Advances in Hierarchical Clustering of Vector Data

Joint work with Moses Charikar, Vaggos Chatziafratis, Rad Niazadeh (Stanford), AISTATS'19

Grigory Yaroslavtsev

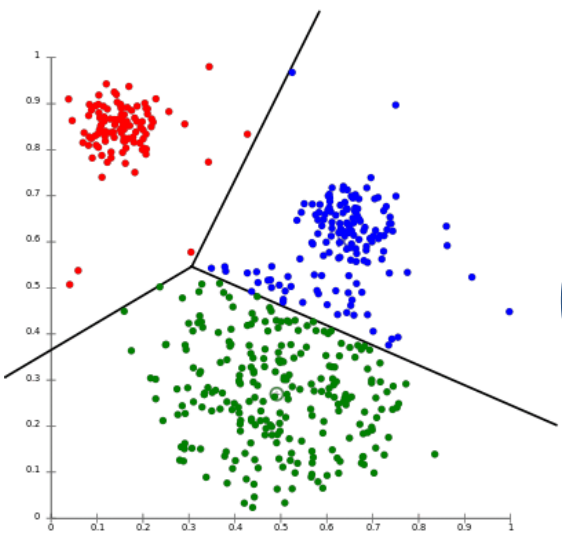
Indiana University (Bloomington)

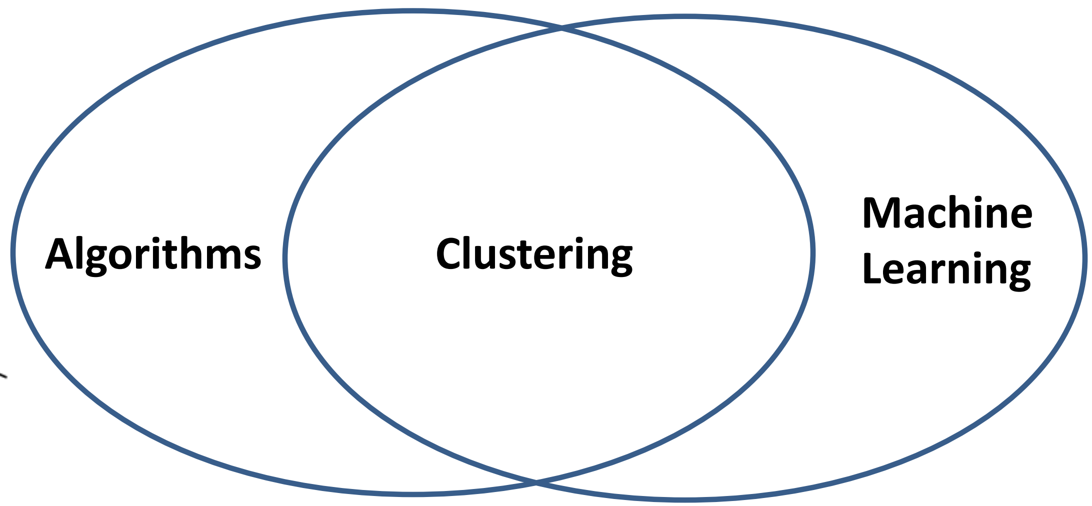
http://grigory.us/blog

Ψ CAML

Clustering

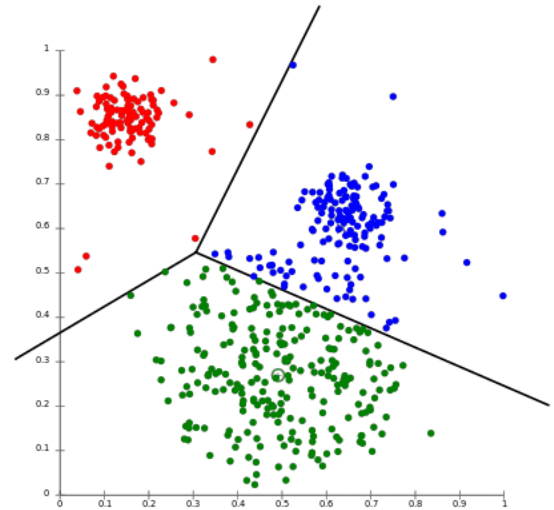
- Key problem in unsupervised learning



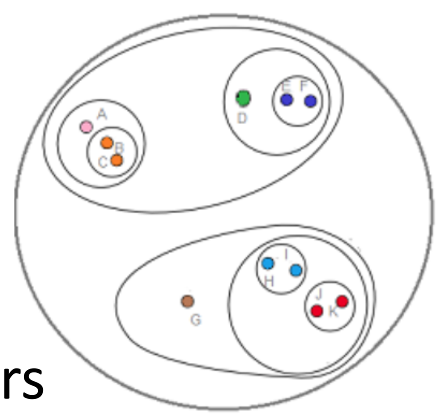


- Workshop at TTI-Chicago **"Recent Trends in Clustering"**
  - September 18-20, http://grigory.us/caml/rtcc.html!

Hierarchical Clustering of Clustering Methods

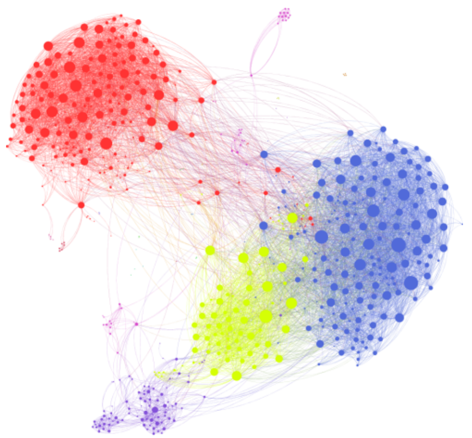


- Flat (single) clustering
  - – # of clusters K is fixed
    - K-means, K-median, K-center, etc
  - – # of clusters selected by algorithm
    - Correlation clustering
    - …
- **Hierarchical clustering**
  - – Tree over data points,
    - Can pick any # of clusters
    - Relationships between clusters

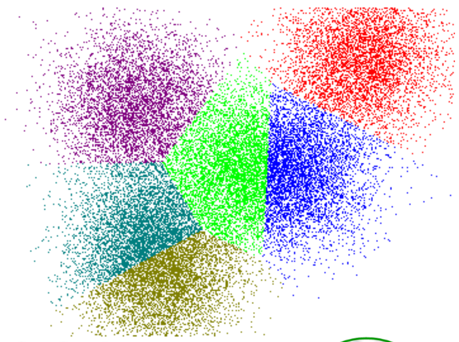


Data we can hierarchically cluster

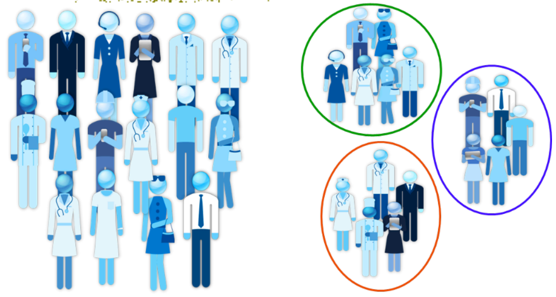
- Graphs [Avdiukhin, Pupyrev, Y. VLDB'19]
  - Recent work with Facebook
  - Max edge locality, clusters balanced on many params
  - Scales to the Facebook graph
  - Up to 10^{10} vertices, 10^{12} edges



- **Vectors** $(v_1, \ldots, v_n \in \mathbb{R}^d)$

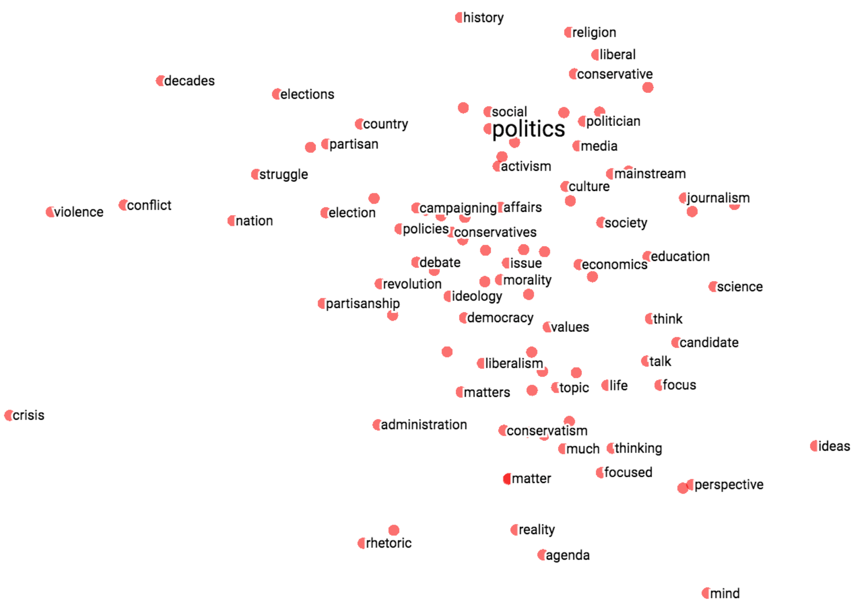


- **Arbitrary**

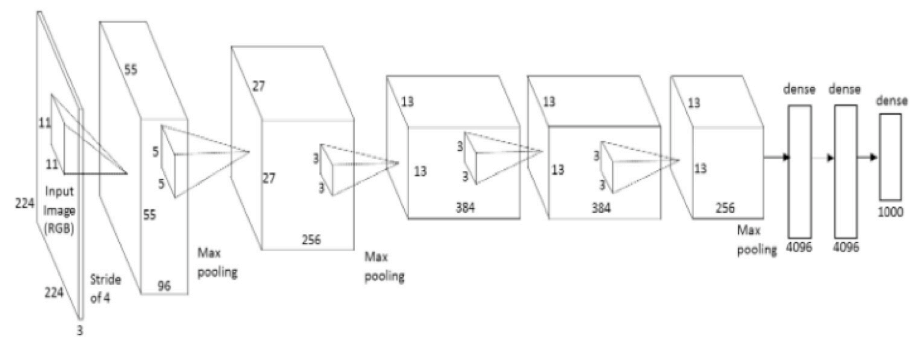


Embedding vectors from deep learning

- Word2Vec embeddings



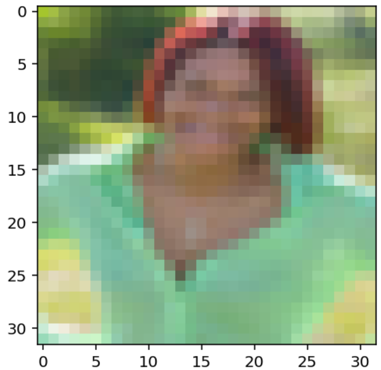
- Image embeddings



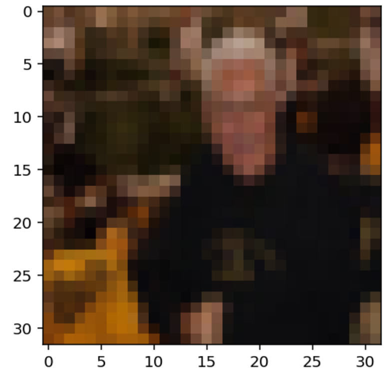
ImageNet embeddings (AlexNet)

Image: Tensorflow documentation
https://www.tensorflow.org/guide/embedding

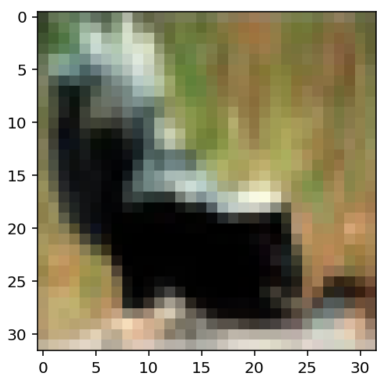
Toy example of HC on CIFAR-100



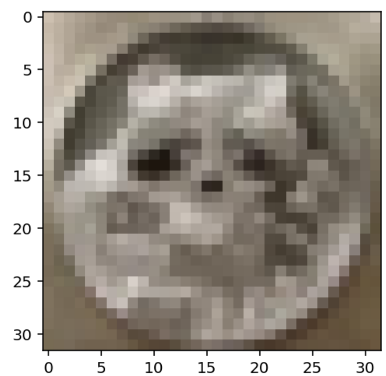
Woman



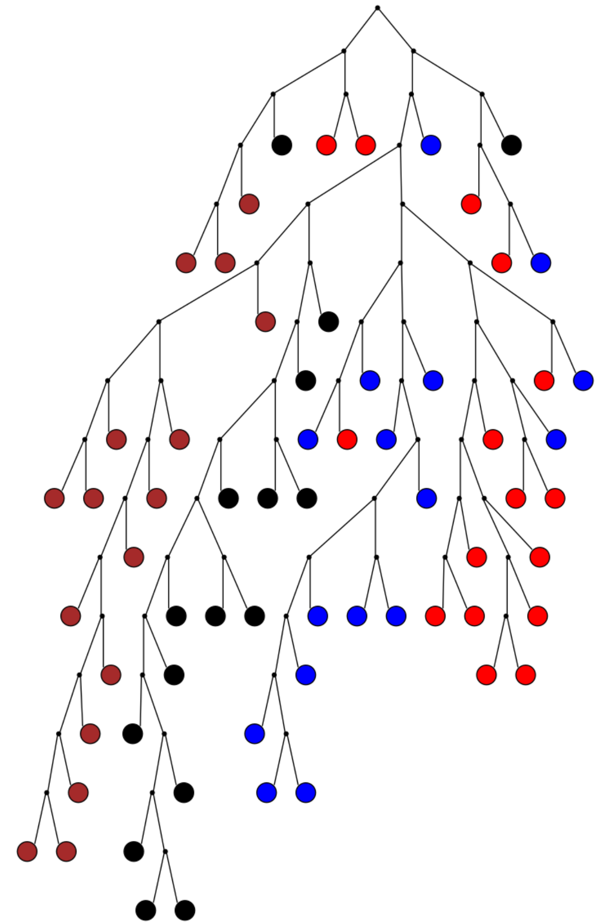
Man



Skunk



Raccoon

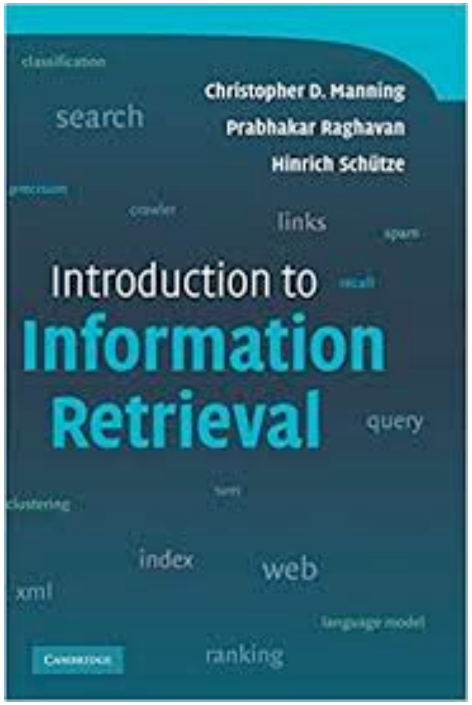


Embedding vectors via PyramidNet
https://arxiv.org/abs/1610.02915
Top-1 error on CIFAR-100: 16-20%

Why little theoretical progress on HC?

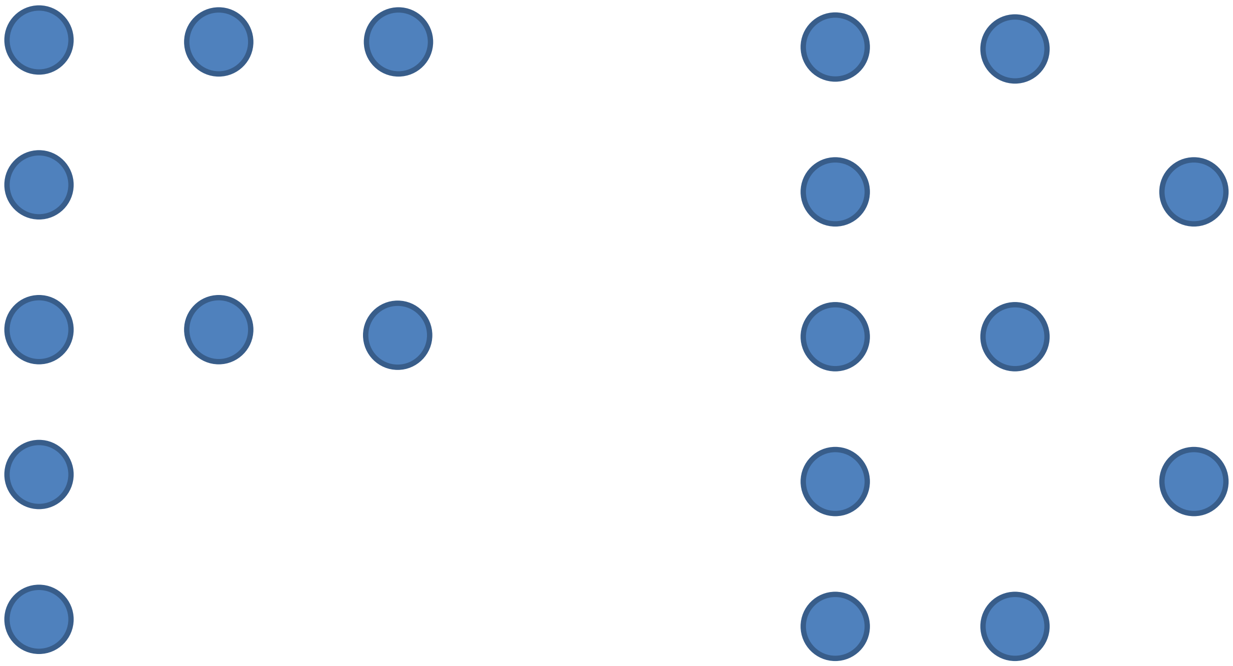
Lots of heuristics, (almost) no rigorous objectives

- **Bottom-up (linkage-based heuristics)**
  - **Single-linkage clustering**
  - **Average-linkage clustering**
  - Complete-linkage clustering
  - Centroid linkage
  - All sorts of other linkage methods
- Implementations of HC in:
  - Mathematica, R, Matlab
  - SciPy, Scikit-learn, …



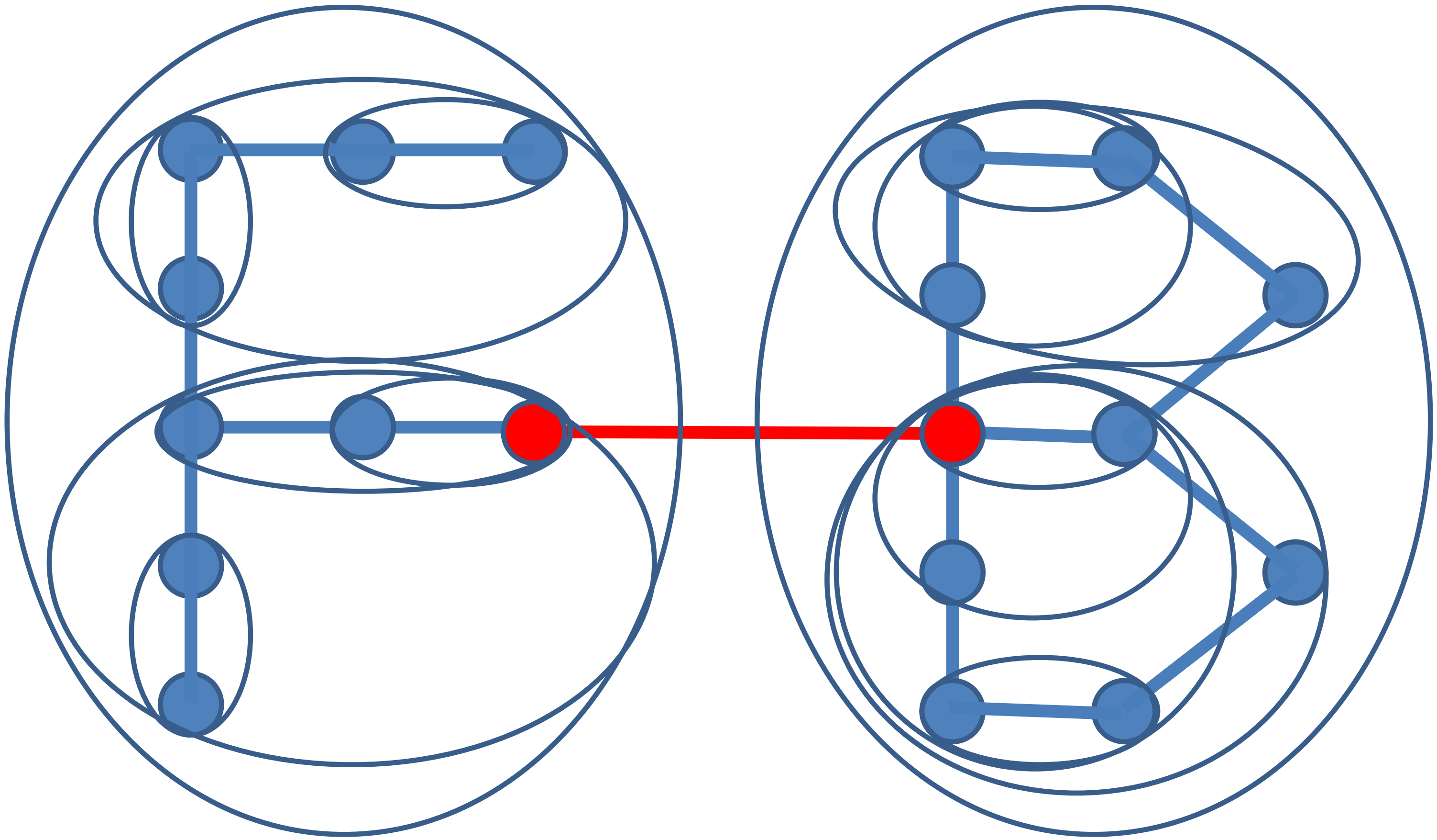
Bottom-up linkage-based heuristics

- Start with singletons
- Iteratively merge two **closest clusters**



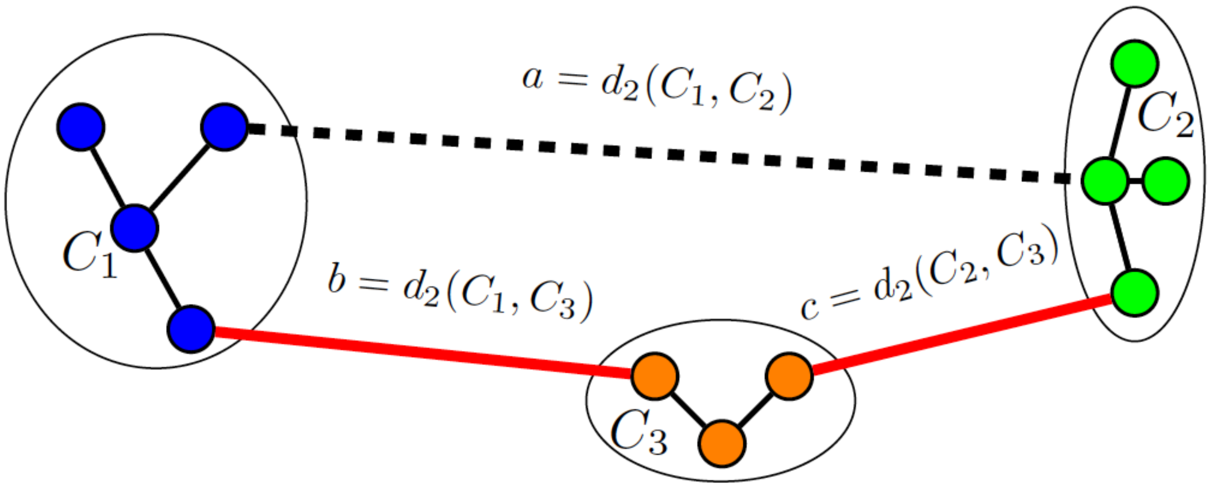
Single-Linkage Clustering

- Distance = distance between two closest points



MST: Single-Linkage Clustering

- [Zahn'71] $\boldsymbol{k}$ clusters: remove $\boldsymbol{k-1}$ longest MST edges
- **Objective**: maximizes **minimum** cluster distance

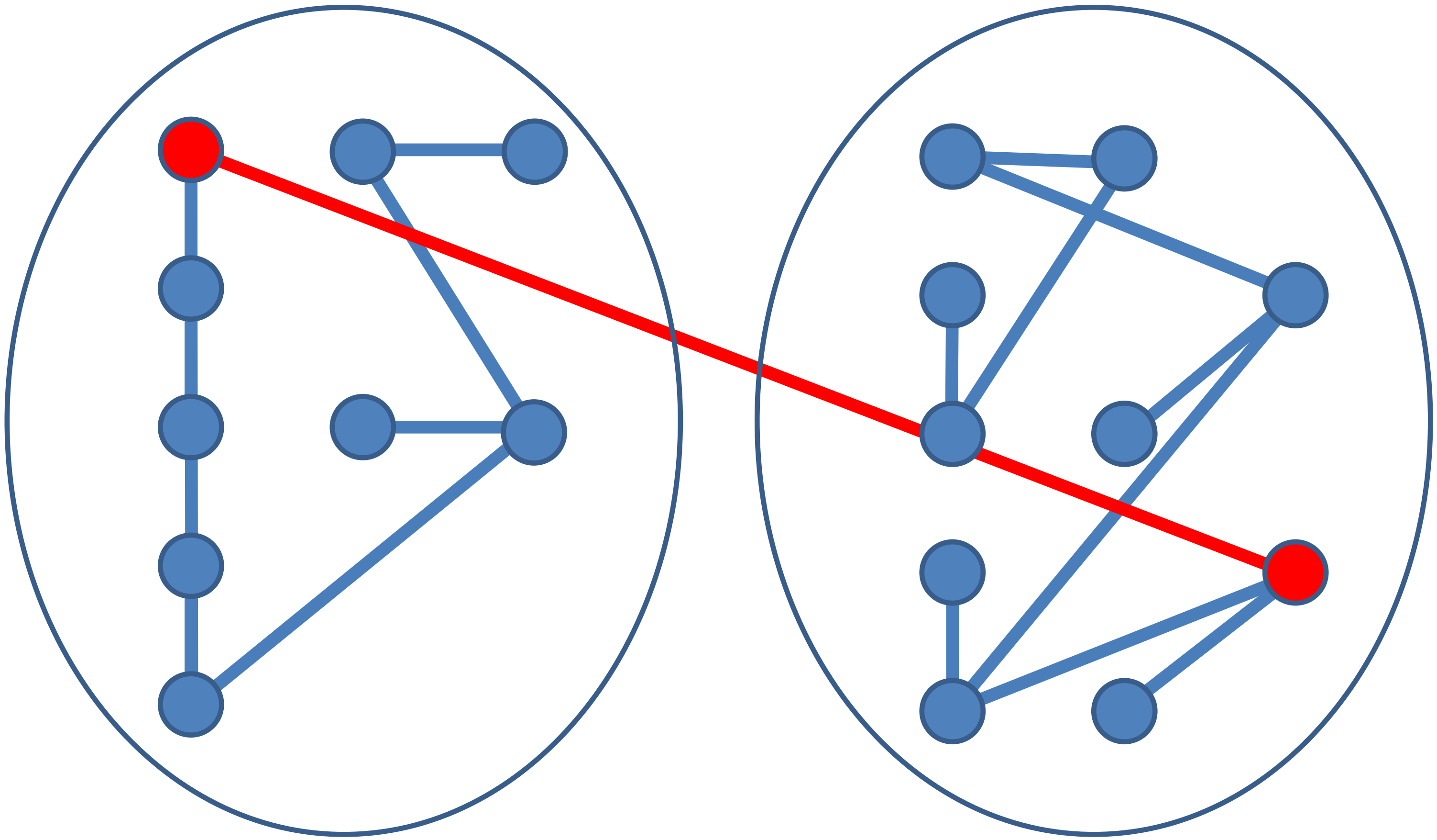


Objective: maximize min(a,b,c)

- Not true if MST is approximate (sum vs. $\boldsymbol{k}$-th edge)!
  – Scalable algorithms for Single-Linkage Clustering of vectors [**Y.**, Vadapalli ICML'18]

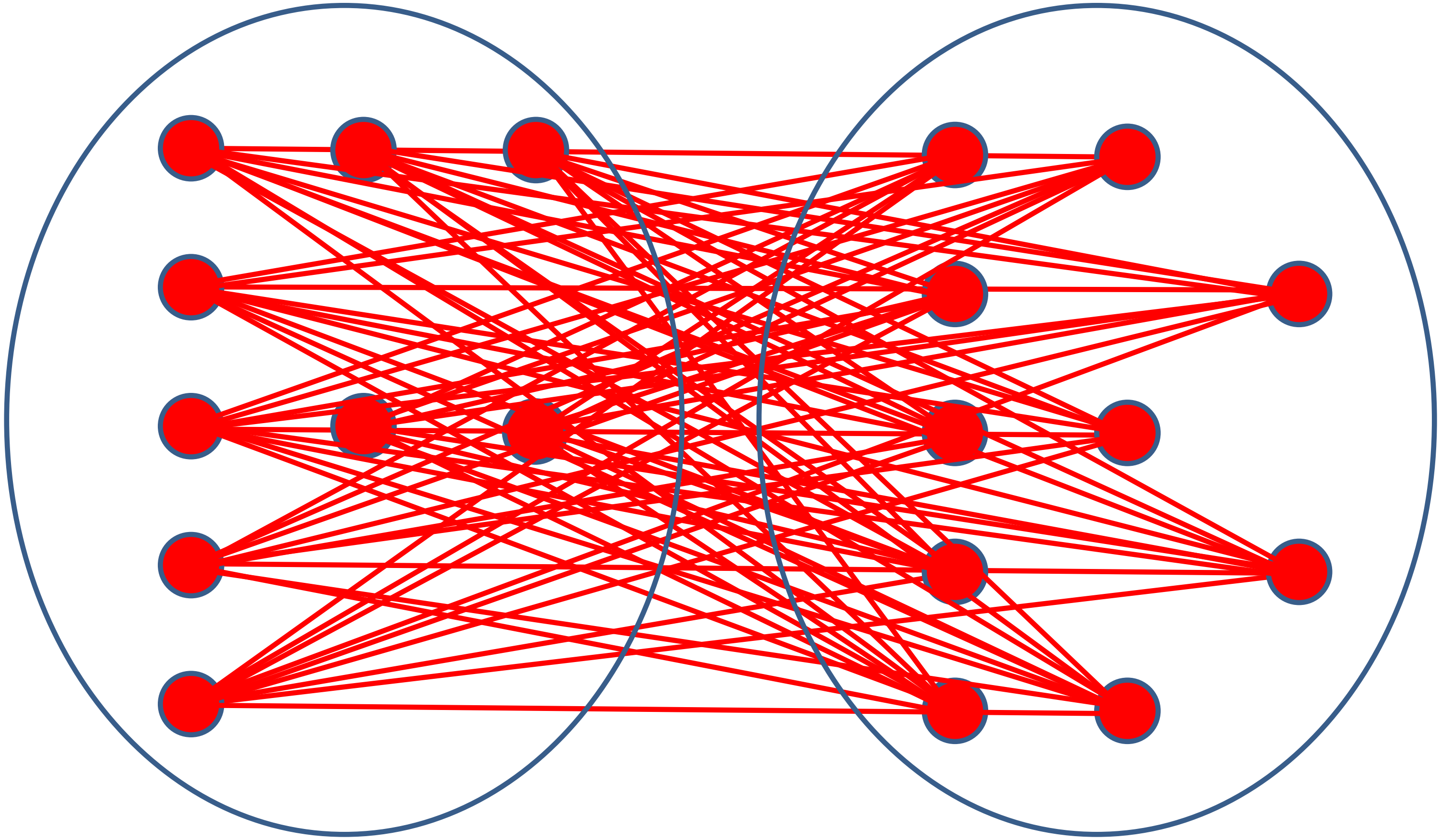
Complete Linkage

- Distance = distance between two furthest points



Average-Linkage Clustering

- Distance = average distance between points



Distance vs. **Similarity**

- Distance $d(u, v)$
  - Arbitrary
  - Metric: $d(u, v)$ satisfies triangle inequality
  - Vectors: $||v_i - v_j||_2$

$$d(v_i, v_j) = ||v_i - v_j||_2 = \sqrt{\sum_{k=1}^{\boldsymbol{d}} (v_{ik} - v_{jk})^2}$$

Distance vs. **Similarity**

Similarity $\rho(u, v) \in [\mathbf{0}, \mathbf{1}]$

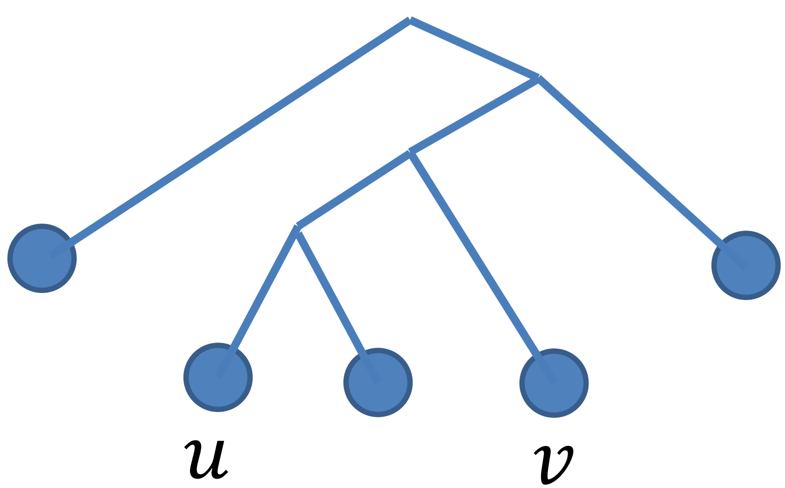
- Arbitrary: symmetric, $\rho(u, u) = 1$
- Metric case: monotone with distance
- Vector case:
  - Threshold: $\rho(v_i, v_j) = \begin{cases} 1, \text{ if } d(v_i, v_j) \le \boldsymbol{\theta} \\ 0, \text{ if } d(v_i, v_j) > \boldsymbol{\theta} \end{cases}$
  - Gaussian kernel: $\rho(v_i, v_j) = e^{-\frac{||v_i - v_j||_2^2}{2\sigma^2}}$

Dasgupta's objective for HC [STOC'16]

Given **n** data points and similarity measure ρ

- Build a tree T with data points as leaves
- For a pair of points (u, v):

penalty(u, v)
$= \rho(u, v) \times (\text{min \# leaves in a subtree containing } u, v)$



penalty$(u, v) = 3 \times \rho(u, v)$

Dasgupta's objective for HC [STOC'2016]

Given **n** data points and similarity measure ρ

- Build a tree T with data points as leaves
- For a pair of points (u, v):

$$\textbf{penalty}(u, v) = \rho(u, v)(\text{min \# leaves in a subtree containing } u, v)$$

Minimize:

$$\sum_{u \neq v} \textbf{penalty}(u, v) = \sum_{u \neq v} \rho(u, v) |LCA(u, v)|$$

- LCA(u,v) = Least Common Ancestor of (u, v) in T
- |LCA(u,v)| = # leaves under LCA(u,v)

Dasgupta's objective for HC [STOC'2016]

Given **n** data points and similarity measure ρ

- Build a tree T with data points as leaves:

$$\textbf{Minimize}: = \sum_{u \neq v} \rho(u,v)|LCA(u,v)|$$

  - |LCA(u,v)| = # leaves under LCA(u,v)

- Currently best known approximation $O(\sqrt{\log n})$
  [Charikar, Chatziafratis, SODA'17; Roy, Pokutta NIPS'16]
  - LP/SDP-based algorithms, don't scale to large datasets
  - As hard as Sparsest Cut (no constant-factor approx. under UGC)

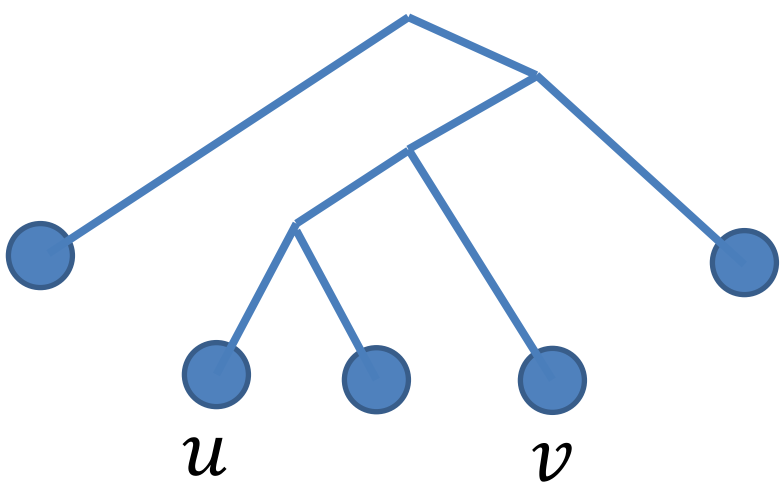
Moseley-Wang objective for HC [NIPS'17]

Given **n** data points and similarity measure ρ

- Build a tree T with data points as leaves:

Maximize: $\sum_{u \neq v} \mathbf{score}(u,v) = \sum_{u \neq v} \rho(u,v)(\boldsymbol{n} - |LCA(u,v)|)$

- $\mathbf{score}(u,v) = \boldsymbol{n} - |LCA(u,v)|$
- |LCA(u,v)| = # leaves under LCA(u,v)



$\mathbf{score}(u,v) = 2 \times \rho(u,v)$

Moseley-Wang objective for HC [NIPS'17]

Given **n** data points and similarity measure ρ

- Build a tree T with data points as leaves:

Maximize: $\sum_{u \neq v} \mathbf{score}(u, v) = \sum_{u \neq v} \rho(u, v)(\boldsymbol{n} - |LCA(u, v)|)$

  - $\mathbf{score}(u, v) = \boldsymbol{n} - |LCA(u, v)|$
  - |LCA(u,v)| = # leaves under LCA(u,v)

- Average-linkage gives 1/3-approximation [Moseley, Wang, NIPS'17]
  - Random recursive partitioning also gives 1/3 (in expectation)
- Best known approximation is 1/3 + $\boldsymbol{\delta}$ [Charikar, Chatziafratis, Niazadeh SODA'19]
  - Uses SDP, doesn't scale to large data
  - Average-linkage can't beat 1/3

Random recursive partitioning [MW'17]

Algorithm: Split points randomly and recurse

Analysis: Decompose the objective over triples

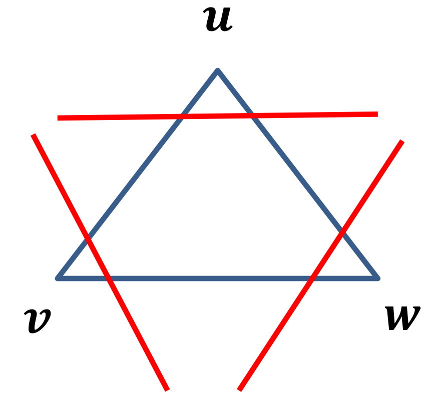
$$\sum_{\boldsymbol{u} \neq \boldsymbol{v}} \rho(\boldsymbol{u}, \boldsymbol{v})(\boldsymbol{n} - |LCA(\boldsymbol{u}, \boldsymbol{v})|) = \sum_{\boldsymbol{u} \neq \boldsymbol{v} \neq \boldsymbol{w}} \rho(\boldsymbol{u}, \boldsymbol{v}) I[\boldsymbol{w} \text{ is not under } LCA(\boldsymbol{u}, \boldsymbol{v})]$$

$$\sum_{\boldsymbol{u} \neq \boldsymbol{v} \neq \boldsymbol{w}} \rho(\boldsymbol{u}, \boldsymbol{v}) I[\boldsymbol{w} \text{ is not under } LCA(\boldsymbol{u}, \boldsymbol{v})] =$$

$$\sum_{\boldsymbol{u} < \boldsymbol{v} < \boldsymbol{w}} \rho(\boldsymbol{u}, \boldsymbol{v}) I[\boldsymbol{w} \text{ is not under } LCA(\boldsymbol{u}, \boldsymbol{v})] +$$

$$\rho(\boldsymbol{u}, \boldsymbol{w}) I[\boldsymbol{v} \text{ is not under } LCA(\boldsymbol{u}, \boldsymbol{w})]$$

$$\rho(\boldsymbol{w}, \boldsymbol{v}) I[\boldsymbol{u} \text{ is not under } LCA(\boldsymbol{w}, \boldsymbol{v})]$$



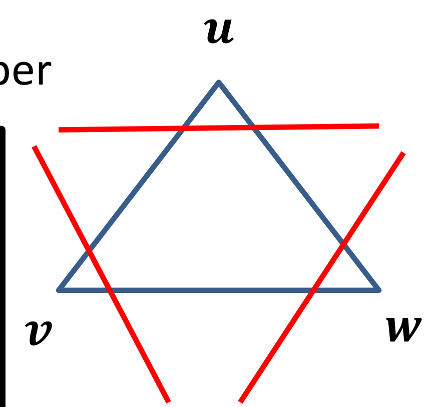
$$OPT \leq \sum_{\boldsymbol{u} < \boldsymbol{v} < \boldsymbol{w}} \max\big(\rho(\boldsymbol{u}, \boldsymbol{v}), \rho(\boldsymbol{u}, \boldsymbol{w}), \rho(\boldsymbol{v}, \boldsymbol{w})\big) = \text{Max-Upper}$$

Random recursive partitioning [MW'17]

Algorithm: Split points randomly and recurse
Analysis: Decompose the objective over triples

$$\sum_{\boldsymbol{u}\neq\boldsymbol{v}} \rho(\boldsymbol{u},\boldsymbol{v})(\boldsymbol{n} - |LCA(\boldsymbol{u},\boldsymbol{v})|) = \sum_{\boldsymbol{u}\neq\boldsymbol{v}\neq\boldsymbol{w}} \rho(\boldsymbol{u},\boldsymbol{v}) I[\boldsymbol{w} \text{ is not under } LCA(\boldsymbol{u},\boldsymbol{v})]$$

$OPT \leq \sum_{\boldsymbol{u}<\boldsymbol{v}<\boldsymbol{w}} \max\big(\rho(\boldsymbol{u},\boldsymbol{v}), \rho(\boldsymbol{u},\boldsymbol{w}), \rho(\boldsymbol{v},\boldsymbol{w})\big)$ = Max-Upper



$$\mathbb{E}[\text{ALG}] = \mathbb{E}\left[\sum_{\boldsymbol{u}\neq\boldsymbol{v}\neq\boldsymbol{w}} \rho(\boldsymbol{u},\boldsymbol{v}) I[\boldsymbol{w} \text{ is not under } LCA(\boldsymbol{u},\boldsymbol{v})]\right]$$

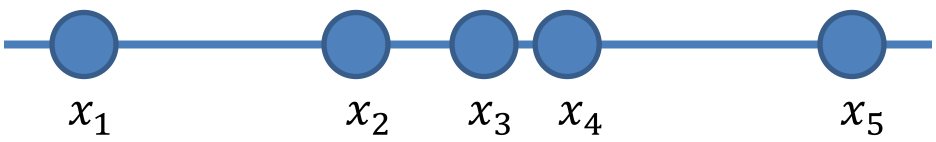
$$= \sum_{\boldsymbol{u}\neq\boldsymbol{v}\neq\boldsymbol{w}} \rho(\boldsymbol{u},\boldsymbol{v})\, Pr[\boldsymbol{w} \text{ is not under } LCA(\boldsymbol{u},\boldsymbol{v})]]$$

$$= \frac{1}{3} \sum_{\boldsymbol{u}\neq\boldsymbol{v}\neq\boldsymbol{w}} \rho(\boldsymbol{u},\boldsymbol{v}) = \frac{1}{3} \sum_{\boldsymbol{u}<\boldsymbol{v}<\boldsymbol{w}} \rho(\boldsymbol{u},\boldsymbol{v}) + \rho(\boldsymbol{u},\boldsymbol{w}) + \rho(\boldsymbol{v},\boldsymbol{w})$$

$$\geq \frac{1}{3} \sum_{\boldsymbol{u}<\boldsymbol{v}<\boldsymbol{w}} \max\big(\rho(\boldsymbol{u},\boldsymbol{v}), \rho(\boldsymbol{u},\boldsymbol{w}), \rho(\boldsymbol{v},\boldsymbol{w})\big) \geq \frac{1}{3} OPT$$

Can we do better for vector data?
[Charikar,Chatziafratis,Niazaheh,**Y.** AISTATS'19]

$\boldsymbol{d} = 1 \; (\rho(\boldsymbol{x_i}, \boldsymbol{x_j}) = f(|\boldsymbol{x_i} - \boldsymbol{x_j}|))$



Algorithm: Random-Cut $(\boldsymbol{x_1} \leq \boldsymbol{x_2} \ldots \leq \boldsymbol{x_n})$

- Pick $\boldsymbol{r}$ uniformly at random in $[\boldsymbol{x_1}, \boldsymbol{x_n}]$
- Recursively cluster points in $[\boldsymbol{x_1}, \boldsymbol{r}]$ and $[\boldsymbol{r}, \boldsymbol{x_n}]$

Analysis: Gives ½-approximation

$$OPT \leq \sum_{\boldsymbol{x_i} < \boldsymbol{x_j} < \boldsymbol{x_k}} \max\left(\rho(\boldsymbol{x_i}, \boldsymbol{x_j}), \rho(\boldsymbol{x_i}, \boldsymbol{x_k}), \rho(\boldsymbol{x_j}, \boldsymbol{x_k})\right)$$

$$\leq \sum_{\boldsymbol{x_i} < \boldsymbol{x_j} < \boldsymbol{x_k}} \max\left(\rho(\boldsymbol{x_i}, \boldsymbol{x_j}), \rho(\boldsymbol{x_j}, \boldsymbol{x_k})\right)$$

$$\mathbb{E}[ALG] \geq \frac{1}{2} \sum_{\boldsymbol{x_i} < \boldsymbol{x_j} < \boldsymbol{x_k}} \max\left(\rho(\boldsymbol{x_i}, \boldsymbol{x_j}), \rho(\boldsymbol{x_j}, \boldsymbol{x_k})\right)$$

Can we do better for vector data? [Charikar,Chatziafratis,Niazaheh,**Y.** AISTATS'19]

$\boldsymbol{d} = 1\ (\rho(\boldsymbol{x_i}, \boldsymbol{x_j}) = f(|\boldsymbol{x_i} - \boldsymbol{x_j}|))$

Average-linkage also gives ½

Conjectures:

- Average-linkage gives ¾?
- Dynamic programming gives the optimum solution?

Can we do better for vector data?

$v_1, \ldots, v_{\boldsymbol{n}} \in \mathbb{R}^{\boldsymbol{d}}$

- In general, **not easier than the general case**
- General hard instances are embeddable into vectors
  - Requires really high dimension $\boldsymbol{d} = \Omega(\boldsymbol{n})$
  - Relies on non-smoothness of the similarity measure
- Average-linkage can't beat 1/3 even for vectors

Projected Random Cut Algorithm

$v_1, \ldots, v_{\boldsymbol{n}} \in \mathbb{R}^{\boldsymbol{d}}$

Algorithm:

- Pick random Gaussian $\boldsymbol{g} \sim \boldsymbol{N_d}(0,1)$
- Compute projections $x_i = \langle v_i, \boldsymbol{g} \rangle$
- Run Random Cut on $(x_1, \ldots, x_n)$

Projected Random Cut Algorithm

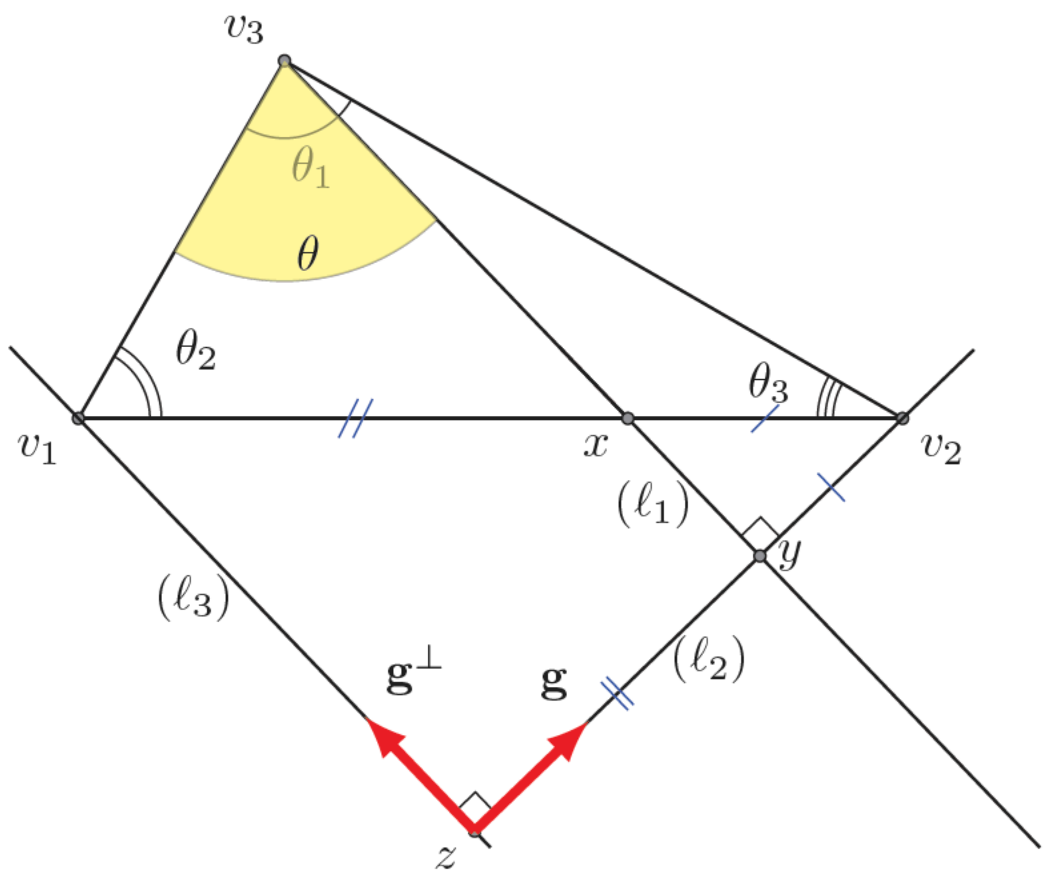
- Gaussian kernel: $\rho(v_i, v_j) = \exp\left(-\frac{||v_i - v_j||_2^2}{2\,\sigma^2}\right)$

- **Theorem:** Projected Random Cut gives $\frac{1+\delta}{3}$ approximation under the **Gaussian kernel similarity**, where $\delta = \min_{i,j} \rho(v_i, v_j)$
- **+ flavors:** similar statements for any smooth (i.e. multiplicatively Lipschitz) similarity measure

- **Key lemma:** probability of not scoring an edge of a triangle is proportional to the opposite angle

Key lemma

- $\Pr[v_3 \; is \; under \; LCA(v_1, v_2)] = \dfrac{\theta_1}{\pi}$



Projected Random Cut gives $\frac{1+\delta}{3}$

$$OPT \leq \sum_{v_i<v_j<v_k} \max\left(\rho(\boldsymbol{v_i},\boldsymbol{v_j}),\rho(\boldsymbol{v_j},\boldsymbol{v_k}),\rho(\boldsymbol{v_i},\boldsymbol{v_k})\right)$$

$\mathbb{E}[ALG] =$

$$\sum_{v_i<v_j<v_k}\left(1-\frac{\theta_{v_iv_j}}{\pi}\right)\rho(\boldsymbol{v_i},\boldsymbol{v_j})+\left(1-\frac{\theta_{v_jv_k}}{\pi}\right)\rho(\boldsymbol{v_j},\boldsymbol{v_k})+\left(1-\frac{\theta_{v_iv_k}}{\pi}\right)\rho(\boldsymbol{v_i},\boldsymbol{v_k})$$

If $\rho(\boldsymbol{v_{i,}}\boldsymbol{v_j})$ is largest then we score it with prob. $\geq \frac{1}{3}$

Experimental results for PRC

σ	PRC	SPECTRAL	AL	MAX-upper	$\frac{\text{PRC}}{\text{MAX-upper}}$
1.5	48	61	28	64	0.75
2	64	83	47	87	0.74
2.5	83	100	66	105	0.79
3	100	112	82	117	0.85
3.5	111	121	95	126	0.87
4	117	128	105	132	0.88
4.5	123	133	114	137	0.91
5	129	137	120	140	0.92

Table 1: Values of the objective (times 10^{-3}) on the Zoo dataset (averaged over 10 runs).

Scaling it up

- Ran PRC on largest vector datasets from UCI ML repository (SIFT 10M, HIGGS)
  - Approx. 10^7 points in up to 128 dimensions
  - $O(\boldsymbol{nd} + \boldsymbol{n}\log\boldsymbol{n})$ running time
  - Can run on much larger data too

- Can scale Single-Linkage clustering too, but
  - Uses PCA to reduce dimension
  - Requires a large Apache Spark cluster

Thank you!

- Questions?

- Some other topics I work(ed) on
  - Other clustering methods (e.g. **correlation clustering**)
  - **Massively parallel**, **streaming**, **sublinear** algorithms
  - **Data compression** methods for data analysis
  - **Submodular optimization**