# K-Means

Machine Learning 1 — Lecture 12

$13^{th}$ June 2023

Robert Peharz
Institute of Theoretical Computer Science
Graz University of Technology

## Supervised Learning

- **Linear regression**, **logistic regression**
- **Neural networks**
- **K-nearest neighbours**
- **Decision trees**, **random forests**
- **Support vector machines** (SVMs)
- **Linear discriminant analysis**

## Unsupervised Learning

- **Principal component analysis**
- **K-means** (**clustering**, today)
- **Gaussian mixture models** (**density estimation**, **soft clustering**, next time)

**Clustering**

Organize $N$ objects in $K$ clusters, i.e., groups of similar objects.

Relation to classification: "classes" are not given, but should be found in unsupervised way.

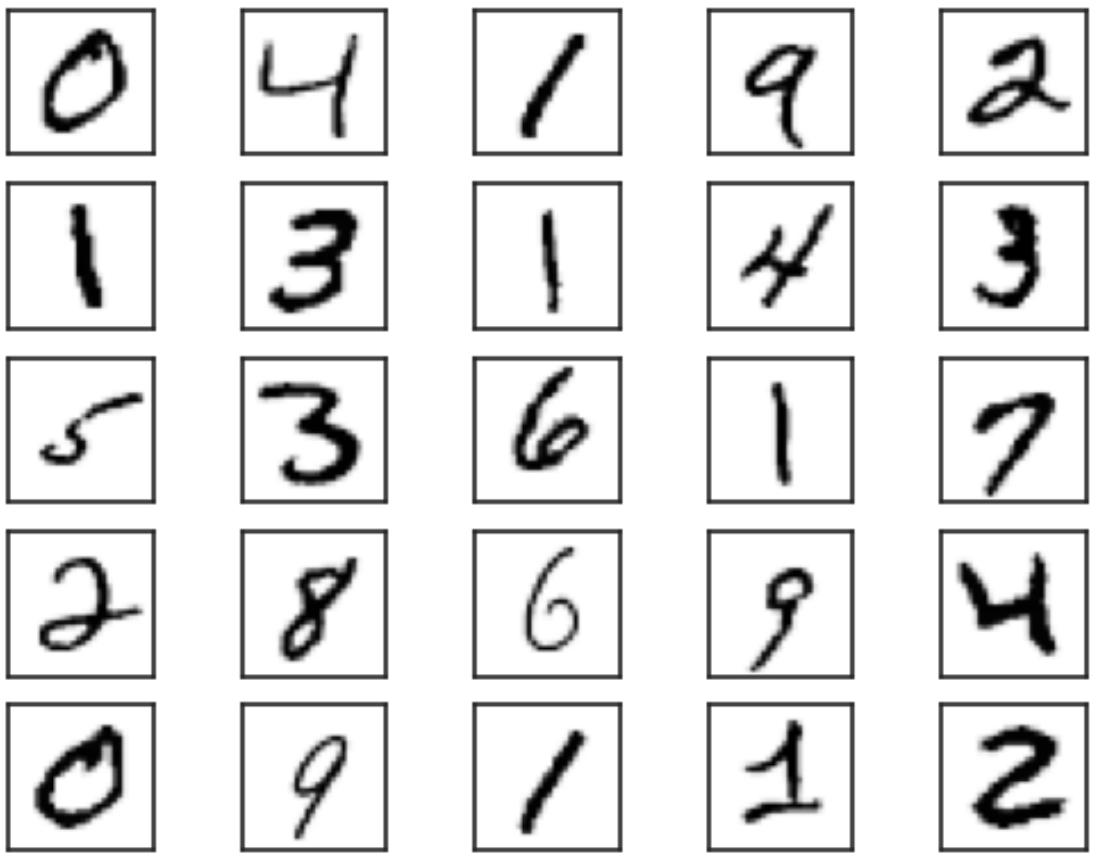
Clustering is not well-defined and there are multiple questions:

- We want to group similar objects together, but what does "similar" mean?
- What is $K$, i.e., the number of clusters? Provided by the user or found automatically?
- Can an object belong to more than one cluster? (**multi-view clustering**)
- Does an object need to belong to a cluster, i.e., do we allow **outliers**?
- Is membership of objects to clusters "hard" (0 or 1) or "soft" (probabilistic, fuzzy)?

How do we cluster a deck of cards?


- suits? (hearts, spades, clubs, diamonds)
- value? (2, 3, . . . , King, Ace)
- court cards vs. numerical?

How to cluster handwritten digits (MNIST data)?



- Objects (data points) are represented with $D$-dimensional vectors $\mathcal{D} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$
- **Cluster centers** are represented as $D$-dimensional vectors $\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}$
- Each data point is assigned to cluster whose center is closest
- Hence, similarity is defined via **Euclidean distance**
- Data points which are closest to $\boldsymbol{\mu}^{(k)}$ form the $k^{\text{th}}$ **cluster** $\mathcal{D}_k = \{\boldsymbol{x} \mid \boldsymbol{x} \in \mathcal{D} \ \wedge \ k = \arg\min_l \|\boldsymbol{x} - \boldsymbol{\mu}^{(l)}\|_2\}$
- $\boldsymbol{\mu}^{(k)}$ is a **representative** or **prototype** of cluster $\mathcal{D}_k$
- $\mathcal{D}_k$ is the set of all data points which fall in the **Voronoi cell** specified by $\boldsymbol{\mu}^{(k)}$

# Voronoi Cells

Example

**Voronoi cells**: Given a set of vectors $\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}$, the $k^{\text{th}}$ Voronoi cell is the set of all points which are closest $\boldsymbol{\mu}^{(k)}$. Voronoi cells form a partition of the Euclidean space.

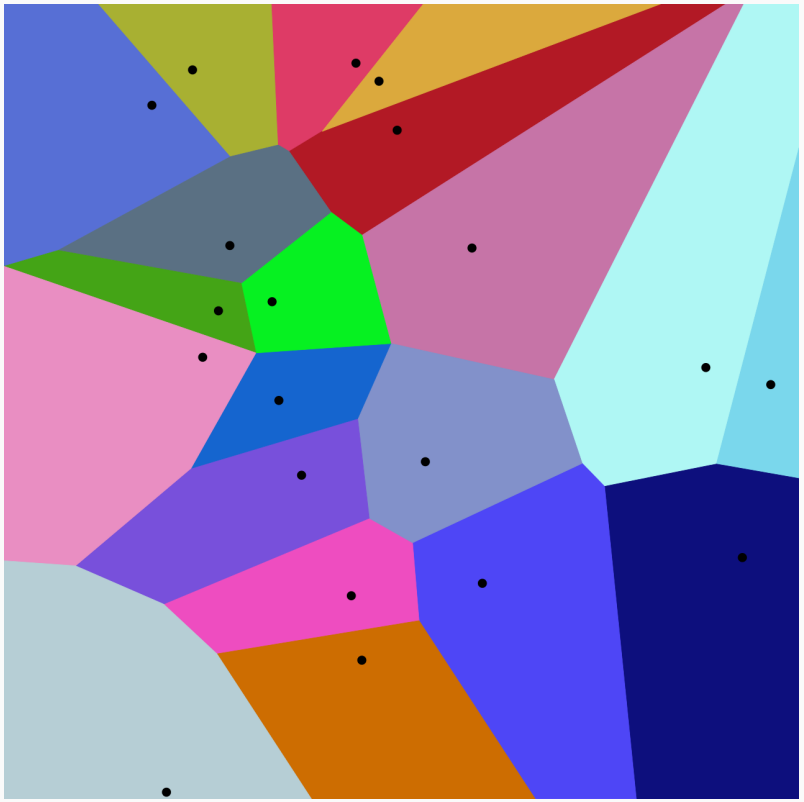


Image: wikipedia.org

- Cluster centers: $\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}$
- Clusters: $\mathcal{D}_k = \{\boldsymbol{x} \mid \boldsymbol{x} \in \mathcal{D} \ \wedge \ k = \arg\min_l \|\boldsymbol{x} - \boldsymbol{\mu}^{(l)}\|_2\}$
- The **k-means problem**: Minimize the sum of squared distances between data points and their assigned cluster center:

$$\min_{\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^K} \mathcal{L}_{WCSS} = \min_{\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^K} \sum_{k=1}^{K} \sum_{\boldsymbol{x}\in\mathcal{D}_k} \left\|\boldsymbol{x} - \boldsymbol{\mu}^{(k)}\right\|_2^2$$

- The objective $\mathcal{L}_{WCSS}$ is called the **within-cluster sum-of-squares** (WCSS), **inertia**, or simply **k-means objective**
- For a proper optimization problem, we should explicitly encode the cluster assignments $\mathcal{D}_k$

- Let $z_{i,k}$ be **binary indicator variables** defined as

$$z_{i,k} = \begin{cases} 1 & \text{if data point } \boldsymbol{x}^{(i)} \text{ is assigned to } \boldsymbol{\mu}^{(k)} \\ 0 & \text{otherwise} \end{cases}$$

- Let $\boldsymbol{Z}$ be the $N \times K$ matrix collecting all indicators:

$$\boldsymbol{Z} = \begin{pmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,K} \\ z_{2,1} & z_{2,2} & \cdots & z_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ z_{N,1} & z_{N,2} & \cdots & z_{N,K} \end{pmatrix}$$

- Rows are a **one-hot encoding**

- We can now write the k-means problem as

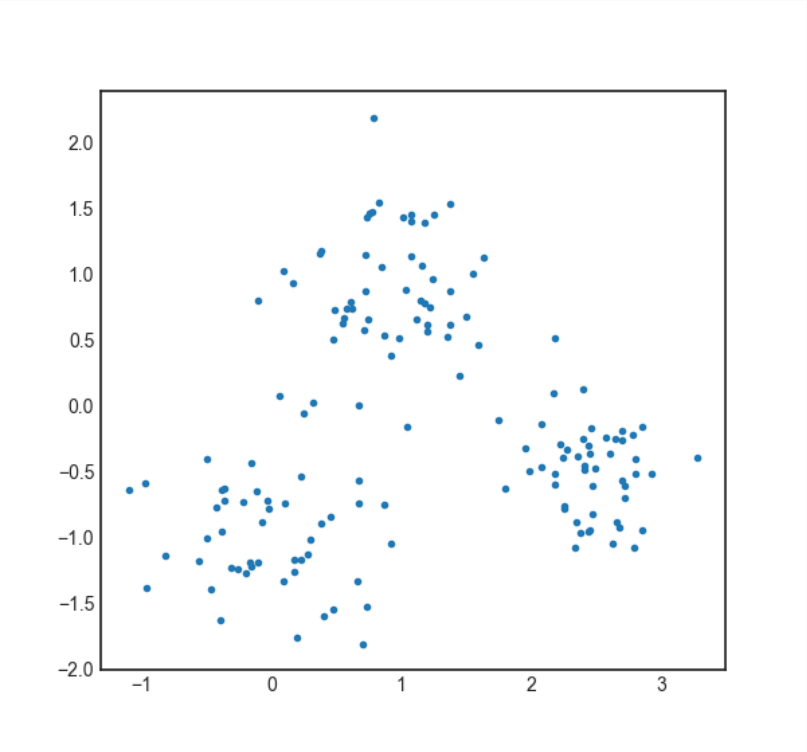
$$\min_{\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^K,\boldsymbol{Z}} \quad \sum_{i=1}^{N}\sum_{k=1}^{K} z_{i,k} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(k)} \right\|^2$$

$$\text{s.t.} \quad z_{i,k} \in \{0,1\} \qquad \text{(integer constraints)}$$

$$\sum_{k} z_{i,k} = 1 \quad i = 1,\ldots,N$$

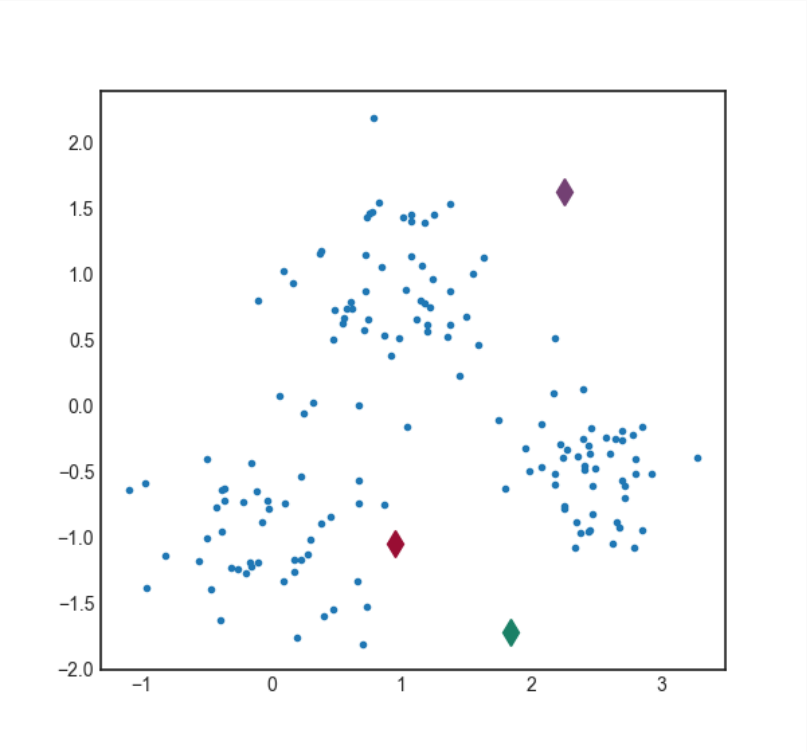
- **Mixed Integer Quadratic Program (MIQP)**
- Constraints for $z_{i,k}$ ensure that each data point is assigned to exactly one cluster center (one hot encoding)
- Multiplication with indicator $z_{i,k}$ in the objective ensures that only the distance to the assigned cluster center counts
- We can now decompose the problem and alternate between $\boldsymbol{Z}$ and $\{\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^K\}$, leading to the **k-means algorithm**

Assume the data set below. It seems to consist of roughly three clusters, so we set $K = 3$.



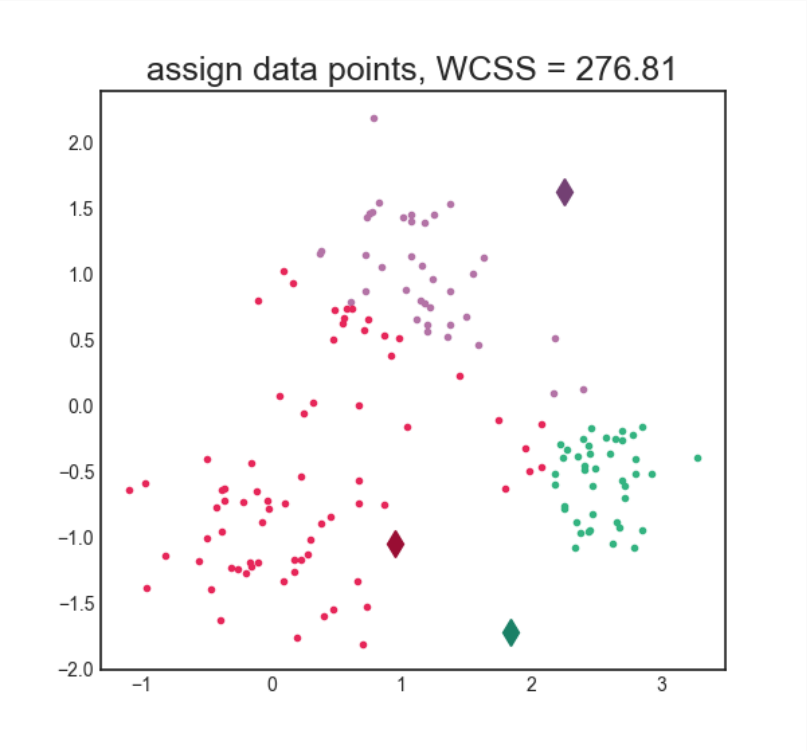
Let's "guess" a first solution for $\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \boldsymbol{\mu}^{(3)}$, e.g. by drawing them uniformly between minimal and maximal range in each dimension.

Given $\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \boldsymbol{\mu}^{(3)}$, how do we chose $\boldsymbol{Z}$?



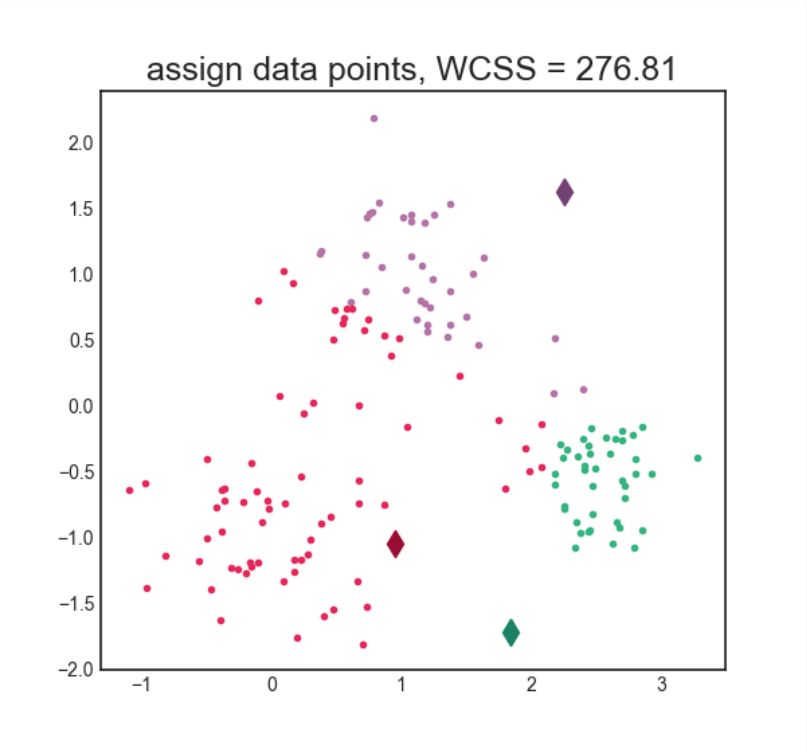
Given $\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \boldsymbol{\mu}^{(3)}$, we can easily find optimal $\boldsymbol{Z}$, by setting

$$z_{i,k} = \begin{cases} 1 & \text{if cluster center } \boldsymbol{\mu}^{(k)} \text{ is closest to } \boldsymbol{x}^{(i)} \\ 0 & \text{otherwise} \end{cases}$$



Now we have a new cluster assignment $\boldsymbol{Z}$ and the cluster centers seem "outdated."

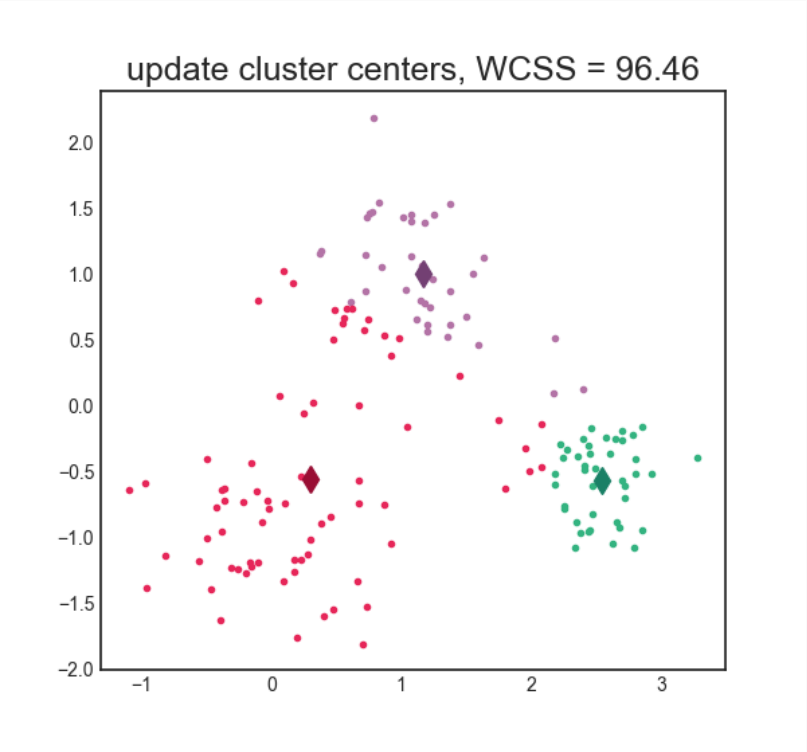
How do we chose new $\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \boldsymbol{\mu}^{(3)}$?



Intuitively, we would set $\boldsymbol{\mu}^{(k)}$ to the **mean vector** of the $k^{\text{th}}$ cluster (hence **k-means**):

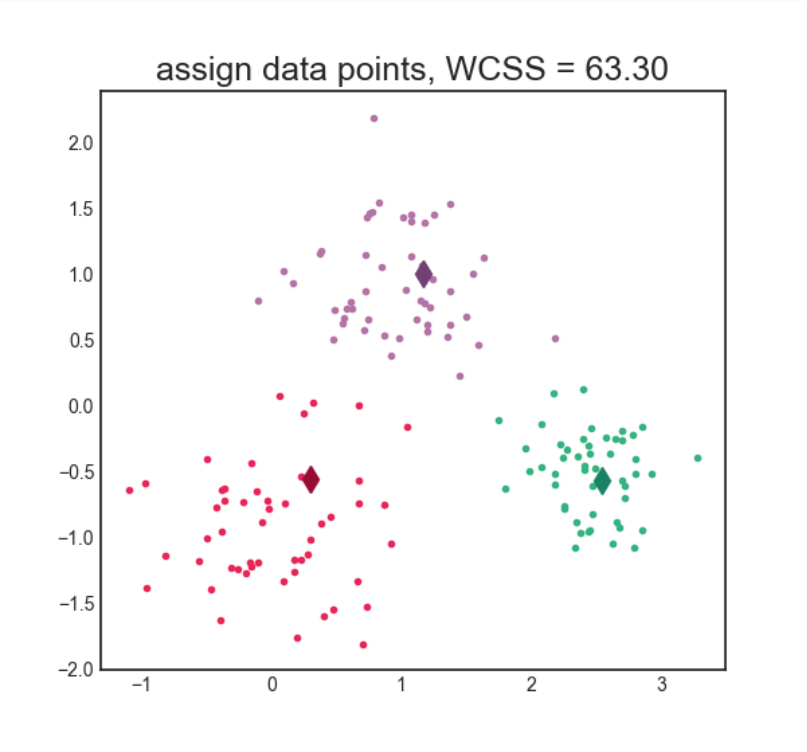
$$\boldsymbol{\mu}^{(k)} \leftarrow \frac{1}{|\mathcal{D}_k|} \sum_{\boldsymbol{x} \in \mathcal{D}_k} \boldsymbol{x}$$

Now $\boldsymbol{Z}$ seems outdated.



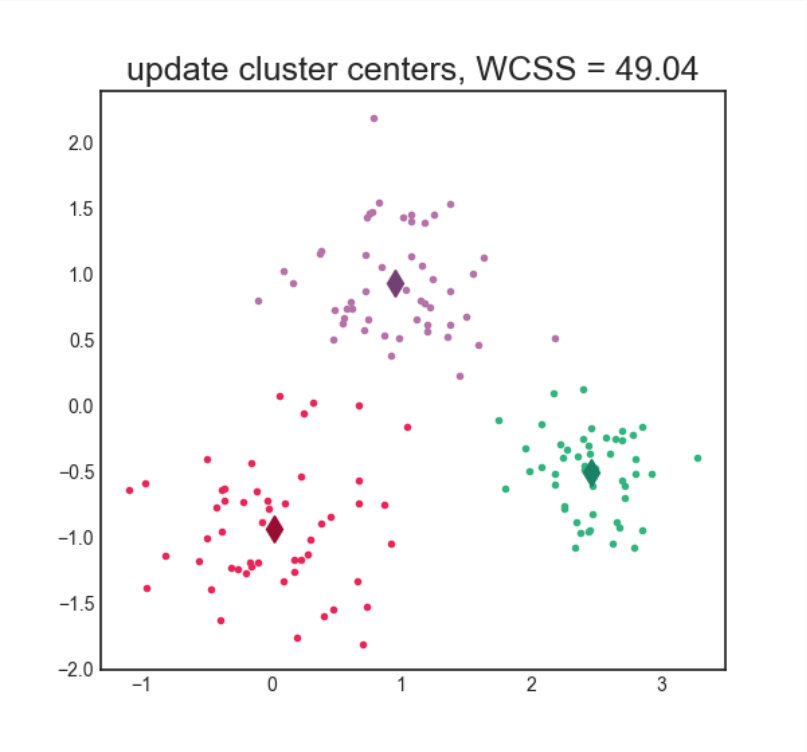
Note that we have effectively the **same situation as after initialization,** i.e. we set again

$$z_{i,k} = \begin{cases} 1 & \text{if cluster center } \boldsymbol{\mu}^{(k)} \text{ is closest to } \boldsymbol{x}^{(i)} \\ 0 & \text{otherwise} \end{cases}$$



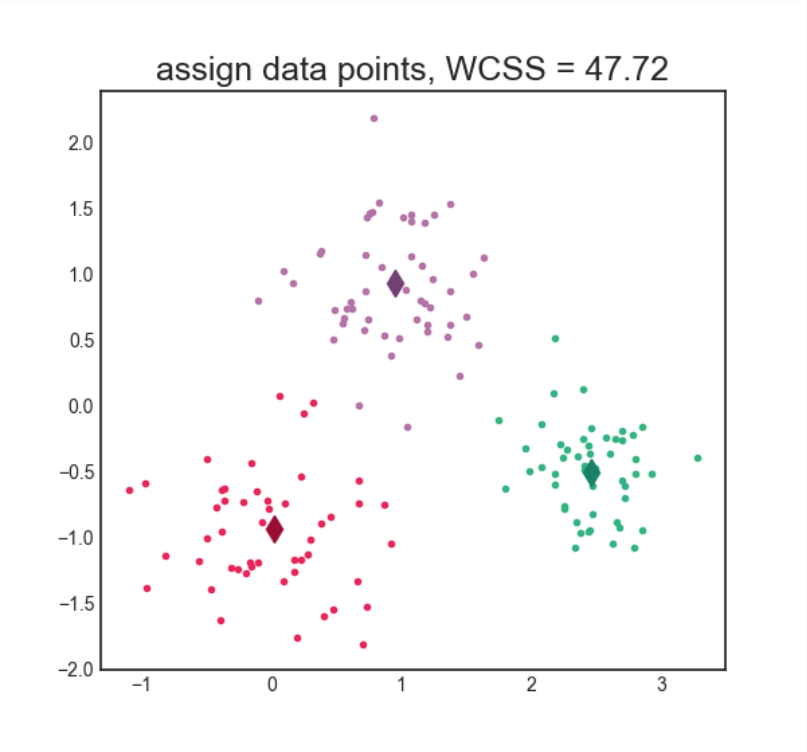
Again, update mean vectors:

$$\boldsymbol{\mu}^{(k)} \leftarrow \frac{1}{|\mathcal{D}_k|} \sum_{\boldsymbol{x} \in \mathcal{D}_k} \boldsymbol{x}$$



Re-assign the data points to clusters:

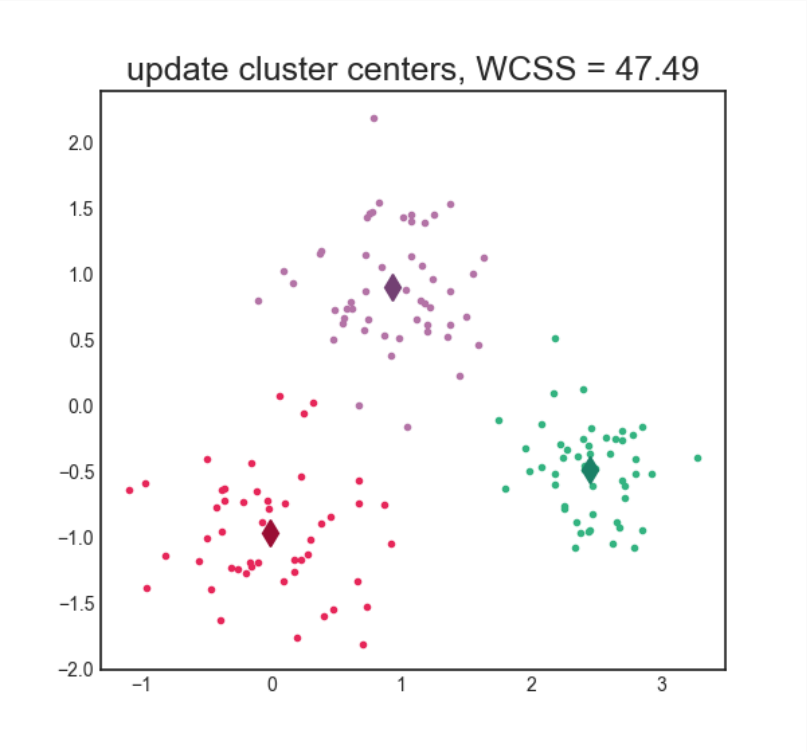
$$z_{i,k} = \begin{cases} 1 & \text{if cluster center } \boldsymbol{\mu}^{(k)} \text{ is closest to } \boldsymbol{x}^{(i)} \\ 0 & \text{otherwise} \end{cases}$$



Update means:

$$\boldsymbol{\mu}^{(k)} \leftarrow \frac{1}{|\mathcal{D}_k|} \sum_{\boldsymbol{x} \in \mathcal{D}_k} \boldsymbol{x}$$

After that, the assignments $\boldsymbol{Z}$ won't change any more. Hence, also $\boldsymbol{\mu}^{(k)}$'s won't change any more. The algorithm has **converged.**



# K-means Algorithm

**input:** Data $\mathcal{D} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\}$

**output:** Cluster centers $\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}$, assignment matrix $\boldsymbol{Z}$

Initialize $\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}$ randomly

**for** *counter* $= 1 \ldots$ *max number of iterations* **do**

  **for** $i = 1 \ldots N$ **do**

    let $k^* = \arg\min_k \ \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(k)}\|_2$

    set $z_{i,k} \leftarrow \begin{cases} 1 & \text{if } k = k^* \\ 0 & \text{otherwise} \end{cases}$

  **end**

  **for** $k = 1 \ldots K$ **do**

    let $\mathcal{D}_k = \{\boldsymbol{x}^{(i)} \mid z_{i,k} = 1\}$

    $\boldsymbol{\mu}^{(k)} \leftarrow \frac{1}{|\mathcal{D}_k|} \sum_{\boldsymbol{x} \in \mathcal{D}_k} \boldsymbol{x}$

  **end**

  **if** *WCSS = WCSS of previous iteration* **then**

    break

  **end**

**end**

# K-Means Problem vs. K-Means Algorithm

- Nitpick: difference between **k-means problem** vs. **k-means algorithm**
- **k-means problem**: the optimization problem

$$\begin{aligned} \min_{\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^K,\boldsymbol{Z}} \quad & \sum_{i=1}^{N}\sum_{k=1}^{K} z_{i,k} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(k)} \right\|_2^2 \\ \text{s.t.} \quad & z_{i,k} \in \{0,1\} \qquad \text{(integer constraints)} \\ & \sum_{k} z_{i,k} = 1 \quad i = 1,\ldots,N \end{aligned}$$

- **k-means algorithm**: the iterative algorithm alternating between updating $\boldsymbol{Z}$ and updating $\boldsymbol{\mu}^{(1)},\ldots,\boldsymbol{\mu}^{(K)}$. Also called **Lloyd's algorithm**.
- Is the k-means algorithm solving the k-means problem?

# Convergence of K-Means Algorithm

Does the k-means algorithm always converge?

Or can it "oscillate"?

# Convergence of K-Means Algorithm

Recall the k-means problem:

$$\begin{aligned} \min_{\boldsymbol{\mu}^{(1)},\ldots,\boldsymbol{\mu}^{(K)},\boldsymbol{Z}} \quad & \sum_{i=1}^{N}\sum_{k=1}^{K} z_{i,k} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(k)} \right\|_2^2 \\ \text{s.t.} \quad & z_{i,k} \in \{0,1\} \\ & \sum_{k} z_{i,k} = 1 \quad i = 1,\ldots,N \end{aligned}$$

The k-means algorithm really alternates the following two steps:

- (globally) optimize the k-means problem w.r.t. $\boldsymbol{Z}$, keeping $\boldsymbol{\mu}^{(1)},\ldots,\boldsymbol{\mu}^{(K)}$ fixed
- (globally) optimize the k-means problem w.r.t. $\boldsymbol{\mu}^{1},\ldots,\boldsymbol{\mu}^{K}$, keeping $\boldsymbol{Z}$ fixed

# (I) Optimality of Cluster Assignments

For (currently) fixed $\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}$, k-means computes $\boldsymbol{Z}$ as:

**for** $i = 1 \ldots N$ **do**

$$\text{let } k^* = \arg\min_k \; \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(k)}\|$$

$$\text{set } z_{i,k} \leftarrow \begin{cases} 1 & \text{if } k = k^* \\ 0 & \text{otherwise} \end{cases}$$

**end**

This is clearly optimal, since any other choice for $z_{i,k}$ would lead to a larger objective:

$$\sum_{i=1}^{N} \sum_{k=1}^{K} z_{i,k} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(k)} \right\|_2^2$$

# (II) Optimality of Means

For (currently) fixed $\boldsymbol{Z}$, k-means computes $\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^K$ as:

**for** $k = 1 \ldots K$ **do**
  let $\mathcal{D}_k = \{\boldsymbol{x}^{(i)} \mid z_{i,k} = 1\}$
  $\boldsymbol{\mu}^{(k)} \leftarrow \frac{1}{|\mathcal{D}_k|} \sum_{\boldsymbol{x} \in \mathcal{D}_k} \boldsymbol{x}$
**end**

Is this minimizing the k-means objective?

$$\min_{\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^K} \quad \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in \mathcal{D}_k} \left\| \boldsymbol{x} - \boldsymbol{\mu}^{(k)} \right\|_2^2$$

Observation: the objective has the form $\sum_{k=1}^{K} \mathcal{L}_k$, where

$$\mathcal{L}_k = \sum_{\boldsymbol{x} \in \mathcal{D}_k} \left\| \boldsymbol{x} - \boldsymbol{\mu}^{(k)} \right\|_2^2$$

depends only on $\boldsymbol{\mu_k}$ and we can minimize $\mathcal{L}_k$ **independently** for each $k$.

# (II) Optimality of Means cont'd

Since $\mathcal{L}_k = \sum_{\boldsymbol{x} \in \mathcal{D}_k} \sum_{d=1}^{D} \left( x_d - \mu_d^{(k)} \right)^2$, the gradient w.r.t. $\boldsymbol{\mu}^{(k)}$ is

$$\frac{\partial \mathcal{L}_k}{\partial \mu_d^{(k)}} = \sum_{\boldsymbol{x} \in \mathcal{D}_k} 2(x_d - \mu_d^{(k)}) \qquad \nabla_{\boldsymbol{\mu}^{(k)}} \mathcal{L}_k = \sum_{\boldsymbol{x} \in \mathcal{D}_k} 2\left(\boldsymbol{x} - \boldsymbol{\mu}^{(k)}\right)$$

Setting the gradient to zero yields indeed the mean:

$$\sum_{\boldsymbol{x} \in \mathcal{D}_k} 2\left(\boldsymbol{x} - \boldsymbol{\mu}^{(k)}\right) = 0$$

$$\sum_{\boldsymbol{x} \in \mathcal{D}_k} \boldsymbol{\mu}^{(k)} = \sum_{\boldsymbol{x} \in \mathcal{D}_k} \boldsymbol{x}$$

$$|\mathcal{D}_k| \, \boldsymbol{\mu}^{(k)} = \sum_{\boldsymbol{x} \in \mathcal{D}_k} \boldsymbol{x}$$

$$\boldsymbol{\mu}^{(k)} = \frac{1}{|\mathcal{D}_k|} \sum_{\boldsymbol{x} \in \mathcal{D}_k} \boldsymbol{x}$$

# Convergence of K-Means Algorithm cont'd

$$\min_{\boldsymbol{\mu}^{(1)},\ldots,\boldsymbol{\mu}^{(K)},\boldsymbol{Z}} \quad \sum_{i=1}^{N}\sum_{k=1}^{K} z_{i,k} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(k)} \right\|^2$$

$$\text{s.t.} \quad z_{i,k} \in \{0,1\}$$

$$\sum_{k} z_{i,k} = 1 \quad i = 1,\ldots,N$$

- We have shown that the k-means algorithm alternates between:
  - globally optimizing w.r.t. $\boldsymbol{Z}$, $\boldsymbol{\mu}^{(1)},\ldots,\boldsymbol{\mu}^{(K)}$ fixed
  - globally optimizing w.r.t. $\boldsymbol{\mu}^{(1)},\ldots,\boldsymbol{\mu}^{(K)}$, $\boldsymbol{Z}$ fixed
- The k-means objective **can only decrease** after each step
- Moreover, there are only **finitely many** indicator matrices $\boldsymbol{Z}$
- We can **strictly** decrease at most as often as there are different $\boldsymbol{Z}$
- **Thus, the k-means algorithm converges in terms of the WCSS objective**

# Global Optimum?

So, k-means performs **globally optimal sub-steps,**

- minimizing $\boldsymbol{Z}$
- minimizing $\{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}\}$

in alternating fashion. This is known as **block-coordinate descent** in "blocks" $\boldsymbol{Z}$ and $\{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}\}$. Thus, does the k-means algorithm globally solve the k-means problem?

# Global Optimum?

So, k-means performs **globally optimal sub-steps,**

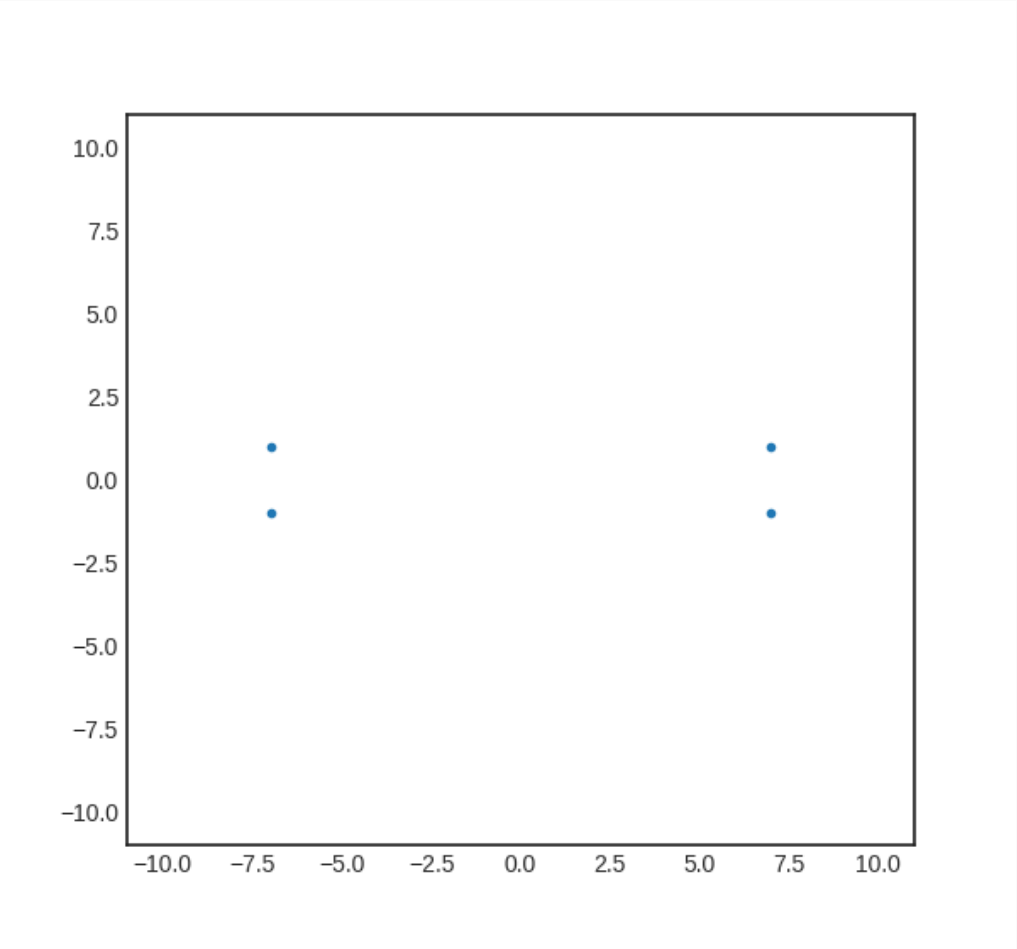
- minimizing $\boldsymbol{Z}$
- minimizing $\{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}\}$

in alternating fashion. This is known as **block-coordinate descent** in "blocks" $\boldsymbol{Z}$ and $\{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}\}$. Thus, does the k-means algorithm globally solve the k-means problem?

**No!**

- The k-means algorithm typically converges to a sub-optimal solution!
- The actual solution depends on the initialization and can in theory be arbitrarily bad
- Solving the k-means problem can be shown to be **NP-hard**!

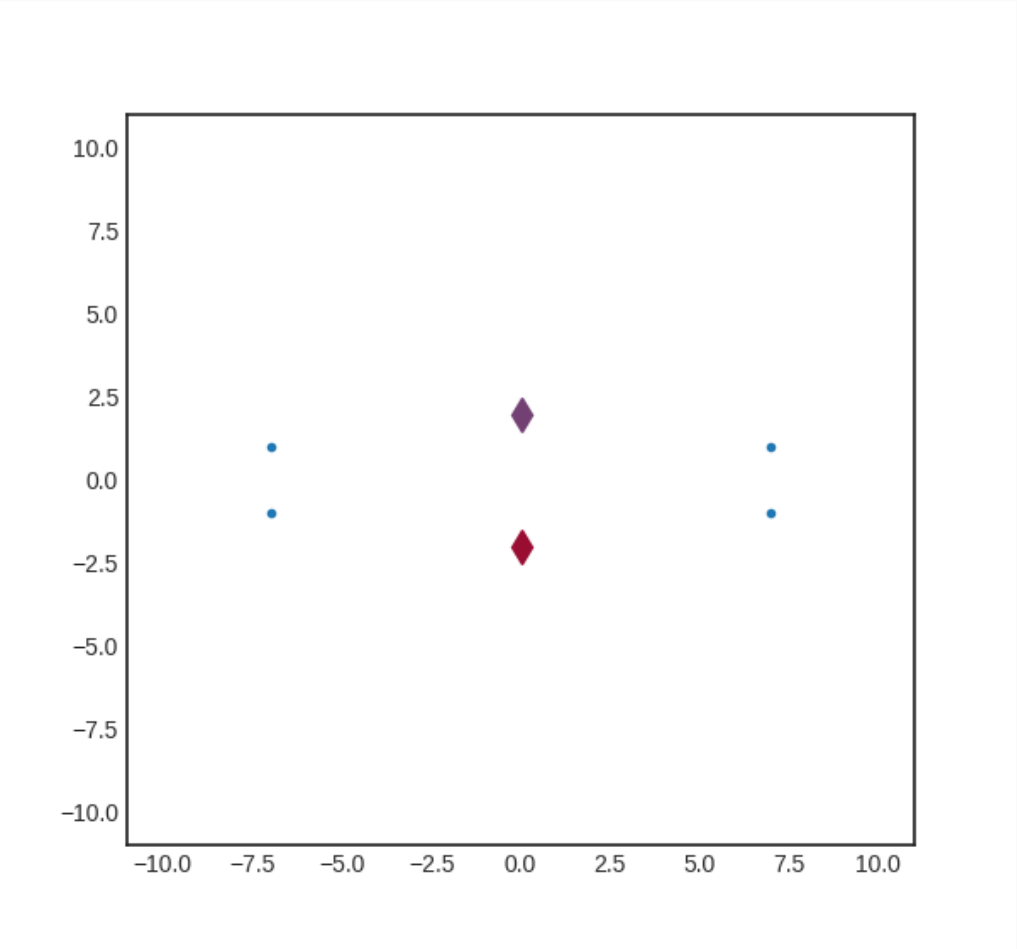
Assume the data set below and assume we set $K = 2$.



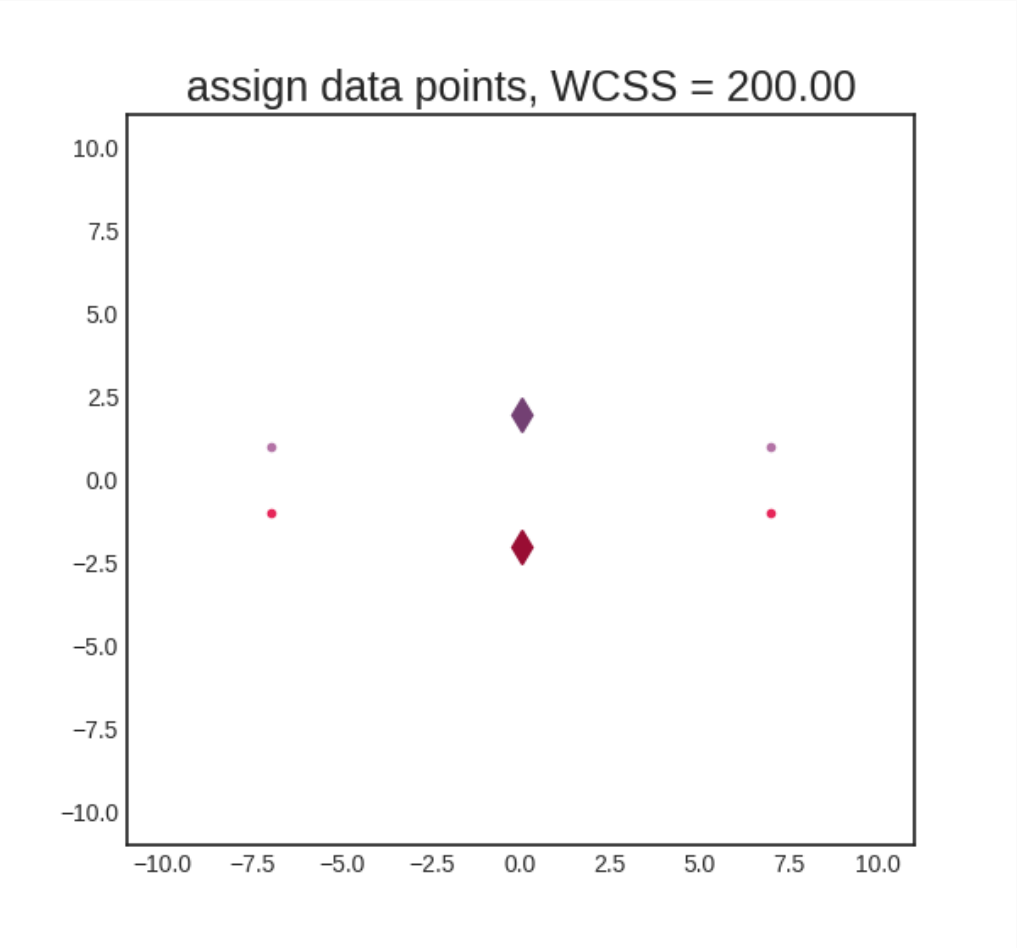
# Sub-optimal Solution

Example

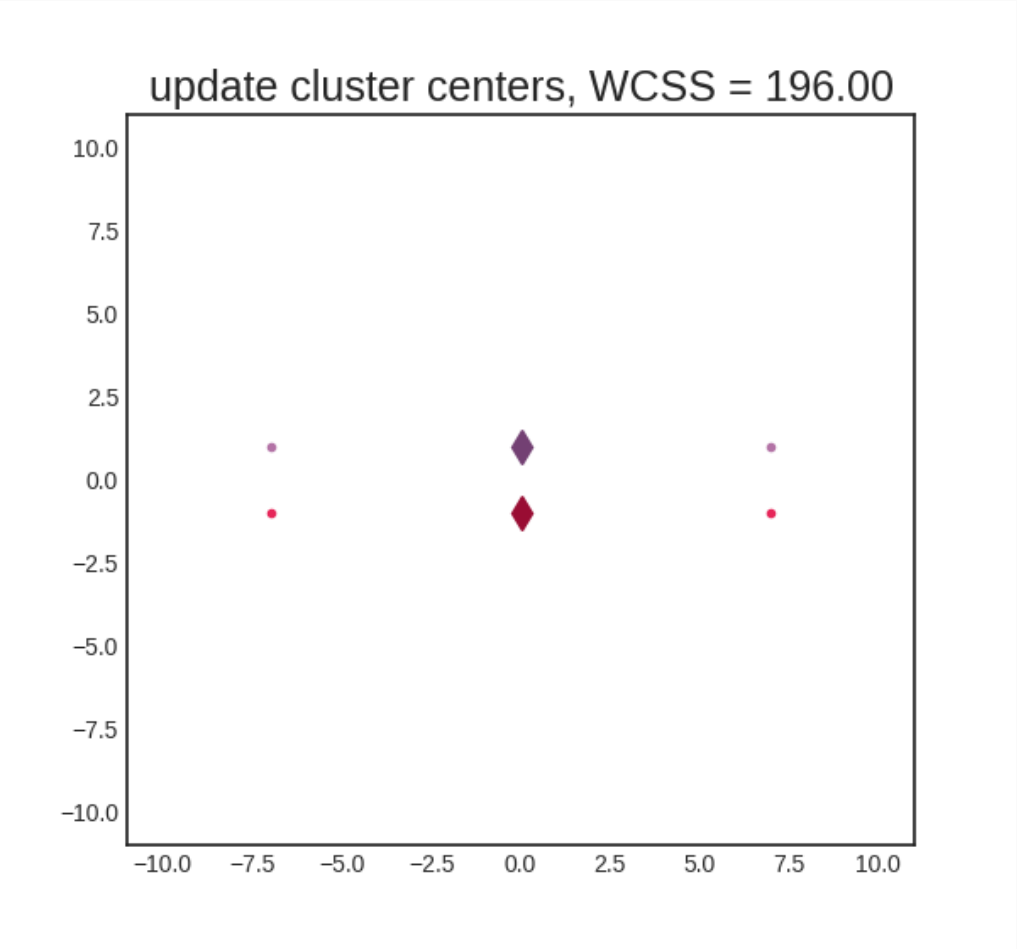
Say the cluster centers are initialized to lie on the $y$-axis.



Assign data points to clusters. . .



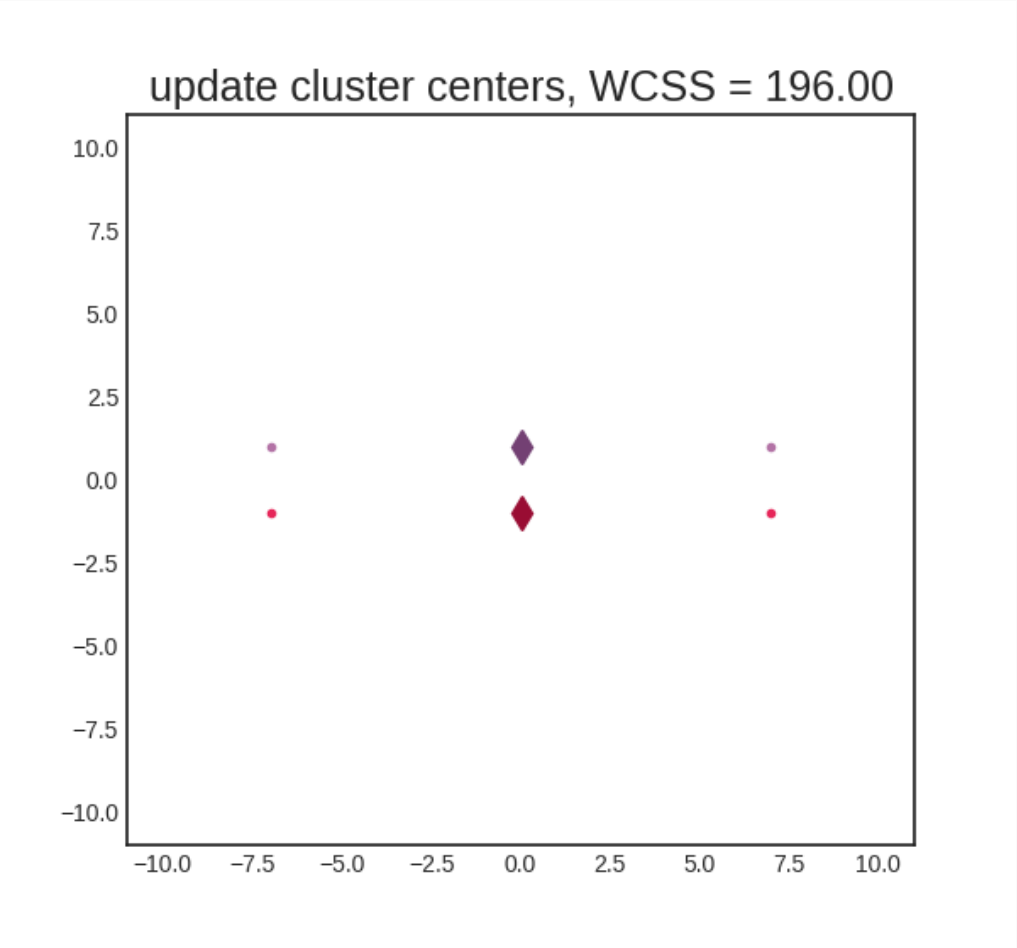
Update cluster centers. . .



# Sub-optimal Solution

Example

K-means has already converged, to an evidently a sub-optimal result. By making the rectangle described by the data points wider, the result can be made arbitrarily bad.



# Avoiding Bad Clustering

- Random re-starts, i.e., run k-means multiple times with different random initializations
- Pick the clustering of the best run, i.e., with the lowest final objective
- Use clever initializations, like k-means++

# K-Means++

- **K-Means++** is standard k-means with a clever initialization, proposed by Arthur & Vassilvitskii, 2007
- **Basic idea: use data points as initial cluster seeds $\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}$, spreaded over the whole dataset**
- $\boldsymbol{\mu}^{(1)}$ is selected uniform at random from $\mathcal{D}$
- For $k > 1$:
  - compute the distance $d_i$ between any $\boldsymbol{x}^{(i)}$ and the closest $\boldsymbol{\mu}^{(l)}$, for $l < k$
  - select $\boldsymbol{\mu}^{(k)} = \boldsymbol{x}^{(i)}$ with probability proportional to $d_i^2$
- Thus, samples which are far from already selected seeds are preferred

# K-Means++ cont'd

- The selection of initial seeds adds to the overall computational time
- However, the authors observed that with their initialization the main phase converges much faster, so that in total k-means++ is usually faster than vanilla k-means
- They typically observed much better clustering results than for vanilla k-means
- Moreover, in expectation over the randomness used in its initialization, k-means++ is guaranteed to achieve an objective which is only a multiplicative factor of $\mathcal{O}(\log K)$ worse than the global optimal solution!
- K-means++ is the de-facto standard implementation of k-means in many libraries

# Extending K-Means to Other Distances

- We can adapt k-means to distances other than Euclidean distance
- For example, we could formulate an $\ell_1$-version of k-means:

$$\begin{aligned} \min_{\boldsymbol{\mu}^{(1)},\ldots,\boldsymbol{\mu}^{(K)},\boldsymbol{Z}} \quad & \sum_{i=1}^{N}\sum_{k=1}^{K} z_{i,k} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(k)} \right\|_1 \\ \text{s.t.} \quad & z_{i,k} \in \{0,1\} \\ & \sum_{k} z_{i,k} = 1 \quad i = 1,\ldots,N \end{aligned}$$

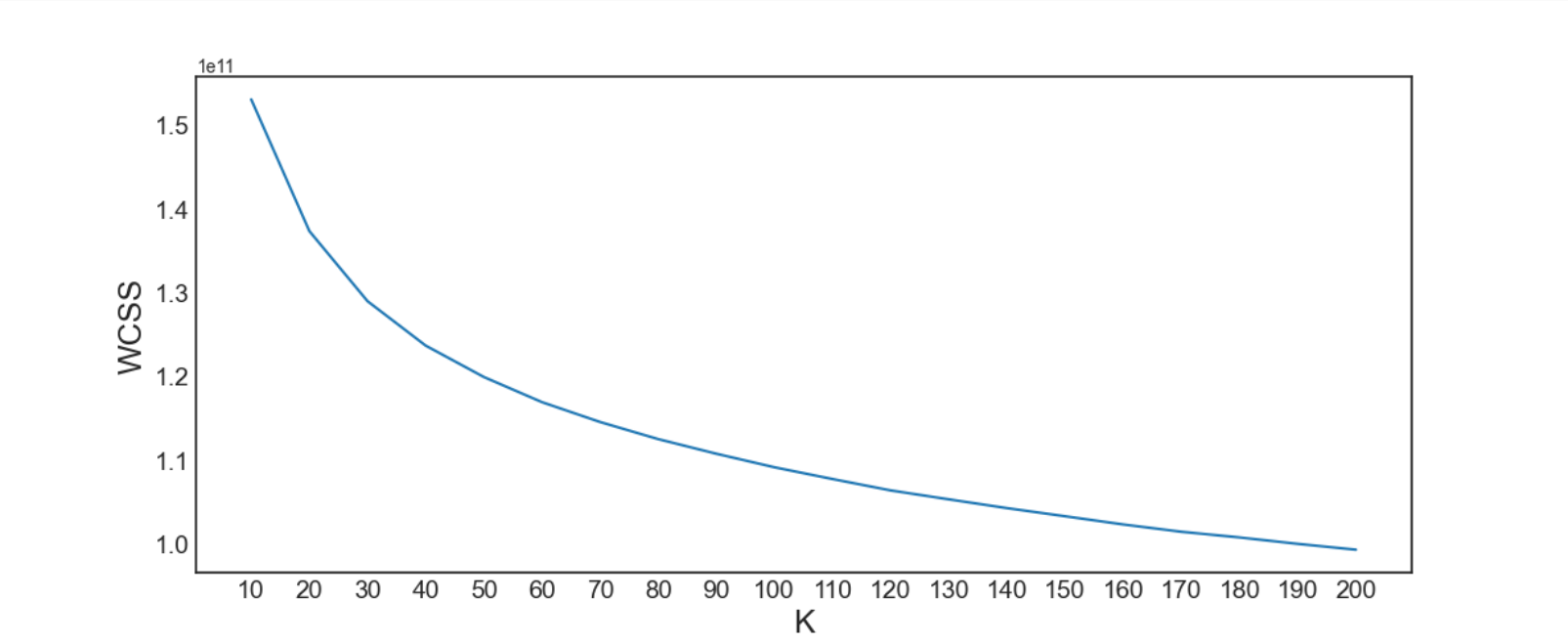
- We can easily adapt the k-means algorithm to this case
  - optimizing $\boldsymbol{Z}$: $\boldsymbol{x}^{(i)}$ is assigned to closest $\boldsymbol{\mu}^{(k)}$ in $\ell_1$-sense
  - $\boldsymbol{\mu}^{(k)}$ are updated by using the **median** instead of the mean:

$$\boldsymbol{\mu}_d^{(k)} \leftarrow \mathit{median}(\mathcal{D}_k)$$

    where median acts dimension-wise
- This method is known as **k-median algorithm**

# Selecting $K$

- The k-means algorithm assumes that $K$ is given
- A simple method to find a suitable $K$ is to run k-means for a range of different $K$'s
- For each $K$, we record the final objective WCSS
- Plotting the objective against the range of $K$ allows us to "visually select" a suitable $K$

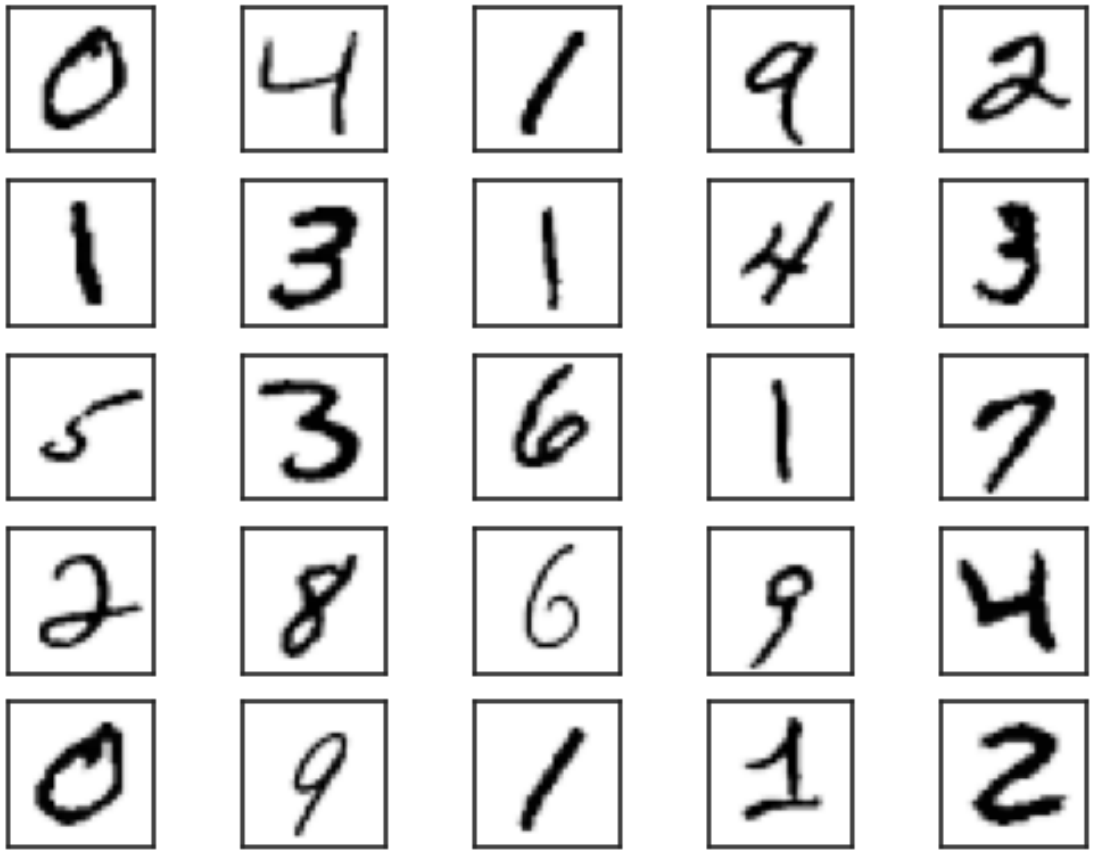


# Application: Data Exploration, Data Summary

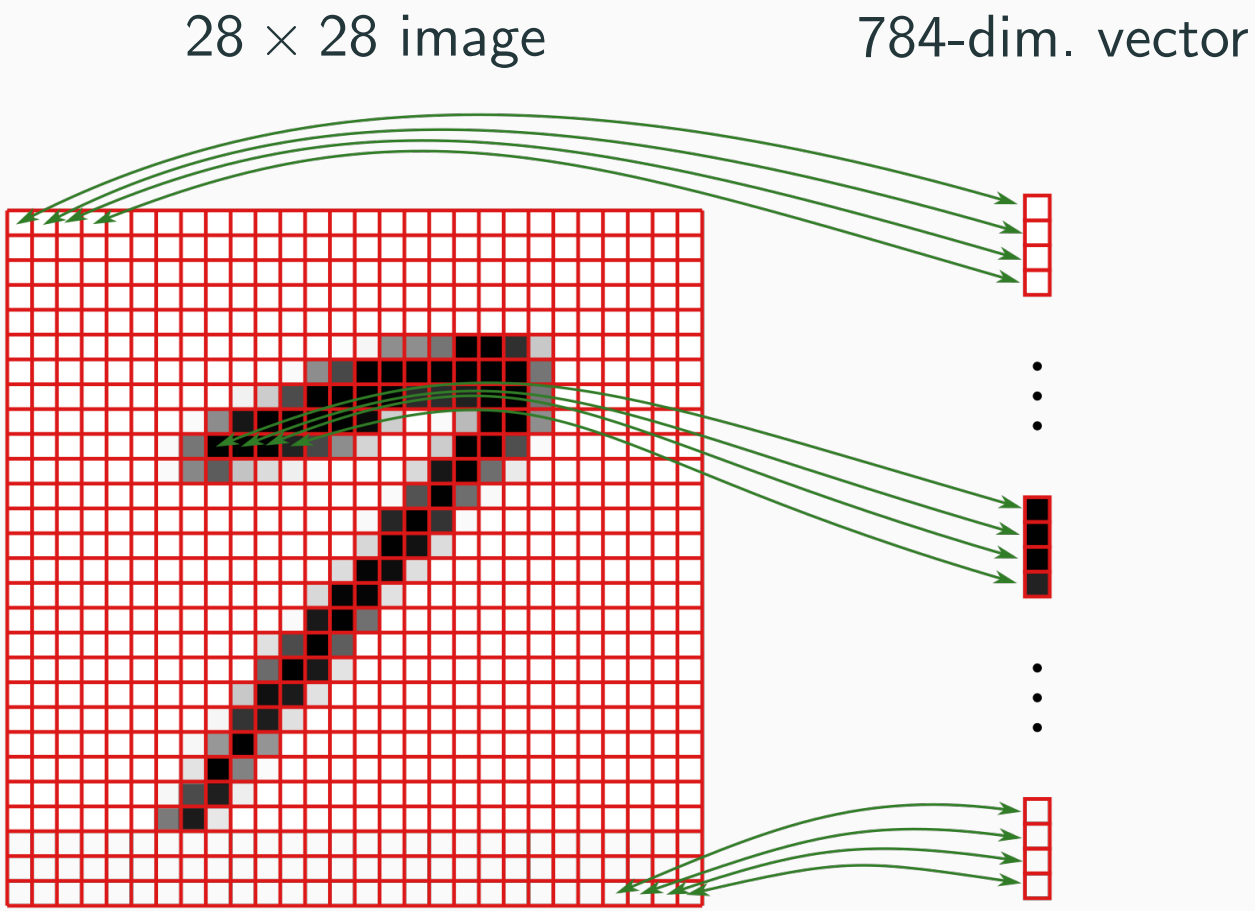
- Data sets can be massive nowadays
- Clustering algorithms like k-means can extract a few exemplars or prototypes, representing whole clusters in the data

# Clustering MNIST

- collection of 60,000 images of handwritten digits
- images are $28 \times 28$ pixels, with 256 gray scales
- white $= 0$, black $= 255$

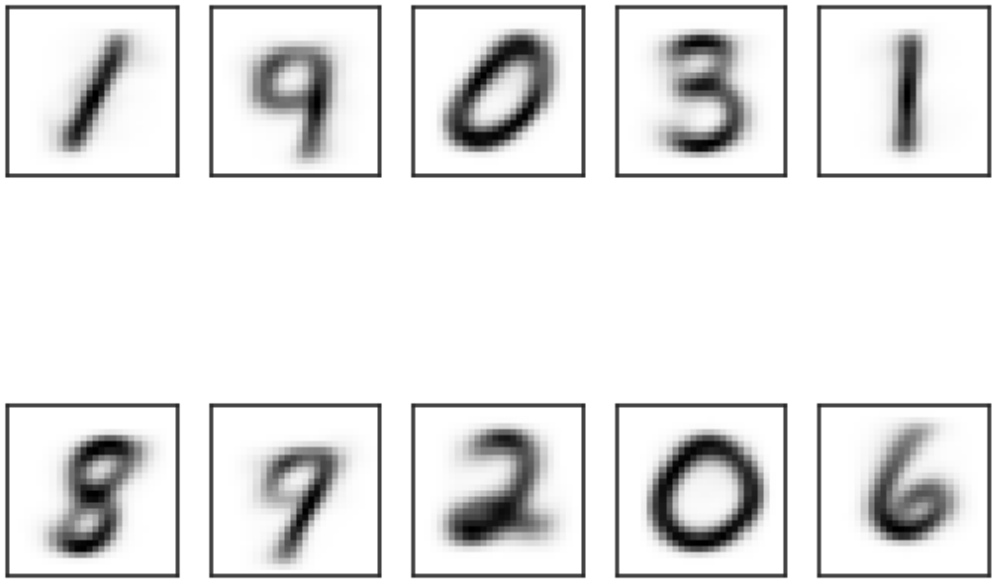


# Interpreting Images as Vectors



# Clustering MNIST cont'd

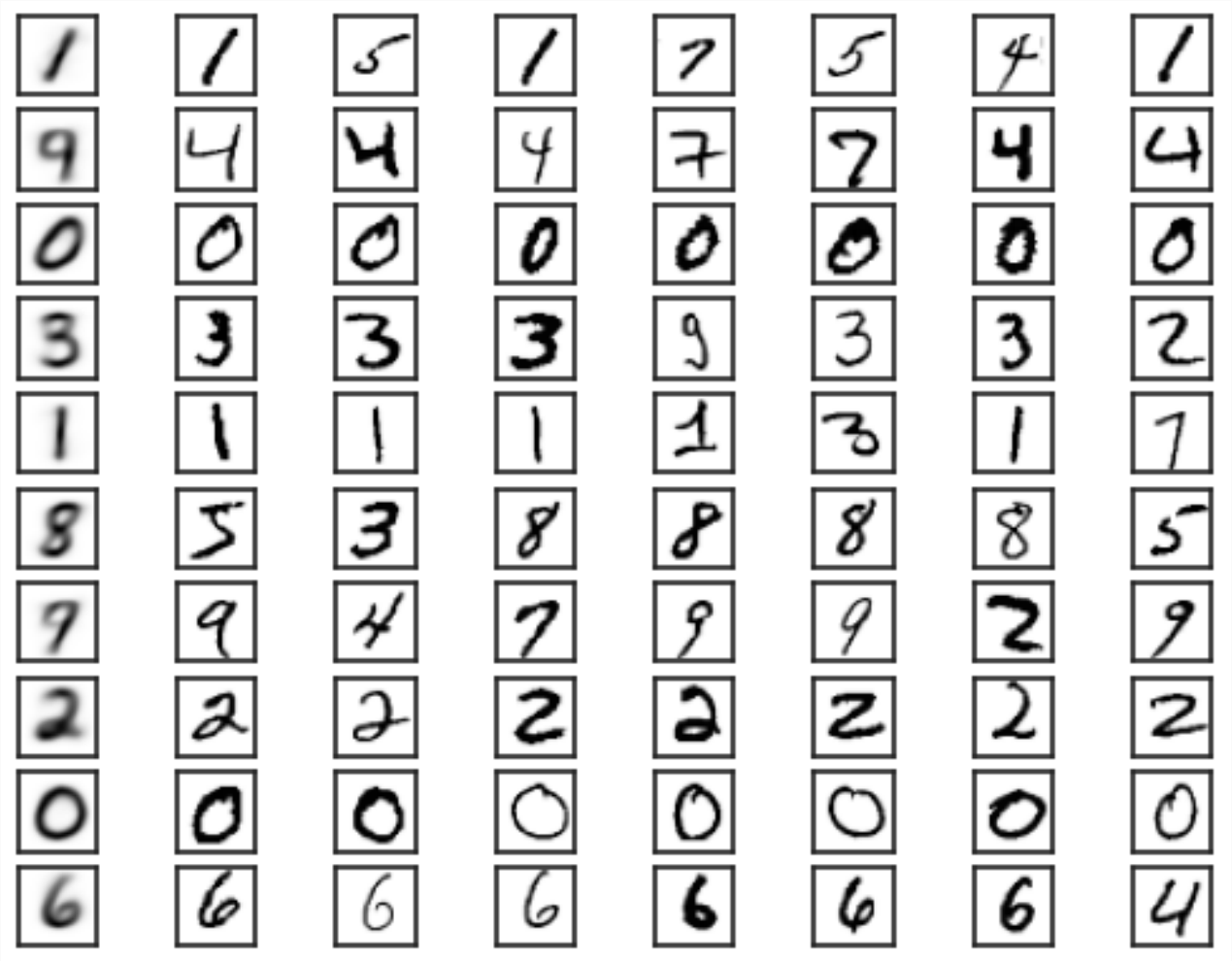
Clustering with $K = 10$, using k-means++ and 10 restarts, delivers following centers $\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(10)}$, which summarize the whole dataset. We see that not all "natural classes" are represented.



# Clustering MNIST cont'd

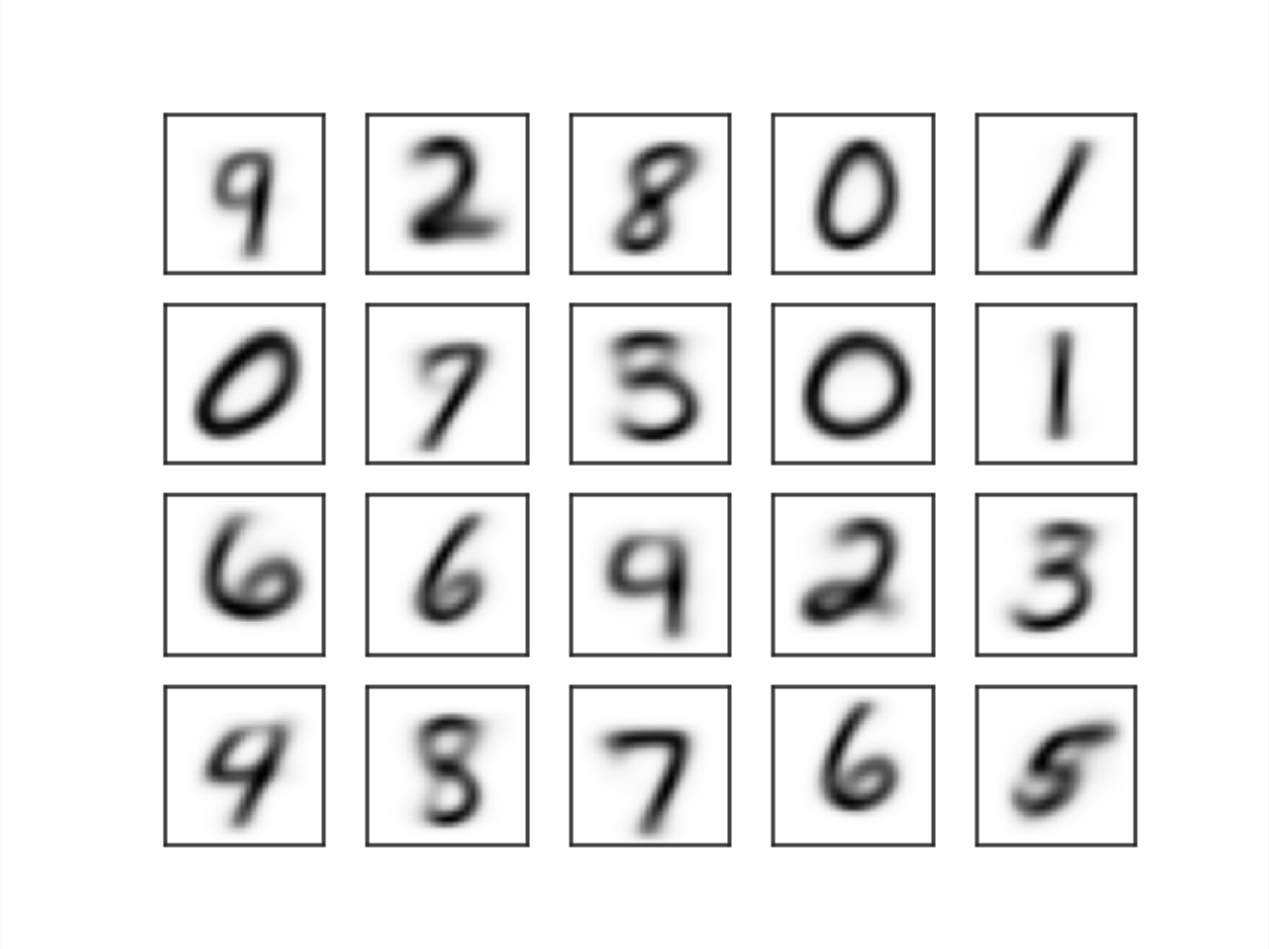
Example

Plotting some examples from each cluster. First column corresponds to $\mu^{(k)}$, columns $2-8$ to examples from the cluster



# Clustering MNIST cont'd

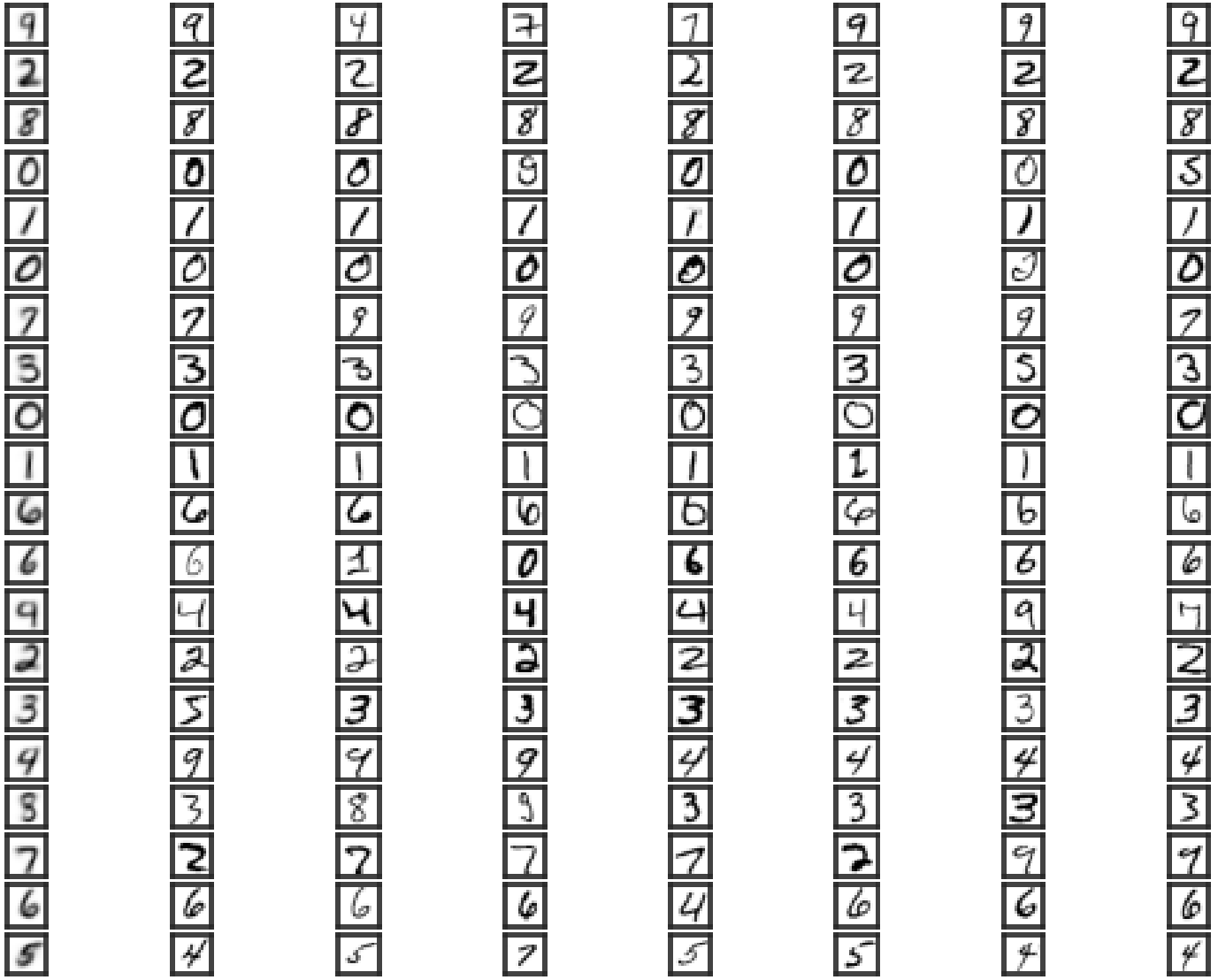
Same as before, but with $K = 20$. Now all "natural classes" are represented (with duplicates).



# Clustering MNIST cont'd

Example

Plotting some examples from each cluster.



# Application: Image Compression

Example

Consider following color image, of size $683 \times 1024$ pixels:

# Application: Image Compression

**Example**

- Each pixel has 3 channels, red, green, and blue
- Each channel is encoded with 1 byte, yielding $16.7M$ colors
- In uncompressed format we require $683 \times 1024 \times 3 = 2MB$ memory to store this image
- We can apply k-means as a simple lossy compression algorithm
- This technique is also known as **vector quantization**

# Image Compression with K-Means (Vector Quantization)

- Interpret each pixel as 3-dimensional data point
- For the previous example image, this yields a "data set" of $683 \times 1024 = 699392$ data points
- Apply k-means with some $K$, yielding $\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}$
- Each $\boldsymbol{\mu}^{(k)}$ is a 3-dimensional vector representing a color, thus $\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}$ can be seen as a "learned color palette" for the input image
- Now, each pixel value is replaced by its closest center $\boldsymbol{\mu}^{(k)}$
- Crucially, for each pixel, we just need to store the index $k$ of the closest $\boldsymbol{\mu}^{(k)}$, which can be done with $\log(K)$ bits
- Additionally, we also need to store $\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}$, which can be done with $K \times 3 \times 8$ bits

# Image Compression $K = 2$

Example

Compression ratio $= 24$, signal-to-noise-ratio $= 10.99\,\mathrm{dB}$

# Image Compression $K = 16$

Example

Compression ratio $= 6$, signal-to-noise-ratio $= 19.93\,\mathrm{dB}$

# Image Compression $K = 64$

Example

Compression ratio $= 4$, signal-to-noise-ratio $= 25.12\,\mathrm{dB}$

# Image Compression $K = 128$

Compression ratio $= 3.42$, signal-to-noise-ratio $= 27.35\,\mathrm{dB}$

# Summary

- **Clustering**: group similar objects
- Not well-defined, many choices to define clustering
- Data exploration, data summary, data compression
- **K-means problem**: find $K$ mean vectors, which specify clusters as set of points which are closest to them
- Objective: **Within-Cluster Sum-of-Squared distances (WCSS)**
- NP-hard to optimize
- **K-mean algorithm**: alternate between
  - optimal assignment of points to cluster centers
  - taking means of cluster
- Reduces the objective in each step, but converges to a sub-optimal solution
- Re-starts, k-means++

# Please Evaluate

ML1 (VO) is nominated for an Award for Excellence in Teaching