EEP 596: Adv Intro ML || Lecture 8

Dr. Karthik Mohan

Univ. of Washington, Seattle

January 31, 2023

Logistics

1. Anything to discuss?

Last time

1. Random forests
2. Multi-class Classification

Today

1. Conceptual Assignment Review
2. Clustering overview
3. kMeans
4. kMeans++

The clustering problem

Clustering vs Classification

clustering – unsupervised Learning

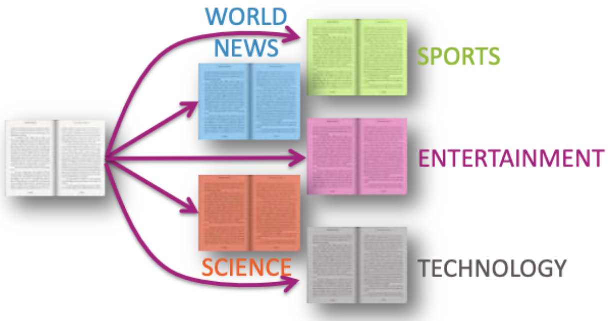
Difference

In the classification problem, you are given (x^i, y_i) - I.e. both the data point i and its true label y_i for training purposes. Example - a flower i and its label (flower type). Whereas in clustering problem, you are just given the data points, i.e. x^i. However, you still want to break up the data points into clusters - where each cluster has relatively similar data points.

Clustering for News

What if the labels are known? Given labeled training data

Can do multi-class classification methods to predict label

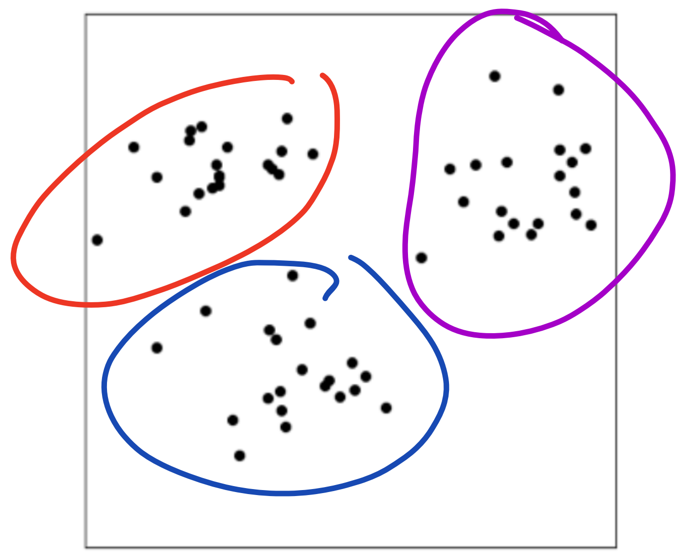


Clustering Basics

- In many real world contexts, there aren't clearly defined labels so we won't be able to do classification
- We will need to come up with methods that uncover structure from the (unlabeled) input data X.
- **Clustering** is an automatic process of trying to find related groups within the given dataset.

$Input: x_1, x_2, \ldots, x_n$

$Output: z_1, z_2, \ldots, z_n$



Clustering Basics

In their simplest form, a **cluster** is defined by

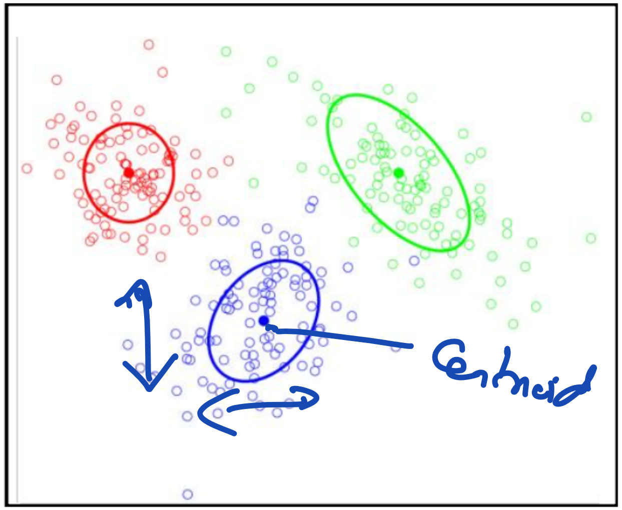
- The location of its center (**centroid**)
- Shape and size of its **spread**

Clustering is the process of finding these clusters and **assigning** each example to a particular cluster.

- x_i gets assigned $z_i \in [1, 2, \ldots, k]$
- Usually based on closest centroid

Will define some kind of score for a clustering that determines how good the assignments are

- Based on distance of assigned examples to each cluster



Centroid

Euclidean

Distance $\leftrightarrow \|x^1 - x^3\|_2$ $\quad \|x^1 - x^2\|_2$

Distance typically used

Euclidean Distance

Distance between two points, x_1, x_1 is given by:

$$\|x_1 - x_2\|_2$$

Clustering on different Data sets

Clustering is easy when distance captures the clusters

Ground Truth (not visible)

Easy to cluster

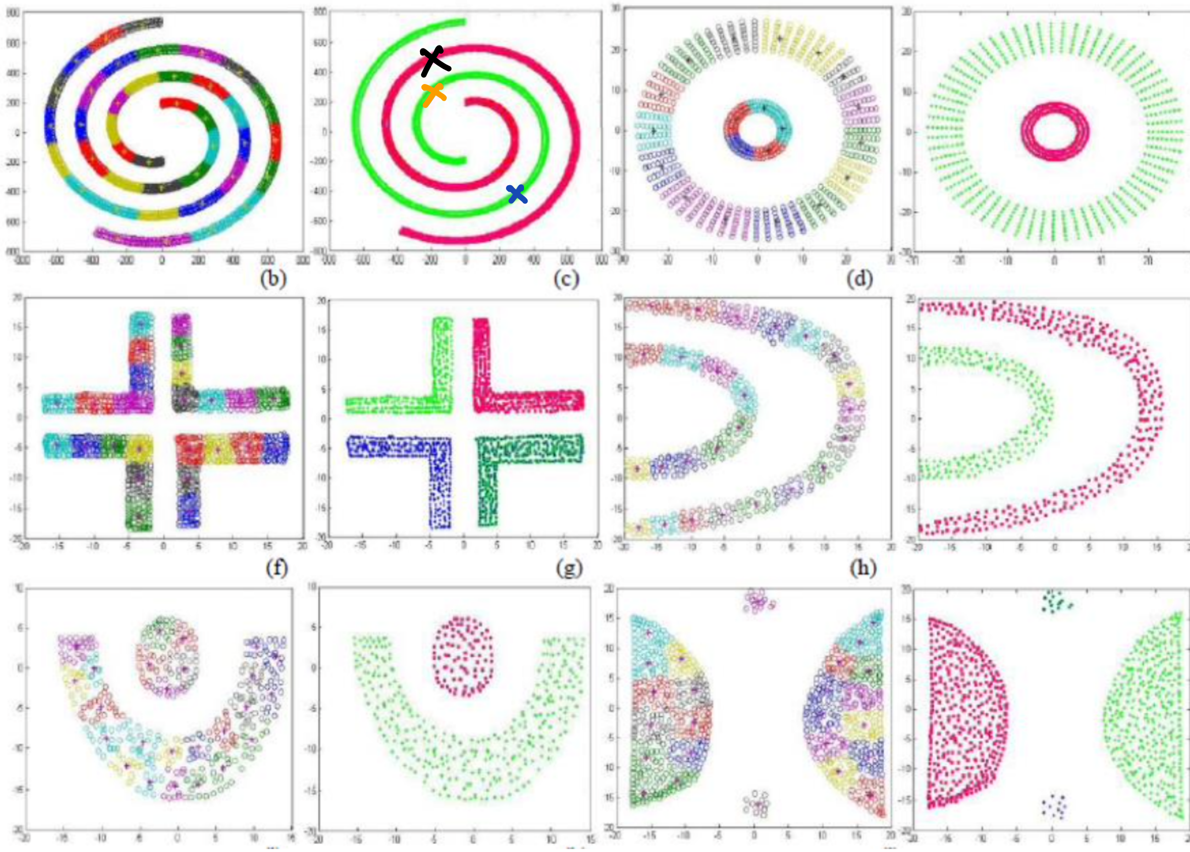
Harder to cluster

Given Data

Clustering - Hard cases

There are many clusters that are harder to learn with this setup

- Distance does not determine clusters



DONUT cluster

k-means

#Clusters

Averaging

k-means++

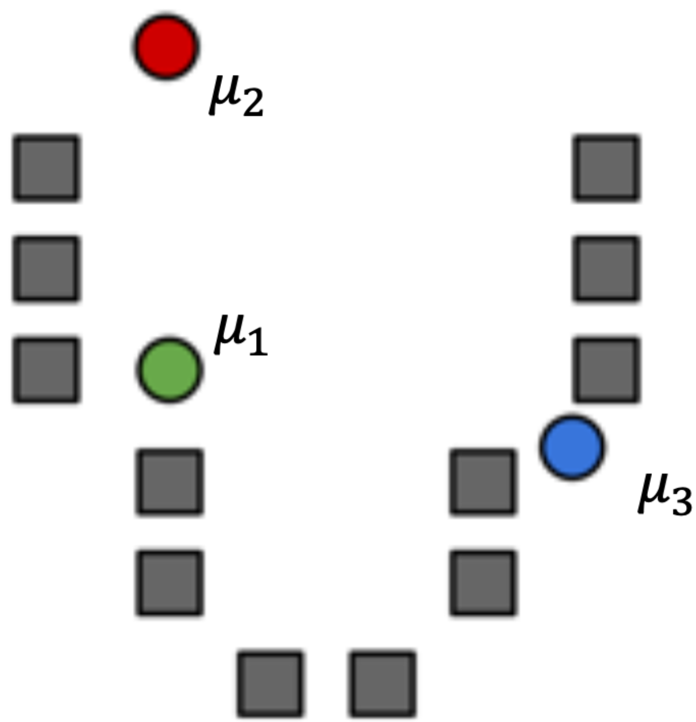
Algorithm 1 k-means algorithm

1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4: **expectation:** Assign each point to its closest centroid.
5: **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.

k-means Clustering

Start by choosing the initial cluster centroids

- A common default choice is to choose centroids at random
- Will see later that there are smarter ways of initializing



μ – Centroid

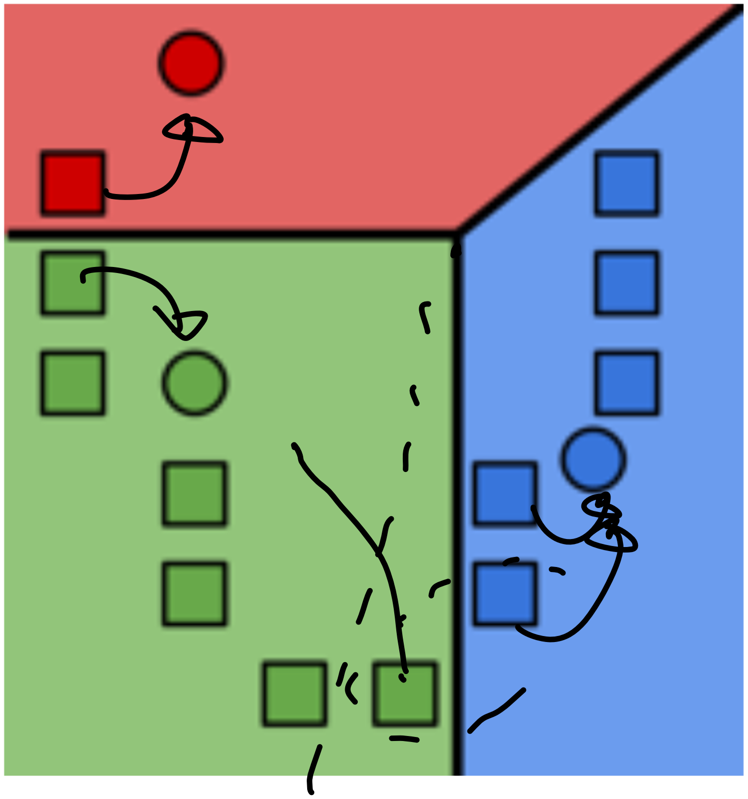
μ_1 – Centroid 1

$\vdots$

k-means Clustering

Assign each example to its closest cluster centroid

$$z_i \leftarrow \underset{j \in [k]}{\operatorname{argmin}} \left|\left|\mu_j - x_i\right|\right|^2$$

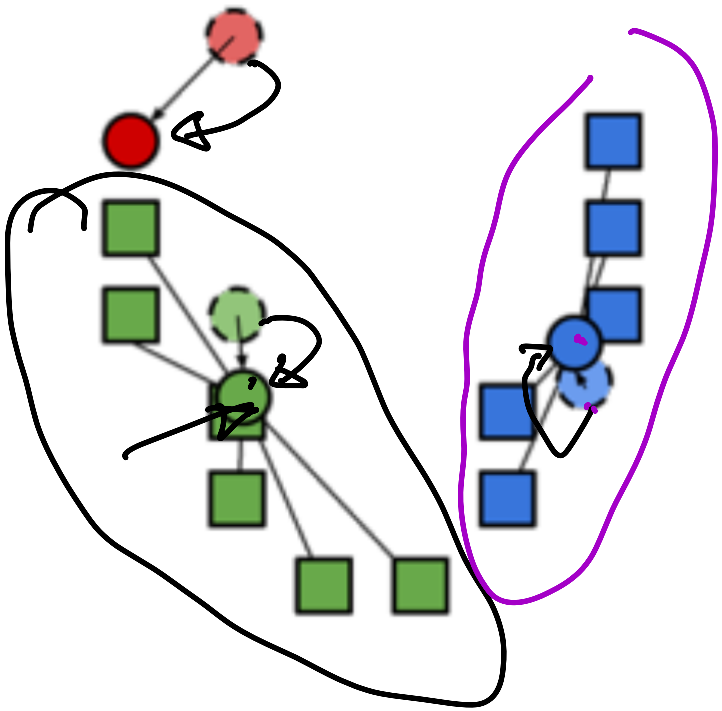


k-means Clustering

Update the centroids to be the mean of all the points assigned to that cluster.

$$\mu_j \leftarrow \frac{1}{n_j} \sum_{i:z_i=j} x_i$$

Computes center of mass for cluster!



k-means Clustering

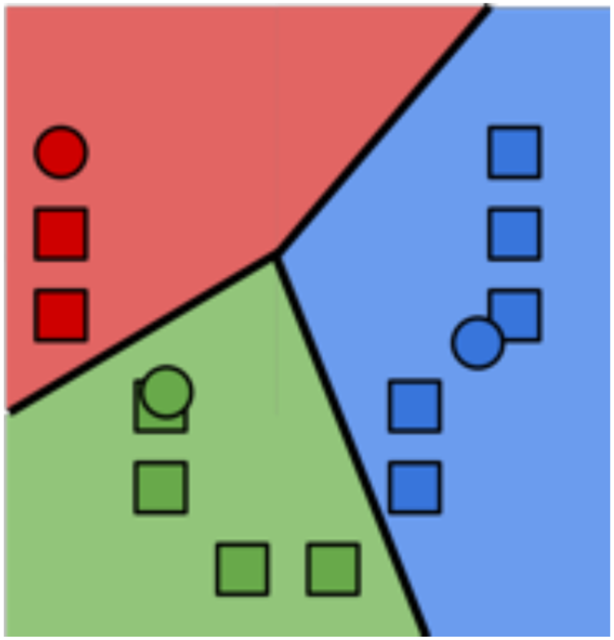
Repeat Steps 1 and 2 until convergence

Will it converge? Yes! Stop when

- Cluster assignments haven't changed
- Some number of max iterations have been passed

What will it converge to?

- Global optimum
- Local optimum
- Neither



k-means Demo

k-means Demo

k-means Demo 2

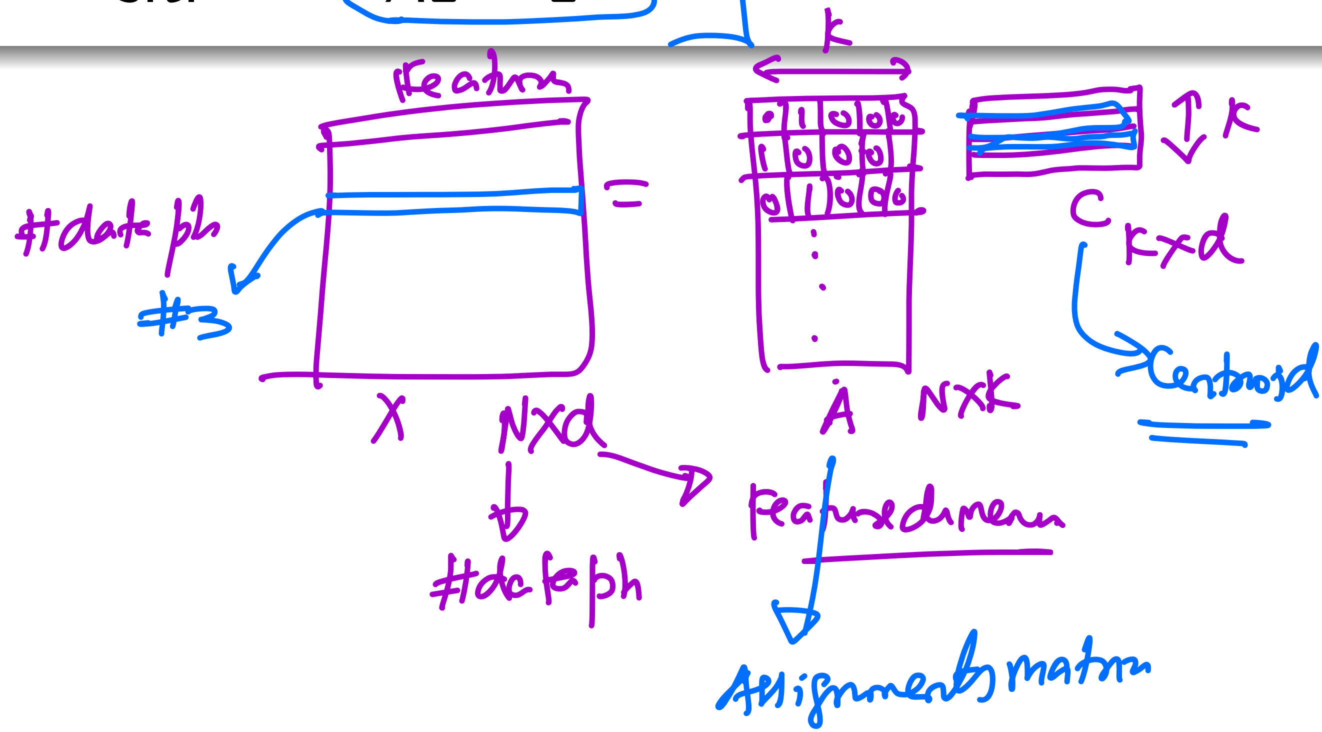
k-means Demo 2

k-means optimization

X – Data Matrix

Loss Function

$$
\begin{aligned}
\min_{C,A} \quad & \|X - AC\|_F^2 \\
\text{s.t.} \quad & A\mathbf{1} = \mathbf{1}
\end{aligned}
$$



k-means optimization

Loss Function

$$\begin{array}{ll} \min_{C,A} & \|X - AC\|_F^2 \\ \text{s.t.} & A\mathbf{1} = \mathbf{1} \end{array}$$

$X_{N \times d}$

ICE #1

For k clusters, the dimensions of the assignments matrix A and the cluster centroids matrix C are:

a. $N \times d \& k \times d$
b. $N \times k \& k \times d$
c. $N \times k \& d \times d$
d. $N \times d \& d \times k$

Deeper Understanding: k-means optimization

Alternating Optimization

$$\begin{array}{ll} \min_{A,C} & \|X - AC\|_F^2 \\ \text{s.t.} & A\mathbf{1} = \mathbf{1} \end{array}$$

$\rightarrow$ Alternatively optimize for A & C gives us K-Means Algorithm!

Deeper Understanding: k-means optimization

Alternating Optimization

$$\begin{array}{ll} \min_{A,C} & \|X - AC\|_F^2 \\ \text{s.t.} & A\mathbf{1} = \mathbf{1} \end{array}$$

1. Optimizing A

Let's say we already have the cluster centroids matrix C^j in the *jth* iteration. And we want to find the optimal assignment A^j given C^j. Then, we optimize:

$$\begin{array}{ll} \min_{A} & \|X - AC^j\|_F^2 \\ \text{s.t.} & A\mathbf{1} = \mathbf{1} \end{array}$$

$$\begin{array}{ll} \min_{A} & \sum_{i=1}^{N} (X_i^T - A_i^T C^j)^2 \\ \text{s.t.} & A\mathbf{1} = \mathbf{1} \end{array}$$

k-means optimization

2. Optimizing C

Now that we have learned an assignment's matrix A^j, can we figure out the new best centroids C^{j+1}?

$$\min_C \quad \|X - A^j C\|_F^2$$

k-means optimization

2. Optimizing C

Now that we have learned an assignment's matrix A^j, can we figure out the new best centroids C^{j+1}?

$$\min_C \quad \|X - A^j C\|_F^2$$

2. Optimizing C

Now that we have learned an assignment's matrix A^j, can we figure out the new best centroids C^{j+1}?

$$\min_C \quad \sum_{i=1}^{N} (X_i - C_{n_i})^2$$

$$\min_C \quad \sum_{p=1}^{K} \sum_{i=1\ p}^{N} (X_{m_i} - C_p)^2$$

k-means

Algorithm 1 k-means algorithm

1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4: **expectation:** Assign each point to its closest centroid.
5: **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.

Computational Complexity

Computational complexity of distance

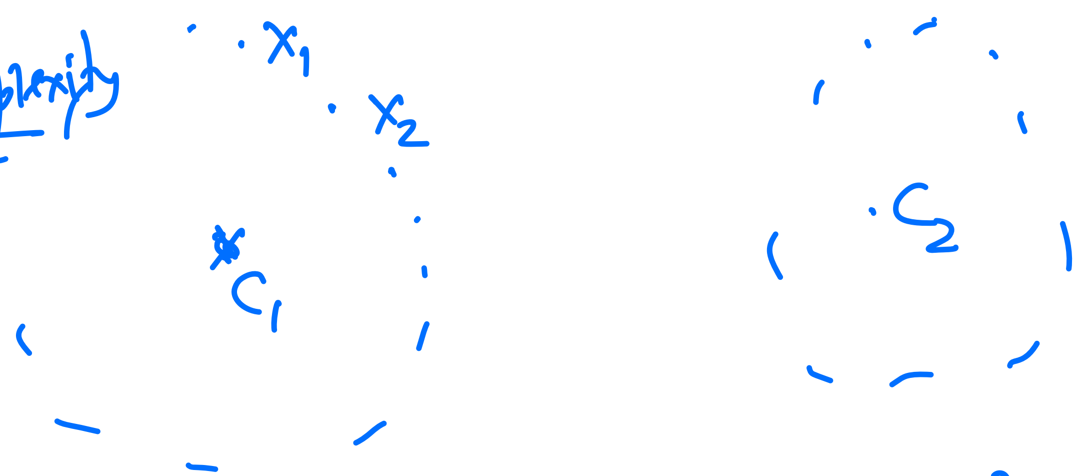
Let x_1 be a data point and c_1 be a cluster centroid. Note both are in $\mathcal{R}^d$. What's the computational complexity of evaluating $\|x_1 - c_1\|_2^2$ How about $\|x_1 - c_1\|_1$?

$\approx o(d)$

$= |x_{11} - c_{11}| + |x_{12} - c_{12}| + --- + |x_{1d} - c_{1d}|$

$\approx o(d)$

Comp. complexity



$x_1 \in \mathbb{R}^d$ $\quad$ $c_1 \in \mathbb{R}^d$

$\|x_1 - c_1\|_2^2 = \underbrace{(x_{11} - c_{11})^2}_{o(1)} + \overset{o(1)}{(x_{12} - c_{12})^2} + --- + \overset{o(1)}{(x_{1d} - c_{1d})^2}$

$= o(d)$ Comp. complexity

Computational Complexity

Computational complexity of distance

Let x_1 be a data point and c_1 be a cluster centroid. Note both are in $\mathcal{R}^d$. What's the computational complexity of evaluating $\|x_1 - c_1\|_2^2$ How about $\|x_1 - c_1\|_1$?

ICE #2 (3 mins)

Let $X \in \mathcal{R}^{N \times d}$. Assume we want to find k clusters through k-means. What is the average computational complexity of computing C and A through k-means in terms of N, d, k?

- a $O(Nd) \& O(kdN)$
- b $O(Nd) \& O(dN)$
- c $O(Nd/k) \& O(dN)$
- d $O(Nd/k) O(kdN)$

N

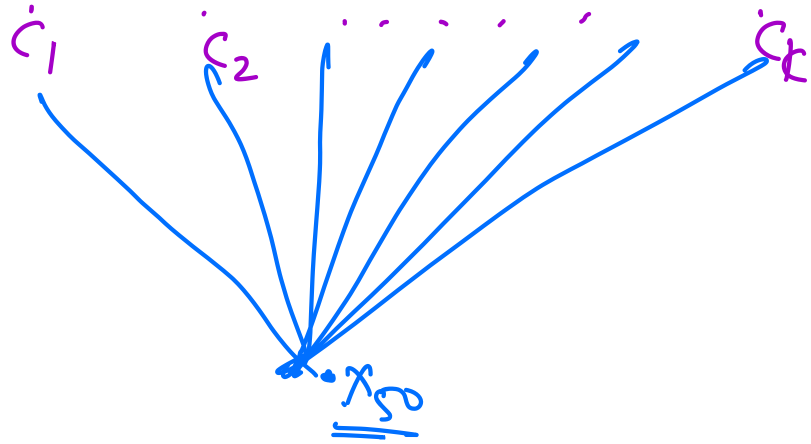
$\in \mathbb{R}^d$ $\in \mathbb{R}^d$

$\dfrac{x_1 + x_2 + \cdots x_{N/k}}{(N/k)}$

$C: O(\frac{Nd}{k}) \times k$

(N/k)

Centroids:-



$$d(c_2, x_{50}) = O(d)$$

1) Cost of assigning x_{50} to any of the centroids
$= O(d) \times k = O(dk)$

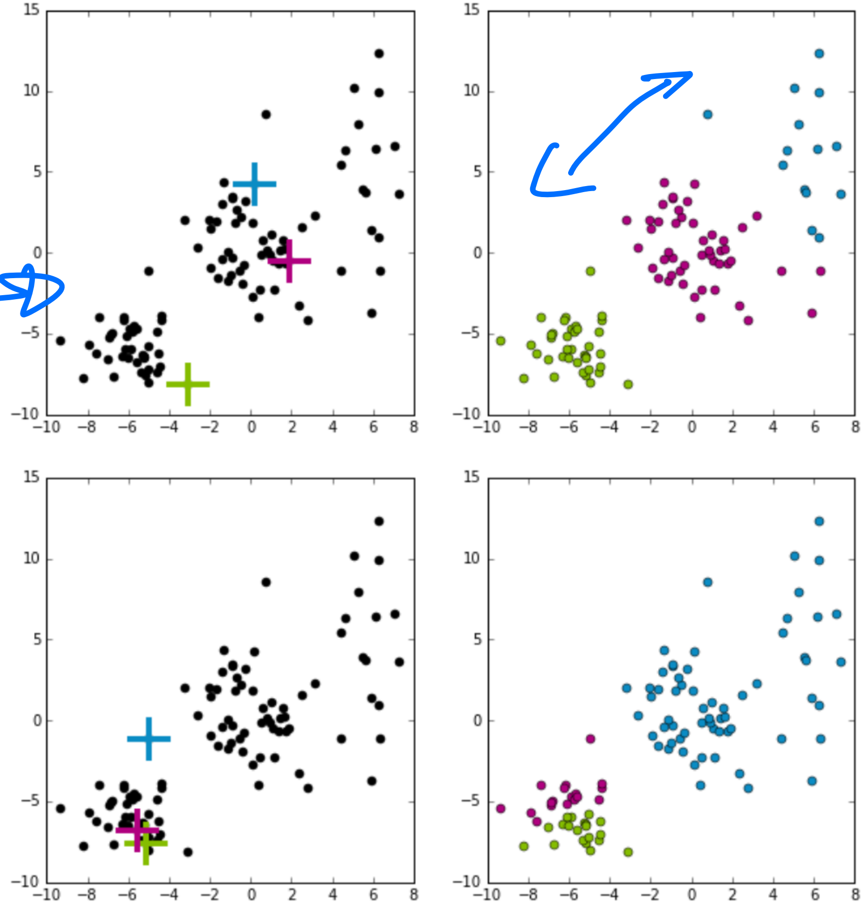
2) There are n data points $= O(ndk)$!

Comp. complexity of C step:- $O(nd)$
—"— of A step:- $O(ndk)$

k-means Local Optima

What does it mean for something to converge to a local optima?

- Initial settings will greatly impact results!



Improving kMeans - kMeans++

Making sure the initialized centroids are "good" is critical to finding quality local optima. Our purely random approach was wasteful since it's very possible that initial centroids start close together.

Idea: Try to select a set of points farther away from each other.

k-means++ does a slightly smarter random initialization

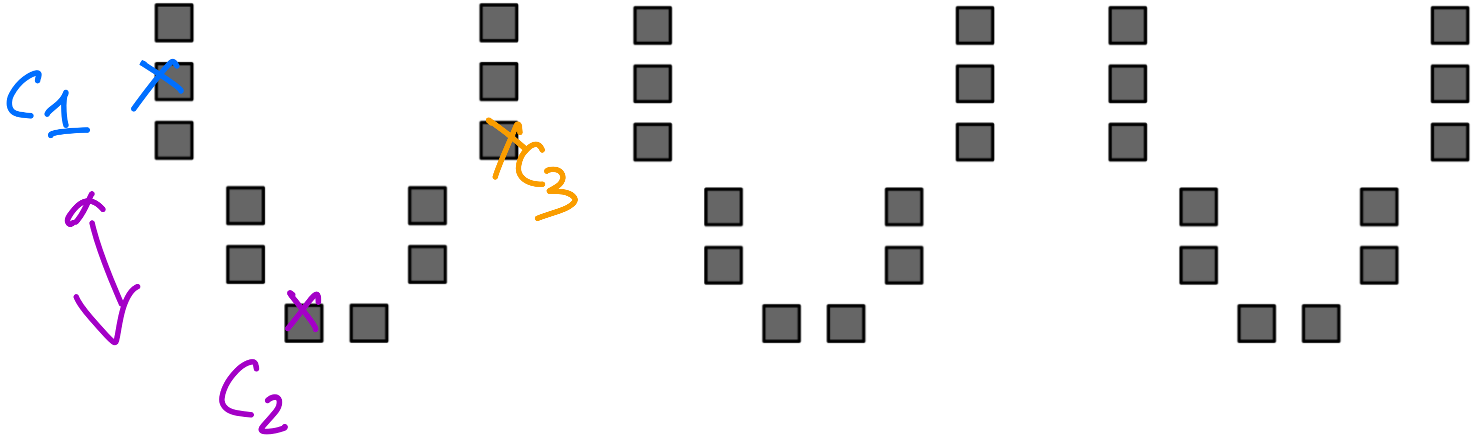
1. Choose first cluster μ_1 from the data uniformly at random
2. For the current set of centroids (starting with just μ_1), compute the distance between each datapoint and its closest centroid
3. Choose a new centroid from the remaining data points with probability of x_i being chosen proportional to $d(x_i)^2$
4. Repeat 2 and 3 until we have selected k centroids

Improving kMeans - kMeans++

Start by picking a point at random

Then pick points proportional to their distances to their centroids

This tries to maximize the spread of the centroids!

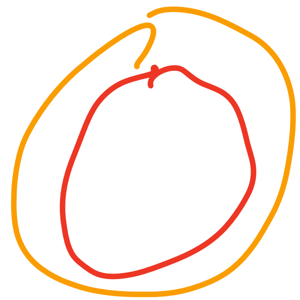


k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.

k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.
2. Clusters make sense if cluster division makes sense based on Euclidean distance.



k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.
2. Clusters make sense if cluster division makes sense based on Euclidean distance.
3. k-means has a computational complexity of $O(Nd)$ for C step and $O(Nkd)$ for A step

k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.
2. Clusters make sense if cluster division makes sense based on Euclidean distance.
3. k-means has a computational complexity of $O(Nd)$ for C step and $O(Nkd)$ for A step
4. Counter examples of clusters that may not be clustered well by k-means. Can use other methods for this.

k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.
2. Clusters make sense if cluster division makes sense based on Euclidean distance.
3. k-means has a computational complexity of $O(Nd)$ for C step and $O(Nkd)$ for A step
4. Counter examples of clusters that may not be clustered well by k-means. Can use other methods for this.
5. k-means++ - Improvement on k-means and yields better quality clusters

k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.
2. Clusters make sense if cluster division makes sense based on Euclidean distance.
3. k-means has a computational complexity of $O(Nd)$ for C step and $O(Nkd)$ for A step
4. Counter examples of clusters that may not be clustered well by k-means. Can use other methods for this.
5. k-means++ - Improvement on k-means and yields better quality clusters
6. Both k-means and k-means++ suffer from the clusters being spherical in nature. What if the true cluster shapes look different? Next lecture: Kernel k-means and Agglomerative clustering!

k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.
2. Clusters make sense if cluster division makes sense based on Euclidean distance.
3. k-means has a computational complexity of $O(Nd)$ for C step and $O(Nkd)$ for A step
4. Counter examples of clusters that may not be clustered well by k-means. Can use other methods for this.
5. k-means++ - Improvement on k-means and yields better quality clusters
6. Both k-means and k-means++ suffer from the clusters being spherical in nature. What if the true cluster shapes look different? Next lecture: Kernel k-means and Agglomerative clustering!
7. Clustering can help with cold-start problem. E.g. recommending new products!

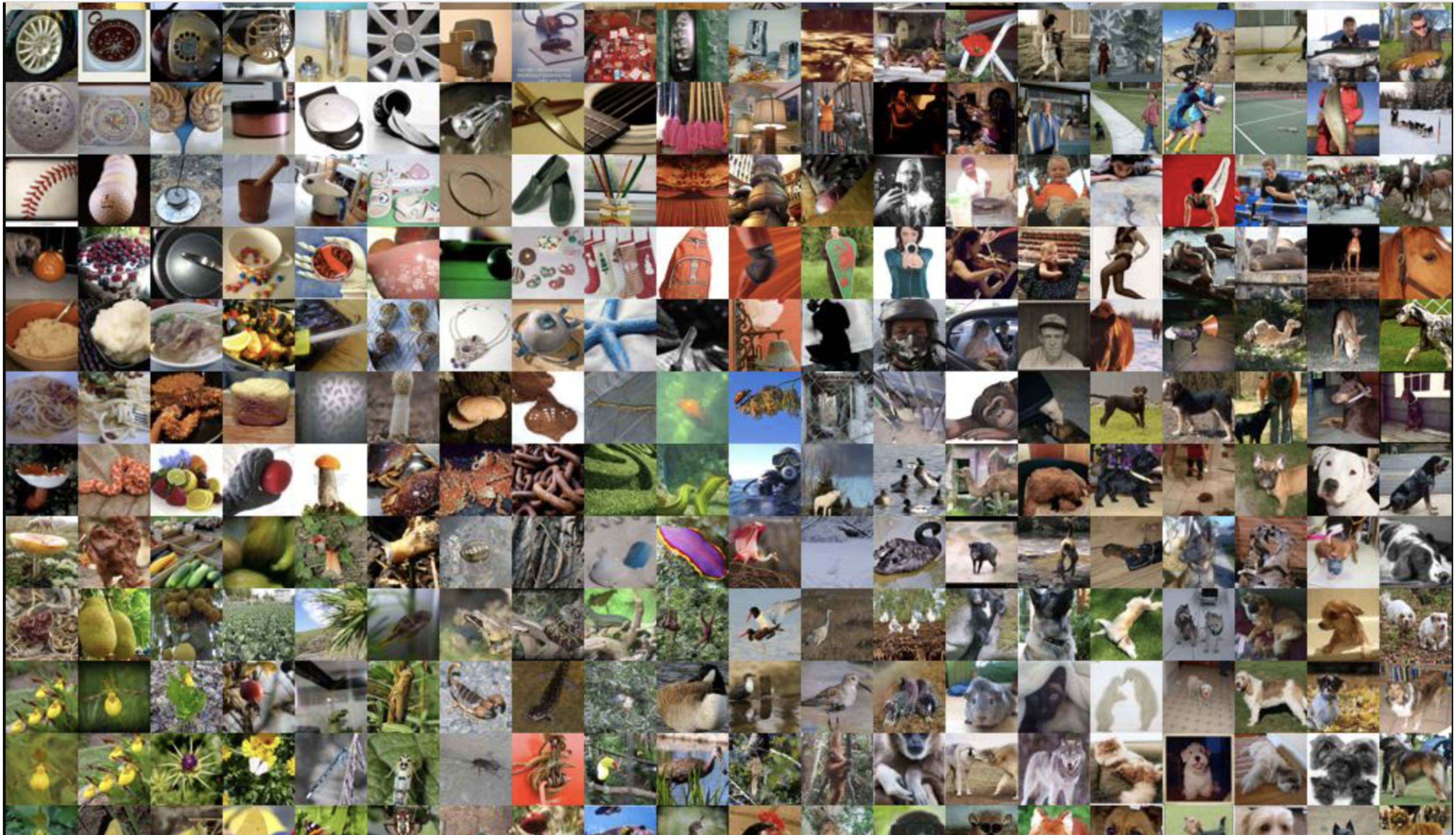
Clustering in 2 dimensions - tSNE!

Clustering for Data Visualization

Images

Let's say we had 1000 images and wanted to "cluster" them onto a super-grid of images so that similar images are closely placed on the super-grid and dis-similar are placed further away. k-means clustering will only get us half-way there!

Data Visualization: Stochastic Neighborhood Embeddings (SNE)!



SNE

High-level Idea

Find an embedding of images in 2 dimensions that put similar images close to each other and dis-similar images further away from each other.

SNE

High-level Idea

Find an embedding of images in 2 dimensions that put similar images close to each other and dis-similar images further away from each other.

Soft clustering

We don't have a K here. But if you look at any neighborhood of the super grid of images - They will look similar! We can call this soft-clustering.

SNE

Similarity measure through Probabilities

Let $x_1, x_2, \ldots$ represent features of the data in their original dimensions (e.g. images).

$$p_{j|i} = \frac{e^{-\|x_i - x_j\|_2^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|x_i - x_k\|_2^2 / 2\sigma_i^2}}$$

SNE

Similarity measure through Probabilities

Let $x_1, x_2, \ldots$ represent features of the data in their original dimensions (e.g. images).

$$p_{j|i} = \frac{e^{-\|x_i - x_j\|_2^2/2\sigma_i^2}}{\sum_{k \neq i} e^{-\|x_i - x_k\|_2^2/2\sigma_i^2}}$$

Low-dimensional embedding Probabilities

Let $y_1, y_2, \ldots$ represent features of the data in lower (embedded) dimensions (e.g. 2 dimensions).

$$q_{j|i} = \frac{e^{-\|y_i - y_j\|_2^2/2\sigma_i^2}}{\sum_{k \neq i} e^{-\|y_i - y_k\|_2^2/2\sigma_i^2}}$$

Estimating low-dimensional embeddings in SNE

A similarity measure for Probabilities - KL Divergence

$$KL(p||q) = \sum_{i=1}^{d} p_i \log \frac{p_i}{q_i}$$

Estimating low-dimensional embeddings in SNE

A similarity measure for Probabilities - KL Divergence

$$KL(p||q) = \sum_{i=1}^{d} p_i \log \frac{p_i}{q_i}$$

Loss function

$$L(y_1, y_2, \ldots, y_N) = \sum_{i=1}^{N} KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

Gradient and GD

Gradient

$$\frac{\partial L}{\partial y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

Gradient and GD

Gradient

$$\frac{\partial L}{\partial y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

GD

$$y^{t+1} = y^t - \eta \frac{\partial L}{\partial y}(y^t)$$

Image Chain

ICE #1 (3 mins break out)

Let's say you want to create a video that has 1000 images (e.g. the one we looked at earlier) in a sequence so that the images in the video transforms smoothly from one to the next. How would you go about doing this if you learned a tSNE representation for the images?

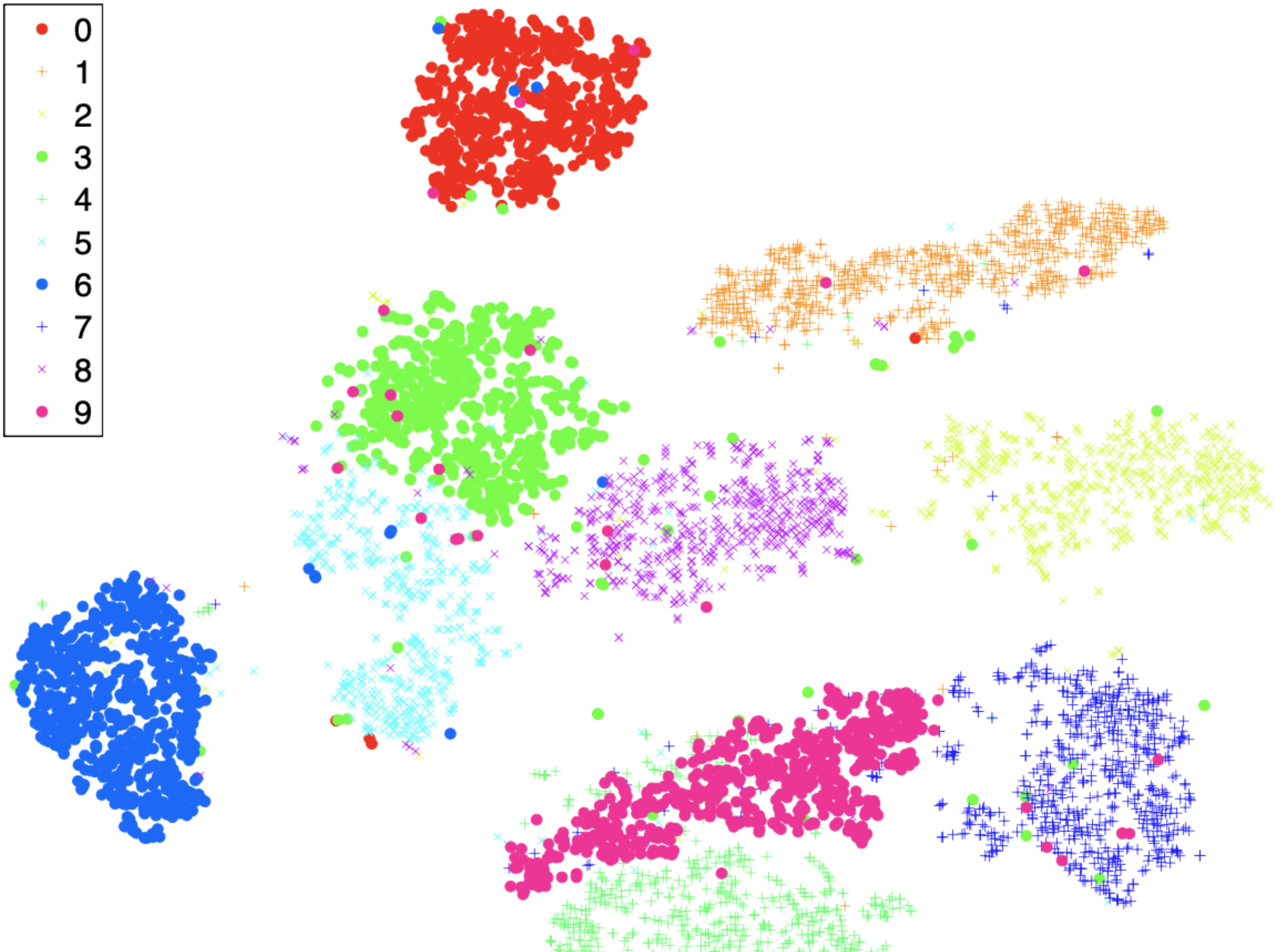
How do we create this grid?

tSNE Notebook Example

Notebook

MNIST digits data set

MNIST tSNE embeddings



Word visualization based on word2vec