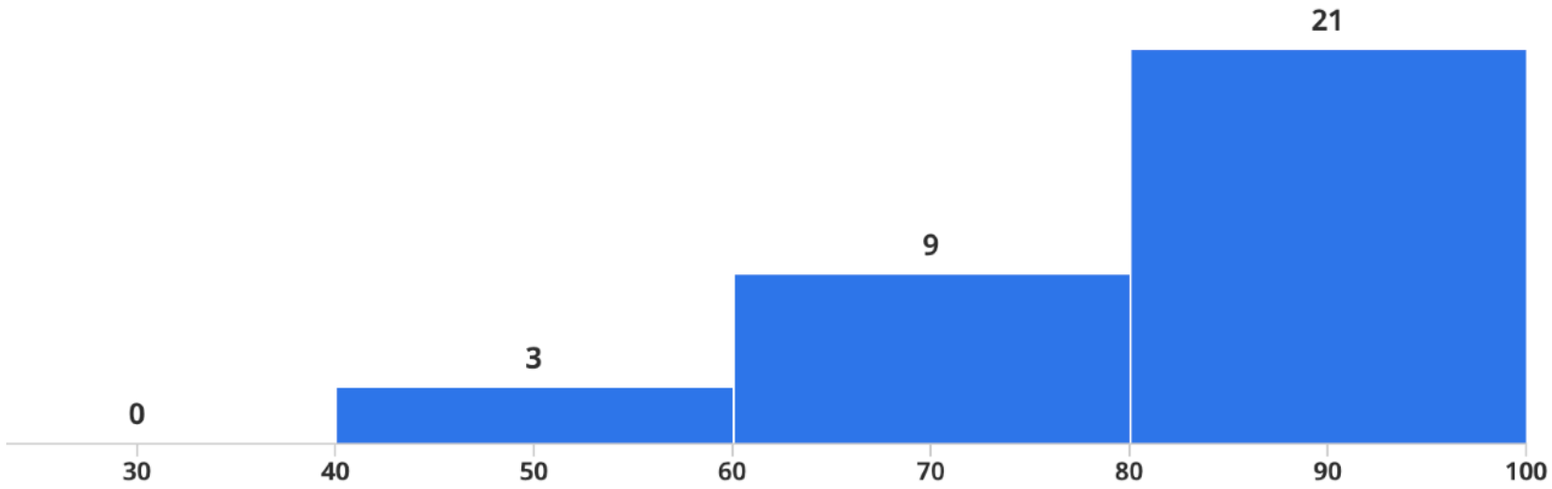
Carnegie Mellon University
Heinz College

94-775 Unstructured Data Analytics for Policy

Lecture 9: Clustering (cont'd), TF-IDF representation, topic modeling

Slides by George H. Chen

Quiz 1



Median	Maximum	Mean	Std Dev
85.0	99.0	83.18	13.27

Remember: letter grades are assigned based on a curve
(please do not panic! — no one is currently at risk of failing the course)

Solutions are in Canvas -> Files -> “Quiz 1 solutions.pdf”

Regrade requests (use Gradescope’s regrade request feature)
are due **Wed April 8, 11:59pm**
(for if you think there’s a genuine grading error)

Some Remarks on Quiz 1 Grading

- Very importantly, I emphasize fairness in grading
  - Two students who make the same amount of progress/same mistake(s) receive the same partial credit
  - Each problem part/subpart is graded by a single grader (to ensure consistency in how it is graded across students)
  - It's possible that some parts/subparts may appear to be graded harsher than others as a result (e.g., some graders may be harsher)

Other Administrivia Items

- HW1 has also been graded—regrades due Wed Apr 8, 11:59pm

- Your final project proposals are due tomorrow (1 email per group) by 11:59pm 😱

Coverage for the Rest of This Week

Today:

- Clustering using HDBSCAN (& DBSCAN)
  - In lecture, I will only go over using these on toy data
- An alternative approach to representing text data using "TF-IDF" representation
  (less intuitive than TF or probability vectors but commonly used)
  - We'll see this on the 20 Newsgroups dataset
- Time permitting: topic modeling with latent Dirichlet allocation (LDA)

Friday recitation (tentative plan):

- Longer version of clustering on text demo
  - Look at how RSS, CH index, silhouette score behave
  - Look at how HDBSCAN works on 20 Newsgroups data
- Clustering on images demo
- Quick check-in on how your final projects are going

An alternative approach that does not require setting the number of clusters

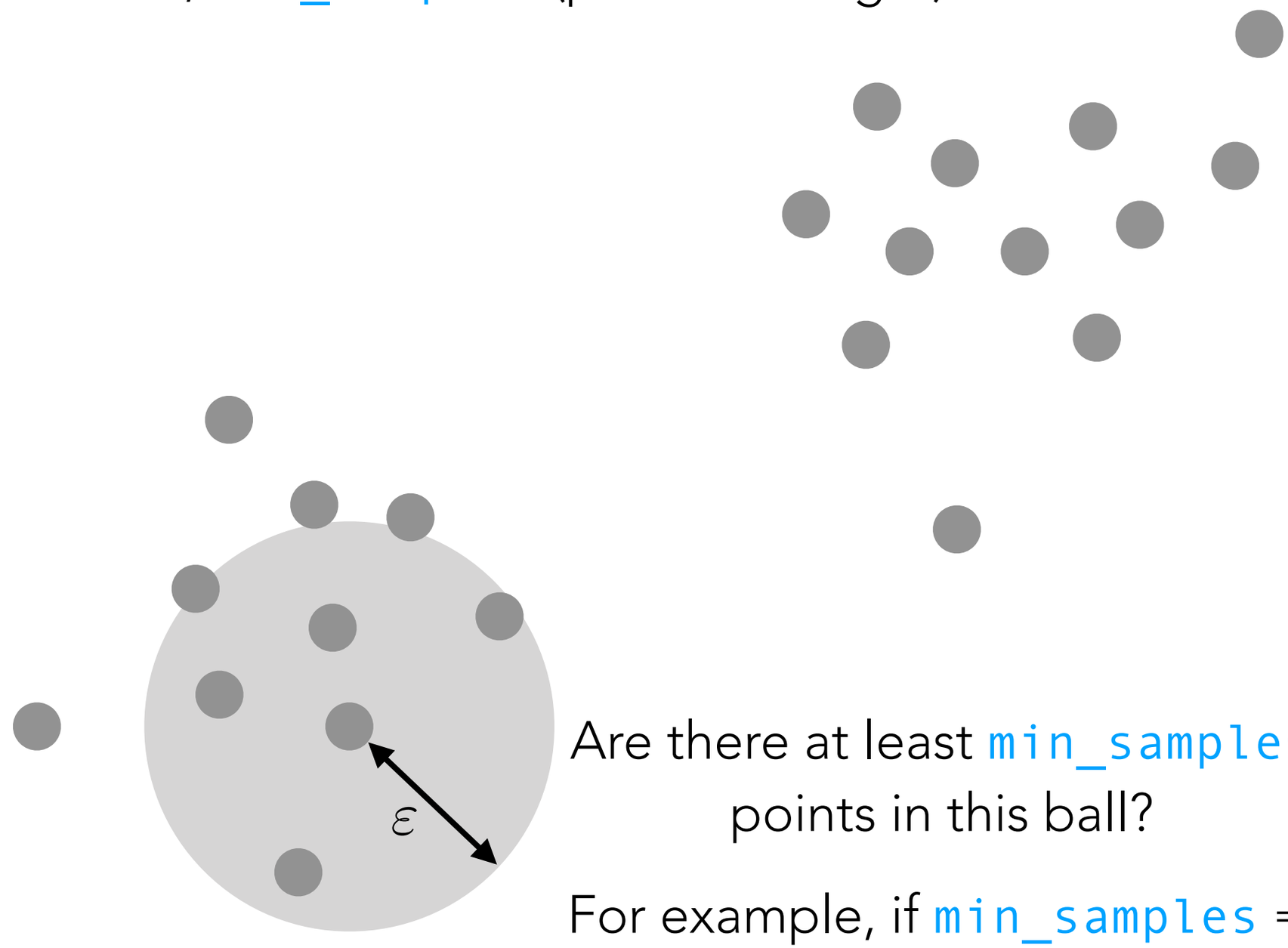
We'll build up to an approach called HDBSCAN
(Hierarchical Density-Based Spatial Clustering of Applications with Noise)

First, we cover DBSCAN

DBSCAN

Let's choose `min_samples` = 5

Pick radius $\varepsilon > 0$, `min_samples` (positive integer)



Are there at least `min_samples` points in this ball?

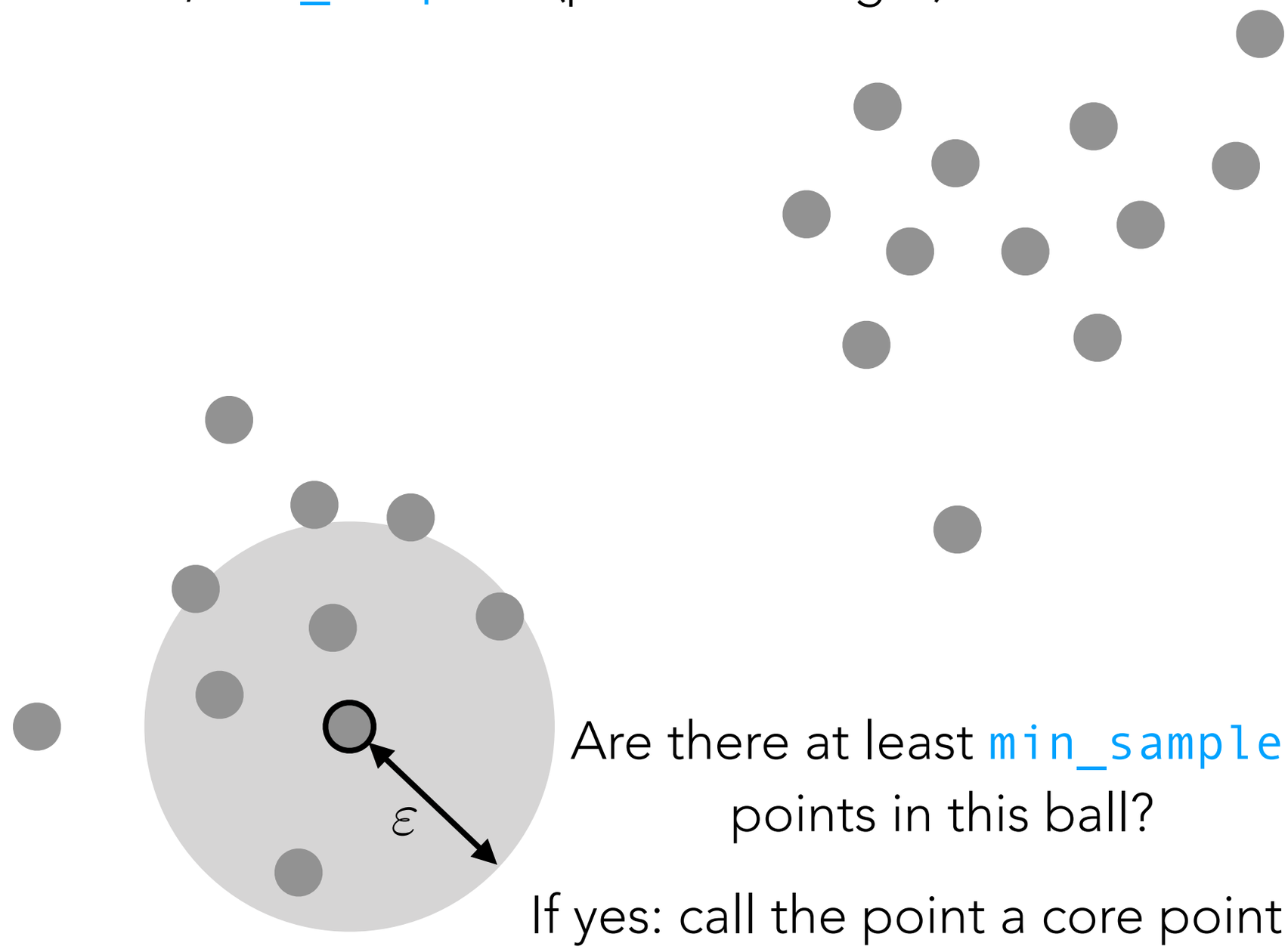
For example, if `min_samples` = 5, then yes

For example, if `min_samples` = 10, then no

DBSCAN

Let's choose `min_samples` = 5

Pick radius $\varepsilon > 0$, `min_samples` (positive integer)



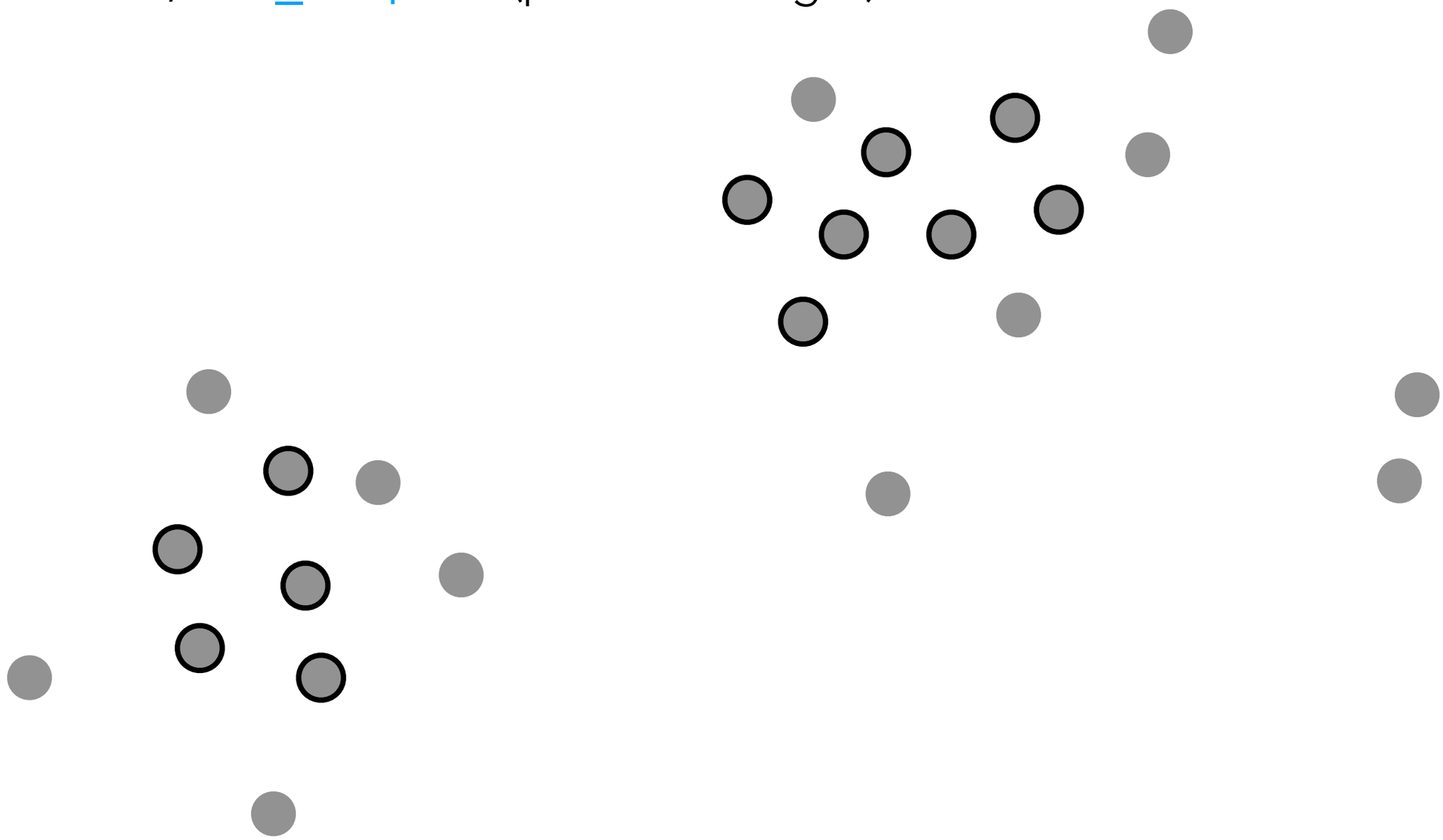
Are there at least `min_samples` points in this ball?

If yes: call the point a core point (**outlined**)

DBSCAN

Let's choose `min_samples` = 5

Pick radius $\varepsilon > 0$, `min_samples` (positive integer)

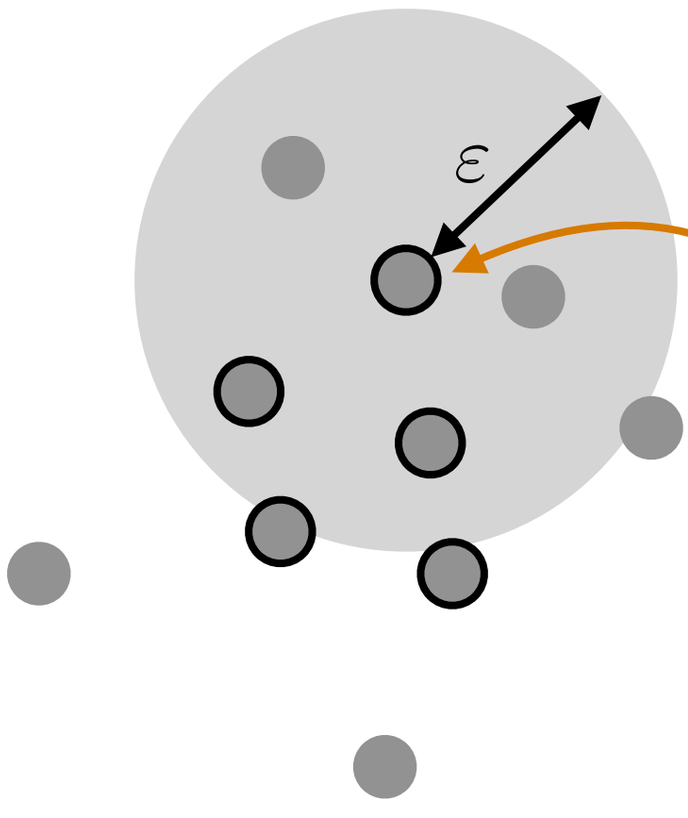


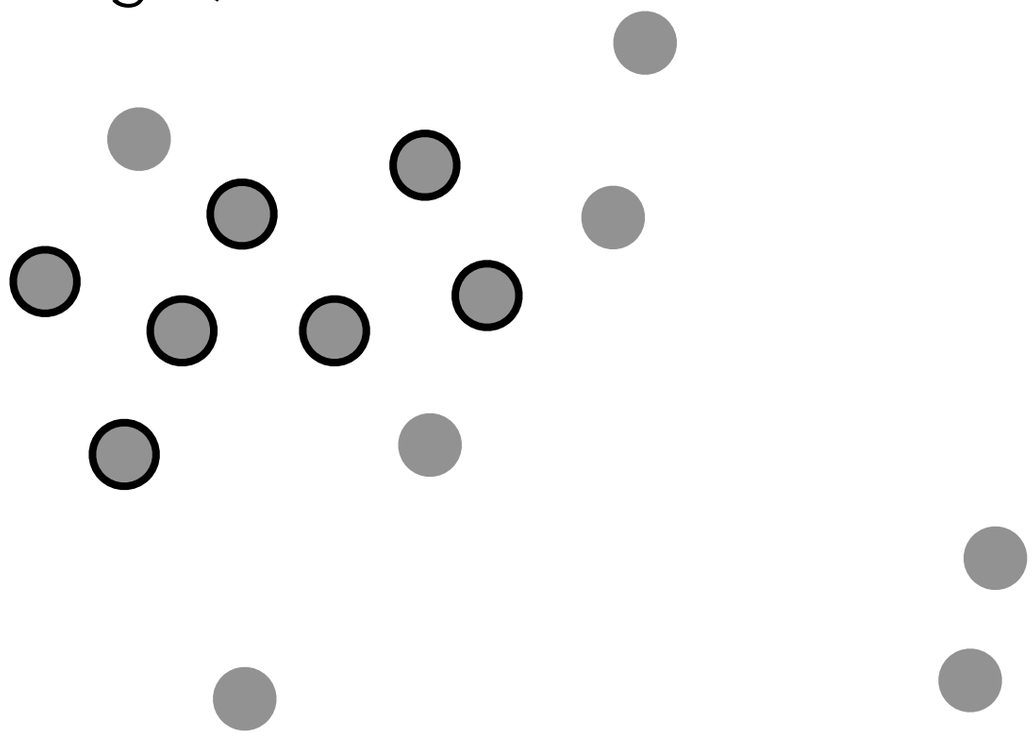
Core points (**outlined**)

DBSCAN

Let's choose `min_samples` = 5

Pick radius $\varepsilon > 0$, `min_samples` (positive integer)





Pick a core point that hasn't been assigned a cluster yet

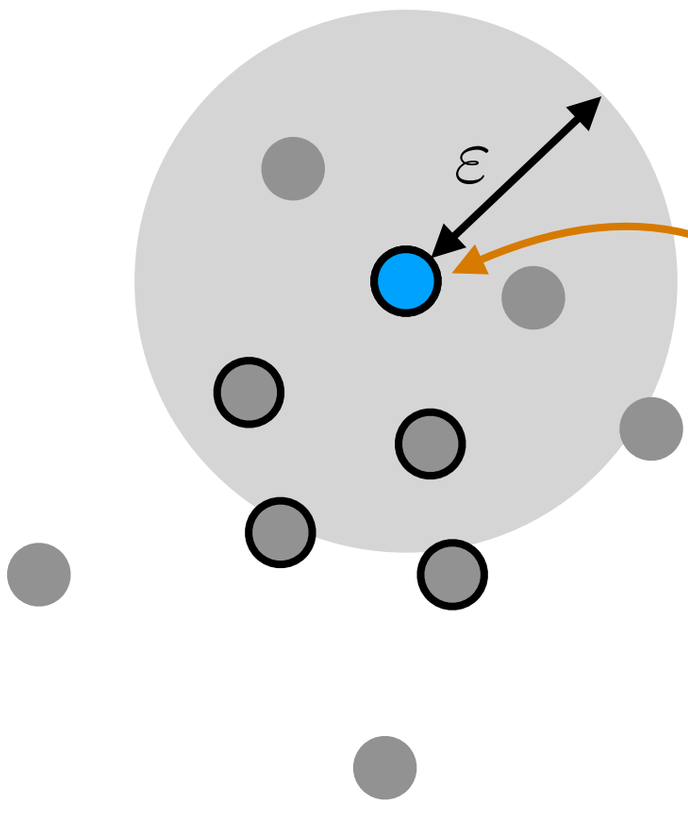
Core points (**outlined**)

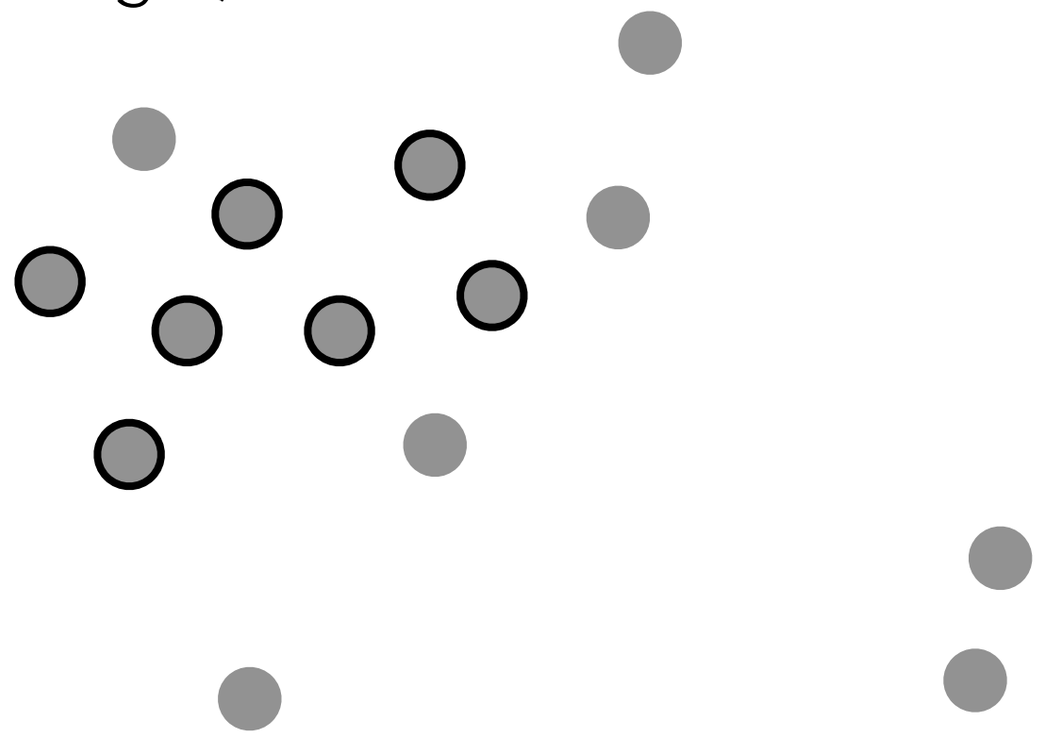
DBSCAN

Let's choose `min_samples` = 5

Pick radius $\varepsilon > 0$, `min_samples` (positive integer)

Start new cluster





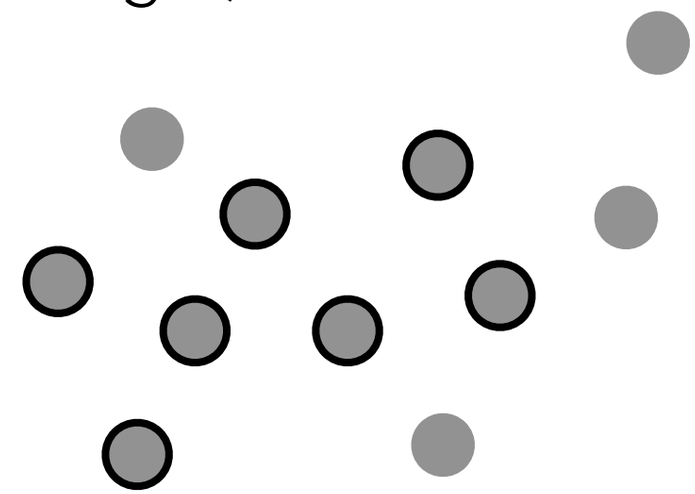
Pick a core point that hasn't been assigned a cluster yet

Core points (**outlined**)

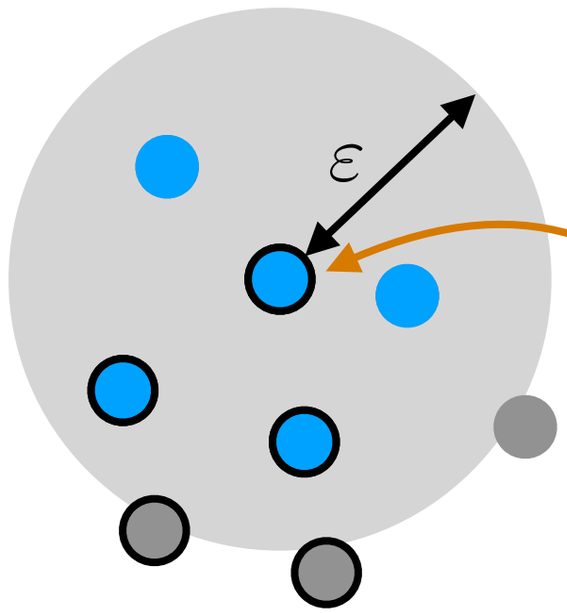
DBSCAN

Let's choose `min_samples` = 5

Pick radius $\varepsilon > 0$, `min_samples` (positive integer)



Start new cluster



Spread cluster label to points within ball

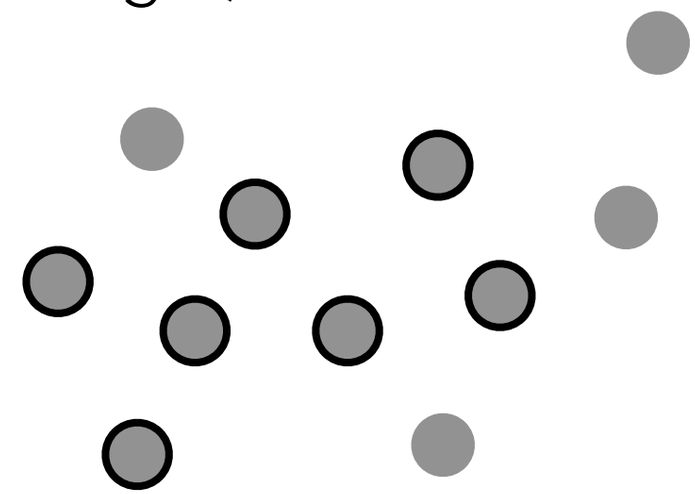
Pick a core point that hasn't been assigned a cluster yet

Core points (**outlined**)

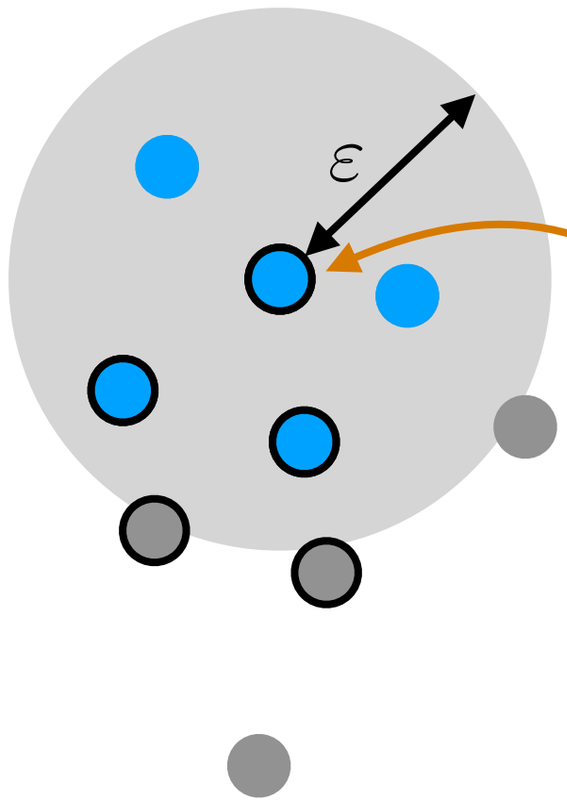
DBSCAN

Let's choose `min_samples` = 5

Pick radius $\varepsilon > 0$, `min_samples` (positive integer)



Start new cluster



Spread cluster label to points within ball

Pick a core point that hasn't been assigned a cluster yet

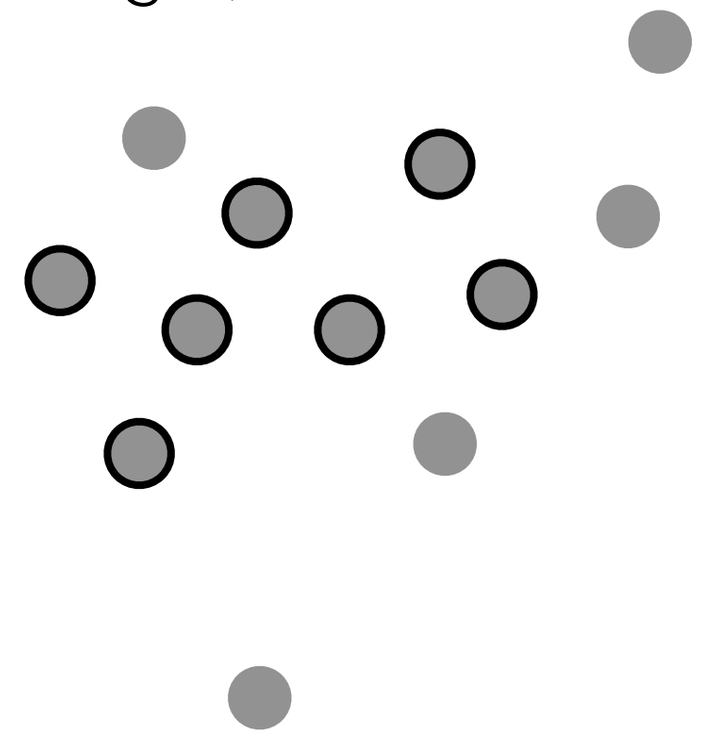
For core points within the ball: spread the cluster label

Core points (**outlined**)

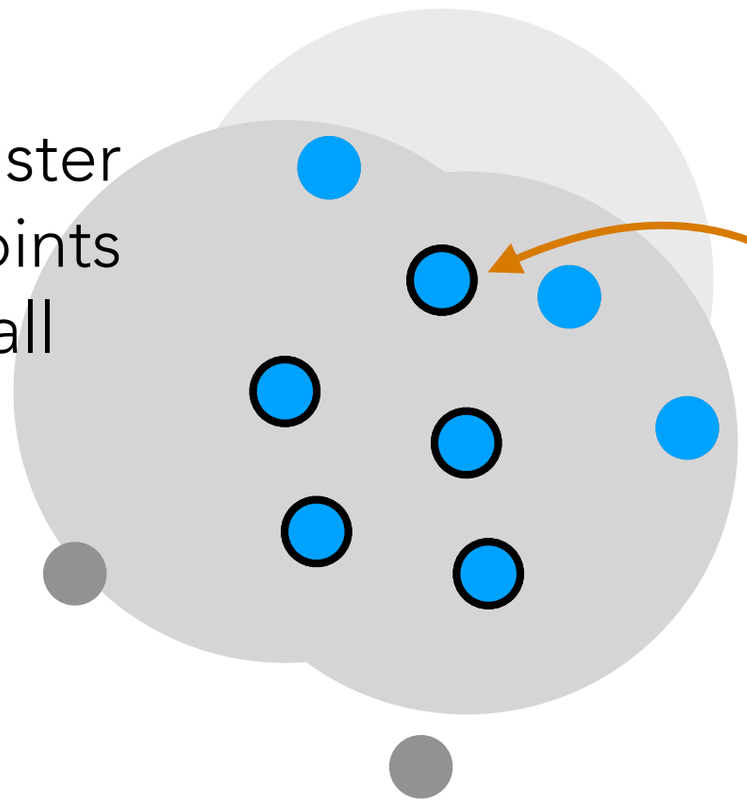
DBSCAN

Let's choose `min_samples` = 5

Pick radius $\varepsilon > 0$, `min_samples` (positive integer)



Start new cluster



Spread cluster label to points within ball

Pick a core point that hasn't been assigned a cluster yet

For core points within the ball: spread the cluster label

Core points (**outlined**)

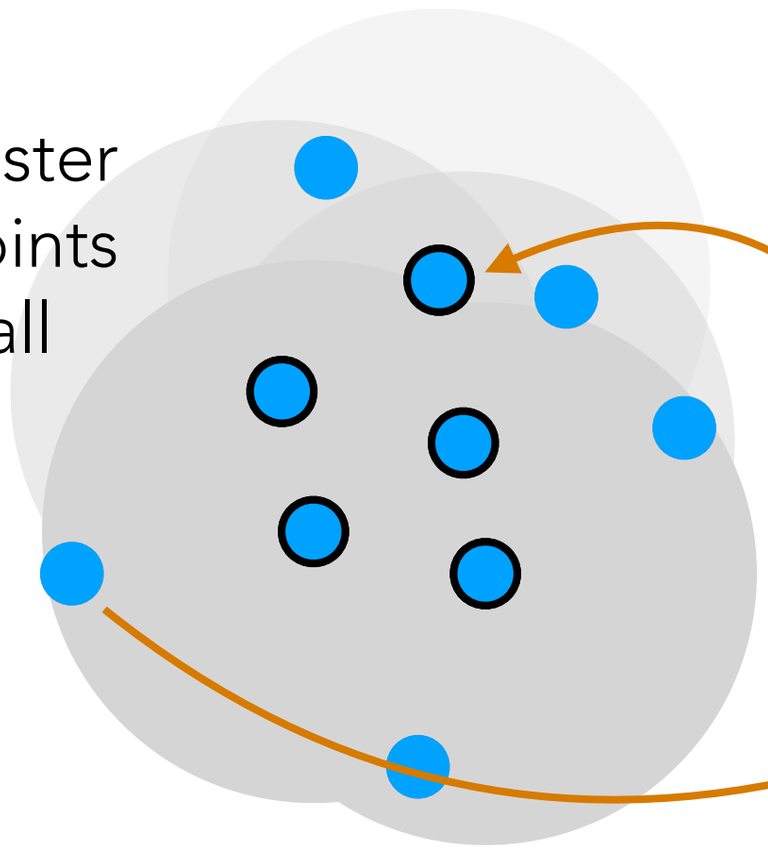
DBSCAN

Let's choose `min_samples` = 5

Pick radius $\varepsilon > 0$, `min_samples` (positive integer)

Start new cluster

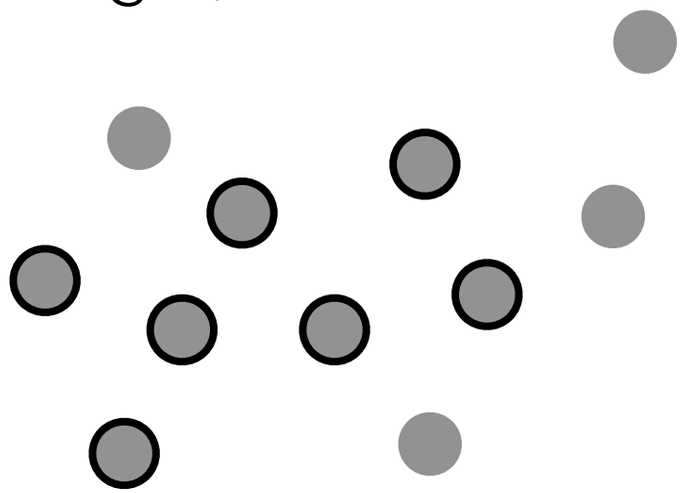
Spread cluster label to points within ball



Pick a core point that hasn't been assigned a cluster yet

Not a core point, so no spreading of label from here!

For core points within the ball: spread the cluster label



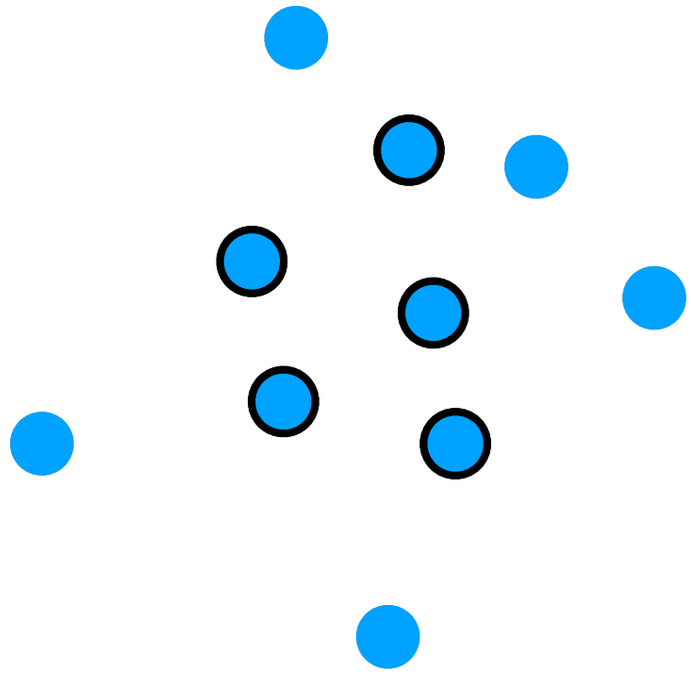
Core points (**outlined**)

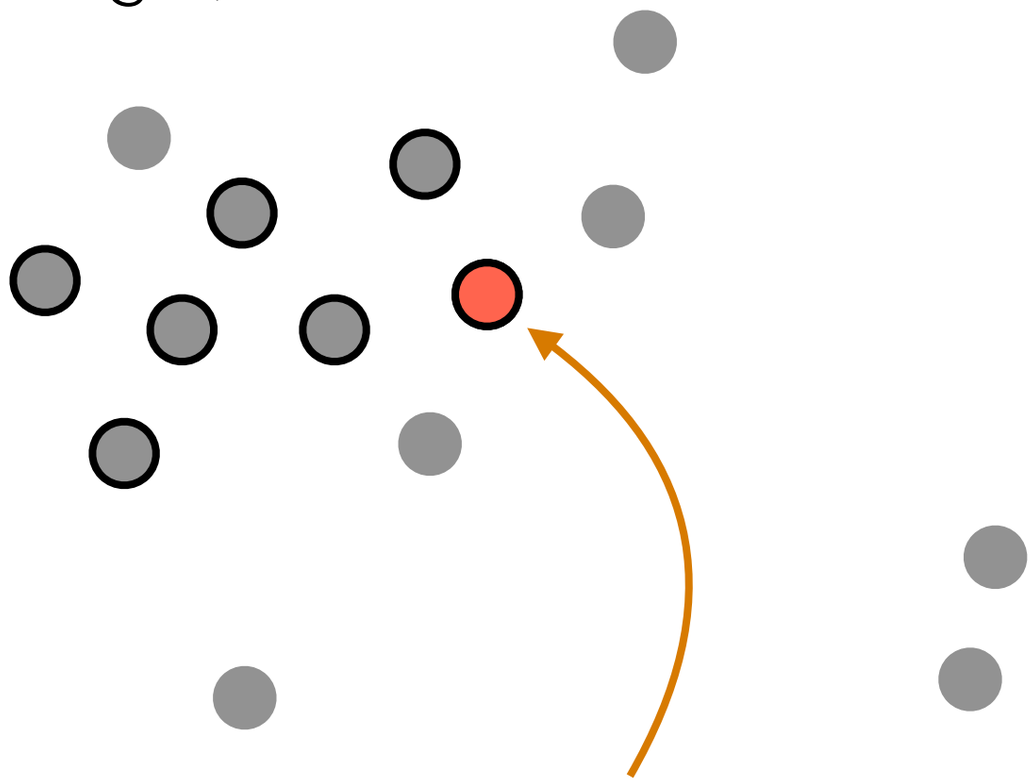
DBSCAN

Let's choose `min_samples` = 5

Pick radius $\varepsilon > 0$, `min_samples` (positive integer)

Repeat virus-like cluster label spreading (again, no spreading starting from non-core points)





Pick a core point that hasn't been assigned a cluster yet

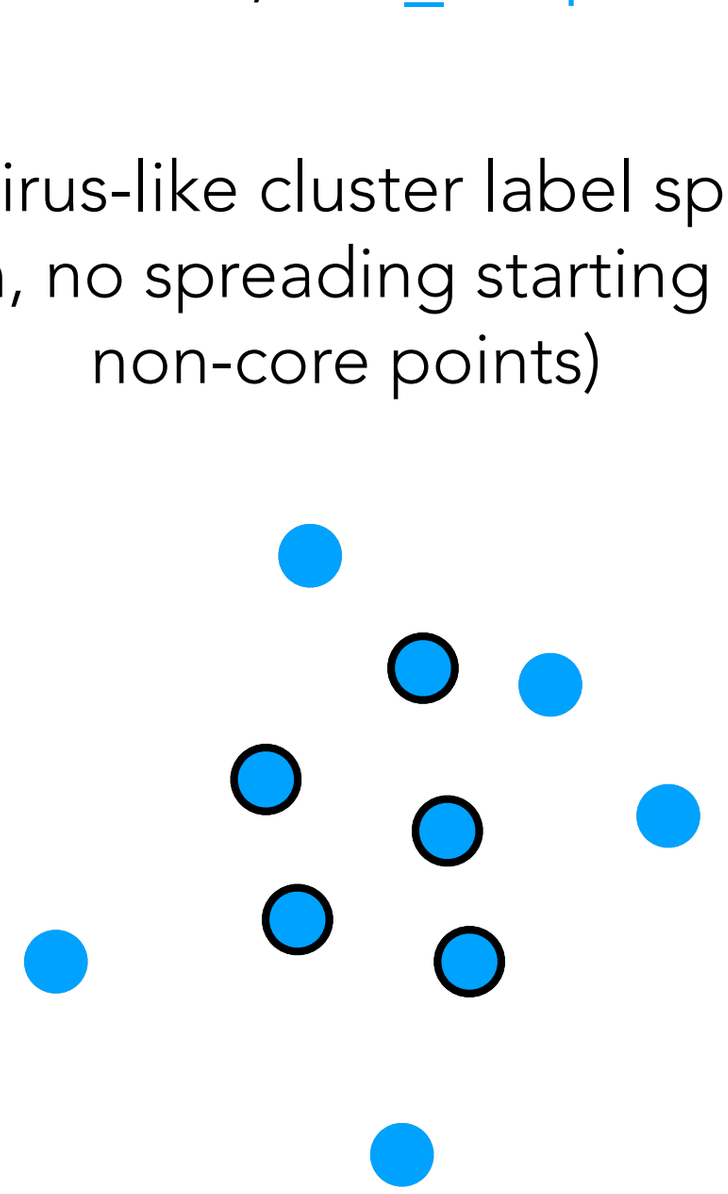
Core points (**outlined**)

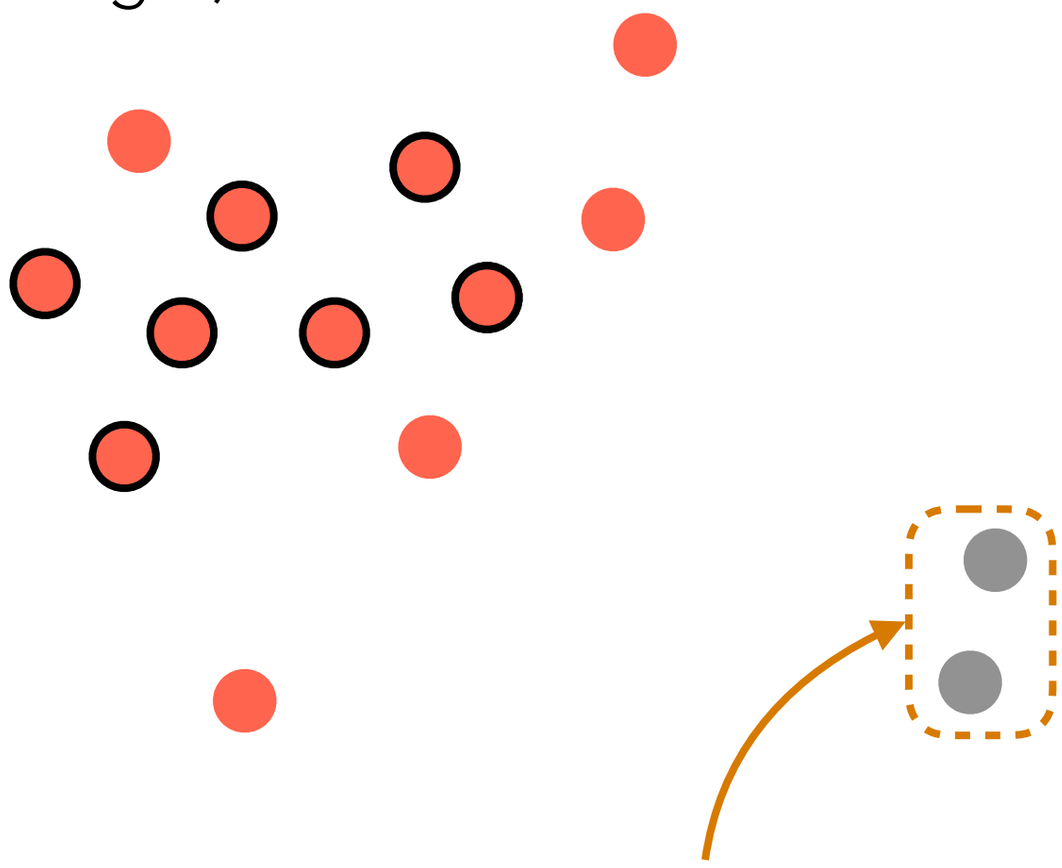
DBSCAN

Let's choose `min_samples` = 5

Pick radius $\varepsilon > 0$, `min_samples` (positive integer)

Repeat virus-like cluster label spreading (again, no spreading starting from non-core points)





Some points might actually *not* get clustered by DBSCAN and are declared as noise!

Core points (**outlined**)

DBSCAN

Demo

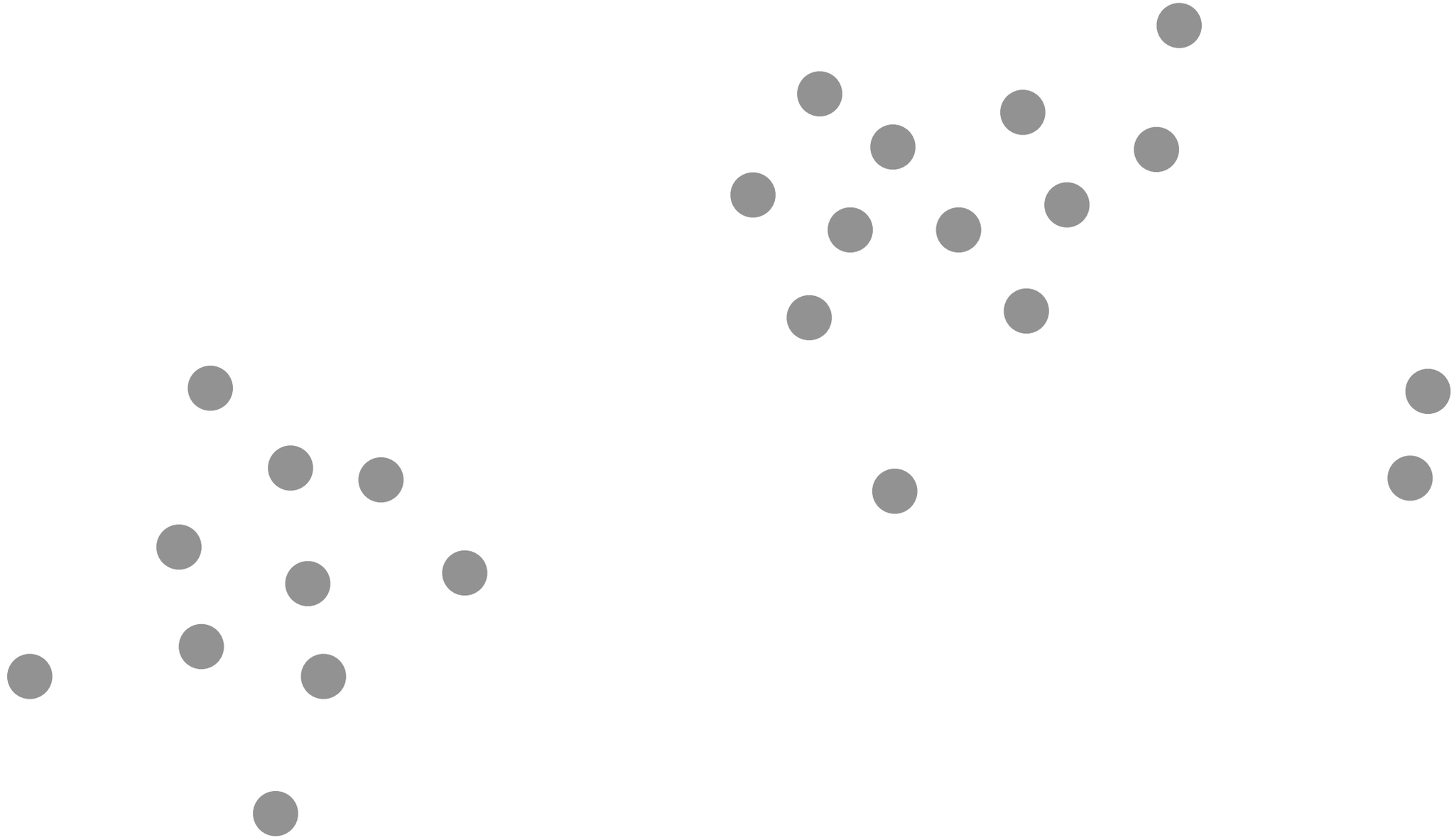
HDBSCAN is basically a version of DBSCAN that sweeps over ε for you

Conceptually, what happens in DBSCAN as we vary ε?

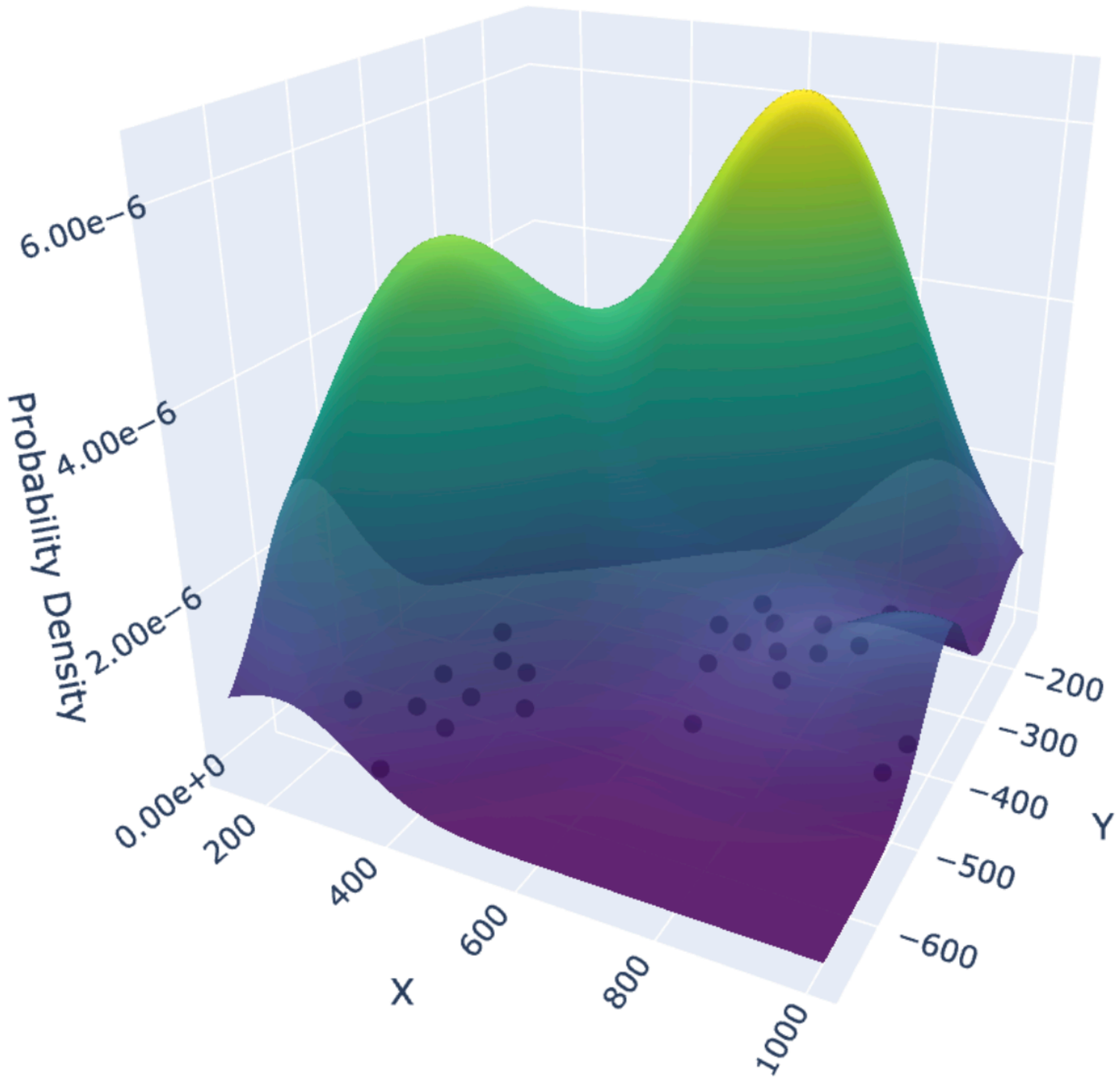
DBSCAN

Demo: varying epsilon for DBSCAN & probability density visualization

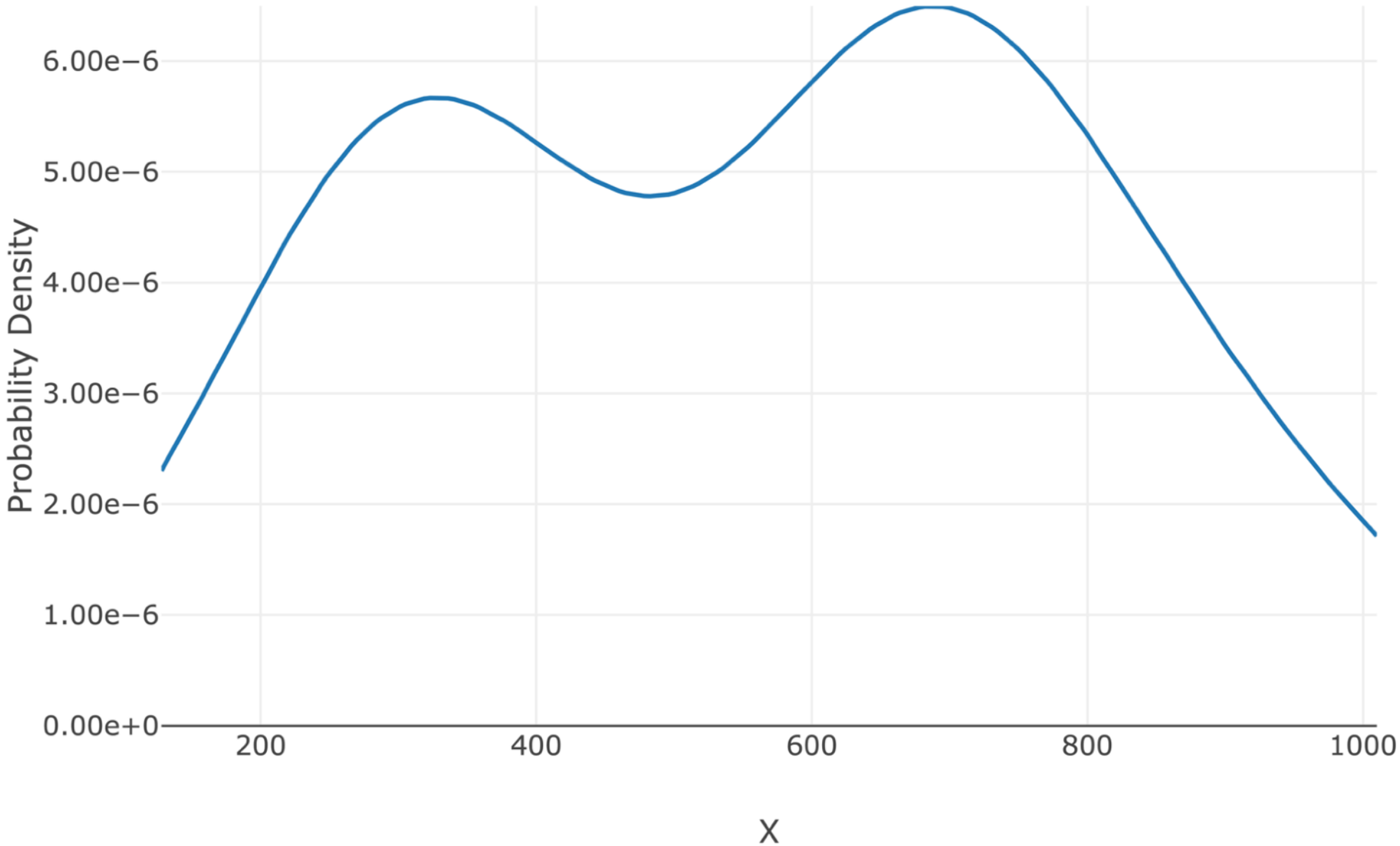
The Original Data Points



Probability Density of the Data



Probability Density of the Data (Showing Cross Section)

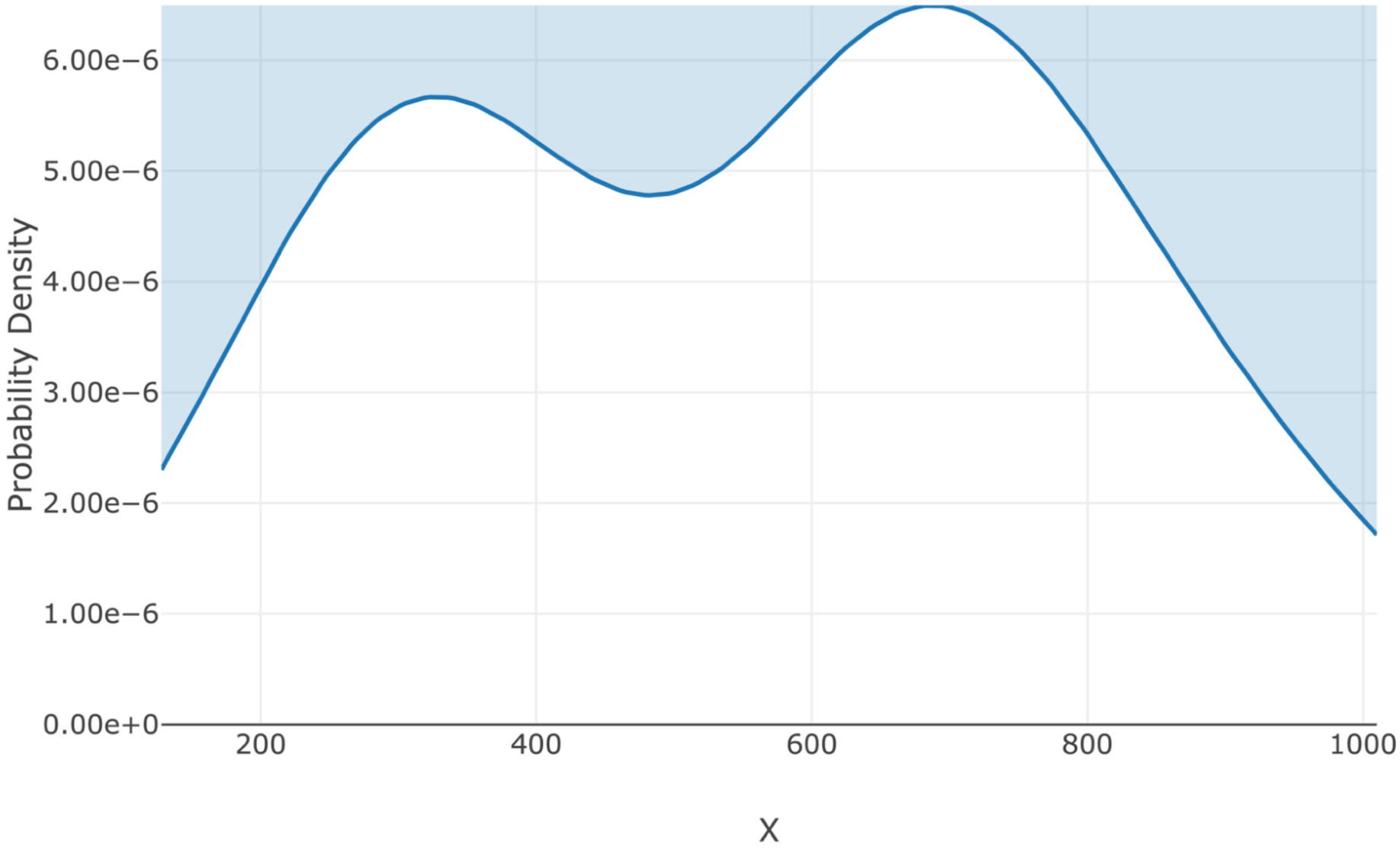


When we vary ε, a rough analogy is that we're getting rid of a flood

Probability Density of the Data (Showing Cross Section)

When we vary ε, a rough analogy is that we're getting rid of a flood

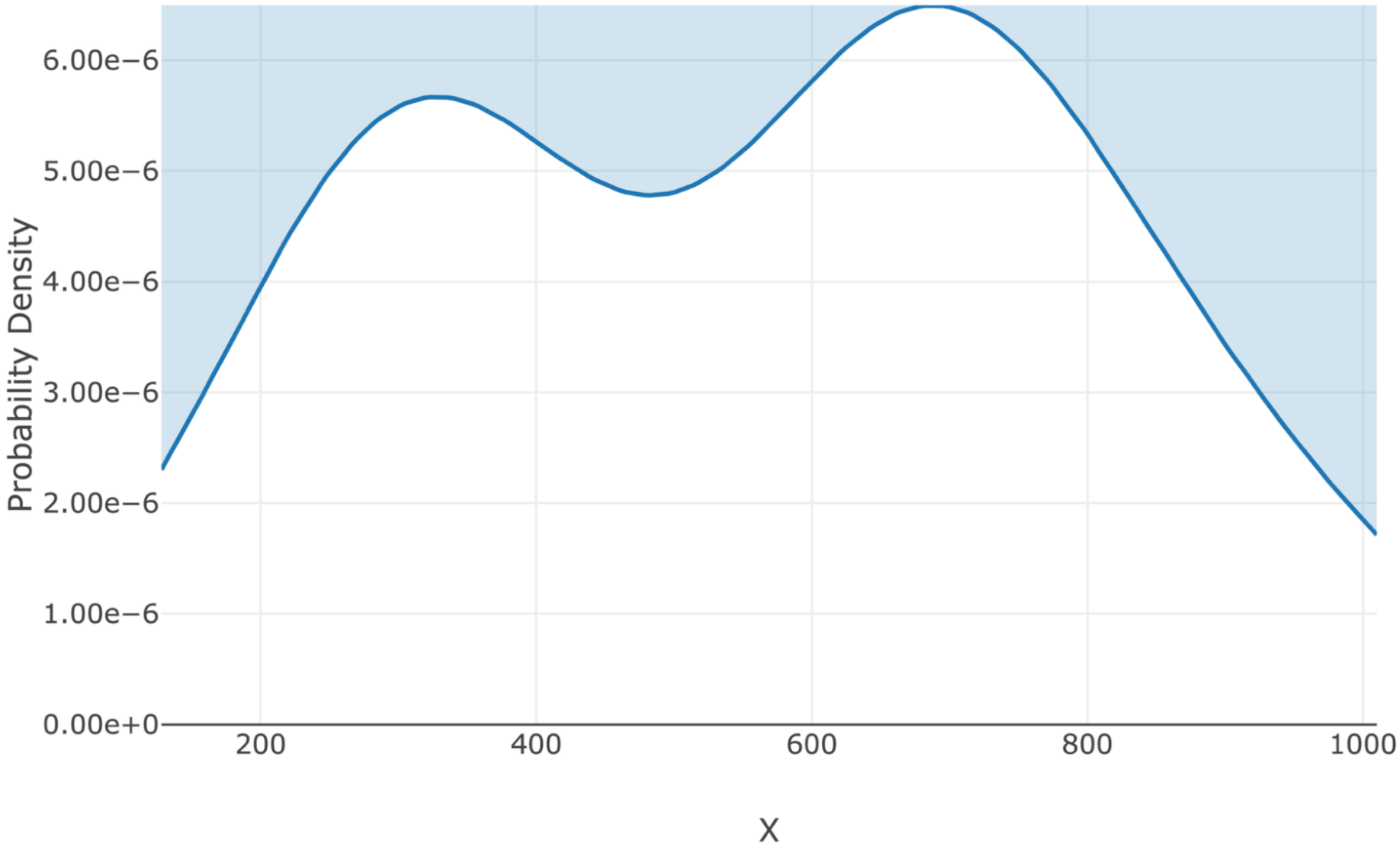
When ε increases, we are *lowering the water level*



Probability Density of the Data (Showing Cross Section)

When we vary ε, a rough analogy is that we're getting rid of a flood

When ε increases, we are *lowering the water level*

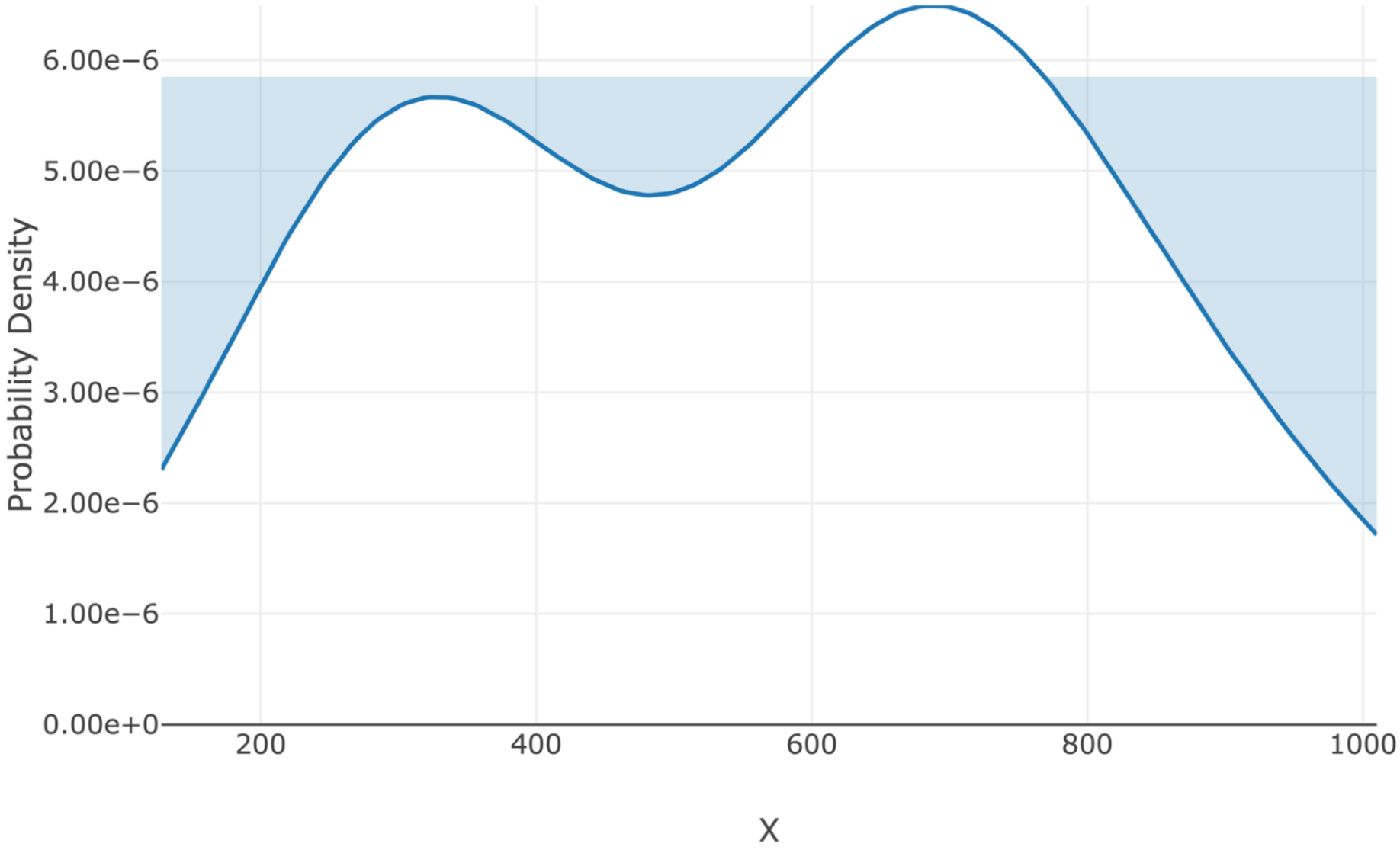


No land is above water → no clusters are detected (all points labeled as noise)

Probability Density of the Data (Showing Cross Section)

When we vary ε, a rough analogy is that we're getting rid of a flood

When ε increases, we are *lowering the water level*

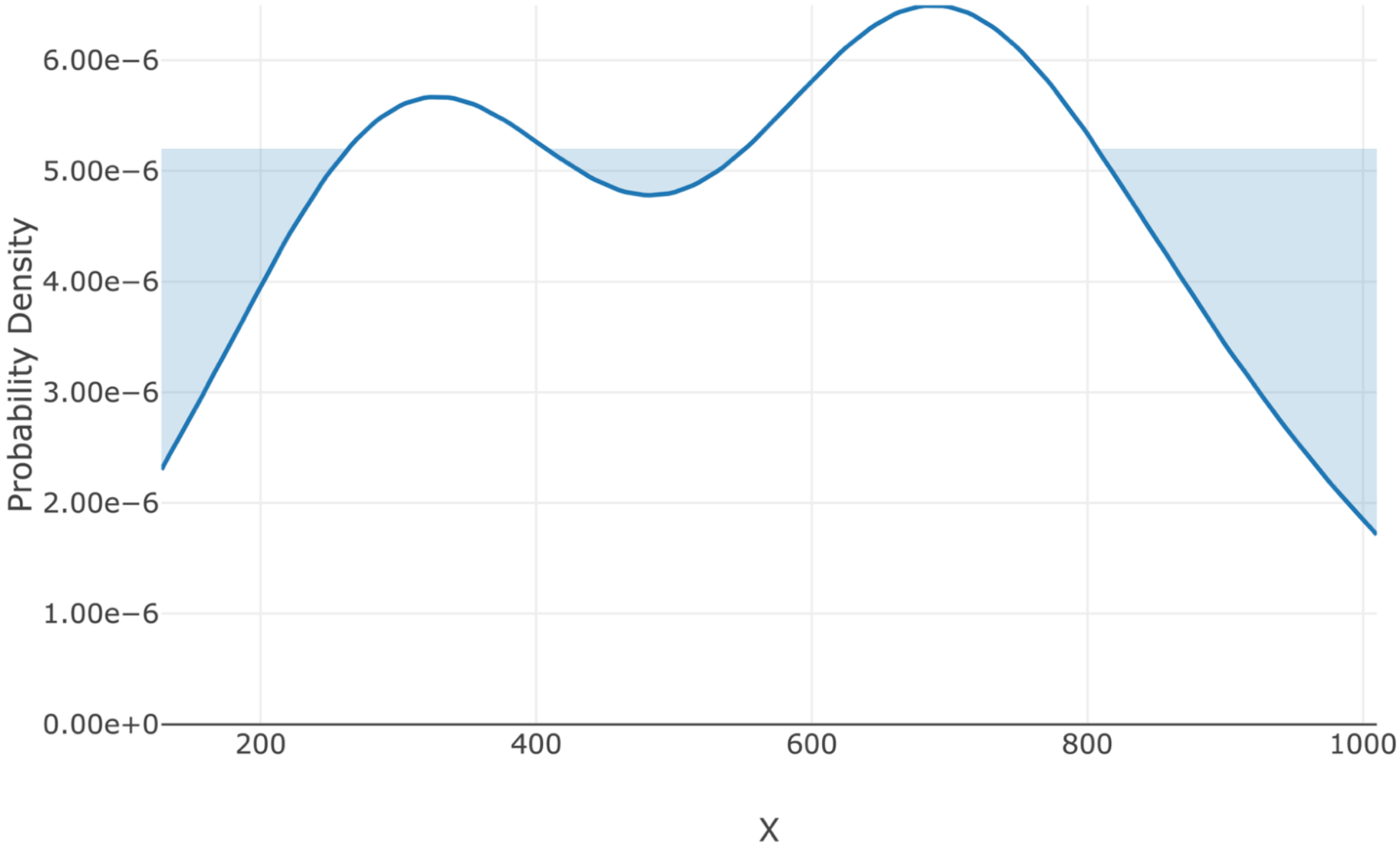


The mountain top that emerges (the right one) is the initial cluster detected

Probability Density of the Data (Showing Cross Section)

When we vary ε, a rough analogy is that we're getting rid of a flood

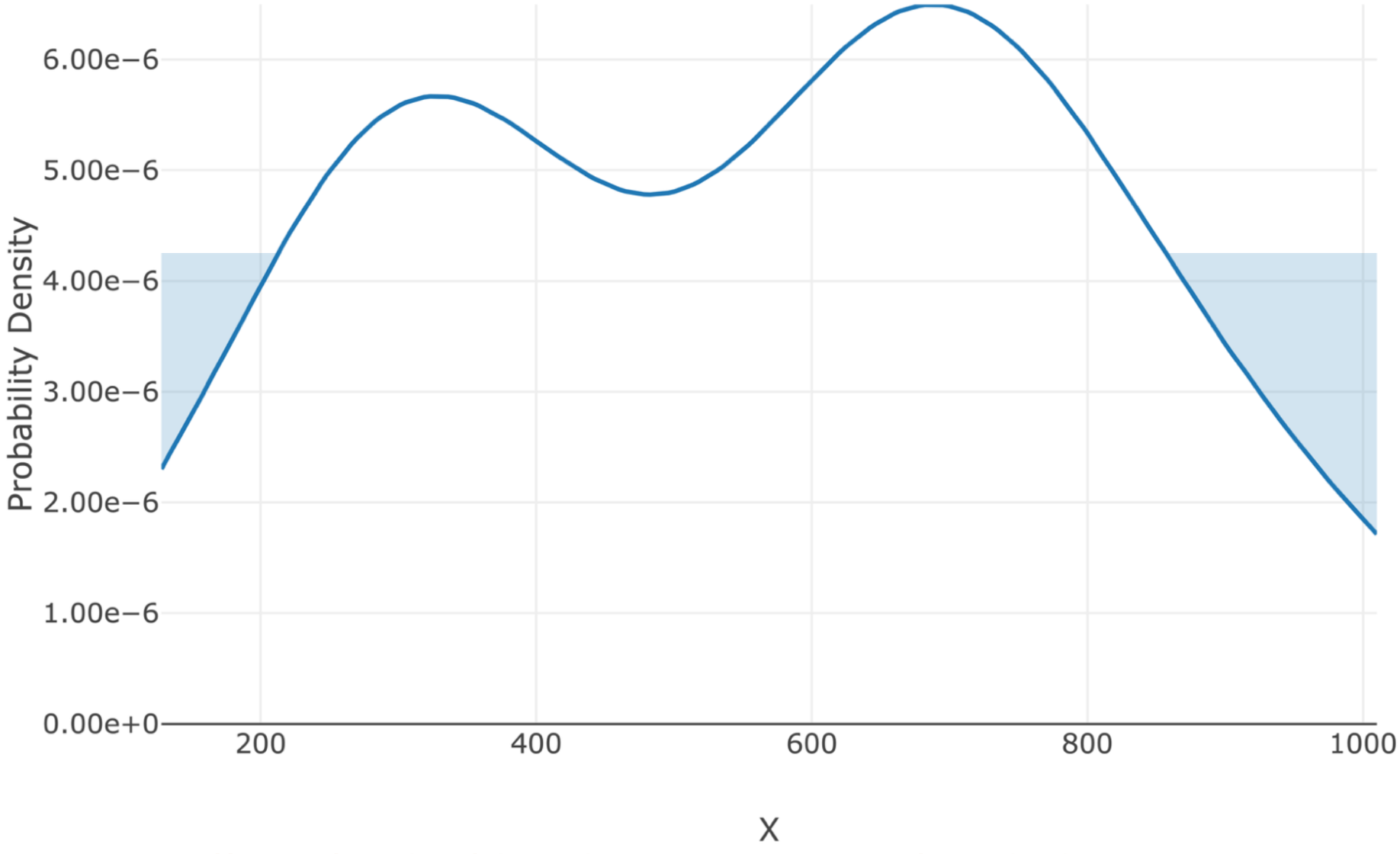
When ε increases, we are *lowering the water level*



Now it appears as if there are two islands (above water), corresponding to two clusters detected

Probability Density of the Data (Showing Cross Section)

Notice: right side mountain lasted longer as a cluster than the left side mountain
→ HDBSCAN considers the right side mountain to be more stable



Eventually, it looks like there's just 1 island above water,
so that the two clusters have been merged into a single cluster

HDBSCAN High-Level Ideas

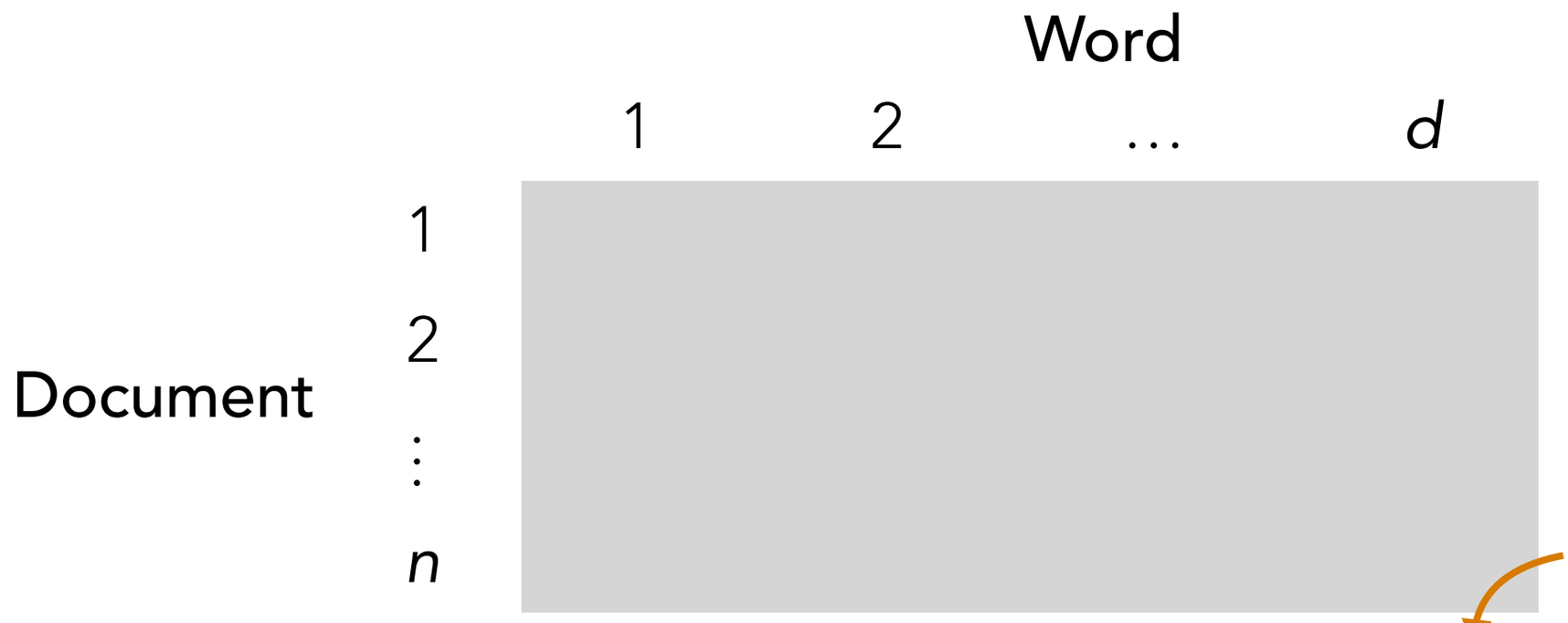
- Behaves as if we run DBSCAN across all possible ε
- HDBSCAN figures out how stable different clusters are
  (rough intuition: as water level drops, a cluster is more stable if it lasts longer across different water levels before it gets merged)
- HDBSCAN has a parameter `min_cluster_size` that basically tries to find the most stable clusters that have number of points in them at least `min_cluster_size`
- HDBSCAN has a parameter `min_samples` which is like DBSCAN's
- HDBSCAN does have other parameters too
  (but don't worry about these other ones for this class)

HDBSCAN

Demo

An Alternative Feature Vector Representation for Text: TF-IDF

Intuition: words that appear in more documents are likely less useful (same intuition as stop words!) — let's *downweight* these words!



Term frequency (TF)

i-th row, *j*-th column: # times doc word *j* appears in doc *i*

multiply TF by $\log \frac{1}{\mathbb{P}(\text{document mentions word } j)}$

$= \log \frac{n}{\#\text{ documents that mention word } j}$

Hack (additive smoothing): can add 1 to numerator & 1 to denominator

An Alternative Feature Vector Representation for Text: TF-IDF

Intuition: words that appear in more documents are likely less useful (same intuition as stop words!) — let's *downweight* these words!

There are many TF-IDF variants! (Lots of hacks!)

Term frequency (TF)

i-th row, *j*-th column: # times doc word *j* appears in doc *i*

$$\times \left[1 + \log \frac{n+1}{\#\text{ documents that mention word } j + 1} \right]$$

Default TF-IDF weighting in sklearn

sklearn's default behavior further normalizes each row to have Euclidean norm 1

An Alternative Feature Vector Representation for Text: TF-IDF

Demo

Is clustering structure enough?

Clustering methods typically assign each data point to a single cluster

(as we just saw, HDBSCAN/DBSCAN also declares some data points to be noise)

In reality, a data point could have “mixed” membership and belong to multiple “clusters”

For example, for news articles, possible topics could be *sports*, *medicine*, *movies*, or *finance*

A news article could be about *sports* and also about *finance*

How do we model this?

Topic Modeling:
Latent Dirichlet Allocation (LDA)

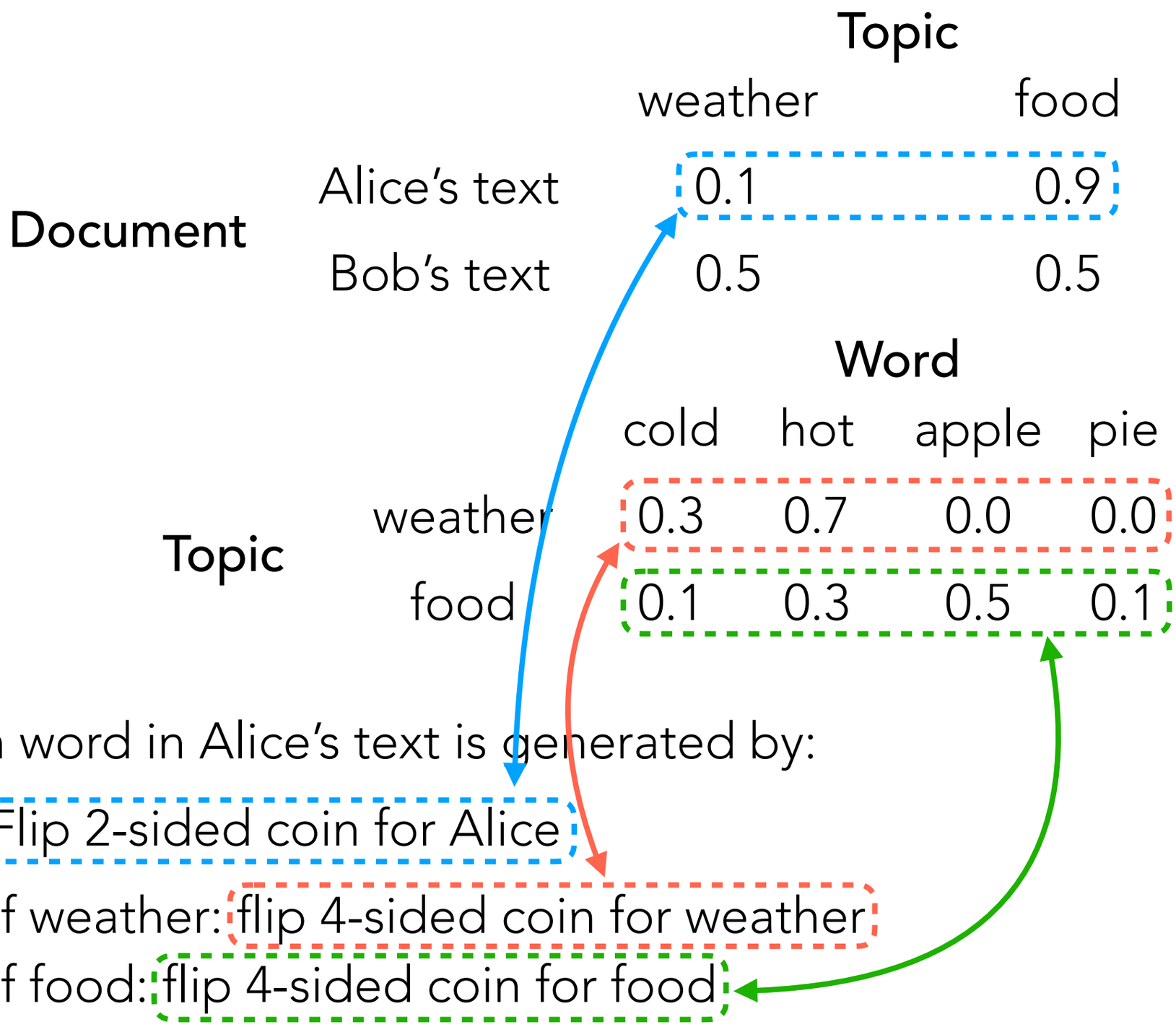
- A generative model

- Input: "document-word" matrix, and pre-specified # topics *k*

		Word			
		1	2	…	*d*
Document	1	Either TF table or TF-IDF table			
	2				
	⋮				
	n				

- Output: what the *k* topics are (details on this shortly)

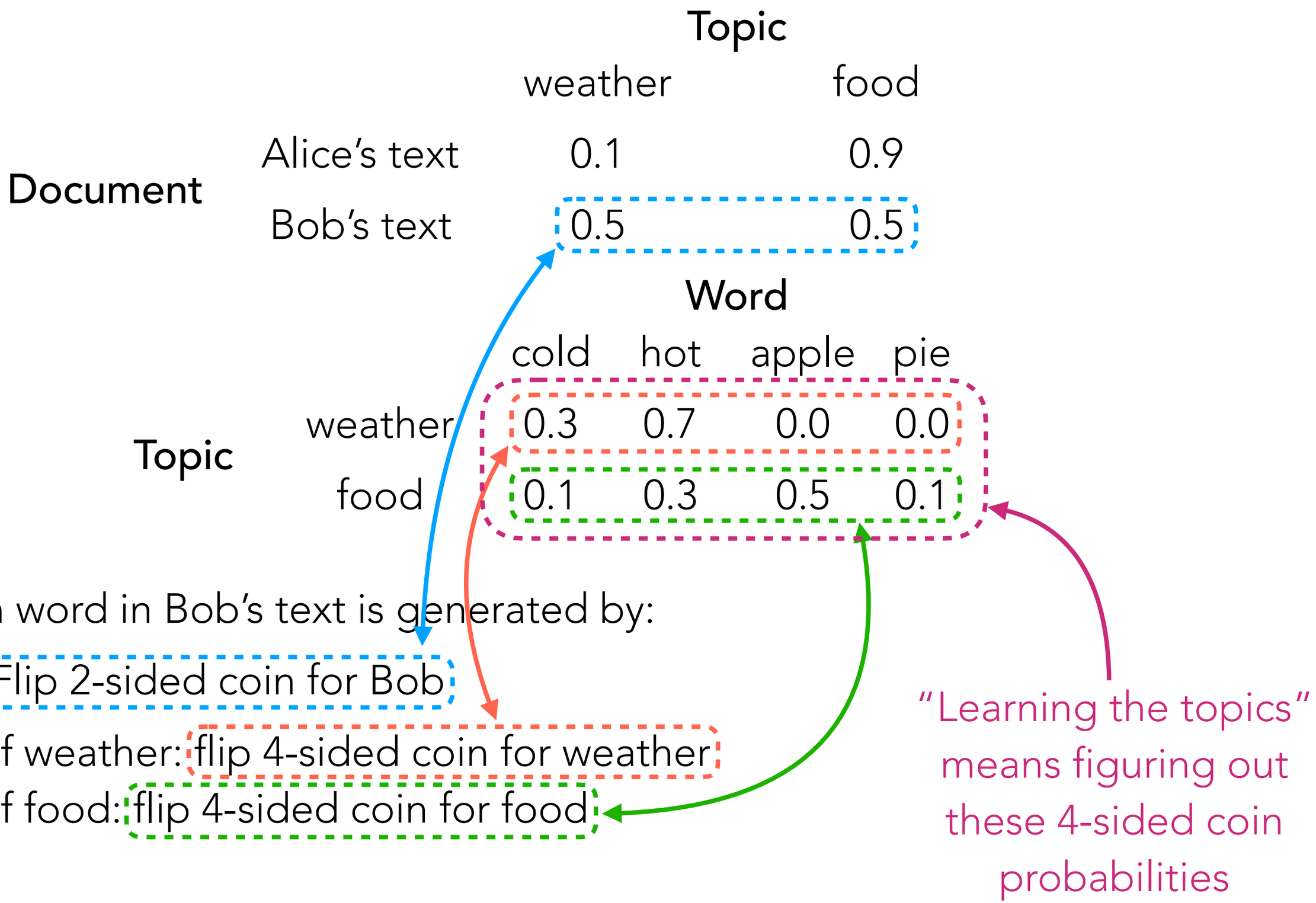
LDA Generative Model Example



Each word in Alice's text is generated by:

1. Flip 2-sided coin for Alice
2. If weather: flip 4-sided coin for weather
   If food: flip 4-sided coin for food

LDA Generative Model Example

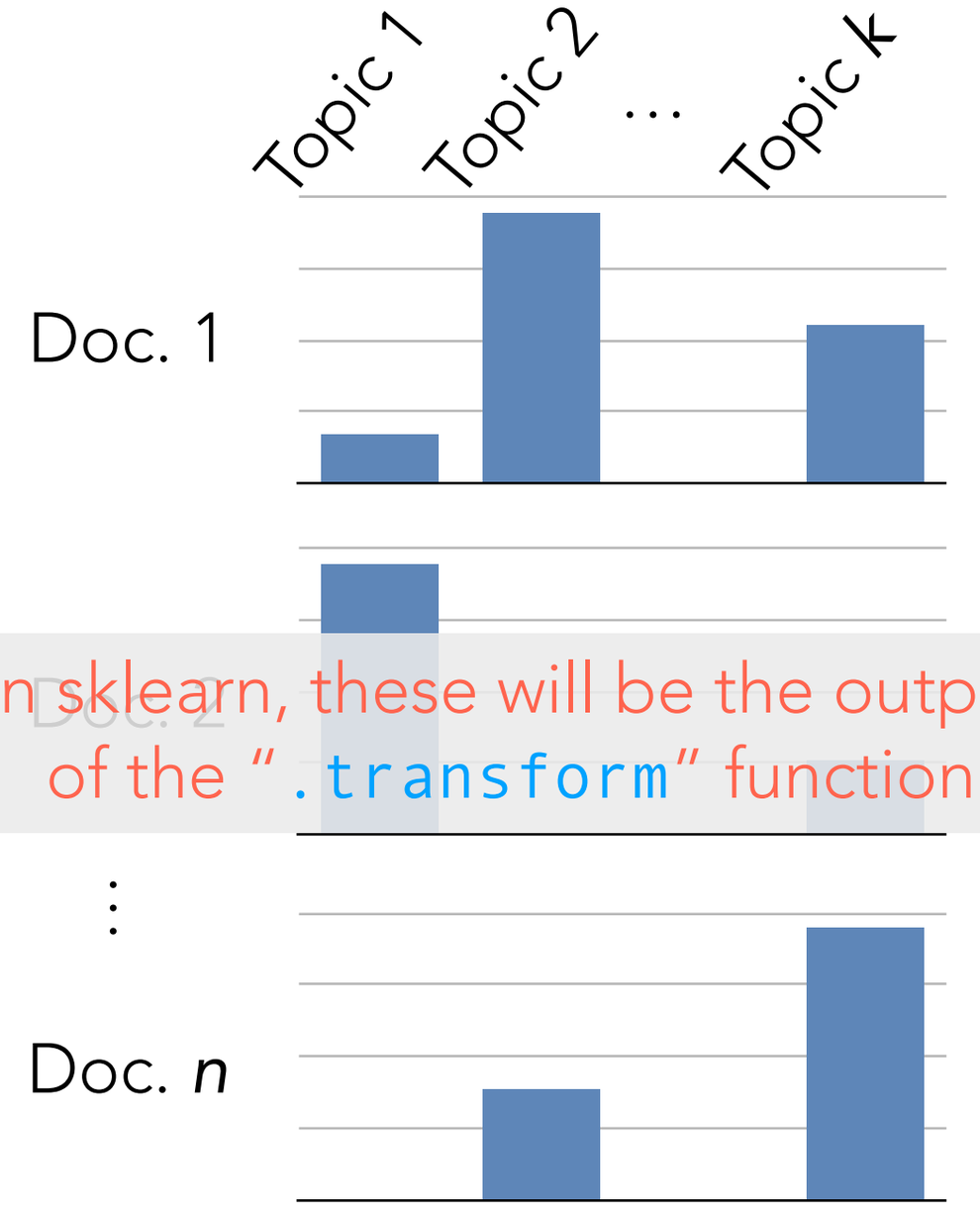


Each word in Bob's text is generated by:

1. Flip 2-sided coin for Bob
2. If weather: flip 4-sided coin for weather
   If food: flip 4-sided coin for food

"Learning the topics" means figuring out these 4-sided coin probabilities

LDA Generative Model



In sklearn, these will be the output of the "`.transform`" function

Goal: Learn these distributions

In sklearn, these will be in "`.components_`"

LDA models each word in document *i* to be generated as:

1. Randomly choose a topic *Z* (use topic distribution for doc *i*)
2. Randomly choose a word (use word distribution for topic *Z*)

Topic Modeling:
Latent Dirichlet Allocation (LDA)

- A generative model

- Input: "document-word" matrix, and pre-specified # topics *k*

		Word			
		1	2	…	*d*
Document	1	Either TF table or TF-IDF table			
	2				
	⋮				
	n				

- Output: the *k* topics' distributions over words