Artificial Intelligence (CS303)

Lecture 7: Unsupervised Learning

Hints for this lecture

- **Ground-truth not available any more.**

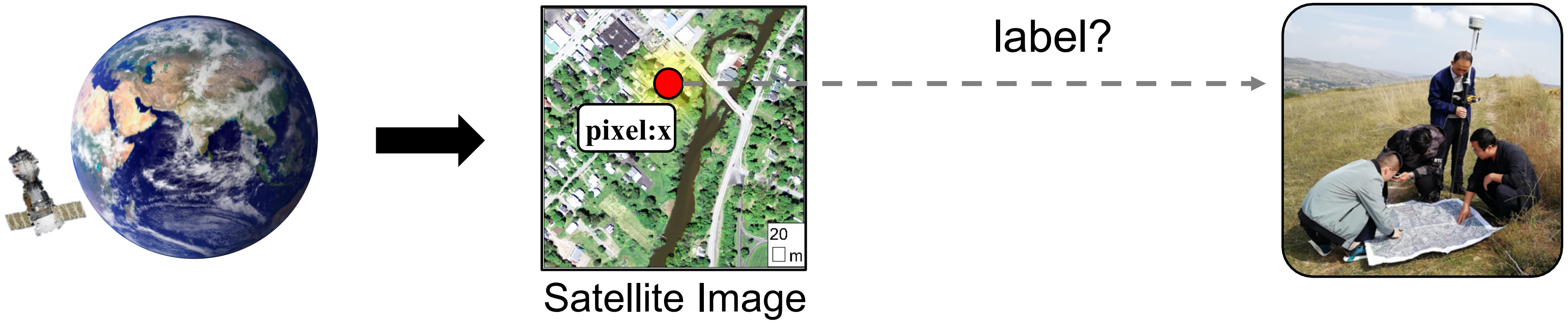
Outline of this lecture

- Why need unsupervised learning?
- Clustering
- Dimensionality Reduction

Why is Unsupervised Learning Important?

- Intuitively, human learning not always rely on a supervision.

- In practice, it might neither be tractable to collect sufficient labelled data



- Instead, it is relatively easy to accumulate large amount of unlabeled data.

Unsupervised vs. Supervised Learning

- Share the same key factors, i.e., representation + algorithm + evaluation
- For supervised learning, since ground-truth is available for the training data, the evaluation (objective function) can be said as **objective**.
- For unsupervised learning, the evaluation is usually less specific and **more subjective**.
- It is more likely that an unsupervised learning problem is ill-defined and the learning output deviate from our intuition.

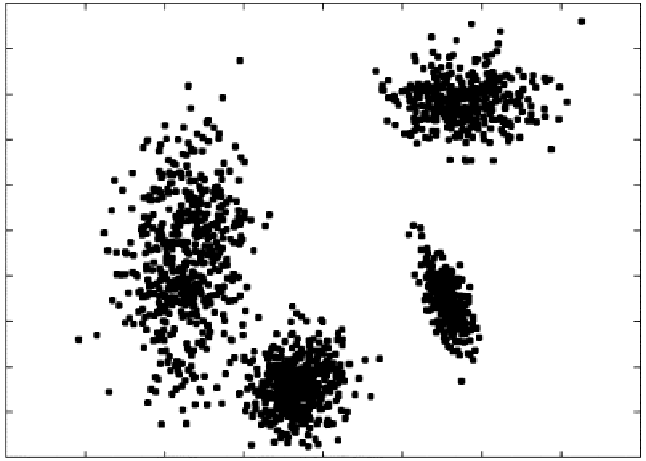
I. Clustering

Problem Description

- A fundamental problem in unsupervised learning.
- Can be viewed as the counterpart of Classification in Supervised Learning.
- Given a set of data, divide them into a number of (say *K*) **clusters** (groups), such that each cluster consists only **similar** instances, while different clusters are not similar to one another.
- Question: when do we need this technique?
  - Example: Define the meaning of a class

Problem Description

- Clustering is a typical ill-defined problem as there is no unique definition of the similarity between clusters.



- Clustering is NP-hard with any objective function (solution space of K^n).

Objective Function - Examples

- Maximize the intra-cluster similarity (i.e., minimize the distance)

$$J = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in D_i} \left|\left|\boldsymbol{x} - \boldsymbol{m_i}\right|\right|^2$$

$$J = \frac{1}{2} \sum_{i=1}^{k} n_i \sum_{\boldsymbol{x}, \boldsymbol{x}' \in D_i} \left|\left|\boldsymbol{x} - \boldsymbol{x}'\right|\right|^2$$

Naïve Approaches

- Naïve approach:
    - Top-down: following the decision tree idea to split the data recursively.
    - Bottom-up: recursively put two instances (or “meta-instances”) into the same group

- Basically you need to define **similarity metric** (e.g., Euclidean distance) first.

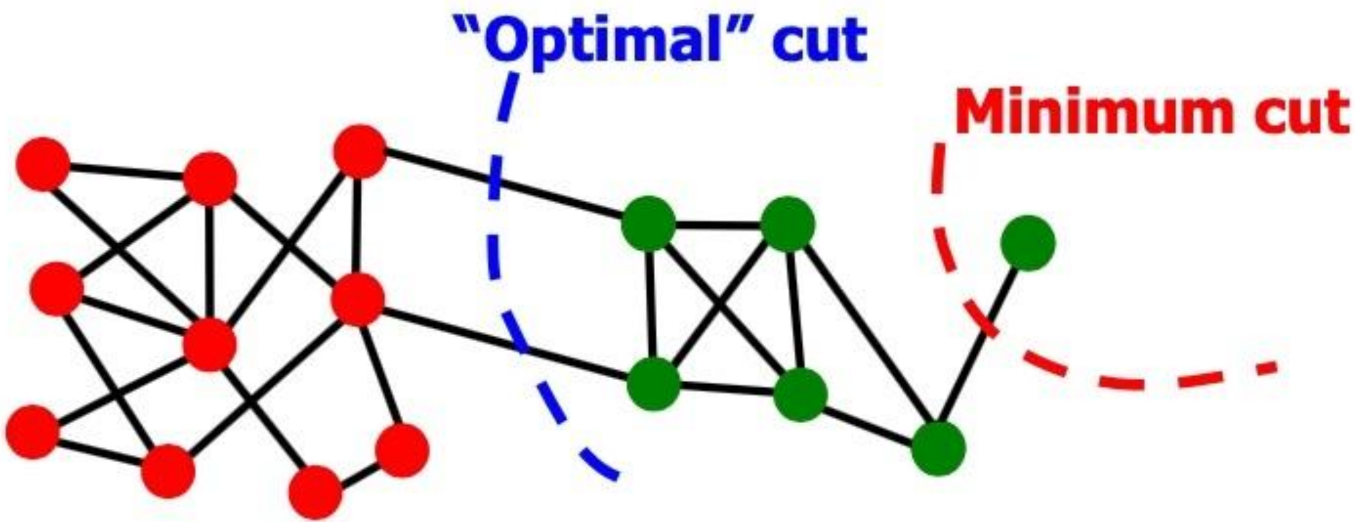
K-Means Algorithm

- Given a predefined *K*
  1. Randomly initialize *K* cluster centers
  2. Assign each instance to the nearest center
  3. Update the *each* center as the mean of all the instances in the cluster
  4. Repeat Step 1-3 until the centers do not change any more

- Not only similarity metric, but also needs calculating of the average.

Clustering for Graph Data

- Community detection on social network
  - The data is not represented as feature vectors
  - Transform the data into a Euclidean space, or
  - Directly treat as a graph-cut problem (many algorithms available)



II. Learning Low-Dimensional Representations

Dimensionality Reduction

- Not restricted to unsupervised learning, but could also use label information.
- However, it is more interesting in UL
  - useful for visualizing the data, and thus aid human to get/understand intuitive knowledge hidden in the data, rather than simply build a good classifier.
- We use two cases to briefly cover this topic.

Case 1: Principal Component Analysis

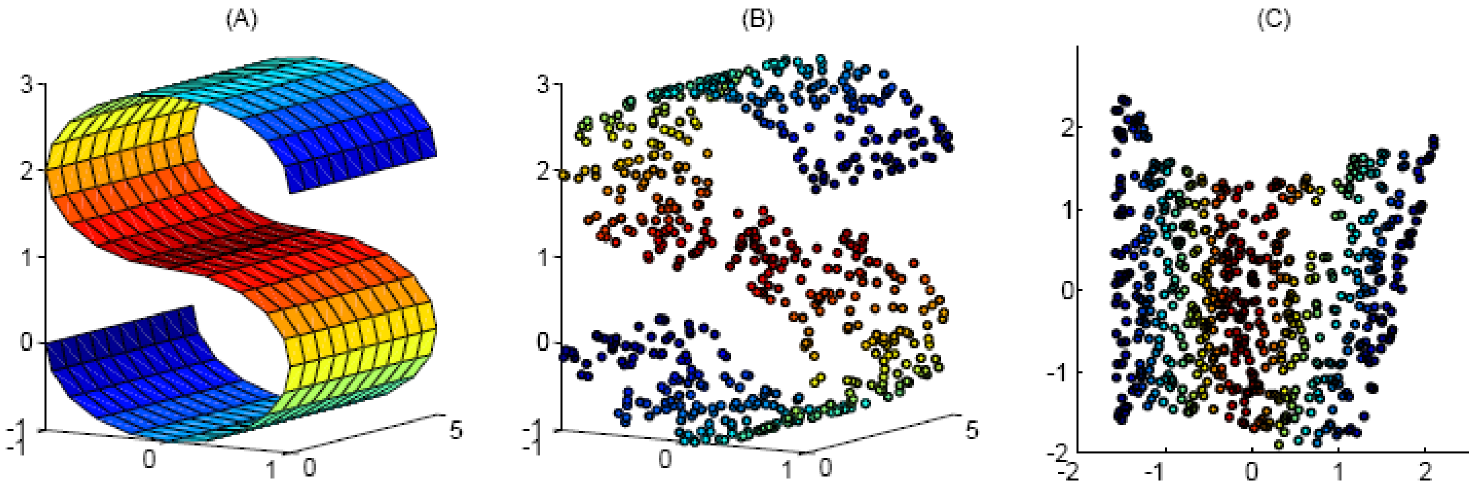
- Given a *n*-by-*d* data set, can we map it into a lower dimensional space with a **linear** transformation, while only introduce the minimum information loss?

- Intuitively,
  - the new space should be composed by a set of linearly independent vectors.
  - Information contained in the data is measured by the variance of the data on each dimension.

- The above two conditions make PCA equivalent to solving a eigenvalue decomposition problem.

Case 2: Locally Linear Embedding

- Can we map the following 3-D data in a lower dimensional space such that the **local relationship between instances** are preserved?



- Intuitively, cannot be done with a linear mapping.

Case 2: Locally Linear Embedding

- **Idea flow:**
  1. Identify nearest neighbors for each instance
  2. Calculate the linear weights for each instances to be reconstructed by its neighbors

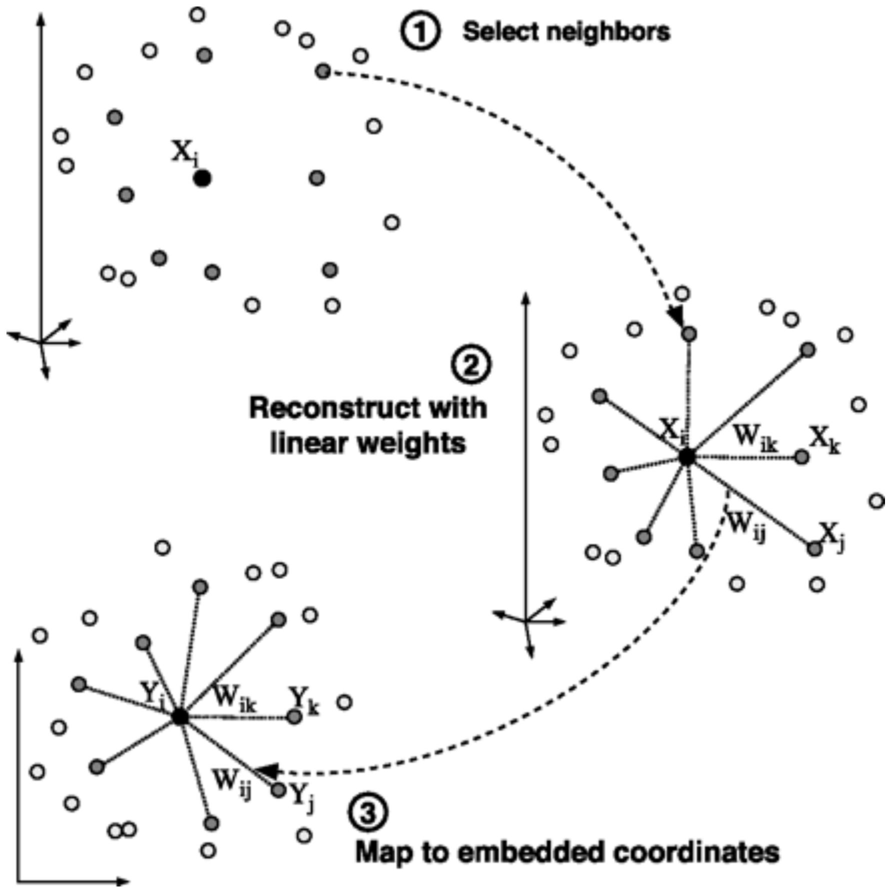
$$\mathcal{E}(W) = \sum_i \left| \vec{X}_i - \sum_j W_{ij} \vec{X}_j \right|^2$$

  3. Use W as the local structure information to be preserved (i.e., fix W), find the optimal values (say Y) for X in the lower dimensional space.

$$\Phi(Y) = \sum_i \left| \vec{Y}_i - \sum_j W_{ij} \vec{Y}_j \right|^2$$

Case 2: Locally Linear Embedding

- **Idea flow:**



Summary

- Unsupervised and supervised learning is not that much different, except that
  - The learning target for UL is usually more subjective or vague.

To be continued