The Complexity of Unsupervised Learning

Santosh Vempala, Georgia Tech

Unsupervised learning

- Data … *(imagine cool images here)*…
- with no labels (or teachers)
- How to
  - Understand it/find patterns in it?
  - Make use of it?
  - What data to collect?
- Interesting extensions:
  - semi-supervised, interactive, lifelong learning

Two high-level approaches

1. Clustering (grouping into similar elements)
    - Choose objective function or other quality measure of a clustering
    - Design algorithm to find (near-)optimal or good clustering(s)
    - Check/hope that this is useful

2. Model fitting
    - Hypothesize model for data
    - Estimate parameters of model
    - Check that parameters were unlikely to appear by chance
    - OR (even better): find best-fit parameters (“agnostic”)

Understanding Unsupervised Learning

- Needs domain knowledge and insight to define the "right" problem
- Theoreticians prefer generalized problems with mathematical appeal
- Some beautiful problems and techniques have emerged. These will be the focus of this talk.
- Many ideas/algorithms in ML are due to neuroscientists (we already saw some)
- There's a lot more to understand!
- How does the brain learn?
  - Much of it is (arguably) unsupervised
  - ("Child, minimize sum-of-squared-distances," is not so common)

Meta-methods

- PCA
- k-means
- EM
- Gradient descent
- …

- Can be "used" on most data sets.
- But how to tell if they are effective? Or if they will converge in a reasonable number of steps?

- Do they work? When? Why?
- (this slide applies to supervised learning as well)

This tutorial

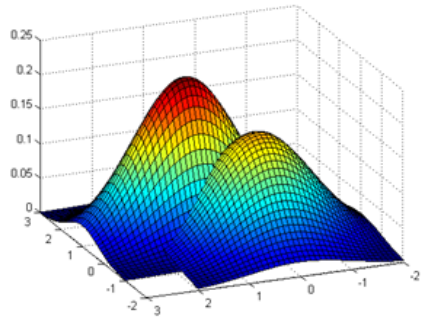
- Mixture Models
- Independent Component Analysis
- Finding Planted Structures (subgraphs, topic models etc.)
- Graph Clustering
- Some relevant (?) snippets from the frontlines

Many other interesting and widely studied models: learning discrete distributions, hidden Markov models, dictionaries, identifying the relevant ("feature") subspace, etc.

Mixture Models

- Classify unlabeled samples from a unknown mixture of distributions; Learn parameters of the mixture.

$$F = w_1 F_1 + w_2 F_2 + \ldots + w_k F_k$$



- E.g., each component F_i is an unknown Gaussi logconcave distribution, etc.
- Goes back to [Pearson 1894]

- Classification needs components to be well-separated.

- Learning Gaussian mixtures does not:

Thm: Gaussian mixtures are uniquely identifiable.

Learning parameters with no assumptions

Thm [2010]. There is a polynomial algorithm to learn a mixture of Gaussians up to any desired accuracy.

[Kalai-Moitra-G.Valiant, Belkin-Sinha, Moitra-G.Valiant]

- Sample Complexity: $n^{f(k)}$
- Lower bound: $2^{k} n$
- Statistical query lower bound: $n^{\Omega(k)}$ [Diakonikolas-Kane-Stewart 2016]

- Could be useful for a small number of components

Techniques

- Random Projection

  [Dasgupta] Project mixture to a low-dimensional subspace to (a) make Gaussians more spherical and (b) preserve pairwise mean separation

  [Kalai] Project mixture to a random 1-dim subspace; learn the parameters of the resulting 1-d mixture; do this for a set of lines to learn the n-dimensional mixture!

  More generally: useful tool to reduce dimensionality while approximately preserving relationships. E.g., efficient learning of robust concepts [Arriaga-V. 1999]

- Method of Moments

  [Pearson] Finite number of moments suffice for 1-dim Gaussians

  [Kalai-Moitra-G.Valiant] 6 moments suffice

Clustering assuming separation

- A1. Pairwise separation between means. (Clustering)

  Separation: $k^{1/4}(\sigma_i+\sigma_j)$ where σ_i^2 = max variance of component i.

  [Dasgupta, D-Schulman, Arora-Kannan, V-Wang, K.-Salmasian-V, Achlioptas-McSherry]

- A2. Each mean is separated from the span of the previous means. (Clustering)

  Separation: $poly(k)$. standard deviation along separating direction

  [Brubaker-V.]

- A3. Matrix of means has a bounded smallest singular value. This implies that each mean is separated from the span of the rest. (Learning)

  Spherical Gaussians: complexity grows as 1/poly(separation).

  [Hsu-Kakade, Goyal-V.-Xiao]

Techniques

PCA:

- Use PCA once

  [V.-Wang]

- Use PCA twice

  [Hsu-Kakade]

- Reweight and use PCA

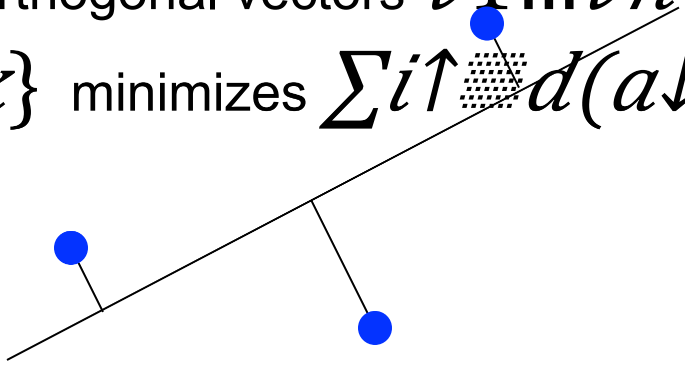
  [Brubaker-V., Goyal-V.-Xiao]

Technique: Principal Component Analysis

Points $a_1 \ldots a_m \ in\ R^n$.

First principal component: line v that minimizes the sum of squared distances to it, $\sum_i^{} d(a_i, v)^2$.

Principal Components are orthogonal vectors $v_1 \ldots v_n$ s.t.
$V_k = span\{v_1 \ldots v_k\}$ minimizes $\sum_i^{} d(a_i, V)^2$
among all k-dim subspaces.

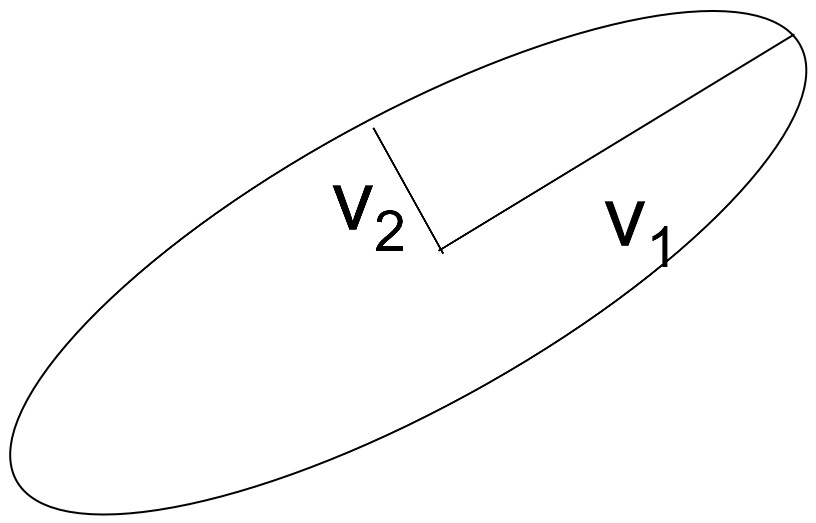


$V_k = V_{k-1}$ + best vector orthogonal to V_{k-1}

Computed via the Singular Value Decomposition.

PCA example

- For a Gaussian, the principal components are the axes of the ellipsoidal level sets.



Why PCA?

- Reduces computation/space.

(Random projection, Random sampling also reduce space)

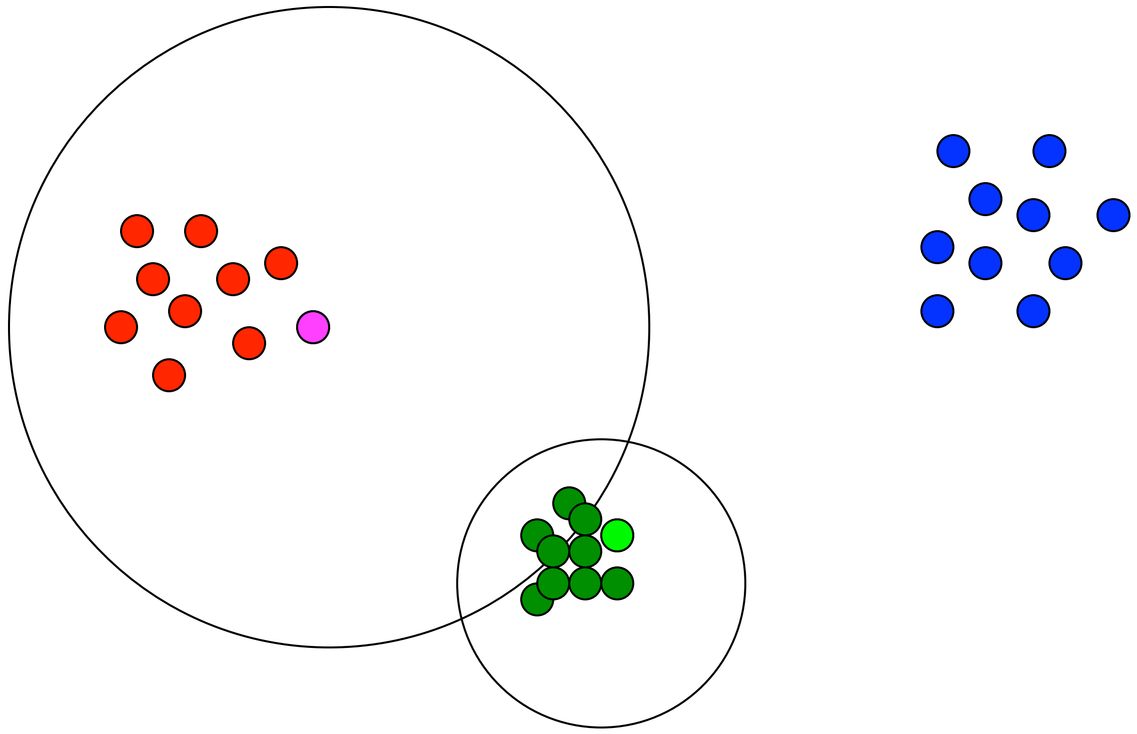
- (sometimes) Reveals interesting structure.

Technique: Principal Component Analysis

- "PCA is a mathematical tool for finding directions in which a distribution is stretched out."
- Discussed by Euler in work on inertia of rigid bodies (1730).
- Principal axes identified as eigenvectors by Lagrange.
- Power method for finding eigenvectors published in 1929, before computers.
- Ubiquitous in practice today:
  - Bioinformatics, Econometrics, Data mining, Computer vision, ...
- Hippocampus uses it!

Distance-based classification

Points from the same component should be closer to each other than those from different components.



Unfortunately, the separation required grows with the ambient dimension.

Algorithm

- Project to span of top k principal components of the data
- Apply distance-based classification in this subspace

Clustering spherical Gaussians [VW02]

- Distance-based clustering:
  - needs separation that grows as $n^{1/4}$
- PCA, then cluster:
  - Separation required grows as $k^{1/4}$:

    $$|\mu_i - \mu_j| > k^{1/4} (\sigma_i + \sigma_j) \log \ldots$$

  - Projection to span of means preserves inter-mean distance and shrinks component Gaussians.
  - Span(means) = PCA subspace of dim k

PCA for spherical Gaussians

- Best line for 1 Gaussian?

  - Line through the mean

- Best k-subspace for 1 Gaussian?

  - Any k-subspace through the mean

- Best k-subspace for k Gaussians?

  - The k-subspace through all k means!

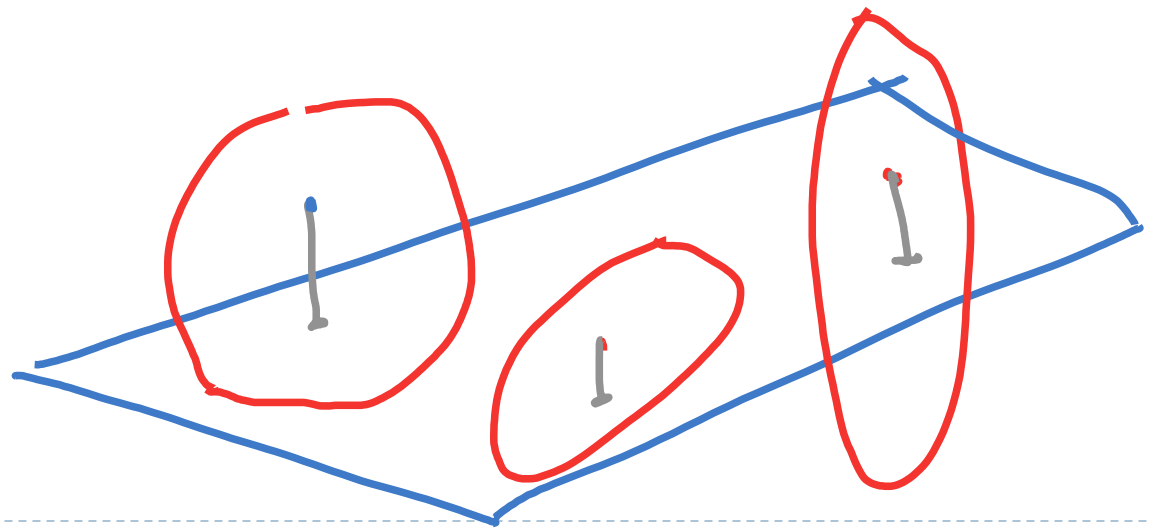
Mixtures of Logconcave Distributions

Thm. PCA subspace is "close" to span of means.

- Separation required for classification:

$|\mu_i - \mu_j| > poly(k)(\sigma_{i,max} + \sigma_{j,max})\log\ldots$

where $\sigma_{i,max}^2$ is the *maximum* directional variance



K-means and PCA

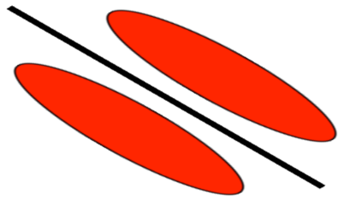
1. Apply PCA to embed in a low-dimensional subspace
2. Run favorite clustering algorithm (e.g., k-means iteration)

Thm. [Kannan-Kumar] Converges efficiently for k-means iteration under a natural pairwise separation assumption.

- (important to apply PCA before running k-means!)

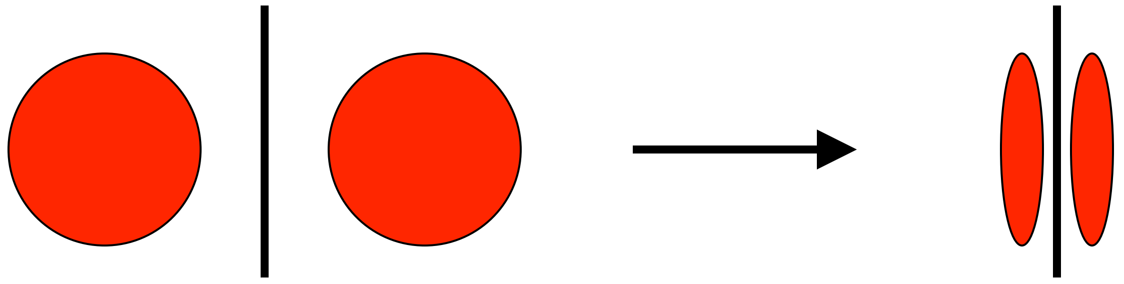
Limits of PCA

- Can fail for a mixture of 2 arbitrary Gaussians

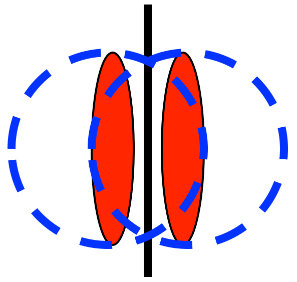


- Algorithm is not affine-invariant or noise-tolerant.
- Any instance can be made bad by an affine transformation or a few "bad" points.

Parallel pancakes

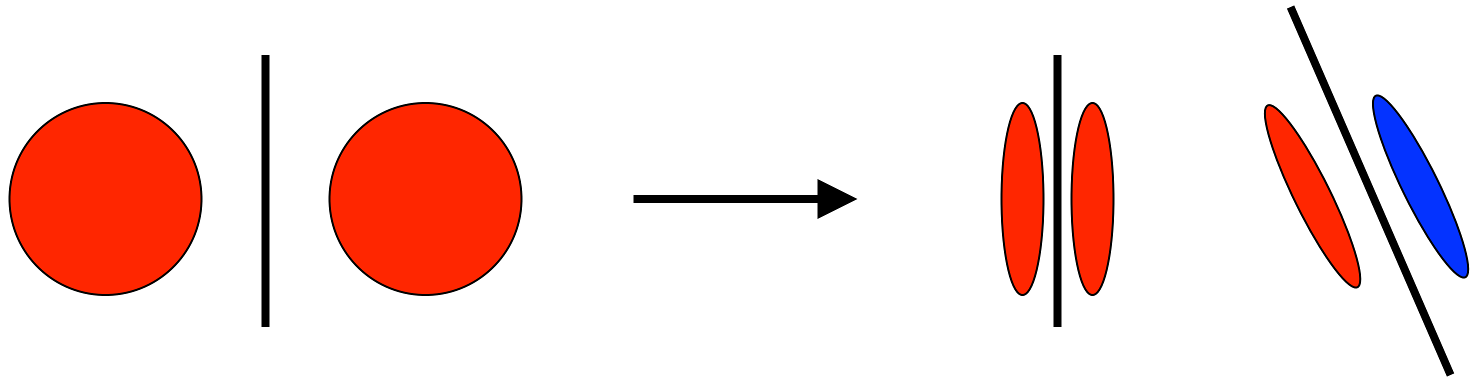


Still separable, but algorithm does not work.



Classifying Arbitrary Gaussian Mixtures

- Component Gaussians must be probabilistically separated for classification to be possible
- Q. Is this enough?
- Probabilistic separation is affine invariant:

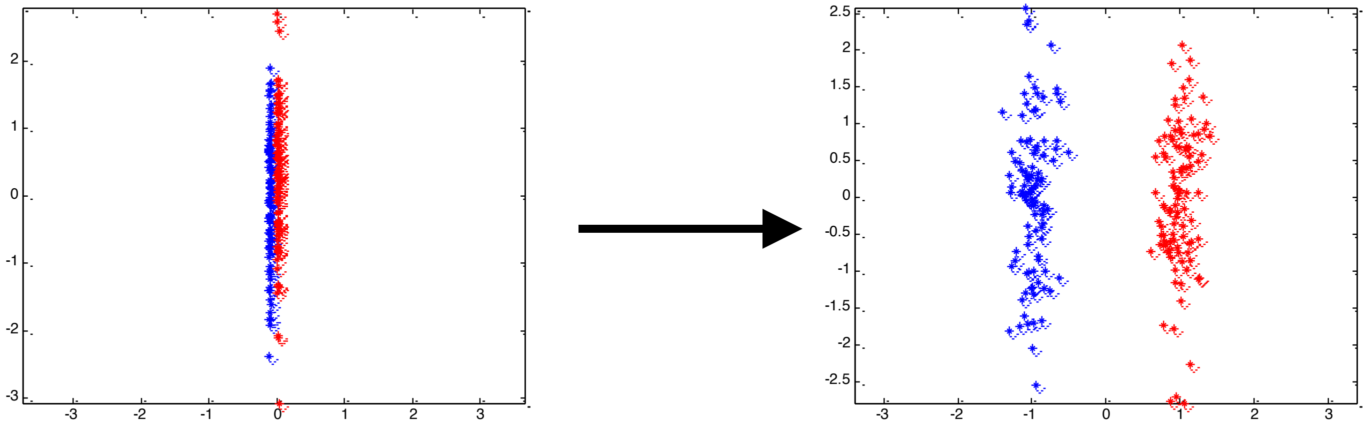


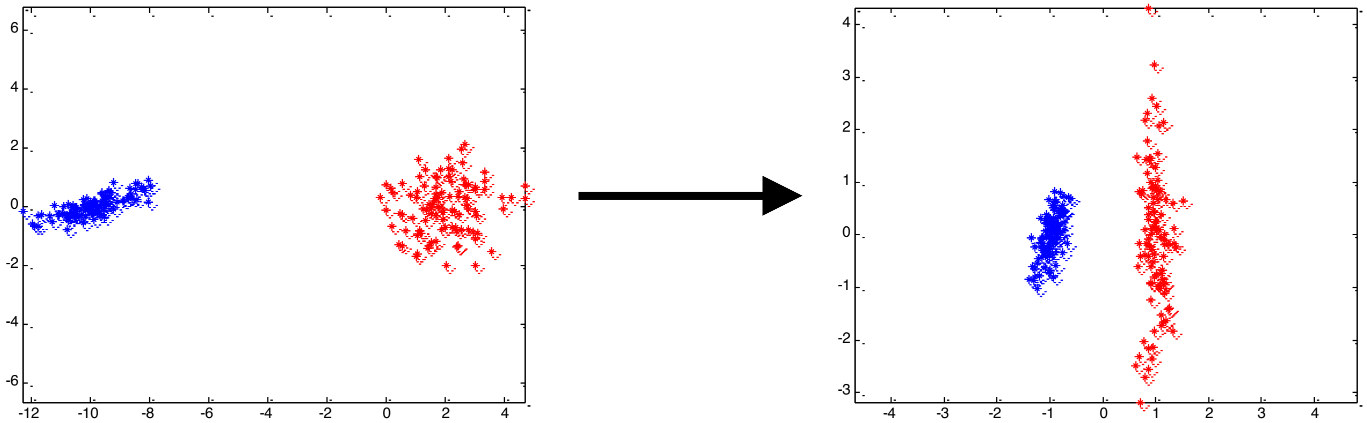
- PCA is not affine-invariant!

Algorithm: Isotropic PCA

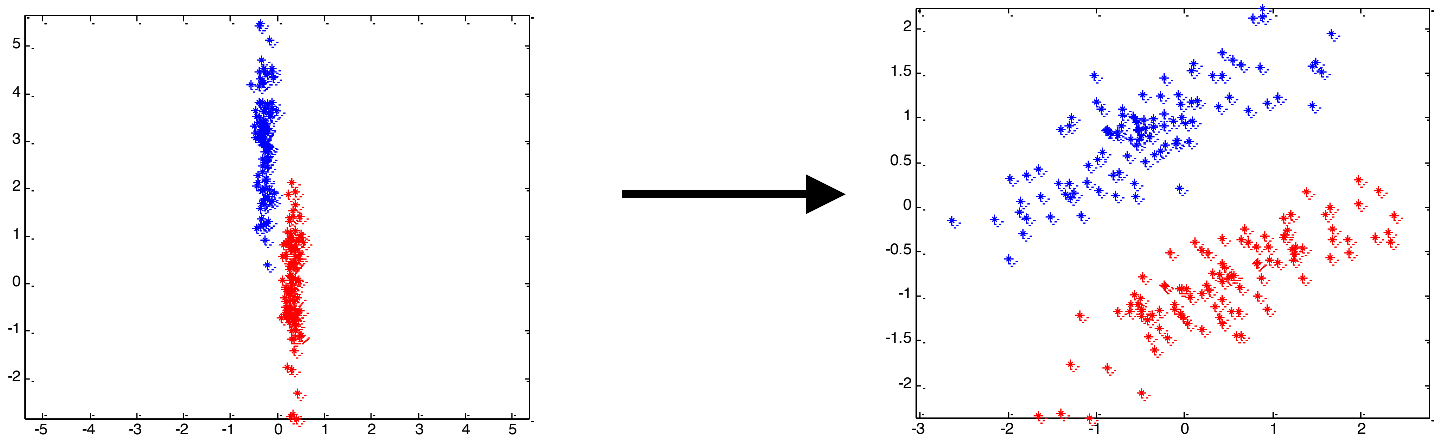
1. Apply affine transformation to make distribution isotropic, i.e., identity covariance.
2. Reweight points (using a spherical Gaussian).
3. If mean shifts, partition along this direction.
   Else, partition along top principal component.
4. Recurse.

Isotropy





Isotropy

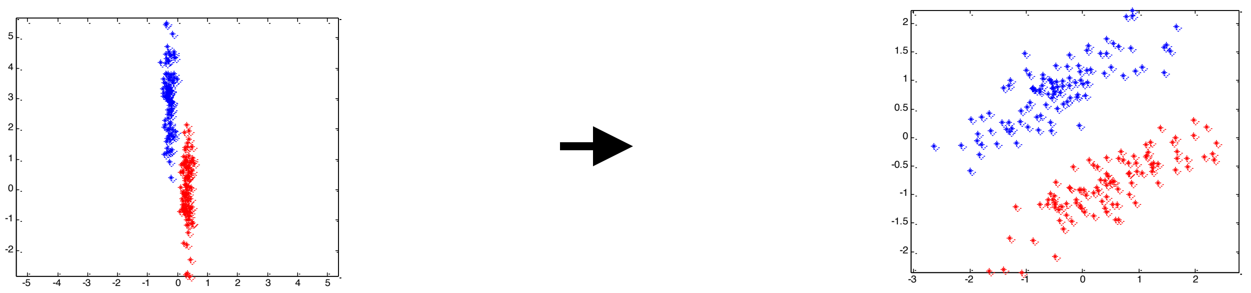


- Turns every well-separated mixture into almost parallel pancakes, separable along the intermean direction.

• But, PCA can no longer help!

Unraveling Gaussian Mixtures

- Isotropy pulls apart the components

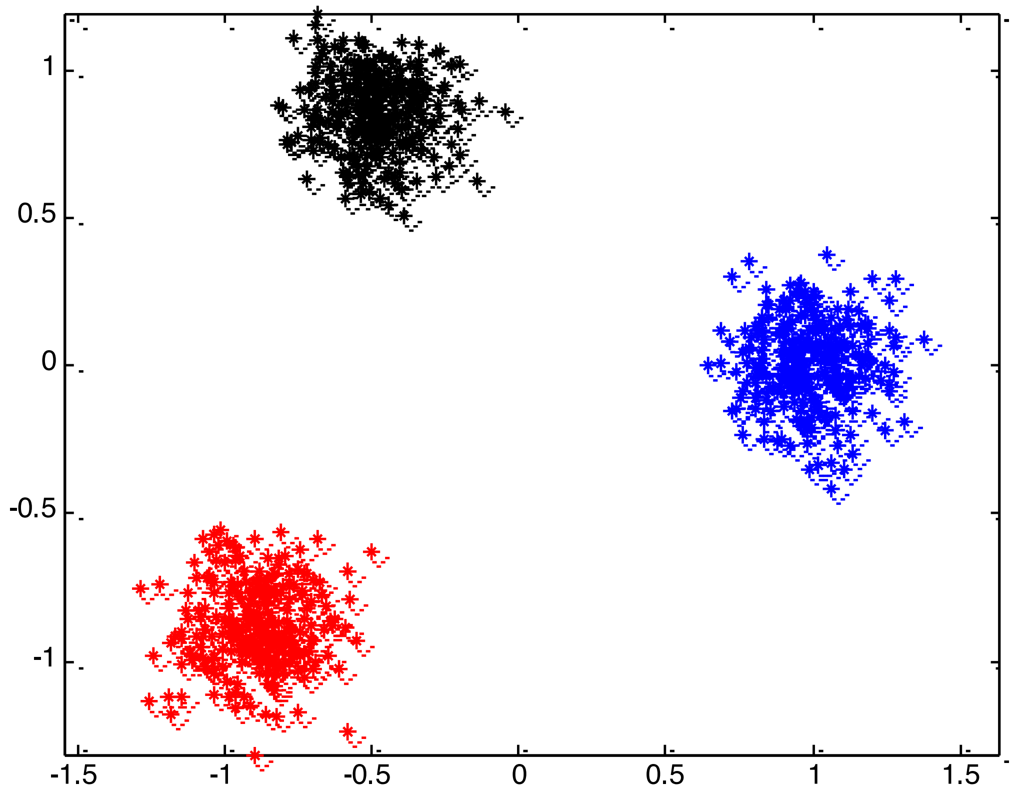


- If some component is heavier, then reweighted mean shifts along a separating direction
- If not, reweighted principal component is along a separating direction

Affine-invariant clustering

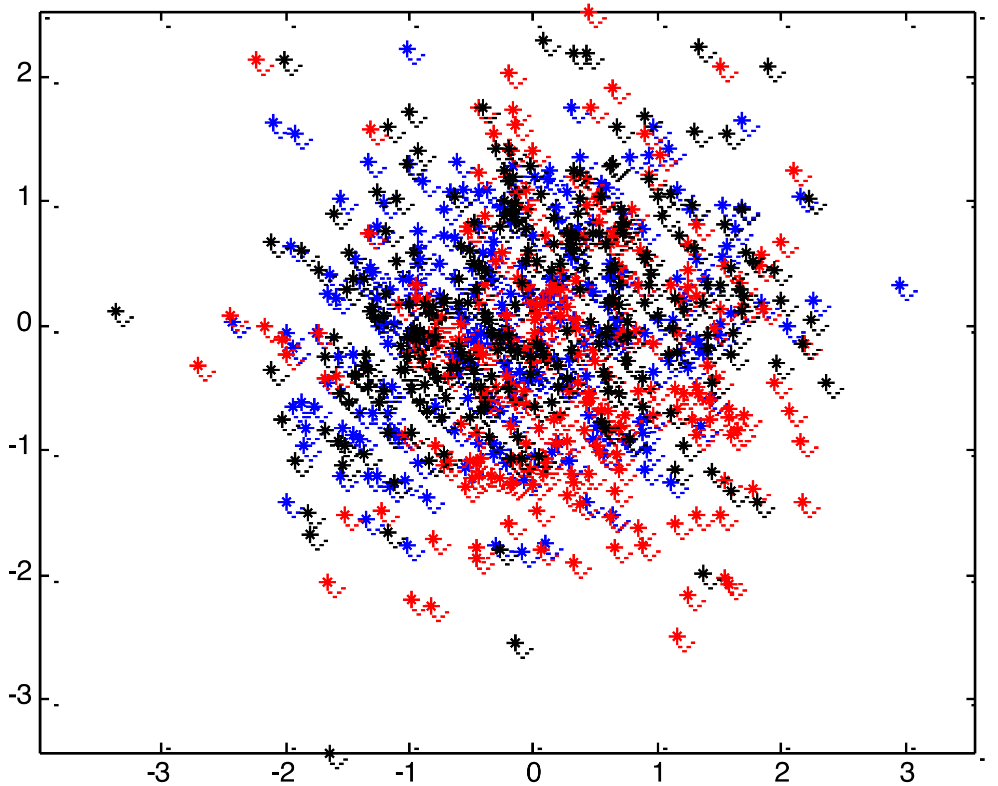
- Thm.[Brubaker-V.08] The algorithm correctly classifies samples from a mixture of k arbitrary Gaussians if each one is separated from the span of the rest. (More generally, if the overlap is small as measured by the Fisher criterion).
- Q: Extend Isotropic PCA to more general mixtures

Original Data

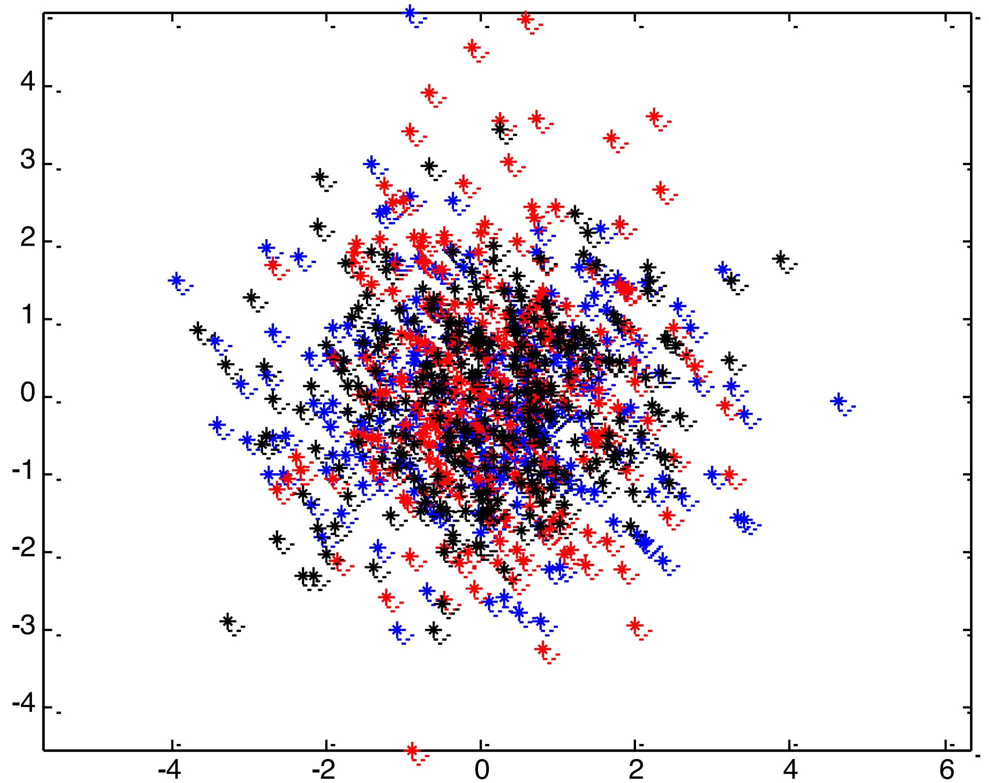


- 40 dimensions, 15000 samples (subsampled for visualization)

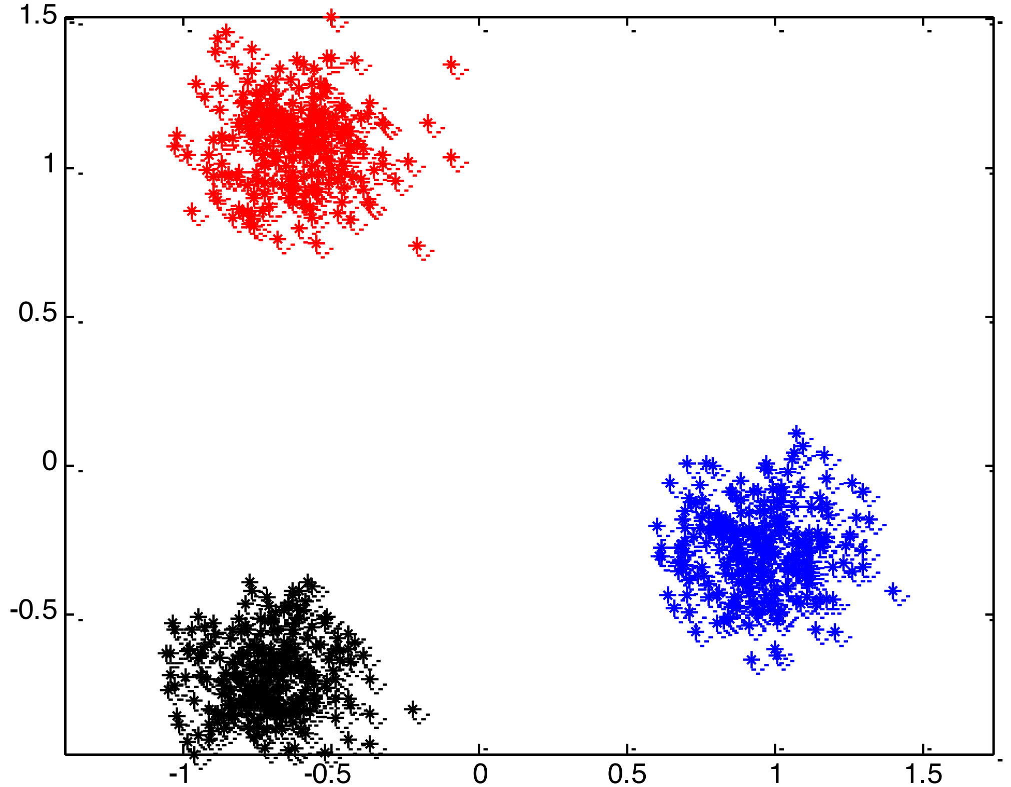
Random Projection



PCA



Isotropic PCA



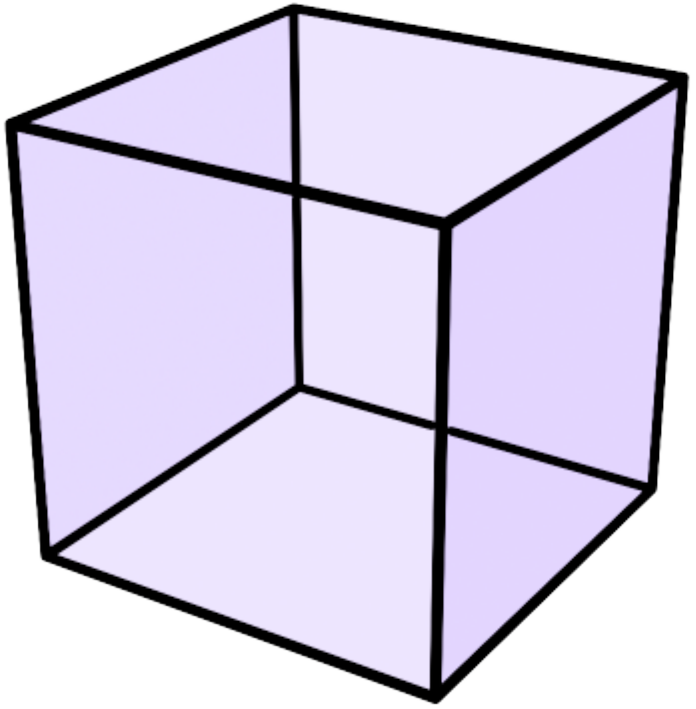
Learning noisy distributions/mixtures

- Data is mostly from a nice distribution (e.g., Gaussian or mixture of Gaussians) but some fraction is not.
- Can we learn (the parameters of) the nice distribution?
- Recent work: Yes, in many interesting cases!
- Agnostic learning is of interest across learning theory.

Outline

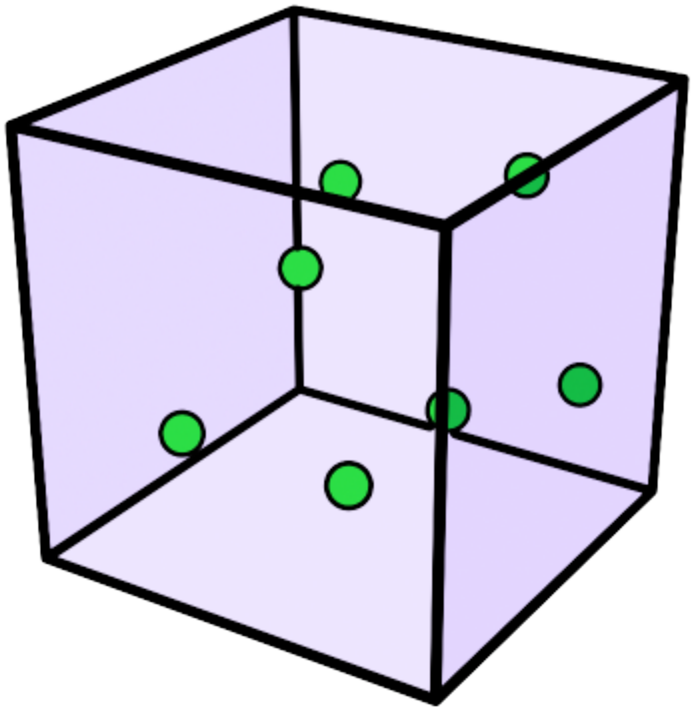
- Mixture Models
- Independent Component Analysis
- Finding Planted Structures (subgraphs, topic models etc.)
- Graph clustering
- Some relevant (?) snippets from the frontlines

Independent Component Analysis (ICA)



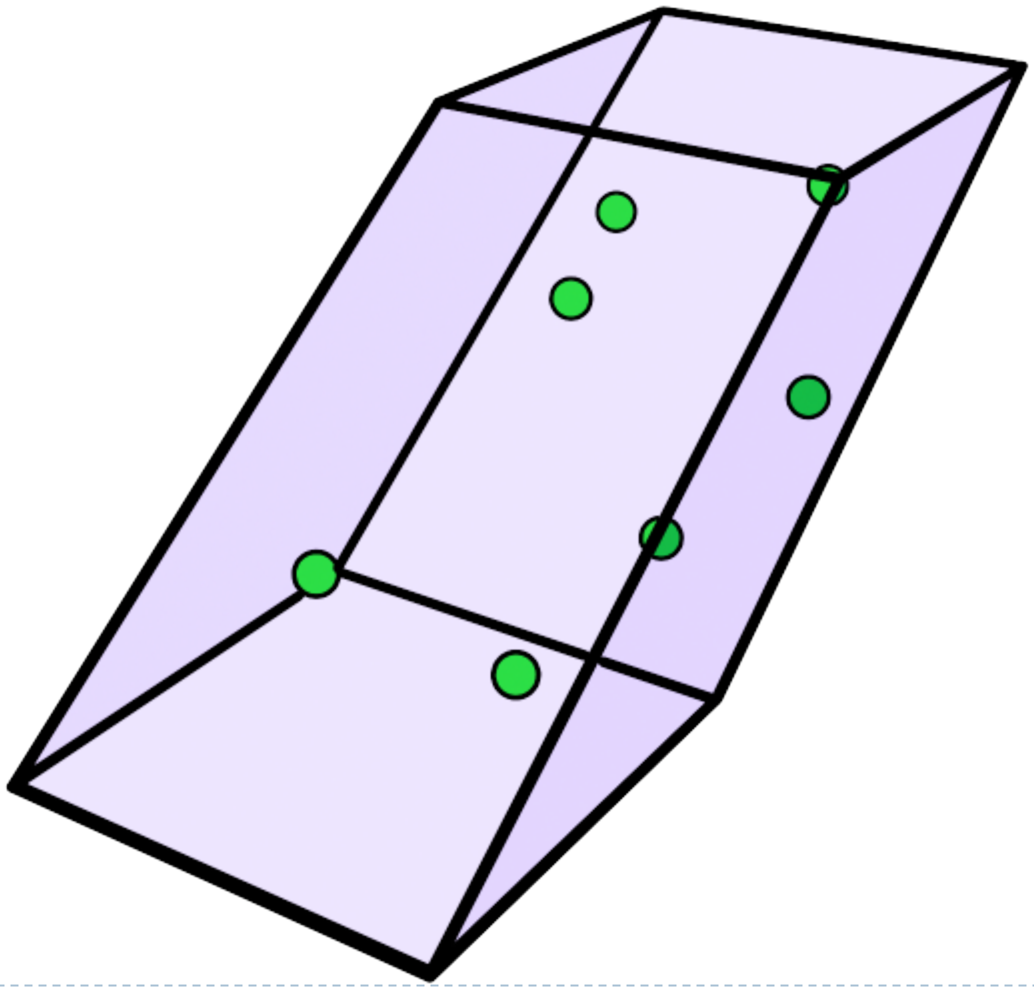
ICA model

- Start with a product distribution



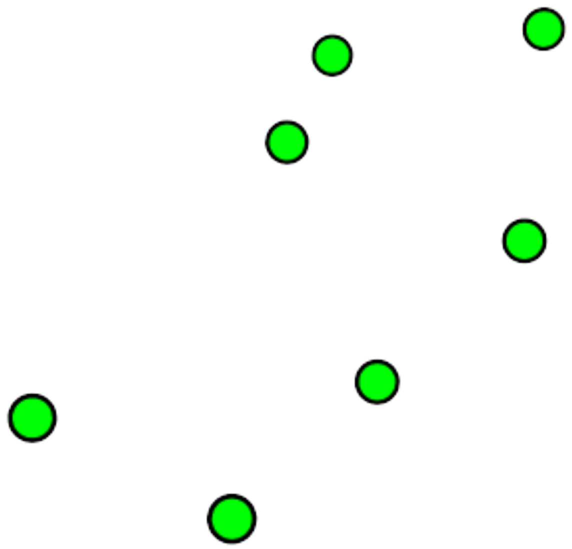
ICA model

- Apply a linear transformation A



ICA model

- Observed sample

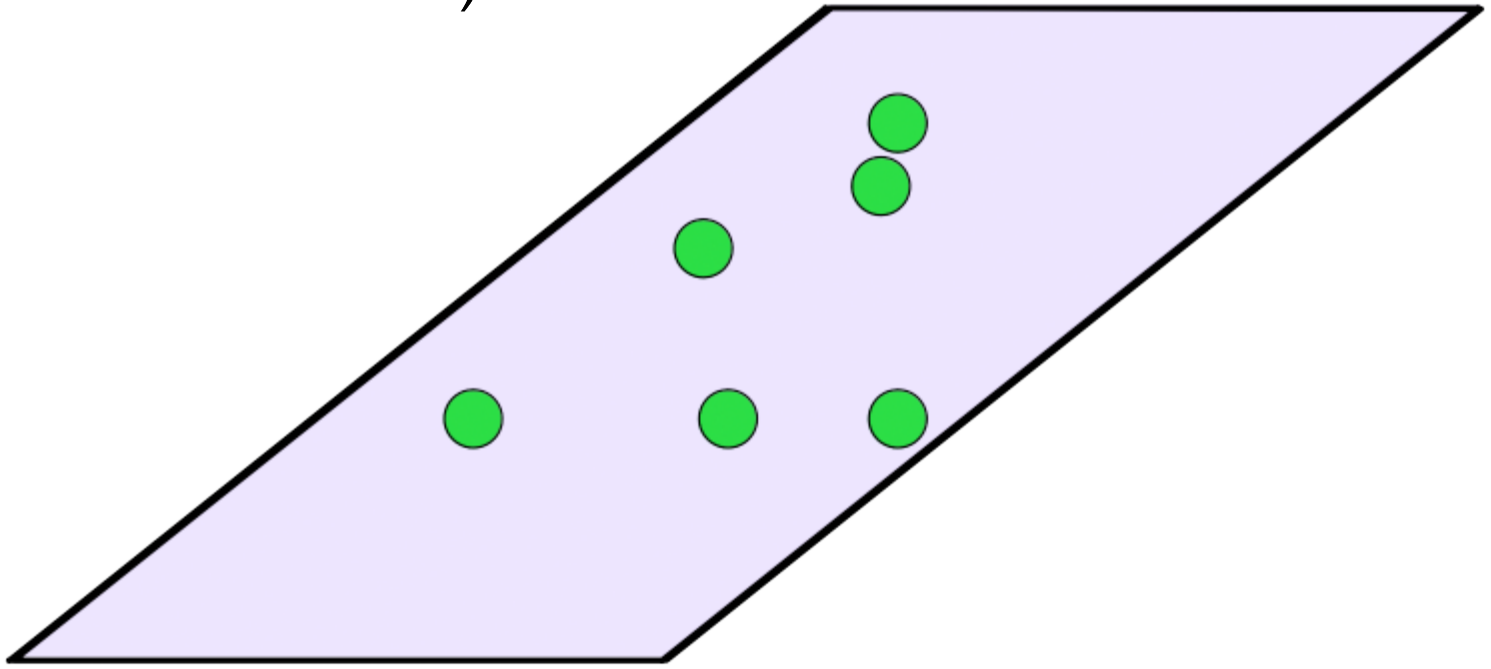


- Problem: Find the hidden transformation A

ICA model

Matrix A might include a projection

(underdetermined ICA)



Independent Component Analysis

- Model: Data is a linear transformation of an unknown product distribution:

$$s \in R^{m},\ A \in R^{n \times m} \quad \text{data} \quad x = As$$

Thm. A is unique up to signs of columns if at most one component s_i is Gaussian

- Problem: Learn A by observing samples x.
- Used in ML, signal processing, neuroscience for 25+ years.
- Many attractive heuristics.

Status: ICA

- Thm [Goyal-V.-Xiao13]. If columns of A satisfy a weak linear independence condition, and component distributions are Δ-far from Gaussian, then A can be estimated with complexity $poly(m,\Delta, 1/\epsilon)$.

- Generalized linear independence: smallest d for which the tensors $\otimes^{d} A_{i}$ are linearly independent.

- Earlier work for d=1 and special classes of distributions [FJK,NR,AGMS,AGR]
- Technique: Robust *tensor* decomposition.

- Thm[VX14]. If columns of A are linearly independent and $k \le 4$, then sample complexity = $O(n)$ and time complexity = O(SVD)

- Both theorems work with Gaussian noise: $x = As + \eta$
- Recent work: agnostic ICA.

Techniques

- PCA
- finds local optima of second moments, i.e., $\max_{u \in R^n} E((u^T x)^2)$

- Local optima of 4th moment. [Frieze-Jerrum-Kannan96]
  - Works if each component differs from Gaussian in the 4th moment, e.g., uniform over a cube.
  - Local optima via local search or a power iteration. [Nguyen-Regev]
  - Tensor view: After making the samples isotropic,

$$E(x \otimes x \otimes x \otimes x) = \sum_j (E(s_i^4) - 3) A_i \otimes A_i \otimes A_i \otimes A_i$$

- Fourier PCA [GVX13].
- Reweight x with Fourier weight $e^{iu^T x}$ for random unit vector u; then apply PCA; more generally, a robust tensor decomposition.

- Recursive FPCA [VX14].
- Partition using largest eigenvalue gap; recurse.

Outline

- Mixture Models
- Independent Component Analysis
- Finding Planted Structures (subgraphs, topic models etc.)
- Graph clustering
- Some relevant (?) snippets from the frontlines

Planted structures

- Planted clique/dense subgraph: Start with a random graph. Add a clique of size $k \gg 2\log n$ on some subset of k vertices.

  Find planted clique.

- Planted partition: Fix a partition of vertices of a graph. Pick random edges with different probabilities within parts and across parts.

  Recover planted partition.

- Planted assignment: Fix an assignment σ on Boolean variables. Generate a random formulas by picking clauses from a distribution that depends on σ.

  Recover planted assignment.

- Planted vector/subspace: Generate random points by adding a random vector from a fixed subspace to random (Gaussian) noise in full space.

  Recover planted vector subspace

Status: Planted Cliques

- Upper bounds: $n^{O(\log n)}$ for any $k > (2+\epsilon)\log n$
- Polynomial time for $k > c\sqrt{n}$

  [Alon-Krivelevich-Sudakov98]
- Lower bound: For $\epsilon > 0$, $k = n^{0.5-\epsilon}$, any statistical algorithm has complexity $n^{\Omega(\log n)}$

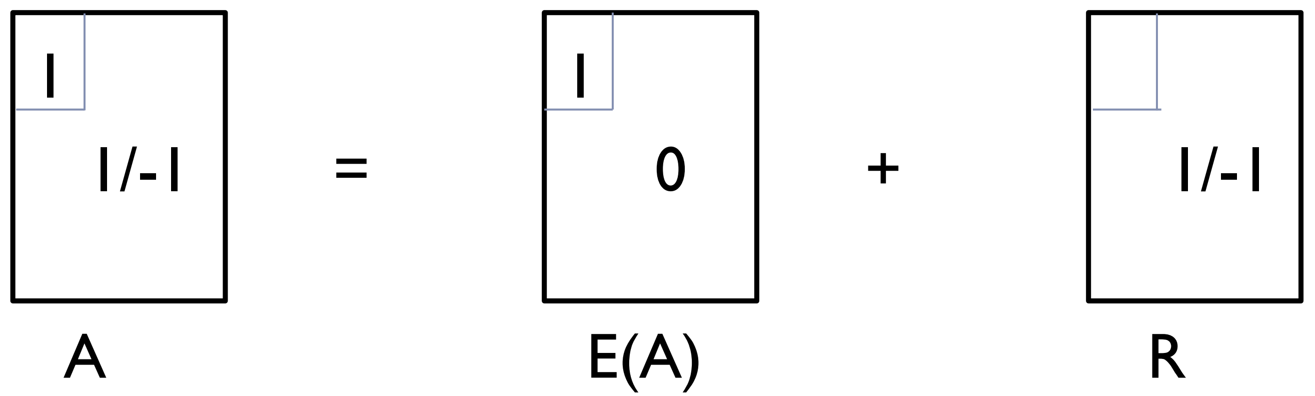
  [Grigorescu-Reyzin-Feldman-V.-Xiao13]
- (formally, this is for bipartite planted cliques, for which the same upper bounds apply)

- Q: Is there a polytime algorithm for $k \ll \sqrt{n}$?

Techniques

- Combinatorial:
- Remove lowest degree vertex iteratively [Feige]

- Spectral:
- Take highest components of principal component [AKS98]



Thm [Furedi-Komlos]. $|R|_2 \leq (2+o(1))\sqrt{n}$.

Status: Planted k-SAT/k-CSP

- Upper bound:
  Information theoretically, $O(n\log n)$ clauses suffice.
  Algorithmically, $n^{k/2}\log n$ clauses suffice
  [Bogdanov-Qiao-Applebaum, Feldman-Perkins-V.14]
  in time linear in number of clauses [FPV14].

- Bound is $n^{r/2}$ for (r-1)-wise independent clause distributions.

- Lower bound:
  $(n/\log n)^{r/2}$ clauses for statistical algorithms.[FPV14]

- OP: Find efficient (nonstatistical) algorithm for planted SAT.

Statistical Algorithms

- Only access to the input distribution: compute arbitrary functions on random samples OR estimate their expectations to within a given tolerance.

- For any $f{:}X\rightarrow[0,1]$, STAT(τ) outputs $E(f(x))\pm\tau$. [Kearns]
- For any $f{:}X\rightarrow\{0,1\}$, 1-STAT outputs f(x) for a random x.
- VSTAT(t): outputs $E\downarrow D\ [f(x)]$ to within the standard deviation of t random samples.

- Complexity of algorithm = number of calls to oracle.

Can statistical algorithms detect planted structures?

- Well-known algorithms can be implemented statistically:
  - Small/large degree
  - Local search
  - PCA (power iteration)
  - Markov Chain Monte Carlo / simulated annealing
  - Gradient descent

$$\nabla_x E_u [f(x,u)] = E_u [\nabla_x f(x,u)]$$

  - Linear programs, conic programs, stochastic optimization

- With one notable exception: Gaussian Elimination over a finite field

Detecting planted solutions

- Many interesting problems (e.g., sparse topics/ dictionaries)
- Potential for novel algorithms
- New computational lower bounds
- Open problems in both directions!

Outline

- Mixture Models
- Independent Component Analysis
- Finding Planted Structures (subgraphs, topic models etc.)
- Graph clustering
- Some relevant (?) snippets from the frontlines

Clustering from pairwise similarities

Input:

A set of objects and a (possibly implicit) function on pairs of objects.

Output:

1. A flat clustering, i.e., a partition of the set
2. A hierarchical clustering
3. A weighted list of features for each cluster

Typical approaches

- Optimize a "natural" objective function
- E.g., k-means, min-sum, min-diameter etc.

- Axiomatic: derive from assumptions on valid solutions

- Using EM/local search OR
- a provable approximation algorithm (less common)

- Issues: quality, efficiency, validity.
- Many natural functions are NP-hard to optimize

Divide and Merge

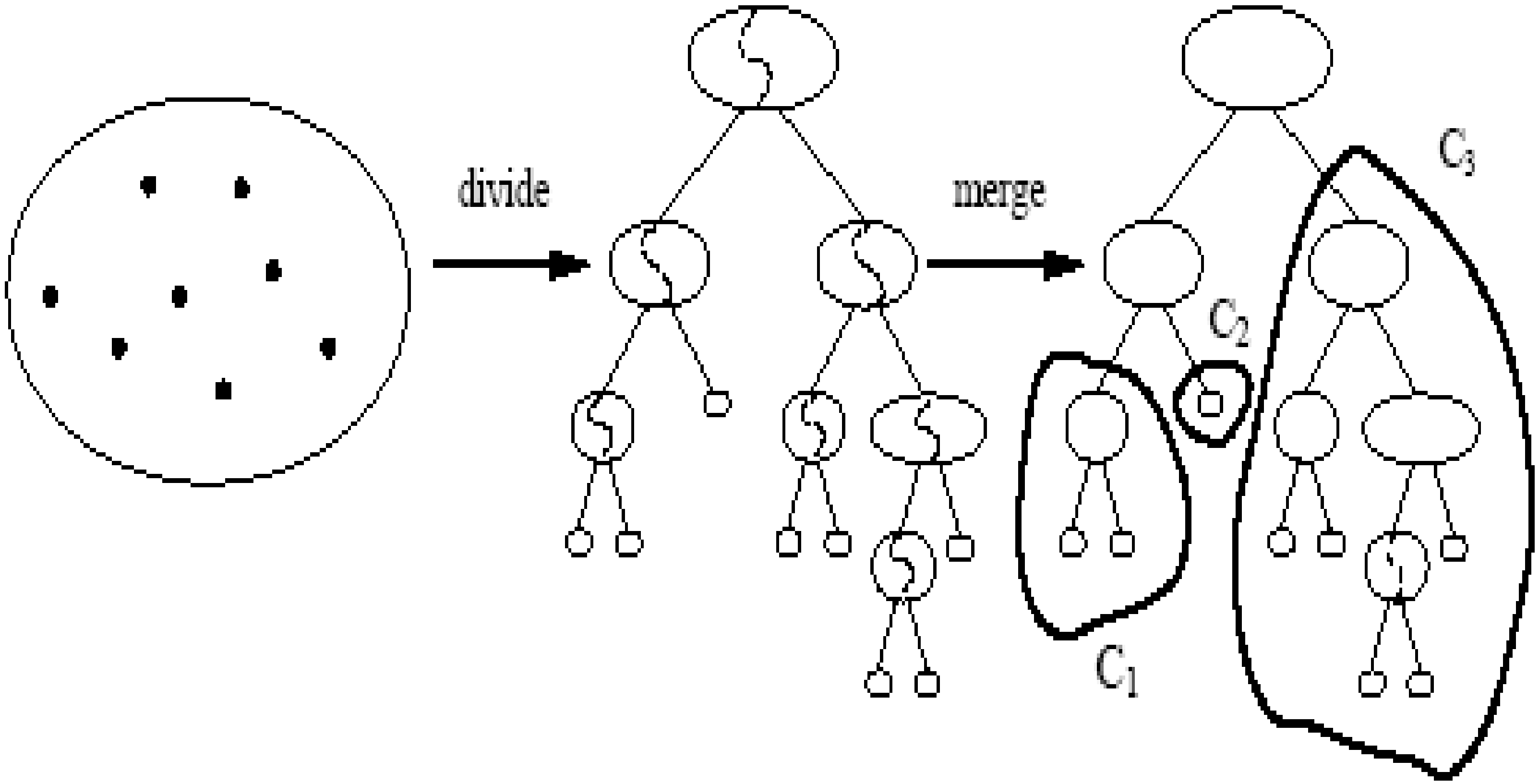
- Recursively partition the graph induced by the pairwise function to obtain a tree

- Find an "optimal" tree-respecting clustering

Rationale: Easier to optimize over trees;

k-means, k-median, correlation clustering all solvable quickly with dynamic programming

Divide and Merge

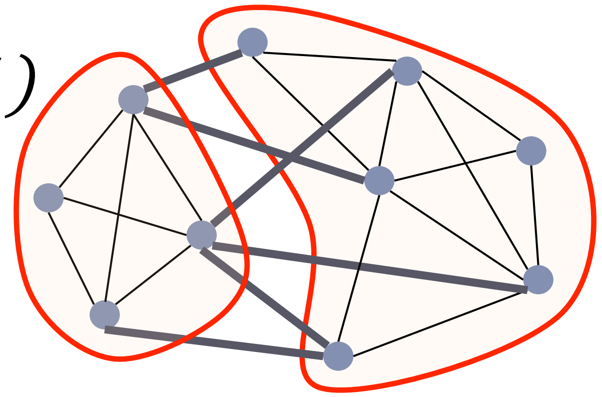


How to cut?

Min cut? (in weighted similarity graph)

Min expansion/conductance cut [Jerrum-Sinclair]

$$\phi(S) = w(S, \bar{S}) / \min w(S), w(\bar{S})$$



Sparsest cut

Normalized cut [Shi-Malik 2000]

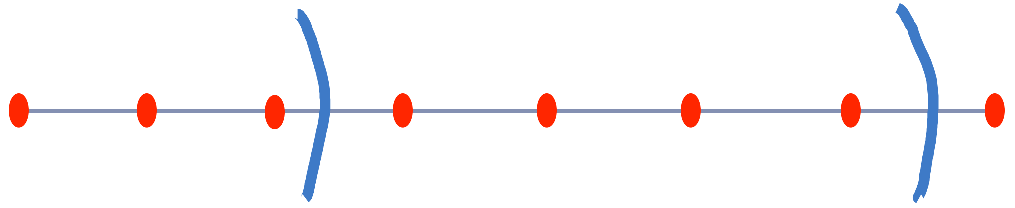
Many applications: analysis of Markov chains, pseudorandom generators, error-correcting codes...

How to cut?

- Min conductance/expansion is NP-hard to compute.

- Leighton-Rao, Linear program: $O(\log n)$
- Arora-Rao-U.Vazirani, Semidefinite program: $O(\sqrt{\log n})$

- Fiedler cut: Sort vertices according to component in 2nd eigenvector of normalized similarity matrix; take best of $n-1$ cuts. $O(\sqrt{OPT})$



Worst-case guarantees

Assume

- we can efficiently find a cut of conductance $a \cdot OPT^{\nu}$
- There exists an (α,ϵ)-clustering where each cluster has conductance at least α and at most ϵ fraction of similarity lies between clusters.

Thm [Kannan-V.-Vetta '00].

If there exists an (α,ϵ)-clustering, then the recursive partitioning algorithm finds a clustering of quality $\left(\alpha^{1/\nu}/a \log n, a\epsilon^{\nu} \log n\right)$

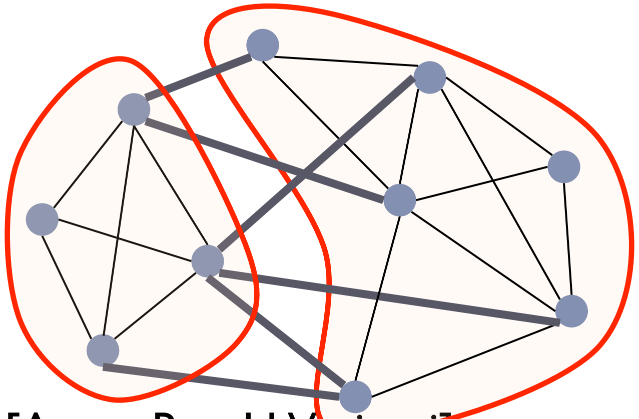
Cor. Recursive spectral partitioning gives $\left(\alpha^{2}/2\log n, 2\sqrt{\epsilon} \log n\right)$

Graph expansion

- $G=(V,E)$, edge weights w
- $S\subset V$

$$\phi(S)=w(S,S\ )/\min w(S),\ \ w(S\ )$$

- $\phi(G)=\min_{\top}S\ \phi(S)$



- NP-hard to compute exactly
- Admits polytime $O(\sqrt{\log n}\ \ )$ approximation [Arora-Rao-U.Vazirani]
- Improving on earlier $O(\log n)$ approximation

  [Leighton-Rao'88, Linial-London-Rabinovich,Aumann-Rabani]

Graph eigenvalues

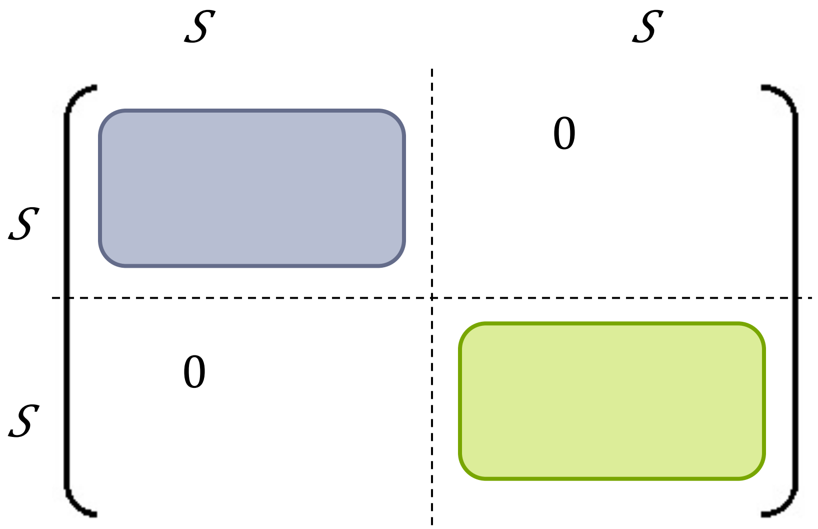
- $A_G = D^{-1/2} A D^{-1/2}$ with $D_{ii} = d_i = \sum_j w_{ij}$
- $A_G = 1/d\ A$ for d-regular graphs
- $L_G = I - A_G$ is positive semidefinite
- $\lambda_1 (L_G) = 0;\ L_G D^{1/2} \mathbf{1} = 0.$

$$\lambda_2 (L_G) = \min_{x \in R^n,\ x \perp D^{1/2} \mathbf{1}} x^T L_G x / x^T x = \min_{x \in R^n,\ x \cdot d = 0} \sum_{ij \in E} w_{ij} (x_i - x_j)^2 \Big/ \sum_i d_i x_i^2 \geq 0$$

Perron-Frobenius

- $\lambda_2 = 0$ if and only if graph is disconnected.

- If $\lambda_2 \approx 0$, then is graph close to disconnected ?



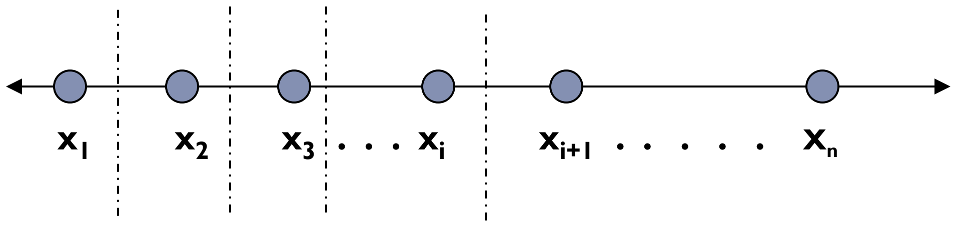
Cheeger's Algorithm

[Cheeger; Alon-Milman] $1/2\ \lambda_2 \leq \phi(G) \leq \sqrt{2\lambda_2}$

x: eigenvector of L_G for λ_2

1. Sort x: $x_1 \leq x_2 \leq \ldots \leq x_n$
2. Consider subsets $S_i = \{x_1, \ldots, x_i\}$
3. Take S: $\operatorname{argmin} \phi(S_i)$

2nd eigenvector of L_G



$\min_i \phi(S_i) \leq \sqrt{2\lambda_2}$, proof via Cauchy-Schwarz

Gives method to certify constant expansion

Cheeger's inequality

[Cheeger; Alon-Milman]

$$\lambda_2/2 \le \phi(G) \le \sqrt{2\lambda_2}$$

$$\lambda_2 = \min_{x\in R^n,\ x\cdot d=0} \frac{\sum_{ij\in E} w_{ij}(x_i - x_j)^2}{\sum_i d_i x_i^2} = \min_{x\in R^n} \frac{\sum_{ij\in E} w_{ij}(x_i - x_j)^2}{\sum_i d_i x_i^2 - \left(\sum_i d_i x_i\right)^2/\sum_i d_i}$$

$$\le \min_{x\in \{0,1\}^n} \frac{\sum_{ij\in E} w_{ij}(x_i - x_j)^2}{\sum_i d_i x_i^2 - \left(\sum_i d_i x_i\right)^2/\sum_i d_i} = \min_S \frac{w(S,\bar{S})\,w(V)}{w(S)\,w(\bar{S})}$$

$$\le 2\phi(G)$$

Soo useful and central

Image segmentation

data clustering

network routing and design

VLSI layout

Parallel/distributed computing

...

certificate for constant edge expansion

mixing of Markov chains

graph partitioning

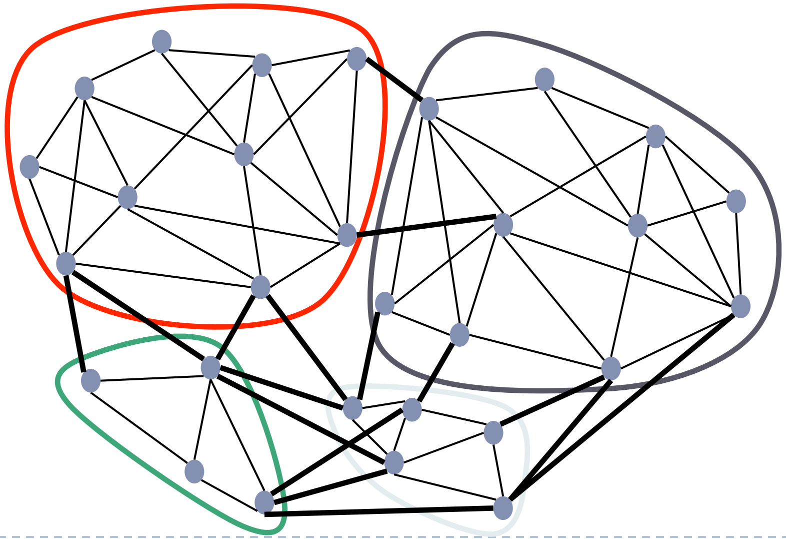
Pseudorandomness

…

Multiple parts

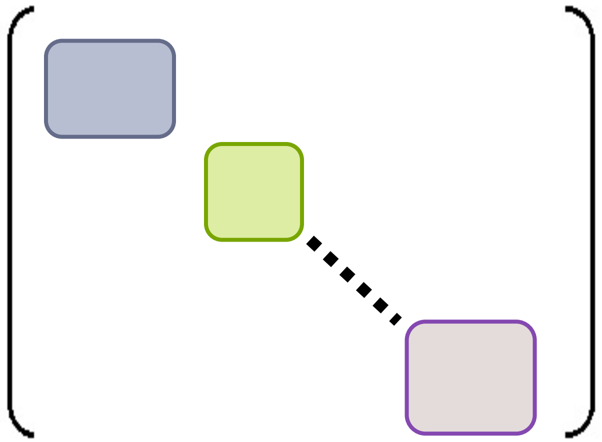
- Given G=(V,E), find k disjoint subsets of vertices S_1, S_2 ,..., S_k s.t. the maximum expansion among these is minimized.

$$\phi_k(G) = \min_{S_1,\ldots,S_k \subset V,\ \text{disjoint}} \max_i \phi(S_i)$$



Perron-Frobenius again

- $\lambda_k = 0$ if and only if graph has at least k connected components.
- If $\lambda_k \approx 0$, then is graph close to having k components ?
- Is there a Cheeger inequality? [Trevisan]



Cheeger’s inequality for multiple parts

Theorem.

[Lee-OveisGharan-Trevisan12; Louis-Raghavendra-Tetali-V.12]

$$\lambda_k/2 \le \phi_k(G) \le C\sqrt{\lambda_{1.01k} \log k}\ .$$

- k disjoint subsets, each with small expansion
- Alternatively, can get $(1-\epsilon)k$ subsets with $\sqrt{\lambda_k \log k}$
- Usual Cheeger is the special case of k=2

Algorithm [Louis-Raghavendra-Tetali-V.'12]

1. [Spectral embedding]

Embed vertices of G using top k eigenvectors

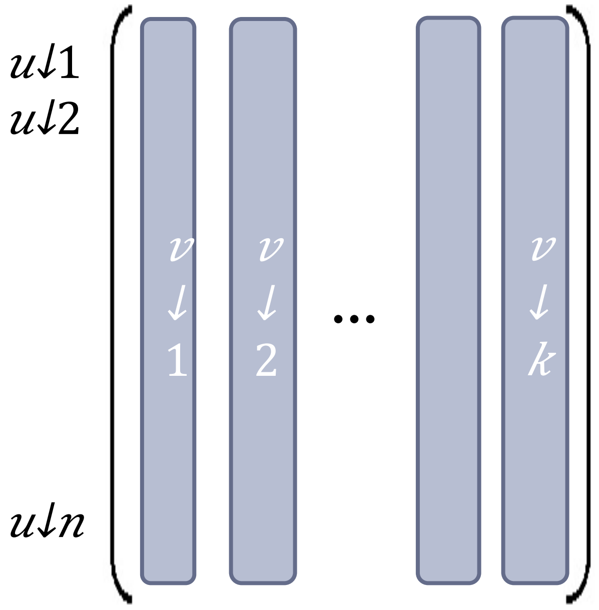
2. [Randomized rounding]

Partition randomly into k ordered subsets

3. [Cheeger cuts]

Apply Cheeger's algorithm to each ordered subset

Spectral embedding



$$u_i = \frac{1}{\sqrt{d_i}} \left(v_1(i), v_2(i), \dots, v_k(i) \right)$$

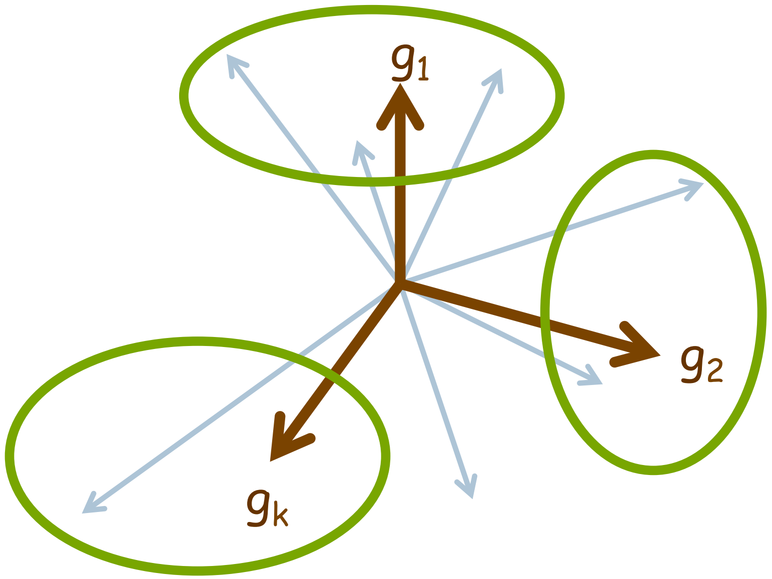
$$\frac{\sum_{i \sim j} \|u_i - u_j\|^2}{\sum_{i \in V} d_i \|u_i\|^2} \le \lambda_k$$

$\{\sqrt{d_i}\, u_i\}$ form an isotropic set of vectors

Randomized rounding

- Pick k random Gaussians: $g_1, g_2, \dots, g_k \sim N(0,1)^k$

- Project each u_i to each g_j.



- Assign each i to Gaussian j that maximizes $|u_i \cdot g_j|$, thus partitioning the vertices into k sets.

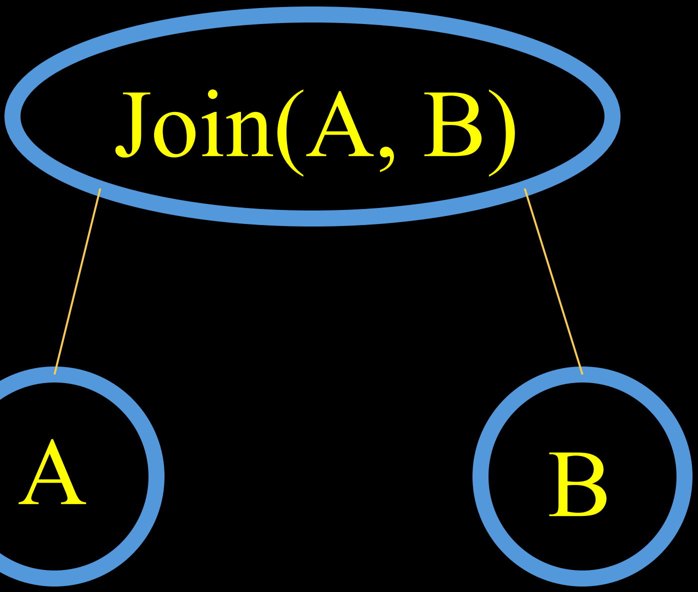
Outline

- Mixture Models
- Independent Component Analysis
- Finding Planted Structures (subgraphs, topic models etc.)
- Graph clustering
- Some relevant (?) snippets from the frontlines

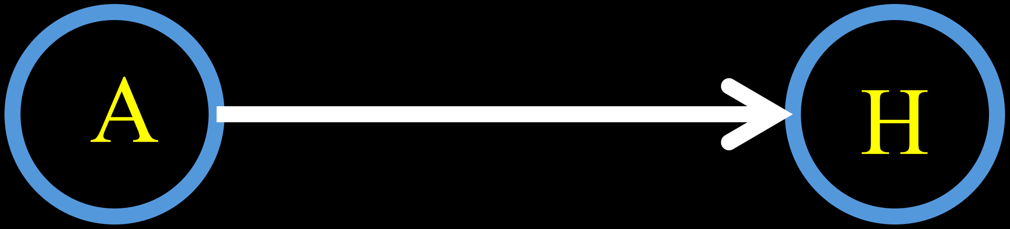
Representation: what is a concept?

1. Subset of neurons, such that if more than a certain fraction “fire”, then the concept is recognized.
2. Distribution over neurons
3. Activity pattern of neurons

Operations on Concepts: Join



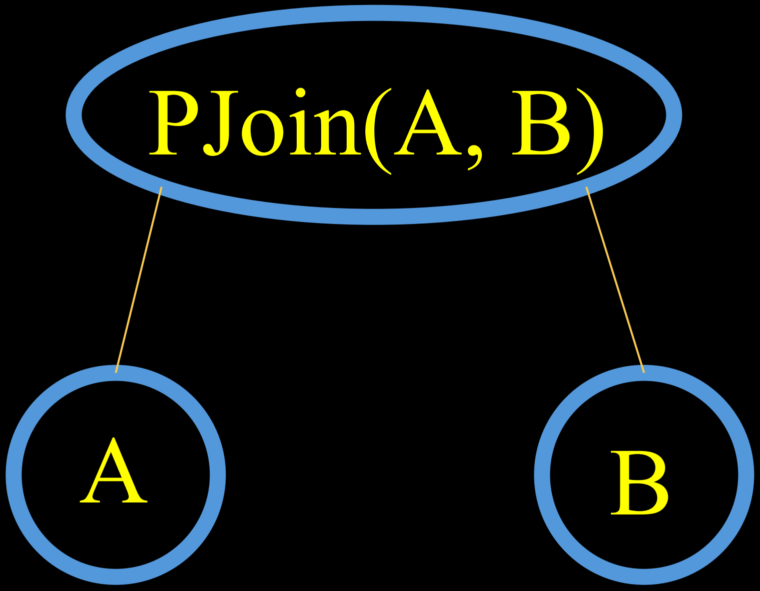
Operations on Items: Link (≈ Variable Binding)



Memorization

- Join is AND, Link is OR.
- Valiant: Join and Link can be used to memorize binary patterns of length two (subsets of Σ x Σ), via a short "neural" program.
  - "blue" and "water" = "ocean"
- What about n > 2?

Predictive Join
(with Christos Papadimitriou, 2015)



Memorization

“Learn a pattern *x*”

=

“on sensory presentation of *x*,

create a top-level item *I(x)*, which

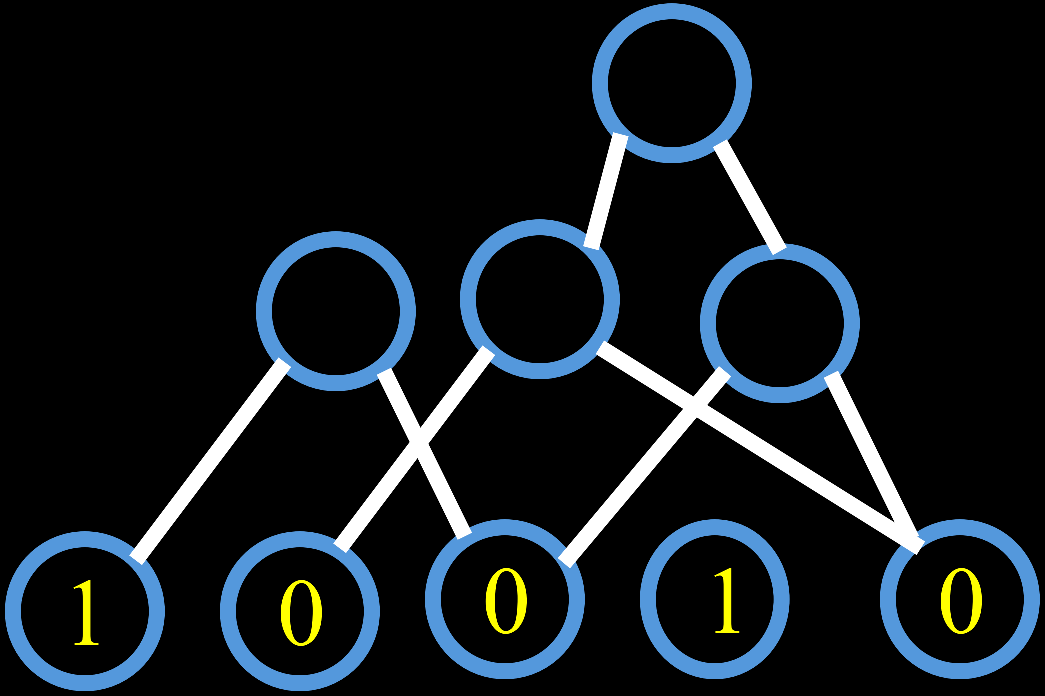
will fire precisely on all subsequent presentations of *x*”

Algorithm (x)

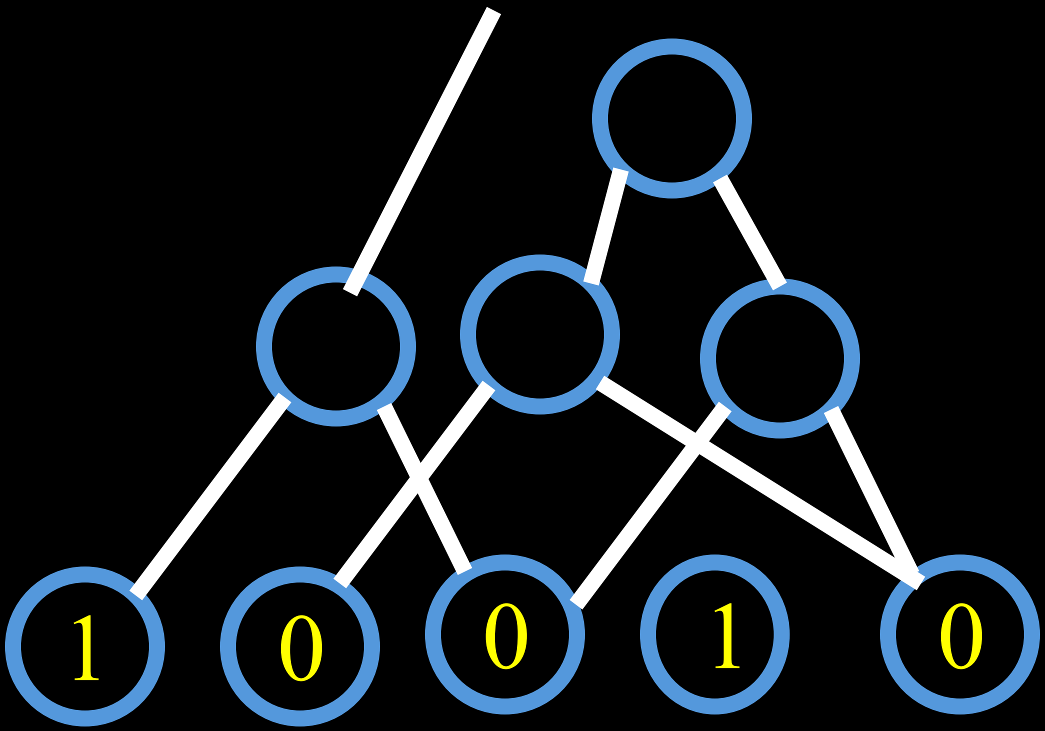
```
Repeat for $S$ steps:
        each sensory input is sensed
                with probability $p$
        PJoins created with probability $q$
        after delay $D$
        while existing Pjoins "do their thing"
```

Pjoin eligibility criteria: two items that have fired recently, with no parent that fired since.

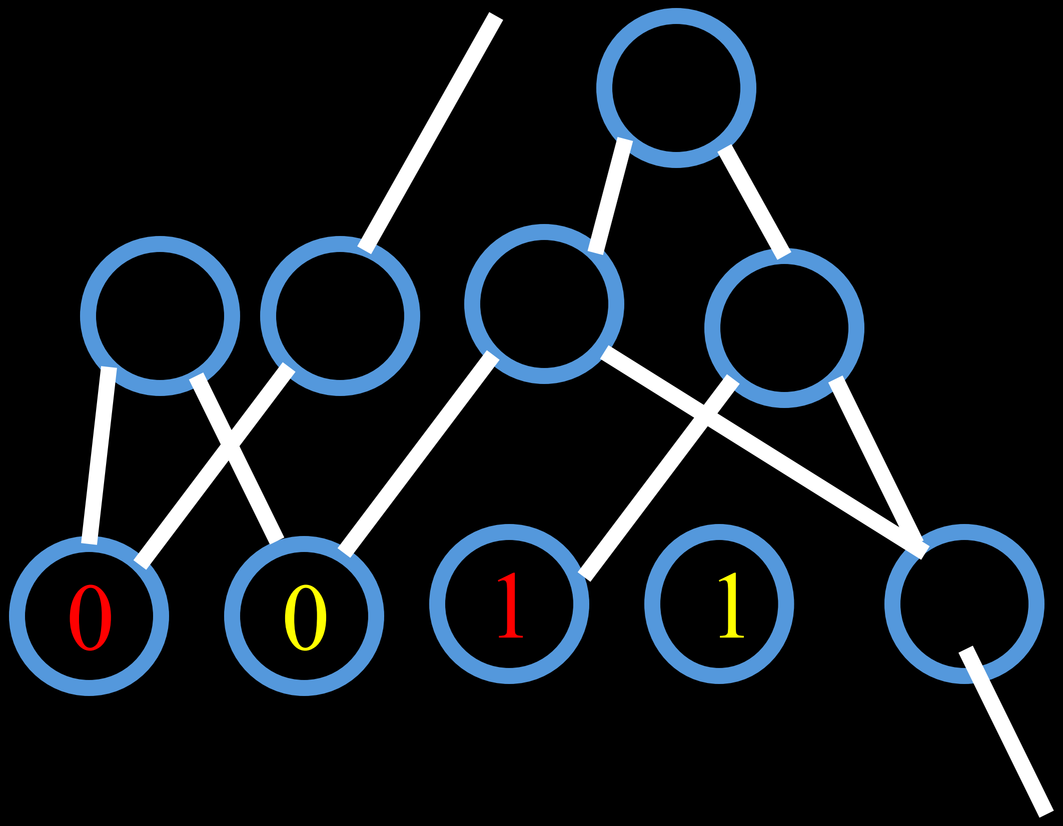
Presentation of a pattern



Second presentation



Other patterns: Share and build



Unsupervised Memorization

Theorem: Any subset of Σ^n of size m can be memorized whp and with total height O(log m + log n), provided that:

$D \geq \log n$, and $S \geq \log n / p$.

- *Any m patterns can be memorized.*

Simulations

- Patterns with up to n = 100 base features
- all learning activity completed in < 80 steps
- sharing as predicted
- *majority of firing traffic downwards*

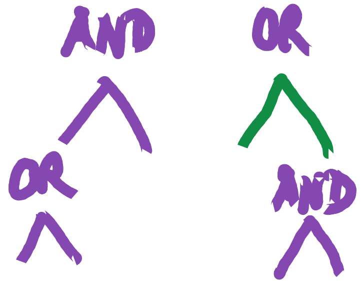
Learning Thresholds, *neurally*
(with Christos Papadimitriou & Samantha Petti, 2016)

Goal: Develop a mathematical framework to explain cognitive function

Neurally plausible = highly distributed, little synchrony, little global control

Algorithm:

Pick one of two small JOIN-LINK trees at random,
apply to a random subset of items
repeat



Thm. For any desired threshold function, there exists a distribution on two trees s.t. later items reliably compute that threshold. (Independent of the number of items!)

Q. Cortical microcircuits for learning?

Emergence of clustering in random graphs

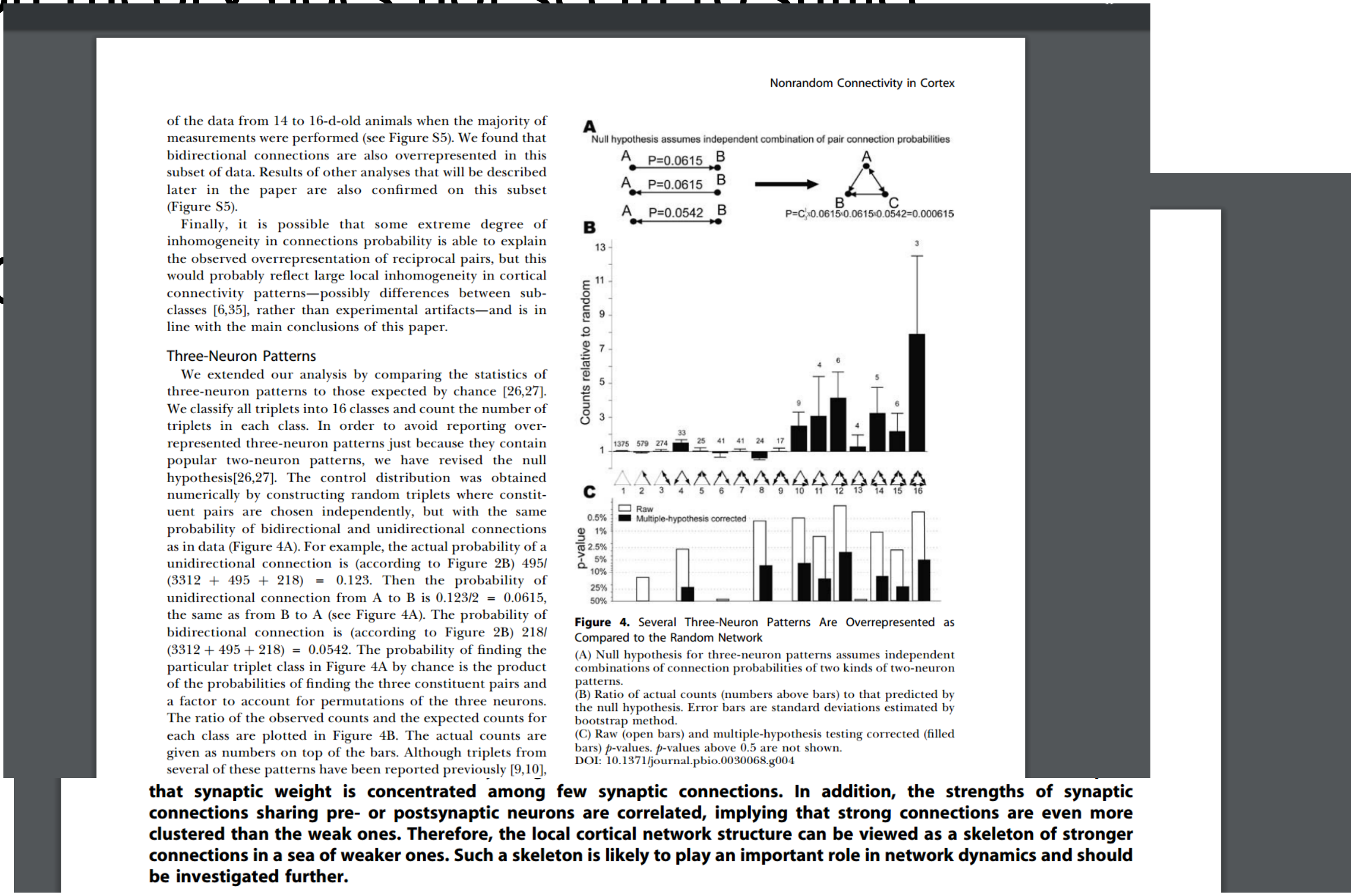
- The classical random graph model $G{\downarrow}n,p$
- Power-law (scale-free) random graphs
- Small-world networks

- Don't capture clustering coefficient: "neighbors are more likely to be connected"

- Random Overlapping Communities (ROC) model

Models of *Connectome*

- Random graph theory does not seem to suffice
- [Song et al 20



of the data from 14 to 16-d-old animals when the majority of measurements were performed (see Figure S5). We found that bidirectional connections are also overrepresented in this subset of data. Results of other analyses that will be described later in the paper are also confirmed on this subset (Figure S5).

Finally, it is possible that some extreme degree of inhomogeneity in connections probability is able to explain the observed overrepresentation of reciprocal pairs, but this would probably reflect large local inhomogeneity in cortical connectivity patterns—possibly differences between subclasses [6,35], rather than experimental artifacts—and is in line with the main conclusions of this paper.

Three-Neuron Patterns

We extended our analysis by comparing the statistics of three-neuron patterns to those expected by chance [26,27]. We classify all triplets into 16 classes and count the number of triplets in each class. In order to avoid reporting overrepresented three-neuron patterns just because they contain popular two-neuron patterns, we have revised the null hypothesis[26,27]. The control distribution was obtained numerically by constructing random triplets where constituent pairs are chosen independently, but with the same probability of bidirectional and unidirectional connections as in data (Figure 4A). For example, the actual probability of a unidirectional connection is (according to Figure 2B) 495/(3312 + 495 + 218) = 0.123. Then the probability of unidirectional connection from A to B is 0.123/2 = 0.0615, the same as from B to A (see Figure 4A). The probability of bidirectional connection is (according to Figure 2B) 218/(3312 + 495 + 218) = 0.0542. The probability of finding the particular triplet class in Figure 4A by chance is the product of the probabilities of finding the three constituent pairs and a factor to account for permutations of the three neurons. The ratio of the observed counts and the expected counts for each class are plotted in Figure 4B. The actual counts are given as numbers on top of the bars. Although triplets from several of these patterns have been reported previously [9,10],

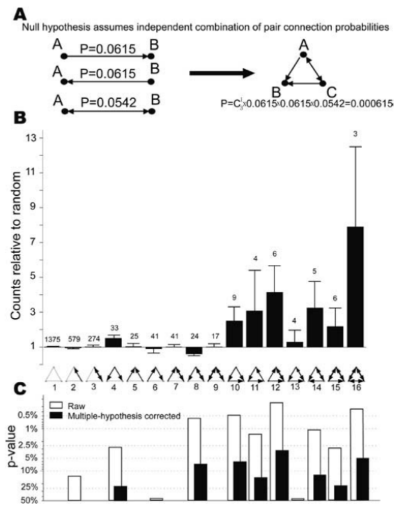


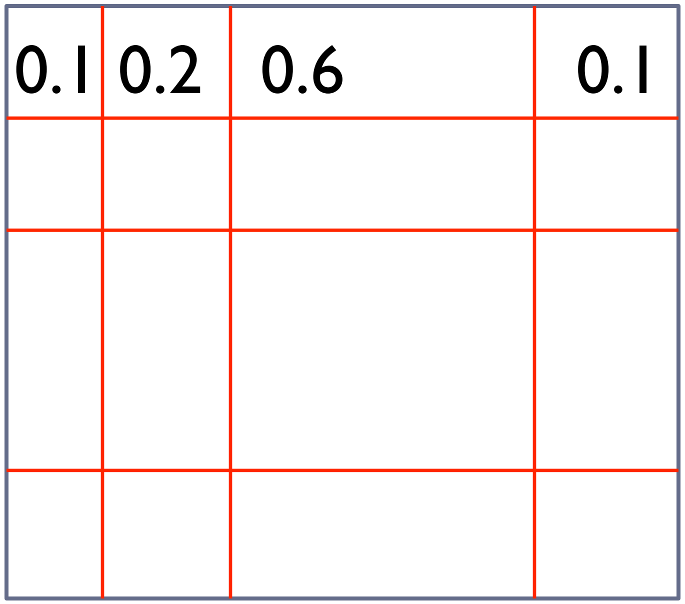
Figure 4. Several Three-Neuron Patterns Are Overrepresented as Compared to the Random Network

(A) Null hypothesis for three-neuron patterns assumes independent combinations of connection probabilities of two kinds of two-neuron patterns.
(B) Ratio of actual counts (numbers above bars) to that predicted by the null hypothesis. Error bars are standard deviations estimated by bootstrap method.
(C) Raw (open bars) and multiple-hypothesis testing corrected (filled bars) p-values. p-values above 0.5 are not shown.
DOI: 10.1371/journal.pbio.0030068.g004

that synaptic weight is concentrated among few synaptic connections. In addition, the strengths of synaptic connections sharing pre- or postsynaptic neurons are correlated, implying that strong connections are even more clustered than the weak ones. Therefore, the local cortical network structure can be viewed as a skeleton of stronger connections in a sea of weaker ones. Such a skeleton is likely to play an important role in network dynamics and should be investigated further.

Capturing edge and triangle density

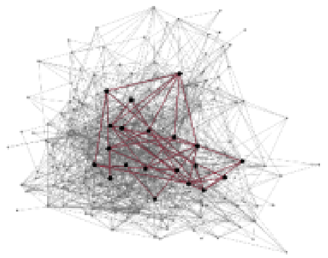
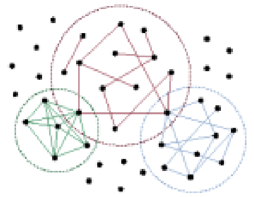
- Impossible for any stochastic block model unless the number of blocks grows with graph size!



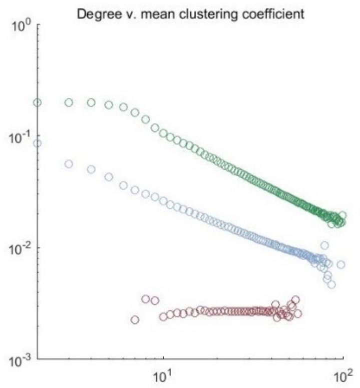
- (e.g., for a hypercube graph)

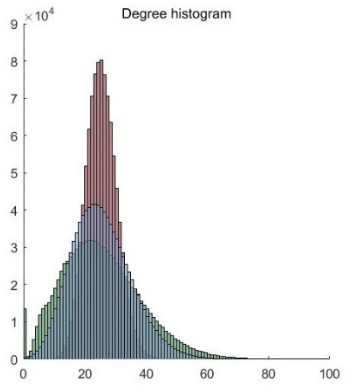
Random Overlapping Communities (ROC)

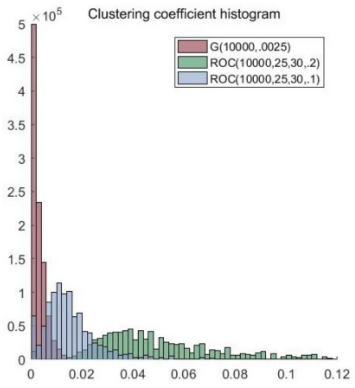
- A graph is built by taking the union of many relatively dense random subgraphs.

- Thm.[Petti-V. 2017] Any realizable clustering coefficient and degree distribution can be approximated by a ROC random graph.







- Higher degree vertices are in fewer triangles

Thank you!