

# Data Driven Transfer Clustering

Maria-Florina (Nina) Balcan

Carnegie Mellon University

# Clustering Problems

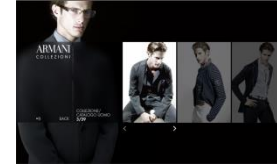
**Goal:** Automatically partition **unlabeled** data into groups of similar datapoints.

**Useful for:**

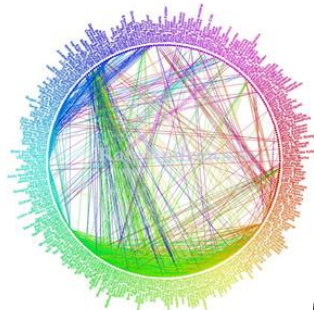
- Automatically organizing data.
- Understanding hidden structure in data.
- Preprocessing for further analysis.
  - Representing high-dimensional data in a low-dimensional space (e.g., for visualization purposes).

# Applications (Clustering comes up everywhere...)

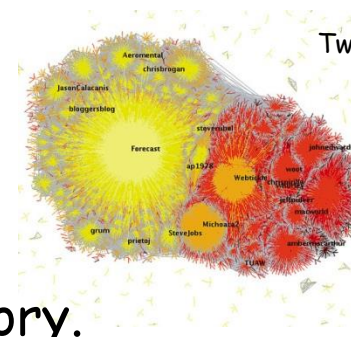
- Cluster news articles or web pages or search results by topic.



- Cluster users of social networks by interest (community detection).



Facebook network



Twitter Network

- Cluster customers according to purchase history.



- And many many more applications....

# Clustering Problems

**Clustering:** Given a set objects organize them into natural groups.

**Classically, one shot clustering:** cluster a one-time dataset well.

Major impossibility results:

- not clear what similarity or objective to use to recover a good clustering for a given dataset.
- even when similarity & objective naturally specified, optimally solving the underlying combinatorial problem is intractable.

# Data Driven Transfer Clustering

## Our work:

Often need do solve such problems repeatedly.

- E.g., clustering news articles (Google news).

**Data driven clustering: use learning & data for algo design.**

- Different methods work better in different settings.
- Large family of methods - what's best in our application?

**Both improved performance and formal guarantees.**

# Outline of the talk

- Motivation.
- Classic one-shot formulation.
  - Lloyd's methods.
  - Linkage techniques.
- Our new approach: data-driven transfer clustering.  
[Hyper-parameter tuning or meta-learning for clustering.]

[Balcan-Nagarajan-Vitercik-White, COLT 2017] [Balcan-Dick-Vitercik, FOCS'18]

[Balcan-Dick-Lang, Arxiv 2019] [Balcan-Dick-White, NeurIPS 2018]

# Clustering: Objective Based Formulations

**Clustering:** Given a set of objects organize them into natural groups.

## Objective based clustering

### *k*-means

Input: Set of objects  $S$ ,  $d$

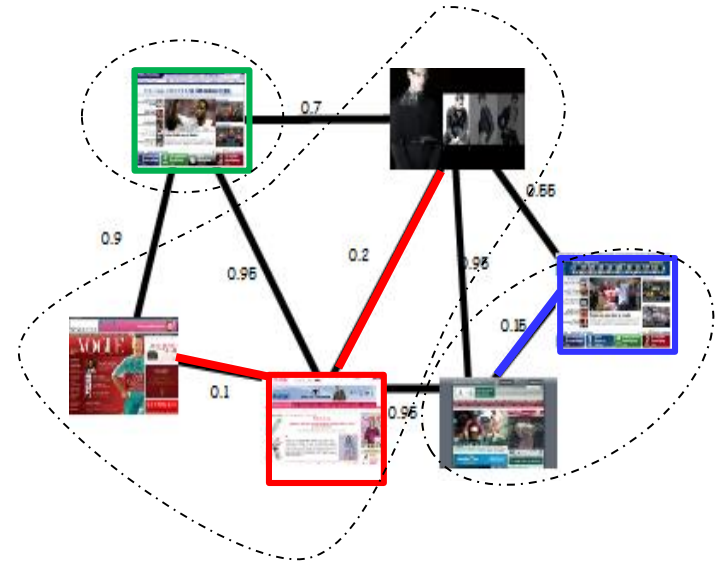
Output: centers  $\{c_1, c_2, \dots, c_k\}$

To minimize  $\sum_p \min_i d^2(p, c_i)$

**k-median:**  $\min \sum_p \min d(p, c_i)$ .

**k-center/facility location:** minimize the maximum radius.

- Finding OPT is NP-hard, so no universal efficient algo that works on all domains.



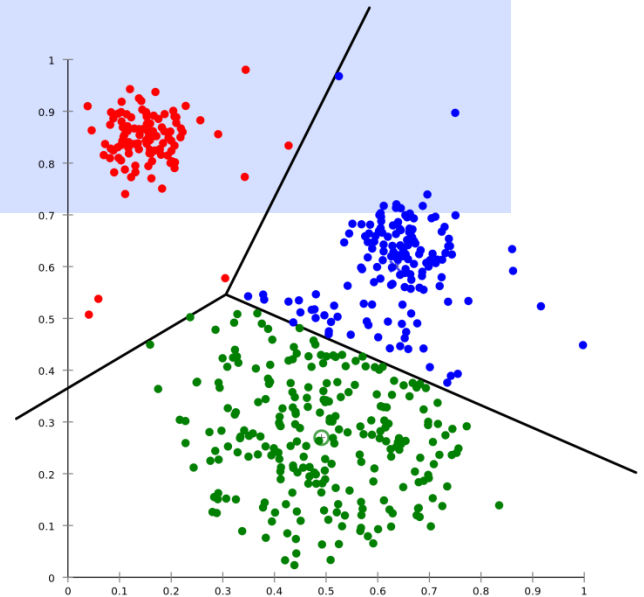
# Euclidean k-means Clustering

**Input:** A set of  $n$  datapoints  $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$  in  $\mathbb{R}^d$   
target #clusters  $k$

**Output:**  $k$  representatives  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in \mathbb{R}^d$

**Objective:** choose  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in \mathbb{R}^d$  to minimize

$$\sum_{i=1}^n \min_{j \in \{1, \dots, k\}} \left\| \mathbf{x}^i - \mathbf{c}_j \right\|^2$$





# Euclidean k-means Clustering

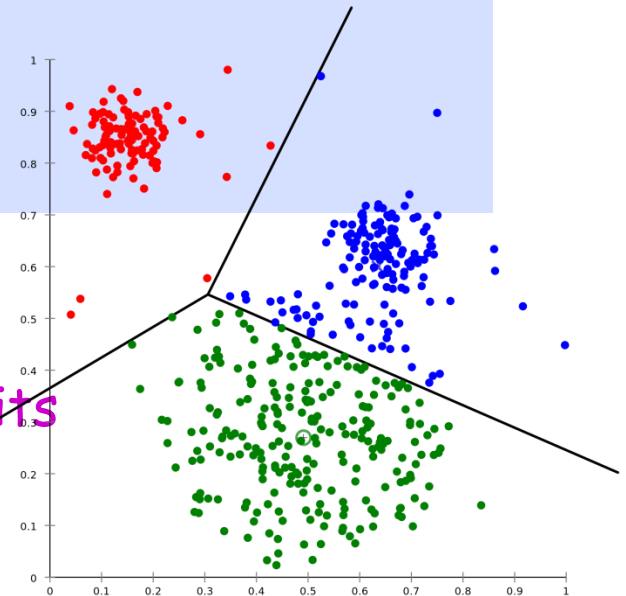
**Input:** A set of  $n$  datapoints  $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$  in  $\mathbb{R}^d$   
target #clusters  $k$

**Output:**  $k$  representatives  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in \mathbb{R}^d$

**Objective:** choose  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in \mathbb{R}^d$  to minimize

$$\sum_{i=1}^n \min_{j \in \{1, \dots, k\}} \|\mathbf{x}^i - \mathbf{c}_j\|^2$$

Natural assignment: each point assigned to its closest center, leads to a Voronoi partition.



# Common Heuristic in Practice: The Lloyd's method

[Least squares quantization in PCM, Lloyd, IEEE Transactions on Information Theory, 1982]

**Input:** A set of  $n$  datapoints  $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$  in  $\mathbb{R}^d$

**Initialize** centers  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in \mathbb{R}^d$  and  
clusters  $C_1, C_2, \dots, C_k$  in any way.

**Repeat** until there is no further change in the cost.

- For each  $j$ :  $C_j \leftarrow \{x \in S \text{ whose closest center is } \mathbf{c}_j\}$
- For each  $j$ :  $\mathbf{c}_j \leftarrow \text{mean of } C_j$

# Common Heuristic in Practice: The Lloyd's method

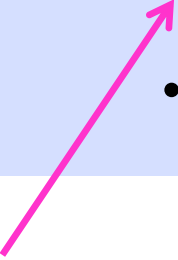
[Least squares quantization in PCM, Lloyd, IEEE Transactions on Information Theory, 1982]

**Input:** A set of  $n$  datapoints  $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$  in  $\mathbb{R}^d$


**Initialize** centers  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in \mathbb{R}^d$  and  
clusters  $C_1, C_2, \dots, C_k$  in any way.

**Repeat** until there is no further change in the cost.

- For each  $j$ :  $C_j \leftarrow \{x \in S \text{ whose closest center is } \mathbf{c}_j\}$
- For each  $j$ :  $\mathbf{c}_j \leftarrow \text{mean of } C_j$



Holding  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k$  fixed,  
pick optimal  $C_1, C_2, \dots, C_k$



Holding  $C_1, C_2, \dots, C_k$  fixed,  
pick optimal  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k$

# Initialization for the Lloyd's method

**Input:** A set of  $n$  datapoints  $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$  in  $\mathbb{R}^d$

**Initialize** centers  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in \mathbb{R}^d$  and  
clusters  $C_1, C_2, \dots, C_k$  in any way.

**Repeat** until there is no further change in the cost.

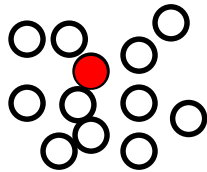
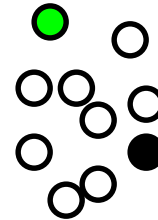
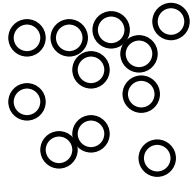
- For each  $j$ :  $C_j \leftarrow \{x \in S \text{ whose closest center is } \mathbf{c}_j\}$
- For each  $j$ :  $\mathbf{c}_j \leftarrow \text{mean of } C_j$

- **Initialization is crucial** (how fast it converges, quality of solution output)
- Techniques commonly used in practice
  - Random centers from the datapoints (repeat a few times)
  - Furthest traversal
  - K-means ++ (has worst case provable guarantees)

# Initialization is Crucial for Lloyd's

Instance 1: well separated Gaussian clusters.

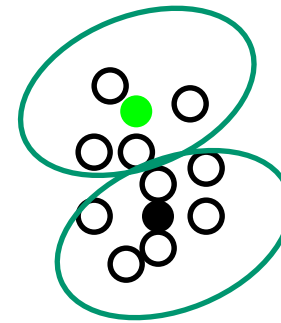
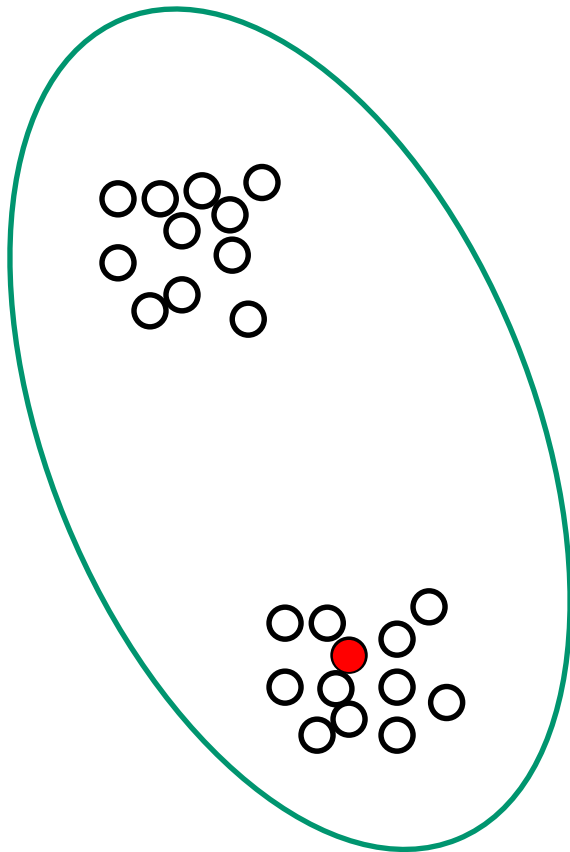
Bad case for random initialization



# Initialization is Crucial for Lloyd's

Instance 1: well separated Gaussian clusters.

Bad case for random initialization



Some Gaussian are combined.....

# Furthest Point Initialization

Choose  $\mathbf{c}_1$  arbitrarily (or at random).

- For  $j = 2, \dots, k$ 
  - Pick  $\mathbf{c}_j$  among datapoints  $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$  that is farthest from previously chosen  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{j-1}$

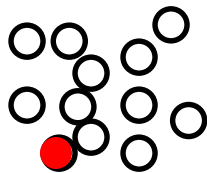
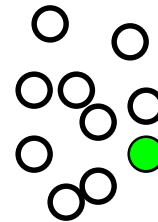
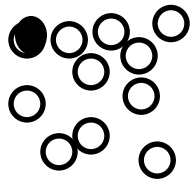
Fixes the Gaussian problem. But it can be thrown off by outliers....

# Initialization is Crucial for Lloyd's

Instance 1: well separated Gaussian clusters.

Bad case for random initialization

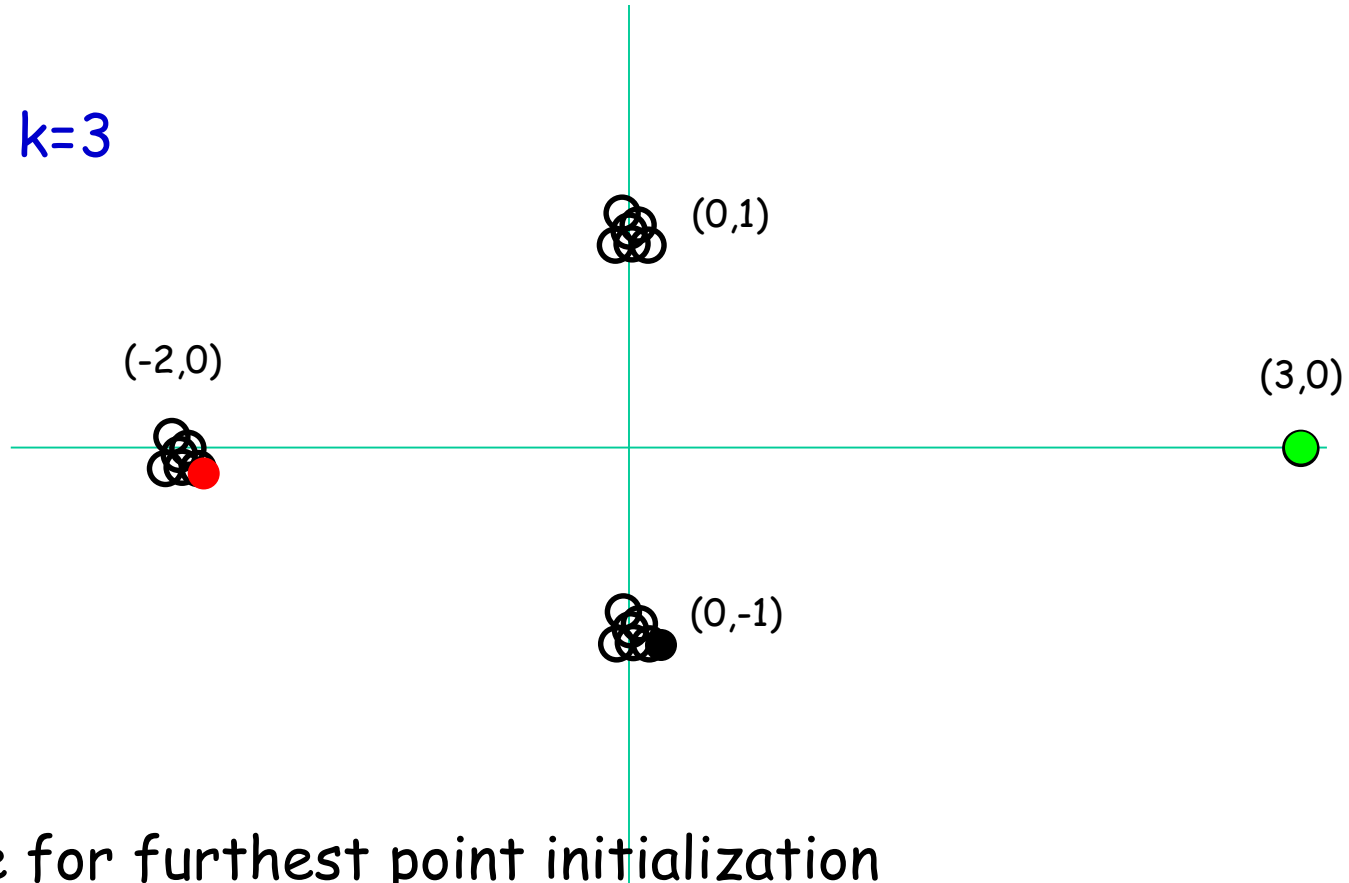
Good case for furthest point initialization





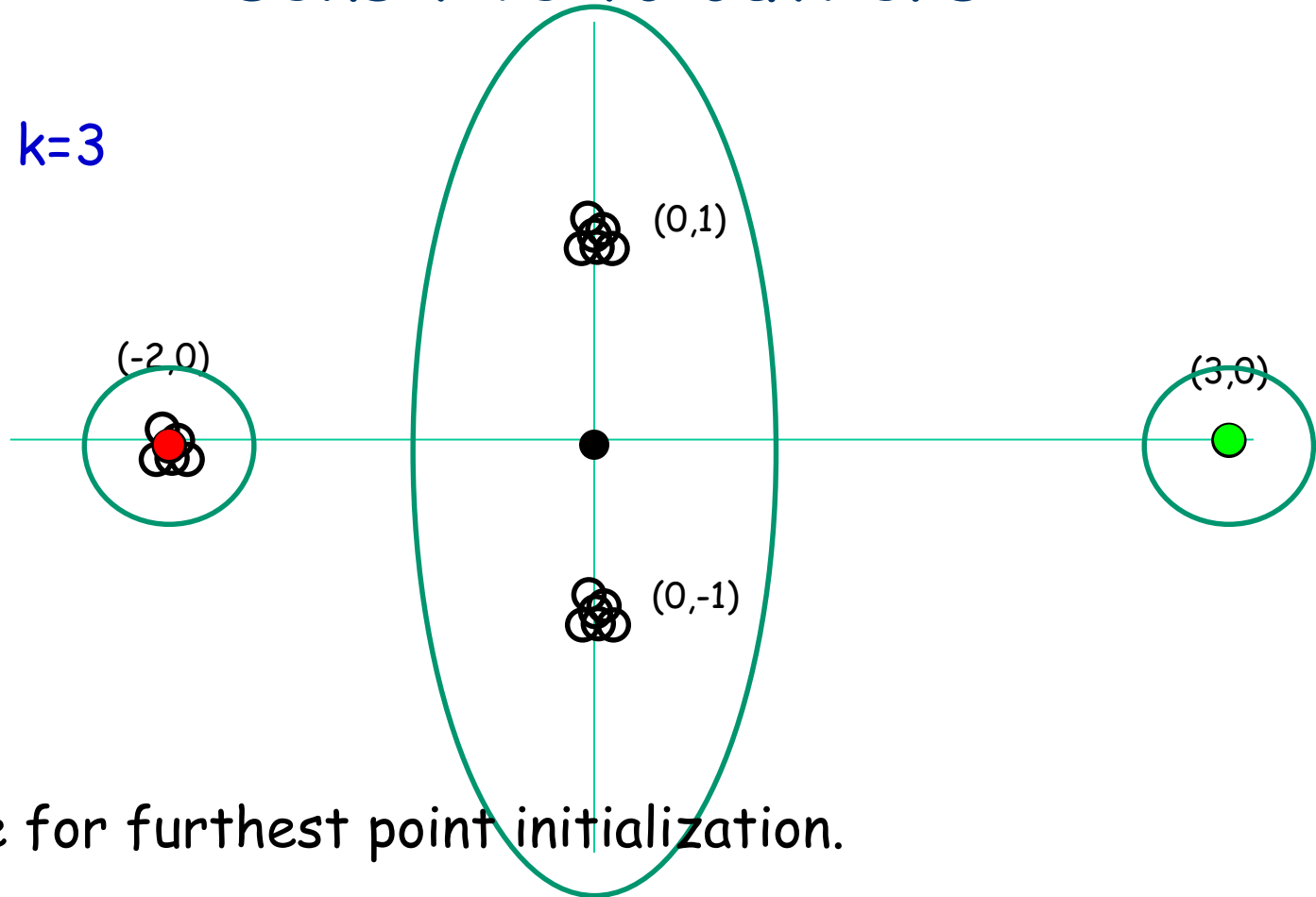
# Furthest point initialization heuristic sensitive to outliers

Assume  $k=3$



# Furthest point initialization heuristic sensitive to outliers

Assume  $k=3$



Bad case for furthest point initialization.

Also bad case for random initialization.

# K-means++ Initialization: $D^2$ sampling [AV07]

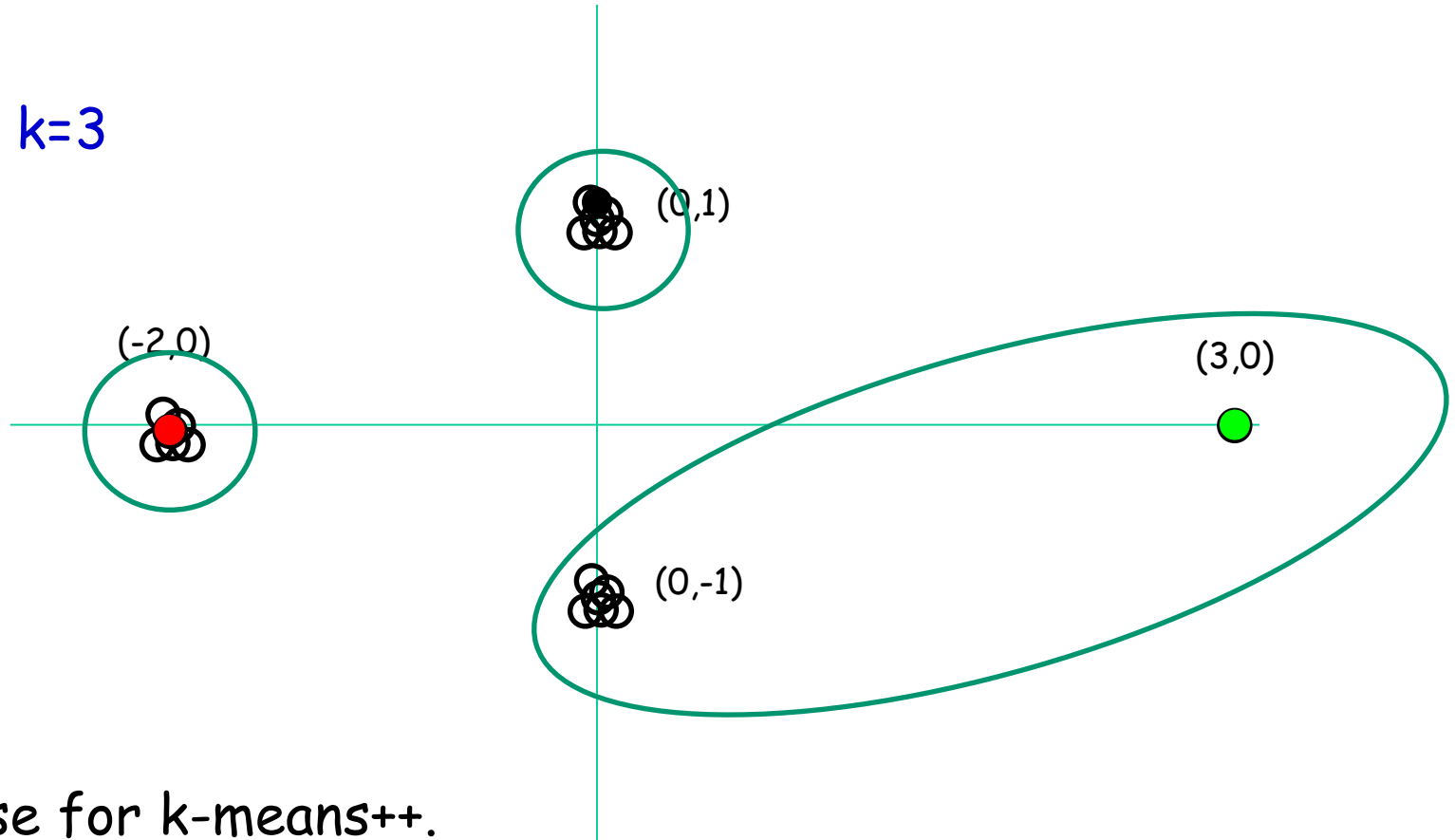
- Interpolate between random and furthest point initialization
- Let  $D(\mathbf{x})$  be the distance between a point  $x$  and its nearest center. Chose the next center proportional to  $D^2(\mathbf{x})$ .

- Choose  $\mathbf{c}_1$  at random.
- For  $j = 2, \dots, k$ 
  - Pick  $\mathbf{c}_j$  among  $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$  according to the distribution

$$\Pr(\mathbf{c}_j = \mathbf{x}^i) \propto \min_{j' < j} \left\| \mathbf{x}^i - \mathbf{c}_{j'} \right\|^2 D^2(\mathbf{x}^i)$$

# A good case for k-means++

Assume  $k=3$



Good case for k-means++.

# K-means++ Initialization: $D^2$ sampling [AV07]

- Interpolate between random and furthest point initialization
- Let  $D(x)$  be the distance between a point  $x$  and its nearest center. Chose the next center proportional to  $D^2(x)$ .

- Choose  $c_1$  at random.
- For  $j = 2, \dots, k$ 
  - Pick  $c_j$  among  $x^1, x^2, \dots, x^n$  according to the distribution

$$\Pr(c_j = x^i) \propto \min_{j' < j} \|x^i - c_{j'}\|^2 D^2(x^i)$$

**Theorem:** K-means++ always attains an  $O(\log k)$  approximation to optimal k-means solution in expectation.

Running Lloyd's can only further improve the cost.

# Our Work: Learn $\alpha$ for $D^\alpha$ sampling

- Interpolate between random and furthest point initialization
- **Parametrized Lloyd's:** let  $D(\mathbf{x})$  be the distance between a point  $x$  and its nearest center. Chose the next center proportional to  $D^\alpha(\mathbf{x})$ .
  - $\alpha = 0$ , random sampling
  - $\alpha = \infty$ , furthest point
  - $\alpha = 2$ , k-means++

# Clustering: Minimize Distance to Ground-truth

**Clustering:** Given a set of objects organize them into natural groups.

**Unsupervised learning: minimize distance to ground-truth**

[sports]



[fashion]



Set of  $n$  objects. [documents, web pages]

"ground truth" clustering  $C_1, C_2, \dots, C_k$  [true clustering by topic]

Goal: clustering  $C'_1, C'_2, \dots, C'_k$  of low error.

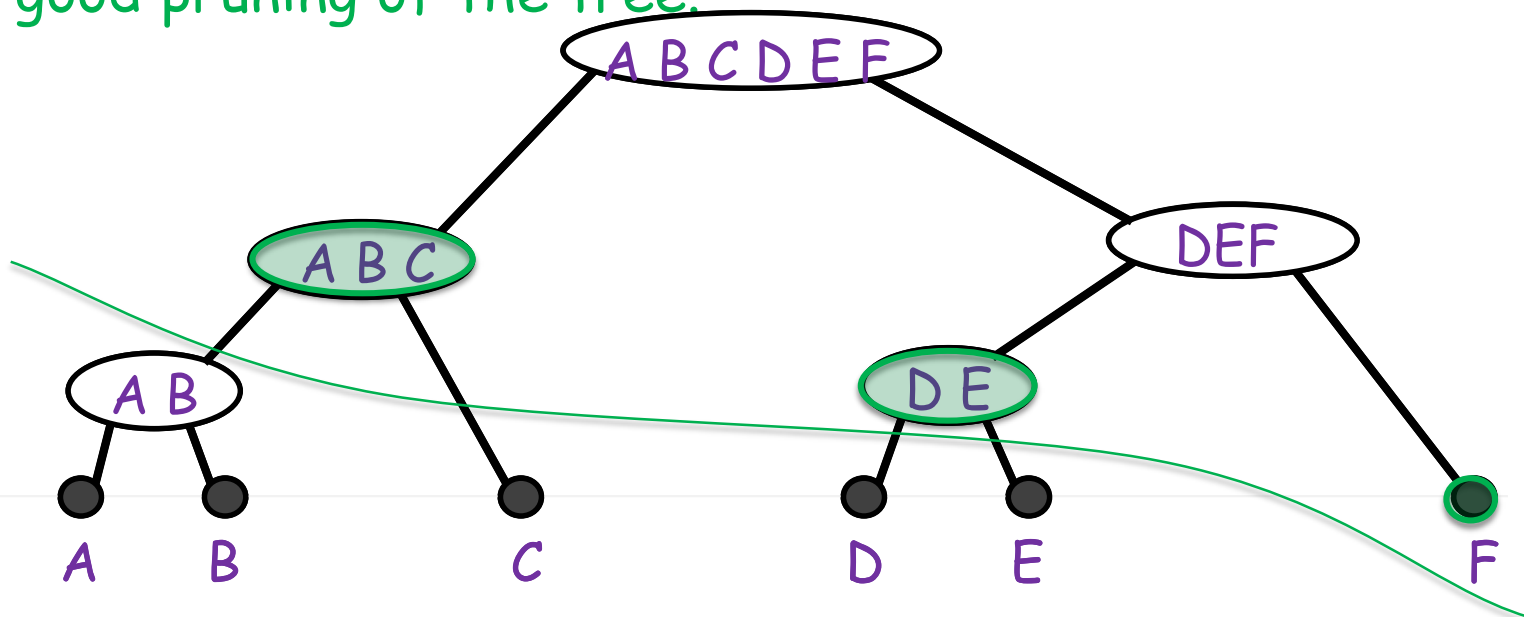
$$\text{error}(C') = \min_{\sigma \in S_k} \frac{1}{n} \sum_{i=1}^k |C_i - C'_{\sigma(i)}|$$

fraction of pts misclassified up to re-indexing of clusters

# Clustering: Linkage + Post-processing

Family of poly time 2-stage algorithms:

1. Use a greedy linkage-based algorithm to organize data into a hierarchy (tree) of clusters.
2. A fixed procedure over this tree (e.g., dynamic programming or output clusters induced by the last k-merges) to identify a good pruning of the tree.

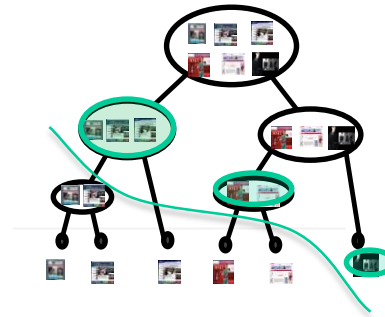
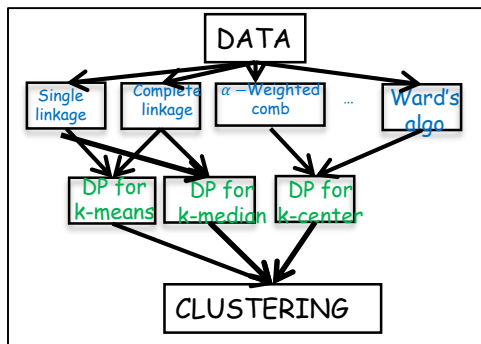




# Clustering: Linkage + Post-processing

1. Use a linkage-based algorithm to get a hierarchy.
2. Post-processing to identify a good pruning.

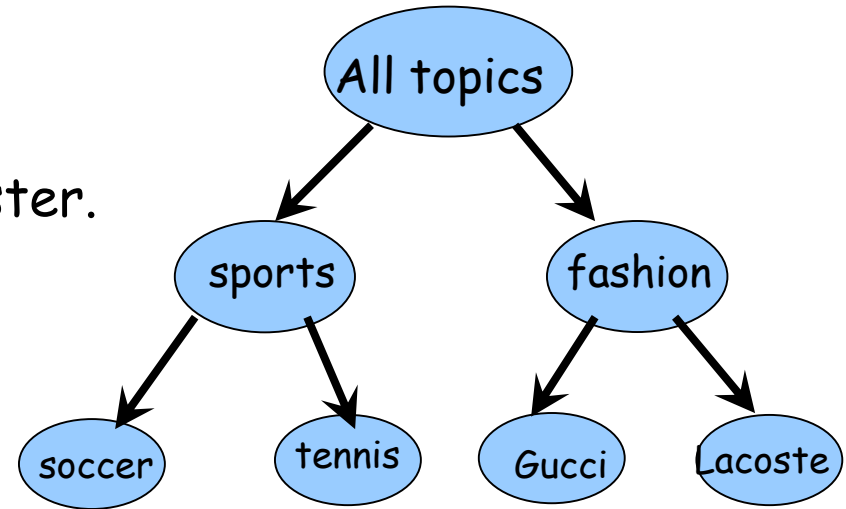
Both steps can be done efficiently.



# Linkage Procedures for Hierarchical Clustering

## Bottom-Up (agglomerative)

- Start with every point in its own cluster.
- Repeatedly merge the "closest" two clusters.



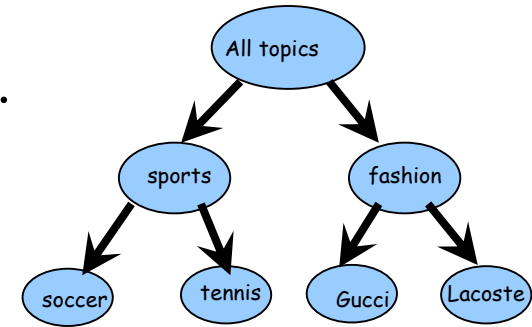
Different defs of "closest" give different algorithms.

# Linkage Procedures for Hierarchical Clustering

Have a **distance** measure on pairs of objects.

$d(x,y)$  - distance between  $x$  and  $y$

E.g., # keywords in common, edit distance, etc

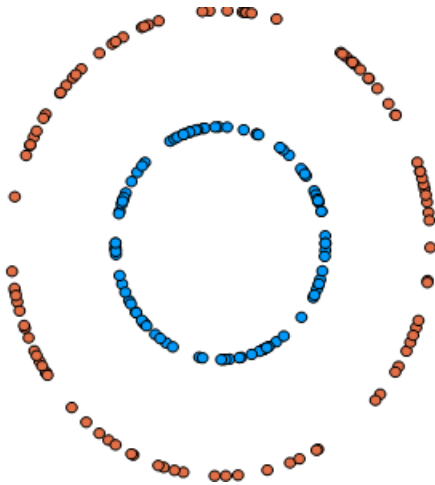


- Single linkage:  $\text{dist}(A, B) = \min_{x \in A, x' \in B} \text{dist}(x, x')$
- Complete linkage:  $\text{dist}(A, B) = \max_{x \in A, x' \in B} \text{dist}(x, x')$

# Different Algos Work in Different Settings

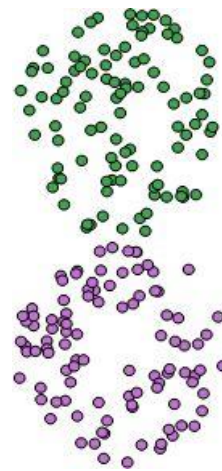
Assume 2 target clusters.

Instance 1



Easy for Single-Linkage  
Hard for Complete-Linkage

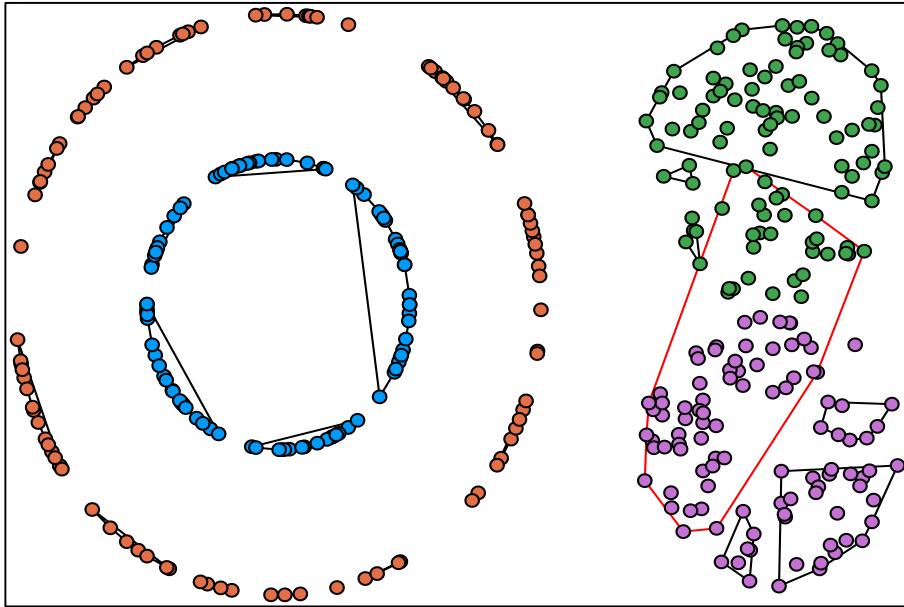
Instance 2



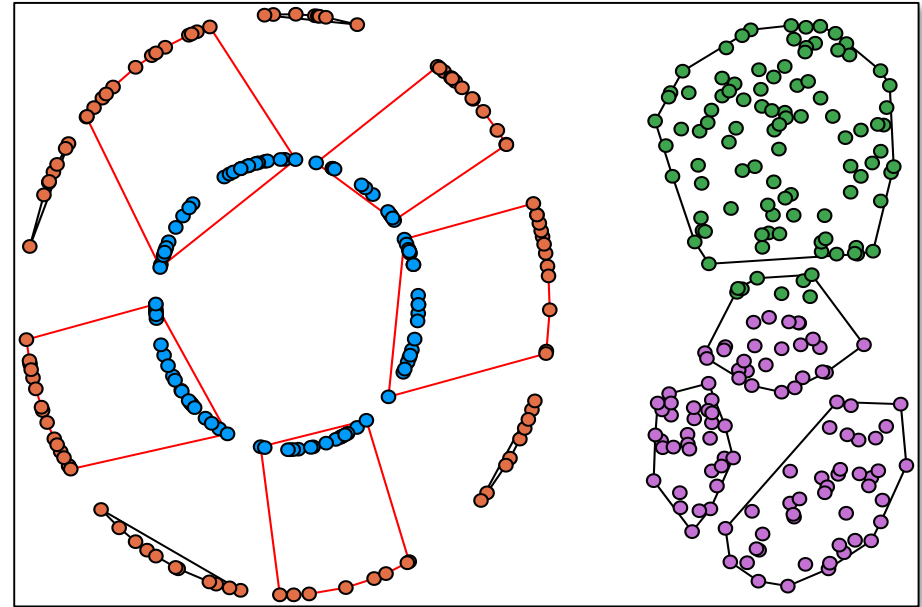
Hard for Single Linkage  
Easy for Complete-Linkage

# Different Algos Work in Different Settings

Single Linkage After 370 Merges



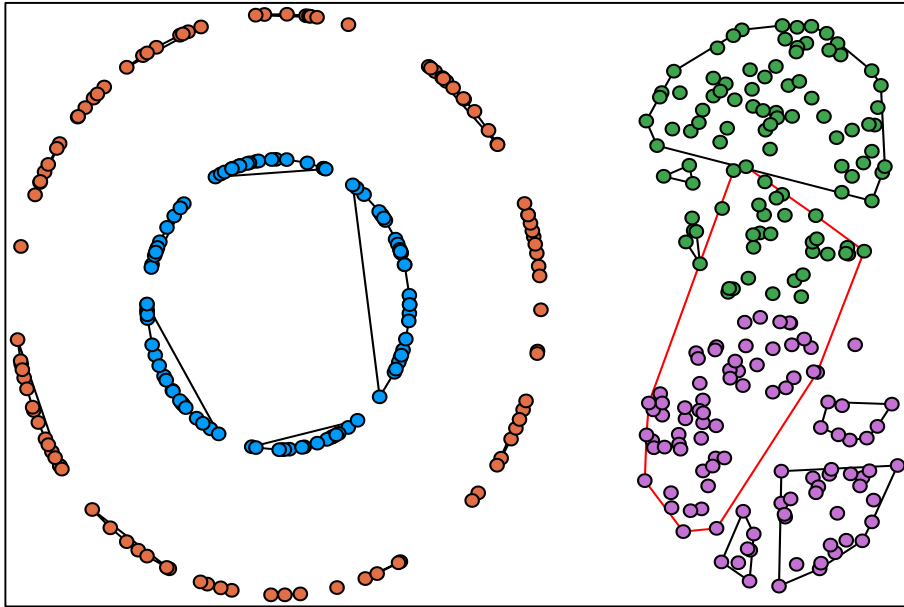
Complete Linkage After 389 Merges



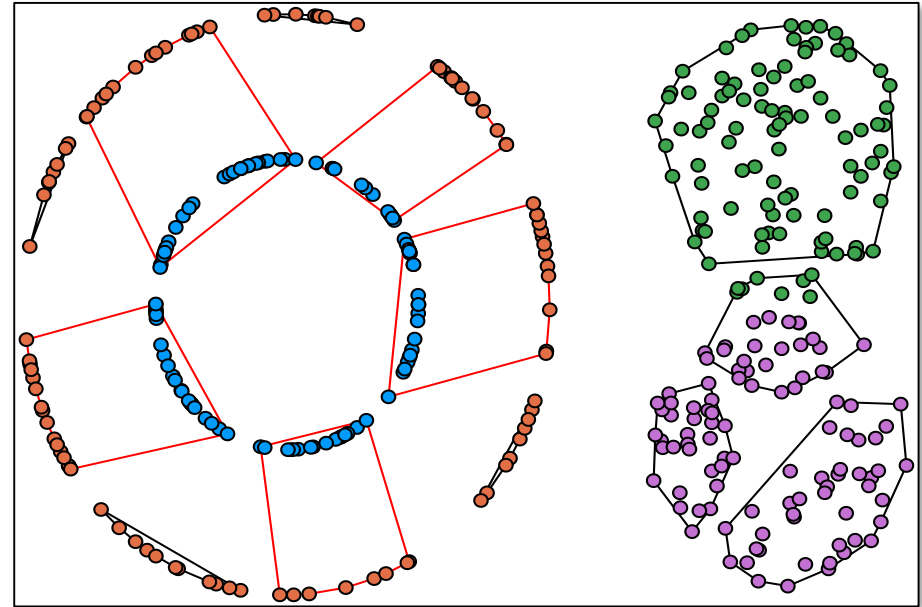
- SL does well on instance 1: finds smallest gap in deciding what to merge next, and so never mistakenly connects a red cluster & a blue cluster.
- SL fails on instance 2: green and purple disks are close, SL makes some mistaken connections between green and purple early on (large cluster outlined by the red boundary that has many points from each), causing a large error.

# Different Algos Work in Different Settings

Single Linkage After 370 Merges



Complete Linkage After 389 Merges



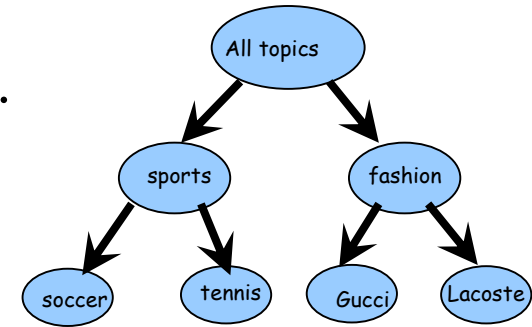
- CL fails on instance 1: tries to keep the diameter of clusters as small as possible (it always merges the two clusters such that the diameter of the merged cluster is as small as possible).

# Linkage Procedures for Hierarchical Clustering

Have a **distance** measure on pairs of objects.

$d(x,y)$  - distance between  $x$  and  $y$

E.g., # keywords in common, edit distance, etc

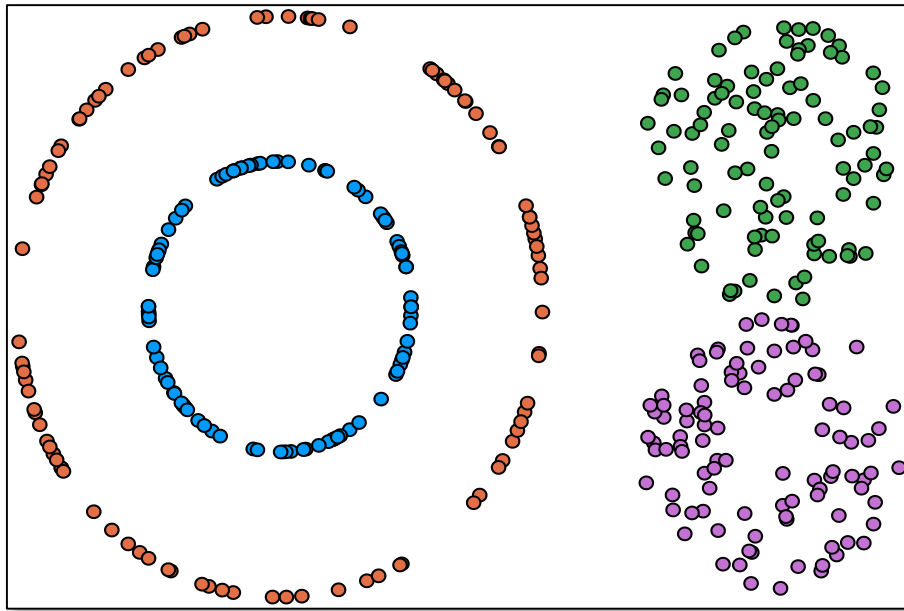


- Single linkage:  $\text{dist}(A, B) = \min_{x \in A, x' \in B} \text{dist}(x, x')$
- Complete linkage:  $\text{dist}(A, B) = \max_{x \in A, x' \in B} \text{dist}(x, x')$
- Parametrized family,  $\alpha$ -weighted linkage:

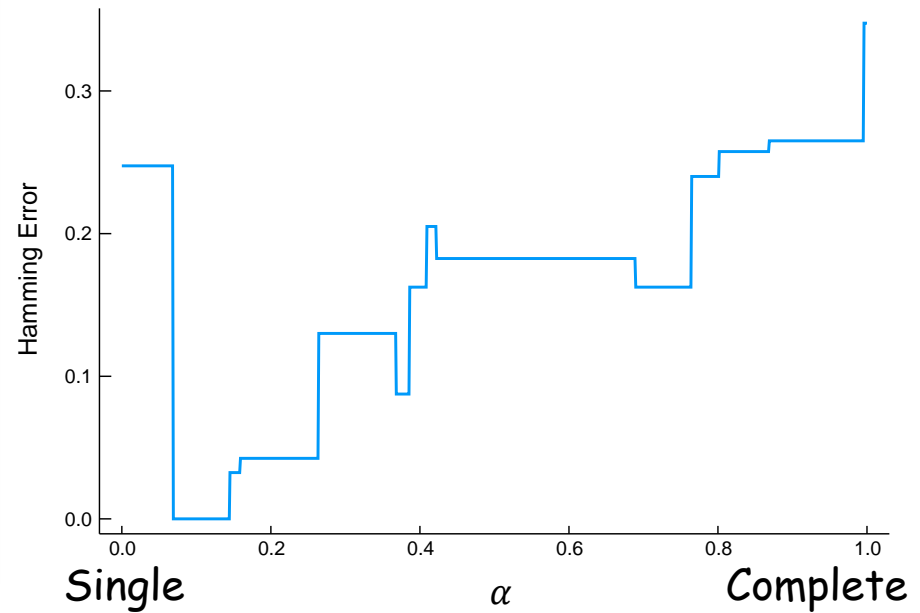
$$\text{dist}(A, B) = (1 - \alpha) \min_{x \in A, x' \in B} \text{dist}(x, x') + \alpha \max_{x \in A, x' \in B} \text{dist}(x, x')$$

# Different Algos Work in Different Settings

Instance 3: assume 4 target clusters.



Single Linkage Error: 0.25  
Complete Linkage Error: 0.35

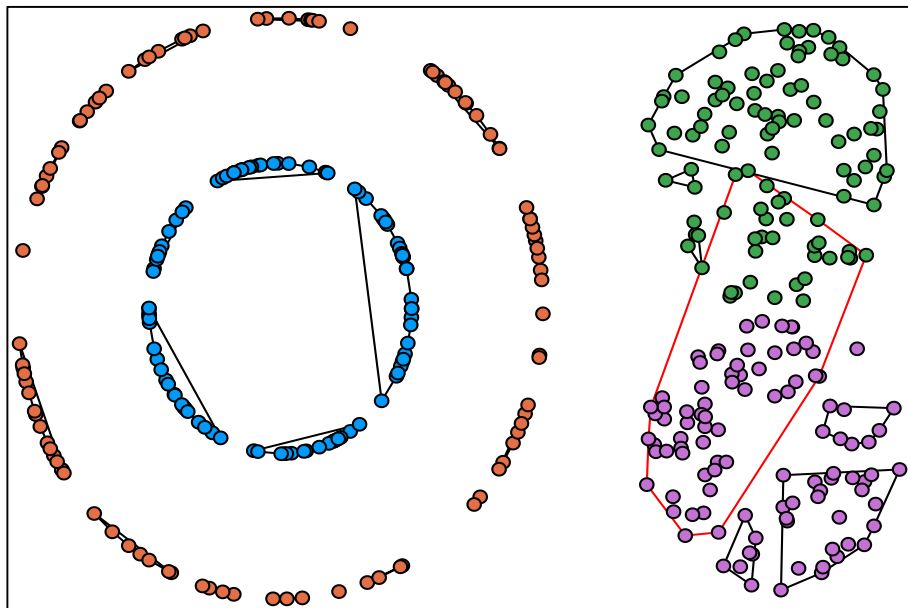


A different algo discusses later gives error 0



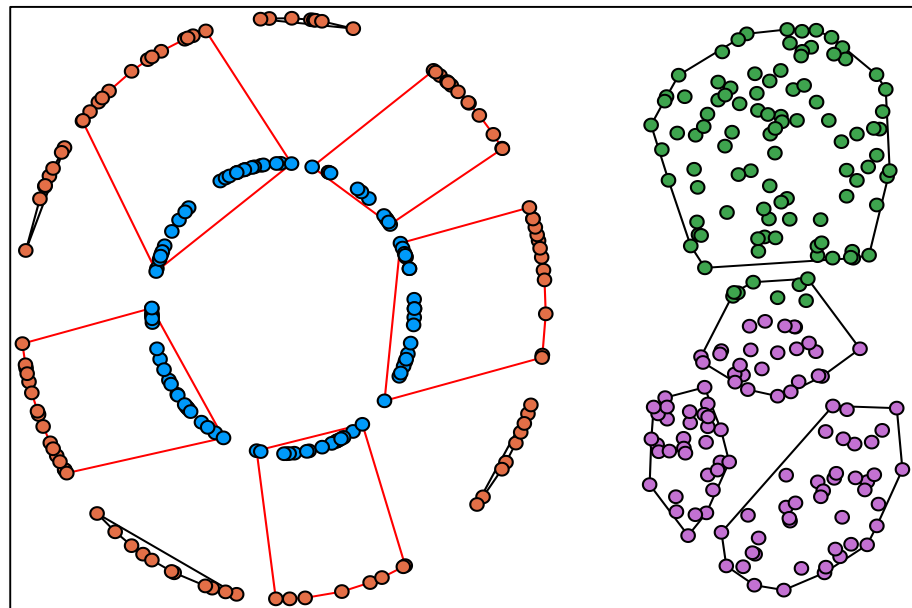
# Synthetic Example

Single Linkage After 370 Merges



"Chaining" across disc clusters.

Complete Linkage After 389 Merges



Can't separate the two rings.

For  $\alpha = 0.12$ :

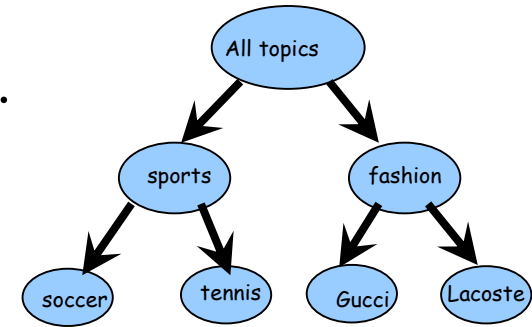
- Closer to single linkage - does well on rings.
- Large-diameter merges penalized  $\rightarrow$  long chains penalized.

# Our Work: Learn $\alpha$ for $\alpha$ -weighted linkage

Have a **distance** measure on pairs of objects.

$d(x,y)$  - distance between  $x$  and  $y$

E.g., # keywords in common, edit distance, etc



- Single linkage:  $\text{dist}(A, B) = \min_{x \in A, x' \in B} \text{dist}(x, x')$
- Complete linkage:  $\text{dist}(A, B) = \max_{x \in A, x' \in B} \text{dist}(x, x')$
- Parametrized family,  $\alpha$ -weighted linkage:

$$\text{dist}(A, B) = (1 - \alpha) \min_{x \in A, x' \in B} \text{dist}(x, x') + \alpha \max_{x \in A, x' \in B} \text{dist}(x, x')$$

# Our Work: Data Driven Transfer Clustering

Empirically, different methods work better in different settings.

Large family of methods - what's best in our application?

Often need to solve clustering problems repeatedly.

- E.g., clustering news articles (Google news).

**Data driven clustering: use learning & data for algo design.**

- Overcomes major impossibility results for one shot clustering.

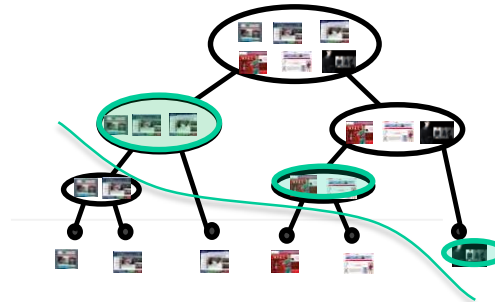
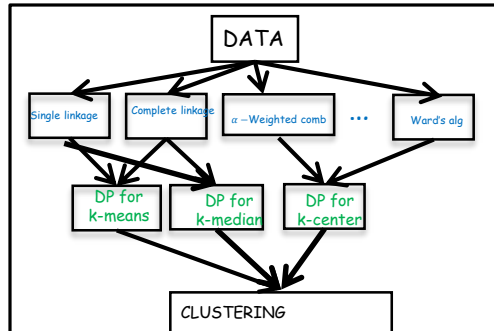
# Data Driven Transfer Clustering

**Our work:** New techniques for data-driven clustering with guarantees.

- Linkage + Post Processing

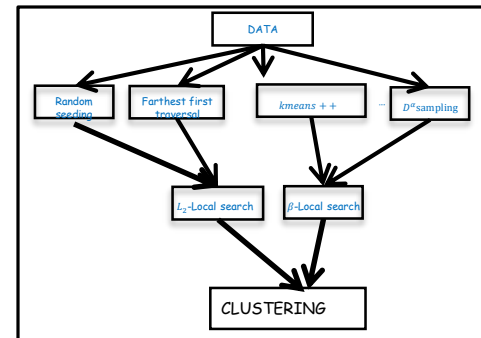
[Balcan-Nagarajan-Vitercik-White, COLT 2017] [Balcan-Dick-Vitercik, FOCS'18]

[Balcan-Dick-Lang, Arxiv 2019]



- Parametrized Lloyds methods

[Balcan-Dick-White, NeurIPS 2018]

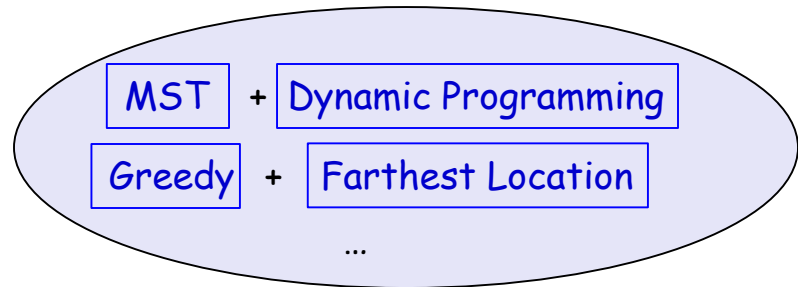


# Algorithm Selection as a Learning Problem

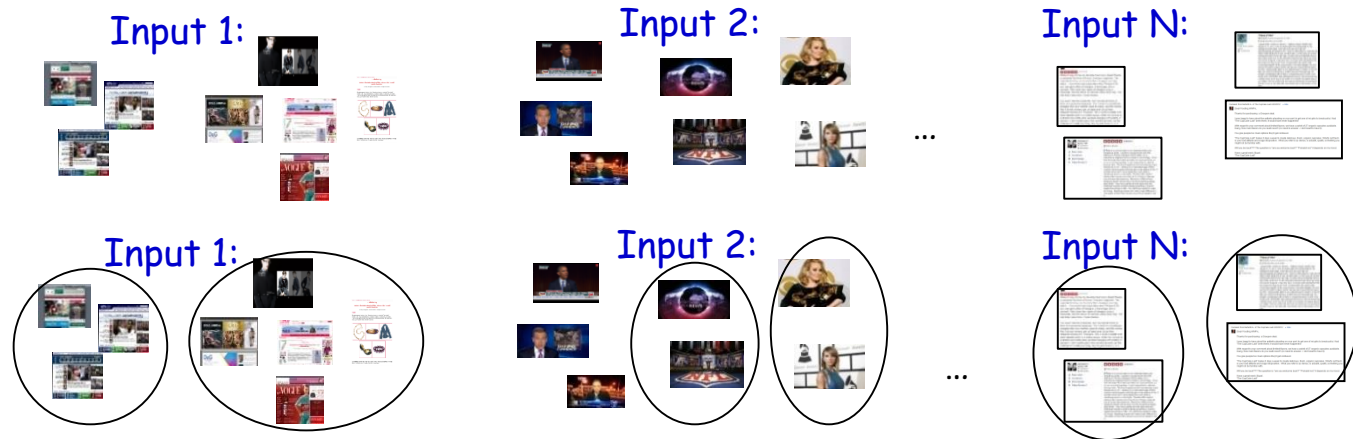
**Goal:** given family of algos  $F$ , sample of typical instances from domain (unknown distr.  $D$ ), find algo that performs well on new instances from  $D$ .

Large family  $F$  of algorithms

Sample of typical inputs



Clustering:



...

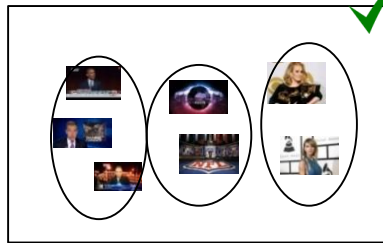
# Sample Complexity of Algorithm Selection

**Goal:** given family of algos  $\mathcal{F}$ , sample of typical instances from domain (unknown distr.  $\mathcal{D}$ ), find algo that performs well on new instances from  $\mathcal{D}$ .

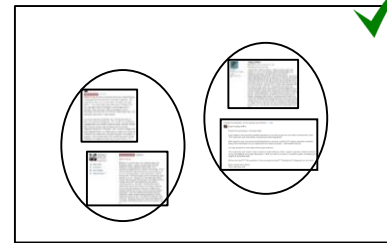
**Approach:** ERM, find  $\hat{A}$  near optimal algorithm over the set of samples.

**Key Question:** Will  $\hat{A}$  do well on future instances?

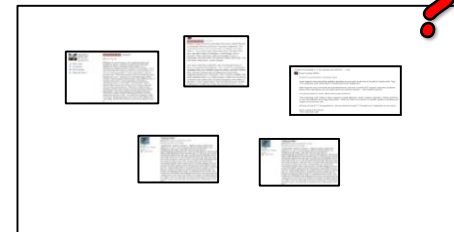
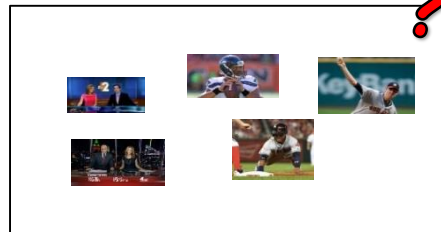
Seen:



...



New:



**Sample Complexity:** How large should our sample of typical instances be in order to guarantee good performance on new instances?

# Sample Complexity of Algorithm Selection

**Goal:** given family of algos  $\mathbf{F}$ , sample of typical instances from domain (unknown distr.  $\mathbf{D}$ ), find algo that performs well on new instances from  $\mathbf{D}$ .

**Approach:** ERM, find  $\hat{\mathbf{A}}$  near optimal algorithm over the set of samples.

**Key tools from learning theory**

- **Uniform convergence:** for any algo in  $\mathbf{F}$ , average performance over samples "close" to its expected performance.
  - Imply that  $\hat{\mathbf{A}}$  has high expected performance.
  - $N = O(\dim(\mathbf{F}) / \epsilon^2)$  instances suffice for  $\epsilon$ -close.

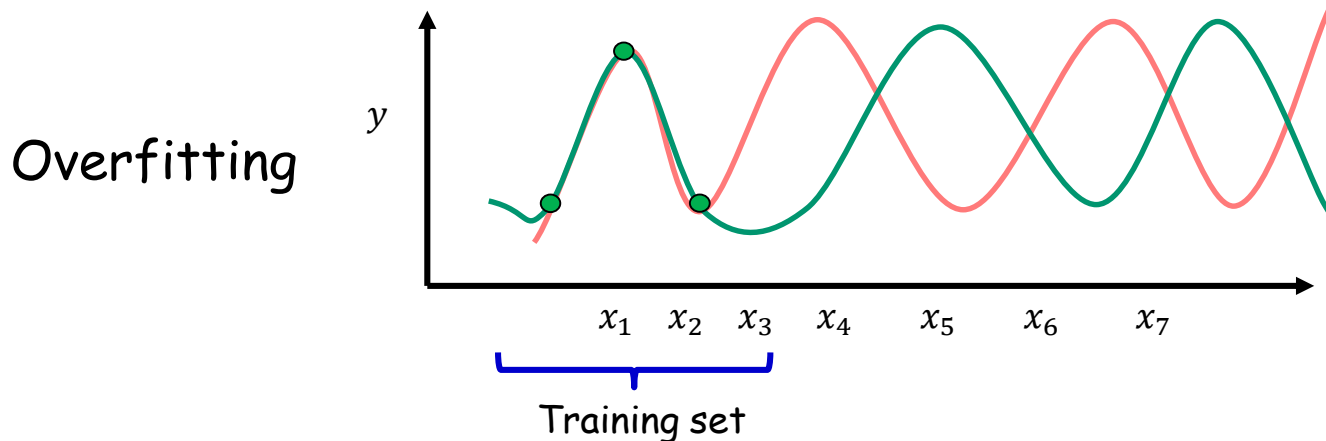
# Sample Complexity of Algorithm Selection

**Goal:** given family of algos  $\mathbf{F}$ , sample of typical instances from domain (unknown distr.  $\mathbf{D}$ ), find algo that performs well on new instances from  $\mathbf{D}$ .

**Key tools from learning theory**

$N = O(\dim(\mathbf{F}) / \epsilon^2)$  instances suffice for  $\epsilon$ -close.

$\dim(\mathbf{F})$  (e.g. pseudo-dimension): ability of fns in  $\mathbf{F}$  to fit complex patterns





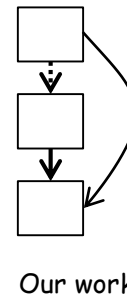
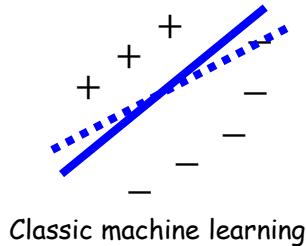
# Sample Complexity of Algorithm Selection

**Goal:** given family of algos  $\mathbf{F}$ , sample of typical instances from domain (unknown distr.  $\mathbf{D}$ ), find algo that performs well on new instances from  $\mathbf{D}$ .

**Key tools from learning theory**

$N = O(\dim(\mathbf{F}) / \epsilon^2)$  instances suffice for  $\epsilon$ -close.

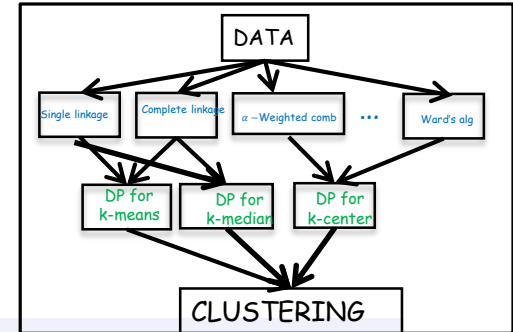
**Challenge:** analyze  $\dim(\mathbf{F})$ , due to combinatorial & modular nature, "nearby" programs/algos can have drastically different behavior.



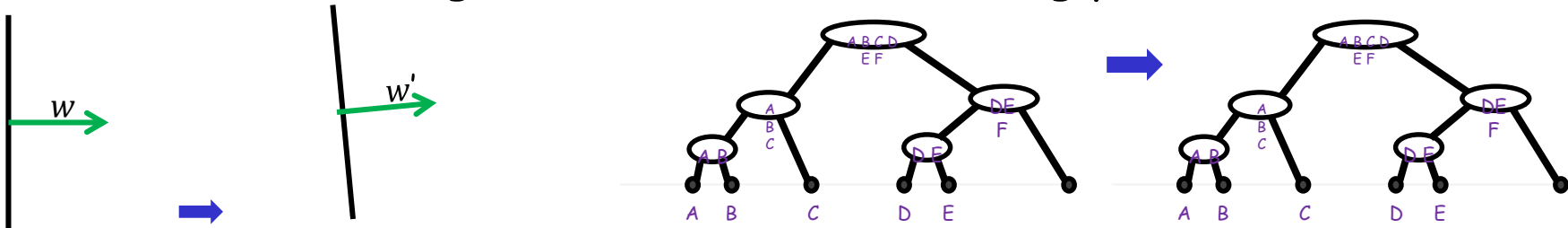
# Clustering: Linkage + Dynamic Programming

## Our Results: $\alpha$ -weighted linkage+DP

- Pseudo-dimension is  $O(\log n)$ , so small sample complexity.
- Given sample  $S$ , find best algo from this family in poly time.



**Key Technical Challenge:** small changes to the parameters of the algo can lead to radical changes in the tree or clustering produced.

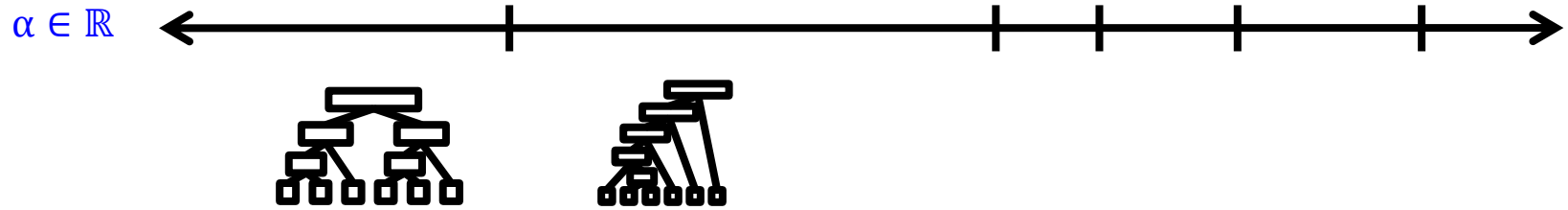


Problem: a single change to an early decision by the linkage algo, can snowball and produce large changes later on.

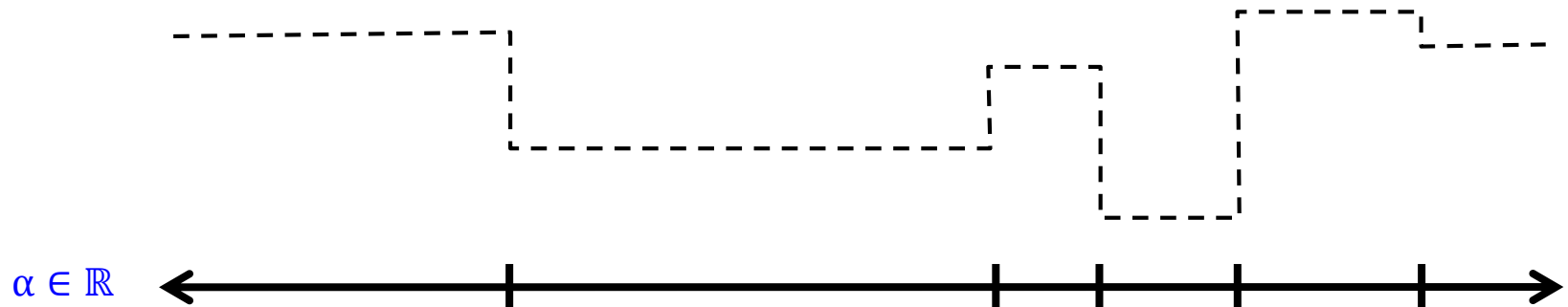
# Clustering: Linkage + Dynamic Programming

**Claim:** Pseudo-dimension of  $\alpha$ -weighted linkage + DP is  $O(\log n)$ , so small sample complexity.

**Key fact:** If we fix a clustering instance of  $n$  pts and vary  $\alpha$ , at most  $O(n^8)$  switching points where behavior on that instance changes.



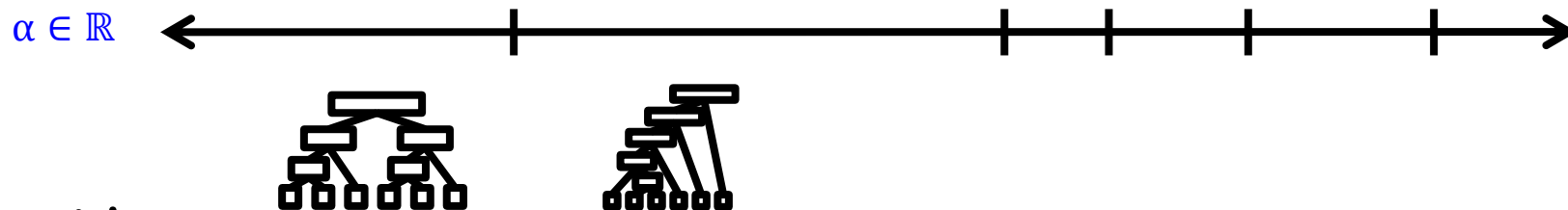
So, the cost function is piecewise-constant with at most  $O(n^8)$  pieces.



# Clustering: Linkage + Dynamic Programming

**Claim:** Pseudo-dimension of  $\alpha$ -weighted linkage + DP is  $O(\log n)$ , so small sample complexity.

**Key fact:** If we fix a clustering instance of  $n$  pts and vary  $\alpha$ , at most  $O(n^8)$  switching points where behavior on that instance changes.



**Key idea:**

- For a given  $\alpha$ , which will merge first,  $\mathcal{N}_1$  and  $\mathcal{N}_2$ , or  $\mathcal{N}_3$  and  $\mathcal{N}_4$ ?

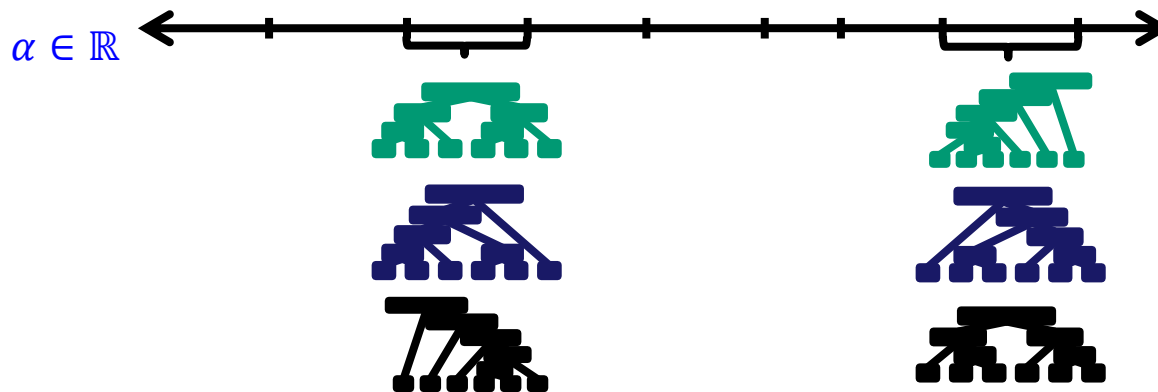


- Depends on which of  $(1 - \alpha)d(p, q) + \alpha d(p', q')$  or  $(1 - \alpha)d(r, s) + \alpha d(r', s')$  is smaller.
- An interval boundary an equality for 8 points, so  $O(n^8)$  interval boundaries.

# Clustering: Linkage + Dynamic Programming

**Claim:** Pseudo-dimension of  $\alpha$ -weighted linkage + DP is  $O(\log n)$ , so small sample complexity.

**Key idea:** For  $m$  clustering instances of  $n$  points,  $O(mn^8)$  patterns.



- Pseudo-dim **largest**  $m$  for which  $2^m$  **patterns** achievable.
- So, solve for  $2^m \leq m n^8$ . Pseudo-dimension is  $O(\log n)$ .

**High level learning theory bit** Exploit properties of dual class

# Clustering: Linkage + Dynamic Programming

**Claim:** Pseudo-dimension of  $\alpha$ -weighted linkage + DP is  $O(\log n)$ , so small sample complexity.

For  $N = O(\log n / \epsilon^2)$ , w.h.p. expected performance cost of best  $\alpha$  over the sample is  $\epsilon$ -close to optimal over the distribution



**Claim:** Given sample  $S$ , can find best algo from this family in **poly time**.

## Algorithm

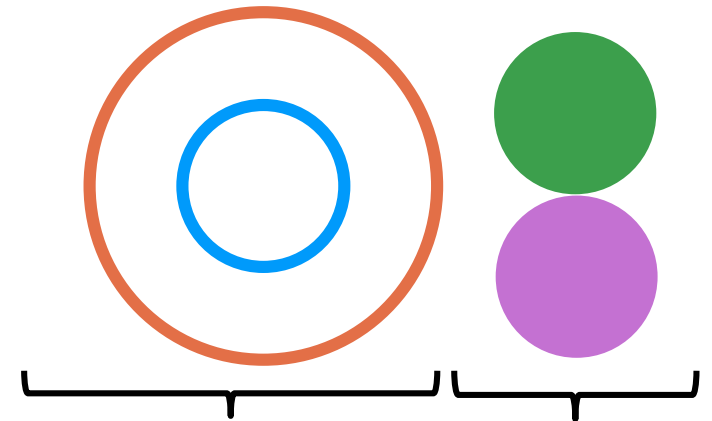
- Solve for all  $\alpha$  intervals over the sample



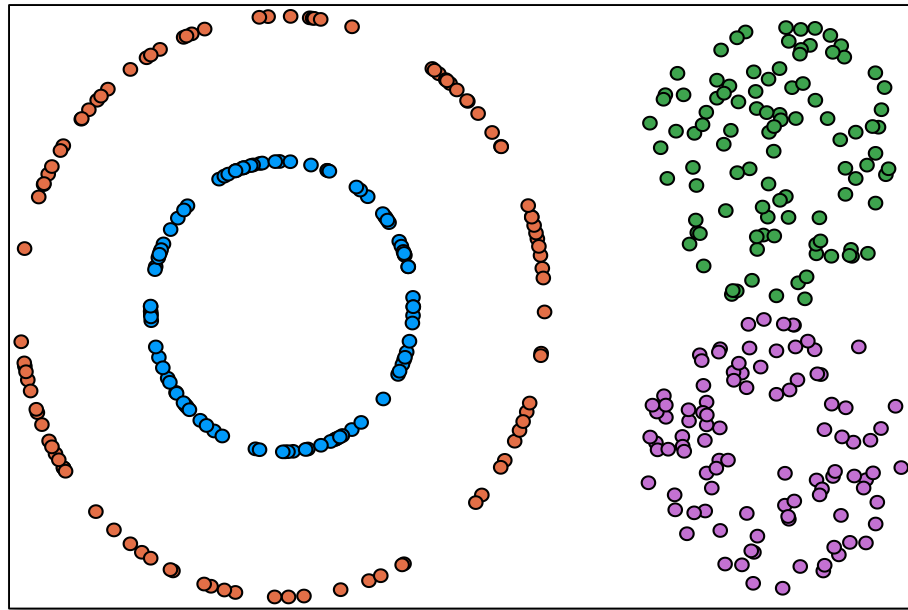
- Find the  $\alpha$  interval with the smallest empirical cost

# Synthetic Example

- 4 target clusters.
- 100 points sampled uniformly from each colored region (total 400 points)
- Measure loss by the Hamming distance of the best pruning to target clustering.



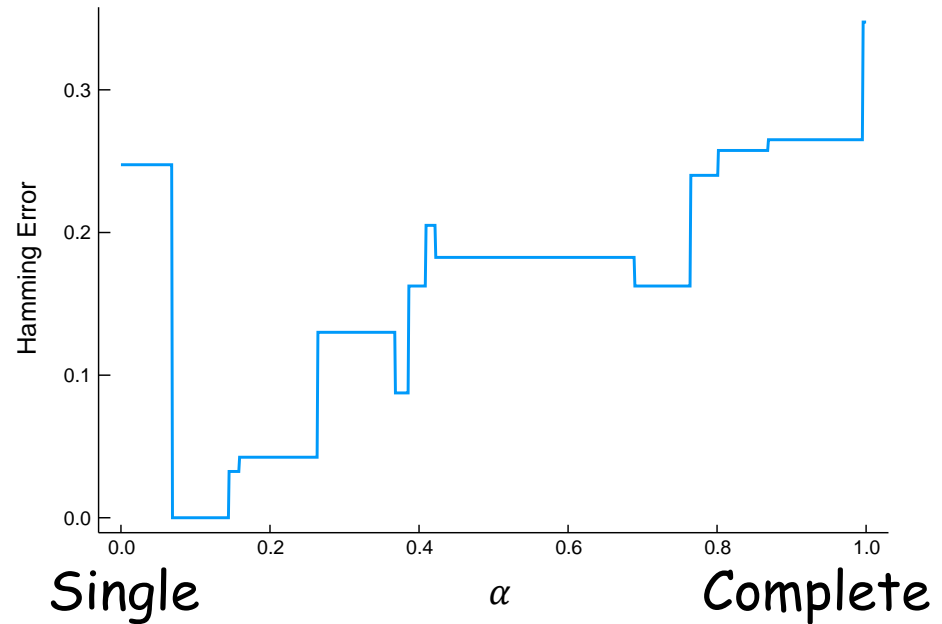
## Sample Instance:



Single Linkage Error: 0.25

Complete Linkage Error: 0.35

Best range:  $\alpha \in [0.07, 0.16]$  gives error 0



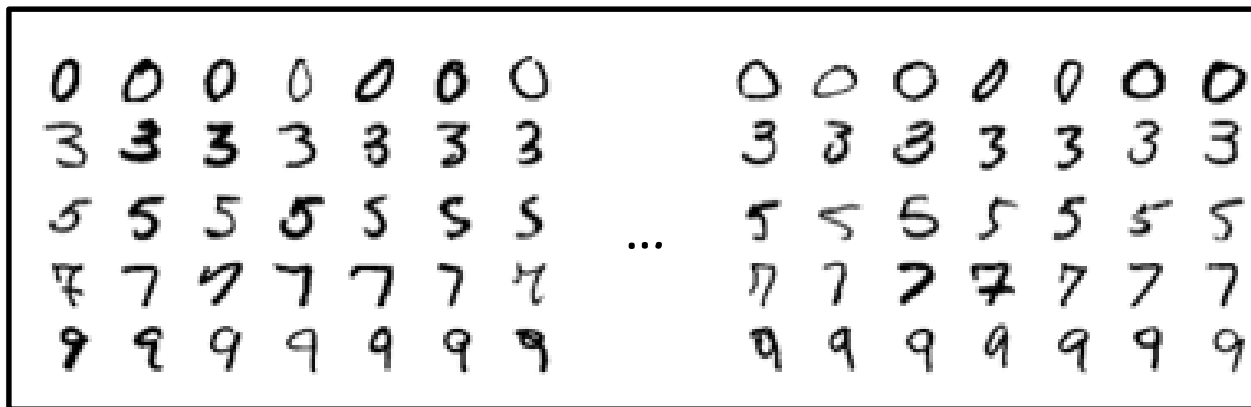
# Clustering Subsets of MNIST

Generate clustering instances from MNIST data.

## Instance Distribution

- Pick 5 random digits
- Pick 200 random images per digit (1000 in total)
- Target clustering is given by the digit classification
- Loss: Hamming distance of best pruning to target clustering.
- Similarity between images measured by  $\ell_2$  distance of pixel vectors.

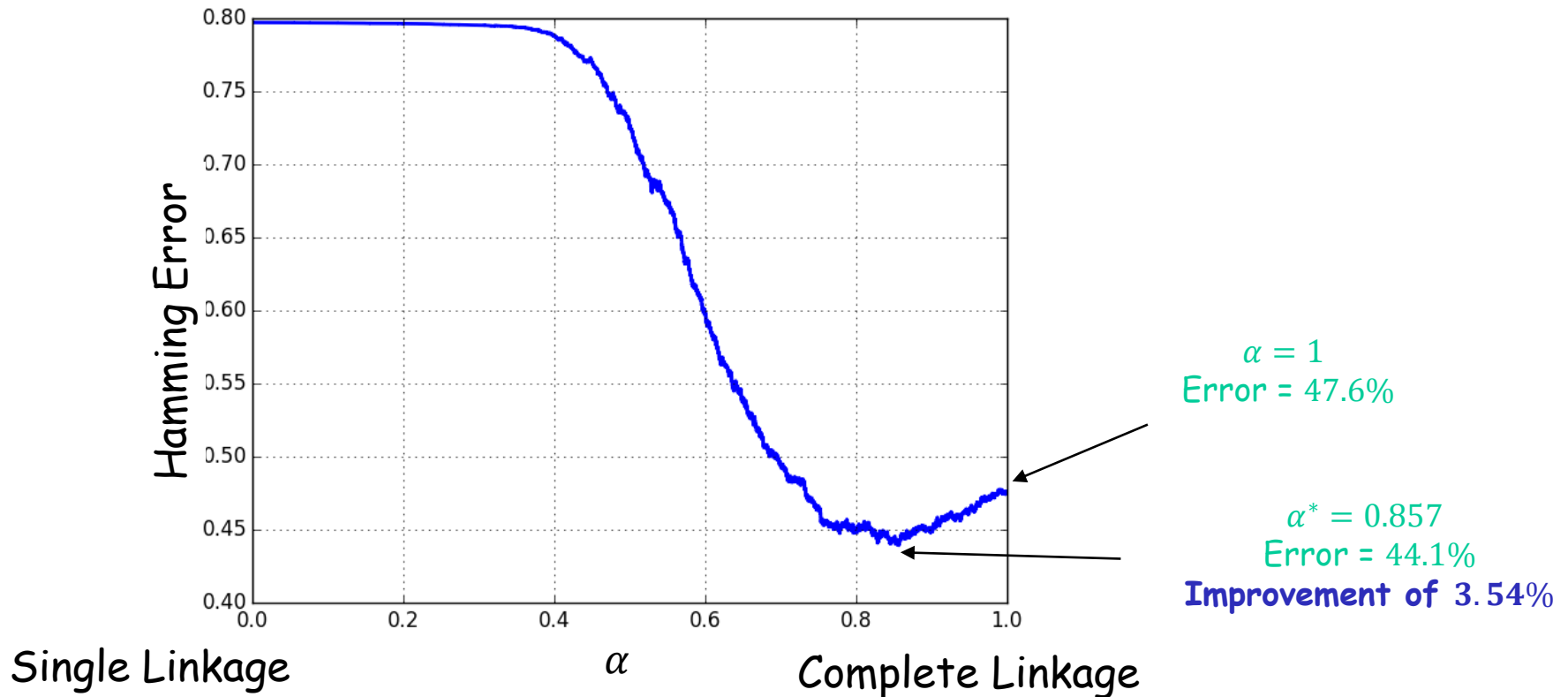
Example instance:





# Clustering Subsets of MNIST

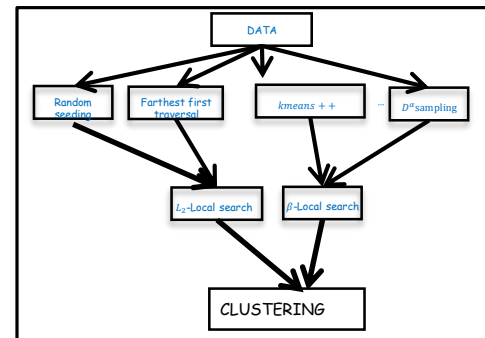
Error averaged over 500 instances



# Clustering: Parametrized Lloyd's

## Our Results: Parametrized Lloyd's

Let  $D(x)$  be the distance between a point  $x$  and its nearest center. Chose the next center proportional to  $D^\alpha(x)$ .



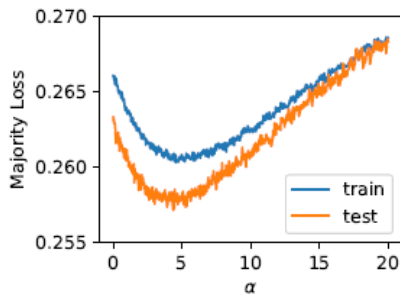
- Pseudo-dimension is  $O(k \log n)$ , so small sample complexity.
- Given sample  $S$ , find best algo from this family in poly time.



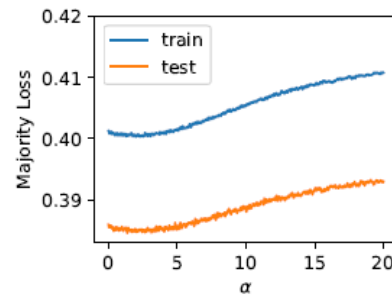
# Clustering: Parametrized Lloyd's

## Our Results: Parametrized Lloyd's

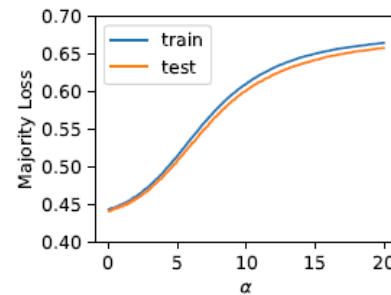
Let  $D(\mathbf{x})$  be the distance between a point  $x$  and its nearest center. Chose the next center proportional to  $D^\alpha(\mathbf{x})$ .



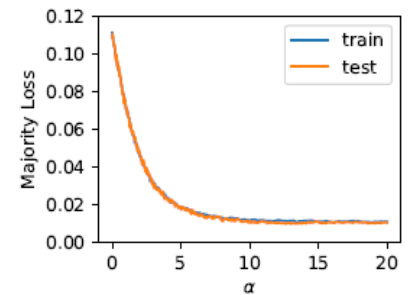
(a) MNIST



(b) CIFAR-10



(c) CNAE-9



(d) Gaussian Grid

# Learning Distance Metrics

**Goal:** Learn the best metric to use with complete linkage.

**Family of Metrics:** Given  $d_0$  and  $d_1$ , define

$$d_\beta(x, x') = (1 - \beta) \cdot d_0(x, x') + \beta \cdot d_1(x, x')$$

E.g.: Captioned images



"Black Cat"



"Bobcat"



"Big Cat"



"Excavator"

$d_0$  = distance measured in terms of the images.

$d_1$  = distance measured in terms of the image captions.

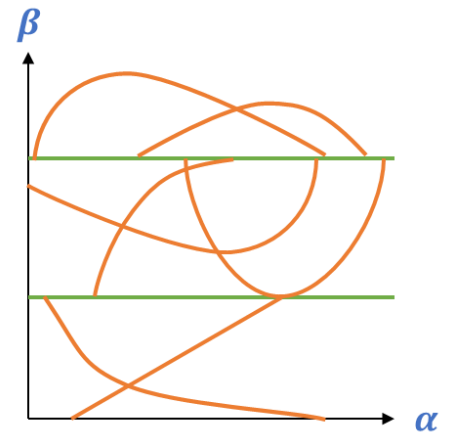
# Learning the Algorithm & the Distance

## Thm:

- For interpolation  $\alpha$  between single and complete linkage and interpolation  $\beta$  between two metrics.
- Distribution  $\mathcal{P}$  over clustering instances with  $n$  points.
- Given  $S_1, \dots, S_N$  drawn i.i.d. from  $\mathcal{P}$  of size  $N = O\left(\frac{1}{\epsilon^2} \log\left(\frac{n}{\delta}\right)\right)$ 
  - w.p.  $\geq 1 - \delta$ ,  $\sup_{(\alpha, \beta)} \left| \frac{1}{N} \sum_i \ell\left(\mathcal{A}_{\alpha, \beta}(S_i)\right) - \mathbb{E}_{S \sim \mathcal{P}} \left[ \ell\left(\mathcal{A}_{\alpha, \beta}(S)\right) \right] \right| \leq \epsilon$ .

## Key Claim:

Fix an instance; vary  $\alpha, \beta$ , partition space with  $O(n^8)$  linear or quadratic equations s.t. within each region, get the same clustering.



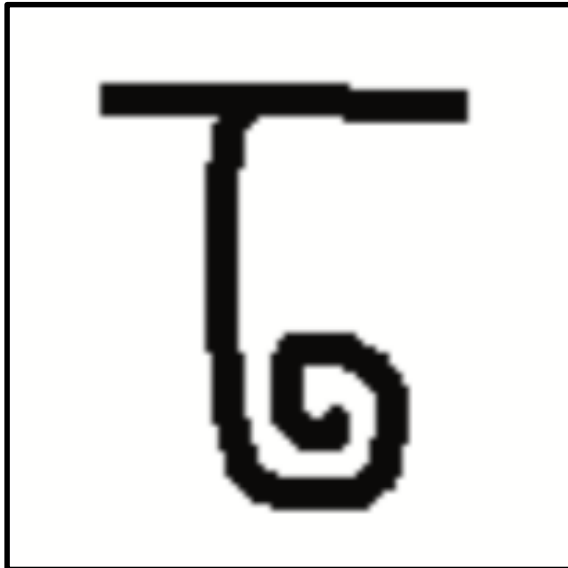
# Clustering Subsets of Omniglot

## Dataset [Lake, Salakhutdinov, Tenenbaum '15]

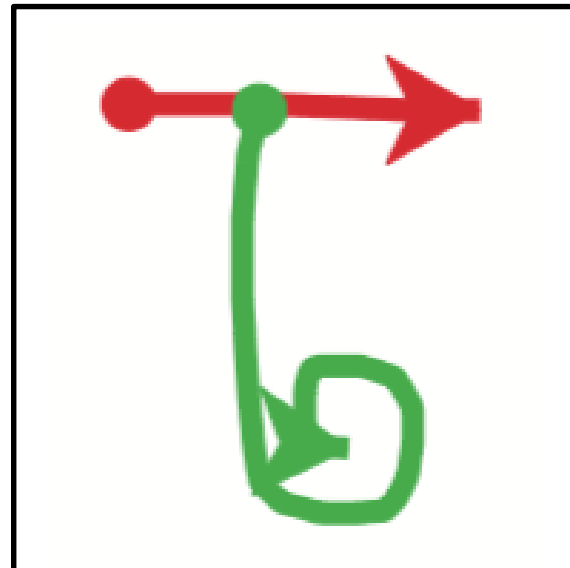
- Standard meta-learning dataset.
- Written characters from 50 alphabets.
- Each character has 20 examples.
  - Image of character
  - Stroke data (trajectory of pen)

ಗ	ಙ	ಕ	ಖ	ಗ
ಚ	ಛ	ಜ	ಝ	ಞ
ತ	ಠ	ಡ	ಢ	ನ
ಪ	ಫ	ಬ	ಭ	ಮ
ಯ	ರ	ಲ	ವ	ಶ
ಷ	ಸ	ಹ	ಳ	ಃ

## Character Image



# Stroke Data



# Clustering Subsets of Omniglot

## Instance Distribution

- Pick random alphabet.
- Pick between 5 and 10 characters.
- Use all 20 examples of chosen characters (100 - 200 points)
- Target clusters are characters.

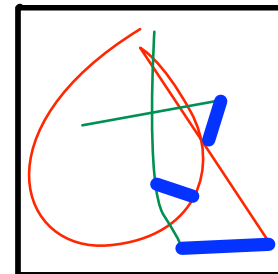
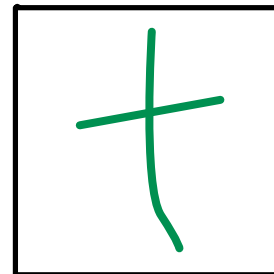
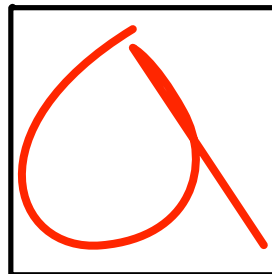
## Distance Metrics:

### MNIST Feature Embeddings:

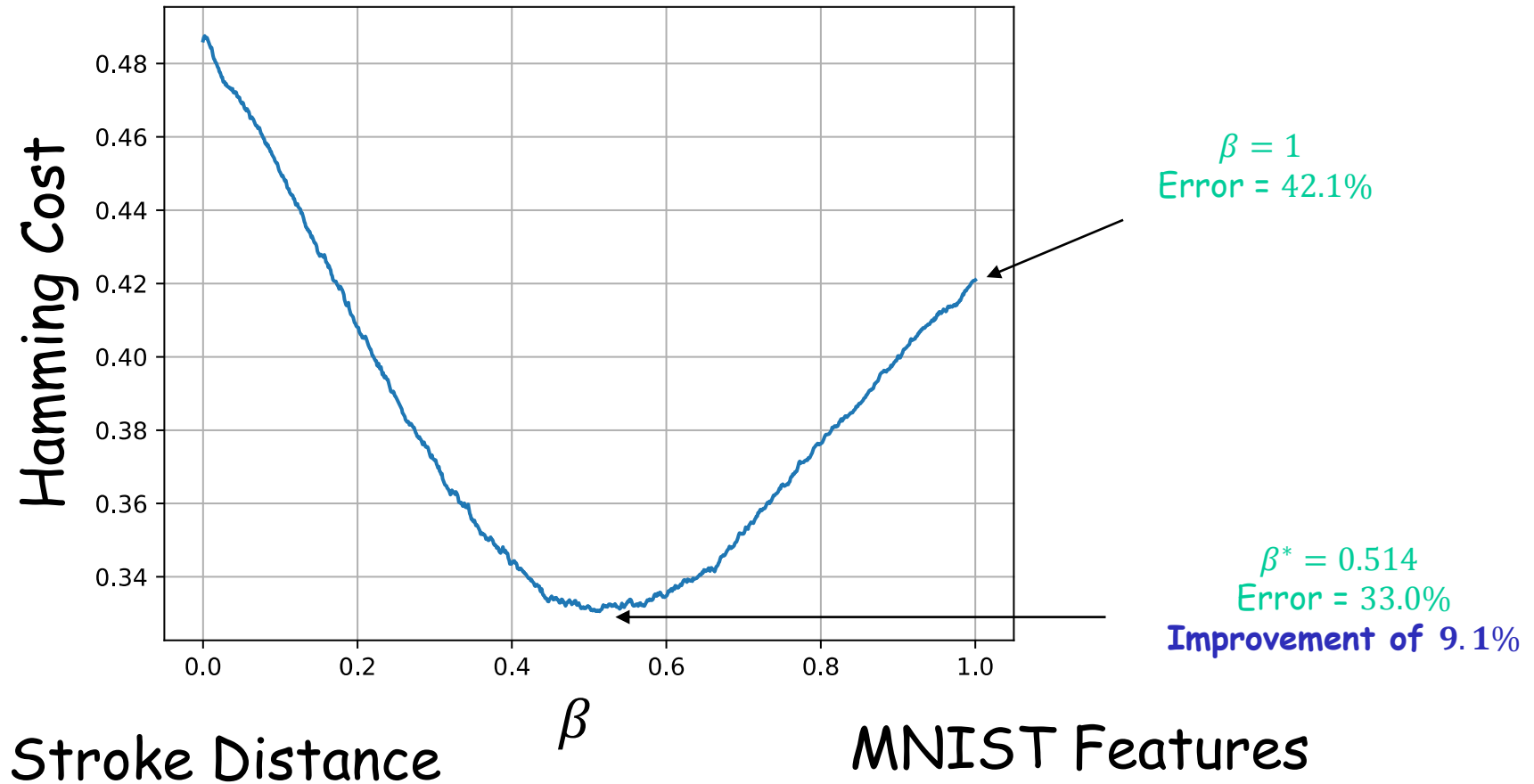
- Uses character images.
- Cosine distance between CNN feature embeddings.
- CNN trained on MNIST.

### Hand-designed Stroke Distance:

- Uses stroke data.
- Average distance from points on each stroke to the nearest point on the other stroke.



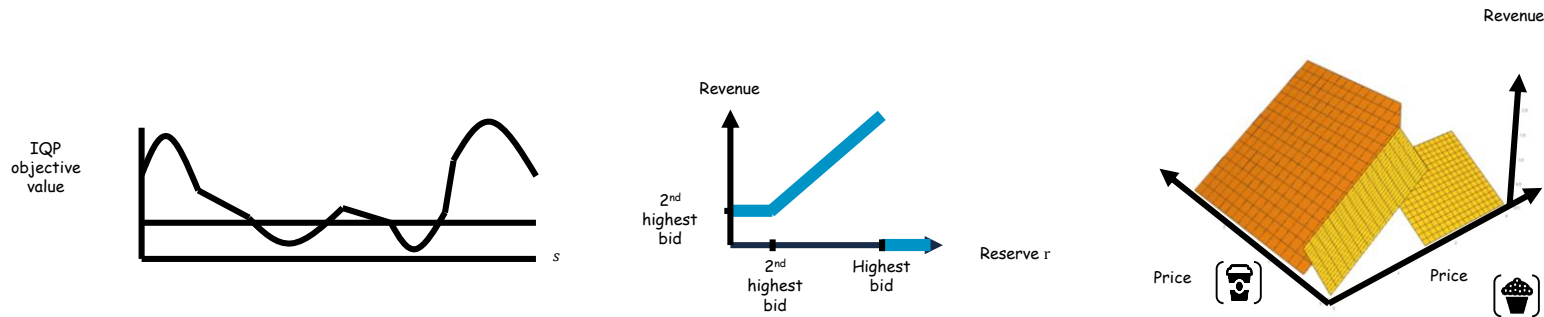
# Clustering Subsets of Omniglot





# Summary and Discussion

- Strong performance guarantees for data driven transfer clustering.
- Provide and exploit structural properties of dual class for good sample complexity.



- Also show empirical improvements on image data (MNIST, Omniglot).

