Towards a Scalable Online Hierarchical Clustering Algorithm

Ishita Doshi (17210038)

Under the supervision of Prof. Anirban Dasgupta

Advisory Committee:

Prof. Manoj Gupta

Prof. Shanmuganathan Raman

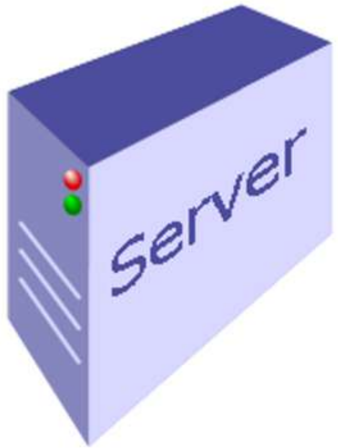
Contents

- Motivation
- Our Contributions
- Hierarchical Clustering
- Online Hierarchical Clustering
- Online Eigenvector Updates
- Conclusion

Motivation

User

Motivation

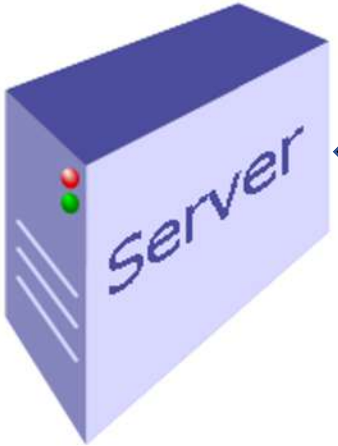


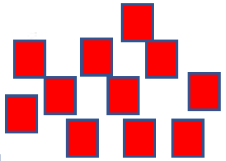
User

Motivation

User

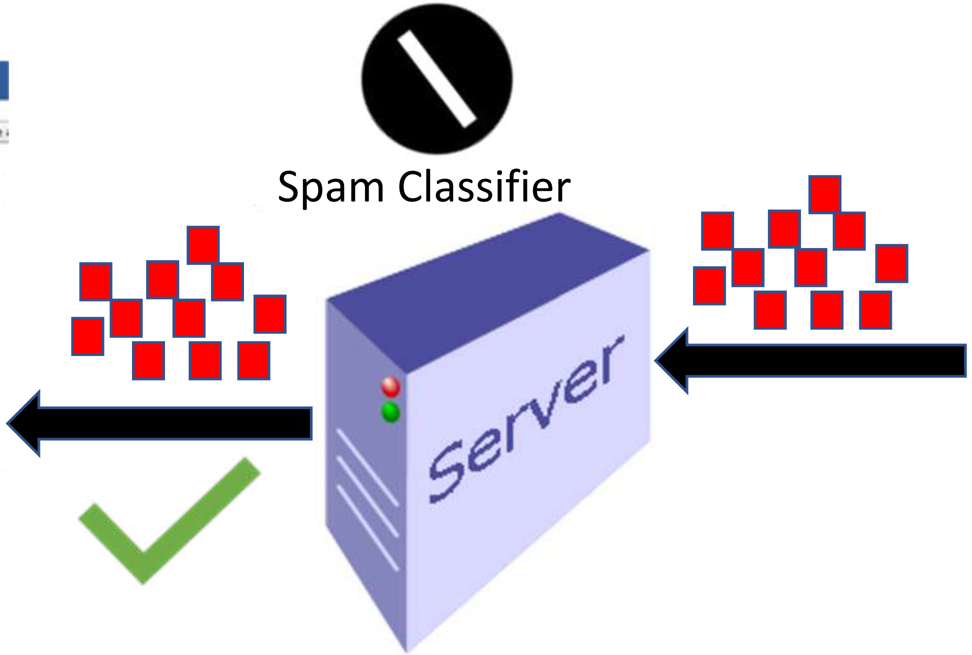
Spam Classifier



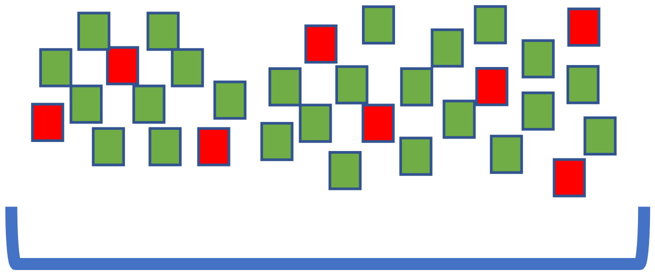


Motivation

User



Motivation

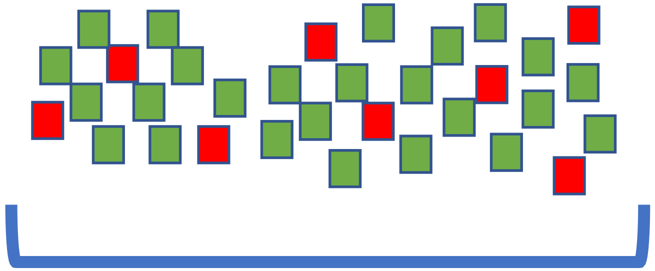


1 microsecond

= 100 comments

= 100 comments

Motivation



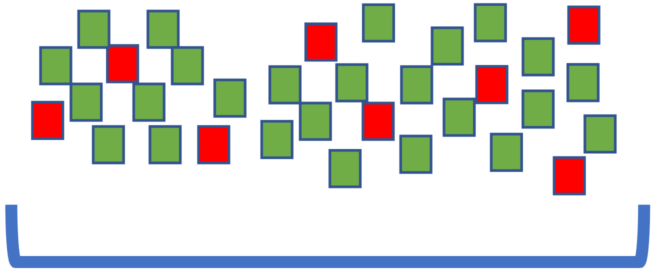
1 microsecond

= 100 comments

= 100 comments

- Feasible to get labels for all incoming data?

Motivation



1 microsecond

= 100 comments

= 100 comments

- Feasible to get labels for all incoming data?
- Can we afford delay in identification of harmful data?

Requirements

- Unsupervised
- Real-time identification
- Incremental updates to the model

Motivation

- Suspicious and malicious content may have a subtype-supertype relationship
- For example:
  - Harassment Spam
    - Bullying – abusive, rumors
    - Profanity – expletives, crude language
    - Trolling
    - Threats – blackmail, threats to life
  - Scam
    - Money scam
    - Malware downloads
    - Phishing

Requirements

- Unsupervised
- Real-time classification
- Incremental updates to the model
- Potentially use the subtype-supertype relationship

Possible Solutions

- Support Vector Machines

✓ Fast querying

✗ No subtype-supertype relation usage, no incremental updates, supervised

Possible Solutions

- Support Vector Machines
  - ✓ Fast querying
  - ✗ No subtype-supertype relation usage, no incremental updates, supervised
- Flat clustering
  - ✓ Unsupervised, fast querying
  - ✗ No subtype-supertype relation usage, no incremental updates

Possible Solutions

- Support Vector Machines
  - ✓ Fast querying
  - ✗ No subtype-supertype relation usage, no incremental updates, supervised
- Flat clustering
  - ✓ Unsupervised, fast querying
  - ✗ No subtype-supertype relation usage, no incremental updates
- Online flat clustering
  - ✓ Unsupervised, fast querying, incremental updates
  - ✗ No subtype-supertype relation usage

Proposed Solutions

- Hierarchical Clustering

✓ Unsupervised, fast querying, subtype-supertype relation usage

✗ No incremental updates

Proposed Solutions

- Hierarchical Clustering

✓ Unsupervised, fast querying, subtype-supertype relation usage

✗ No incremental updates

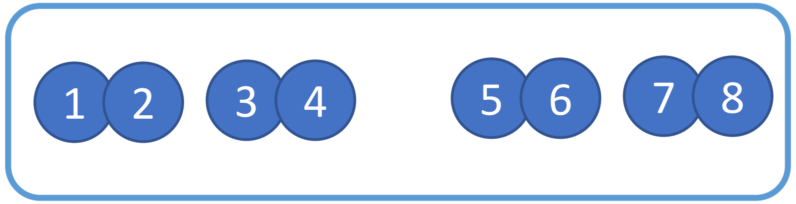
- Online Hierarchical Clustering

✓ Unsupervised, fast querying, subtype-supertype relation usage, incremental updates

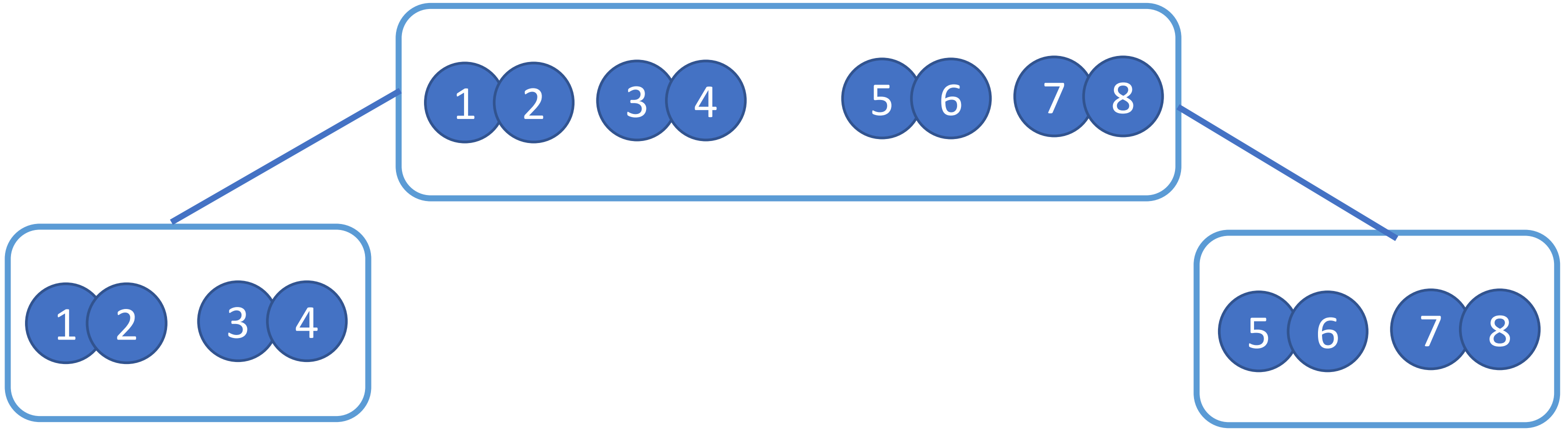
Our Contributions

- Practically efficient Hierarchical Clustering with guarantees on the quality of solution produced
- Demonstration of use of Hierarchical Clustering for
    - Real Time Classification
    - Anomaly detection
- Heuristics for Online Hierarchical Clustering
- Algorithm for Online Eigenvector Updates in a Dynamic Stream

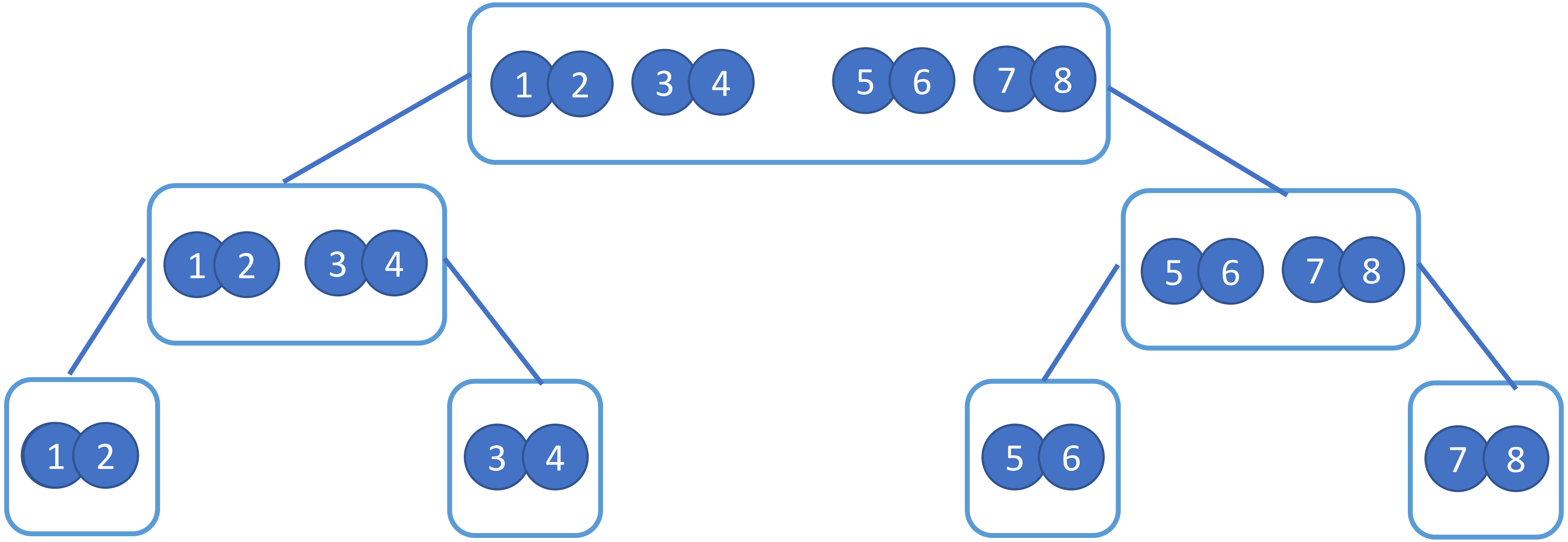
Hierarchical Clustering



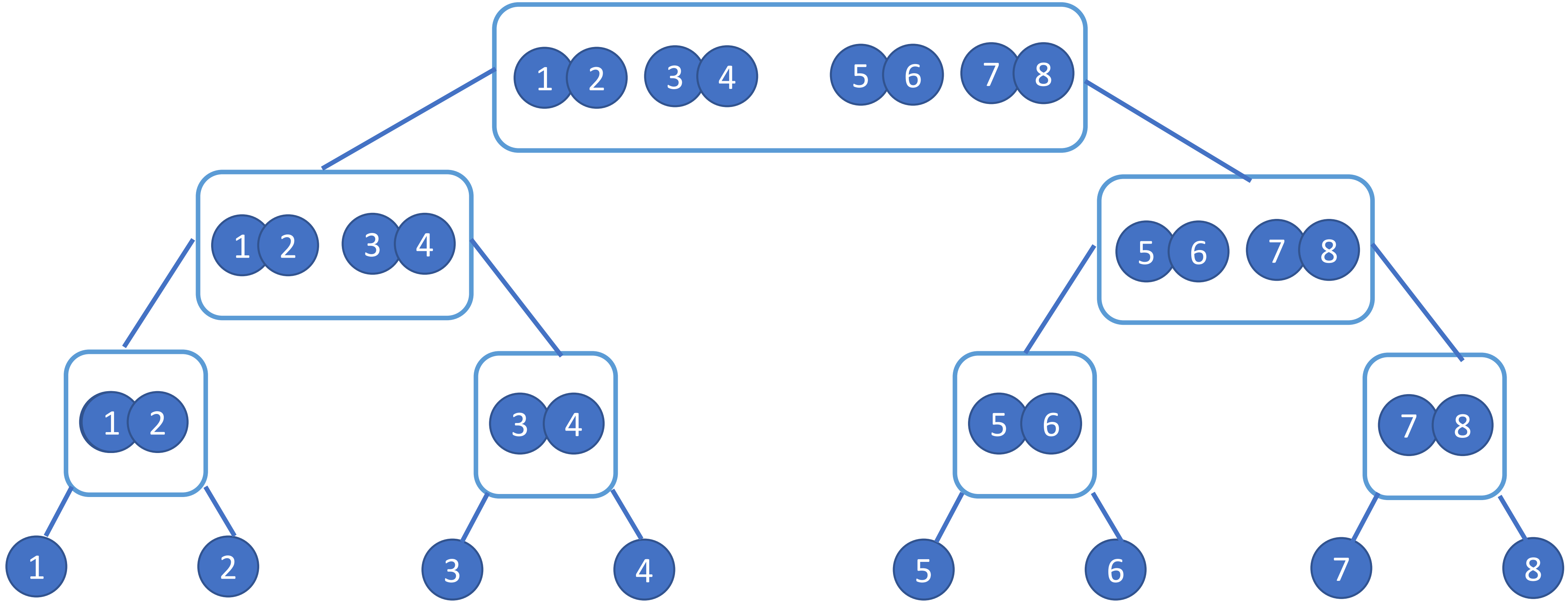
Hierarchical Clustering



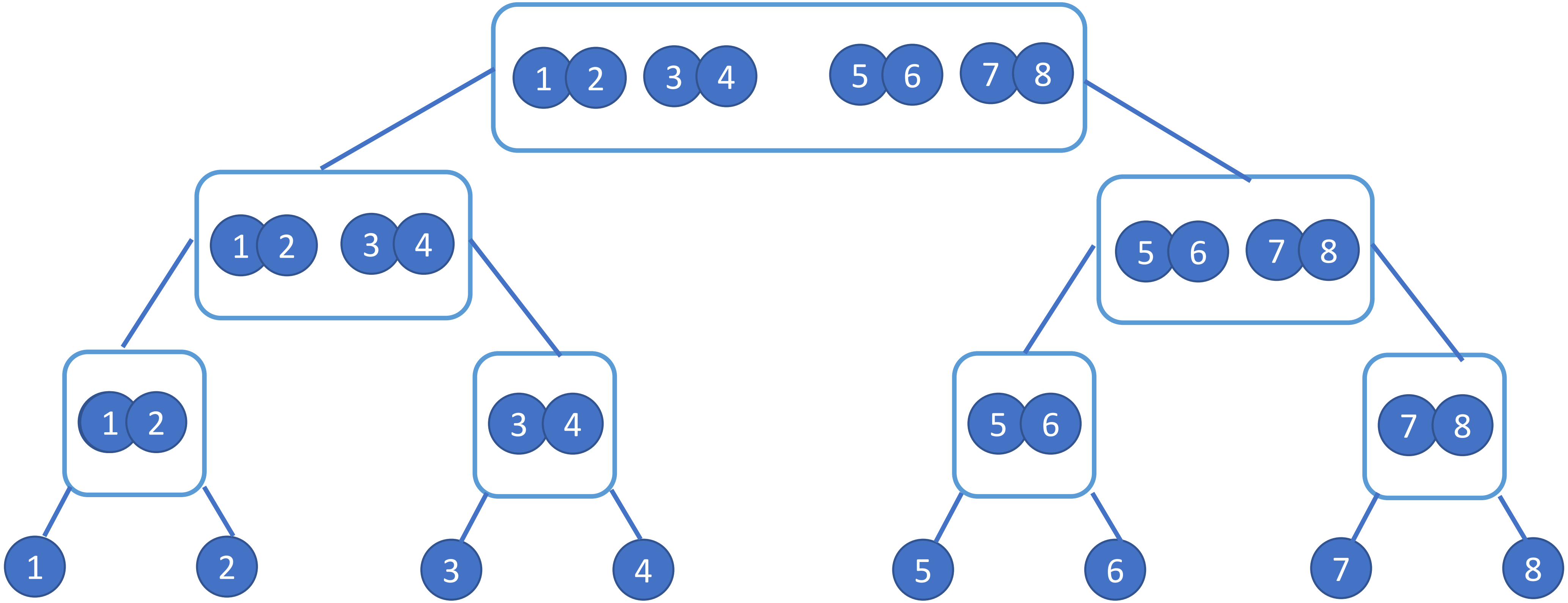
Hierarchical Clustering



Hierarchical Clustering



Hierarchical Clustering



How do you measure the Quality?
How is the splitting performed?

Objective Function

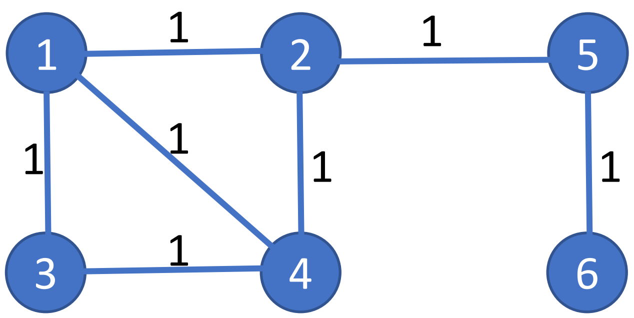
$$cost_{\mathcal{G}}(\mathcal{T}) = \sum_{(i,j)\in\mathcal{E}} w_{ij}|leaves(i \vee j)|$$

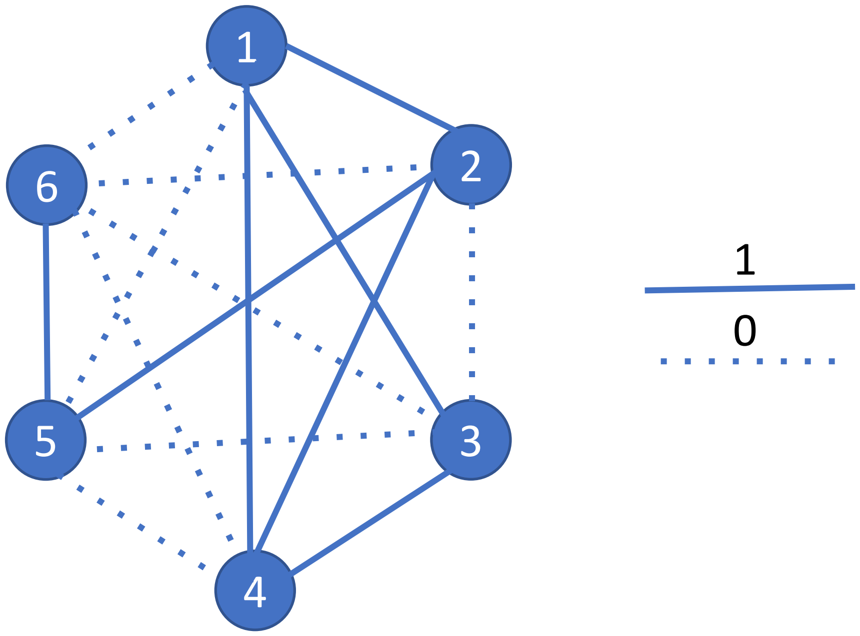
Where, $\mathcal{T}$ is a rooted hierarchical clustering tree for graph $\mathcal{G}$, $leaves(\mathcal{T})$ are the nodes of $\mathcal{G}$, $i \vee j$ denotes the *least common ancestor* of i, j in $\mathcal{T}$, w_{ij} denotes the similarity between nodes i and j, and $|leaves(\mathcal{N})|$ denote the number of leaves in the subtree rooted at $\mathcal{N}$.

$$cost_{\mathcal{G}}(\mathcal{T}^*) = min_{\mathcal{T}}\ cost_{\mathcal{G}}(\mathcal{T})$$

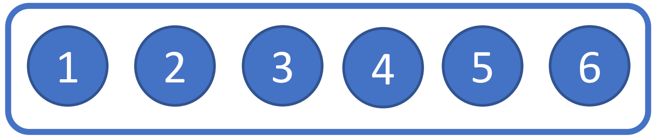
[1] Dasgupta, S. (2015). A cost function for similarity-based hierarchical clustering. *arXiv preprint arXiv:1510.05043*.

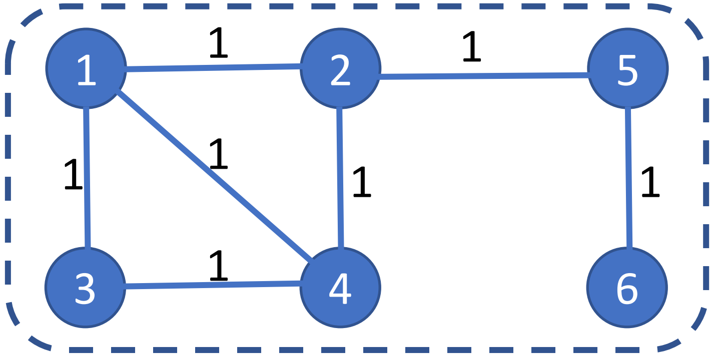
Objective Function



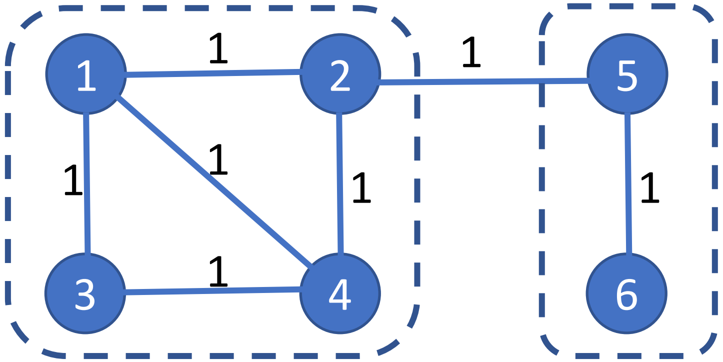


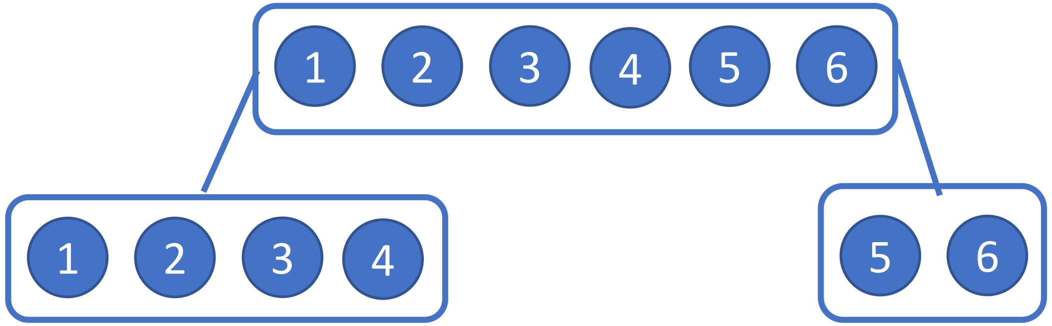
Objective Function





Objective Function



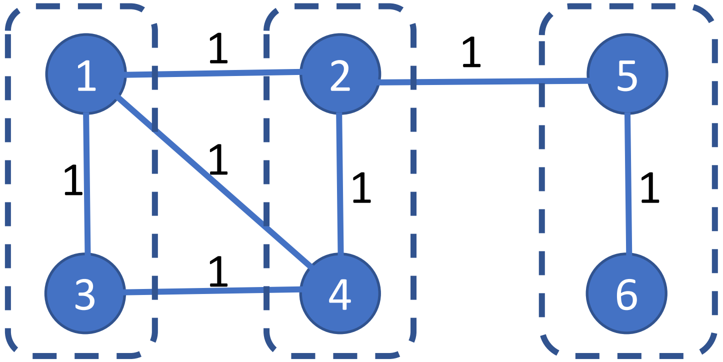


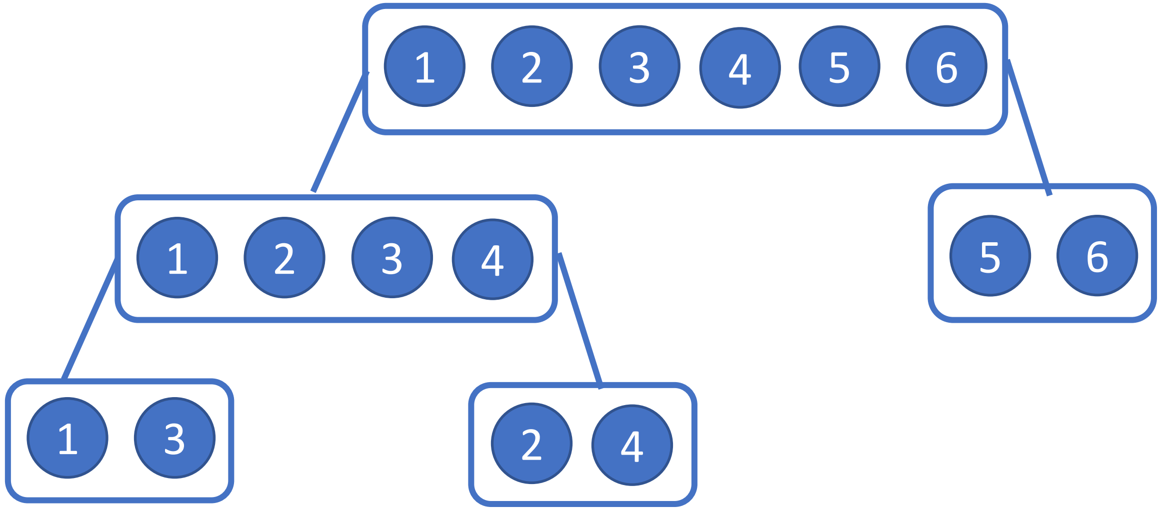
Edges split: (2,5)
Weight of the edge (2,5): 1

Number of nodes in cluster: 6

Cost: 1x6 = 6
Total Cost = 6

Objective Function



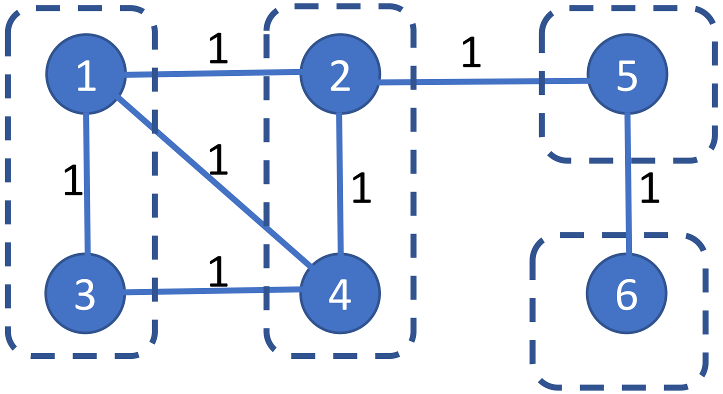


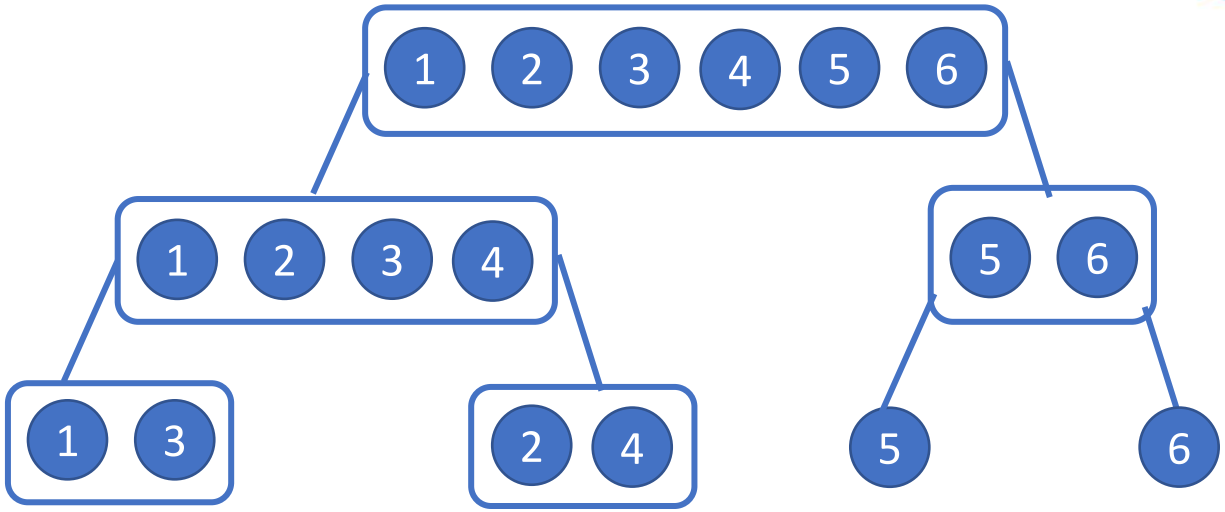
Edges split: (1,2), (1,4), (3,4)
Weight of the edge (1,2): 1
Weight of the edge (1,4): 1
Weight of the edge (3,4): 1

Number of nodes in cluster: 4

Cost: (1+1+1) x 4 = 12
Total Cost = 6 + 12 = 18

Objective Function



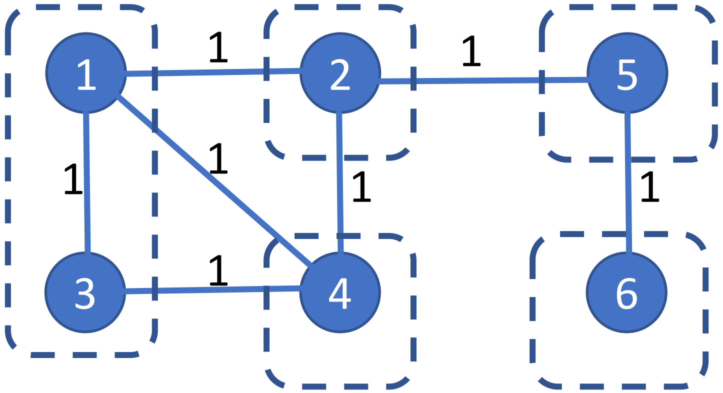


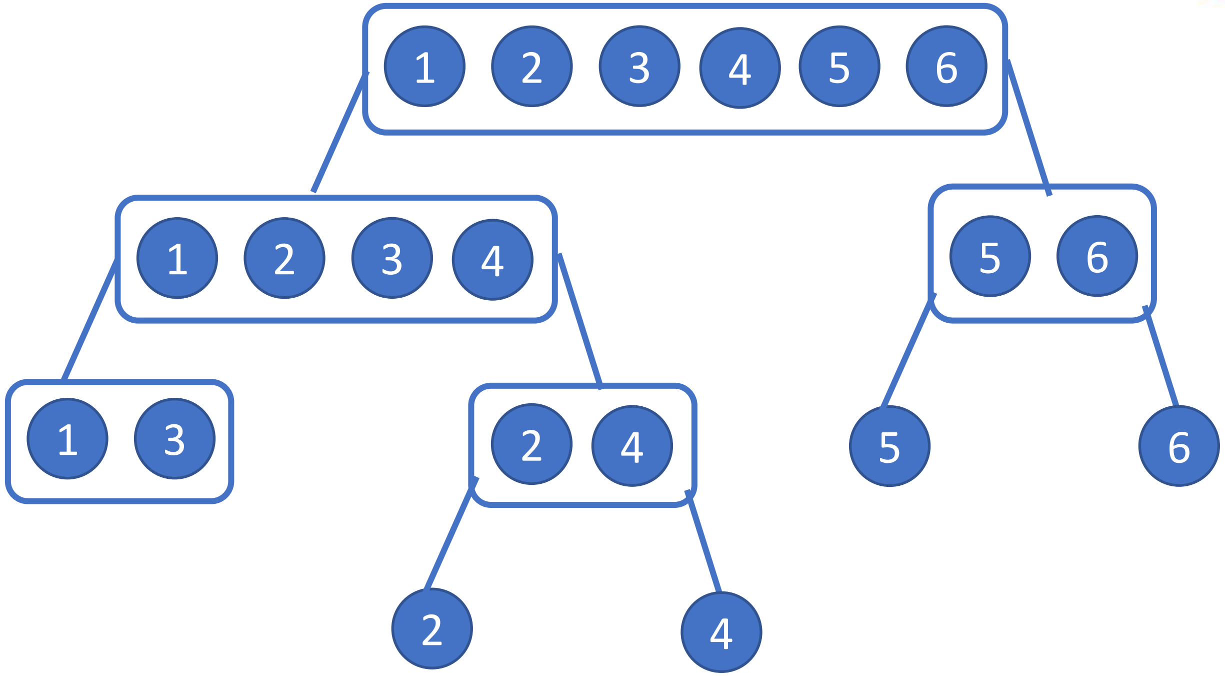
Edges split: (5,6)
Weight of the edge (5,6): 1

Number of nodes in cluster: 2

Cost: 1x2 = 2
Total Cost = 18 + 2 = 20

Objective Function



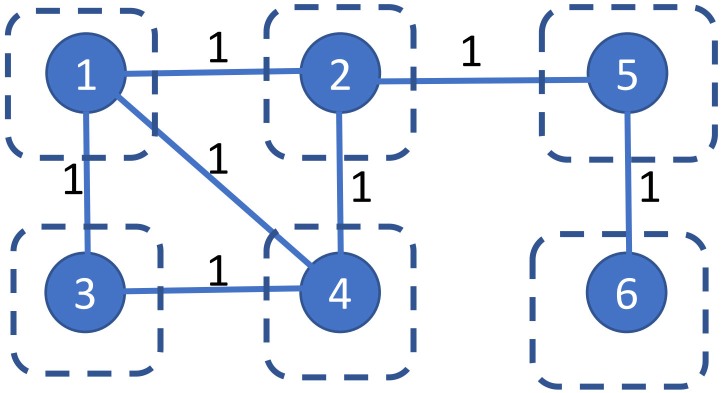


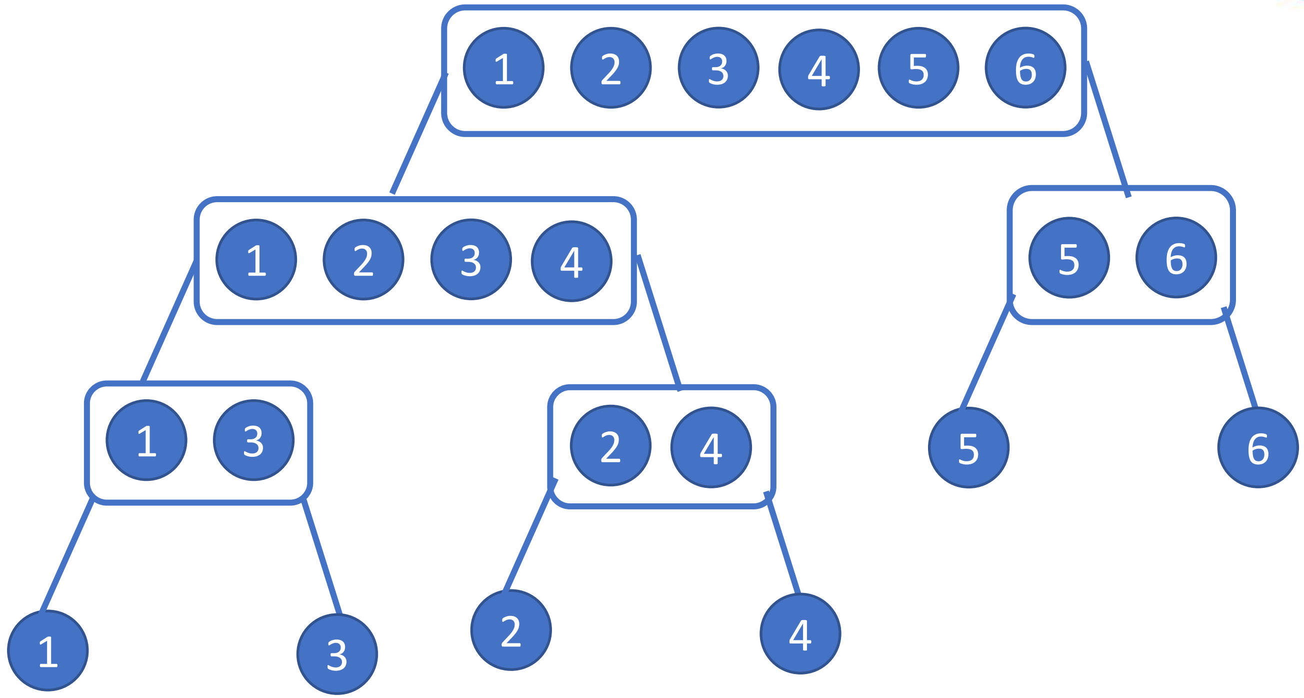
Edges split: (2,4)
Weight of the edge (2,4): 1

Number of nodes in cluster: 2

Cost: 1x2 = 2
Total Cost = 20 + 2 = 22

Objective Function



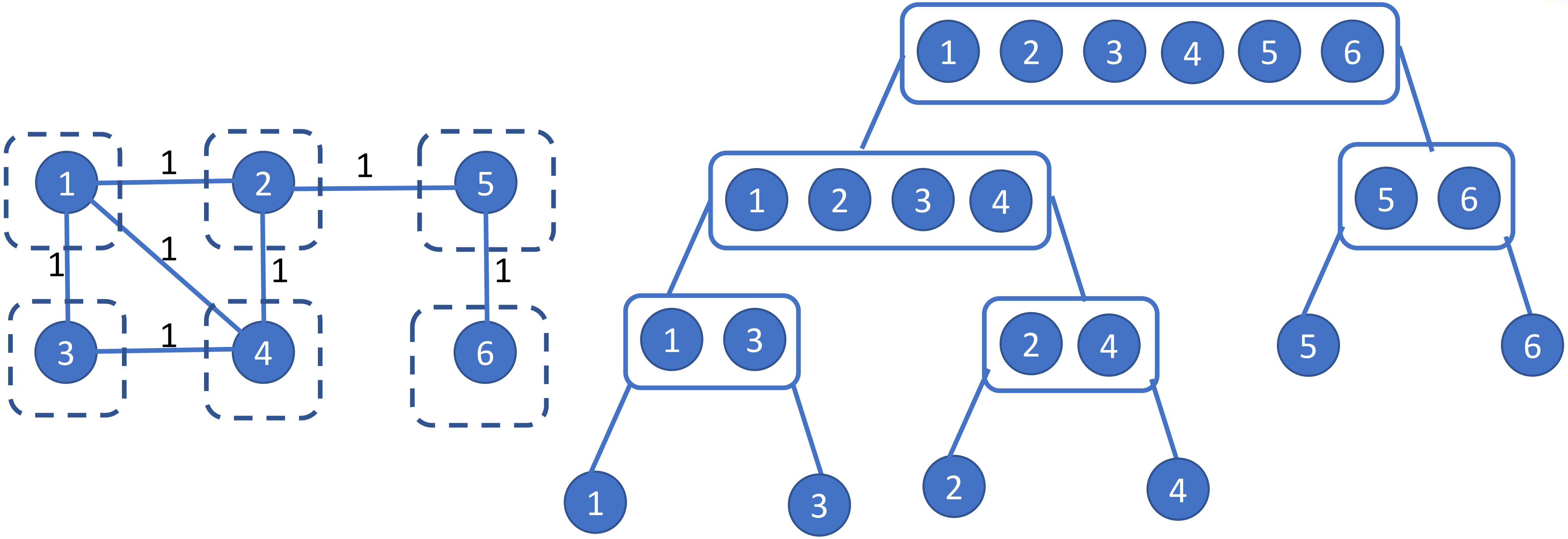


Edges split: (1,3)
Weight of the edge (1,3): 1

Number of nodes in cluster: 2

Cost: 1x2 = 2
Total Cost = 22 + 2 = 24

Objective Function



Total Cost = 24

Objective Function

- NP-Hard!
- Natural Greedy Criterion:
  - Sparsest cut at every internal node.

Sparsest Cut, Expansion and Conductance

For a graph $\mathcal{G}(V, E, w)$, if $|S|$ denotes the number of elements in S, $E(S, \overline{S}) = \sum_{i \in S, j \in \overline{S}} w_{ij}$, and $vol(S) = \sum_{i,j \in S} w_{ij}$, then

Sparsest cut: $\sigma(S) = \frac{E(S,\overline{S})}{|S||\overline{S}|}$, $\quad \sigma(G) = \min_{0<|S|\leq|V|/2} \sigma(S)$

Expansion: $\phi(S) = \frac{E(S,\overline{S})}{min(|S|,|\overline{S}|)}$, $\quad \phi(G) = \min_{0<|S|\leq|V|/2} \phi(S)$

Conductance: $\gamma(S) = \frac{E(S,\overline{S})}{min(vol(S),vol(\overline{S}))}$, $\quad \gamma(G) = \min_{0<|S|\leq|V|/2} \gamma(S)$

[2] Trevisan, L. (2013). Lecture notes on expansion, sparsest cut, and spectral graph theory.

Sparsest Cut, Expansion and Conductance

Cheeger Inequality: If the graph $\mathcal{G}$ is represented by the adjacency matrix $\mathcal{C}$, D is the degree matrix with $D_{ii} = \sum_j C_{ij}$ and we have $\tilde{C} = D^{-1/2} C D^{-1/2}$, if λ_2 is the second **largest** eigenvalue of $\tilde{C}$,

$$\frac{1 - \lambda_2}{2} \leq \phi(\mathcal{G}) \leq \sqrt{2(1 - \lambda_2)}$$

[2] Trevisan, L. (2013). Lecture notes on expansion, sparsest cut, and spectral graph theory.

Splitting Rules

- Related Work [1]
  - Based on Arora Rao Vazirani Algorithm for Sparsest Cut [3]
    - Quality: $O((\sqrt{\log n})OPT)$
    - Time: $\tilde{O}(n^4)$
  - Based on Leighton Rao Algorithm for Sparsest Cut [4]
    - Quality: $O((\log n)OPT)$
    - Time: $O(n^3)$

[1] Dasgupta, S. (2015). A cost function for similarity-based hierarchical clustering. *arXiv preprint arXiv:1510.05043*.
[3] Arora, S., Rao, S., & Vazirani, U. (2009). Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, *56*(2), 5.
[4] Leighton, T., & Rao, S. (1999). Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, *46*(6), 787-832.

Splitting Rules

- Our Work
  - Based on Eigenvectors for Sparsest Cut
    - Quality: $O(n\Delta \log n \sqrt{OPT})$
    - Time: $O(\min(nd^2, n^2d))$

$\Delta = deg_{max}/\sqrt{deg_{min}}$

Splitting Rules

- Our Work
  - Based on Eigenvectors for Sparsest Cut
    - Quality: $O(n\Delta \log n \sqrt{OPT})$
    - Time: $O(\min(nd^2, n^2d))$
  - Based on Approximate Eigenvectors for Sparsest Cut
    - Quality: $O(n\Delta \log n \sqrt{(OPT + \varepsilon)})$
    - Time: $O(\min(nd^2, n^2d))$

$\Delta = deg_{max}/\sqrt{deg_{min}}$

Splitting Rules

- Our Work
  - Based on Eigenvectors for Sparsest Cut
    - Quality: $O(n\Delta \log n \sqrt{OPT})$
    - Time: $O(\min(nd^2, n^2d))$
  - Based on Approximate Eigenvectors for Sparsest Cut
    - Quality: $O(n\Delta \log n \sqrt{(OPT + \varepsilon)})$
    - Time: $O(\min(nd^2, n^2d))$
  - Based on Random Hyperplanes
    - Quality : $O(n\ OPT)$ (Expected Cost)
    - Time : $O(nd)$

$\Delta = deg_{max}/\sqrt{deg_{min}}$

Splitting Rules

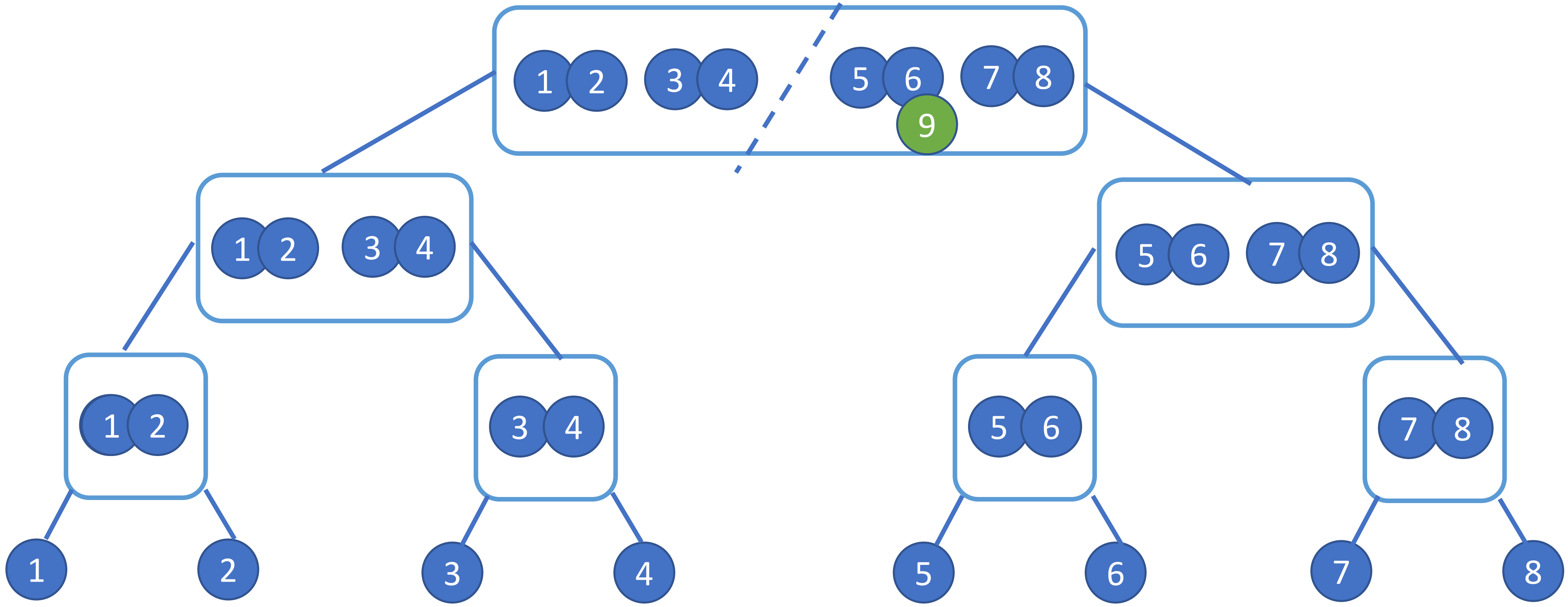
- Our Work
  - Based on Eigenvectors for Sparsest Cut
    - Quality: $O(\ n\Delta \log n \sqrt{OPT})$
    - Time: $O(\min(nd^2, n^2d))$
  - Based on Approximate Eigenvectors for Sparsest Cut
    - Quality: $O(\ n\Delta \log n \sqrt{(OPT + \varepsilon)})$
    - Time: $O(\min(nd^2, n^2d))$
  - Based on Random Hyperplanes
    - Quality : $O(n\ OPT\ )$ (Expected Cost)
    - Time : $O(nd)$
  - 2-Means
    - Quality : ??
    - Time: $O(2nd)$

$\Delta = deg_{max}/\sqrt{deg_{min}}$

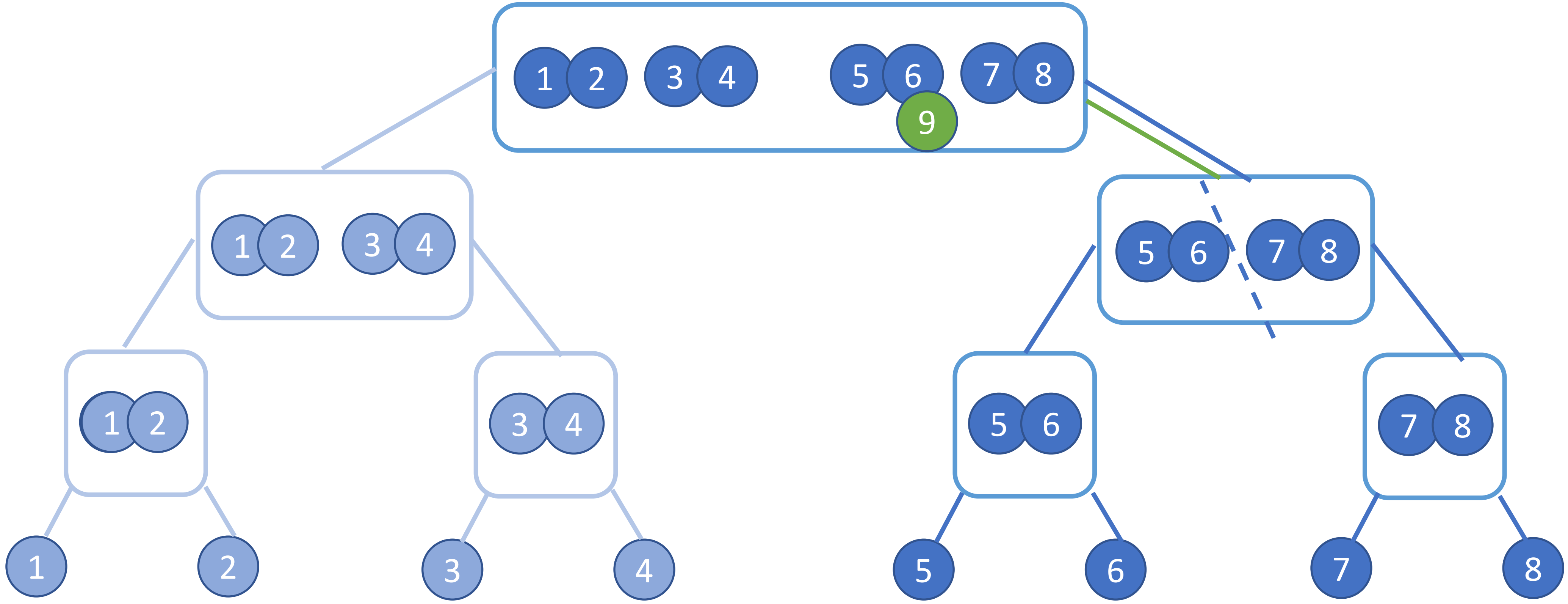
Implementation Details

- Data, X, is normalized.
- Adjacency matrix created as X^TX.
- We work with *(nxd)* matrix $D^{-½}X$ instead of $D^{-½}X^TXD^{-½}$ which is *nxn*.
- Store the *d*-dimensional singular vector.
- Data is not normalized for large datasets.
- Balanced split point is chosen for practical reasons.

How do we use it for querying?



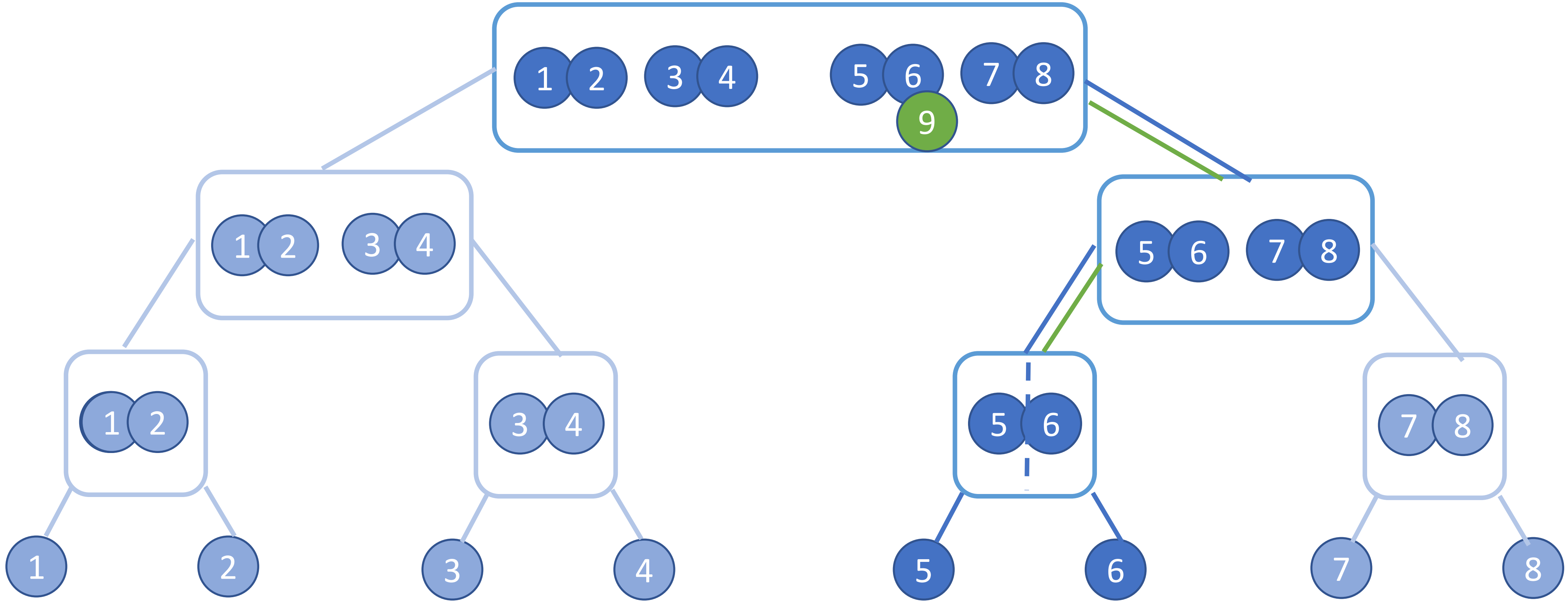
How do we use it for querying?



Candidate Size: 3 or 4

Return: 5 6 7 8

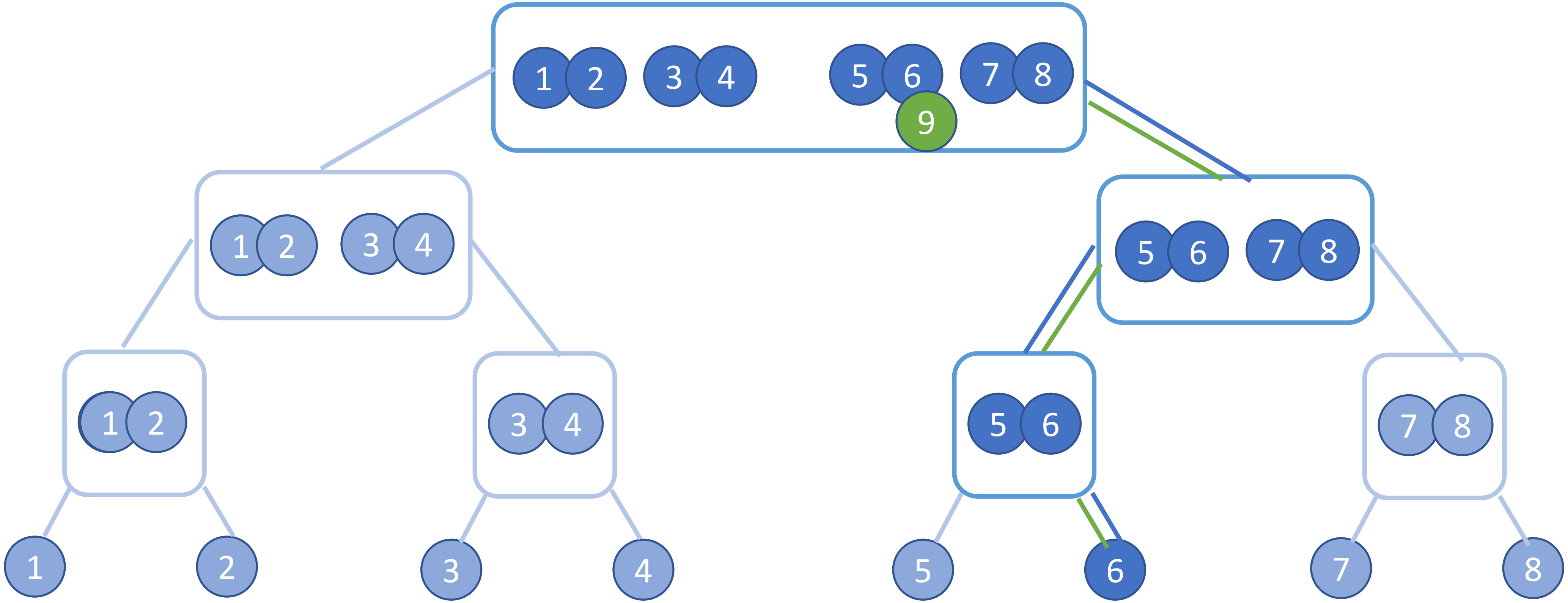
How do we use it for querying?



Candidate Size: 2

Return: 5 6

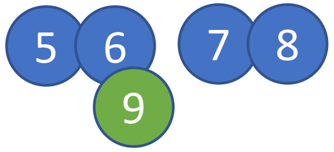
How do we use it for querying?



Candidate Size: 1

Return: 6

How do we find Nearest Neighbors



- Let Candidate Set Size be 4
- Returned points: 5 6 7 8

- Suppose we want 3 approximate nearest neighbors
- Calculate distance of 9 from each of 5 6 7 8

- Return the 3 closest ones, I.e., 6 5 7

Classification using the Hierarchy

- Fix some candidate set size, and number of nearest neighbors desired
- Traverse the tree and find the nearest neighbors
- All the points returned have a label associated with them
- Take the maximum vote among them

Classification Results

Dataset	Number of Classes	Dimensions	Train Size	Test Size
MNIST	10	784	60000	10000
ALOI	1000	128	88000	20000
CoverType	7	53	480102	100000
Comments (LinkedIn)	3	50	106537	20000

Comments Dataset

- 3 Classes
  - Spam: Content which is offensive, harmful, abusive or disruptive. These comments violate the Terms and Conditions of the networking site and are to be completely removed from the site.
  - Low-Quality: Content which is irrelevant to the discussion or unappealing. Such content may receive a diminished visibility on the site.
  - Clear: Content which is okay to receive unrestricted distribution on the site.
- Due to business requirements, treated as 2 binary classification problems.

Classification Results

	Method	MNIST Precision	Recall	F1-Score	Query Time	ALOI Precision	Recall	F1-Score	Query Time
Hierarchical Tree Based Methods	EV	**0.93**	**0.929**	**0.93**	**0.702**	**0.893**	**0.889**	**0.891**	**0.701**
	AEV	0.924	0.924	0.924	0.729	0.893	0.890	0.891	0.690
	RP	0.818	0.810	0.814	0.789	0.776	0.768	0.772	0.550
	2-means	0.929	0.929	0.929	1.243	0.892	0.889	0.890	0.97
Linkage	Single	0.114	0.955	0.204	~2k	0.036	0.670	0.069	~2k
	Average	0.100	0.999	0.182	~2k	0.001	0.971	0.002	~2k
	Complete	0.180	0.393	0.247	~2k	0.124	0.519	0.200	~2k
	Wards	0.548	0.602	0.574	~2k	0.393	0.562	0.463	~2k
	SVM	**0.930**	**0.930**	**0.930**	**-**	**0.843**	**0.830**	**0.836**	**-**
ANN	LSH	**0.954**	**0.953**	**0.953**	**0.850**	**0.936**	**0.920**	**0.934**	**0.68**
	Kd-Tree	**0.969**	**0.968**	**0.968**	**540.38**	**0.949**	**0.948**	**0.948**	**22.30**

Classification Results

Method	CoverType				Comments Clear vs (Low Quality + Spam)			Comments (Clear + Low Quality) vs Spam		
	Precision	Recall	F1-Score	Query Time	Precision	Recall	F1-Score	Precision	Recall	F1-Score
EV	**0.925**	**0.921**	**0.923**	**0.380**	**0.750**	**0.840**	**0.80**	**0.739**	**0.740**	**0.740**
AEV	0.925	0.922	0.923	0.370	0.760	0.840	0.80	0.738	0.740	0.740
RP	0.904	0.897	0.901	0.370	-	-	-	-	-	-
2-means	0.930	0.930	0.930	0.810	0.63	0.61	0.62	0.64	0.70	0.67
SVM	0.211	0.308	0.250	-	-	-	-	-	-	-
LSH	0.581	0.739	0.650	44.72	**0.71**	**0.74**	**0.73**	**0.62**	**0.63**	**0.62**
Kd-Tree	**0.942**	**0.935**	**0.938**	**0.760**	-	-	-	-	-	-

Anomaly Detection using the hierarchy

- Maintain a lookup table with class-wise average pairwise distances
- Classify the query (say C)
- Find the average distance of the query from the nearest neighbors, (say d1)
- Lookup average pairwise distance for C (say d2)
- If d2 + threshold < d1
  - Detect it as an anomaly
- Else
  - Report its' class

SVM as Baseline for Anomaly Detection

- Use prediction probability
- For a non-anomaly point, the prediction probability of its' actual class is high
- If prediction probability of the query < threshold
    - Detect it as anomaly
- Else
    - Report its' class

Anomaly Detection Results

- Dataset Creation
  - ALOI Dataset partitioned into two subsets
    - Train Set: 950 classes,
    - Test Set: 50 **new** classes + points from old classes

Dataset	Train Set Size	Test Set Size	Number of points from	
			Unseen Class	Seen Class
Dataset 1	92451	15549	5400	10149
Dataset 2	92443	15557	5400	10157
Dataset 3	92288	15712	5400	10312

Anomaly Detection Results

	Approx Eigenvector Threshold	Precision	Recall	F1 Score	Support Vector Machine Threshold	Precision	Recall	F1 Score
Dataset 1	**0.2**	**0.53**	**0.78**	**0.63**	**0.05**	**0.42**	**0.82**	**0.55**
	0.3	0.58	0.66	0.61	0.08	0.42	0.76	0.54
	0.4	0.62	0.53	0.57	0.1	0.42	0.56	0.48
Dataset 2	**0.2**	**0.53**	**0.80**	**0.64**	0.05	0.42	0.84	0.56
	0.3	0.58	0.66	0.62	**0.08**	**0.44**	**0.80**	**0.56**
	0.4	0.63	0.54	0.58	0.1	0.45	0.61	0.52
Dataset 3	**0.2**	**0.52**	**0.78**	**0.62**	0.05	0.42	0.86	0.57
	0.3	0.57	0.67	0.62	**0.08**	**0.43**	**0.78**	**0.55**
	0.4	0.61	0.53	0.57	0.1	0.43	0.60	0.50

Requirements Satisfied

- Unsupervised ✓
- Real-time classification ✓
- Incremental updates to the model ✗
- Potentially use the subtype-supertype relationship ✓

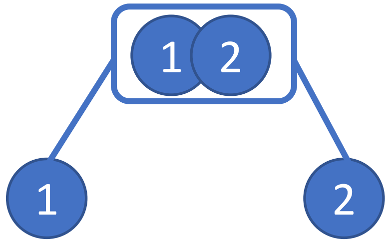
Online Hierarchical Clustering

- Has the ability to perform incremental updates to the model.
- Recent work:
  - [1] proposed an algorithm based on bounding boxes called PERCH.
- Our work:
  - We propose two heuristics.

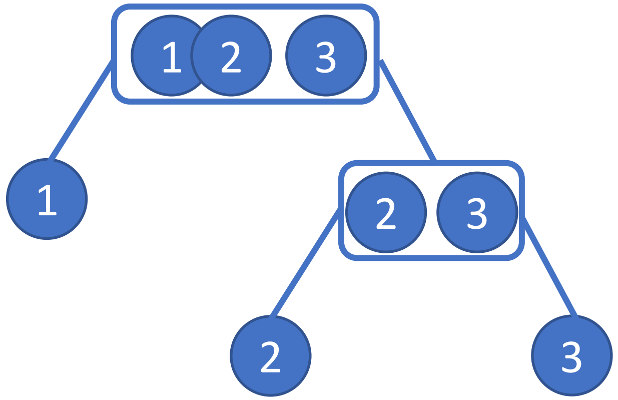
[5] Kobren, A., Monath, N., Krishnamurthy, A., & McCallum, A. (2017, August). A hierarchical algorithm for extreme clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 255-264). ACM.

Online Hierarchical Clustering

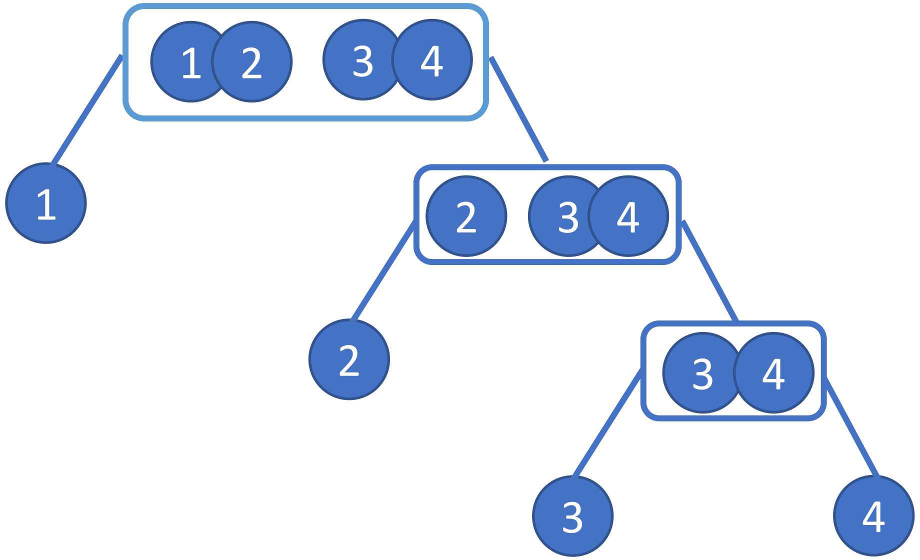
Online Hierarchical Clustering



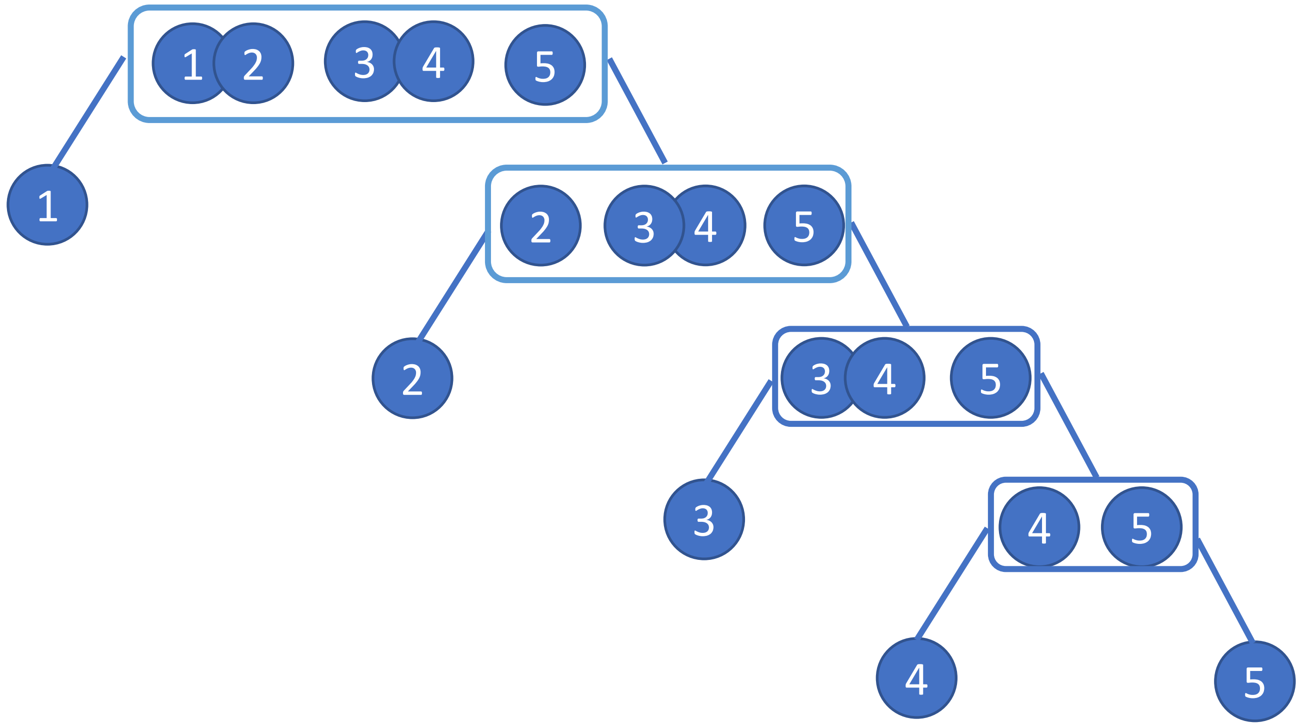
Online Hierarchical Clustering



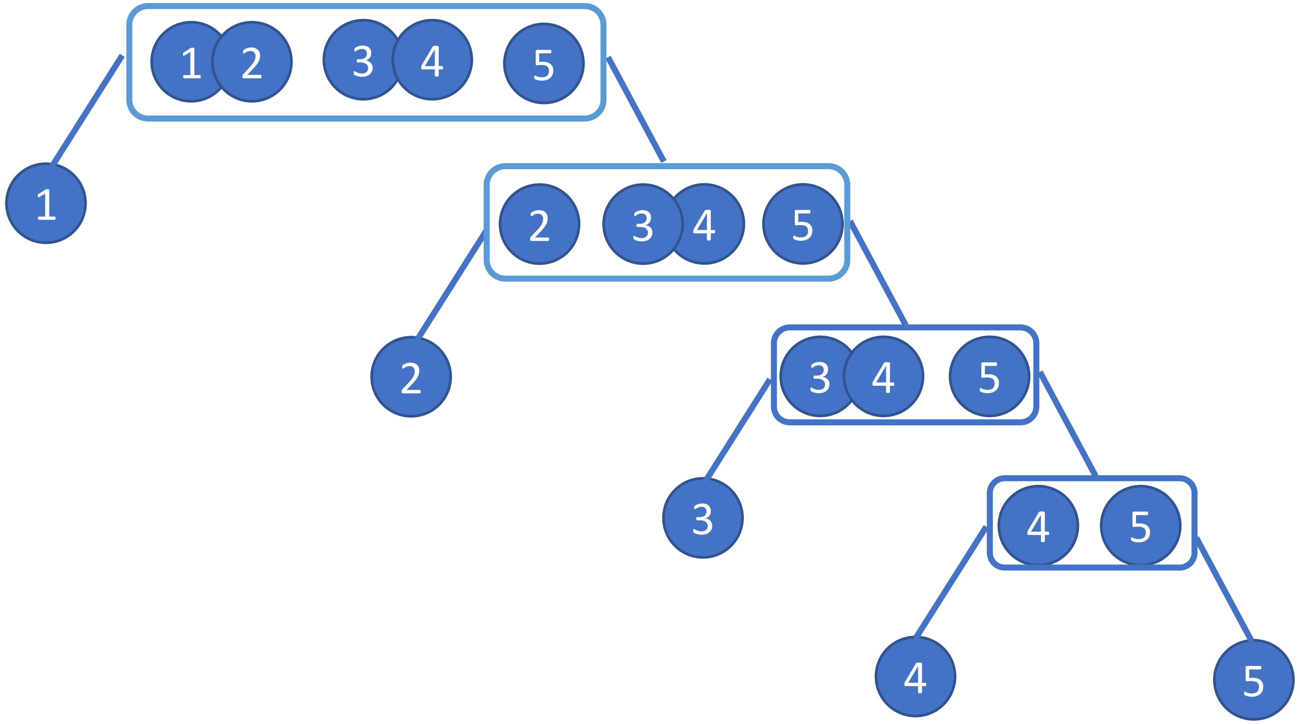
Online Hierarchical Clustering



Online Hierarchical Clustering

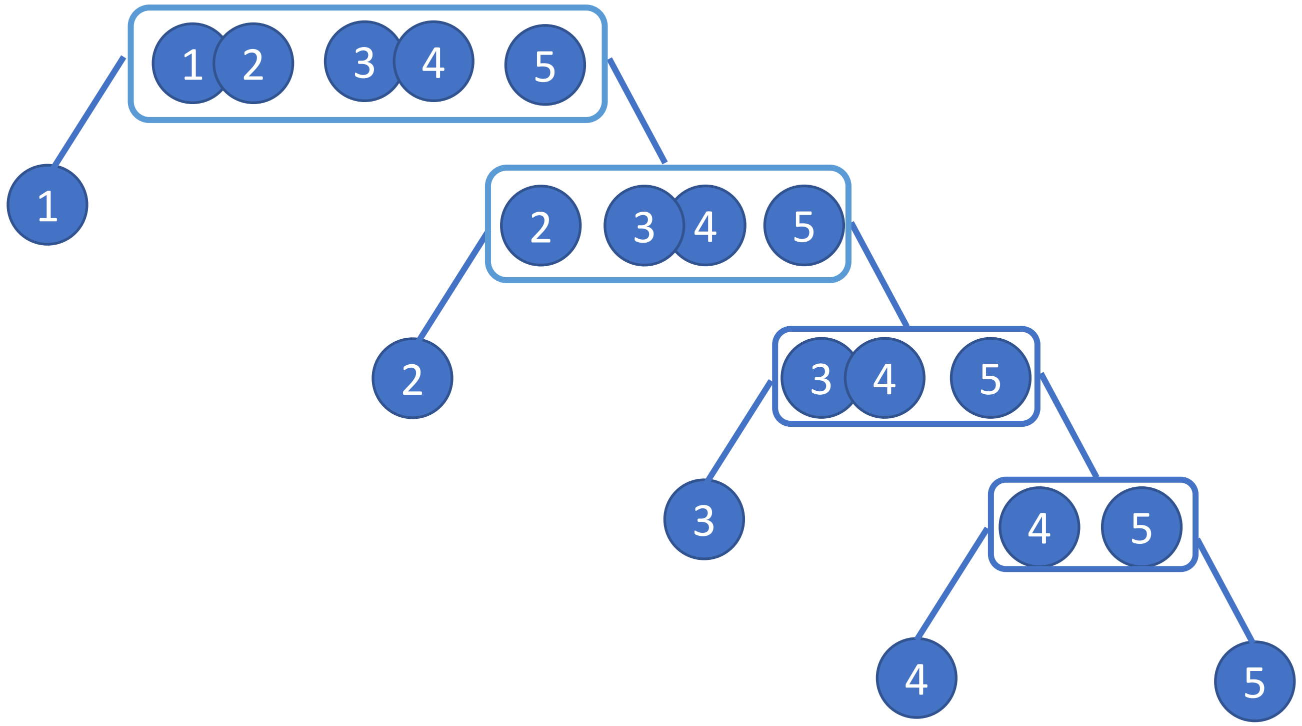


Online Hierarchical Clustering



Query Time?

Online Hierarchical Clustering



Query Time? O(n)

Doubling Heuristic

- Rebuild subtree when the number of points at root of subtree is doubled.

Balancing Heuristic

- Rebuild subtree when balance of subtree is distorted.
- Unbalanced if number of nodes in one side is more than double of nodes in the other side.

Results

Dataset	Doubling Heuristic			Balancing Heuristic		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
MNIST	0.917	0.916	0.916	0.893	0.890	0.891
ALOI	0.826	0.806	0.816	0.807	0.795	0.801
CoverType	0.880	0.877	0.878	0.882	0.876	0.879

Results

Method	(Clear + Low Quality) vs Spam			Clear vs (Low Quality + Spam)		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
AEV	0.753	0.853	0.80	0.685	0.686	0.685
PERCH	0.708	0.971	0.82	0.48	0.454	0.466

Requirements Satisfied

- Unsupervised ✓
- Real-time classification ✓
- Incremental updates to the model ✓
- Potentially use the subtype-supertype relationship ✓

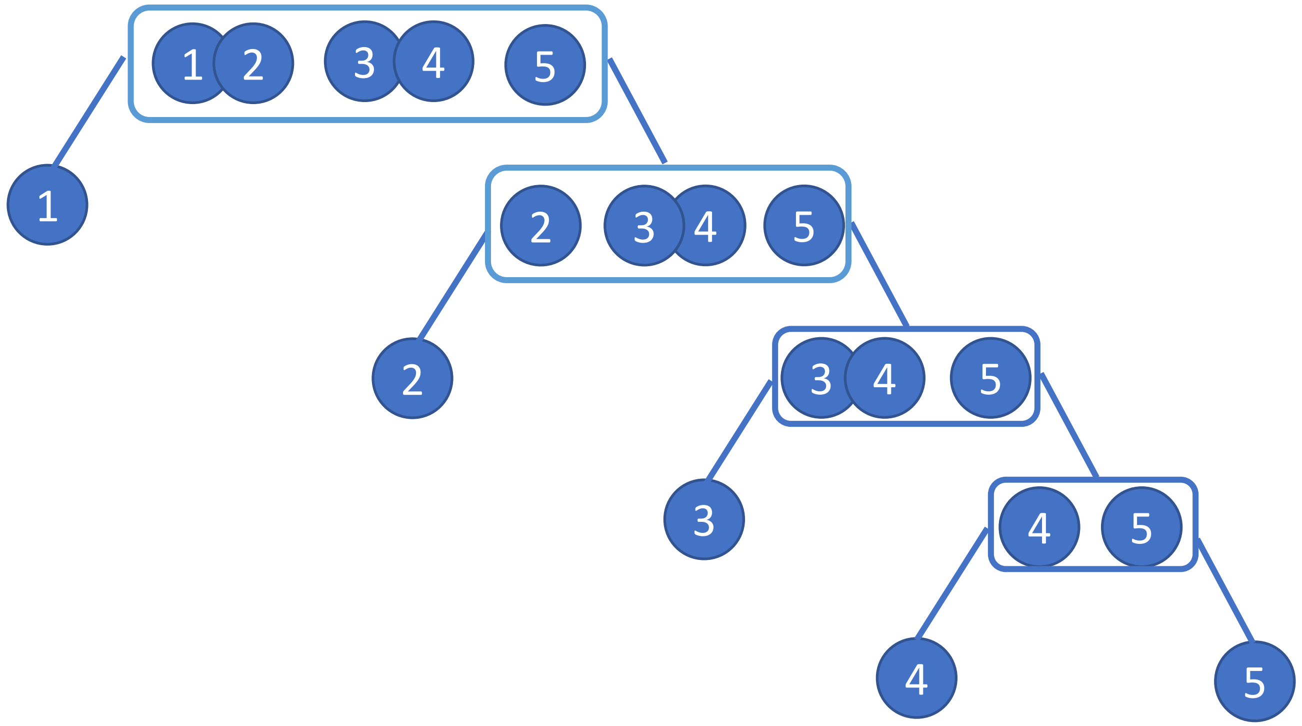
Limitations

- No theoretical bound on the quality of the solution

Next Steps

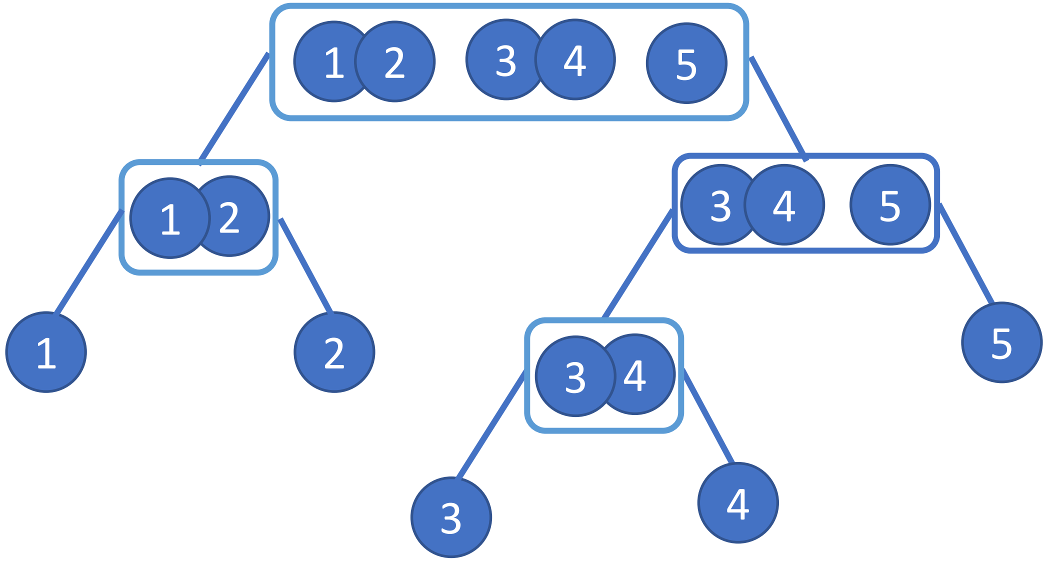
- Recall that we had a splitting strategy of using eigenvectors.
- Can we somehow update the eigenvectors in an online way?

Online Hierarchical Clustering



Query Time? O(n)

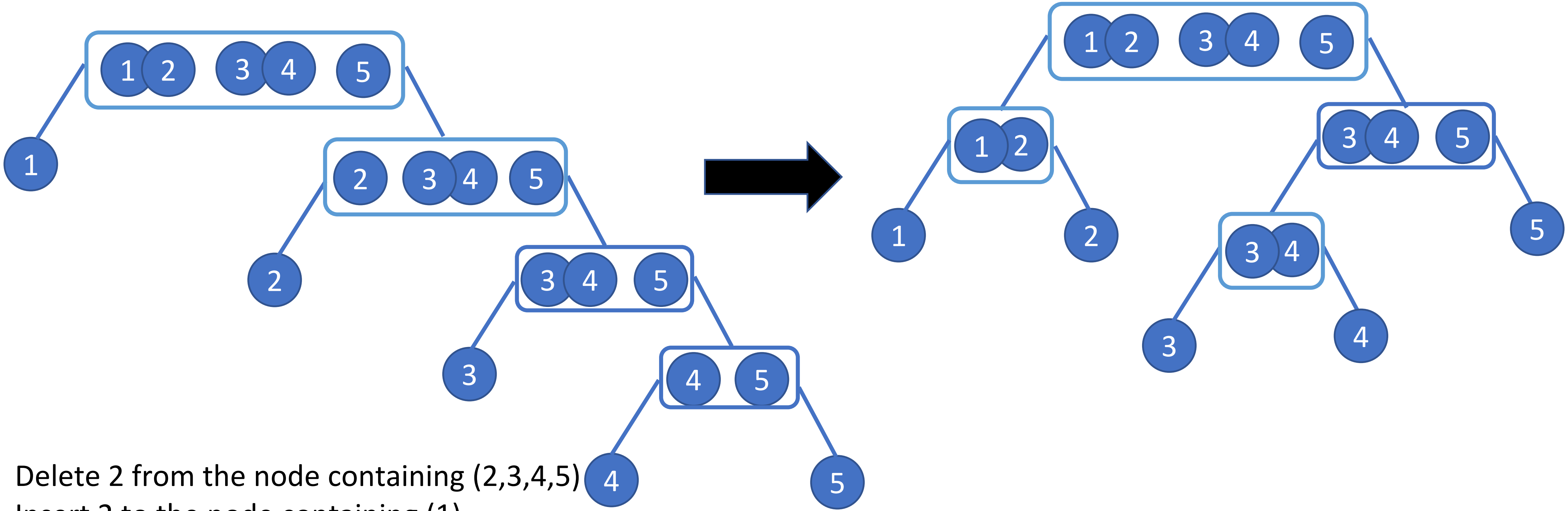
Online Hierarchical Clustering



Logarithmic querying time

Points that are close are separated lower in the tree

Online Hierarchical Clustering



Delete 2 from the node containing (2,3,4,5)
Insert 2 to the node containing (1)
Delete 4 from the node containing (4,5)
Insert 4 to the node containing (3)

Next Steps

- Recall that we had a splitting strategy of using eigenvectors
- Can we somehow update the eigenvectors in a streaming way?
- But we need insertion and deletion of points!

Online Eigenvector Updates

- Partitioning in Dynamic Graphs
- Sensors
- Removal of Adversarial Inputs

Online Eigenvector Updates

- Related work when data is only inserted
  - Streaming PCA with limited memory, evaluated on the spiked covariance model [6].
  - Based on regret minimization [7].
  - Sub-sampling and dimensionality reduction [8,9].

[6] Mitliagkas, I., Caramanis, C., & Jain, P. (2013). Memory limited, streaming PCA. In Advances in Neural Information Processing Systems (pp. 2886-2894).
[7] Garber, D., Hazan, E., & Ma, T. (2015, July). Online Learning of Eigenvectors. In ICML (pp. 560-568).
[8] Clarkson, K. L., & Woodruff, D. P. (2009, May). Numerical linear algebra in the streaming model. In Proceedings of the forty-first annual ACM symposium on Theory of computing (pp. 205-214). ACM.
[9] Halko, N., Martinsson, P. G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM review, 53(2), 217-288.

Orthogonal Dual Spiked Covariance Model

- Inspired by the Spiked Covariance Model [6]
- Insert from
  - $\mathbf{x_t} = \mathbf{u}z_t + \mathbf{w_t}$
  - $\mathbf{x_t} = \mathbf{v}z_t + \mathbf{y_t}$
- Delete from
  - $\mathbf{x_t} = \mathbf{v}z_t + \mathbf{y_t}$

where $z_t \sim N(0,1)$, $\mathbf{w_t} \sim N(0, \sigma_1^2 I)$, $\mathbf{y_t} \sim N(0, \sigma_2^2 I)$

All z_t, $\mathbf{w_t}$, $\mathbf{y_t}$ are mutually independent and $\mathbf{u^T v} = \mathbf{v^T u} = \mathbf{0}$.

[6] Mitliagkas, I., Caramanis, C., & Jain, P. (2013). Memory limited, streaming PCA. In Advances in Neural Information Processing Systems (pp. 2886-2894).

Algorithm

procedure $\textsc{Insert}(B, \mathbf{q}, \lambda)$
  $\mathbf{s} \leftarrow 0$
  for $\mathbf{x_t} \in B$ **do**
    $\mathbf{y} \leftarrow < \mathbf{q}, \mathbf{x_t} > \mathbf{x_t}$
    $\mathbf{s} \leftarrow \mathbf{s} + \mathbf{y}$
    $\lambda \leftarrow \lambda + < \mathbf{q}, \mathbf{y} >$
  $\mathbf{s} \leftarrow \frac{1}{B}\mathbf{s}$
  $\mathbf{q} \leftarrow \frac{\mathbf{s}}{||\mathbf{s}||_2}$
  return $\mathbf{q}, \lambda$

procedure $\textsc{Delete}(B, \mathbf{q}, \lambda)$
  $\mathbf{s} \leftarrow 0$
  $\lambda_t \leftarrow 0$
  for $\mathbf{x_t} \in B$ **do**
    $\mathbf{y} \leftarrow < \mathbf{q}, \mathbf{x_t} > \mathbf{x_t}$
    $\mathbf{s} \leftarrow \mathbf{s} + \mathbf{y}$
    $\lambda_t \leftarrow \lambda_t + < \mathbf{q}, \mathbf{y} >$
  $\mathbf{s} \leftarrow \lambda\mathbf{q} - \mathbf{s}$
  $\mathbf{q} \leftarrow \frac{\mathbf{s}}{||\mathbf{s}||_2}$
  $\lambda \leftarrow \lambda - \lambda_t$
  return $\mathbf{q}, \lambda$

Theoretical Guarantees

$\mathbf{q}_\tau = \sqrt{\alpha_\tau}\mathbf{u} + \sqrt{\gamma_\tau}\mathbf{v} + \sqrt{\delta_\tau}\mathbf{g}_\tau$ where $\mathbf{g}_\tau$ is orthogonal to both $\mathbf{u}$, $\mathbf{v}$, and $\alpha_\tau + \gamma_\tau + \delta_\tau = 1$.

$$\alpha_\tau = (\mathbf{u}^\mathbf{T}\mathbf{q}_\tau)^2 = \frac{(\mathbf{u}^\mathbf{T}\mathbf{s}_\tau)^2}{||\mathbf{s}_\tau||_2^2}$$

$$\gamma_\tau = (\mathbf{v}^\mathbf{T}\mathbf{q}_\tau)^2 = \frac{(\mathbf{v}^\mathbf{T}\mathbf{s}_\tau)^2}{||\mathbf{s}_\tau||_2^2}$$

$$\delta_\tau = (\mathbf{g}_\tau^\mathbf{T}\mathbf{q}_\tau)^2 = \frac{(\mathbf{g}_\tau^\mathbf{T}\mathbf{s}_\tau)^2}{||\mathbf{s}_\tau||_2^2}$$

Insertion

Lemma 1.1. *On insertion of a batch of points sampled from Model 1, the components along* $\mathbf{u}$, $\mathbf{v}$ *and* $\mathbf{g}_\mathcal{T}$ *change as:*

$$\alpha_{\mathcal{T}+1} = \frac{\alpha_\mathcal{T}(1+\sigma_1^2+\epsilon)^2}{(1+\sigma_1^2+\epsilon)^2 - (1-\alpha_\mathcal{T})(1+2(\sigma_1^2+\epsilon))}$$

$$\gamma_{\mathcal{T}+1} = \frac{\gamma_\mathcal{T}(\sigma_1^2+\epsilon)^2}{(\sigma_1^2+\epsilon)^2 + \alpha_\mathcal{T}(1+2(\sigma_1^2+\epsilon))}$$

$$\delta_{\mathcal{T}+1} = \frac{\delta_\mathcal{T}(\sigma_1^2+\epsilon)^2}{(\sigma_1^2+\epsilon)^2 + \alpha_\mathcal{T}(1+2(\sigma_1^2+\epsilon))}$$

Insertion

Lemma 1.2. *On insertion of a batch of points sampled from Model 2, the components along* $\mathbf{u}$, $\mathbf{v}$ *and* $\mathbf{g}_\tau$ *change as:*

$$\alpha_{\tau+1} = \frac{\alpha_\tau(\sigma_2^2 + \epsilon)^2}{(\sigma_2^2 + \epsilon)^2 + \gamma_\tau(1 + 2(\sigma_2^2 + \epsilon))}$$

$$\gamma_{\tau+1} = \frac{\gamma_\tau(1 + \sigma_2^2 + \epsilon)^2}{(1 + \sigma_2^2 + \epsilon)^2 - (1 - \gamma_\tau)(1 + 2(\sigma_2^2 + \epsilon))}$$

$$\delta_{\tau+1} = \frac{\delta_\tau(\sigma_2^2 + \epsilon)^2}{(\sigma_2^2 + \epsilon)^2 + \gamma_\tau(1 + 2(\sigma_2^2 + \epsilon))}$$

Deletion

Lemma 1.3. *On deletion of a batch of points sampled from Model 2, the components along* $\mathbf{u}$, $\mathbf{v}$ *and* $\mathbf{g}_\tau$ *change as:*

$$\alpha_{\tau+1} = \frac{\alpha_\tau(\lambda_\tau - B(\sigma_2^2 + \epsilon))^2}{(\lambda_\tau - B(\sigma_2^2 + \epsilon))^2 - \gamma_\tau(2B\lambda_\tau - B^2(1 + 2\sigma_2^2 + 2\epsilon))}$$

$$\gamma_{\tau+1} = \frac{\gamma_\tau(\lambda_\tau - B(1 + \sigma_2^2 + \epsilon))^2}{(\lambda_\tau - B(1 + \sigma_2^2 + \epsilon))^2 + (1 - \gamma_\tau)(B^2(1 + 2\sigma_2^2 + 2\epsilon) - 2B\lambda_\tau)}$$

$$\delta_{\tau+1} = \frac{\delta_\tau(\lambda_\tau - B(\sigma_2^2 + \epsilon))^2}{(\lambda_\tau - B(\sigma_2^2 + \epsilon))^2 - \gamma_\tau(2B\lambda_\tau - B^2(1 + 2\sigma_2^2 + 2\epsilon))}$$

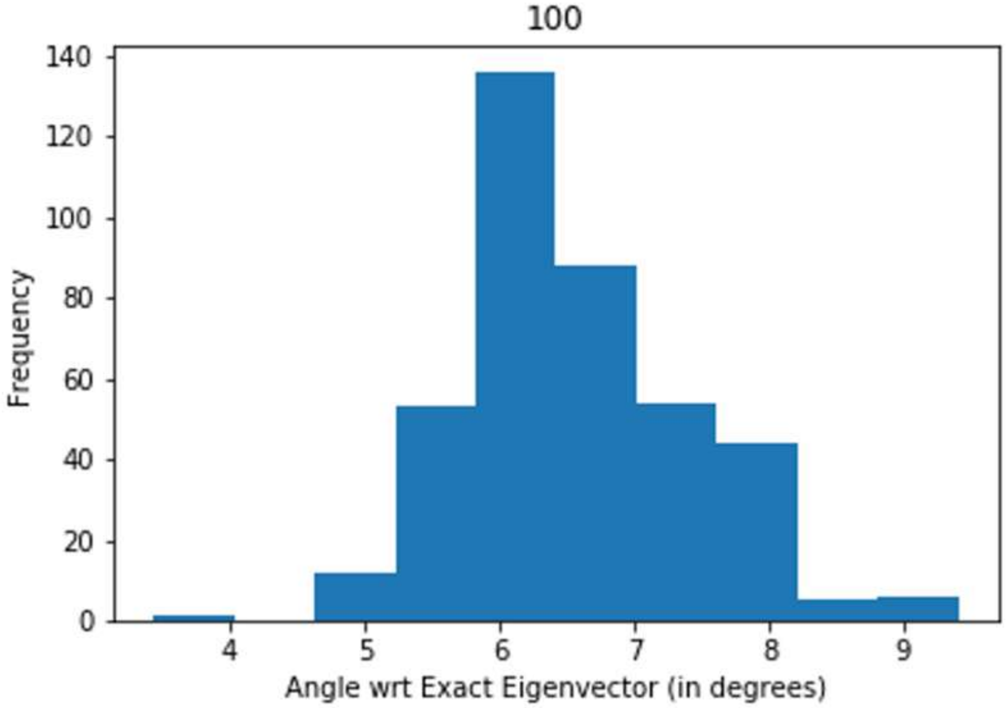
Theoretical Guarantee

Theorem 1.4. *For any batch of insertions or deletions,*

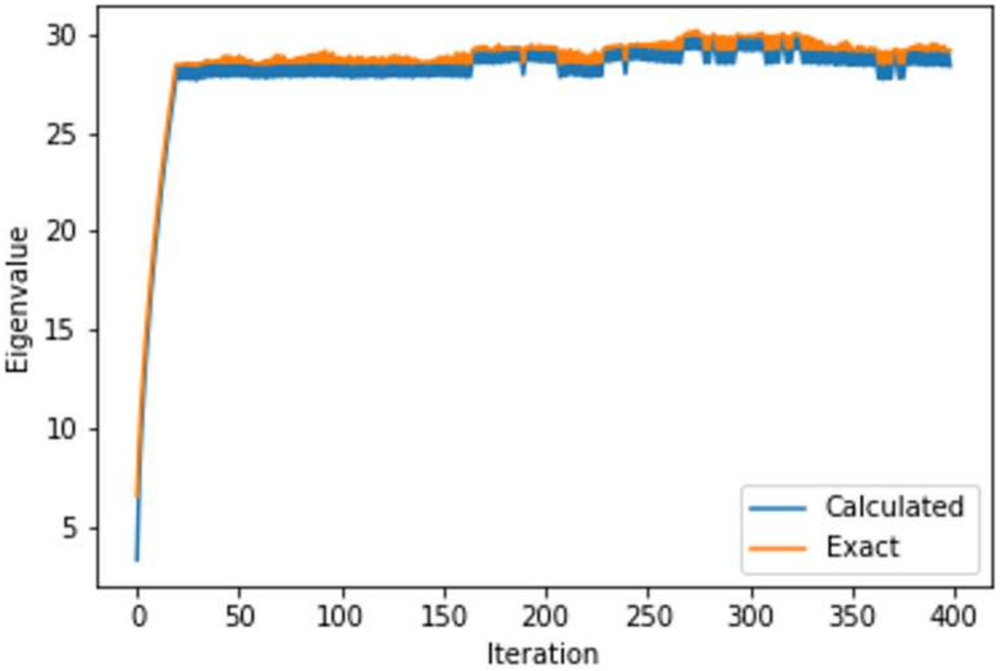
$$\frac{\alpha_{\tau+1} + \delta_{\tau+1}}{\alpha_\tau + \delta_\tau} \leq \frac{\alpha_{\tau+1}}{\alpha_\tau}$$

Results

Batch Size = 100, 40k operations, Probability of Insertion = 0.5



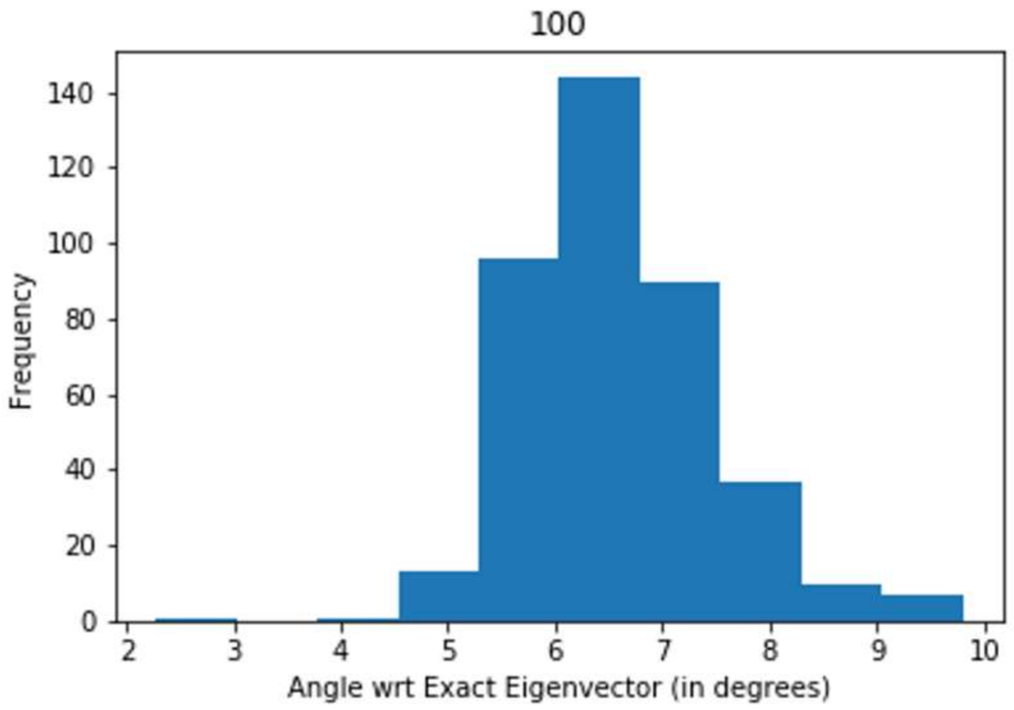
Error (Degrees)



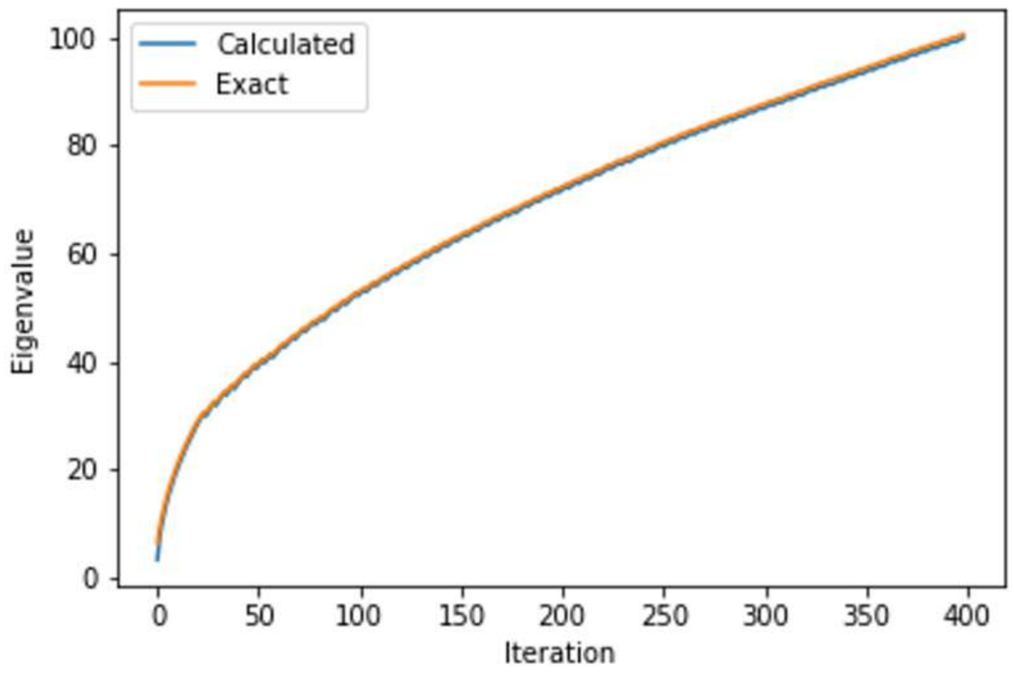
Eigenvalue

Results

Batch Size = 100, 40k operations, Probability of Insertion = 0.8



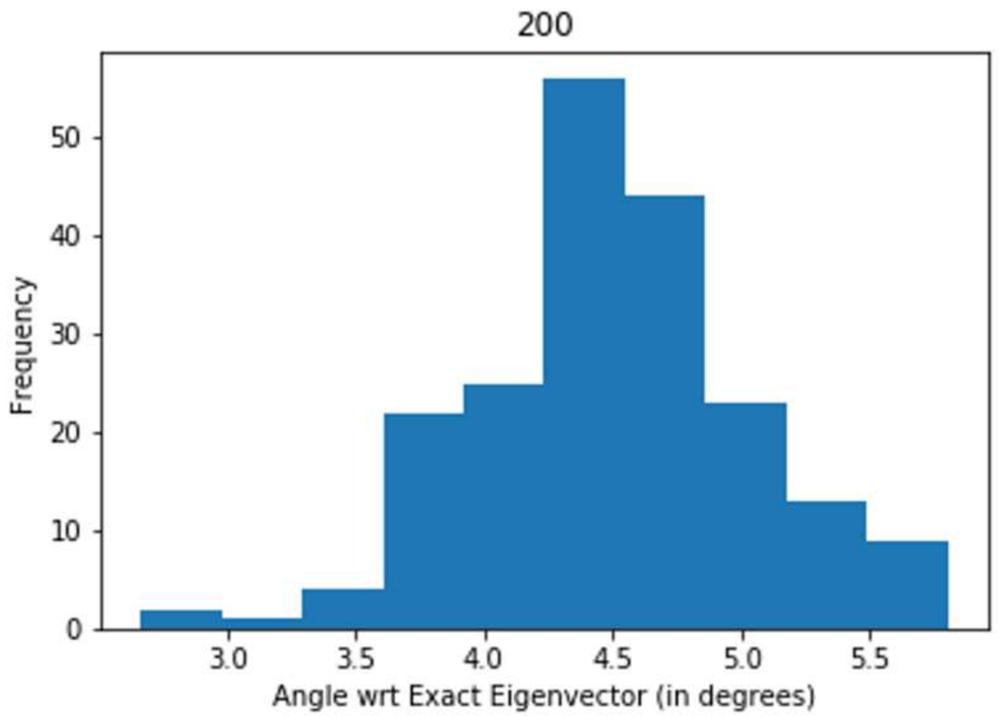
Error (Degrees)



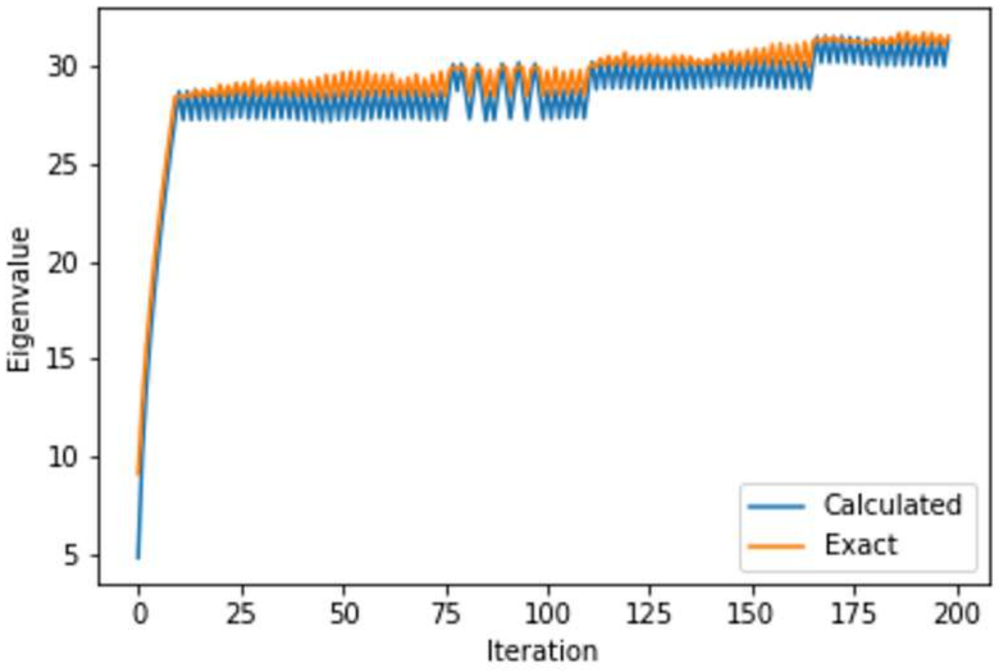
Eigenvalue

Results

Batch Size = 200, 40k operations, Probability of Insertion = 0.5



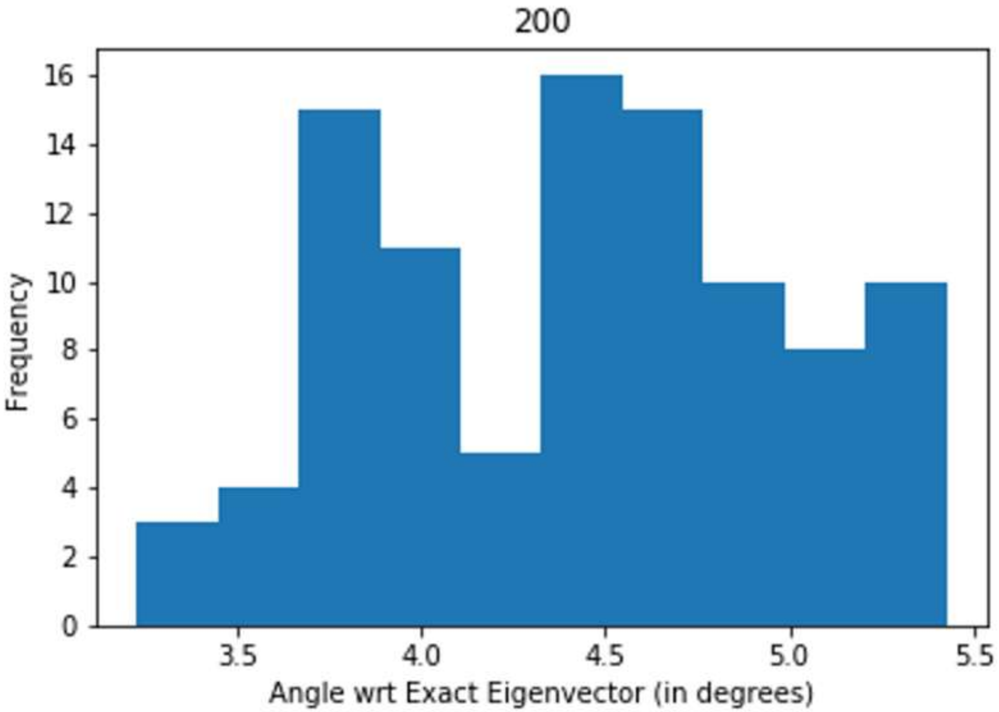
Error (Degrees)



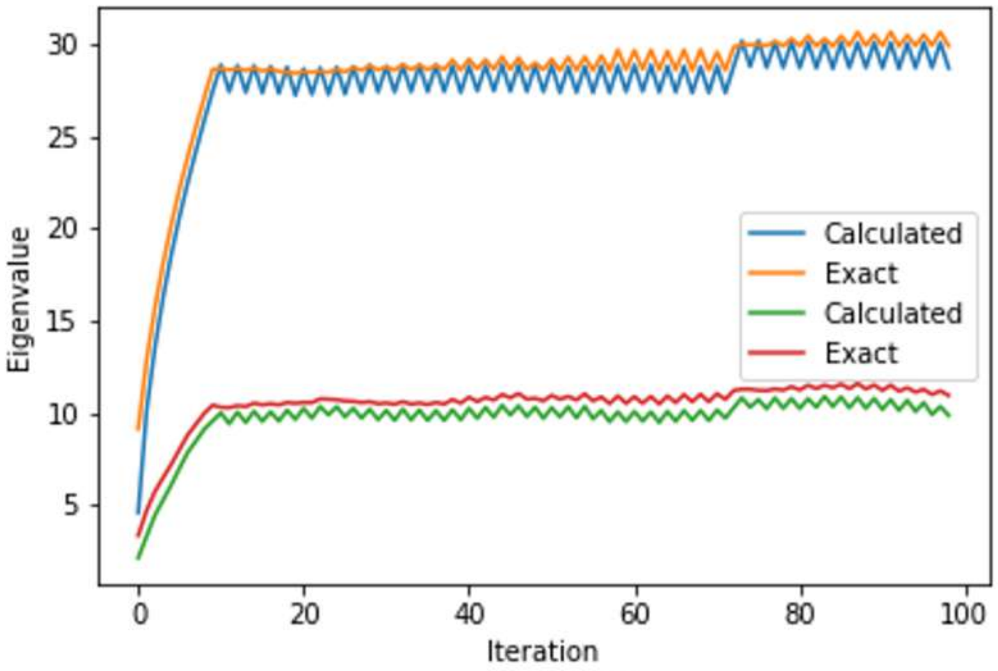
Eigenvalue

Results - Rank 2

Batch Size = 200, 40k operations, Probability of Insertion = 0.5



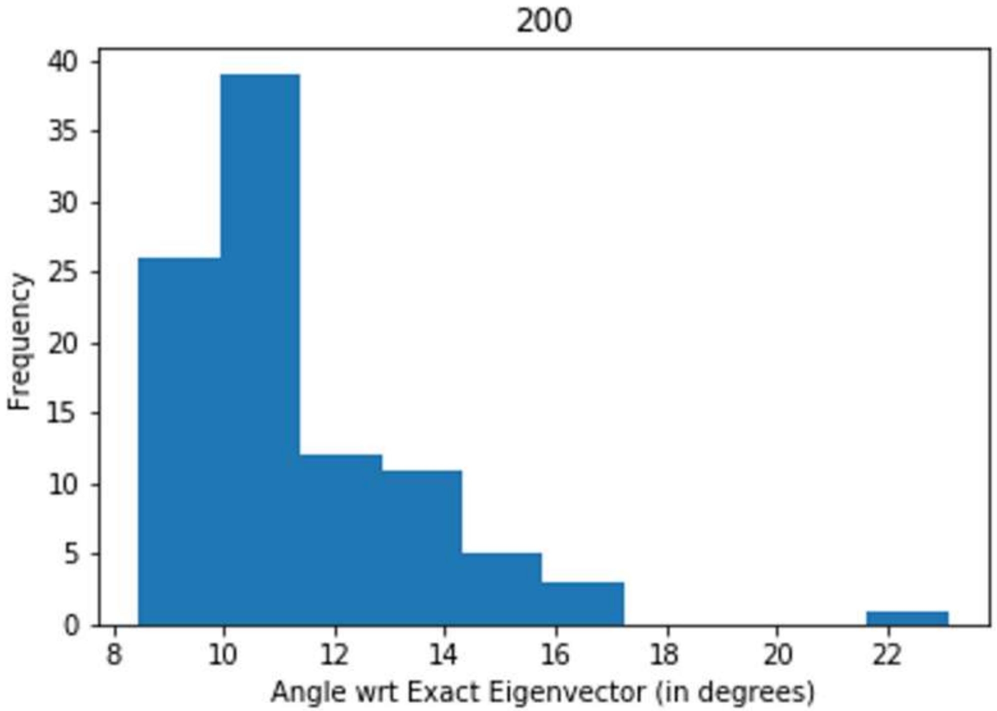
Error (Degrees)



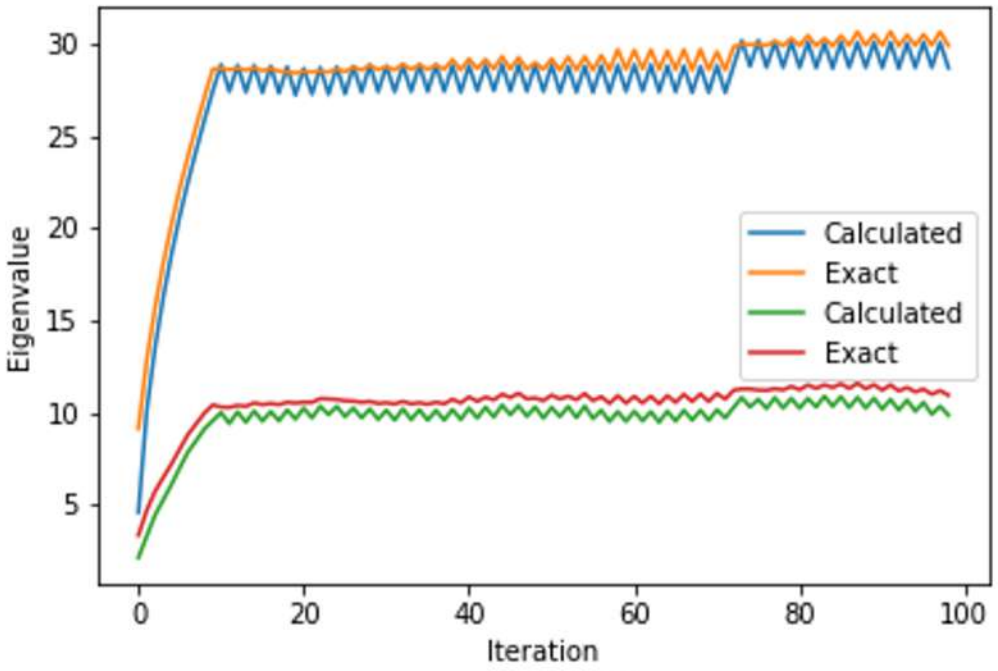
Eigenvalue

Results - Rank 2

Batch Size = 200, 40k operations, Probability of Insertion = 0.5



Error (Degrees)



Eigenvalue

Conclusion

- We propose three splitting rules for hierarchical clustering with guaranteed worst case performance.
- We propose two heuristics for Online Hierarchical Clustering.
- We identify and propose an algorithm for Online Eigenvector Updates.
- We evaluate the model on the *orthogonal dual spiked covariance model* and empirically on the MNIST dataset.

Future Work

- Generalize online eigenvector update results for rank-k case.
- Use them to obtain a bound on Online Hierarchical Clustering.
- Look at Distributed Hierarchical Clustering or Distributed Online Hierarchical Clustering problems.

References

[1] Dasgupta, S. (2015). A cost function for similarity-based hierarchical clustering. *arXiv preprint arXiv:1510.05043*.

[2] Trevisan, L. (2013). Lecture notes on expansion, sparsest cut, and spectral graph theory.

[3] Arora, S., Rao, S., & Vazirani, U. (2009). Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, *56*(2), 5.

[4] Leighton, T., & Rao, S. (1999). Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, *46*(6), 787-832.

[5] Kobren, A., Monath, N., Krishnamurthy, A., & McCallum, A. (2017, August). A hierarchical algorithm for extreme clustering. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 255-264). ACM.

References

[6] Mitliagkas, I., Caramanis, C., & Jain, P. (2013). Memory limited, streaming PCA. In Advances in Neural Information Processing Systems (pp. 2886-2894).

[7] Garber, D., Hazan, E., & Ma, T. (2015, July). Online Learning of Eigenvectors. In ICML (pp. 560-568).

[8] Clarkson, K. L., & Woodruff, D. P. (2009, May). Numerical linear algebra in the streaming model. In Proceedings of the forty-first annual ACM symposium on Theory of computing (pp. 205-214). ACM.

[9] Halko, N., Martinsson, P. G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM review, 53(2), 217-288.