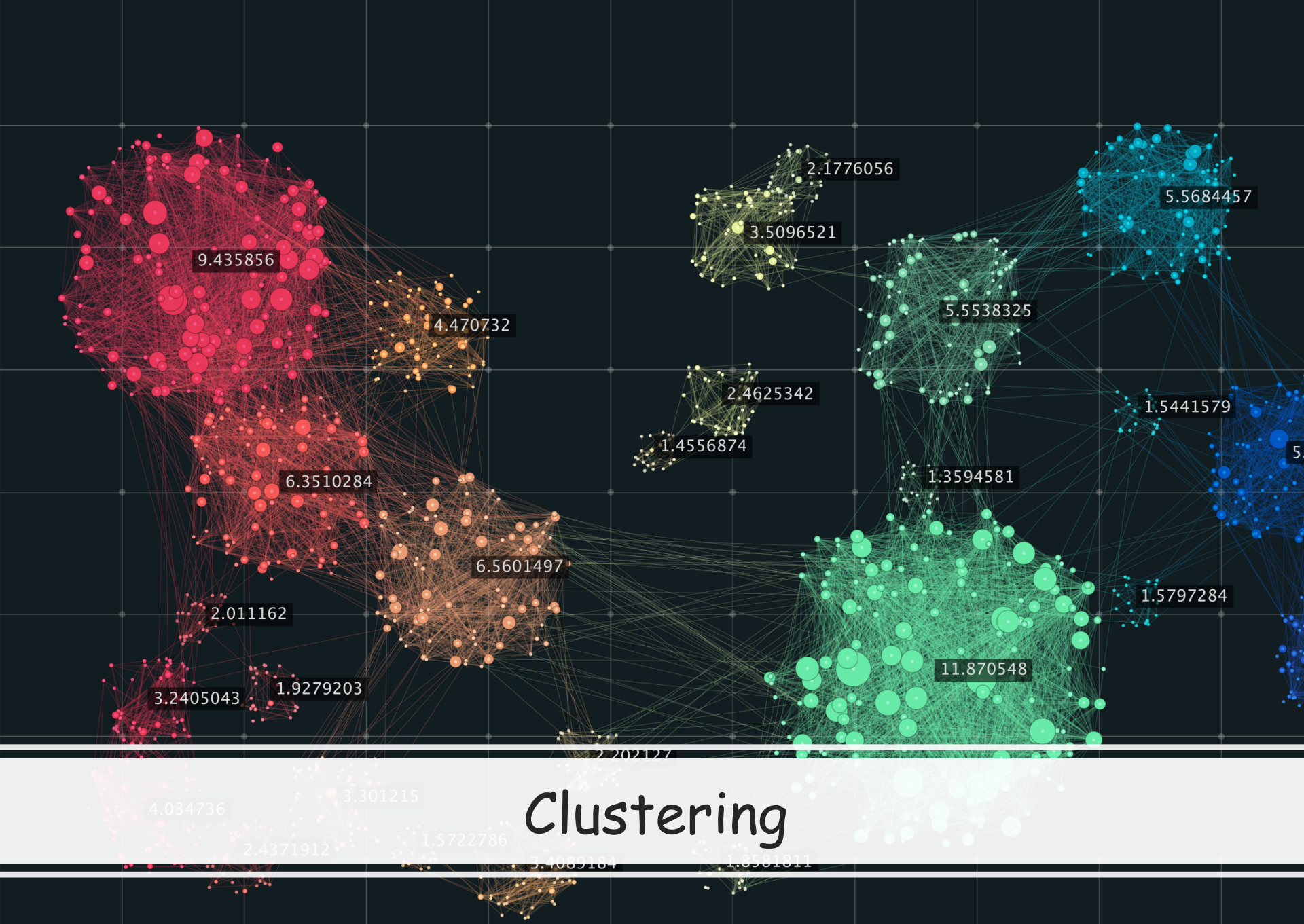

9.435856
4.470732
6.3510284
6.5601497
2.011162
3.2405043
1.9279203
2.1776056
3.5096521
5.5684457
5.5538325
2.4625342
1.4556874
1.5441579
1.3594581
1.5797284
11.870548

Clustering

Cluster Analysis

- Grouping objects in meaningful way
  - Clustered data fits together in some way
  - Often done to make sense of (big) data
  - Useful analysis technique in many fields
- Many different clustering strategies
- We consider 2 clustering methods
  - **K-means** and **EM clustering**
- But first, some background topics…

Intrinsic vs Extrinsic

- **Intrinsic clustering** relies on unsupervised learning
  - No predetermined labels on objects
  - Apply analysis directly to data
- **Extrinsic** relies on category labels
  - Requires pre-processing of data
  - Can be viewed as a form of supervised learning

Agglomerative vs Divisive

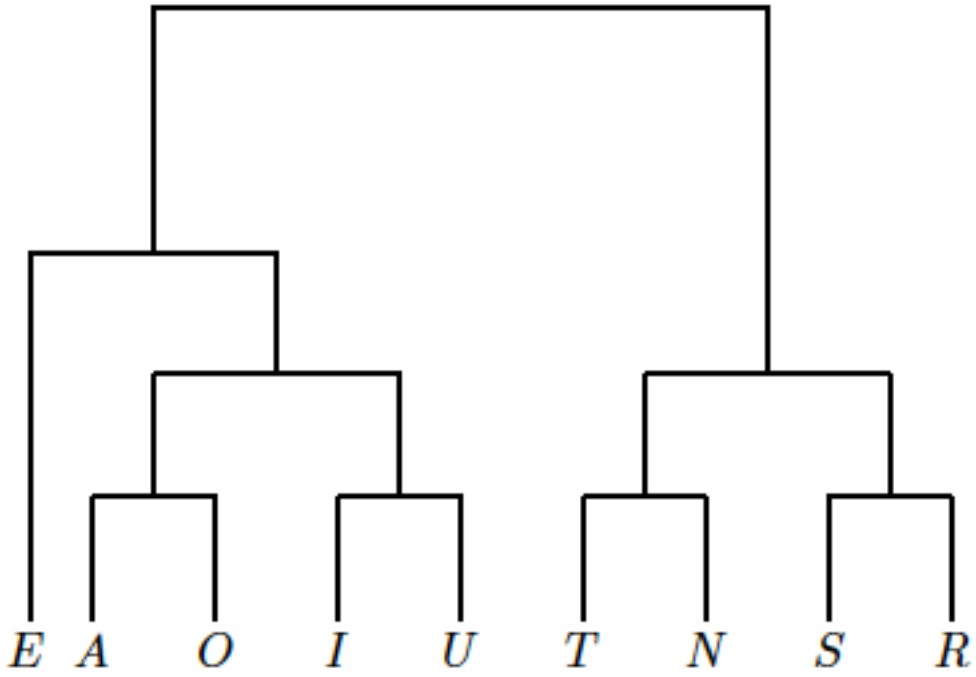
- **Agglomerative**
  - Each object starts in its own cluster
  - Cluster by merging existing 'small' clusters
  - A "bottom up" approach
- **Divisive**
  - All objects start in one 'very large' cluster
  - Clustering process splits existing clusters
  - A "top down" approach

Hierarchical vs Partitional

- **Hierarchical clustering**
  - "Child" and "parent" clusters
  - Can be viewed as dendrograms
- **Partitional clustering**
  - Partition objects into disjoint clusters
  - No hierarchical relationship
- We consider **K-means** and **EM** in detail
  - These are both partitional

Dendrogram

- Example
  - Obtained by hierarchical clustering



Hierarchical Clustering

- Example of hierarchical clustering...

1. Start with every point is its own cluster
2. While number of clusters exceeds 1
   - Find 2 "nearest" clusters and merge
3. End while

- This would be based on a function that defines '**distance**' between clusters
  - But it's not very theoretically sound... yet

This is also an agglomerative approach

Hierarchical Clustering

1: **Given:**

Data points $x_1, x_2, \ldots, x_n$ to cluster

Number of clusters K, where $K \leq n$

2: **Initialize:**

n clusters, each of size 1

3: Let $m = n$

4: **while** $m > K$ **do**

5: Find two nearest clusters and merge

6: $m = m - 1$

7: **end while**

Distance

- Distance between data points?
- Let's see two very common examples.

  Suppose:

  $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$

  where each x_i and y_i are real numbers
- **Euclidean distance** is:

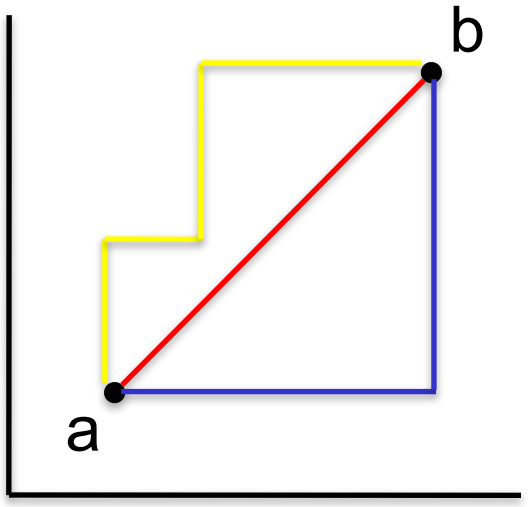
$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

- **Manhattan** (taxicab) **distance** is:

$$d(x, y) = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_n - y_n|$$

Distance

- Euclidean distance — red line
- Manhattan distance — blue or yellow
  - Or any similar right-angle only path



Distance

- Lots and lots more distance measures
- Other examples (kinda less intuitive) include:
  - **Mahalanobis distance** — includes means and covariance in distance measure
  - **Simple substitution distance** — measure of "decryption" distance
  - **Chi-squared** "distance" — statistical
- Sometimes use non-distance measures, such as **cosine similarity**
  - Usually for spherical K-means clustering

An Approach to Clustering…

- Given data points $x_1, x_2, x_3, \ldots, x_m$
- Want to partition into K **clusters**
- A **centroid** specified for each cluster
  - Let $c_1, c_2, \ldots, c_K$ denote current centroids
- Each x_i associated with one centroid
  - Let centroid(x_i) be centroid for x_i
  - If c_j = centroid(x_i), then x_i is in cluster j

Questions

- Two general questions…

1. How to determine **centroids**, $\mathbf{c}_j$?
2. How to determine clusters, that is, how to assign the $\{x_i\}$ to centroids?

- But first, what makes a cluster "good"?
  - For now, focus on one individual cluster
  - Relationship between clusters later…

Distortion

- Intuitively, "compact" clusters good
  - Depends on data and K, which are given
  - And depends on centroids and assignment of x_i to clusters, which we can control

➢How to measure "goodness"?

Note that we are assuming K is specified in advance.

Distortion

➢How to measure "goodness"?

- Define **distortion** as:

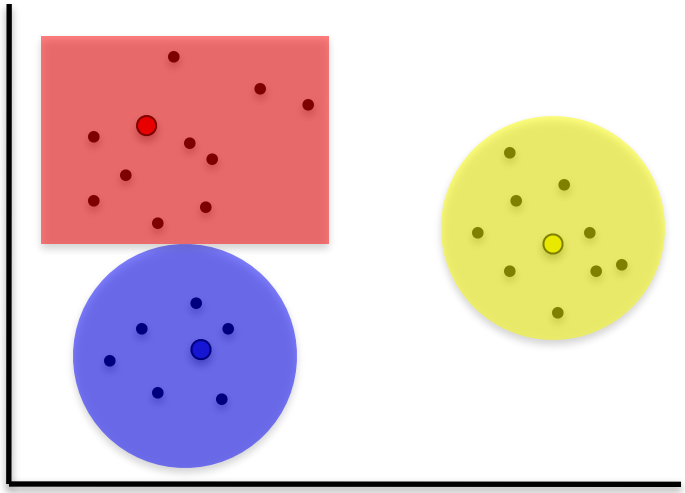
$$\text{distortion} = \sum_{i=1}^{n} d\big(X_i, \text{centroid}(X_i)\big)$$

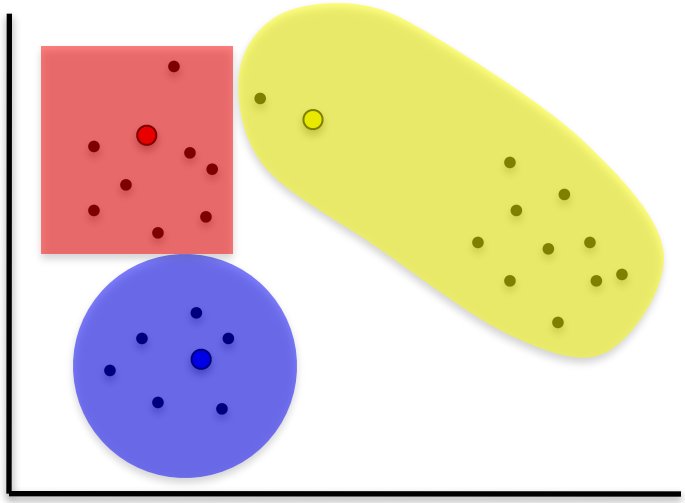
  - Where **d(x,y)** is a distance measure

- Given K, let's try to <u>minimize distortion</u>

Example

- Consider this 2-d data
  - Choose **K = 3** clusters
- Same data for both
  - Which has smaller distortion?
- How to **minimize** distortion?
  - Good question…





Minimize Distortion

- Note that distortion depends on K
  - So, we should probably write **distortion$_K$**

- Want to solve the following problem:
  - Given: K and $x_1, x_2, x_3, \ldots, x_m$
  - Minimize: distortion$_K$

  - Best choice of K is a separate issue
    - Briefly considered this later
    - For now, assume K is given and fixed

How to Minimize Distortion?

- Assume we are given **m** data points and **K**
- Minimize distortion via exhaustive search?
  - Can we try all possible cases?
  - Way too much work for realistic size data set

- An exact solution is NP-complete problem
  - Even for 2-dimensional data
- An approximate solution will have to do

How to Minimize Distortion?

- **Key Observations**:
  1. Each x_i clustered with nearest centroid
  2. Centroid must be at the center of its cluster

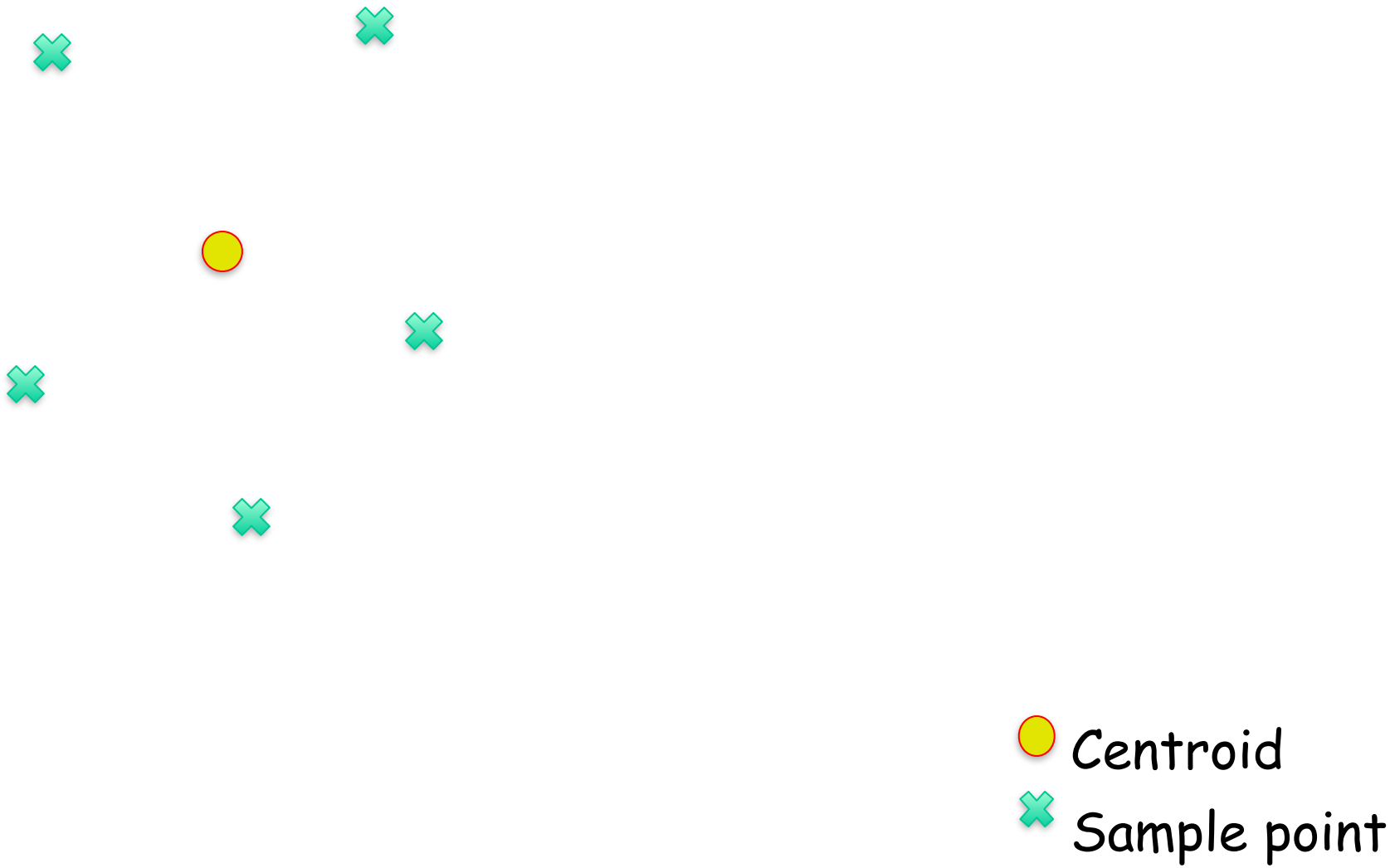
- "Key observation" #1 is obvious
- "Key observation" #2 is not quite so obvious, but can be verified using some basic calculus
  - Minimize distortion function by taking partial derivatives and set to 0

K-Means

- Previous slide implies that we can improve suboptimal cluster by either…
  1. Reassign each x_i to nearest centroid
  2. Recompute centroids as cluster centers

- No improvement from applying either 1 or 2 more than once in succession
- But **alternating** might be useful
  - In fact, that is the **K-means algorithm**

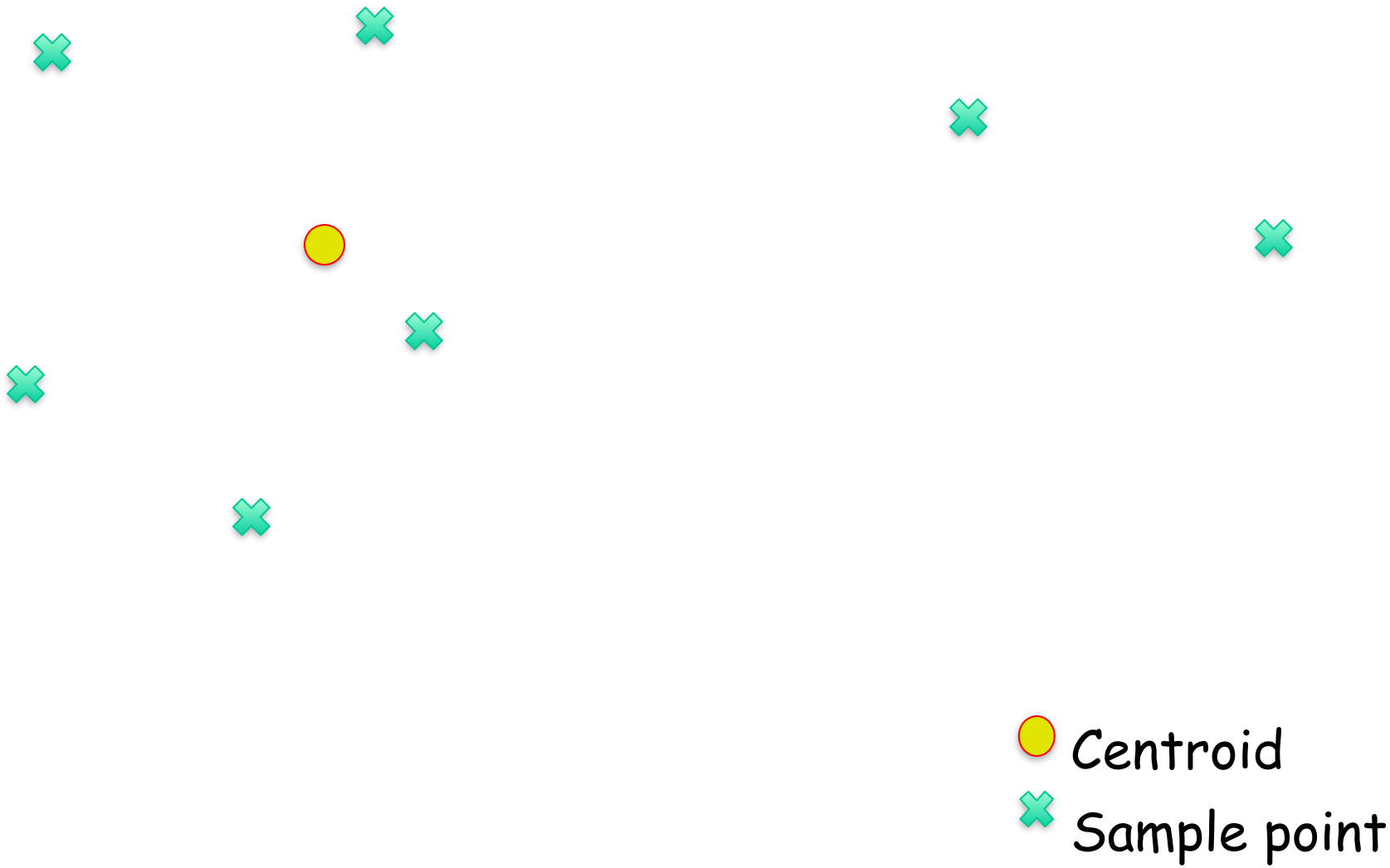
We have a single cluster and its centroid



What happens when a new point is added to such cluster?

What happens when a new point is added to such cluster?

What happens when a new point is added to such cluster?



K-Means Algorithm

- Given dataset…

1. Select a value for K (how?)
2. Select initial centroids (how?)
3. Group data by nearest centroid
4. Recompute centroids (cluster centers)
5. If "significant" change, then goto 3; else stop

K-Means

- Are we assured of optimal solution?
  - Definitely not!
- Why not?
  - Initial centroid locations matter (a lot)
    - Dependence on initial centroids
  - This is a common issue in iterative processes (HMM training, for example)
    - This is a generic problem for a hill climb

K-Means Initialization

Recall, K is the number of clusters

- **How to choose K?**
  - No obvious "best" way to do so
- But **K-means** is fast
  - So trial and error may be OK
    - That is, experiment with different K
    - Similar to choosing N in HMM
- Is there a better way to choose K?

Optimal K?

- Even for trial and error, we need a way to measure "goodness" of results
- Choosing optimal **K** is tricky
  - Intuitive measures of quality will likely improve for **larger K**
  - ➢ But if K is "too big", may overfit
    - In extreme, every point is its own cluster
- When is K big enough, but not too big?

K-Means Initialization

- How to choose initial centroids?
- Again, no best way to do this
  - Counterexamples to any "best" approach
- Often just choose at random
- Or uniform/maximum spacing
  - Or some variation on this idea
- Other?

K-Means Initialization

- In practice, we might do following:
  1. Try several different choices of K
  2. For each K, test several initial centroids
  3. Select the result that is best
     - Might not be very scientific, but often it's good enough
- Okay, but how to measure "best"?

Variations on K-Means

- **K-mediods**
  - **Centroids** point must be actual data point
    - More robust to outliers, but more costly
    - Basically it could be called k-median
- **Spherical K-means**
  - Use cosine similarity as "distance"
  - Normalize all data to unit vectors, so all vectors of same magnitude and sum of vectors in cluster centroid
- **Fuzzy K-means**
  - Computationally easy modification

And many other variations…

Measuring Cluster Quality

- How can we judge clustering results?
  - In general, not just for K-means
- Comparison to typical training/scoring?
  - Suppose we test new scoring method
  - E.g., score malware and benign files
  - Compute ROC curves, AUC, etc.
  - Many tools to measure success/accuracy
- Clustering is different (Why? How?)

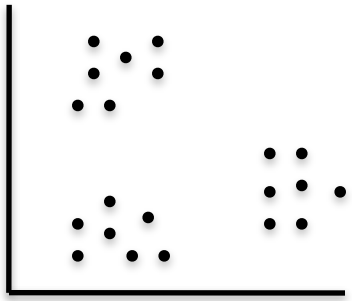
Clustering Quality

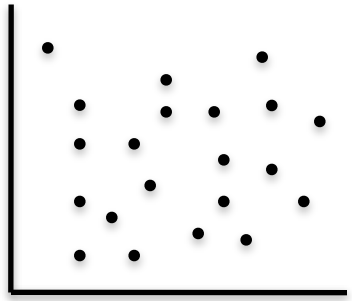
- Clustering typically a fishing expedition
  - Not sure what we are looking for
  - Hoping to find structure, "data discovery"
  - If we know the answer, then why cluster?

- Might find something that's not there
  - Random data can be clustered!

Cluster-ability?

- Clustering "tendency"
  - ➢How suitable is dataset for clustering?
    - Which dataset below is cluster-friendly?





- We can always apply clustering…
  - …but expect better results in some cases

How to Validate Clusters?

- **External validation**
  - Compare clusters based on data labels
  - Similar to usual training/scoring scenario
  - Good idea if know something about data

- **Internal validation**
  - Determine quality based only on clusters
  - E.g., spacing between/within clusters
  - Harder to do, but always applicable

It's All Relative

- Comparing clustering results
  - That is, compare one clustering result with others for same dataset
    - Could enable us to "climb" to better clustering results
- Still need a way to quantify "quality"

Internal Validation

- Direct measurement of clusters
  - Might call it "topological" validation
- We'll consider the following:
  - Cluster correlation
  - Similarity matrix
  - Sum of squares error
  - Cohesion and separation
  - Silhouette coefficient

Correlation Coefficient

- For $X=(x_1,x_2,\ldots,x_n)$ and $Y=(y_1,y_2,\ldots,y_n)$
  - Correlation coefficient r_{XY} is:

$$r_{XY} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{n}(x_i - \mu_X)(y_i - \mu_Y)}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_X)^2 \sum_{i=1}^{n}(y_i - \mu_Y)^2}}$$

  - Can show $-1 \leq r_{XY} \leq 1$
    - If $r_{XY} > 0$ then positive cor (and vice versa)
    - Magnitude is strength of correlation

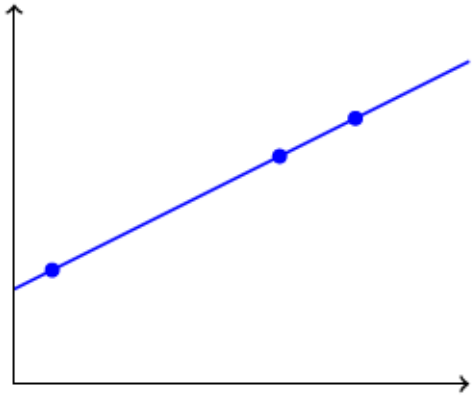
Correlation Coefficient

- It's the normalized version of the Covariance

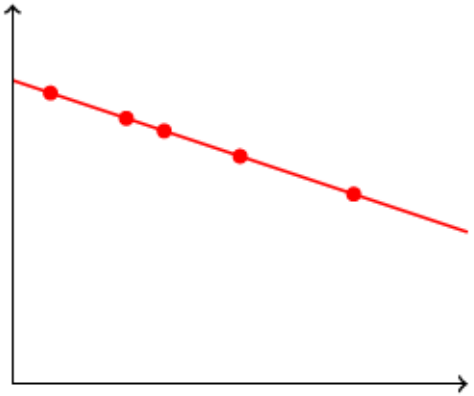
$$r_{XY} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{n}(x_i - \mu_X)(y_i - \mu_Y)}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_X)^2 \sum_{i=1}^{n}(y_i - \mu_Y)^2}}$$

- It shows the tendency in the linear relationship between the variables
  - R_{XY} equal to 1 is total positive linear correlation, 0 is no linear correlation, and –1 is total negative linear correlation

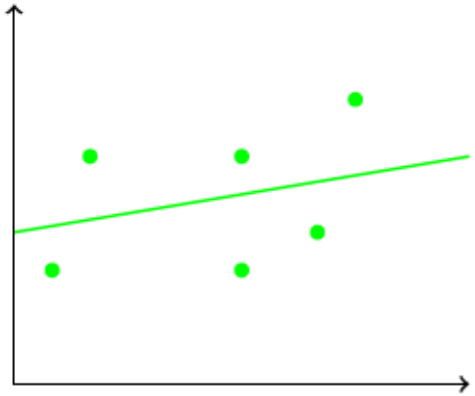
Examples of r_{XY} in 2-d



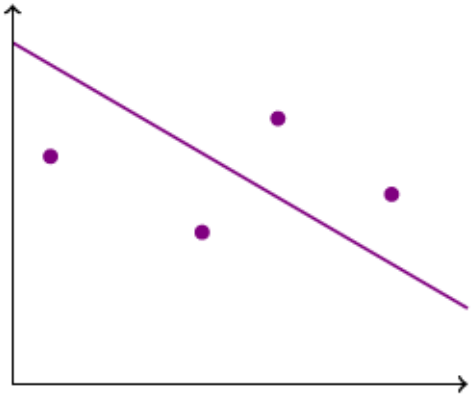
(a) Correlation $r_{XY} = 1$



(b) Correlation $r_{XY} = -1$

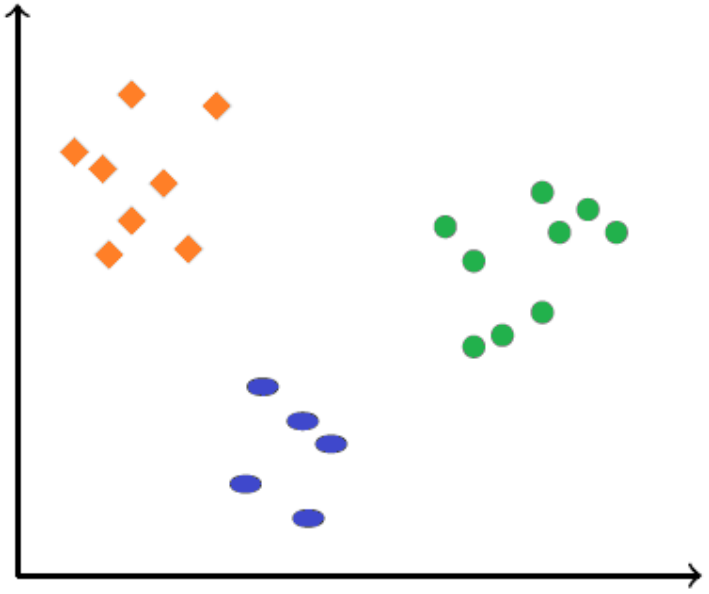


(c) Correlation $0 < r_{XY} < 1$

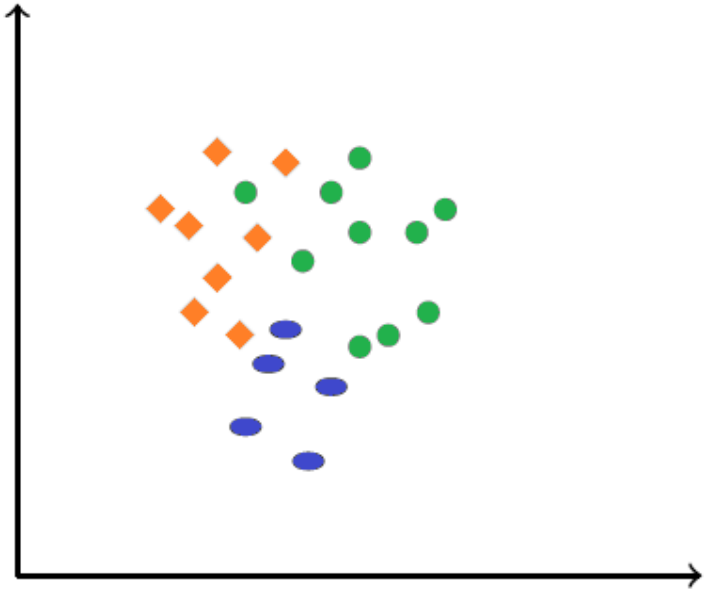


(d) Correlation $-1 < r_{XY} < 0$

Examples of r_{XY} in 2-d



(a) Correlation $r_{AD} = -0.8652$



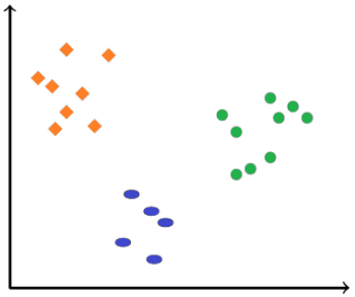
(b) Correlation $r_{AD} = -0.5347$

- Farer from 0, the better!

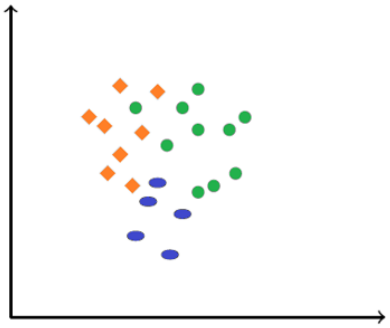
Similarity Matrix

- Form "similarity matrix"
  - Could be based on just about anything
  - For example, distance matrix $D = \{d_{ij}\}$, where $d_{ij} = d(x_i, x_j)$
- Group rows and columns by cluster
- Make a **heat map** of resulting matrix
  - Provides visual representation of similarity within clusters (so look at it…)

Heat Maps



X	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	0.00	0.36	0.70	0.85	0.75	1.61	0.95	1.41	2.17	2.25	1.71	2.49	2.81	3.28	3.50	3.82	4.46	4.46	3.96	3.20	2.97	3.86	4.23
2	0.36	0.00	0.56	0.49	0.51	1.25	0.78	1.10	2.23	2.42	1.77	2.55	2.94	3.16	3.37	3.67	4.25	4.25	3.75	3.01	2.75	3.62	4.00
3	0.70	0.56	0.00	0.62	1.03	1.27	1.31	1.44	1.72	2.01	1.28	2.03	2.48	2.61	2.83	3.13	3.76	3.76	3.26	2.50	2.27	3.16	3.53
4	0.85	0.49	0.62	0.00	0.61	0.76	0.89	0.83	2.31	2.63	1.88	2.61	3.09	2.97	3.16	3.42	3.92	3.92	3.42	2.72	2.42	3.25	3.65
5	0.75	0.51	1.03	0.61	0.00	1.14	0.29	0.70	2.73	2.93	2.28	3.05	3.46	3.56	3.75	4.02	4.52	4.52	4.03	3.32	3.03	3.85	4.26
6	1.61	1.25	1.27	0.76	1.14	0.00	1.31	0.76	2.75	3.21	2.38	3.02	3.59	3.01	3.14	3.32	3.64	3.64	3.16	2.57	2.21	2.92	3.35
7	0.95	0.78	1.31	0.89	0.29	1.31	0.00	0.71	3.01	3.18	2.55	3.33	3.72	3.85	4.04	4.30	4.79	4.79	4.29	3.60	3.30	4.11	4.52
8	1.41	1.10	1.44	0.83	0.70	0.76	0.71	0.00	3.13	3.45	2.71	3.43	3.92	3.66	3.82	4.02	4.39	4.39	3.91	3.29	2.94	3.68	4.10
9	2.17	2.23	1.72	2.31	2.73	2.75	3.01	3.13	0.00	0.74	0.46	0.32	0.85	1.63	1.90	2.30	3.21	3.21	2.79	2.05	2.11	2.90	3.11
10	2.25	2.42	2.01	2.63	2.93	3.21	3.18	3.45	0.74	0.00	0.86	0.83	0.63	2.33	2.60	3.00	3.92	3.92	3.52	2.79	2.85	3.64	3.84
11	1.71	1.77	1.28	1.88	2.28	2.38	2.55	2.71	0.46	0.86	0.00	0.78	1.22	1.88	2.15	2.53	3.38	3.38	2.93	2.15	2.13	2.98	3.24
12	2.49	2.55	2.03	2.61	3.05	3.02	3.33	3.43	0.32	0.83	0.78	0.00	0.68	1.51	1.78	2.18	3.11	3.11	2.72	2.03	2.15	2.87	3.04
13	2.81	2.94	2.48	3.09	3.46	3.59	3.72	3.92	0.85	0.63	1.22	0.68	0.00	2.09	2.33	2.73	3.68	3.68	3.33	2.68	2.82	3.51	3.65
14	3.28	3.16	2.61	2.97	3.56	3.01	3.85	3.66	1.63	2.33	1.88	1.51	2.09	0.00	0.27	0.67	1.60	1.60	1.25	0.75	1.08	1.48	1.56
15	3.50	3.37	2.83	3.16	3.75	3.14	4.04	3.82	1.90	2.60	2.15	1.78	2.33	0.27	0.00	0.40	1.35	1.35	1.03	0.70	1.07	1.30	1.33
16	3.82	3.67	3.13	3.42	4.02	3.32	4.30	4.02	2.30	3.00	2.53	2.18	2.73	0.67	0.40	0.00	0.96	0.96	0.72	0.75	1.13	1.05	0.98
17	4.46	4.25	3.76	3.92	4.52	3.64	4.79	4.39	3.21	3.92	3.38	3.11	3.68	1.60	1.35	0.96	0.00	0.00	0.50	1.27	1.50	0.74	0.32
18	4.46	4.25	3.76	3.92	4.52	3.64	4.79	4.39	3.21	3.92	3.38	3.11	3.68	1.60	1.35	0.96	0.00	0.00	0.50	1.27	1.50	0.74	0.32
19	3.96	3.75	3.26	3.42	4.03	3.16	4.29	3.91	2.79	3.52	2.93	2.72	3.33	1.25	1.03	0.72	0.50	0.50	0.00	0.79	1.00	0.38	0.32
20	3.20	3.01	2.50	2.72	3.32	2.57	3.60	3.29	2.05	2.79	2.15	2.03	2.68	0.75	0.70	0.75	1.27	1.27	0.79	0.00	0.39	0.85	1.10
21	2.97	2.75	2.27	2.42	3.03	2.21	3.30	2.94	2.11	2.85	2.13	2.15	2.82	1.08	1.07	1.13	1.50	1.50	1.00	0.39	0.00	0.90	1.26
22	3.86	3.62	3.16	3.25	3.85	2.92	4.11	3.68	2.90	3.64	2.98	2.87	3.51	1.48	1.30	1.05	0.74	0.74	0.38	0.85	0.90	0.00	0.43
23	4.23	4.00	3.53	3.65	4.26	3.35	4.52	4.10	3.11	3.84	3.24	3.04	3.65	1.56	1.33	0.98	0.32	0.32	0.32	1.10	1.26	0.43	0.00



X	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	0.00	0.36	0.45	0.85	0.75	1.53	0.95	1.41	0.74	1.01	0.80	1.32	1.56	1.46	1.70	2.05	2.11	2.42	1.66	1.10	1.23	1.66	2.05
2	0.36	0.00	0.54	0.49	0.51	1.17	0.78	1.10	0.79	1.23	0.69	1.31	1.70	1.35	1.55	1.86	1.82	2.12	1.35	0.79	0.89	1.31	1.70
3	0.45	0.54	0.00	0.86	1.05	1.55	1.30	1.61	0.29	0.70	0.43	0.87	1.17	1.05	1.30	1.68	1.84	2.16	1.45	0.93	1.35	1.57	1.96
4	0.85	0.49	0.86	0.00	0.61	0.70	0.89	0.83	1.00	1.55	0.74	1.36	1.90	1.24	1.37	1.60	1.41	1.69	0.91	0.41	0.51	0.82	1.20
5	0.75	0.51	1.05	0.61	0.00	1.01	0.29	0.70	1.30	1.72	1.17	1.80	2.21	1.78	1.95	2.20	2.00	2.26	1.50	1.02	0.64	1.31	1.66
6	1.53	1.17	1.55	0.70	1.01	0.00	1.17	0.61	1.66	2.23	1.35	1.89	2.51	1.63	1.66	1.73	1.25	1.43	0.82	0.76	0.38	0.43	0.67
7	0.95	0.78	1.30	0.89	0.29	1.17	0.00	0.71	1.57	1.95	1.45	2.09	2.47	2.07	2.24	2.48	2.25	2.50	1.75	1.30	0.79	1.52	1.83
8	1.41	1.10	1.61	0.83	0.70	0.61	0.71	0.00	1.81	2.31	1.57	2.19	2.72	2.03	2.12	2.26	1.85	2.04	1.39	1.13	0.35	1.03	1.25
9	0.74	0.79	0.29	1.00	1.30	1.66	1.57	1.81	0.00	0.59	0.34	0.59	0.92	0.81	1.08	1.47	1.74	2.06	1.40	0.95	1.51	1.60	1.97
10	1.01	1.23	0.70	1.55	1.72	2.23	1.95	2.31	0.59	0.00	0.92	0.83	0.63	1.22	1.48	1.89	2.27	2.58	1.97	1.53	2.05	2.18	2.55
11	0.80	0.69	0.43	0.74	1.17	1.35	1.45	1.57	0.34	0.92	0.00	0.64	1.17	0.67	0.90	1.26	1.43	1.75	1.07	0.62	1.25	1.26	1.63
12	1.32	1.31	0.87	1.36	1.80	1.89	2.09	2.19	0.59	0.83	0.64	0.00	0.68	0.43	0.67	1.07	1.54	1.84	1.37	1.13	1.86	1.70	2.02
13	1.56	1.70	1.17	1.90	2.21	2.51	2.47	2.72	0.92	0.63	1.17	0.68	0.00	1.10	1.30	1.67	2.21	2.51	2.05	1.75	2.41	2.36	2.69
14	1.46	1.35	1.05	1.24	1.78	1.63	2.07	2.03	0.81	1.22	0.67	0.43	1.10	0.00	0.27	0.67	1.12	1.42	1.00	0.90	1.68	1.37	1.65
15	1.70	1.55	1.30	1.37	1.95	1.66	2.24	2.12	1.08	1.48	0.90	0.67	1.30	0.27	0.00	0.40	0.93	1.21	0.93	0.99	1.77	1.35	1.57
16	2.05	1.86	1.68	1.60	2.20	1.73	2.48	2.26	1.47	1.89	1.26	1.07	1.67	0.67	0.40	0.00	0.71	0.91	0.92	1.19	1.91	1.35	1.48
17	2.11	1.82	1.84	1.41	2.00	1.25	2.25	1.85	1.74	2.27	1.43	1.54	2.21	1.12	0.93	0.71	0.00	0.32	0.50	1.03	1.54	0.83	0.82
18	2.42	2.12	2.16	1.69	2.26	1.43	2.50	2.04	2.06	2.58	1.75	1.84	2.51	1.42	1.21	0.91	0.32	0.00	0.78	1.33	1.76	1.01	0.87
19	1.66	1.35	1.45	0.91	1.50	0.82	1.75	1.39	1.40	1.97	1.07	1.37	2.05	1.00	0.93	0.92	0.50	0.78	0.00	0.56	1.06	0.43	0.65
20	1.10	0.79	0.93	0.41	1.02	0.76	1.30	1.13	0.95	1.53	0.62	1.13	1.75	0.90	0.99	1.19	1.03	1.33	0.56	0.00	0.78	0.65	1.03
21	1.23	0.89	1.35	0.51	0.64	0.38	0.79	0.35	1.51	2.05	1.25	1.86	2.41	1.68	1.77	1.91	1.54	1.76	1.06	0.78	0.00	0.75	1.04
22	1.66	1.31	1.57	0.82	1.31	0.43	1.52	1.03	1.60	2.18	1.26	1.70	2.36	1.37	1.35	1.35	0.83	1.01	0.43	0.65	0.75	0.00	0.39
23	2.05	1.70	1.96	1.20	1.66	0.67	1.83	1.25	1.97	2.55	1.63	2.02	2.69	1.65	1.57	1.48	0.82	0.87	0.65	1.03	1.04	0.39	0.00

Other techniques

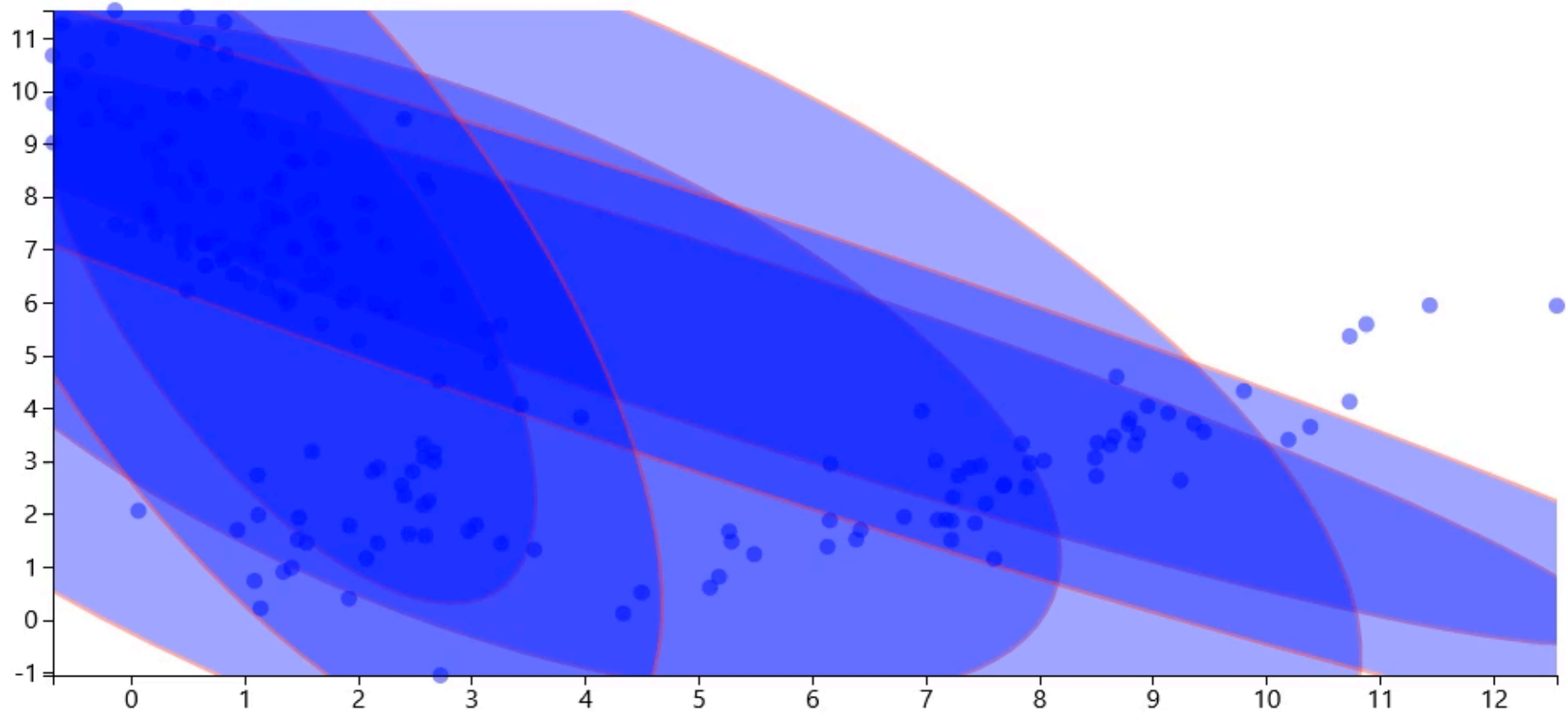
- Residual sum of squares (RSS)
- Cohesion and Separation
- Silhouette Coefficient
  - Basically Cohesion and Separation in a single value
- External Validation
  - Several techniques that measure quality based on data in clusters, not relying on cluster topology ("shape")

EM Clustering

- Samples might be from different probability distributions
  - Euclidean distance might be poor measure
  - Maybe better to cluster by distribution
- Cluster using **probability distributions**?
  - Good idea, but distributions are unknown…
- **Expectation maximization** (EM)
  - Can determine unknown parameters of probability distributions

EM for Clustering

- How is EM relevant to clustering?
- Can use EM to obtain parameters of K "hidden" distributions
  - That is, **means** and **variances**, μ_i and σ_i^2
- Then, use μ_i as centers of clusters
  - And σ_i (**standard deviations**) as "radii"
  - Often use Gaussian (normal) distributions
- Is this better than K-means?


11
10
9
8
7
6
5
4
3
2
1
0
-1
0
1
2
3
4
5
6
7
8
9
10
11
12

EM vs K-Means

- Whether it is better or not, EM is obviously different than K-means…
  - …or is it?
- K-means is like **a special case of EM**
  - Using "distance" instead of "probabilities"
  - In K-means, we re-assign points to centroids
    - Like "E" in EM, which "re-shapes" clusters
  - In K-means, we recompute centroids
    - Like "M" of EM, where we recompute parameters

Conclusion

- Clustering is fun, entertaining, and useful
  - Can explore mysterious data, and more…
- And K-means is **really**, **really** simple
- EM is powerful and not much more difficult
  - Different distributions yield different "shapes"
- Measuring clustering success is not so easy
  - "Good" clusters? Useful results? Or noise?
  - Anything can be clustered!
- Clustering is often just a starting point
  - Can helps us decide if any "there" is there

Clustering Application

Clustering for malware classification

Swathi Pai[1] · Fabio Di Troia[2] · Corrado Aaron Visaggio[2] · Thomas H. Austin[1] · Mark Stamp[1]

- We analyze just part of this research work

Quest for the Holy Grail

- Holy Grail of malware research is to detect previously unseen malware
  - So-called "zero day" malware
- If you solve this problem, you'll be rich
- We don't consider this problem here
- But we do consider something similar
- Problem here… classify "new" malware
  - New in a sense…

Training HMMs

- Train HMM for each of the following…
- Four compilers
  - GCC, MinGW, TurboC, Clang
- Hand-written assembly code
  - We refer to this model as TASM
- Two metamorphic families
  - NGVCK and MWOR

Scoring

- For each malware sample under consideration…
- Score sample with each of 7 models
  - GCC, Clang, TurboC, MinGW, TASM, NGVCK, MWOR
- Each score is normalized (LLPO)
- For each malware sample, obtain a vector of 7 (normalized) HMM scores

Clustering Details

- Suppose we have N malware samples,

  $m_1, m_2, \ldots, m_N$

- Suppose score vector for m_i is

  $(a_i, b_i, c_i, d_i, e_i, f_i, g_i)$

- Let $a_{\min} = \min\{a_i\}$ and $a_{\max} = \max\{a_i\}$
- Given K, the number of cluster
  - Let $a = (a_{\max} - a_{\min}) / (K+1)$
- Define b, c, d, e, f, g similarly

Initial Centroids

- For $j = 1,2,\ldots,K$ define **initial** centroids

  $C_j = (a_{min} + j\,a,\ b_{min} + j\,b,\ \ldots,\ g_{min} + j\,g)$
- Note that we have divided each range into equal-sized segments
- Uniformly spaced initial centroids
- Once initial centroids are computed, samples clustered to nearest centroid

Update Centroids

- Suppose cluster *j* has *n* malware samples
- Denote these samples as $m_1, m_2, \ldots, m_n$
- Then the scores are given by

Malware sample	Hidden Markov Models						
	GCC	MinGW	TurboC	Clang	TASM	MWOR	NGVCK
m_1	a_1	b_1	c_1	d_1	e_1	f_1	g_1
m_2	a_2	b_2	c_2	d_2	e_2	f_2	g_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
m_n	a_n	b_n	c_n	d_n	e_n	f_n	g_n

Update Centroids

Malware sample	Hidden Markov Models						
	GCC	MinGW	TurboC	Clang	TASM	MWOR	NGVCK
m_1	a_1	b_1	c_1	d_1	e_1	f_1	g_1
m_2	a_2	b_2	c_2	d_2	e_2	f_2	g_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
m_n	a_n	b_n	c_n	d_n	e_n	f_n	g_n

- Let $a_{mean} = (a_1 + a_2 + \ldots + a_n) / n$
  - And similarly for b_{mean}, c_{mean}, …, g_{mean}
- Then the new centroid is

  $C_j = (a_{mean}, b_{mean}, c_{mean}, \ldots, g_{mean})$
- And thus the name, K-means

Update Clusters

- After all K centroids computed…
- Regroup malware samples, based on nearest centroid
- Recompute centroids (as on previous slide) and regroup…
- Continue until no significant change occurs when regrouping
- Definition of K-means clustering!!!

Experiments

- Performed clustering, K = 2 to K = 15
- Results on next slides…
  - Using all 7 scores
  - Uniform initial centroids
  - And N = 2 hidden states in all HMMs
- Also experimented with
  - Combinations of 7 (or less) scores, uniform vs random initial centroid, N in HMMs, …

Measuring Success

- Can we quantify the quality of results?
- Ideally, each cluster should include only one family (i.e., one color)
  - Here, we focus on 3 dominant families
  - Winwebsec, Zbot, Zeroaccess
- How to measure this?

Measuring Success

- Let $C_1, C_2, \ldots, C_k$ be final the clusters
- Let

  x_i = number of Winwebsec in C_i

  y_i = number of Zbot in C_i

  z_i = number of Zeroaccess in C_i

  $M_i = \max\{x_i, y_i, z_i\}$
- Then define score = $(M_1 + M_2 + \ldots + M_k) / T$
  - Where T is total of Winwebsec, Zbot, and Zeroaccess files

Measuring Success

- Recall, score = $(M_1 + M_2 + \ldots + M_k) / T$
  - Note that $0 < \text{score} \leq 1$
  - And, score = 1 implies all clusters are uniform (wrt 3 major families)
- Suppose we classify simply based on dominant family in a cluster
- Then score = 1 is a perfect result
  - Wrt the three major families

Clusters for Classification

- Our "score" is accuracy if we classify based on dominant type in cluster
  - That is, score samples of unknown type by clustering to nearest centroid
  - And classify by dominant type in cluster
- Previous slide says we can get more than 0.82 correct in this manner
- Good? Bad? Compared to what?

Clusters for Classification

- From previous table
  - 4361 Winwebsec
  - 2136 Zbot
  - 1306 Zeroaccess
  - Total of these three is 7803
- Suppose we expect to see only these 3 families, and at these rates
- Can use expected number to "classify"

Classification

- Classifying based on expected number
- Probability of success is about 0.415
  - Why?
- So, classifying at 0.82 is not too bad
- Is this good enough to be of any use?
  - Much better than "random"
  - But how might we actually use it?

Discussion

- Here HMMs not specific to families
- Results show we get decent results
- Can expect to "classify" previously unseen malware at about these rates
- How could this be useful?
  - Malware may be similar to that in cluster
  - So, possibly faster analysis/response
- Also relevant to classification/naming

Conclusion

- HMM scoring scheme able to classify unrelated malware with good accuracy
  - Malware is unrelated to scores used
- Not accurate enough for detection
  - Only 82% accuracy in this work
- But, other potential uses
  - As an aid in analysis of new malware
  - As a tool for classification/naming