Distance in High Dimensions and Clustering

Chris Cornwell

April 29, 2025

Outline

Outline

Clustering uses Distance

The aim of clustering is to group the training data $\mathcal{S}$ into **clusters** $C_1, C_2, \ldots, C_k$ so that every point is in some cluster C_i (i.e., $\mathcal{S} = C_1 \cup C_2 \cup \ldots \cup C_k$) and clusters are disjoint.[1] The notion is for points in the same cluster to be "similar" in some sense.

While most types of ML algorithms are affected in some way by how densely packed points in $\mathcal{S}$ are, clustering algorithms typically use distance (from a point to the nearest points in $\mathcal{S}$) in order to decide how similar points are. This makes it especially important to consider a phenomenon called the **curse of dimensionality**.

[1] These conditions are the common ones. However, sometimes one may wish to exclude an *outlier* from being in a cluster, and sometimes clusters are "fuzzy," meaning that points have a probability for being in each cluster.

What is the Curse of Dimensionality?

Not universally, clearly defined what falls under the umbrella of the curse of dimensionality and what does not.

- Strict interpretation: The amount of training data used needs to increase exponentially in the number of features, or independent variables. (At least, if the number of samples needed to see how position/value of one feature might affect y labeling is roughly constant over the features.)
- Broader interpretation: With large number of features (so, large d, where $\mathbf{x}_i \in \mathbb{R}^d$), our intuition for the way that the distance between points relates to properties we care about will break down. (Distance in high dimensions is *weird*.)

Spheres in $\mathbb{R}^d$

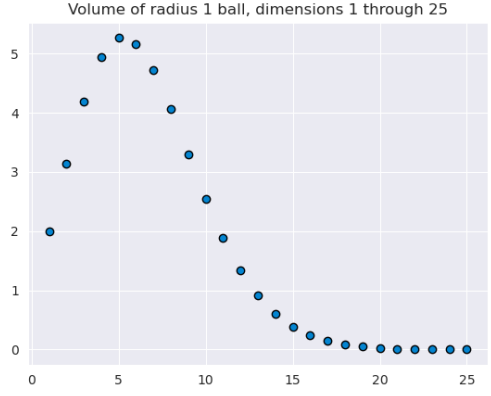
Often, want to work with those points that are within some given distance r from a fixed point. These are points in a d-dimensional "ball" (inside a d-dimensional sphere):

$$B_r(\mathbf{x}_0) = \{\mathbf{x} \in \mathbb{R}^d \mid |\mathbf{x} - \mathbf{x}_0| \leq r\}.$$

(Here, *distance* between $\mathbf{x}$ and $\mathbf{x}_0$ is $d(\mathbf{x}, \mathbf{x}_0) = |\mathbf{x} - \mathbf{x}_0|$, the usual Euclidean norm.)

The volume of $B_r(\vec{\mathbf{0}})$: $\dfrac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)} r^d$.

Γ is Euler's gamma function; for even d, you have $\Gamma(\frac{d}{2}+1) = (\frac{d}{2})!$. When d is odd, it's roughly like that: $(\frac{d}{2})(\frac{d}{2}-1)\ldots(\frac{1}{2})\pi^{1/2}$.



Spheres in $\mathbb{R}^d$

So, for any fixed radius $R > 0$, the volume of the d-dimensional $B_R(\vec{\mathbf{0}})$ approaches 0 as $d \to \infty$.
Additionally, *where* the space inside of $B_R(\vec{\mathbf{0}})$ is distributed changes as dimension increases.
Choose ε with $0 < \varepsilon < 1$. What proportion of points in $B_R(\vec{\mathbf{0}})$ are at least ε away from the boundary sphere. That is, how large is $B_{R-\varepsilon}(\vec{\mathbf{0}})$ in comparison to $B_R(\vec{\mathbf{0}})$? In dimension $d = 2$, with $R = 1$:

$$\frac{\operatorname{Vol} B_{1-\varepsilon}}{\operatorname{Vol} B_1} = \frac{\pi(1-\varepsilon)^2}{\pi} = (1-\varepsilon)^2.$$

For example, if $\varepsilon = 0.05$ then this is a little more than 0.9.
However, generally,

$$\frac{\operatorname{Vol} B_{R-\varepsilon}}{\operatorname{Vol} B_R} = \frac{\pi^{d/2}(R-\varepsilon)^d}{\Gamma(d/2+1)} \frac{\Gamma(d/2+1)}{\pi^{d/2}R^d} = \left(\frac{R-\varepsilon}{R}\right)^d = \left(1-\frac{\varepsilon}{R}\right)^d.$$

Since $1 - \frac{\varepsilon}{R} < 1$, this approaches 0 as $d \to \infty$. Returning to $\varepsilon = 0.05$ and $R = 1$, the ratio is less than 0.05 if $d \geq 59$ – so, more than 95% of the volume of $B_R(\vec{\mathbf{0}})$ is contained in a shell near the boundary (within 0.05 of the boundary sphere).

Most points near the boundary

What is a consequence? If you pick a point and are looking for points in the data set within a given distance of it, then the higher the dimension, the more likely nearly all of these points will be close to the same distance from the chosen point. So, for example, which *one* is closest is more subject to random chance (from a small amount of noise, say).
A computational verification: points were selected with random coordinates (i.i.d., with a mean-zero normal distribution). Then, distances between all pairs of these points were calculated and plotted in a histogram.

Another example of high dimensional weirdness

In $\mathbb{R}^2$, say that we consider the five depicted circles within a 4×4 square. The four "corner" circles are tangent to (two) edges of the square and are tangent to each other, with each having radius 1. The "center" circle shares its center with the square and is tangent to all four of the corner circles. The radius of the center circle is $\sqrt{2} - 1$. Hence, it is smaller than each of the corner circles (as is visibly apparent). Generalize this construction to a $4 \times 4 \times \ldots \times 4$ (hyper)cube in $\mathbb{R}^d$. In general, there are 2^d corner spheres, each having radius 1, and there is one center sphere, with the same center as the hypercube and which is tangent to all of the corner spheres.

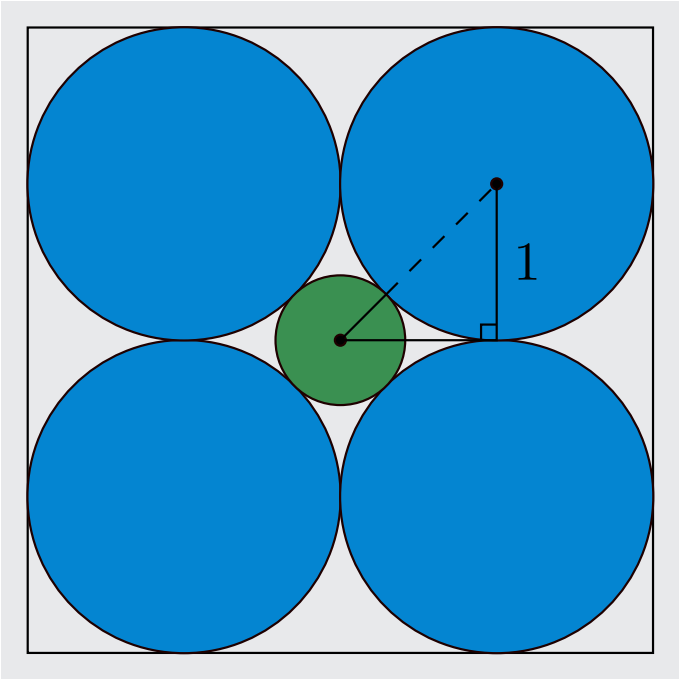


Figure: Filling 2D square with corner circles and center circle

Another example of high dimensional weirdness

As a consequence of the distance formula in $\mathbb{R}^d$, the radius of the center sphere is necessarily $\sqrt{d} - 1$. Note the consequences:

- When $d = 4$, then the center sphere is the same size as the corner spheres, since $\sqrt{4} - 1 = 1$.
- The center sphere is larger than the corner spheres when $d \geq 5$, and once $d = 9$ we have that the radius of the center sphere is 2. So, that center sphere intersects the boundary of the cube.
- For $d \geq 10$, the center sphere contains points that are *outside of* the hypercube. (Despite still being tangent to all 2^d corner spheres, which "surround" it and are *entirely contained* within the hypercube. Orthogonal projection to any pair of coordinates still looks exactly like the $d = 2$ case.)

Outline

What is clustering?

Intuitively, want to group together data points into subsets, i.e., *clusters*, so that *similar* points are in the same cluster and dissimilar points are in different clusters. These data do not have labels.

- Presuming the numerical coordinates of the data have meaning, "similar" points could be those that are sufficiently close.

Goal: Given sample data $\mathcal{S} = \{\mathbf{x}_i\}_{i=1}^n$, want to determine clusters $C_1, C_2, \ldots, C_k$ (for some k), so that $C_1 \cup C_2 \cup \ldots \cup C_K = \mathcal{S}$ and $C_i \cap C_j = \varnothing$ when $i \neq j$.

An inherent difficulty: the clustering problem is ill-defined.

- For example, you might have a sequence of data points $\mathbf{x}_1, \ldots, \mathbf{x}_m$ such that $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ are similar (close enough to be called similar) for each $1 \leq i \leq m-1$, however $\mathbf{x}_1$ and $\mathbf{x}_m$ might be very dissimilar. While $\mathbf{x}_i$ and x_{i+1} should be in the same cluster, this causes the problem of putting $\mathbf{x}_1$ and $\mathbf{x}_m$ ending up in the same cluster.

Applications of clustering

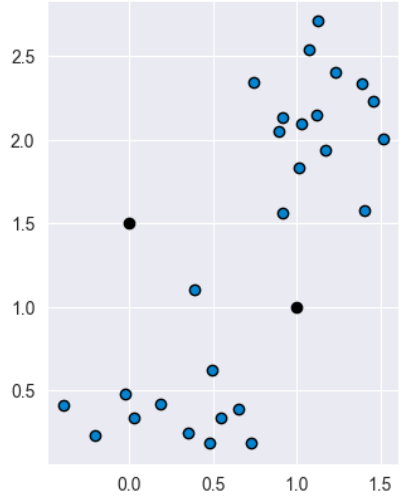
Despite its ill-defined nature, clustering is needed (sometimes in an initial processing step) for many data analysis tasks.
Some examples include:

- Market segmentation: clustering expected customers based on characteristics, either for targeted marketing or to inform product design.
- Gene expression clustering: computational biologists will cluster genes based on how their expression in different experiments.
- Astronomical data analysis: clustering stars or galaxies based on their proximity.
- Social network analysis: for example, identifying “communities” (clusters of users) based on interactions on social media.

k-means

Let $\mathcal{S} = \{\mathbf{x}_i\}_{i=1}^n$, with each $\mathbf{x}_i \in \mathbb{R}^d$ for some dimension d.

- The k-means algorithm is a common approach for clustering $\mathcal{S}$.
- You choose k, and your goal is to find k clusters, $C_1, C_2, \ldots, C_k$ so that $\mathcal{S} = C_1 \cup \ldots C_k$, where we want the sum $\sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} |\mathbf{x}_i - \mu_j|^2$ to be minimized.
- Here, μ_j is the centroid of C_j, meaning that μ_j is the point in $\mathbb{R}^d$ that minimizes, over $\mu \in \mathbb{R}^d$, the sum $\sum_{\mathbf{x} \in C_j} |\mathbf{x} - \mu|^2$.
  - Note that, in fact, $\mu_j = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$.



k-means Procedure

The procedure to carry out k-means clustering is the following. First, randomly initialize k centroids. Then:

1. For each $i = 1, 2, \ldots, n$, determine $1 \leq j(i) \leq k$, which is the index such that $\mu_{j(i)}$ is the closest centroid to $\mathbf{x}_i$. A point $\mathbf{x}_i$ is in C_j when $j = j(i)$.
2. Update $\mu_1, \mu_2, \ldots, \mu_k$ so that, for $1 \leq j \leq k$, the vector μ_j is the centroid of the points in C_j.
3. Iterate steps 1 and 2 until there is no $\mathbf{x}_i$ that "changes its cluster", i.e., until the assignment $i \mapsto j(i)$ that is made in 1 is the same as it was in the previous iteration.

At each iteration, if some point has changed the cluster it is in, then the value of $\sum_{j=1}^{k} \sum_{\mathbf{x}_i \in C_j} |\mathbf{x}_i - \mu_j|^2 = \sum_{i=1}^{n} |\mathbf{x}_i - \mu_{j(i)}|^2$ decreases. Hence, the algorithm terminates because, as there are finitely many points in $\mathcal{S}$, there are only a finite number of possibilities for the list $\mu_1, \mu_2, \ldots, \mu_k$.

Visual of k-means in $\mathbb{R}^2$

We show a visualization of the
procedure for k-means on data in $\mathbb{R}^2$, with $k = 2$.

Initially, we choose
two arbitrary centroids. These are drawn in black.
Then, the first step is to
go through all of our data (drawn in) and determine
which of the two centroids is closest to each point.
This assigns each point to one of the two clusters.

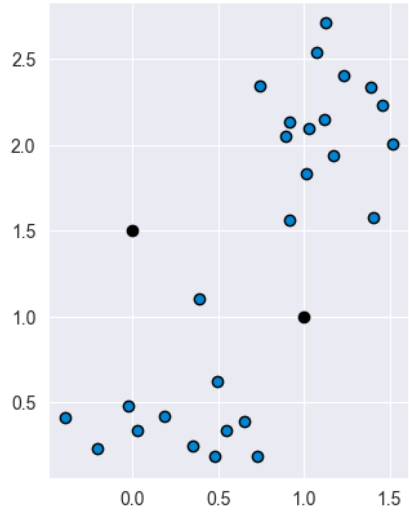


Figure: Initial centroids

Visual of k-means in $\mathbb{R}^2$

We show a visualization of the
procedure for k-means on data in $\mathbb{R}^2$, with $k = 2$.

Here, the two clusters are shown.
The points in one are drawn in light
blue and the points in the other are drawn in red.
The centroid of
each cluster is displayed, outlined in the color of its
cluster. The next step is to recompute the centroids.

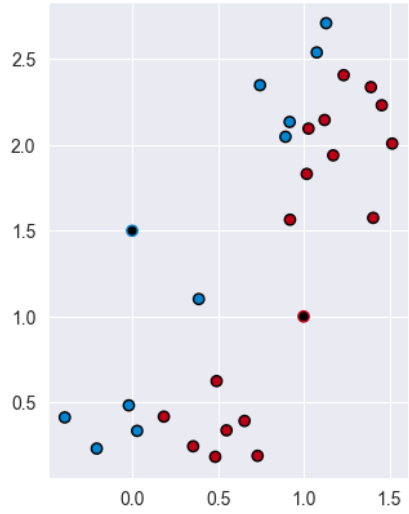


Figure: Assigned clusters

Visual of k-means in $\mathbb{R}^2$

We show a visualization of the
procedure for k-means on data in $\mathbb{R}^2$, with $k = 2$.

The two newly computed centroids are shown.
In addition, we reassign
points based on the (new) centroid that is closest.
We notice that 4 of points have been removed from the blue cluster into the red cluster; also, 3 points have been removed from red cluster into the blue.

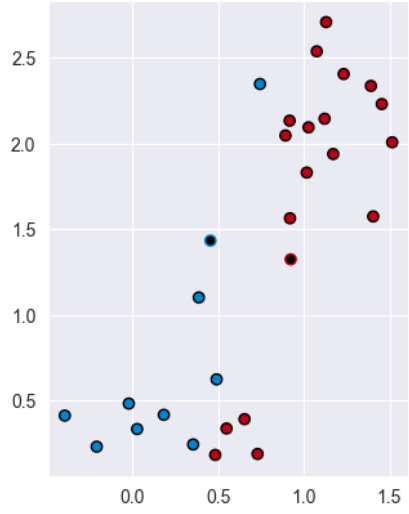


Figure: Updated clusters

Visual of k-means in $\mathbb{R}^2$

We show a visualization of the
procedure for k-means on data in $\mathbb{R}^2$, with $k = 2$.

Again, we compute new centroids and
reassign points based on the centroid that is closest.
In this step,
there is 1 point that was previously in the blue cluster
which changes into red. And 4 points that were in the
red cluster have been moved into the blue cluster.

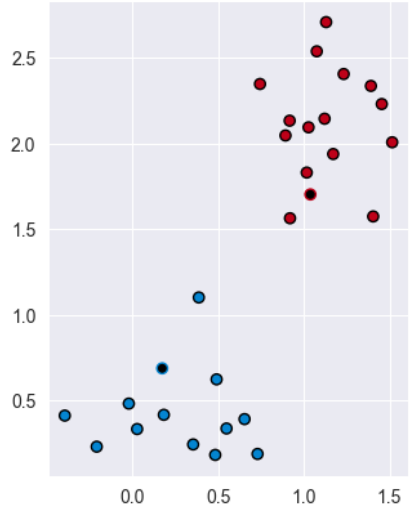


Figure: Updated clusters

Visual of k-means in $\mathbb{R}^2$

We show a visualization of the
procedure for k-means on data in $\mathbb{R}^2$, with $k = 2$.

We compute new centroids again.
This time there
is no change in the way that points are assigned
to clusters when finding the closest centroid.
Since there is no change to the clustering assignment,
the procedure terminates with these centroids.

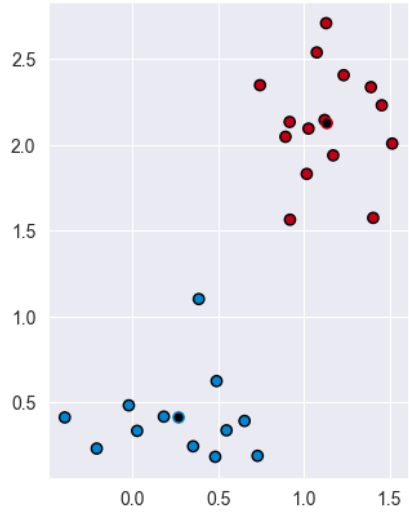


Figure: Updated centroids

k-means depends on Initial Centroids

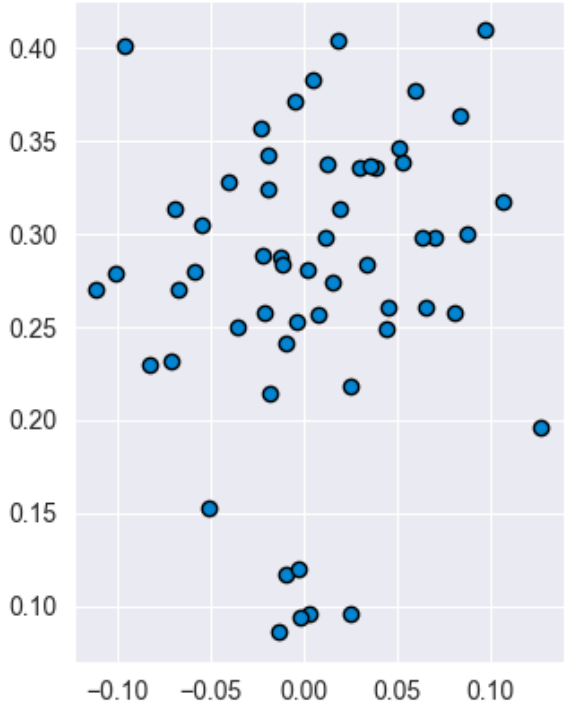


Figure: A data set to cluster

k-means depends on Initial Centroids

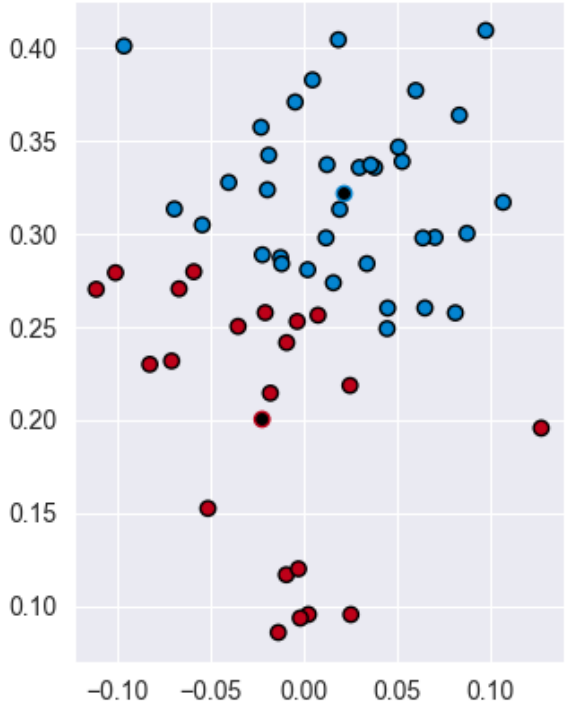


Figure: A data set to cluster

With initial centroids of $(0, 0.5)$ and $(0, 0.25)$.

k-means depends on Initial Centroids

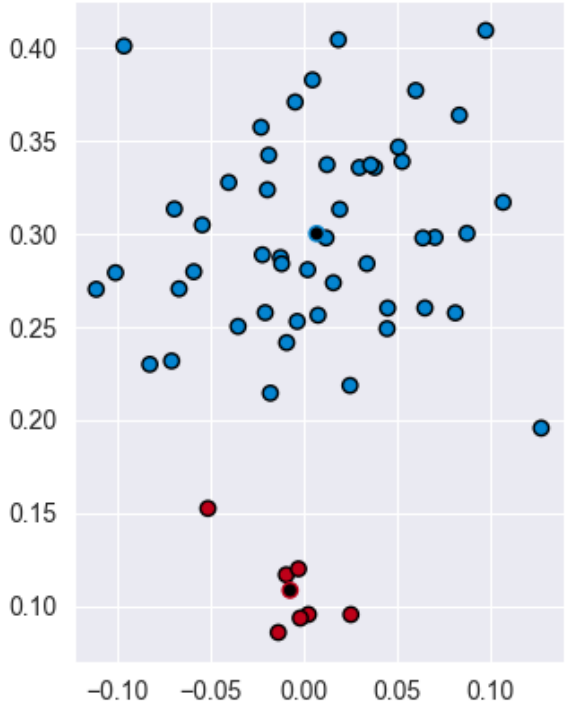


Figure: A data set to cluster

With initial centroids of $(0, 0.26)$ and $(0, 0.1)$.

Density-Based Spatial Clustering for Applications with Noise

DBSCAN clustering does not require choosing a number of clusters at the start.

- Near a given point in $\mathcal{S}$, whether it gets included in a nearby cluster is based on the *density* of points (in $\mathcal{S}$) near that given point.
- The procedure builds a directed graph (network of vertices and edges); vertices are points in $\mathcal{S}$; which edges are included is based on a choice of parameters, and takes into account the nearby density of points.
- Connectedness through edges of the graph determines when points are in the same cluster.

An example of a graph built in the DBSCAN procedure is drawn below. Points are colored according to their cluster.