Segmentation algorithms

Ananth Bharathwaj T A (21BCE1079)
Sankari Karthik (21BCE1396)

VIT Chennai

3rd February, 2024

Agenda

Definition

Algorithms

Conclusion

What is segmentation?

- Assignment of labels to pixels
- Input: Image
- Output: Matrix with various elements specifying the object instance to which each pixel belongs
- Based on? color, intensity, contrast, boundary, texture

What role does segmentation play in AI?

- Reduces the complexity of the image
- Enables further processing or analysis of each image segment
- Example: Object Detection
- Use an image segmentation algorithm to find objects of interest
- Detector can operate on a bounding box already defined
- Prevents processing the entire image (improves accuracy, reduces time)

More real life examples

- Medical image analysis
- Computer vision for autonomous vehicles
- Face recognition and detection, video surveillance
- Satellite image analysis
- Compression

General types of algorithms

- Thresholding (Global, Adaptive)
- Edge-based (Canny, Sobel, Laplacian of Gaussian)
- Clustering (K-means, Mean shift, Hierarchical Agglomerative)
- Neural network based

K-means

- A top-down algorithm
- k = number of clusters to split into, given by user
- Steps:
  - Initialize k centroids
  - Assign each data point to the closest centroid
  - Compute the new centroid of each cluster
  - Repeat till convergence
- "closest"?
- "initialize"?

K-means (contd.)

- Choice of K is often ambiguous (Elbow method)
- The Random Initialization Trap
- Pros: scalability, ease of interpretation, guarantees convergence
- Cons: need to pre-determine k, sensitivity to outliers, risk of getting stuck in local minima
- Time Complexity: running a fixed number t of iterations of the algorithm takes O(t*k*n*d), for n (d-dimensional) points, where k is the number of clusters

K-means (contd.)

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2$$

- k = number of clusters
- μ_i = centroid of cluster i
- j = number of cases for cluster i
- Want to minimize this value

Mean Shift Clustering

- A density-based algorithm
- Shifts each data point towards the mode (i.e., the highest density) of the distribution of points within a certain radius
- Does not require specifying the number of clusters or the initial cluster centers
- Steps:
  - Initialize the data points as cluster centroids.
  - Choose a window size (aka 'Bandwidth Parameter').
  - Compute the mean of all points within the window for each data point.
  - Shift the window to mean location and repeat until convergence or maximum iterations have passed
  - Select the final clusters by deleting overlapping windows or merging close windows.

Mean Shift (contd.)

- Pros:
  - Does not need data regarding clusters, it is auto-determined
  - Can handle noise and outliers by simply ignoring the clusters formed by them
  - Does not make assumption on data distribution, allowing it to handle arbitrary shapes
- Cons:
  - Computationally expensive
  - Heavily sensitive to the Bandwidth parameter: Too small a value overfits and too large underfits the clusters.
  - Can have difficulty in finding clusters of different densities, as a single bandwidth may not be suitable for all regions of the data space.

Mean Shift (contd.)

- Time Complexity: $O(n^2)$, n = no of Data points
- Space Complexity: O(n), n = no of Data points

$$f_K(\mathbf{u}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{u} - \mathbf{u}_i}{h}\right)$$

- u = central point of cluster
- n = number of datapoints
- h = Bandwidth Parameter
- d = dimensions of data
- u_i = subpoints in the cluster
- K = Kernel Function: $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$

Plan of action

- Implement both of these algorithms
- Analyze their space and time complexities, their pros and cons, particular use cases
- Come up with a modified algo based on the merits of these two algos

Thank you!

Open to any questions and suggestions