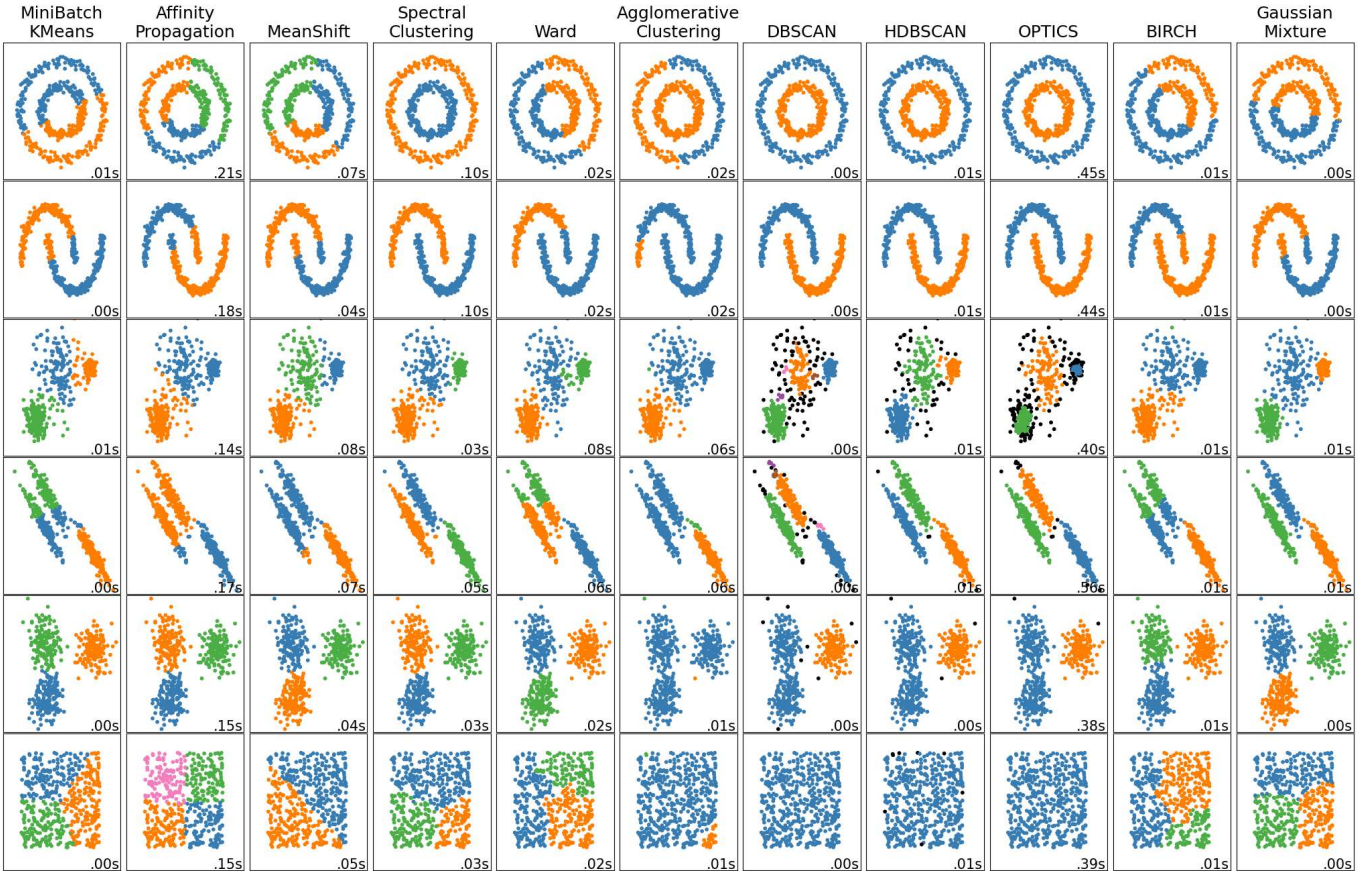
4. Ward's method: Minimize variance increase

1. Initialization: Start with each data point as its own cluster.
2. Distance calculation: Compute the pairwise distances between all clusters using squared Euclidean distance.
3. Merging process:
    - Find the two clusters that, when merged, result in the smallest increase in total within-cluster variance.
    - Combine these two clusters into a new cluster.
    - Update distances: Recalculate the distances between the new cluster and all other existing clusters.

---

4. Probabilistic (e.g., Gaussian Mixture Models)
6. Density-based (e.g., DBSCAN, OPTICS)
7. Grid-based (e.g., STING, CLIQUE)
8. Neural network-based (e.g., Self-Organizing Maps)





https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

---

Dimensionality Reduction

Purpose: Reducing feature space from d to k dimensions ($k < d$) while preserving information

1. Linear techniques (Linear Models/Vectors)
2. Non Linear techniques (Projections)
3. Neural network techniques

Linear techniques

-
    1. Principal Component Analysis (PCA) (maximize variance explained by vector)
    2. Linear Discriminant Analysis (LDA) (maximize distance between classes, minimize variance in class)
    3. Factor Analysis (starts with PCA, then maximizes variance, then adds rotation)

- **Principal Component Analysis (PCA)**

  Key concept is variance explained by each component (Scree plot)

  Principle components are orthogonal vectors
  creating a subset of parameter that optimizes:
  $\{\frac{1}{n}\sum_{i=1}^{n}(\sum_{j=1}^{p}\phi_{j1}x_{ij})^2\}$
  with
  $\sum_{j=1}^{p}\phi_{j1}^2 = 1$

  - Steps:
    1. Standardize data: $Z = \frac{X-\mu}{\sigma}$
    2. Compute covariance matrix: $\Sigma = \frac{1}{n-1}Z^T Z$
    3. Calculate eigenvectors and eigenvalues: $\Sigma v = \lambda v$
    4. Sort eigenvectors by decreasing eigenvalues
    5. Project data onto principal components: $Y = ZW$
  - Scree plot for determining number of components

  ---

  explains the highest fraction of variance with each consecutive element. Important concept is that each PCA can be understood as a coordinate transformation which is defined in many dimensions, but remember the "curse of dimensions".

  SVD is often used to solve PCA

  Decompose of Matrix into three components $M = U\Sigma V^T$
  where U and V are unitary (or orthogonal if real)matrices and Σ is a rectangular diagonal matrix that contains the singular values = the entries in the scree plot

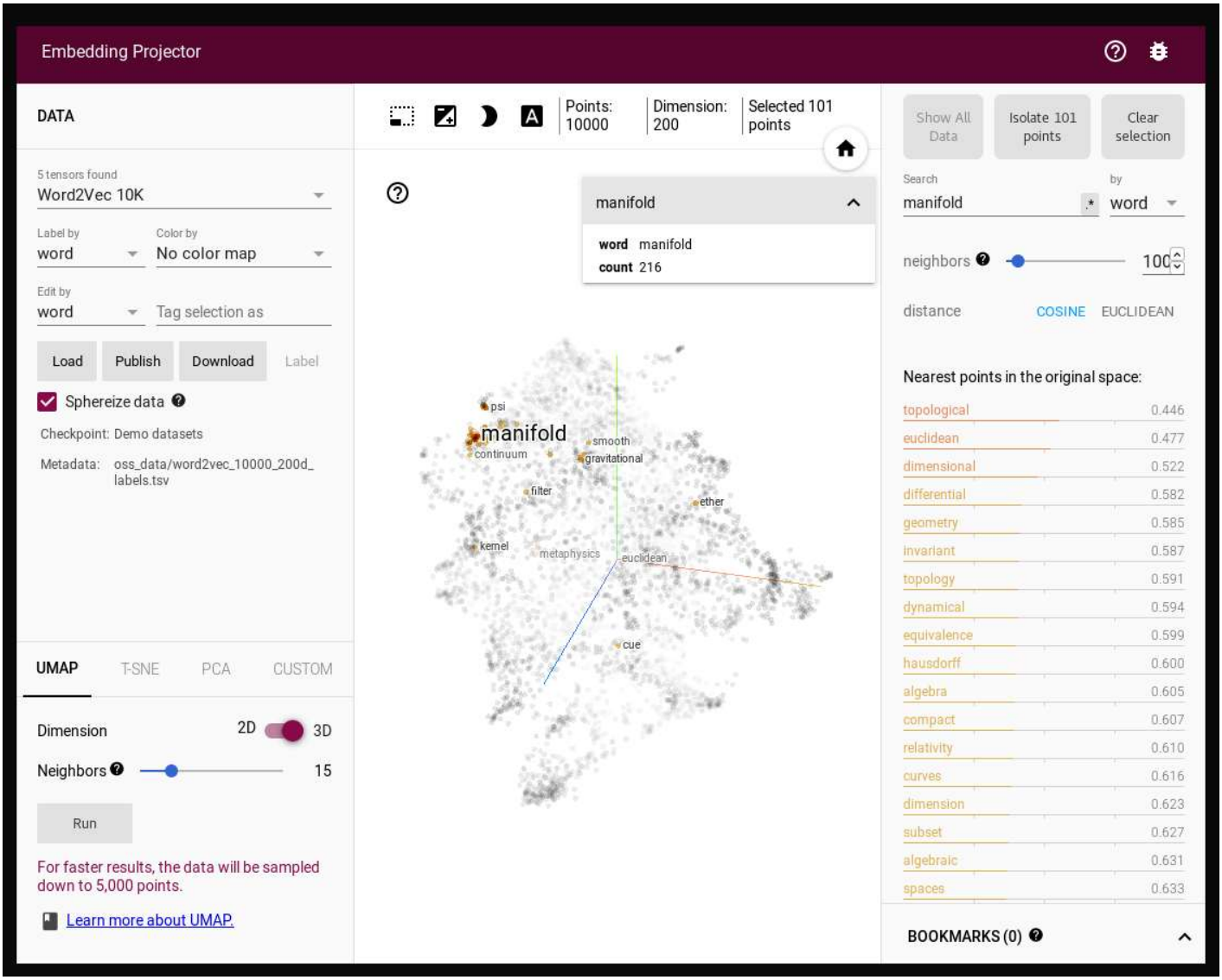
  ---

  In many applications, a low-rank approximation of the original matrix is sufficient. This is achieved through truncated SVD, where only the k largest singular values and their corresponding singular vectors are retained.
  Classical algorithm is: np.linalg.lstsq - Solves the equation with diagonalization (think gaussian elimination) and only if fails with least_squares.

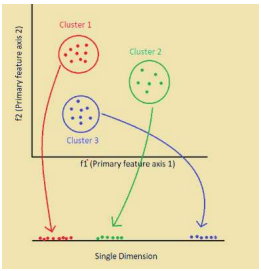
  Challenge it is also a pure mathematical approach (see lab) Advanced approaches would e.g. create the left vectors via a physical model and use calculate the right vectors using a technique commonly called "Global analysis".

Non-linear techniques

Mapping a distribution to a lower dimension



https://projector.tensorflow.org

1.
    1. t-Distributed Stochastic Neighbor Embedding (t-SNE)
        1. Calculates pairwise similarities between data points in high-dimensional space using Gaussian distributions.
        2. Creates a similar probability distribution in low-dimensional space using t-distributions.
        3. Minimizes the difference between these two distributions using gradient descent.
        4. 

1.

2. Uniform Manifold Approximation and Projection (UMAP)
    1. Compute nearest neighbors for each data point
    2. Create a fuzzy simplicial set representation of the data
    3. Find a low-dimensional projection that minimizes the cross-entropy between the high and low-dimensional representations

---

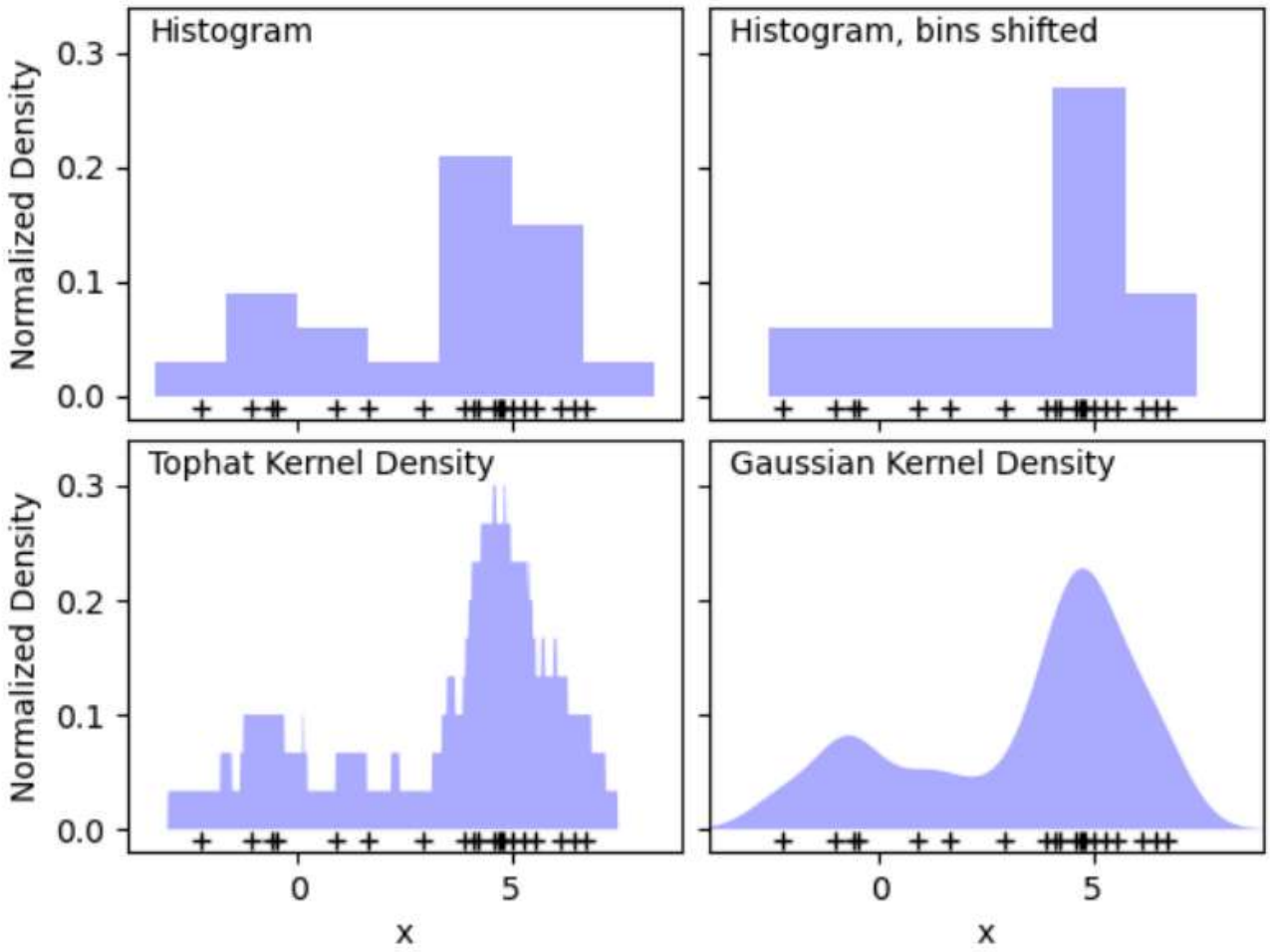
3. Isomap (Self organizing maps)
    - Constructs a neighborhood graph of the data points
    - Computes geodesic distances between points using the graph
    - Applies multidimensional scaling (MDS) to the geodesic distance matrix
4. Locally Linear Embedding (LLE)
    1. Find k-nearest neighbors for each data point in the high-dimensional space.
    2. Compute weights that best linearly reconstruct each point from its neighbors.
    3. Map the data to a lower-dimensional space while preserving these local geometric relationships.Self organizing maps)

- Neural network-based:
    1. (Variational Autoencoders (VAEs)
       Train machine to create indistinguishable datapoints
       https://www.tensorflow.org/tutorials/generative/autoencoder

---

Density Learning

- With given form (parametric): Specifies a family of distributions (thing Gaussian Mixture Models)
- Without given form (non-parametric): kernel density estimation or histograms



Straight from https://scikit-learn.org/stable/modules/density.html

Association Rule Learning

- Definition: Discovering relationships between variables in large databases
- Key concepts: Itemsets: Groups of items that frequently occur together
  1. Support: Frequency of an itemset in the dataset / think letters in language, discover ciphers: $supp(X) = \frac{|t \in T; X \subseteq t|}{|T|}$
  2. Confidence: Probability of finding the consequent given the antecedent *think smartphone word prediction: $conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$
  3. Lift: Strength of the association between items: $lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) \times supp(Y)}$

- Algorithms:
  1. Apriori algorithm> make set and verify
  2. Eclat algorithm> Equivalence Class Clustering and bottom-up Lattice Traversal * count items first then make set
  3. FP-growth algorithm> Uses a frequent pattern tree structure

Learning questions:

1. What differs unsupervised learning from the two other main groups of machine Learning.
2. Why is it a challenge to use for a large number of dimensions.
3. Describe in a few words the main concept of clustering
    1. name at least 3 different clustering methods.
    2. What is the common main challenge in clustering?
    3. How could you overcome this challenge
4. Dimensionality reduction:
    1. Name the key concept
    2. explain the concept of one linear and one non linear
    3. Explain the key concept of PCA
    4. Why can SVD be used to solve PCA?
    5. Why are in most cases the achieve vectors not representative of the reality.
    6. What is the relation between PCA and linear regression?