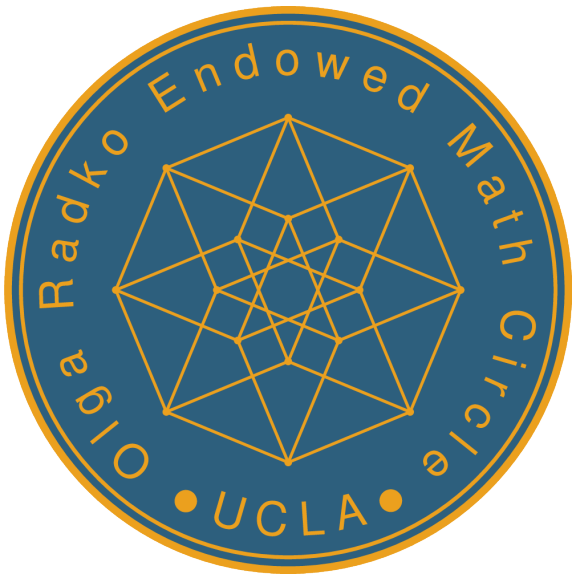
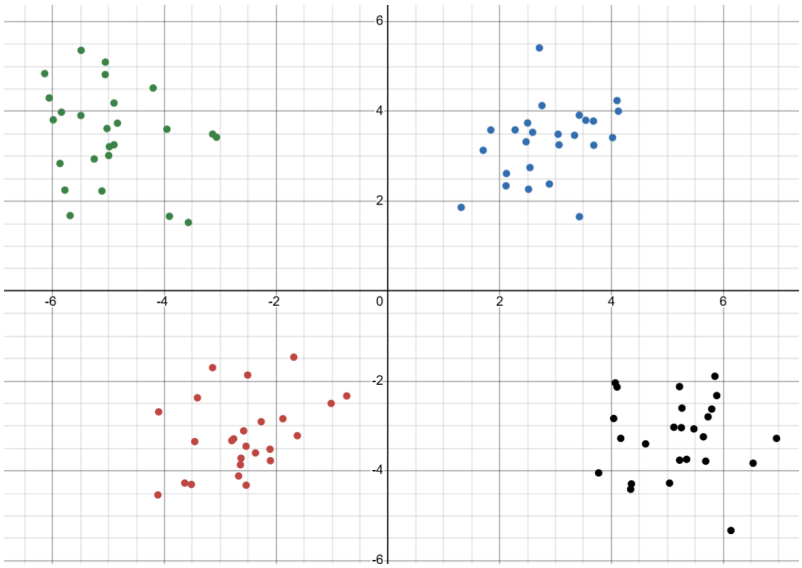


Data Clustering

Oleg Gleizer, Kiana Avestimehr, Brandon Holland

Clustering

Say you have data (meaningless for now) on axes like so:



You probably already have an intuitive idea that this data is clustered, but this packet will be devoted to understanding:

1. What does it mean, mathematically, for data to be clustered?
2. If we know where the clusters in a dataset are, can we find what cluster a new data point should belong to?
3. How can we, given a completely arbitrary dataset (possibly in 3, 4, or N dimensions), determine clusters?

Once we've answered these questions, we will try to combine those applications to approach a real world problem. Namely, we will take **a dataset** of the vocal qualities of patients suspected to have **Parkinson's Disease**, cluster these data points as either having Parkinson's or not, and use this prediction machine to **determine if a new patient has the disease or not**.

Finally, we'll walk through some of the **mathematical analysis** of these algorithms to highlight precisely how mathematicians came to understand them and why we know they work. This sort of analysis will introduce you to a budding multidisciplinary field in computer science and mathematics.

It is important to remember as you work through this packet just how applicable this sort of work is. **Try to imagine problems in your own life you could solve like this.** There are undoubtedly many!

Defining Clustering

As it tends to be in applied math, the *way* you define your problem is just as important as how you solve it. Therefore, let's think about how we can define clusters.

In our minds, "**clusters**" are defined as **sets of similar points**. Thus, in order to define our clusters, we only need to understand what "similar" means. There are actually many different ways to approach this, most of them geometric in nature.

With this in mind, we will adopt some **basic linear algebra notation** for our data points, treating a point with dimension N as a vector $v \in \mathbb{R}^N$ (just saying v is a vector N long with real-valued entries).

Problem 1

What ways can you think of to define "similarity" of data? What parameters would you use? (Be creative — there are millions of ways data can look, so there are even more answers to how we can understand it!)

The most natural definition is one of **distance**. To denote this distance, we will use a common notation in linear algebra called the *norm*.

Definition: For a vector v in N dimensions (taking v_i as the i-th coordinate of v), the **norm** of v is

$$\|v\|_2 = \left(\sum_{i=1}^{N} v_i^2 \right)^{\frac{1}{2}}$$

Problem 2

Show that $\|v\|_2$ simplifies to our $2D$ "distance formula" for $N = 2$. Then, find an expression for the distance between two points, u and v, using the norm. We will say that points for which this quantity is small are *similar* or may be *clustered*.

Sidenote on Distance

For those curious, we can also define other notions of distance. For example, we write a so-called L_p norm

$$\|v\|_p = \left(\sum_{i=1}^{N} |v_i|^p \right)^{\frac{1}{p}}.$$

When $p = 1$, this simplifies to

$$\|v\|_1 = \sum_{i=1}^{N} |v_i|$$

which is called the "Manhattan distance," since it adds distances along gridlines (like driving on the city streets of Manhattan). We will use $\|v\|_2$ throughout this packet, but try to think about how certain processes may change if we measured distance differently!

Problem 3

Consider the case with u, v being vectors in $2D$ such that $\|u\|_1 = \|v\|_1$. Do we necessarily have $\|u\|_2 = \|v\|_2$? What about the converse (does $\|u\|_2 = \|v\|_2$ mean $\|u\|_1 = \|v\|_1$)? Show your answers mathematically.

Problem 4

Draw the set of points in 2 dimensions for which

1. $\|v\|_2 = 1$
2. $\|v\|_1 = 1$
3. $\|v\|_3 = 2$
4. $\|v\|_\infty = 4$

Supervised and Unsupervised Clustering

A few pages ago, we defined the three questions. We've answered the first one, but the latter two are really just restatements of the two main types of clustering problems, defined here:

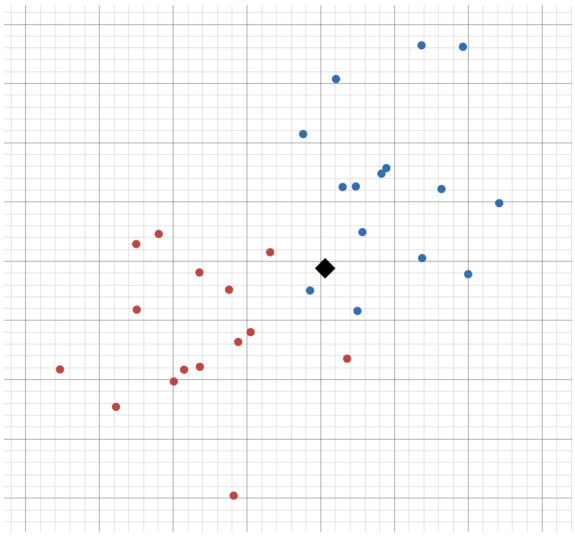
Definition: A clustering algorithm is said to be **supervised** if the data are already given with cluster assignments. Typically, we try to assign a new data point to an existing cluster in these applications.

Definition: On the other hand, an algorithm is said to be **unsupervised** if the data do *not* have cluster assignments prior. In this case, we typically try to assign the full dataset to new clusters. This is typically the *harder* of the two problems.

We will now start by illustrating one supervised clustering algorithm and two unsupervised algorithms, all applied to our real-world dataset.

Clustering Algorithms: K-Nearest Neighbors

To solve our second problem, we will consider an algorithm called **K-Nearest Neighbors (KNN)**. This algorithm is widely used in **supervised** clustering for its simplicity and effectiveness. Consider the following dataset:



Where there are two clusters already defined (blue: top right, red: bottom left). We would like to decide which cluster the diamond in the middle belongs to. Using KNN:

1. First, define some K value ($K \in \mathbb{N}$, meaning K is a natural number).
2. Now, find the K closest points to our query point (this may be done qualitatively in this case).
3. Now, find the cluster which a majority of the closest points belong to. This is the cluster which your point should be in. (If there's a tie, increase K and try again.)

Problem 5

What cluster do you intuitively think the point should belong to? Try running the KNN algorithm for a few different K values. Does your answer agree with the algorithm?

Problem 6

Can you think of a case for which this algorithm wouldn't work? Draw it and explain the confused result.

Remark: Hopefully the simplicity of this algorithm is clear. It seems boring for the time being, but for a computer, that simplicity allows us to generalize our result to hundreds of dimensions and millions of data points without taking too long to compute!

There are also many extensions of this algorithm. Imagine, instead of simply counting neighbors, weighting the effect of each nearby point by $\|v\|_2^{-1}$ (thereby making closer points more influential). You could even change the distance metric ($\|v\|_1$, for example) as discussed above for even more complex behavior.

Problem 7

Could either a different distance metric or weighting by distance solve your confused result from Problem 7? If not, can you think of a different version of the algorithm which could?

Clustering Algorithms: K-Means

Now that we know how to place a new point into an existing cluster, how can we generate clusters from an entirely unknown dataset? One algorithm to do so is called "K-Means." The process is split into an initialization and an iterative scheme:

Initialization:

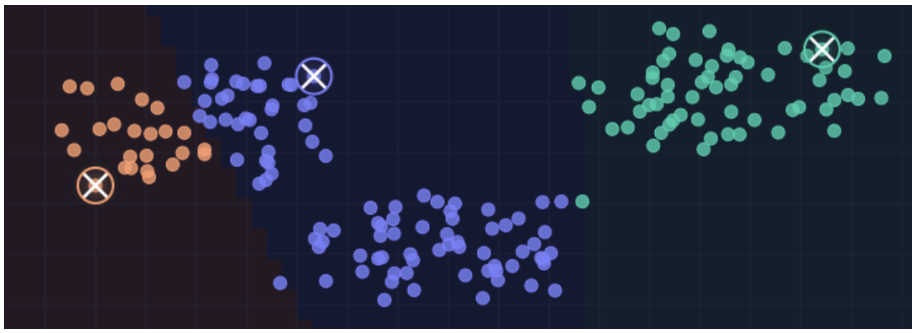
1. Again, decide a value of $K \in \mathbb{N}$ (this will represent the number of clusters).
2. Choose K data points randomly from the dataset. We define these as our initial "centroids."

This should look something like the following after this process:

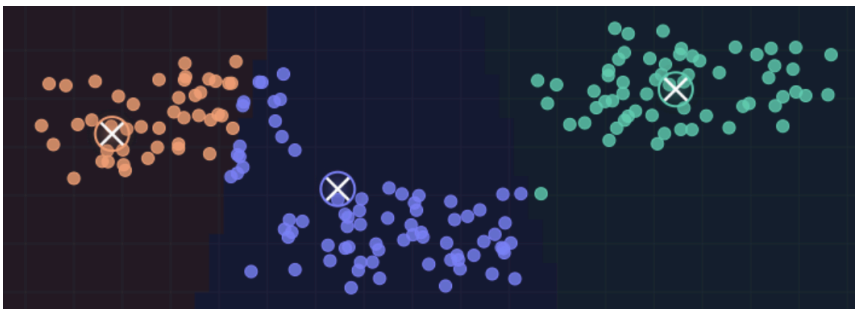
Iterative Scheme:

1. First, we iterate over every point in the dataset and assign it to its nearest centroid. In this way, the centroids define our clusters.
2. Then, taking all points defined to be a part of one cluster, redefine the centroid of that cluster as the **average** of all points in that cluster (where the average is taken over each dimension of the data uniquely).
3. If this process doesn't move the centroids, then we terminate the algorithm. Otherwise, repeat the process until the termination condition.

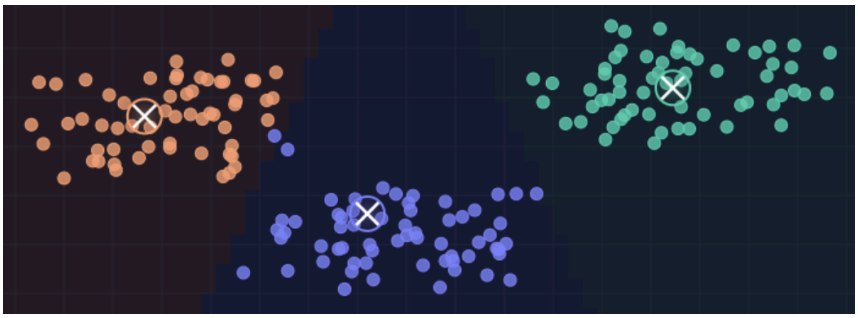
For example, from the above example, the steps would look like:



Then, after one step of the iterative scheme:



We run it again:



And after running one more time, finally converging on a satisfactory clustering of:

Problem 8

This algorithm works for our distance-based understanding of clustering, but can you think of a dataset we could reasonably call clustered that this wouldn't solve? Draw it and step through the algorithm until you're sufficiently convinced of its failure.

Problem 9

Walk through the K-Means algorithm qualitatively using the above dataset with both $K = 2$ and $K = 4$. How does the result change? Is it still a sensible clustering? Does this highlight a sensitivity or issue with the algorithm?

Application: Detecting Parkinson's Disease Through Voice

Parkinson's Disease (PD) is a neurological disorder that affects motor control, including the fine muscle movements that produce speech. People with Parkinson's (PwPD) often develop subtle but measurable changes in their voice: increased breathiness, irregular pitch, and a rougher, more chaotic vocal signal. Healthy Controls (HC) tend to have smoother, more periodic voices.

In our study we used 65,000 voice recordings from the mPower dataset, collected from roughly 5,800 participants using a smartphone app. Two features proved especially useful for separating PwPD from HC:

1. **Box Counting Dimension (BCD):** A measure of the fractal complexity of the voice signal. A perfectly periodic signal (like a pure tone) has low BCD, close to 1. A chaotic, irregular signal has higher BCD, approaching 2. PD tremor and vocal instability push BCD upward.
2. **Harmonics-to-Noise Ratio (HNR, measured in dB):** The ratio of periodic (harmonic) energy to random (noise) energy in the voice. A healthy voice is mostly periodic, with high HNR. A PD-affected voice has more noise mixed in, giving lower HNR.

Together, these two features create a natural 2D space where HC and PwPD tend to separate into distinct clusters. In the problems below, you will apply K-Means and KNN to a small representative sample of this data, entirely by hand.

Problem 10

The table below shows 8 voice recordings drawn from Avestimehr's dataset, each labeled by true diagnosis. Your task is to cluster them using K-Means with $K = 2$, without using the true labels, just as an unsupervised algorithm would.

Patient ID	BCD	HNR (dB)	True Label
P1	1.22	20.5	HC
P2	1.35	19.2	HC
P3	1.28	21.8	HC
P4	1.40	18.5	HC
P5	1.58	12.1	PwPD
P6	1.72	10.3	PwPD
P7	1.65	13.5	PwPD
P8	1.80	9.8	PwPD

Problem 11: K-Means Clustering of Voice Recordings

The table below shows 8 voice recordings drawn from Avestimehr's dataset, each labeled by true diagnosis. Your task is to cluster them using K-Means with $K = 2$, *without* using the true labels, just as an unsupervised algorithm would.

Patient ID	BCD	HNR (dB)	True Label
P1	1.22	20.5	HC
P2	1.35	19.2	HC
P3	1.28	21.8	HC
P4	1.40	18.5	HC
P5	1.58	12.1	PwPD
P6	1.72	10.3	PwPD
P7	1.65	13.5	PwPD
P8	1.80	9.8	PwPD

(a) Set up the plot. The coordinate grid below shows all 8 points plotted on BCD vs. HNR axes. Study the plot carefully before proceeding.

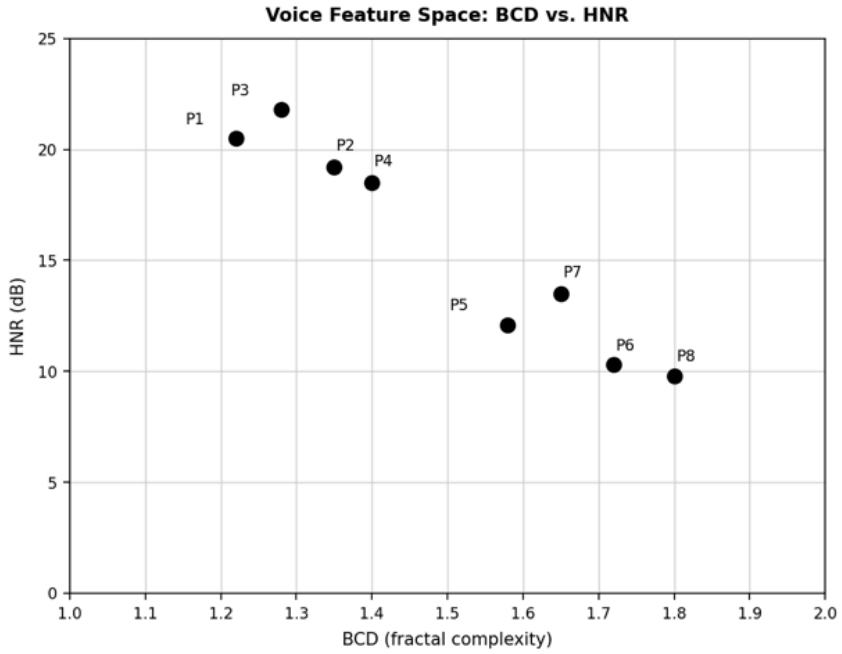


Figure 1: Figure 1. Voice feature space for 8 patient recordings. All points are unlabeled. Apply K-Means to find clusters.

Notice that the 8 points appear to form two natural groups. Without using the true labels, circle where you see the two groups on the plot.

(b) A deliberately bad initialization.

In practice, K-Means chooses its initial centroids randomly. Sometimes it gets lucky; sometimes it does not. To see what happens in the unlucky case, we start with two intentionally poor centroids, both sitting in the middle of the data, far from either true cluster:

$$\mu_1 = (1.30,\ 15.0) \quad \mu_2 = (1.60,\ 15.0)$$

Mark both centroids on the plot above with a star ($\star$) and label them μ_1 and μ_2. Notice that both stars sit at HNR $= 15$ dB, roughly halfway between the HC group (HNR $\approx$ 18–22) and the PwPD group (HNR $\approx$ 9–14). This is a bad start. Neither centroid is close to either true cluster center.

Before computing anything, just by looking at the plot, which points do you *expect* will be assigned to μ_1, and which to μ_2?

My guess: μ_1 will attract _____ and μ_2 will attract _____

(c) Iteration 1, Step 1 — Assign points to nearest centroid.

For each of the 8 points, compute the Euclidean distance to each centroid and assign the point to the nearer one:

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Patient	Dist. to $\mu_1 = (1.30, 15.0)$	Dist. to $\mu_2 = (1.60, 15.0)$	Assigned
P1 (1.22, 20.5)			
P2 (1.35, 19.2)			
P3 (1.28, 21.8)			
P4 (1.40, 18.5)			
P5 (1.58, 12.1)			
P6 (1.72, 10.3)			
P7 (1.65, 13.5)			
P8 (1.80, 9.8)			

After filling in the table, list your Iteration 1 cluster assignments:

Cluster 1 (μ_1): _____

Cluster 2 (μ_2): _____

(d) Iteration 1, Step 2 — Recompute centroids.

Using your cluster assignments from part (c), compute the new centroid of each cluster by averaging the BCD and HNR values of all points assigned to it:

$$\mu_{\text{new}} = \left(\frac{\sum x_i}{n}, \frac{\sum y_i}{n} \right)$$

Show your arithmetic below.

New μ_1:

$$\text{BCD avg} = \frac{\text{_____}}{n} = \text{_____} \qquad \text{HNR avg} = \frac{\text{_____}}{n} = \text{_____}$$

$$\text{New } \mu_1 = \left(\text{_____}, \text{_____} \right)$$

New μ_2:

$$\text{BCD avg} = \frac{\text{_____}}{n} = \text{_____} \qquad \text{HNR avg} = \frac{\text{_____}}{n} = \text{_____}$$

$$\text{New } \mu_2 = \left(\text{_____}, \text{_____} \right)$$

Mark the new centroid positions on the plot with an open circle (○) and draw a small arrow from each old centroid (⋆) to its new position. In which direction did each centroid move?

μ_1 moved: _____

μ_2 moved: _____

(e) Iteration 2, Step 1 — Reassign points.

Using your updated centroids from part (d), recompute the distance from each point to each new centroid and reassign.

Patient	Dist. to new μ_1	Dist. to new μ_2	Assigned
P1 (1.22, 20.5)			
P2 (1.35, 19.2)			
P3 (1.28, 21.8)			
P4 (1.40, 18.5)			
P5 (1.58, 12.1)			
P6 (1.72, 10.3)			
P7 (1.65, 13.5)			
P8 (1.80, 9.8)			

Did any points change their cluster assignment compared to Iteration 1?

Points that switched clusters: _____ (write "none" if no changes)

(f) Iteration 2, Step 2 — Recompute centroids again.

Recompute the centroids using the Iteration 2 assignments.

$$\text{New } \mu_1 = \left(_____,\ _____\right) \quad \text{New } \mu_2 = \left(_____,\ _____\right)$$

The true cluster centers (computed from the ground-truth labels) are:

$$\text{True HC centroid: } (1.3125,\ 20.00) \quad \text{True PwPD centroid: } (1.6875,\ 11.425)$$

How close are your Iteration 2 centroids to the true values?

Distance from μ_1 to true HC centroid: _____

Distance from μ_2 to true PwPD centroid: _____

(g) Convergence check.

Compare your Iteration 2 cluster assignments to your Iteration 1 assignments. Did any points move between clusters in Iteration 2? If no points moved, the algorithm has **converged**. The centroids will not change on the next iteration, and K-Means terminates.

Did the algorithm converge after Iteration 2? _____ (Yes / No)

If not, run Iteration 3 in the space below and check again.

(h) Reflection: Bad start, good finish.

Answer the following questions in 2–3 sentences each.

(i) We started with both centroids at HNR $= 15.0$, right in the middle of the data, not near either true cluster. Yet K-Means still found the correct clusters within 2 iterations. Why does this work? What property of the algorithm drives the centroids toward the true cluster centers even from a bad start?

(ii) Look back at part (b), where you guessed the initial assignments before computing. Was your geometric intuition correct? What does this tell you about the role of the HNR feature versus the BCD feature in separating the two groups?

(iii) Can you construct a starting configuration (two initial centroids placed somewhere on the plot) from which K-Means would *not* converge to the correct clusters? Describe it or draw it. (*Hint: think about what happens if both centroids start inside the same true cluster.*)

(iv) Interpretation.

Now reveal the true labels from the data table. Which cluster (μ_1 or μ_2) corresponds to HC, and which to PwPD? What do the final centroid locations tell you about the medical difference between the two groups? Specifically, what do the BCD and HNR values of each centroid mean for a patient's voice?

Problem 12: Measuring Clustering Accuracy

Now that K-Means has assigned each recording to a cluster, we can evaluate how well it performed by comparing the cluster assignments to the true labels.

(a) Compute the accuracy. Fill in the table below using your cluster assignments from Problem 9.

Patient	True Label	K-Means Cluster	Correctly Clustered?
P1	HC		
P2	HC		
P3	HC		
P4	HC		
P5	PwPD		
P6	PwPD		
P7	PwPD		
P8	PwPD		

Define **accuracy** as:

$$\text{Accuracy} = \frac{\text{number of correctly clustered points}}{\text{total number of points}}$$

What is the accuracy of K-Means on this dataset? Express your answer as a fraction and as a percentage.

(b) Is accuracy enough? Suppose K-Means achieved 75% accuracy on a larger dataset, but *all* of its errors were PwPD patients incorrectly placed in the HC cluster. The algorithm *missed* Parkinson's cases entirely. Is 75% accuracy acceptable for a medical screening tool? What is the consequence of a **false negative** (telling a PD patient they are healthy) versus a **false positive** (telling a healthy person they might have PD)?

(c) Sensitivity and specificity. In medical testing, accuracy alone is often insufficient. Two more precise measures are:

- **Sensitivity** (true positive rate): the fraction of actual PwPD patients correctly identified as PwPD.

$$\text{Sensitivity} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

- **Specificity** (true negative rate): the fraction of actual HC individuals correctly identified as HC.

$$\text{Specificity} = \frac{\text{true negatives}}{\text{true negatives} + \text{false positives}}$$

Using your K-Means results from part (a), compute the sensitivity and specificity of the clustering, treating PwPD as the positive class. Which is more important to maximize for a Parkinson's screening tool, and why?

Remark: In Avestimehr's full study on 65,000 recordings, the best classical models achieved sensitivities of only around 45–55% at the recording level, meaning roughly half of all PD recordings were missed. This is precisely why novel features like BCD and Visibility Graph Entropy (VGE) are being explored: they capture the fractal irregularity of PD voices in ways that standard acoustic features do not.

Problem 13: Classifying a New Patient

A new patient records a voice sample using the mPower smartphone app. Their extracted features are:

$$\text{BCD} = 1.32, \quad \text{HNR} = 20.1 \ \ \text{dB}$$

(a) Add this new point to the scatter plot from Problem 9 using a different marker (e.g., a triangle $\triangle$) so it stands out from the 8 training points.

(b) Rather than running K-Means again, we can classify this new point using the **converged centroids** from Problem 9. Compute the Euclidean distance from the new patient's point $(1.32,\ 20.1)$ to each final centroid:

$$d((1.32,\ 20.1),\ \mu_{\text{HC}}) = \underline{\qquad\qquad\qquad\qquad}$$

$$d((1.32,\ 20.1),\ \mu_{\text{PwPD}}) = \underline{\qquad\qquad\qquad\qquad}$$

(c) Assign the new patient to the nearer centroid. What is the predicted label — HC or PwPD?

Predicted label: _____

(d) Does the location of this point on the plot match your intuition about the prediction? Is this a confident prediction or a borderline case? What would it mean medically if we were uncertain about a patient's classification?

(e) *From hand calculation to real research.* The centroid-based prediction you just performed is the conceptual foundation of the supervised classifiers used in Avestimehr's research. In the full study, predictions are made across 65,000 recordings using models like Support Vector Machines and gradient-boosted trees (XGBoost), achieving AUC scores around 0.73. The unsupervised clustering you performed in Problem 9, finding that BCD and HNR naturally separate HC from PwPD, is what motivates using these features in the first place. Without that geometric separation, no classifier, however sophisticated, could reliably distinguish the two groups.

Clustering Algorithms: DBSCAN

For our final clustering algorithm, we will cover DBSCAN (Density Based Spatial Clustering of Applications with Noise). This process is not only more complex, but significantly more powerful. Importantly, part of this strength results from a unique approach to defining "clustered" data. Precisely,

Definition: In DBSCAN, data are considered clustered to one another not based on distance to some centroid or distance away from other clusters, but rather as points which are "**dense**" (close to one another).

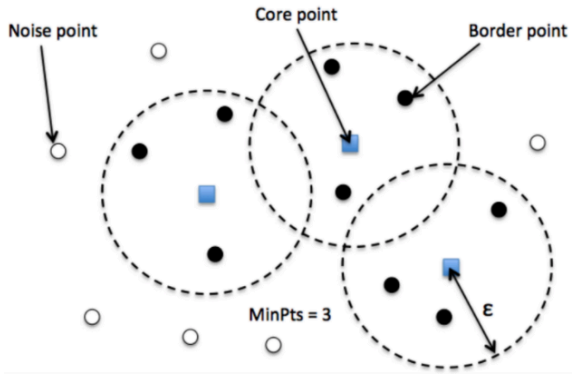
In order to begin understanding how this works, let's first walk through the algorithm itself:

Initialization:

1. First, we will define some parameters. Namely:

$$\begin{aligned} \varepsilon &= \text{maximum distance for points to be called clustered} \\ M &= \text{minimum number of nearby points to constitute a cluster} \end{aligned}$$

*Note that we do **not** need to specify the number of clusters — this is a strength of DBSCAN as opposed to K-Means.*



Iterative Scheme:

1. Select a random point (P_0) which does not currently belong to a cluster. If there are at least M points P with $\|P - P_0\|_2 \leq \varepsilon$, call it a "core point." If not, skip steps 2, 3.
2. Define all points P with $\|P - P_0\|_2 \leq \varepsilon$ as border points (even if they were previously labelled as noise).
3. For each border point, check if there are M points P with $\|P - P_0\|_2 \leq \varepsilon$. If so, it is also a core point of this cluster and repeat the process.
4. Once we exhaust all such points, we terminate, and
   - If we have less than M core points in this cluster, we do *not* assign these points to an actual cluster and simply label them as *noise*.
   - If we have M or more core points, this set will become a cluster and these points will be set aside in following iterations.

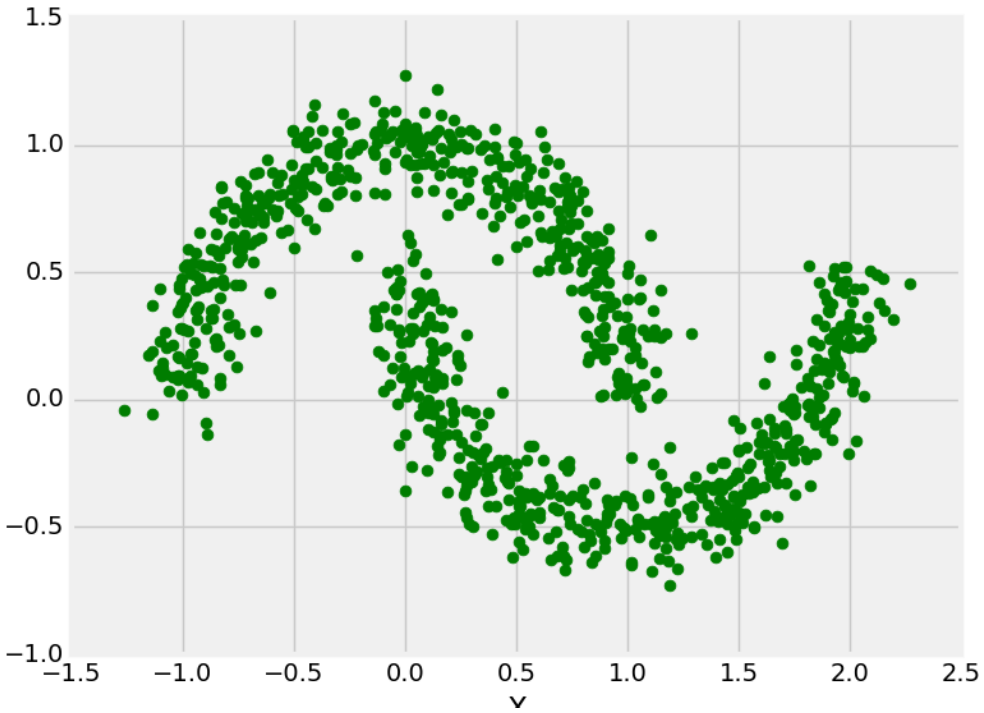
Remark: We see, then, that DBSCAN has some advantages over other methods like K-Means. Precisely, we don't presume to know anything about the data before clustering — there is no K value to guess, just a general idea of the "strength" of clustering.

Additionally, DBSCAN inherently accounts for "noisy" or non-sense data and removes it. For this reason, DBSCAN is actually commonly used to filter noisy data out of complex, n-dimensional datasets even in non-clustering applications.

Remark: For those of you with some computer science experience, you may realize that the above algorithm is not *nearly* optimized. In fact, it is a good exercise to try to optimize this process with some basic data structures (using a queue, perhaps).

Problem 14

Given the dataset below (a common test dataset in data science called "moons" for its crescent moon shape), draw what you think the clusters would be for K-Means. Repeat for DBSCAN. Which is more sensible? What differences does this highlight between the two algorithms?



Problem 15

Use the space below to draw another dataset that you might expect DBSCAN to succeed on which K-Means wouldn't. Then also draw one which you think DBSCAN might not work for. Could you tune DBSCAN's parameters to fix it for this dataset or not?

Some Introductory Analysis

Mathematical analysis is a very deep and complex field of study, but we may borrow on some basic principles to understand these algorithms more deeply. More precisely, let's ask the question:

> How do we even know algorithms like K-Means or DBSCAN terminate?

Clearly, it would be an issue if they didn't, and we could maybe even imagine some ugly datasets which seem not to (consider a dataset that forces the centroids of K-Means to oscillate between two positions, for example). It would be very useful to prove that such a case couldn't possibly exist — we will borrow on some analysis to do so!

First, the case for DBSCAN is easier, so we start there.

Problem 16

Convincingly argue that, for a dataset of finite length, DBSCAN will necessarily exhaust all points.

Hint: This doesn't need to be super rigorous; a counting argument will do. Just show that there will certainly be no points missed.

More interesting is the case of K-Means. How can we be convinced that the **centroids of K-Means will converge** to a certain point and terminate? It will be helpful to define the following ideas from machine learning and analysis:

Definition: A **Loss Function** is a function which represents how far an algorithm is from its desired result. This is often represented by a distance (like $\|v\|_2$ or $\|v\|_1$) and is minimized in the process of the algorithm.

Specifically, in the case of clustering, we often use a **Sum of Squared Errors (SSE)** loss function. Mathematically, for centroid C and a set of points $\mathcal{P}$ with $|\mathcal{P}|$ points and $\mathcal{P}_i$ denoting the i-th point, this is written:

$$\mathcal{L}(C, \mathcal{P}) = \sum_{i=1}^{|\mathcal{P}|} \left(\|C - \mathcal{P}_i\|_2^2 \right)$$

Theorem: (*Monotone Convergence Theorem*) For an infinite sequence a_n with a_i denoting the i-th element of the sequence, we know

$$a_{n+1} < a_n \quad \text{and} \quad a_n \text{ bounded below} \quad \Longrightarrow \quad a_n \quad \text{converges.}$$

Notably, this is also true for

$$a_{n+1} > a_n \quad \text{and} \quad a_n \text{ bounded above} \quad \Longrightarrow \quad a_n \quad \text{converges.}$$

In English, this is saying: if you have a sequence which is always decreasing (or always increasing) but is still always above (or below) some value — meaning it cannot go off to infinity — then the sequence must converge to some finite value. This is an extremely important result in analysis, and one you'll have the privilege to use now!

Sequence Convergence Practice

For each of the following, prove that the sequence converges using the Monotone Convergence Theorem:

1. $a_n = \frac{1}{n}$
2. $a_n = c - c^{-n}$
3. $a_1 = 3, a_{n+1} = \sqrt{2a_n}$

Problem 17: Convergence of K-Means

Using the above tools and theorems, prove *rigorously* that the K-Means algorithm necessarily converges to a terminal result. This is a tough exercise, so some hints are provided below and the page is given as workspace:

Hint: The general sketch of the proof is as follows. Read as much as you see fit and try to do the rest:

1. Define some quantity which represents the sum of the loss across all clusters.
2. Show that, in both the reassignment step *and* the recentering step, this quantity decreases.
3. Use this to show the sequence is monotone decreasing.
4. Show a lower bound on the value of the loss function.
5. Argue that, if the loss function converges, so does the location of the centroids (use that there are a finite number of locations for the centroids).

Additionally, thanks to Kiana who provided the Parkinson's data for this packet, we have an interactive demo of K-Means, DBSCAN, and another type of clustering called "Hierarchical Clustering." Ask your instructor if you'd like to use the demo or, after class, look at the demo on your own. A link will have been sent home, or you can copy the link from here: https://tinyurl.com/ormcclustering

Homework

Problem 18

Show, for $0 < a < b$ and $v \in \mathbb{R}^n$ (meaning v is an n-dimensional vector):

$$\|v\|_a \leq \|v\|_b$$

Problem 19

Imagine you have a very, very, very large dataset to run K-Means on. You could imagine, despite necessarily converging, it might take significantly longer time to converge for certain random initializations than others. Propose a method of intelligently deciding the initialization of centroids which could solve this problem.

Problem 20

Using the Monotone Convergence Theorem, prove

$$a_n = \left(1 + \frac{1}{n}\right)^n$$

converges to a real number. Now, define

$$e \coloneqq \lim_{n \to \infty} a_n.$$

This is one common formalization of the well known Euler number, e, for which you've just proved convergence.