

Clustering in Software Engineering

CS454 AI-Based Software Engineering

Shin Yoo | COINSE@KAIST

Clustering

- Unsupervised machine learning technique that groups items based on their similarities
 - Since similarity is inherent in the target objects, clustering does not require supervision. However, we do have to provide the similarity (or distance) metric!
 - From application perspective, clustering is useful when you need a fewer number of **representatives** of your large dataset: the reason why can be diverse.

Major Types of Clustering

- Centroid-based: associate data to a set of centre points (i.e., centroids)
- Hierarchical: order gradual merges (usually expressed in trees called dendrogram) and pick a stopping point
- Density-based: find areas that are densely populated, label outliers as noise
- Distribution-based: try to represent the data as a set of distributions (e.g., multiple Gaussians)

Centroid-Based Clustering

K-Means

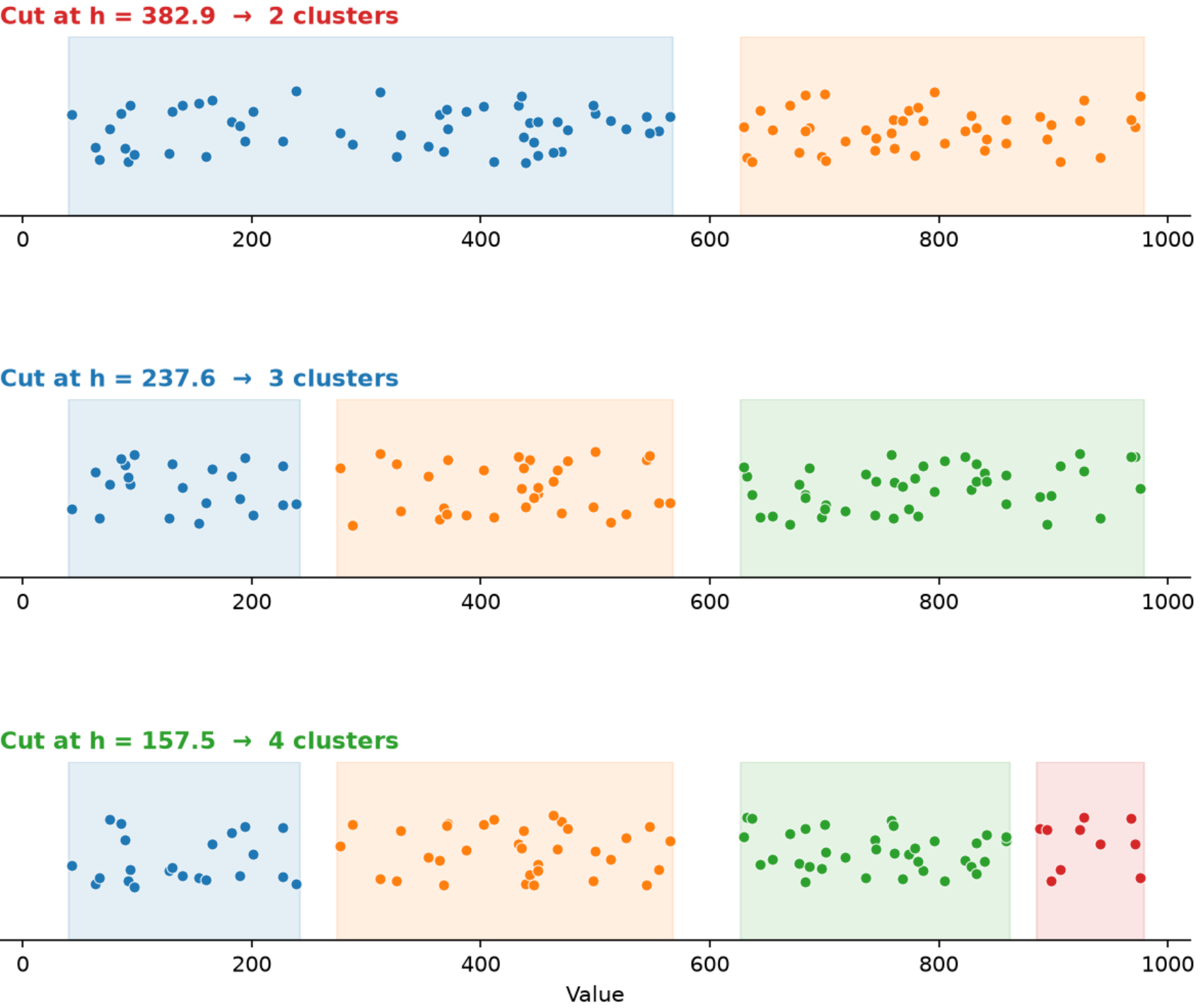
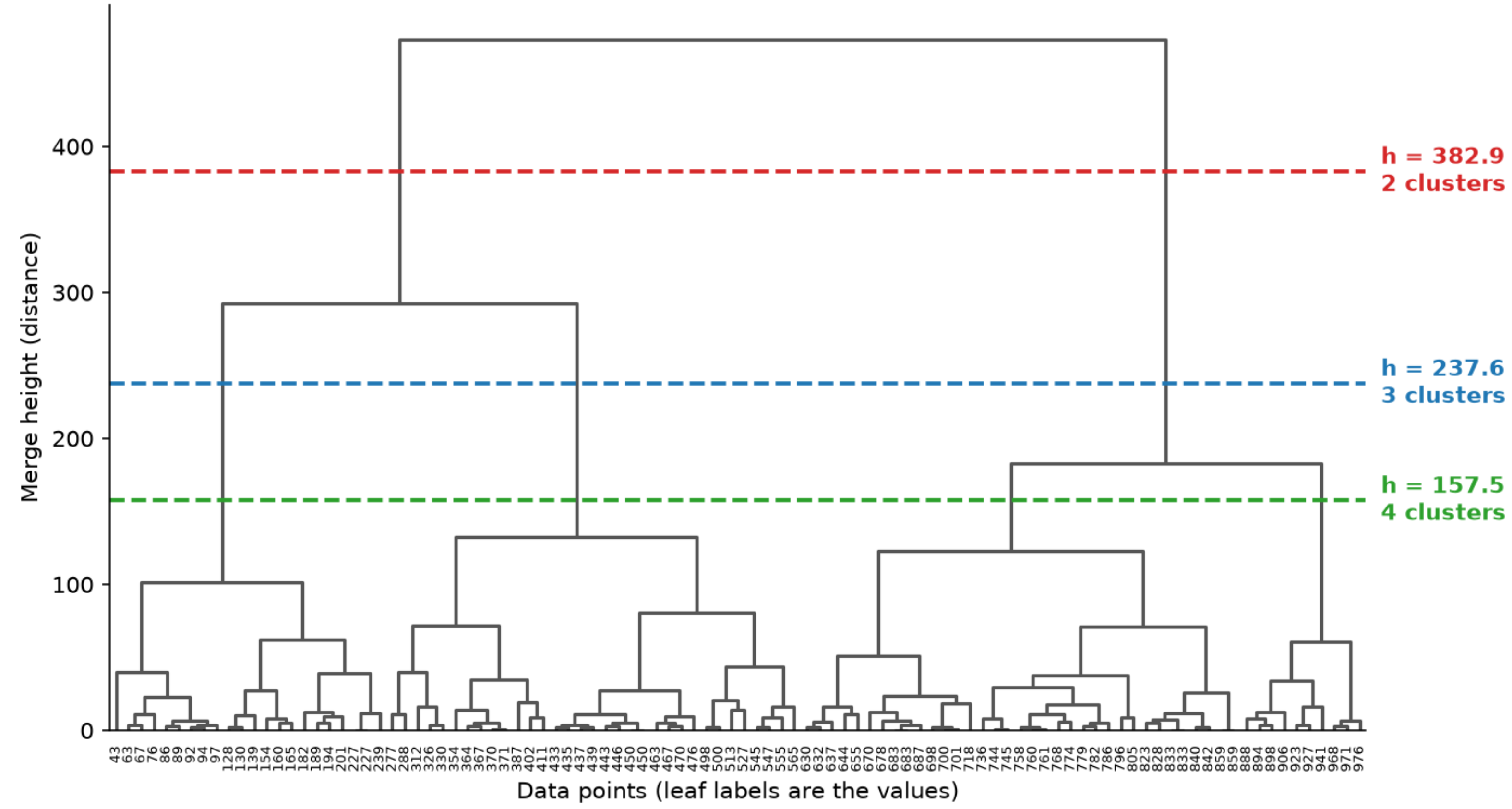
- Decide k , i.e., the number of clusters you want
- Randomly pick k points from your dataset: they now represent k clusters as centroids.
- Assign the rest of points to the closest cluster (i.e., one whose centroid is the closest).
- For each cluster, now set the centroid as the average of all datapoints.
- Repeat until centroids do not move anymore, or the budget runs out.

Hierarchical Clustering

Agglomerative Hierarchical Clustering (i.e., bottom-up)

- Start with each datapoint in their own cluster of size 1
- Each cluster is merged with the closest cluster (various linkage options)
- Repeat until you merge everything to one cluster: you have dendrogram
- Pick a height, and cut the dendrogram: you have clusters

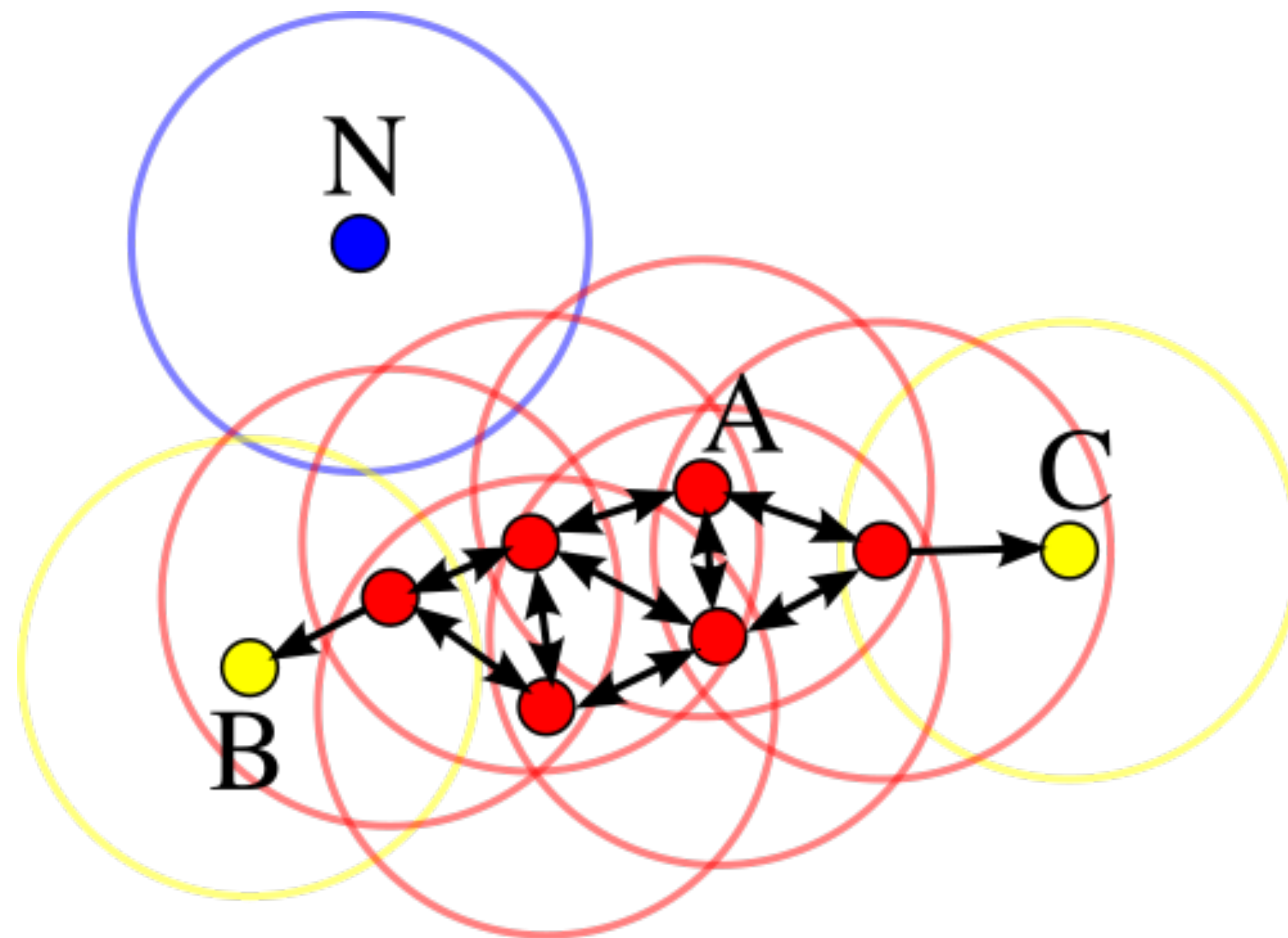
Dendrogram — agglomerative clustering, average linkage, $d(a, b) = |a - b|$ (n = 100)



Density-based Clustering

DBSCAN (Density-based Spatial Clustering of Applications with Noise)

- Given ϵ , radius, and minPts, number of points:
 - A point p is a core point if at least minPts other points are within ϵ
 - A point q is *directly* reachable from p if q is within ϵ of p
 - A point q is reachable from p if q is within ϵ of p if there is a chain of directly reachable points that connect p and q , i.e., $(p, p_1), (p, p_2), \dots, (p_n, q)$
 - Any point that is not reachable from any other point is an outlier
- A cluster is all points reachable from core points



- Core Points
(i.e., there are 4 others within ϵ)
- Non-core but Reachable Points
(i.e., fewer than 4 others within ϵ)
- Outliers
(i.e., not reachable from cluster)

ϵ : circle radius, minPts: 4

Distribution-based Clustering

Gaussian Mixture Model

- Think of GMM as a softer version of K-means: instead of centroid, we maintain k Gaussian distributions - GMM is Expectation-Maximisation.
 - Expectation step: each datapoint calculates how likely it came from a specific Gaussian distribution out of the current k
 - Maximisation step: each distribution looks at the data - if a datapoint has a high probability of belonging to itself, the centre of the distribution is shifted towards the datapoint (i.e., weighted average to get centroid)
- In the end, we get a set of Gaussian distributions that collectively best describes the given datapoints; membership of individual kernel is clustering.

Fundamental Question in Clustering

How many clusters are (should be) there?

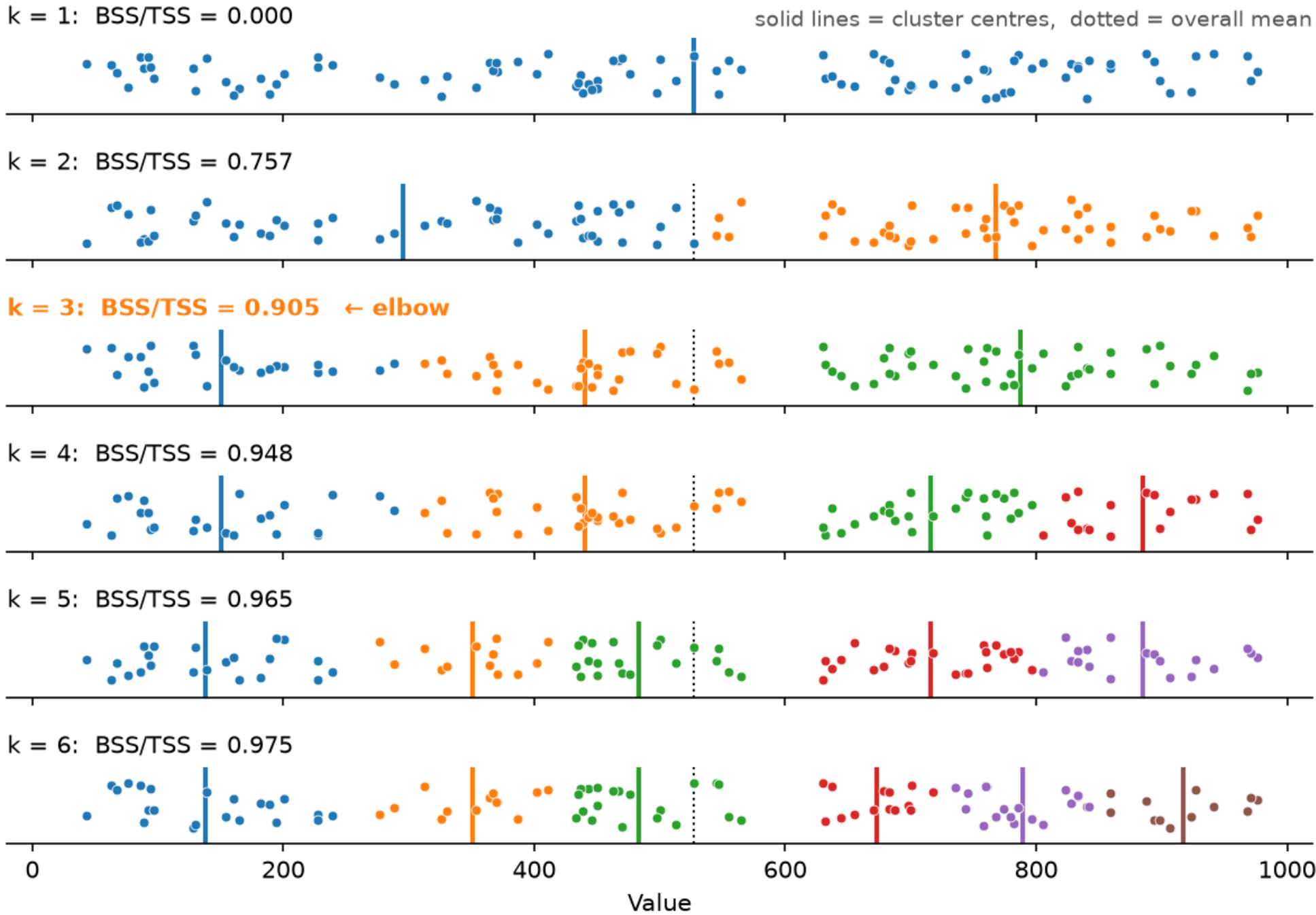
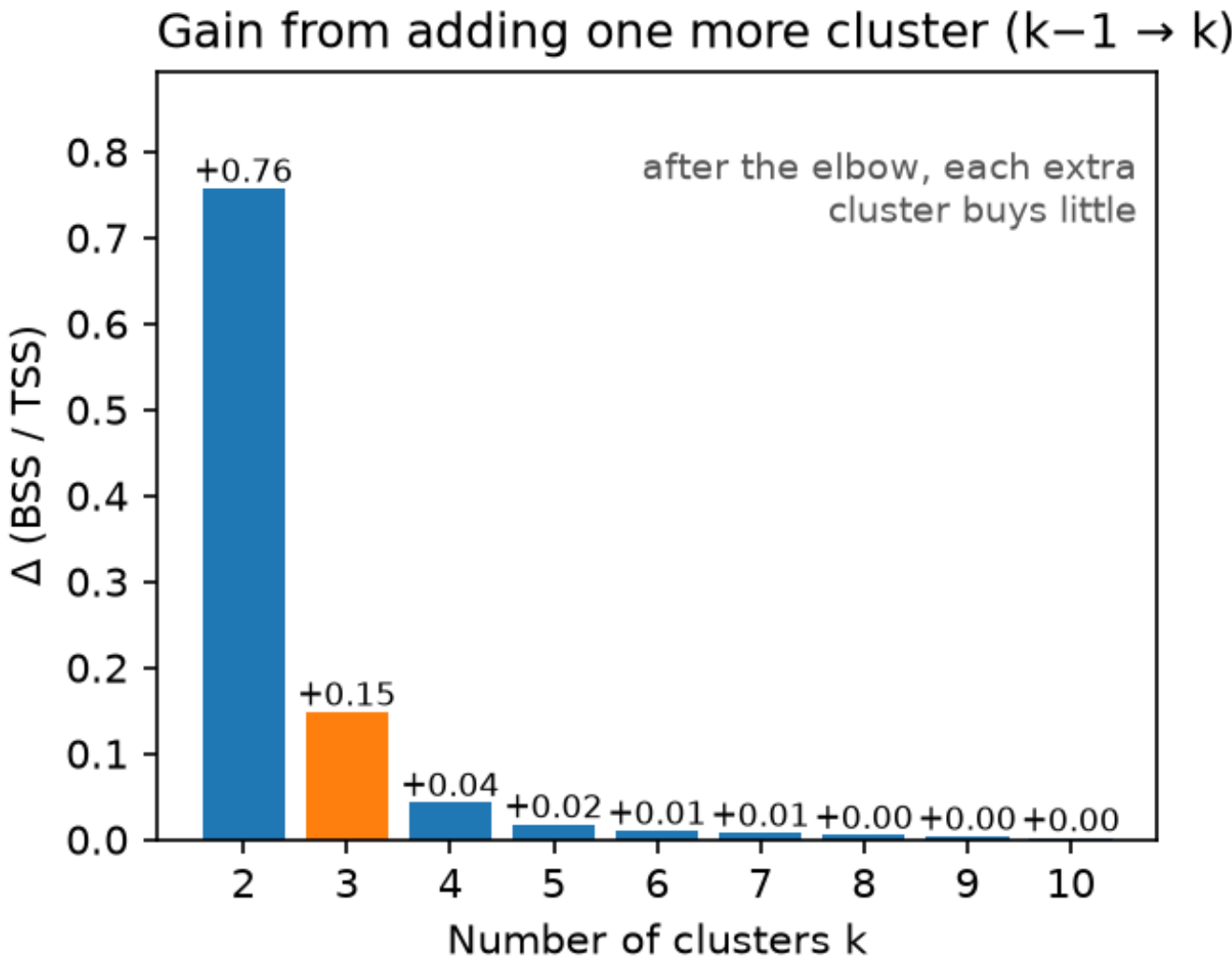
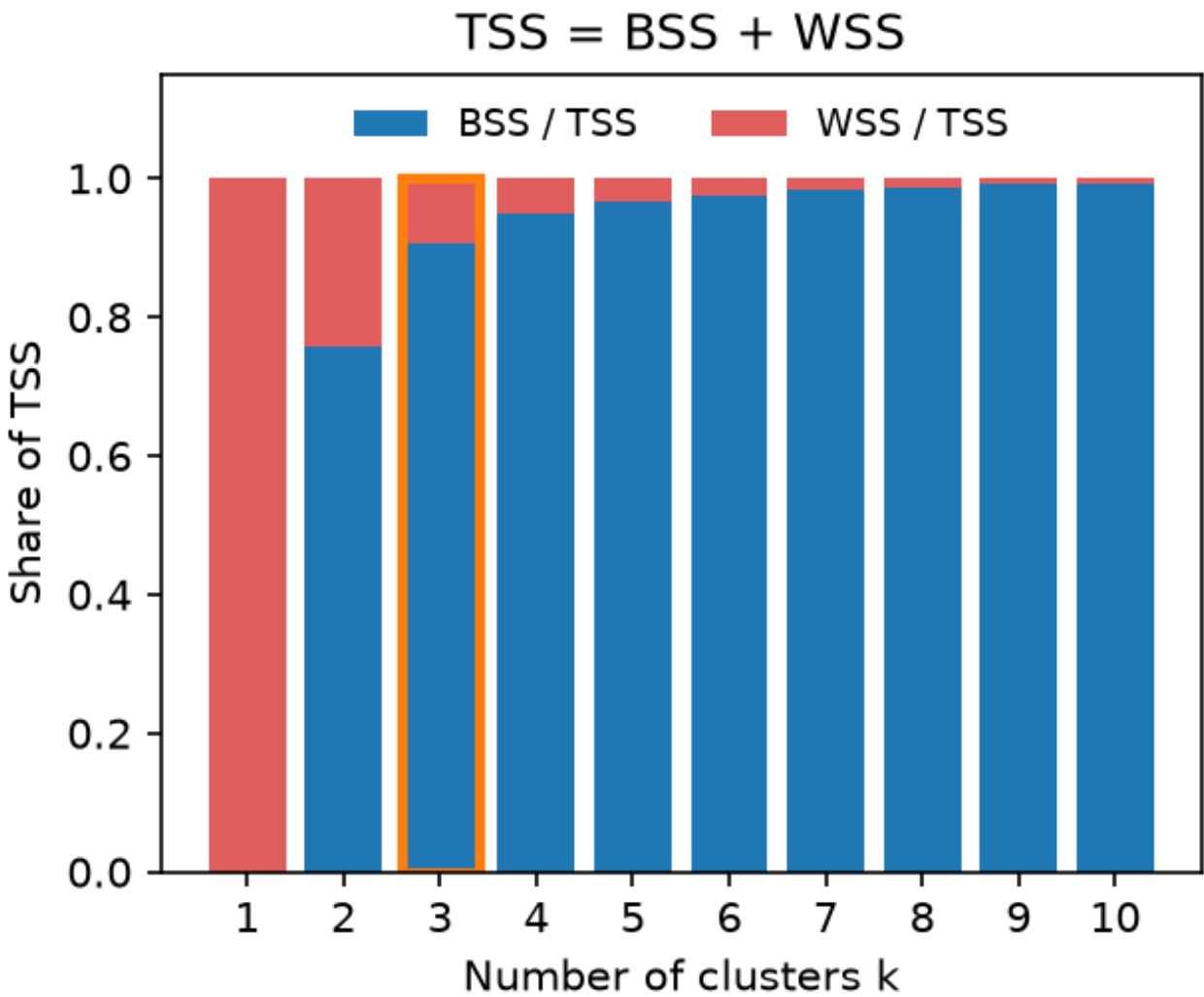
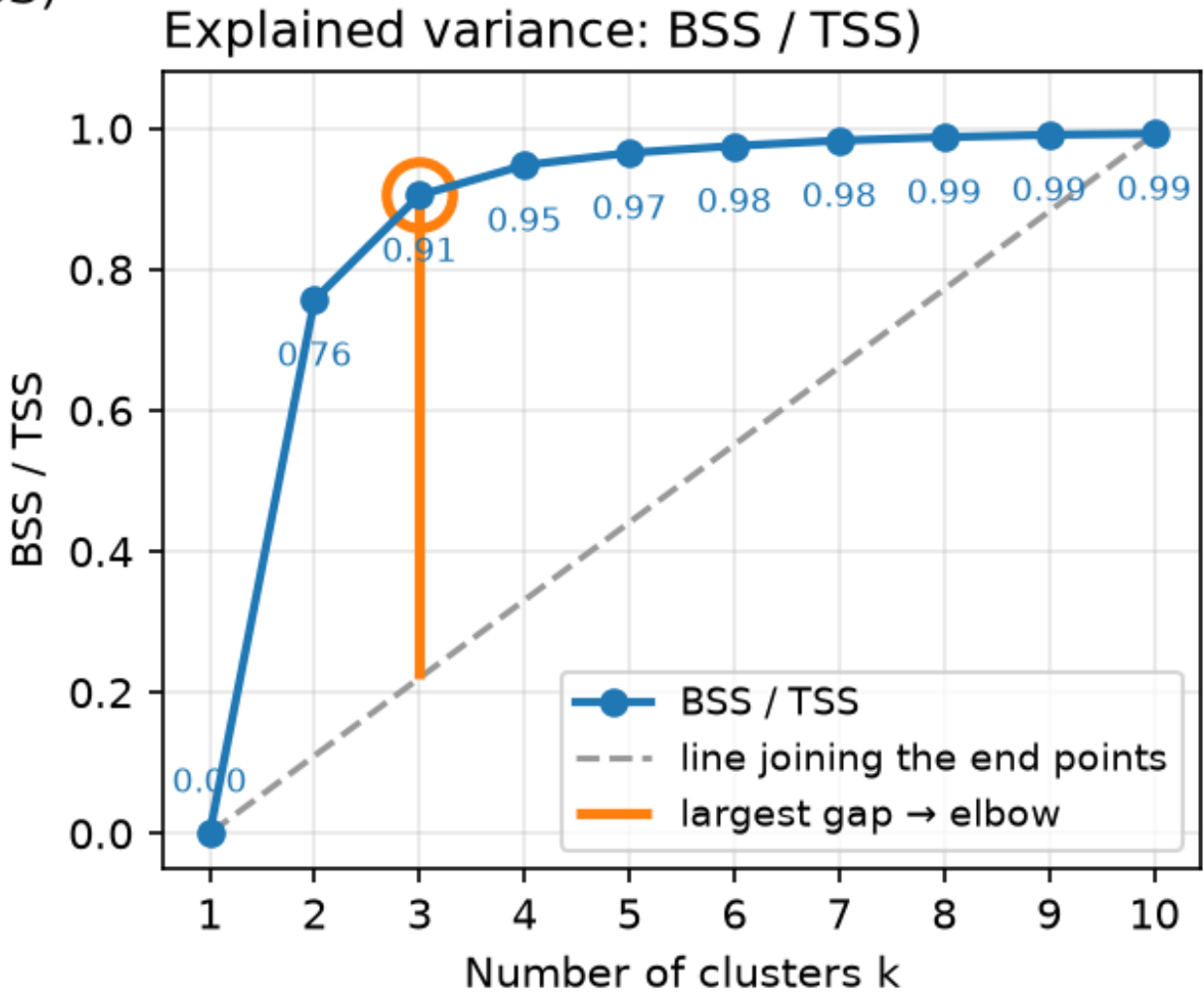
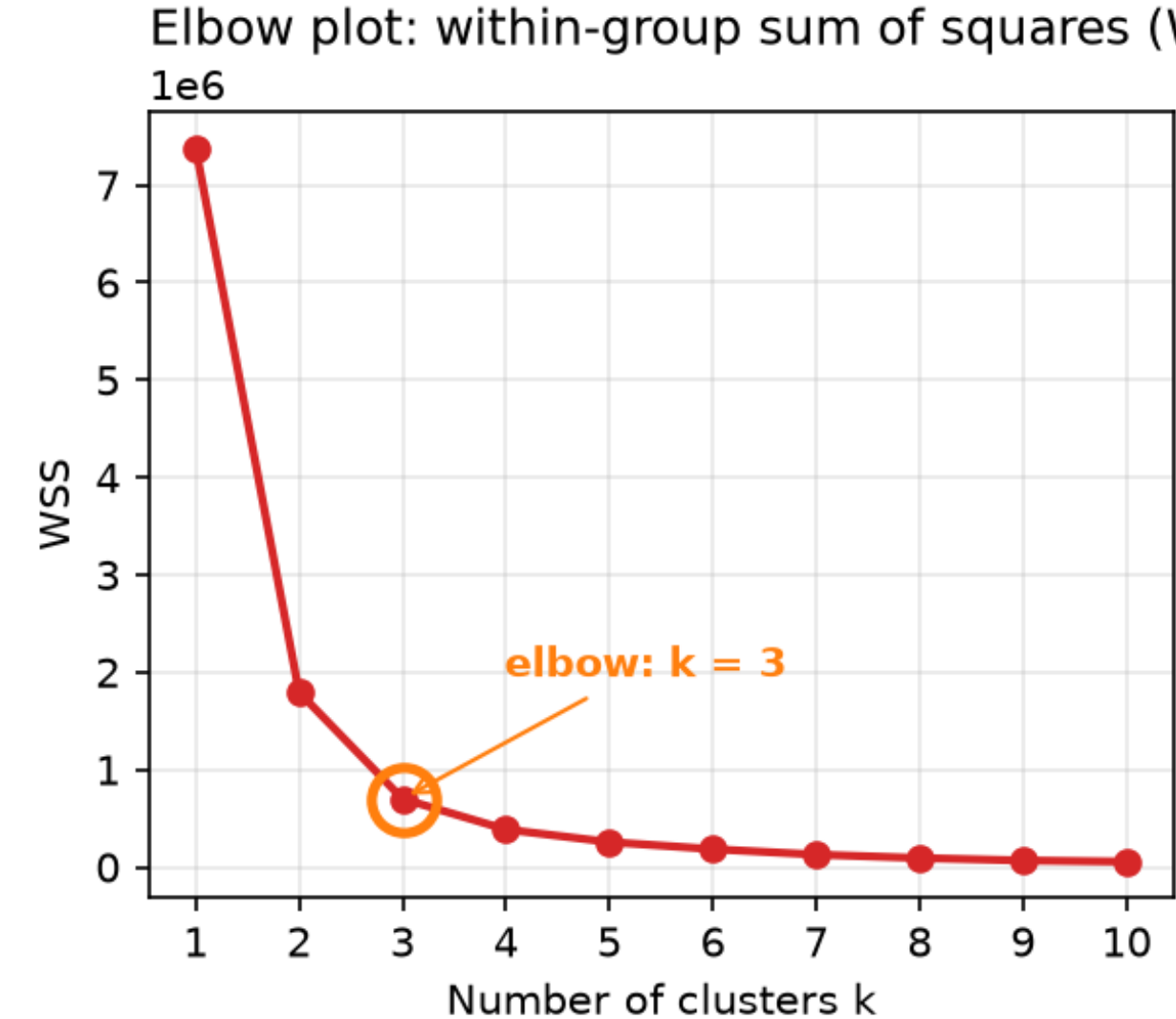
- Since clustering is unsupervised, we have no idea: what is the most natural (i.e., the best supported by evidence) number of groupings?
 - K-means and Distribution-based clustering explicitly require k to be specified
 - Hierarchical clustering does not require k ; however, to use it, you are often eventually requested to cut the dendrogram.
 - Density-based clustering does not require k ; however, your choice of ϵ and minPts will effectively determine k - how do we know if this is ideal? 😎
- Success of many practical applications depends on this question; there are a few heuristics: elbow, silhouette...

Elbow Method

How much of the data variance does my clustering explain?

- Each additional cluster adds decreasing amount of new *explanation*. An elbow point is where this diminishing return is first observed.
- The widely used measure of “explanation” is the ratio between WSS/BSS and TSS
 - Within-group Sum of Squares (WSS): sum of per cluster sum of the square of deviations, i.e.,
$$\sum_{c_i \in C} \sum_{x_j \in c_i} (\bar{x}_{c_i} - x_j)^2$$
 - Between-group Sum of Squares (BSS): how much mean of each clusters deviates from the total mean, i.e.,
$$\sum_{c_i \in C} |c_i| (\bar{x}_{c_i} - \bar{x}_X)^2$$
 - Total Sum of Squares (TSS): sum of squares of deviations for all data, i.e.,
$$\sum_{x_i \in X} (\bar{x}_X - x_i)^2$$

k-means and the elbow method on 100 random integers in [0, 1000]



Silhouette Method

How cohesive are my clusters?

- For each datapoint $i \in C_i$, calculate the following two numbers.
 - Cohesion $a(i)$: average distance to all other points in the same cluster, i.e.,

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

- Separation $b(i)$: average distance to members of the nearest cluster, i.e.,

$$b(i) = \min_{j \neq i} \frac{1}{|C_j|} \sum_{l \in C_j} d(i, l)$$

Silhouette Method

How cohesive are my clusters when there are k of them?

- Silhouette score $s(i)$ for a single datapoint i : $s(i) = \frac{b(i) - a(i)}{\max(a_i, b_i)}$
- Silhouette Coefficient for a clustering with k clusters:
 - $\tilde{s}(k)$: average of $s(i)$ for all datapoints when there are k clusters
 - $SC = \max_k \tilde{s}(k)$
- We can set k to the value that maximises SC (i.e., the best separation and the tightest individual clusters).

Similarity/Distance Function

If the distance function is not meaningful, there is nothing to learn!

- If the entities are numeric, Euclidean distance is the natural choice.
 - For density-based clustering, numeric entities are also straightforward to model.
- For strings, Levenshtein distance (aka edit distance) is the standard: count the minimum number of single character edits (insertion, deletion, substitution) required to change string A to string B
 - Has been adapted to complex data types, e.g., tree edit distance (minimum number of node insertions, deletions, and substitutions)
- Increasingly, cosine similarities between embedding vectors are used, if there exists a good embedding model for the data

Dimensionality Reduction

A critically related statistical technique

- Humans tend to visualise things, including clusters, in 2D. Often we have to reduce the dimensionality down to 2 (or sometimes 3) to *visualise* groupings. Dimensionality Reduction is a big area in itself.
- Linear DR: tries to linearly project the datapoints into a lower dimension, while preserving the variance in the data as much as possible. Primary example would be Principal Component Analysis
- Non-linear DR: tries to find a lower dimension mapping of the data, while preserving the relative distances between individual points as much as possible. Primary example would be Multidimensional Scaling (MDS) or t-SNE.
- Note that some non-linear DR visualisation will highlight local structures.

Applications of Clustering in Software Engineering

We will investigate individual cases via publications

- **Are these the same thing?:** Failure Deduplication (*Igor*, CCS 2021)
- **Can we scale this down?:** Human-guided Test Prioritisation (*CBR+AHC*, ISSTA 2009)
- **Can we model input density?:** Pre-generation Test Adequacy for LLMs (*Clotho*, FSE 2026)

Failure Deduplication

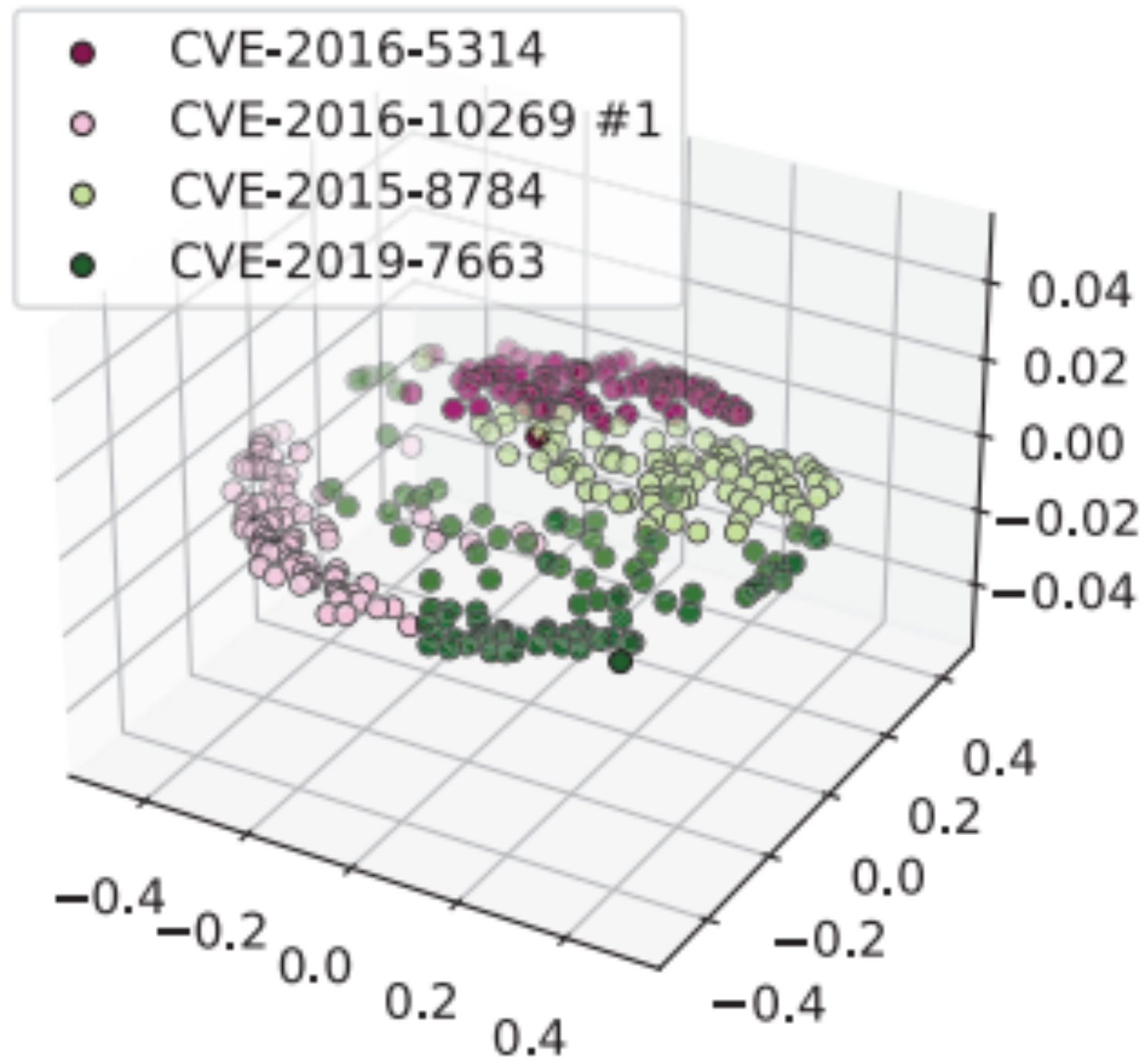
Is this failure the same as that one? :)

- Sometimes we need to know whether a newly reported failure is 1) entirely new, or 2) something already known.
 - This is typically because you learnt of this new failure, without necessarily knowing its root cause.
 - Fuzzing can find the same vulnerability via multiple paths: each path will be reported with different test inputs that crash the target program.
 - Users reported crashes: you can get unique reports that stem from the same root cause.

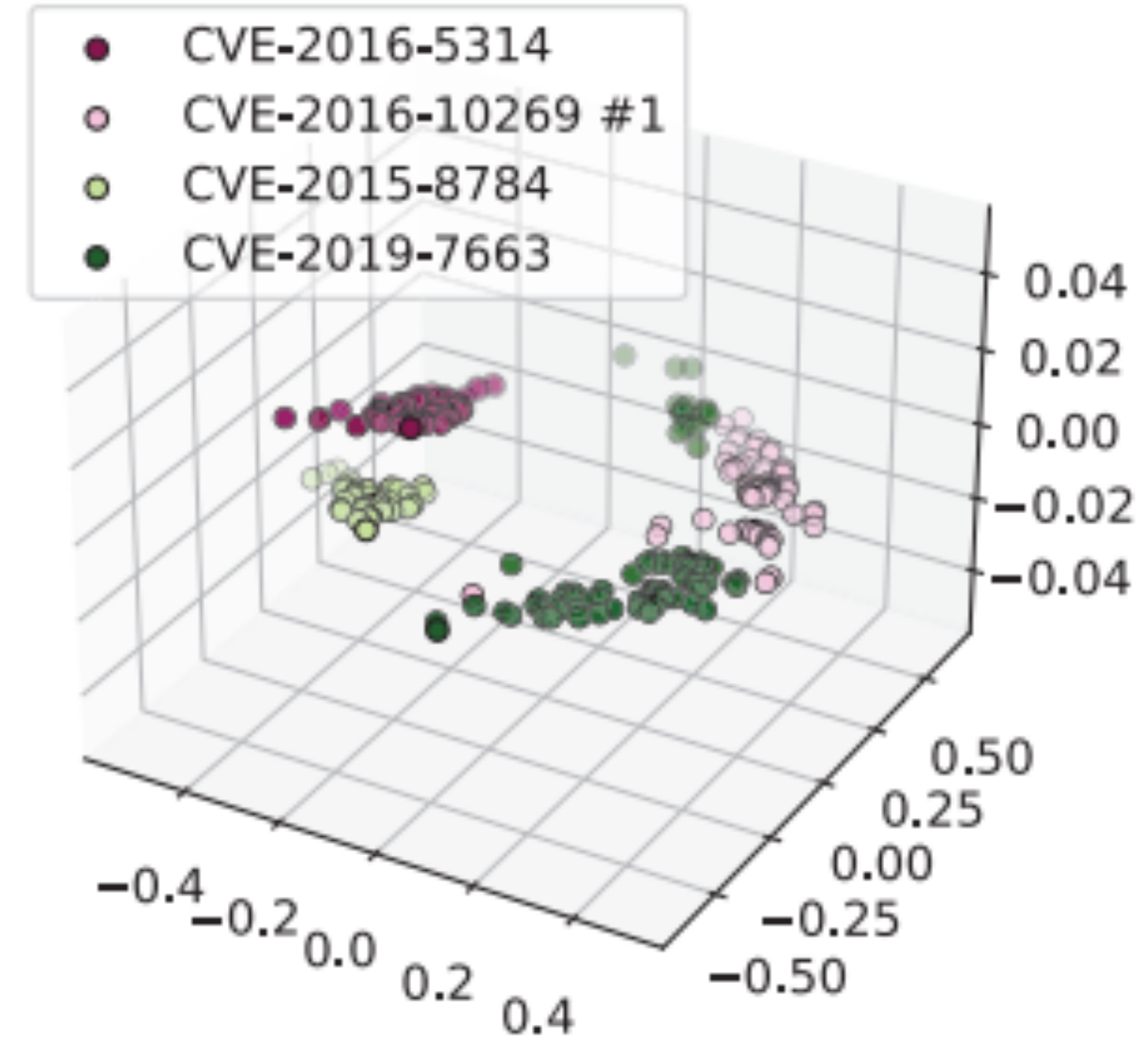
Igor: Crash Deduplication through Root Cause Clustering

CCS 2021 (<https://dl.acm.org/doi/pdf/10.1145/3460120.3485364>)

- Once a crash is found, Igor tries to find the minimum code trace that results in the same crash, i.e., shortest bug-triggering path
- Given multiple crashes, we get multiple shortest bug-triggering paths, which is a set of graphs
- Igor uses *Weisheiler-Lehman* subtree graph kernel to capture graph representations, which captures local structures into feature vectors. Similarity is computed as inner product of two vectors.
- We can do either K-means or hierarchical clustering.



(a) Original distribution.



(b) Distribution after reduction.

Figure 7: Data distribution before and after reduction.

Test Prioritisation and Clustering

First, a little bit of background

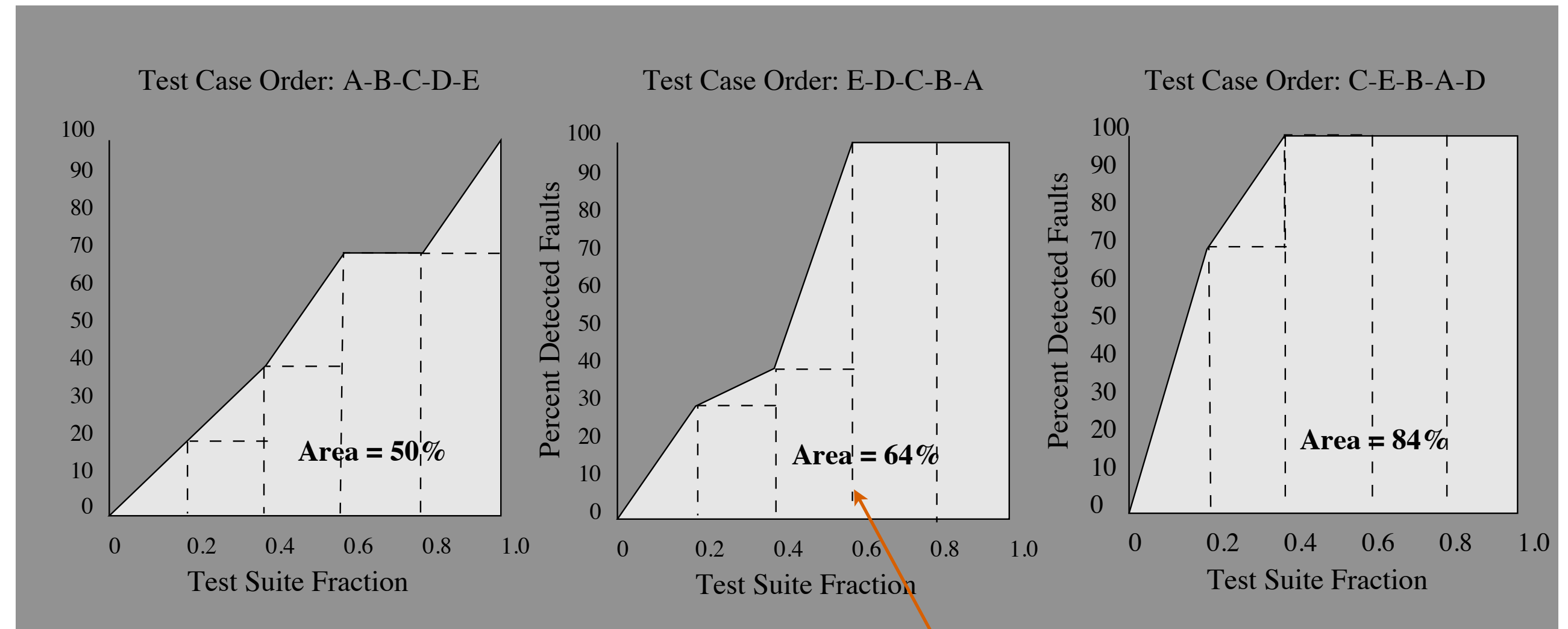
- Suppose you have 100 test cases, 99 of which pass and 1 fails.
 - Ideally, you want to execute the 1 failing test case first: this is the best use of your time.
 - The worst scheduling is to execute the failing test last: you could have spent more time fixing the last test that fails.
- This is the test case prioritisation problem: since you cannot know which test will fail in advance, you would like to find the best approximate heuristic.
 - The literature is dominated by coverage, i.e., you want to cover as much as possible, as early as possible.

Average Percentage of Fault Detection (APFD)

Elbaum, Malishevsky, and Rothermel, ISSTA 2000 (<https://dl.acm.org/doi/10.1145/347324.348910>)

- Measures how quickly your prioritisation detected the faults.

	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9
t0	x				x					
t1	x				x	x	x			
t2	x	x	x	x	x	x	x			
t3					x					
t4								x	x	x

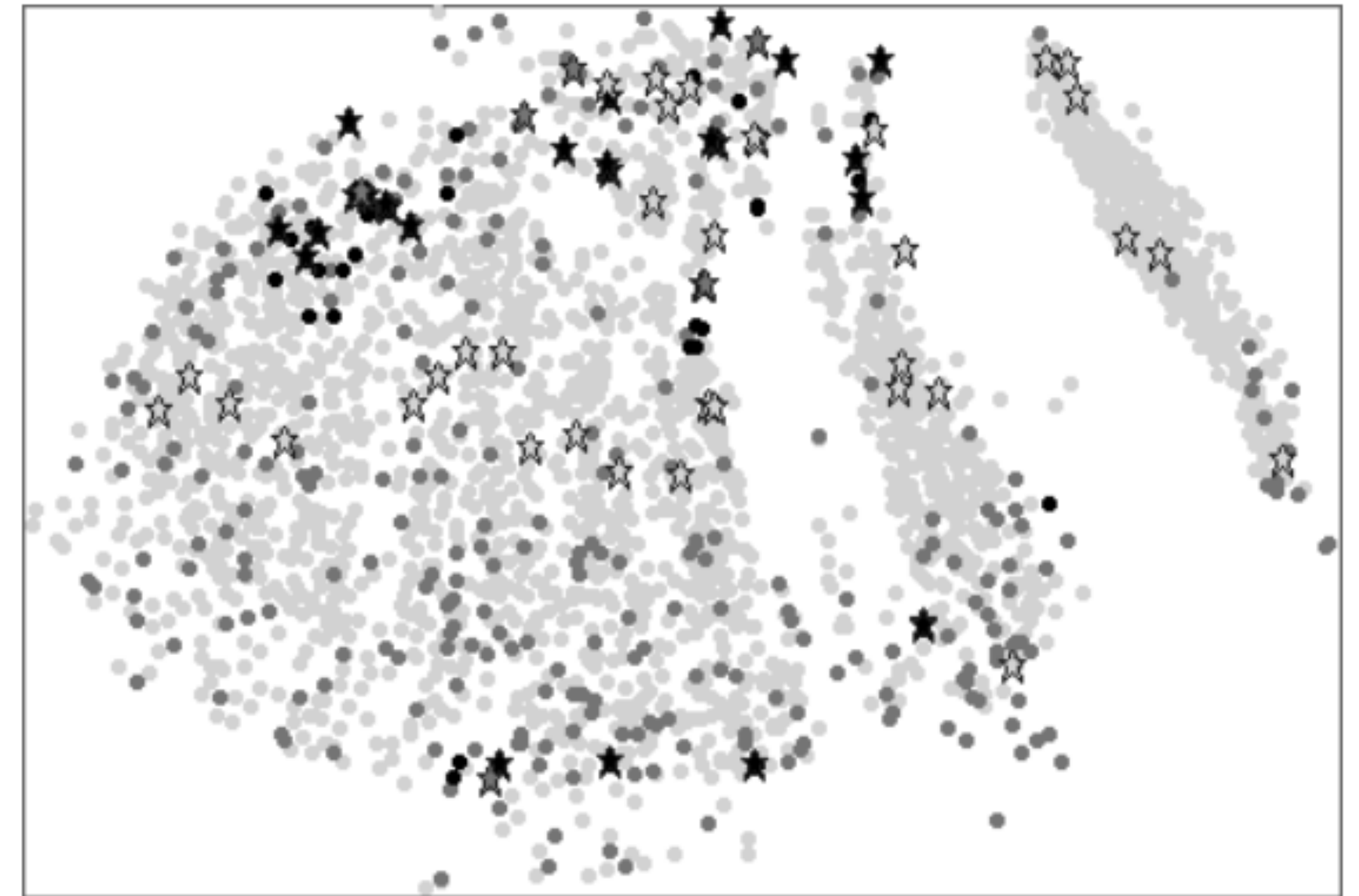


Intuitively, the metric measures the area beneath.

Distribution of Program Failures

Dickinson, Leon & Podgurski, FSE 2001

- Given a distance metric that can quantify distances between test executions, you can observe clusters of failures
- They show that hierarchical clustering + per-cluster sampling is an efficient way of identifying unique failures



Failures of GCC

<https://dl.acm.org/doi/10.1145/503209.503243>

Distribution-based Test Prioritisation

Leon & Podgurski, ISSRE 2003 (<https://ieeexplore.ieee.org/document/1251065>)

- If you can cluster test cases based on their execution profiles (e.g., coverage), you can get test ordering in various ways:
 - Pick only one from each cluster, repeat
 - Pick the most unusual one first, since it is a very unique execution
 - Pick the most different one as the next

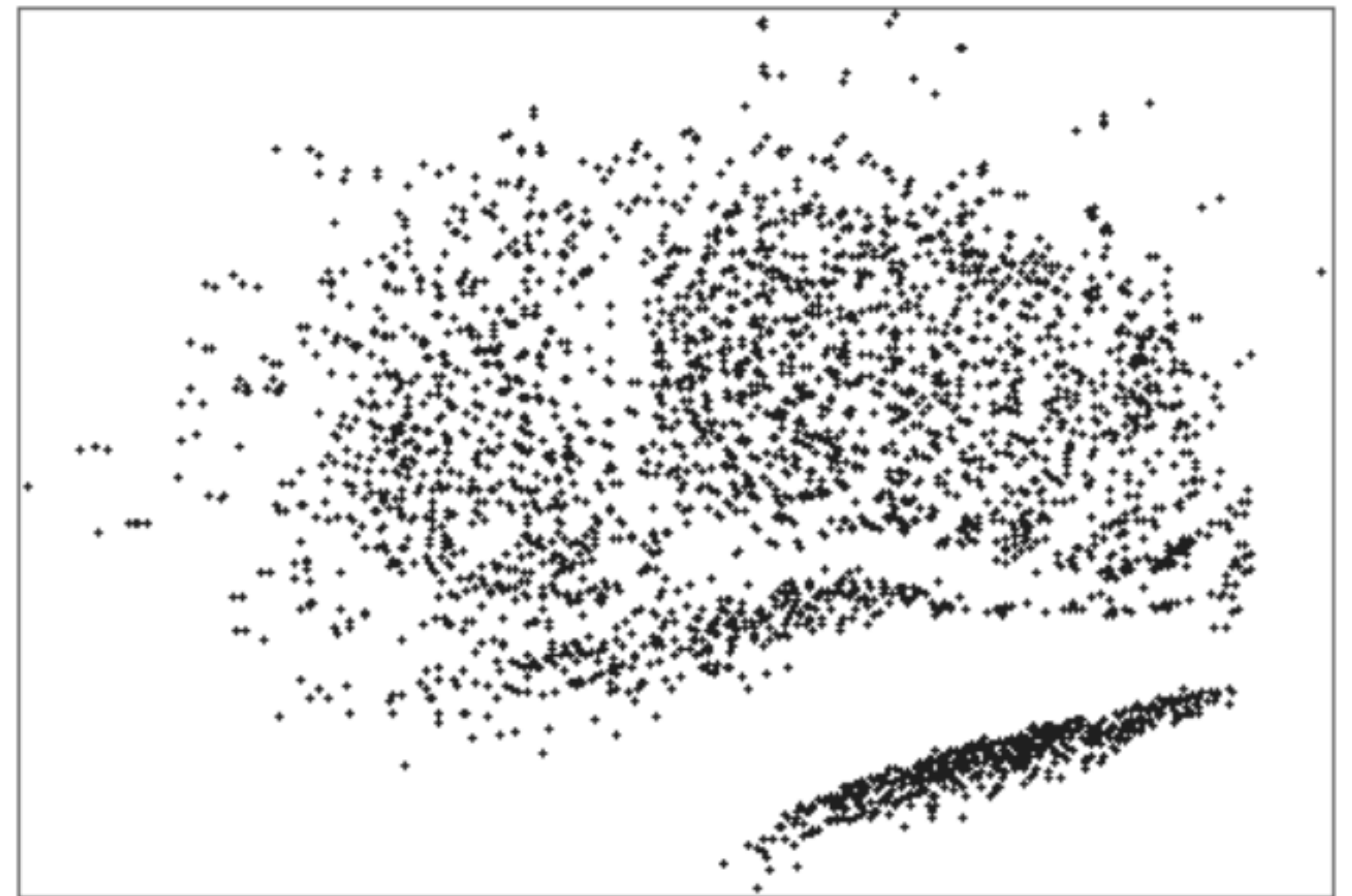


Figure 1: Multidimensional Scaling display of the GCC dataset. Points represent test executions.

Case-Based Ranking (CBR) for Prioritisation

Tonella, Avesani, and Susi, ICSM 2006 (<https://ieeexplore.ieee.org/document/4021329>)

- Case-Based Ranking: a machine learning technique that ranks items based on a number of pairwise comparisons, i.e., “*which do you prefer between A and B?*”
 - Essentially you permeate the ordering while trying to minimise the number of violations to the given pairwise comparisons
- This has been applied to test prioritisation: ask the tester for a set of comparisons (which test is more important, A or B?), then rank them.
 - Pros: can incorporate human domain expertise
 - Cons: expensive, and humans can be inconsistent after a while (cognitive fatigue)

Clustering Tests to Incorporate Human Knowledge

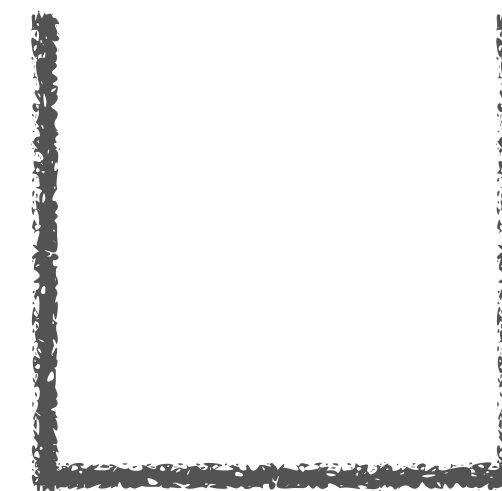
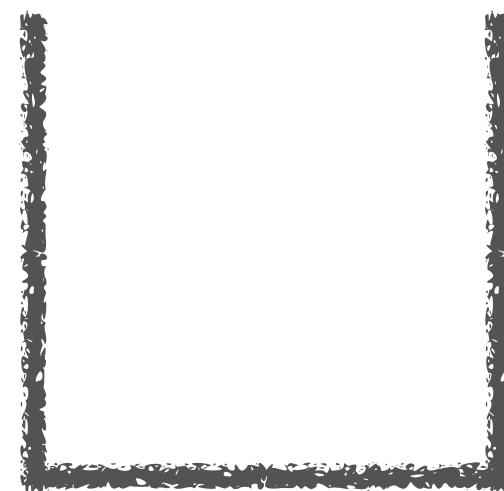
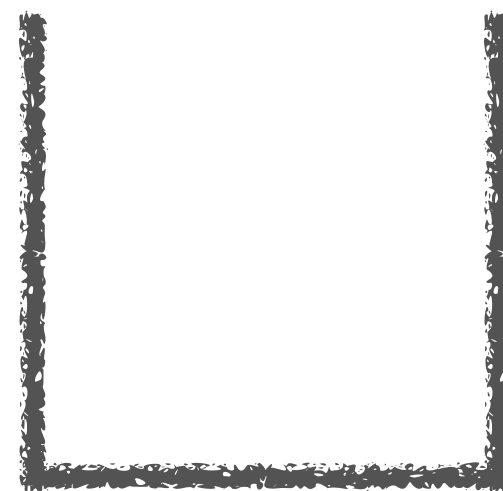
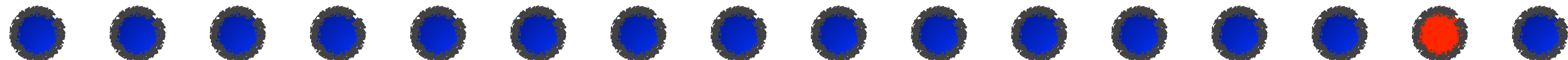
Yoo, Harman, Tonella, and Susi, ISSTA 2009 (<https://dl.acm.org/doi/abs/10.1145/1572272.1572296>)

- Let's say that, in CBR, humans cannot maintain consistency beyond k comparisons: how do we reduce the number of required comparisons?
 - Answer: scale the problem down by forming k clusters - ask to compare clusters, instead of individual test cases
 - The clustering should allow any k : Agglomerative Hierarchical Clustering 😎

Clustering Tests to Incorporate Human Knowledge

Yoo, Harman, Tonella, and Susi, ISSTA 2009 (<https://dl.acm.org/doi/abs/10.1145/1572272.1572296>)

Fault detected by a test with small coverage contribution



Fault-detecting test can be executed earlier.
Human engineer can compare buckets, not tests.

Modelling LLM Internal Space

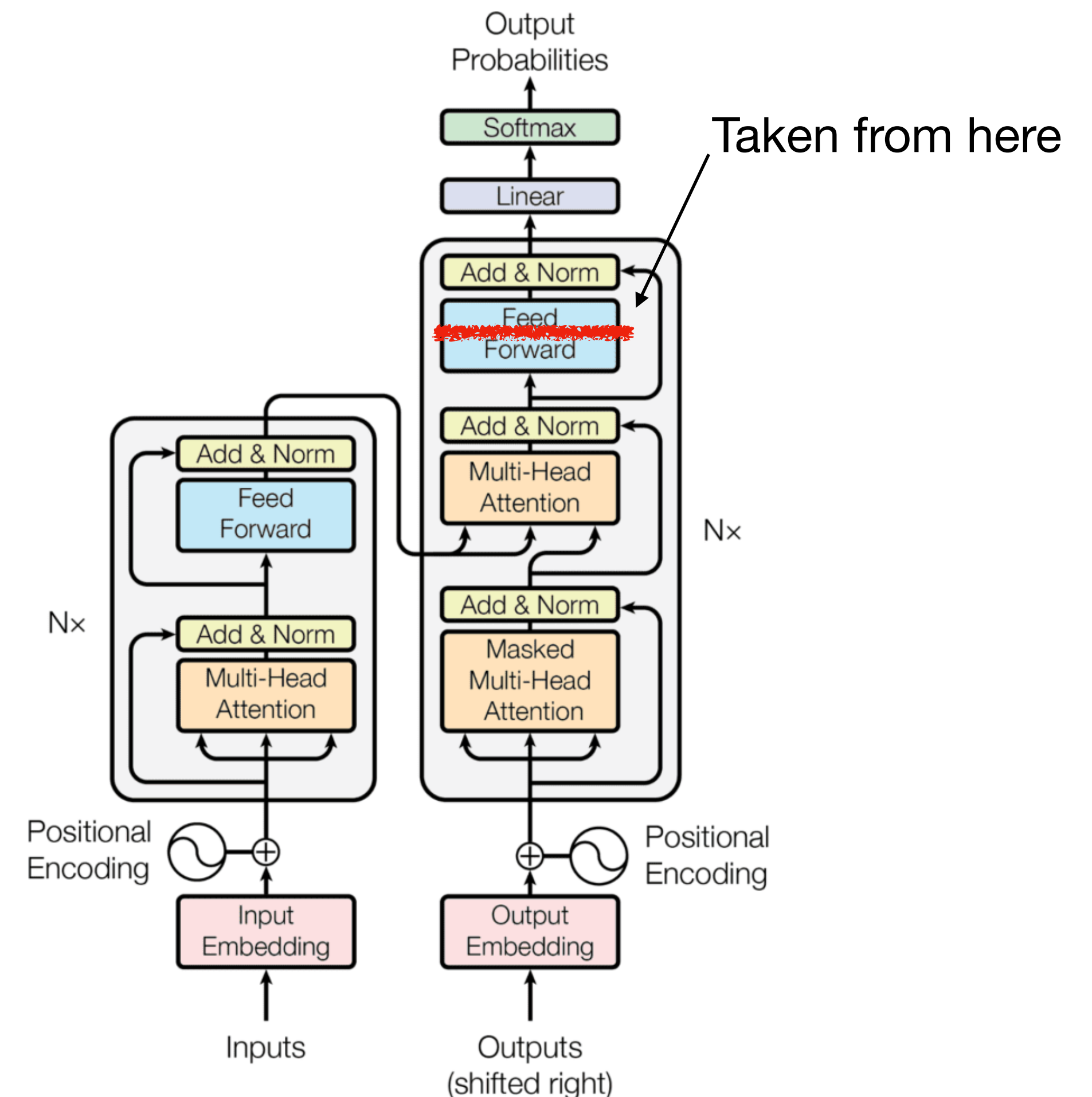
Yoon, Kim, Feldt, and Yoo, FSE 2026 (<https://arxiv.org/abs/2509.17314>)

- Imagine a task designed for an LLM — it consists of a prompt template (i.e., the instruction), and a user input (i.e., the concrete problem instance)
- Checking the correctness of the output (i.e., the oracle) is expensive: you need the generation cost (tokens) as well as the human oracle cost
- It would be really nice to have a quantitative measure that correlates with correctness (or, consequently, failures)
 - This would allow test prioritisation, without generating any output!

Modelling LLM Internal Space

Yoon, Kim, Feldt, and Yoo, FSE 2026 (<https://arxiv.org/abs/2509.17314>)

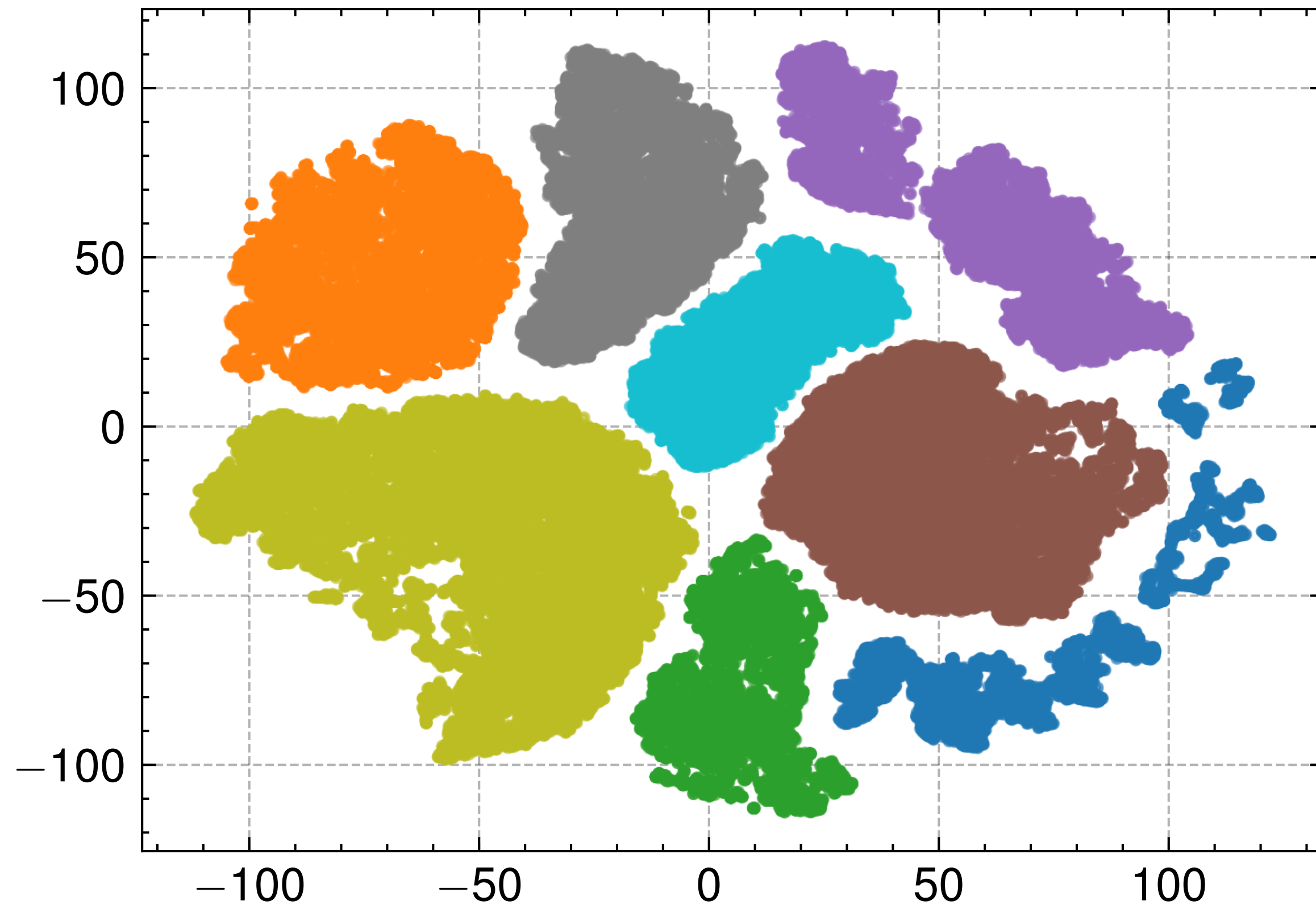
- It is known that the “internal state” of an LLM contains rich information about the answer, e.g., whether it is hallucinating (<https://arxiv.org/abs/2304.13734>).
- We want to model the distribution of known passing inputs based on their internal states.
 - If the internal state of a new input is similar to the known distribution, it may be more likely to pass.



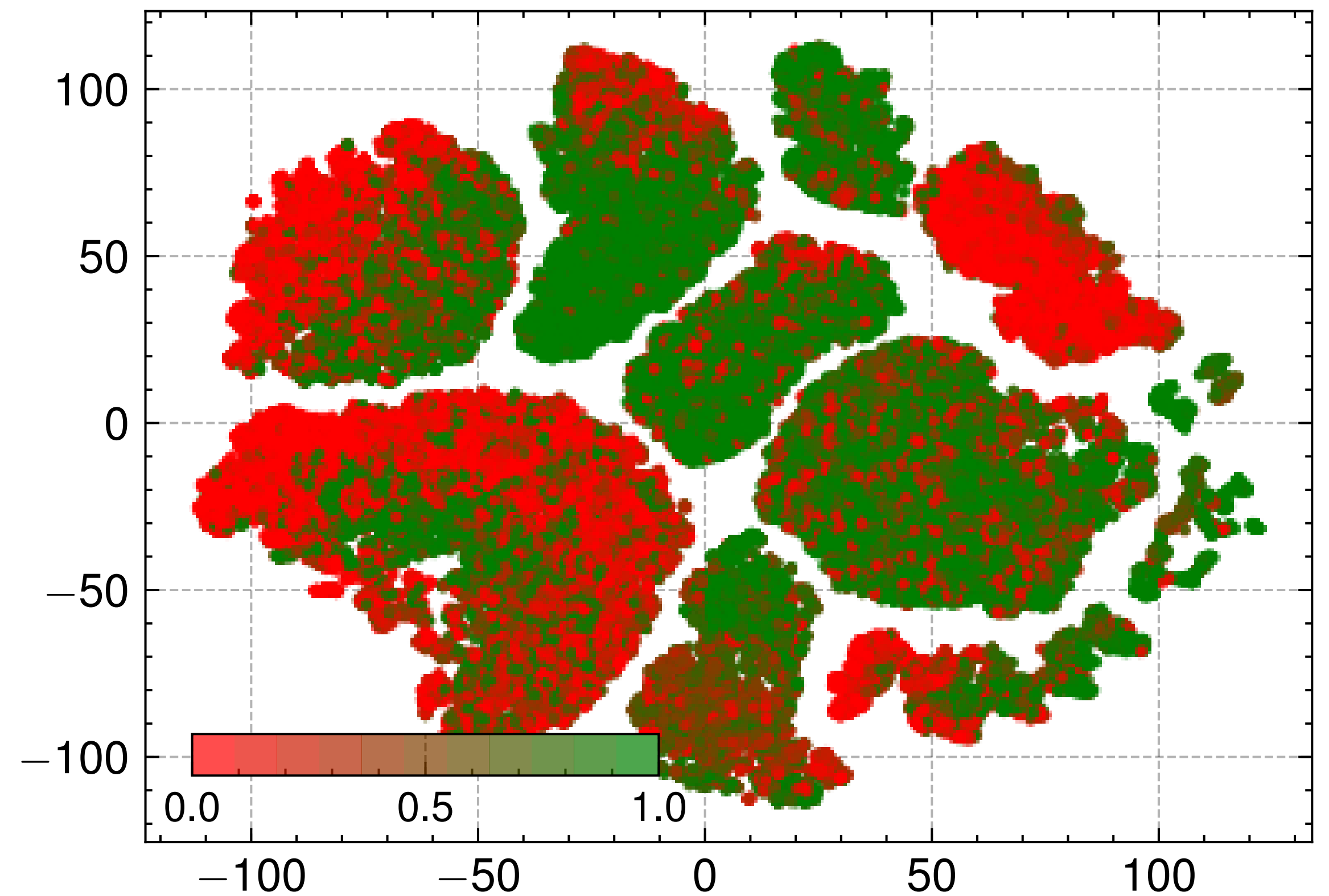
Modelling LLM Internal Space

Yoon, Kim, Feldt, and Yoo, FSE 2026 (<https://arxiv.org/abs/2509.17314>)

Input Hidden States Colored by Task



Input Hidden States Colored by Pass/Fail



Modelling LLM Internal Space

Yoon, Kim, Feldt, and Yoo, FSE 2026 (<https://arxiv.org/abs/2509.17314>)

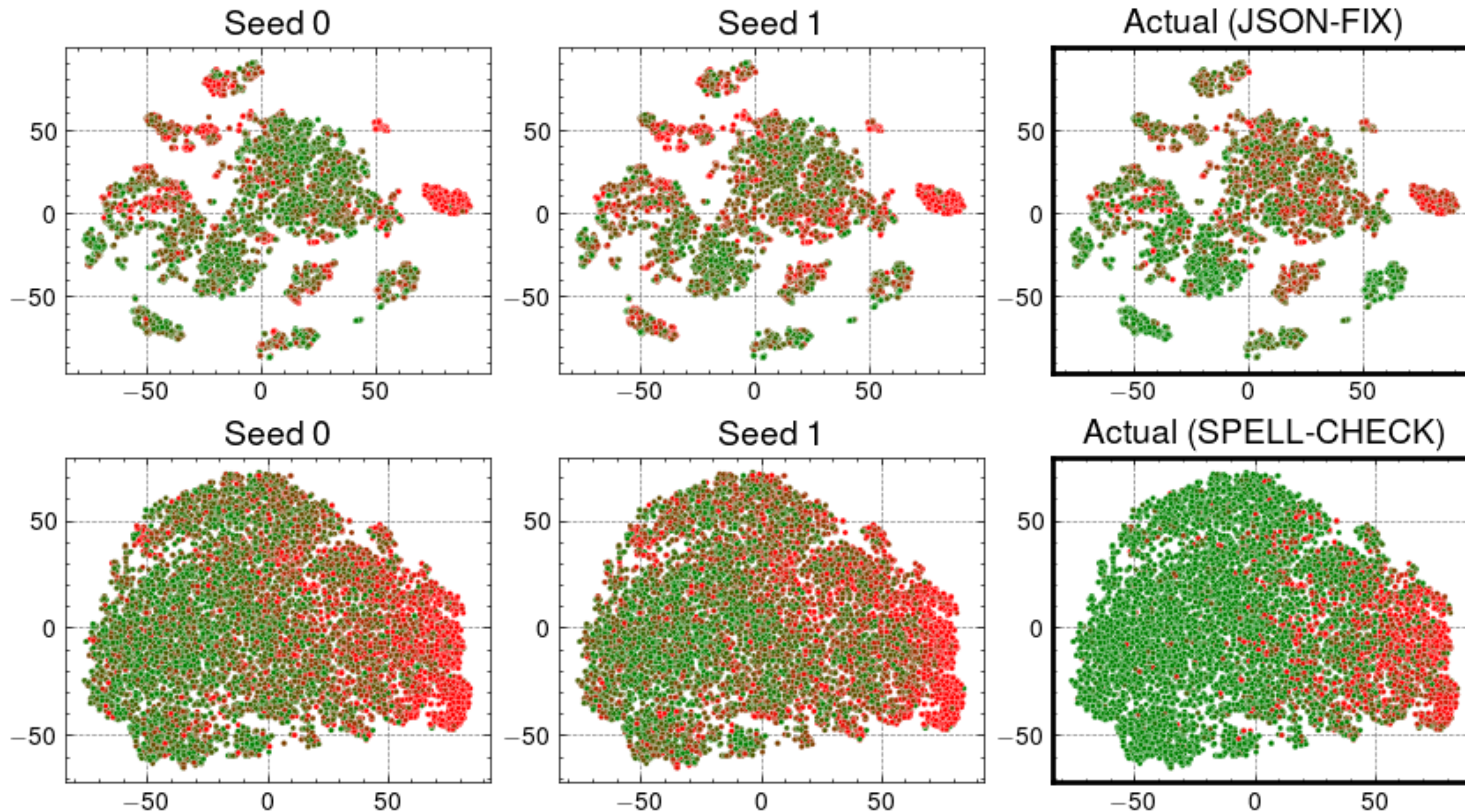
- We fit a Gaussian Mixture Model (essentially a form of density-based clustering) to explain the locations of *known reference passing inputs* in the Last Input-token Hidden State (the activation of the chosen layer for the last input token, before any output generation)
- When a new input x is given:
 - Find the nearest GMM component: for hidden space of x , $h(x)$, GMM gives density $p_{\theta}(h(x))$: a higher value indicates that x is within regions commonly associated with passing inputs
 - We can compute Likelihood Surprise Adequacy, $LSA(x) = -\log p_{\theta}(h(x))$: the higher this is, the more likely that x will fail.

We will come back to this later

Modelling LLM Internal Space

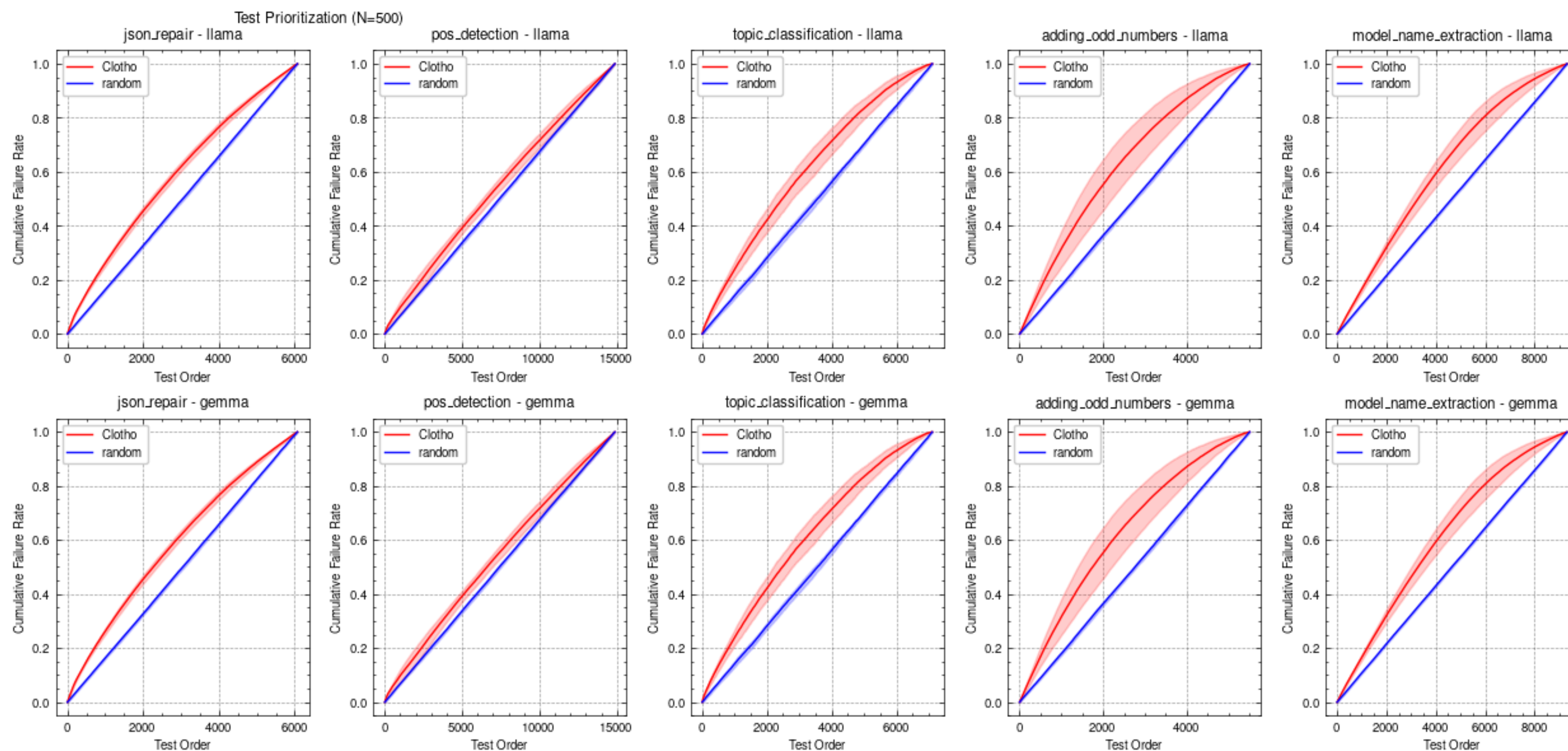
Yoon, Kim, Feldt, and Yoo, FSE 2026 (<https://arxiv.org/abs/2509.17314>)

Clotho (GMM w/ Adaptive Sampling and Training)



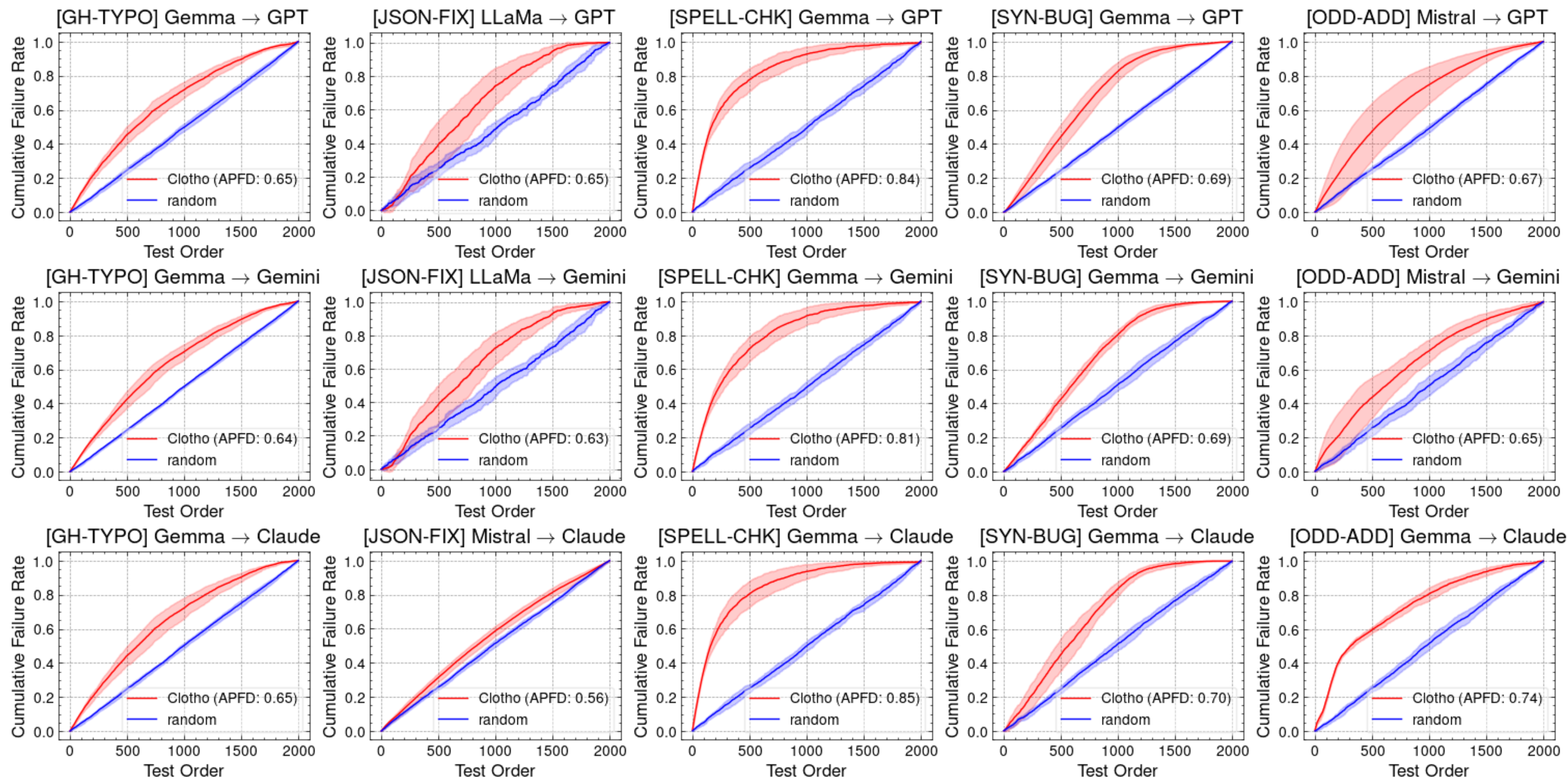
Modelling LLM Internal Space

Yoon, Kim, Feldt, and Yoo, FSE 2026 (<https://arxiv.org/abs/2509.17314>)



Modelling LLM Internal Space

Yoon, Kim, Feldt, and Yoo, FSE 2026 (<https://arxiv.org/abs/2509.17314>)



Summary

- Clustering is an unsupervised learning that groups data based on their internal similarities. It has many applications as part of bigger techniques.
 - Clustering can scale down a problem by working as an abstraction: multiple datapoints can be represented as a single representative (e.g., centroids)
 - Clustering can identify underlying structures, if the similarity/distance metrics can capture their impact (e.g., failure deduplication)
 - Clustering can explain the data, especially the density-based one, as the clusters essentially become models of the data (e.g., Gaussian Mixture Models)