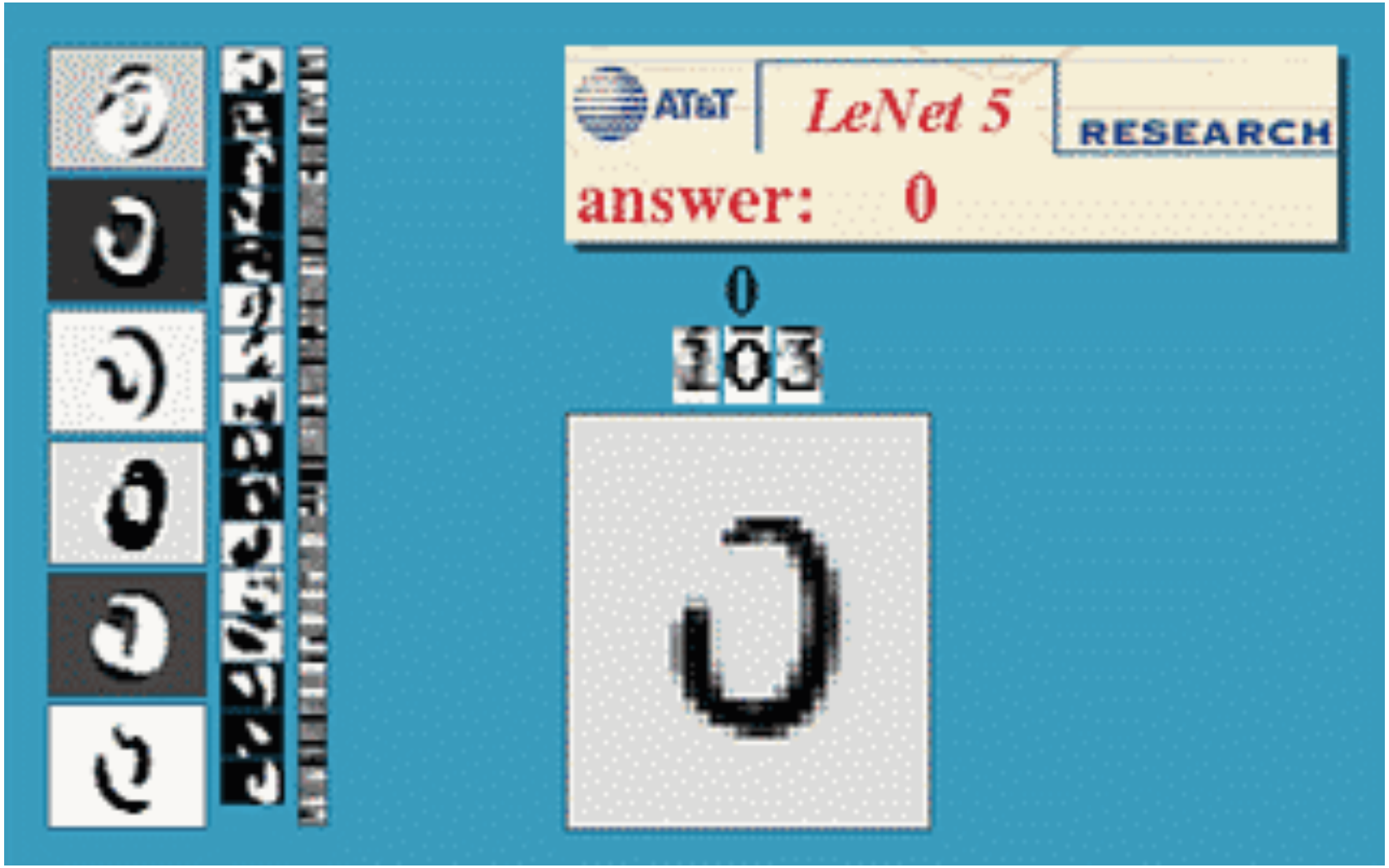
K-Means Clustering

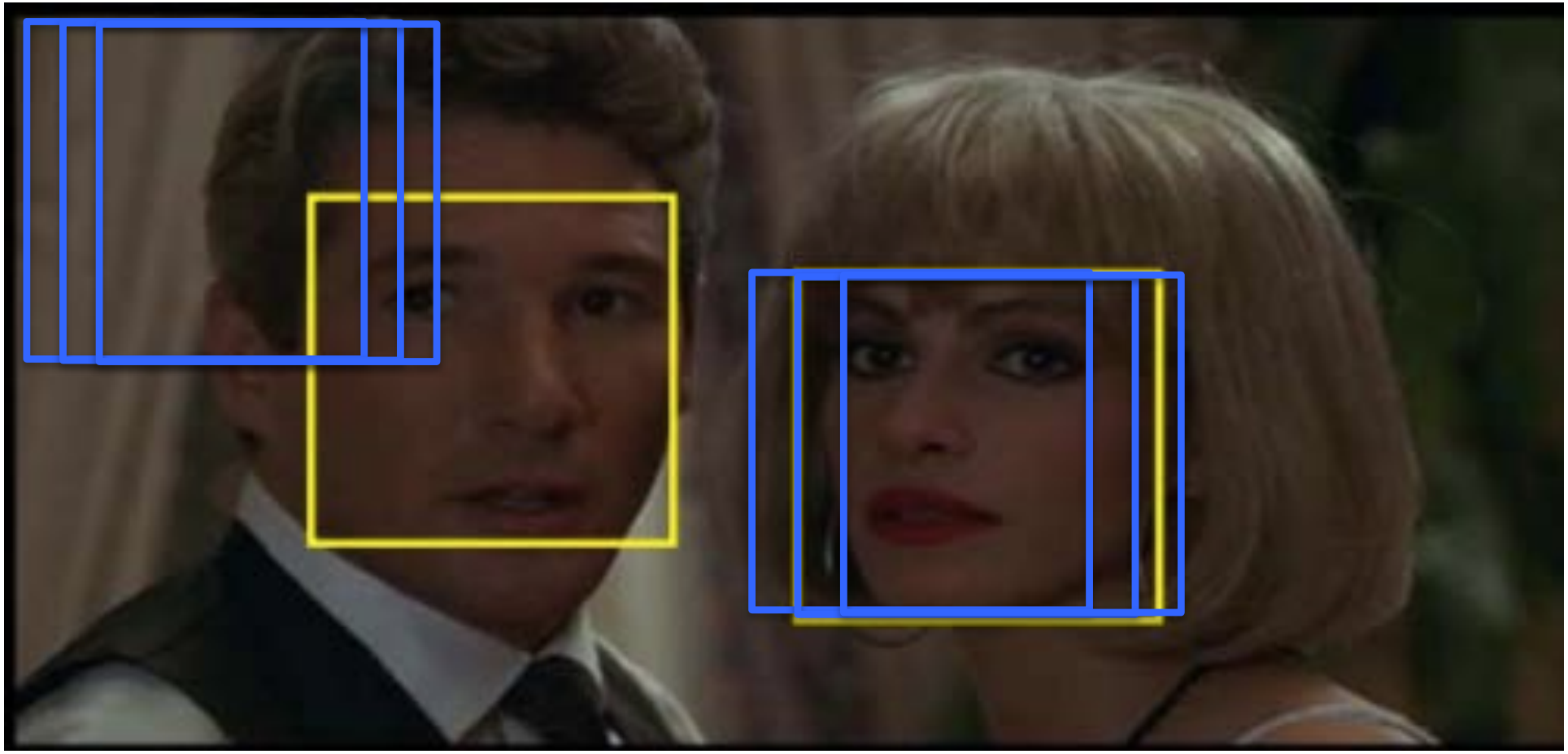
Pascal Fua
IC-CVLab

Reminder: Recognizing Hand-Written Digits



LeNet (1989-1999)

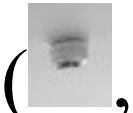
Reminder: Recognizing Faces



$$\mathrm{y} : \mathbf{x} \in \mathbb{R}^{m \times n} \rightarrow \{\text{face}, \text{non-face}\} \quad ?$$

Reminder: Supervised Learning

Train using an annotated training set:

{( , face), ( , face), ( , face), … , ( , not-face), ( , not-face), ( , not-face), …..}

{( , two), ( , three), ( , one), … , ( , four), ( , three), ( , one), …..}

Test on previously unseen images:

 —> Face or not?

 —> What digit?

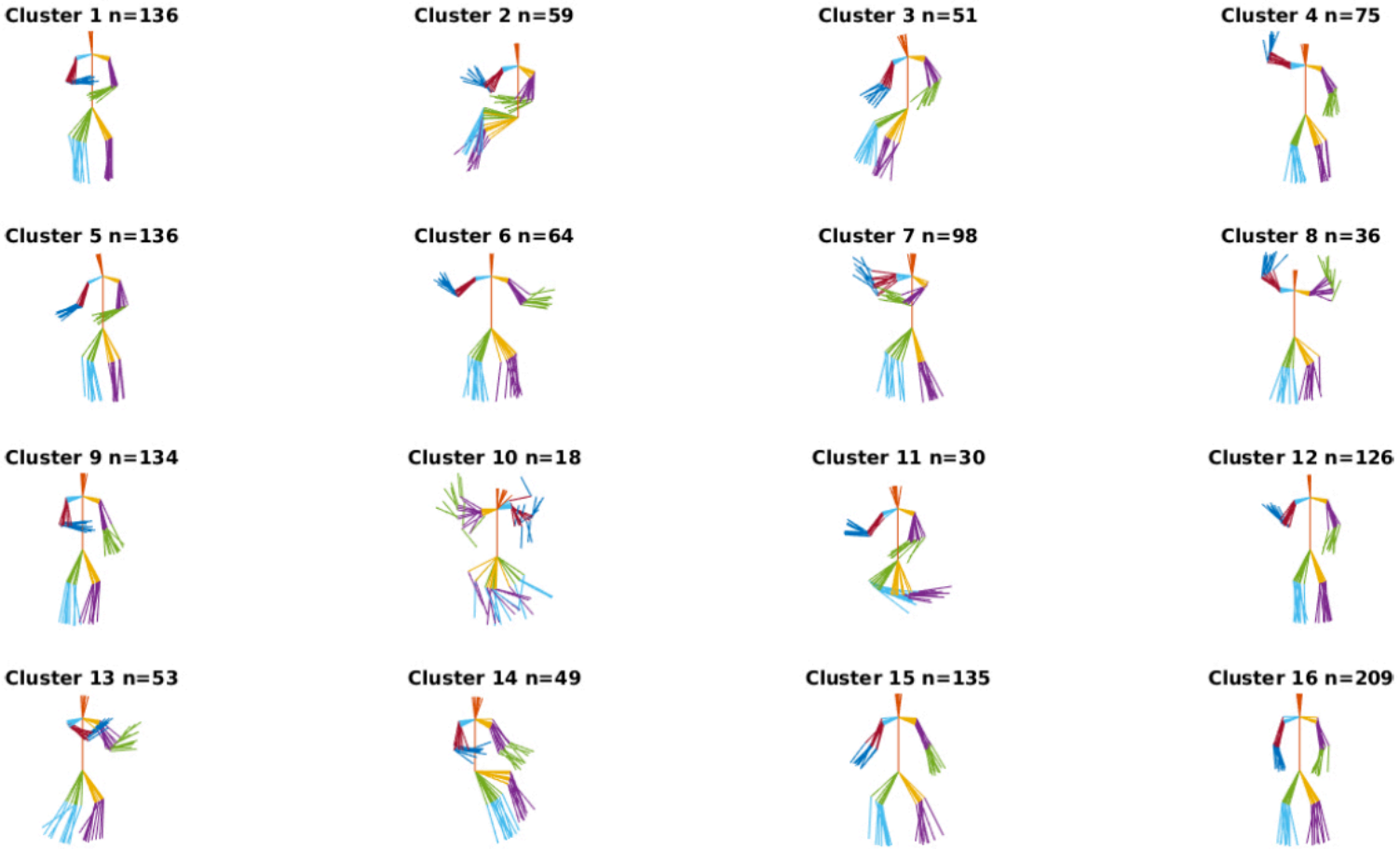
Unsupervised Learning

The training set is not annotated and the system must also learn the classes.

Digital Humanities Application

- Paintings by different artists representing the same scene depict the characters in the same pose.
- Can be used to search large databases of paintings.

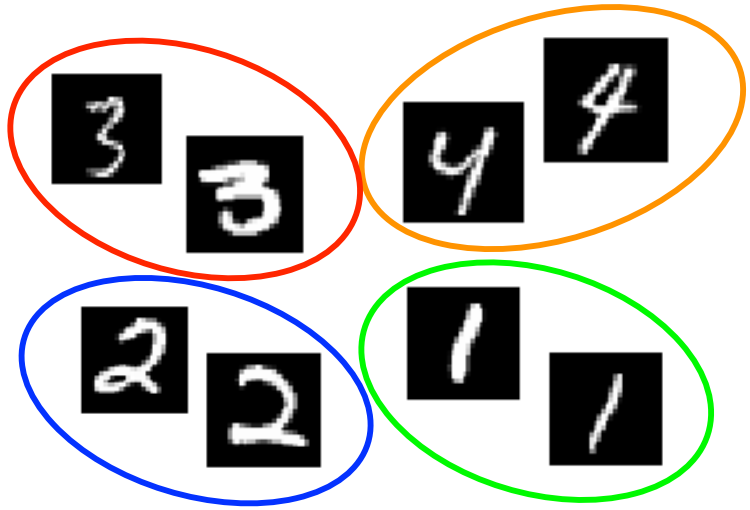
Digital Humanities Application



One can then think of analyzing this phenomenon by clustering the poses observed in a collection of paintings

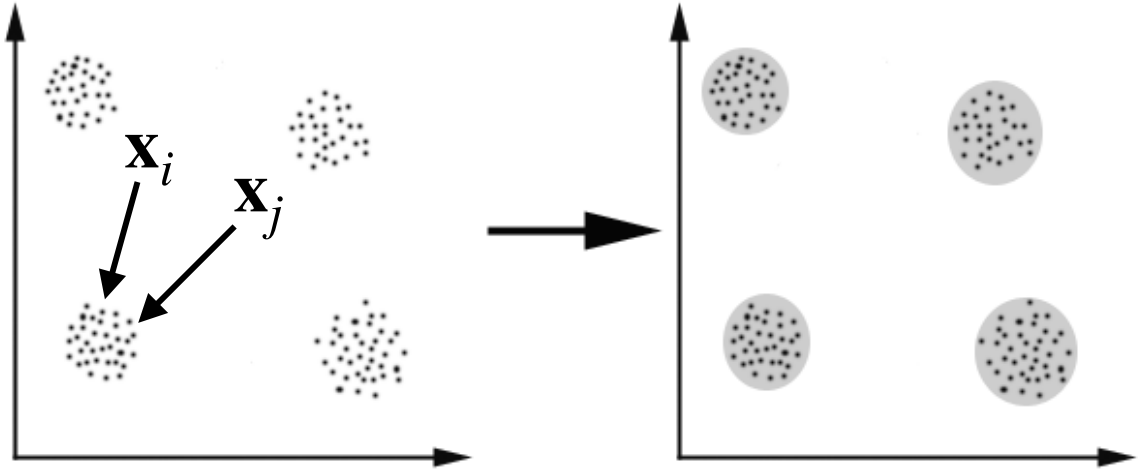
In the Simpler MNIST Context

Given input data **without** labels:



- Can we identify the groups, without performing any data transformation?
- Yes, and this operation is known as **clustering**.

K-Means Clustering

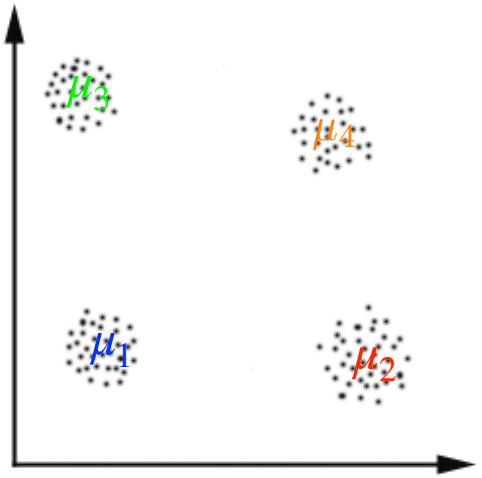


Given a set of input samples:

- Group the samples into K clusters.
- K is assumed to be known/given.
- In the toy example above, each data sample is a point in 2D.
- In our previous examples, each data sample was a human pose or an image.

$\mathbf{x}_i =$  or $\mathbf{x}_i =$ 

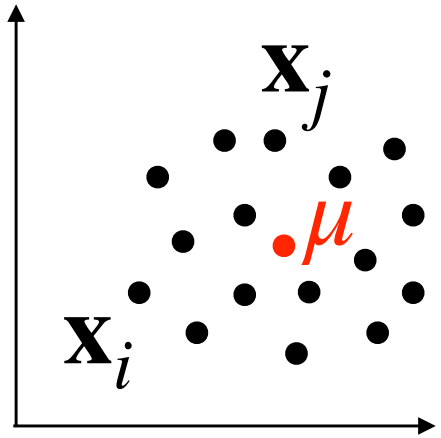
K-Means Clusters



- Cluster k is formed by the points $\{\mathbf{x}_{i_1^k}, \ldots, \mathbf{x}_{i_{n^k}^k}\}$.
- μ_k is the **center of gravity** of cluster k.

The mean of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathbb{R}^D$ is

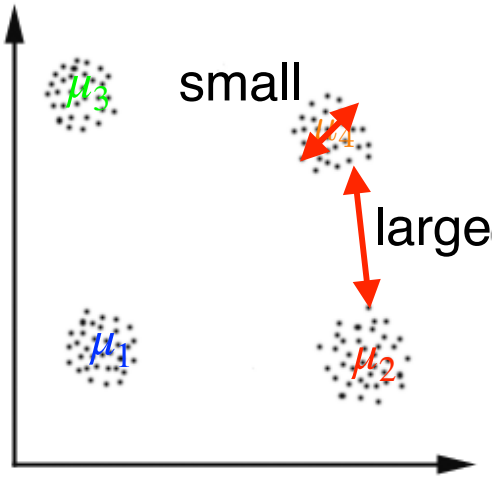
$$\mu = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i \,, \mu \in \mathbb{R}^D$$



In 2D

- If the $\mathbf{x}_i$ were physical points of equal mass, m would be their center of gravity.
- This applies in any dimension.

Formalization



- Cluster k is formed by the points $\{\mathbf{x}_{i_1^k}, \ldots, \mathbf{x}_{i_{n^k}^k}\}$.
- μ_k is the **center of gravity** of cluster k.

- The distances between the points within a cluster should be small.
- The distances across clusters should be large.
- This can be encoded via the distance to cluster centers $\{\mu_1, \ldots, \mu_K\}$:

$$\text{—> Minimize } \sum_{k=1}^{K} \sum_{j=1}^{n_k} (\mathbf{x}_{i_j^k} - \mu_k)^2$$

where $\{\mathbf{x}_{i_1^k}, \ldots, \mathbf{x}_{i_{n^k}^k}\}$ are the n^k samples that belong to cluster k.

Difficult Minimization Problem

Minimize

$$\sum_{k=1}^{K} \sum_{j=1}^{n_k} (\mathbf{x}_{i_j^k} - \mu_k)^2$$

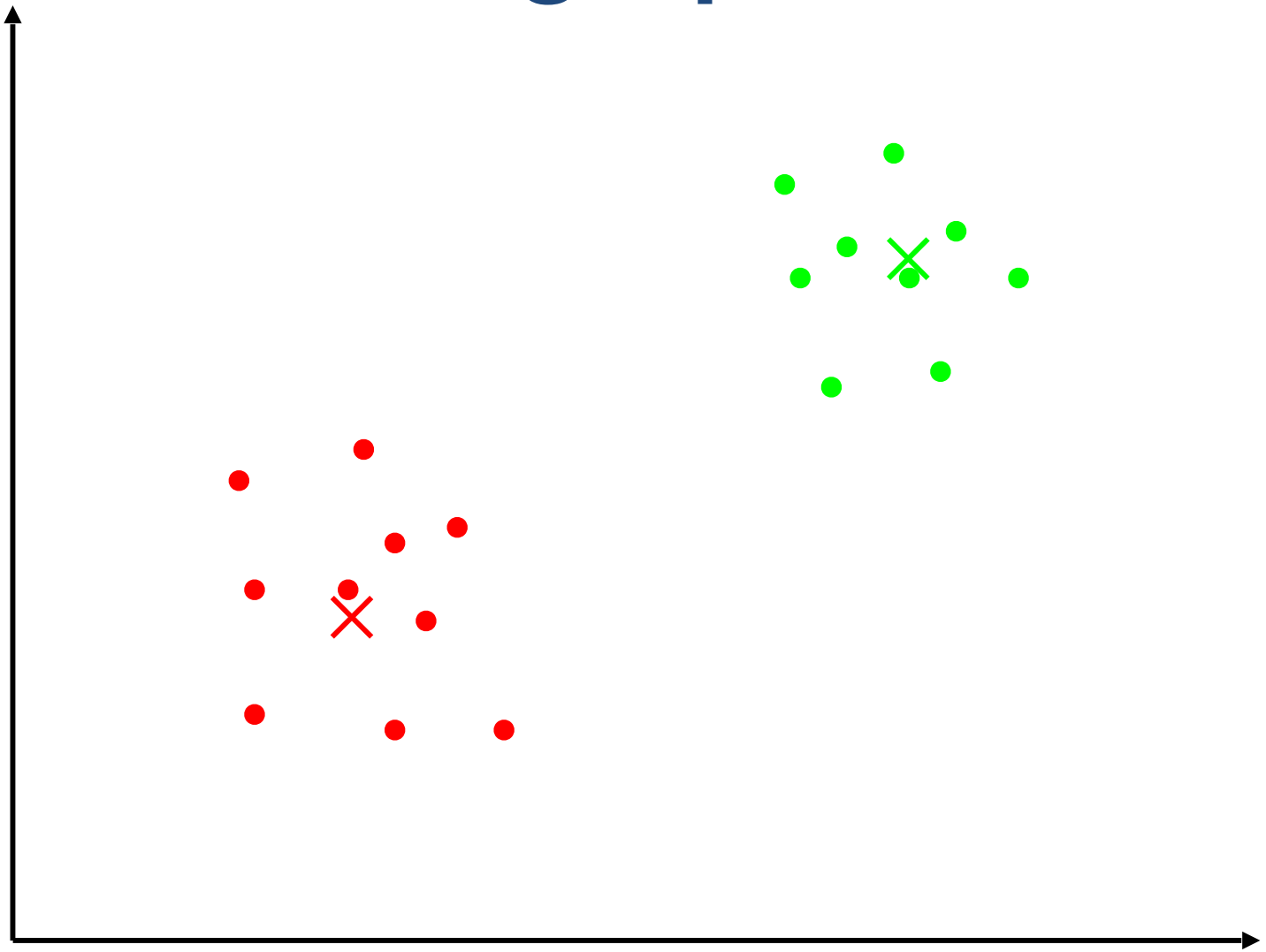
but:

- We don't know what points belong to what cluster.
- We don't know the center of gravity of the clusters.

Simple Solution to the Problem

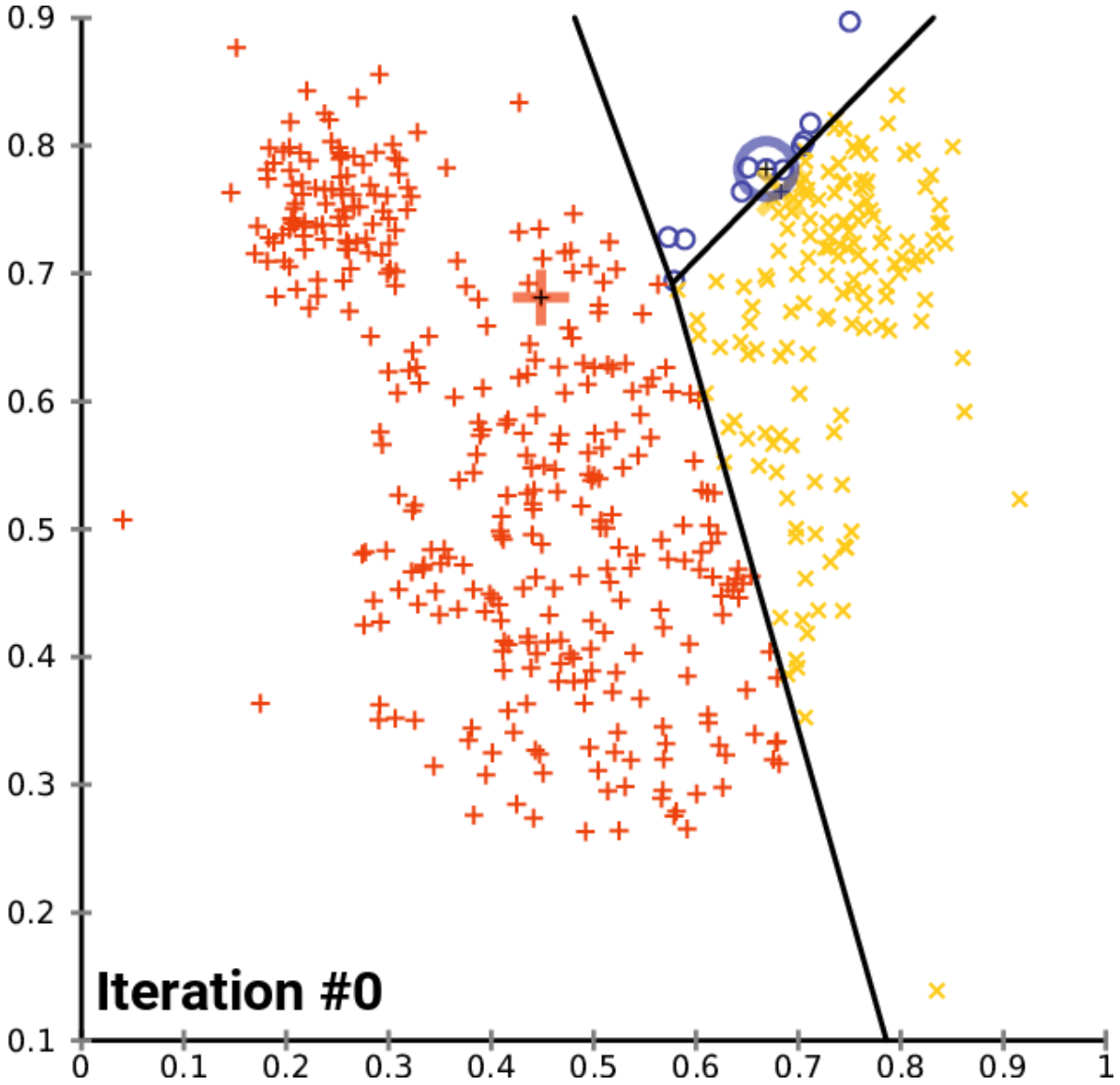
1. Initialize $\{\mu_1, \ldots, \mu_K\}$, randomly if need be.
2. Until convergence
   - 2.1. Assign each point $\mathbf{x}_i$ to the nearest center μ_k
   - 2.2. Update each center μ_k given the points assigned to it

Alternating Optimization



- Initialize
- Associate point to centers
- Recompute centers

Three Classes



More details

- Step 2.1: Assign each point $\mathbf{x}_i$ to the nearest center μ_k.
  - For each point $\mathbf{x}_i$, compute the Euclidean distance to every center $\{\mu_1, \ldots, \mu_K\}$.
  - Find the smallest distance.
  - The point is said to be assigned to the corresponding cluster. Note that each point is assigned to a single cluster.

- Step 2.2: Update each μ_k given the points assigned to it.
  - Recompute each center μ_k as the mean of the points that were assigned to it.

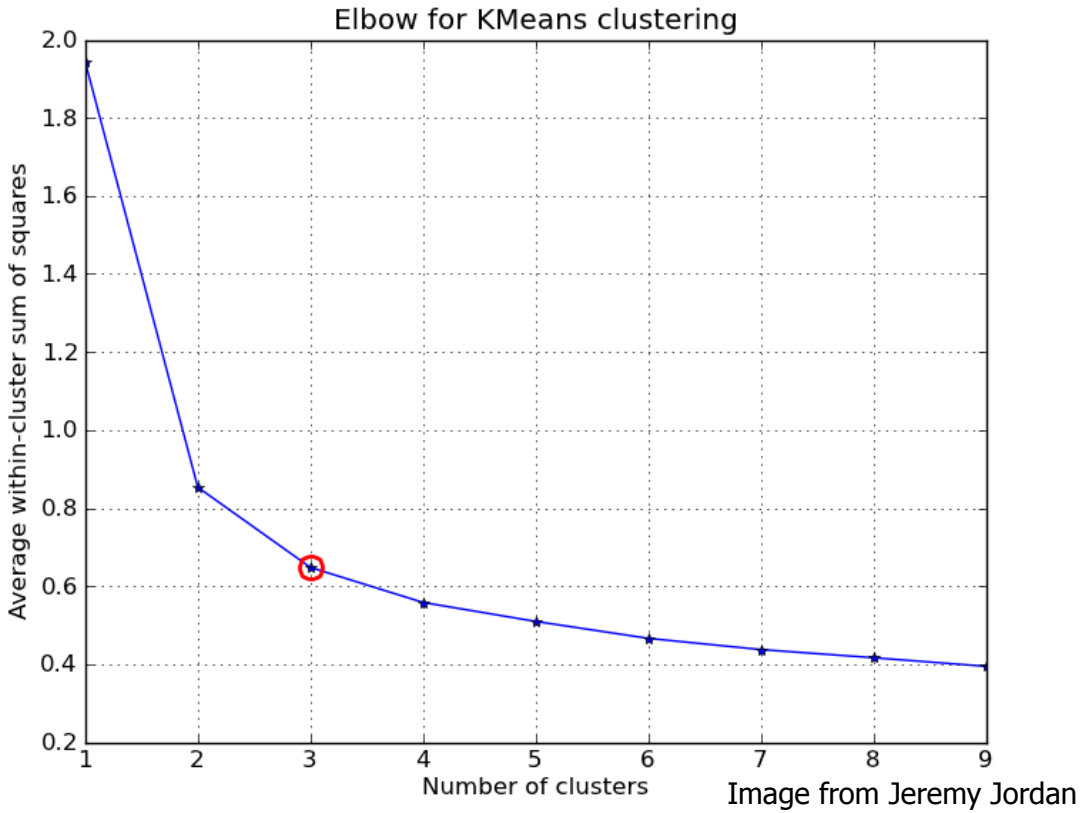
Stopping Criterion

- The algorithm iteratively updates the cluster assignment for each point and the cluster centers. We need to eventually stop iterating.
- This could be achieved by a fixed number of iterations. However, setting this number is arbitrary and a too small number can lead to bad results.
- The algorithm is guaranteed to converge to a stable solution, where the cluster centers and the point assignments are fixed, that is, do not change as more iterations are performed.
- The difference in assignments or center locations between two iterations can be used as criteria to stop the algorithm.

Pseudo Code

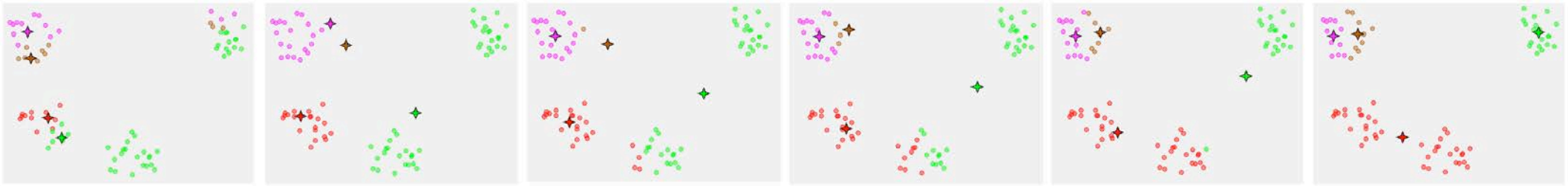
```
def k_means_cluster(k, points):

    # Initialization: choose k centroids.
    centroids = [c1, c2, ..., ck]

    # Loop until convergence
    while True:
        # Initialize clusters list
        clusters = [[] for _ in range(k)]

        # Assign each point to the "closest" centroid
        for point in points:
            distances_to_each_centroid = [distance(point, centroid) for centroid in centroids]
            cluster_assignment = argmin(distances_to_each_centroid)
            clusters[cluster_assignment].append(point)

        # Calculate new centroids
        new_centroids = [calculate_centroid(cluster) for cluster in clusters]

        # Stop or continue
        if (new_centroids == centroids):
            return clusters
        centroids  = new_centroids
```

Heuristic for Choosing K



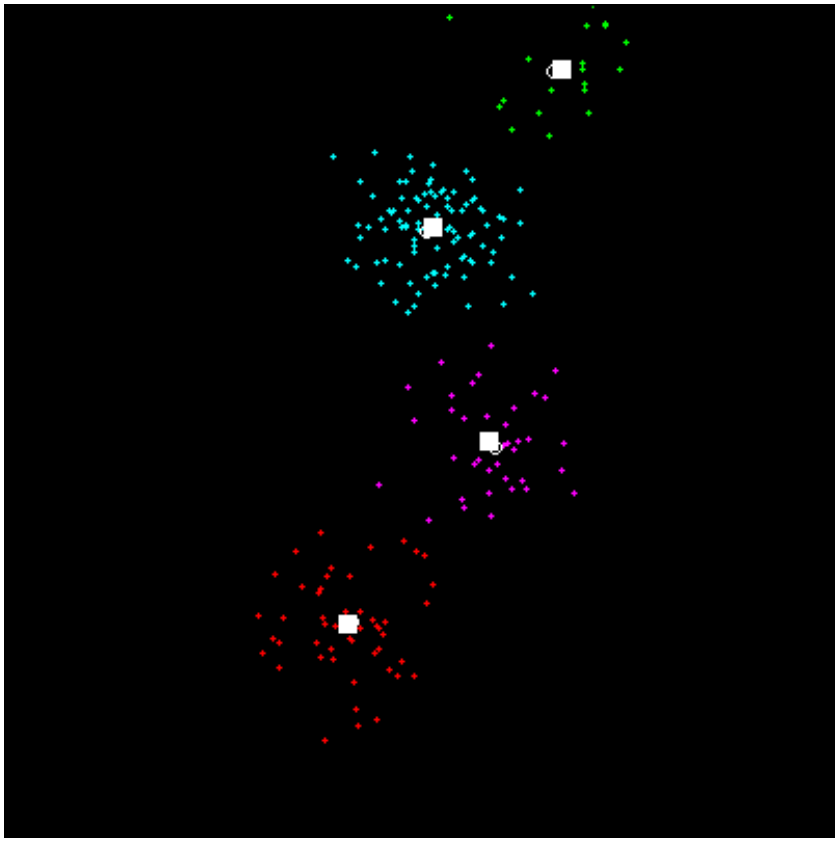
- The average within-cluster distance typically decreases towards zero but using too many clusters make the results meaningless.
- The elbow of the curve is where the drop in within-cluster distances becomes less significant

Potential Issues

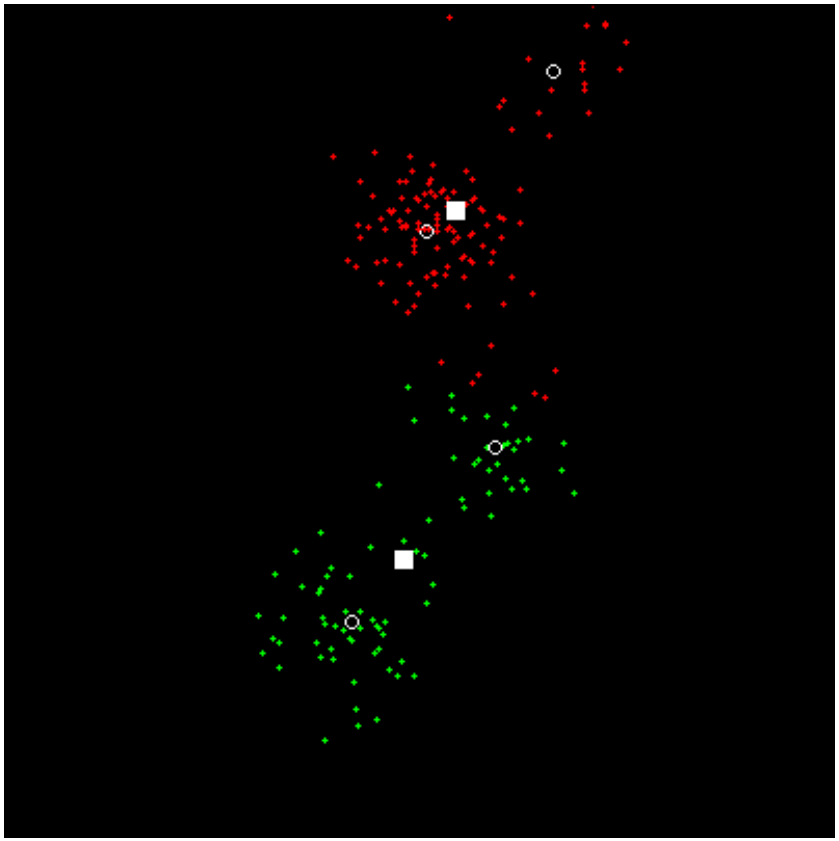


- Even though the algorithm always converges, it does not always converge to the best (desired) solution.
- The solution depends on the initialization.

Initial Conditions Matter



Initially, the points are assigned to the clusters at random—> Success.

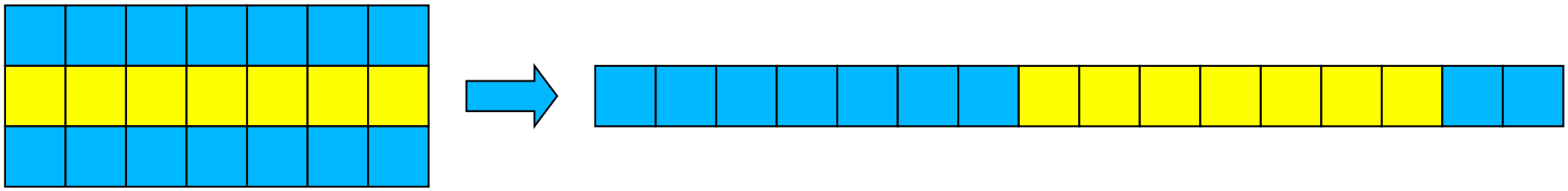


Initially, the points are assigned to the closest cluster —> Failure.

—> In practice try several different random initialization and keep the one that yields the best result in term of the sum of square distances.

Reminder: Black and White Images

136 134 161 159 163 168 171 173 173 171 166 159 157 155
152 145 136 130 151 149 151 154 158 161 163 163 159 151
145 149 149 145 140 133 145 143 145 145 145 146 148 148
148 143 141 145 145 145 141 136 136 135 135 136 135 133
131 131 129 129 133 136 140 142 142 138 130 128 126 120
115 111 108 106 106 110 120 130 137 142 144 141 129 123
117 109 098 094 094 094 100 110 125 136 141 147 147 145
136 124 116 105 096 096 100 107 116 131 141 147 150 152
152 152 137 124 113 108 105 108 117 129 139 150 157 159
159 157 157 159 135 121 120 120 121 127 136 147 158 163
165 165 163 163 163 166 136 131 135 138 140 145 154 163
166 168 170 168 166 168 170 173 145 143 147 148 152 159
168 173 173 175 173 171 170 173 177 178 151 151 153 156
161 170 176 177 177 179 176 174 174 176 177 179 155 157
161 162 168 176 180 180 180 182 180 175 175 178 180 180

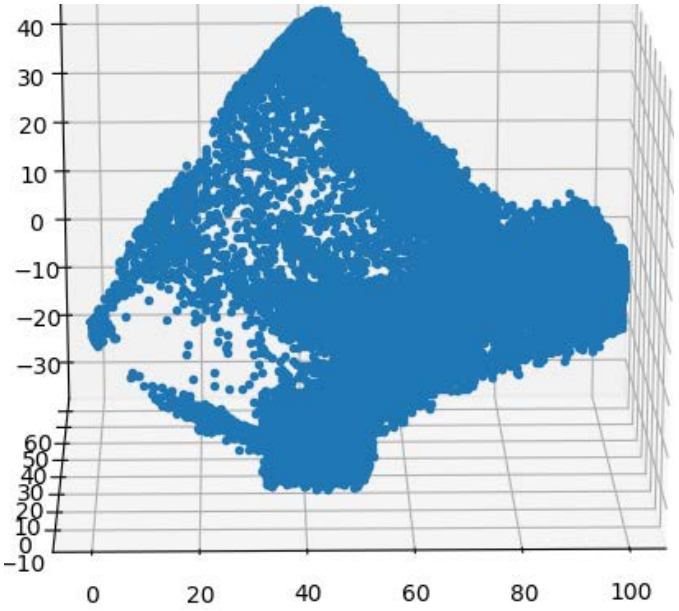


- A MxN image can also be represented as an MN vector.

Color Images

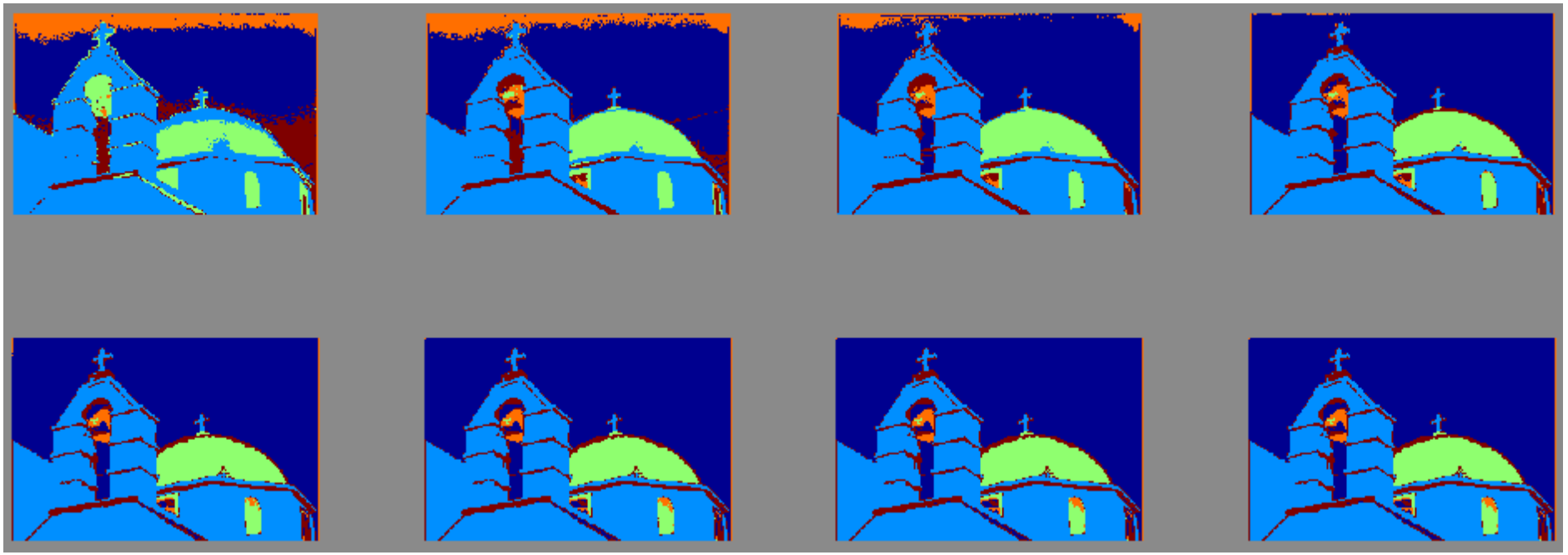
A color image is often represented by three 8-bit images, one for red, one for green, and one for blue.

Color Image as 3D Point Cloud



- Each pixel can be thought of as a 3D point.
- We can run k-means on the cloud of 3D points.

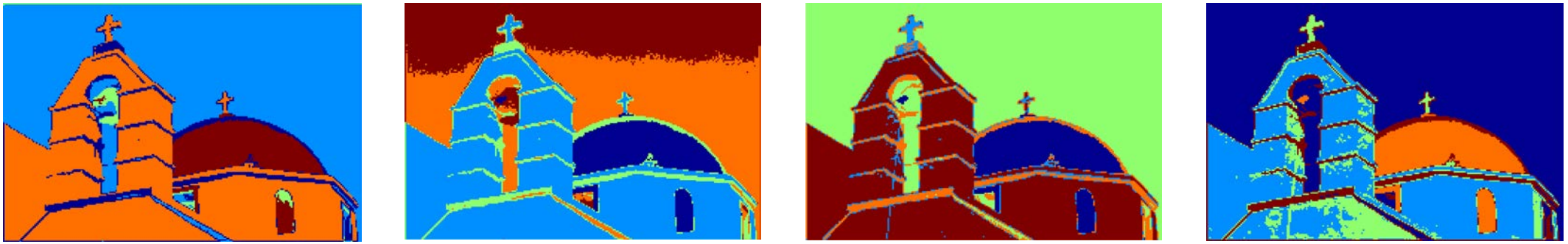
K-Means Clustering for Images



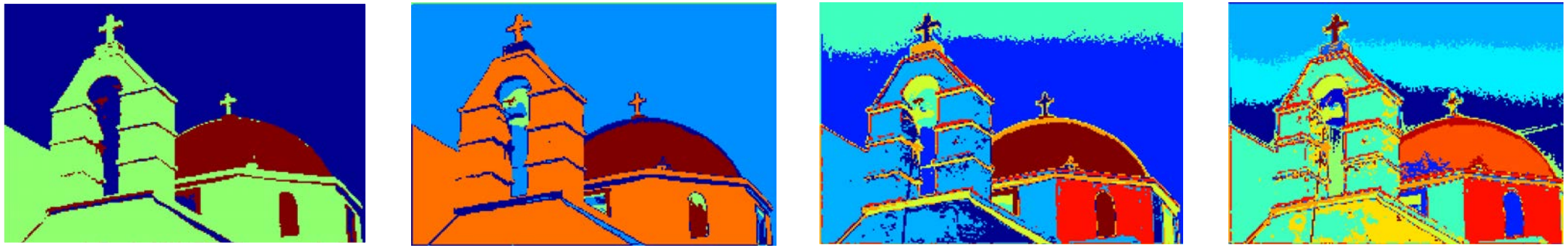
8 iterations for k=5

K-Means Clustering for Images

Different Initializations for k=5



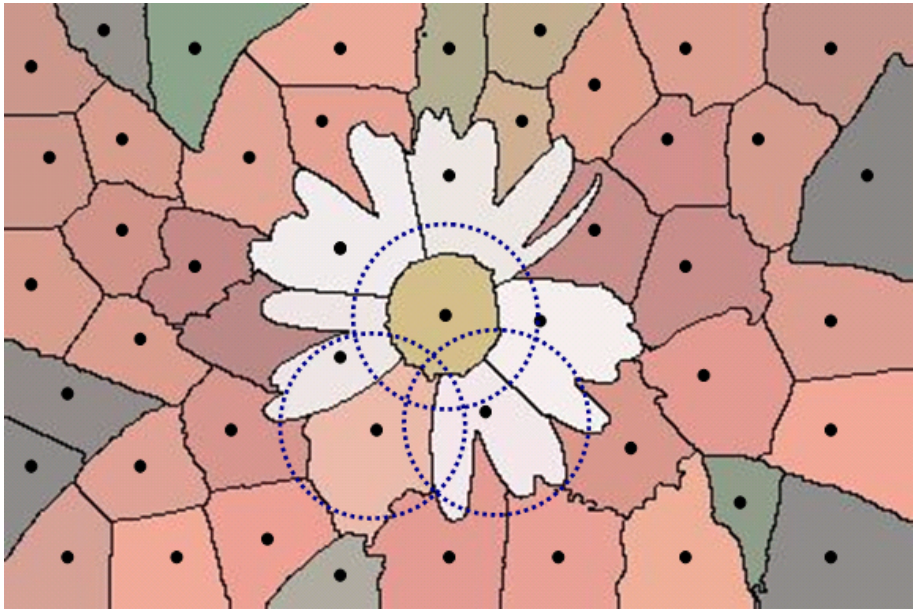
Different values of k



K=3 K=5 K=8 K=15

- Different results for different initializations.
- How do we choose k?

UV + Color (5D)



$$E(\mathcal{C}_1, \ldots, \mathcal{C}_k, \mathbf{c}_1, \ldots, \mathbf{c}_k) = \sum_j \sum_{i \in \mathcal{C}_j} d(\mathbf{x}_i, \mathbf{c}_j)^2$$

$$\mathbf{x} = \begin{bmatrix} u \\ v \\ L \\ a \\ b \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} u \\ v \\ I \\ 0 \\ 0 \end{bmatrix}$$

$$d(\mathbf{x}, \mathbf{c})^2 = \frac{(\mathbf{x}[0] - \mathbf{c}[0])^2 + (\mathbf{x}[1] - \mathbf{c}[1])^2}{h_s^2} + \frac{(\mathbf{x}[2] - \mathbf{c}[2])^2 + (\mathbf{x}[3] - \mathbf{c}[3])^2 + (\mathbf{x}[4] - \mathbf{c}[4])^2}{h_r^2}$$

Run K-Means algorithm with regularly spaced seeds on a grid and using a distance that is a weighted (h_s and h_r) sum of distances in image space and in gray level/color space.

—> Superpixels

Achanta et al. PAMI'12

SLIC Superpixels

1024x1024

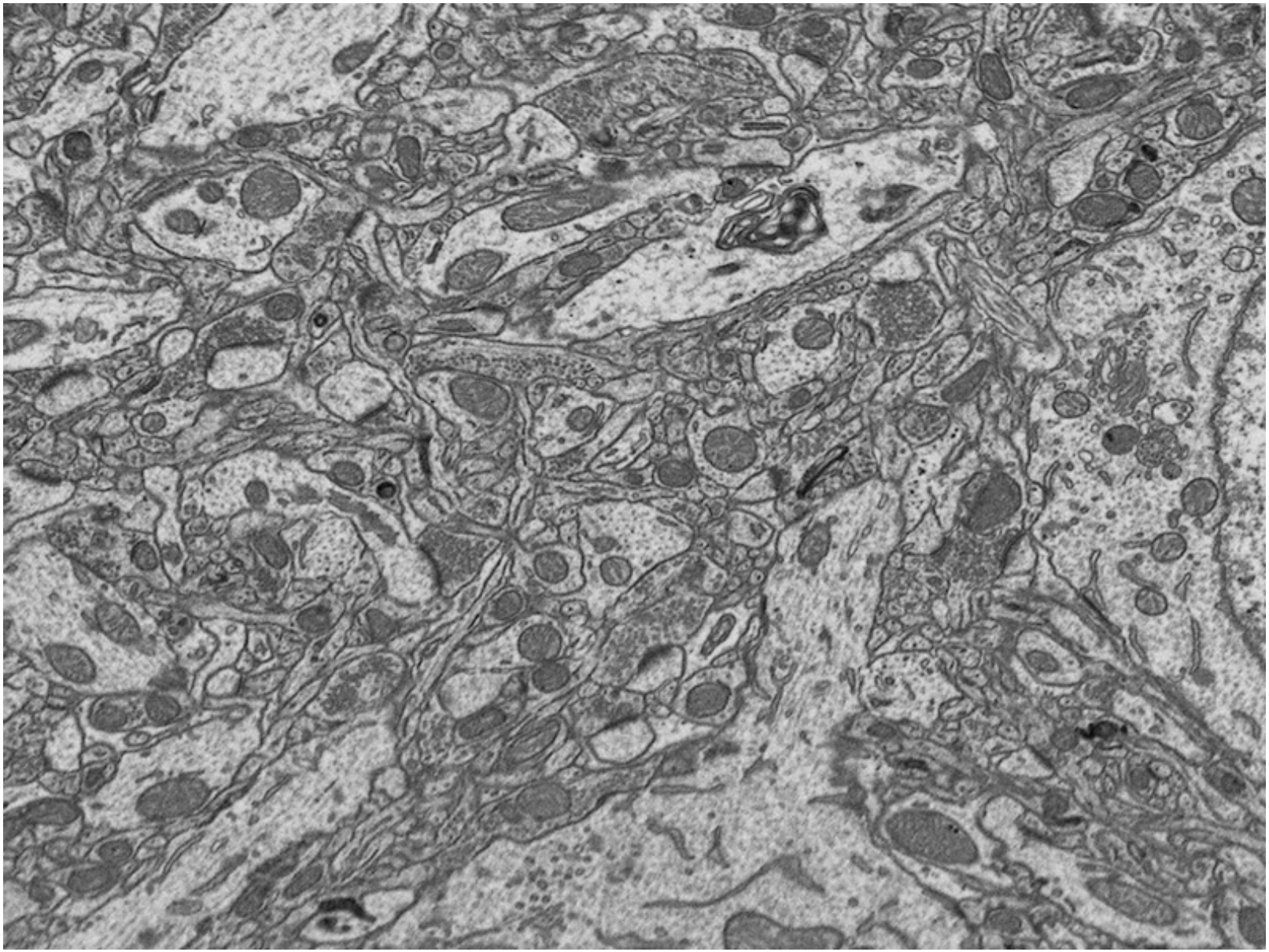
256x256

256x256

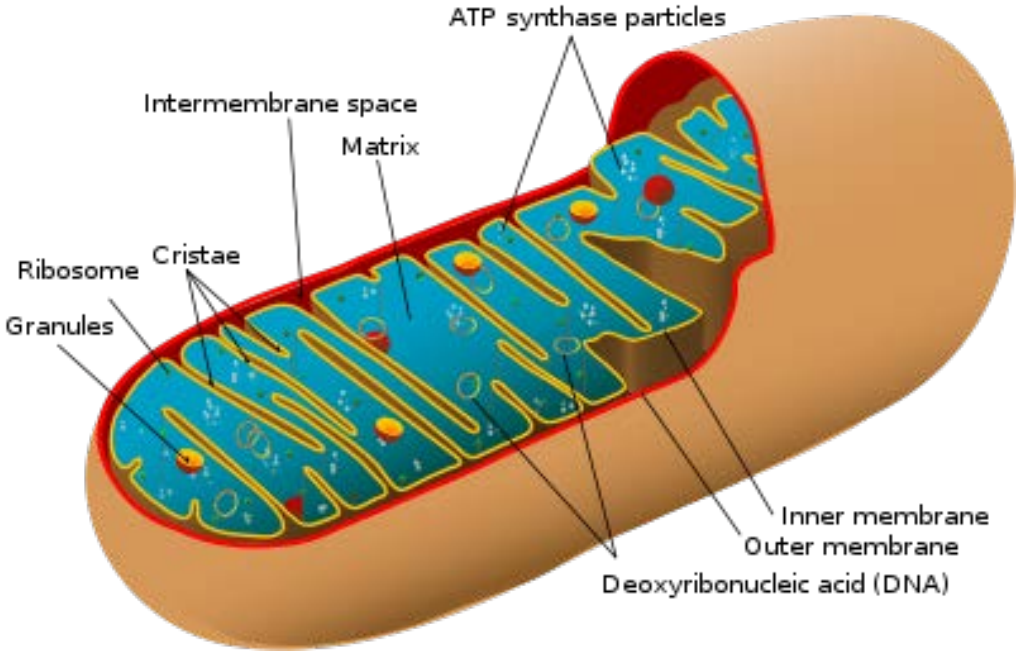
64x64

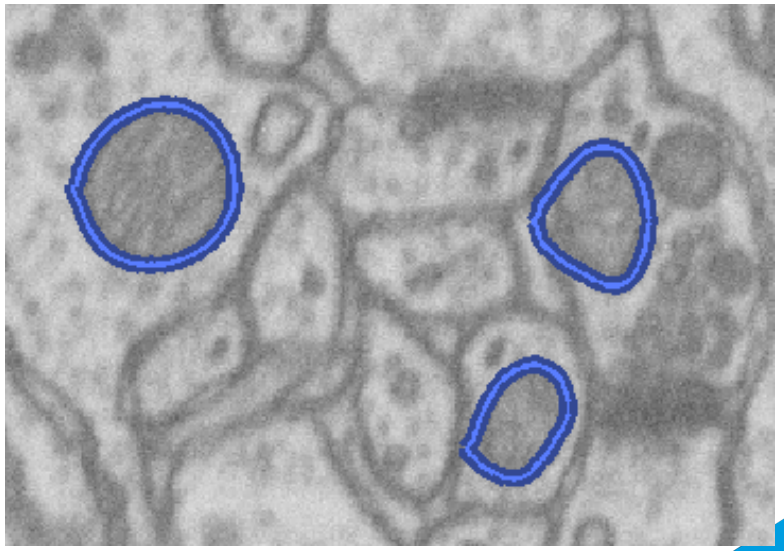
- Superpixel segmentations with centers on a 64x64, 256x256, and 1024x1024 grid.
- Can be used to describe the image in terms of a set of small regions.

Electron Microscopy

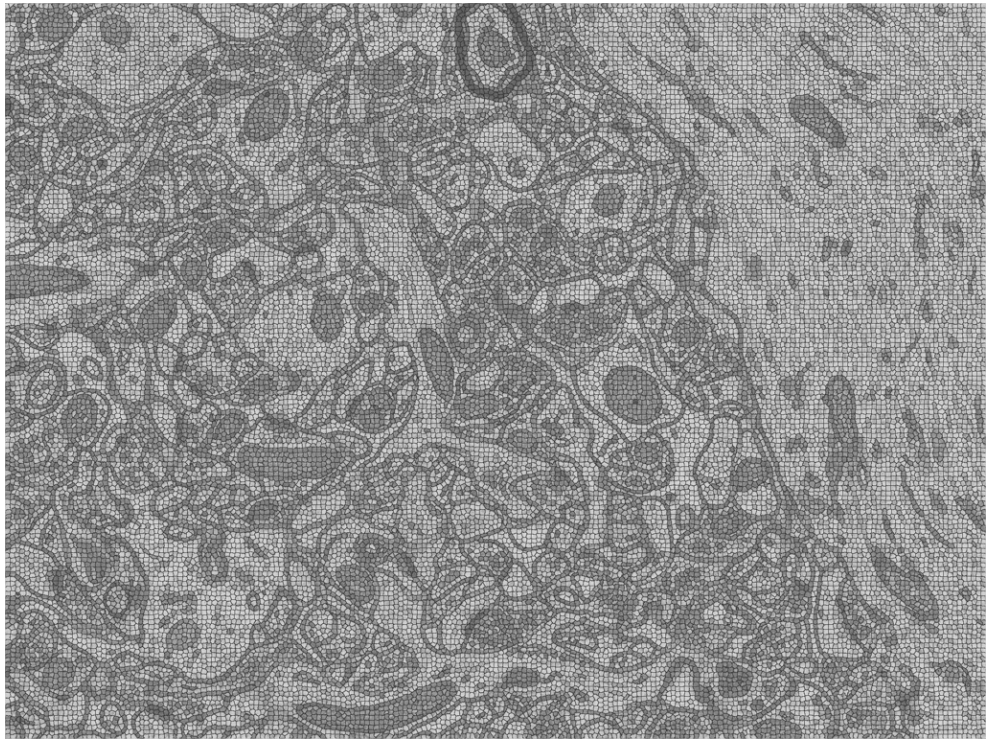


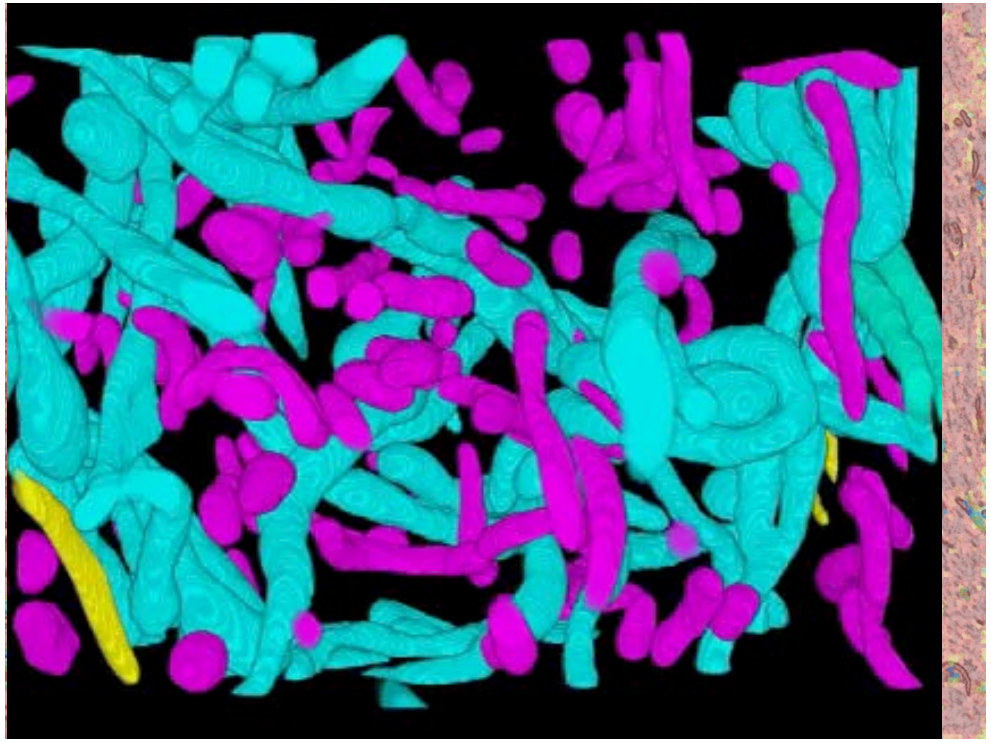
Mitochondria Segmentation





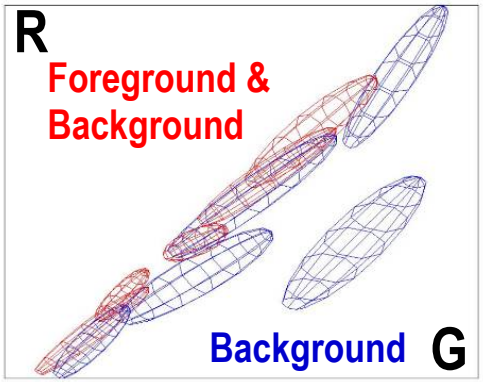
Assigning Probabilities



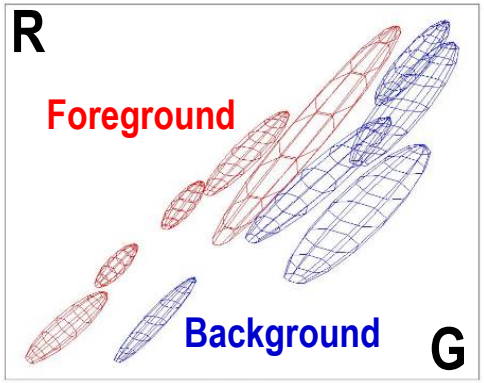


- Compute image statistics for each superpixel.
- Train a classifier to assign a probability to be within a mitochondria.
- Can be used to produce segmentations using graph-based techniques.

Interactive Foreground Extraction



Iterated
graph cut

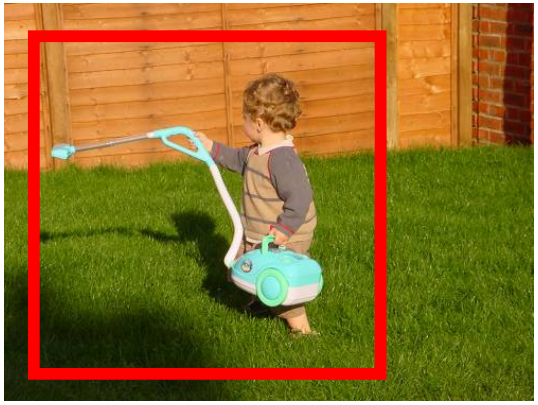


- K-means to learn color distributions
- Graph cuts to infer the segmentation

Optional

GrabCut Rother & al. SIGGRAPH 04

Relatively Easy Examples



Optional

GrabCut Rother & al. SIGGRAPH 04

More Difficult Examples

Fine structure

No telepathy

Initial Rectangle

Camou

Low C

Initial Result

Optional

Inhomogeneous Data

✓ The Euclidean distance is most appropriate for data with homogeneous dimensions:

- In the 2D toy data, both dimensions are of commensurate magnitude.
- In the color image, each dimension represents a color channel and varies in the range $[0,255]$.

x In practice, this is not always the case:

- Different data dimensions may have different magnitudes.
- They can encode different types of information.
- We have already seen this in the case of superpixels.

Example of Data with Heterogeneous Dims

- Wine dataset from the UCI ML repository:
  - 178 wines from 3 different producers.
  - Each wine is represented by 13 attributes, such as quantity of alcohol, malic acid concentration, and magnesium.

- Two samples from the dataset:

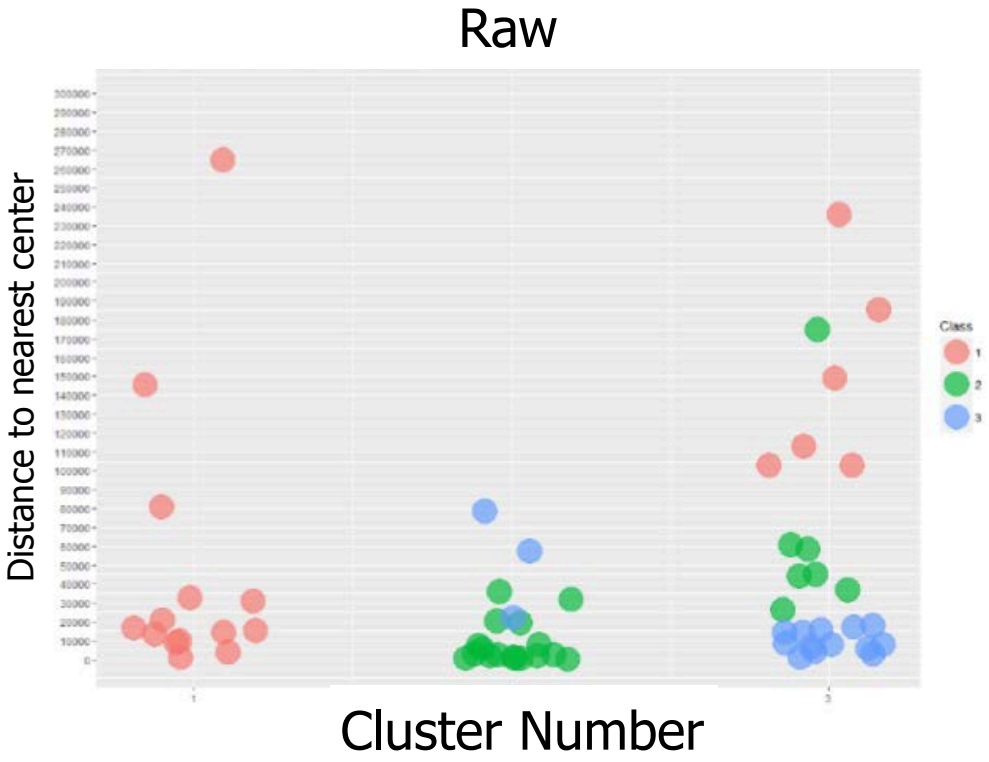
14.37	1.95	2.5	16.8	113	3.85	3.49	0.24	2.18	7.8	0.86	3.45	1480
13.24	2.59	2.87	21	118	2.8	2.69	0.39	1.82	4.32	1.04	2.93	735

- Large values will contribute a lot to the Euclidean distance and small ones much less.
- It does not mean that they are more or less meaningful for clustering!

Solutions

- Scale each dimension by subtracting the smallest value and scaling the result to be between 0 and 1.
- Use a different metric such as the Manhattan distance we saw earlier.
- ……

Wine Example



The color represents the true producer. Ideally all samples in one cluster should have the same color.

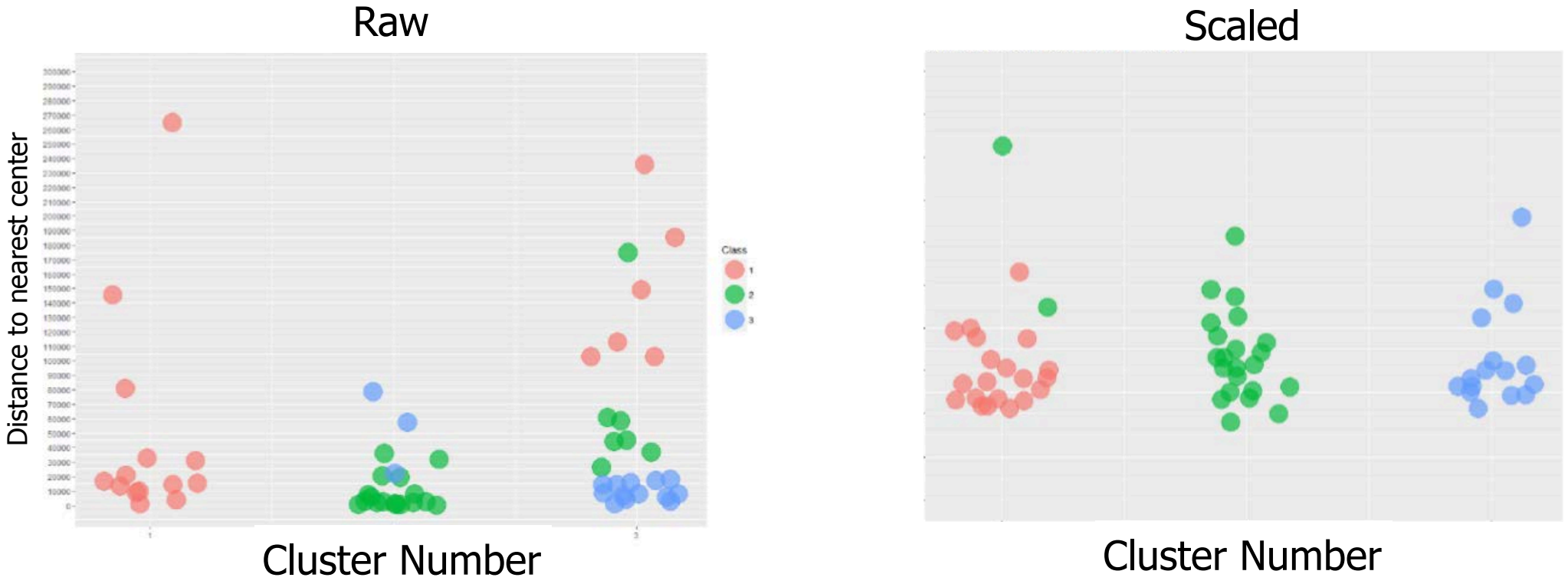
The vertical coordinate is the distance to the cluster center.

Accuracy is defined as the percentage of wines assigned to the correct cluster.

Euclidean distance with raw data: 73% accurate.

https://dinajankovich.com/2018/04/28/k-means-clustering-of-wine-data/

Wine Example



Euclidean distance with scaled data: 93.2% accurate.

Manhattan distance with scaled data: 94.5% accurate.

https://dinajankovich.com/2018/04/28/k-means-clustering-of-wine-data/

Even More Heterogeneous Data

Although I realize that principle is not one of your strongest points, I would still like to know why do do not ask any question of this sort about the Arab countries.

If you want to continue this think tank charade of yours, your fixation on Israel must stop. You might have to start asking the same sort of questions of Arab countries as well. You realize it would not work, as the Arab countries' treatment of Jews over the last several decades is so bad that your fixation on Israel would begin to look like the biased attack that it is.

Everyone in this group recognizes that your stupid 'Center for Policy Research' is nothing more than a fancy name for some bigot who hates Israel.

Whoops!! Wrong group. Soooooooooooooooorry folks..

I'd like to see this info as well. As for wavelength, I think you're primarily going to find two - 880 nM +/- a bit, and/or 950 nM +/- a bit. Usually it is about 10 nM either way. The two most common I have seen were 880 and 950 but I have also heard of 890 and 940. I'm not sure that the 10 nM one way or another will make a great deal of difference.

Another suggestion - find a brand of TV that uses an IR remote, and go look at the SAMS photofact for it. You can often find some very detailed schematics and parts list for not only the receiver but the transmitter as well, including carrier freq. specs. and tone decoding specs. if the system uses that.

- How can we compute a distance between these three pieces of text?
- One solution is to turn them into vectors.

—> We will revisit this issue when we talk about deep nets.

K-Means in Short

- A simple yet effective clustering algorithm.
- Sensitive to initialization.
- Works best on homogeneous data.