Geometric Tools for Identifying Structure in Large Social and Information Networks

Michael W. Mahoney

Stanford University

(ICML 2010 and KDD 2010 Tutorial)

(For more info, see:
http:// cs.stanford.edu/people/mmahoney/
or Google on "Michael Mahoney")

Lots of "networked data" out there!

- Technological and communication networks
  - AS, power-grid, road networks
- Biological and genetic networks
  - food-web, protein networks
- Social and information networks
  - collaboration networks, friendships; co-citation, blog cross-postings, advertiser-bidded phrase graphs ...
- Financial and economic networks
  - encoding purchase information, financial transactions, etc.
- Language networks
  - semantic networks ...
- Data-derived "similarity networks"
  - recently popular in, e.g., "manifold" learning
- ...

Large Social and Information Networks

• Social nets	Nodes	Edges	Description
LIVEJOURNAL	4,843,953	42,845,684	Blog friendships [4]
EPINIONS	75,877	405,739	Who-trusts-whom [35]
FLICKR	404,733	2,110,078	Photo sharing [21]
DELICIOUS	147,567	301,921	Collaborative tagging
CA-DBLP	317,080	1,049,866	Co-authorship (CA) [4]
CA-COND-MAT	21,363	91,286	CA cond-mat [25]
• Information networks			
CIT-HEP-TH	27,400	352,021	hep-th citations [13]
BLOG-POSTS	437,305	565,072	Blog post links [28]
• Web graphs			
WEB-GOOGLE	855,802	4,291,352	Web graph Google
WEB-WT10G	1,458,316	6,225,033	TREC WT10G web
• Bipartite affiliation (authors-to-papers) networks			
ATP-DBLP	615,678	944,456	DBLP [25]
ATP-ASTRO-PH	54,498	131,123	Arxiv `astro-ph` [25]
• Internet networks			
AS	6,474	12,572	Autonomous systems
GNUTELLA	62,561	147,878	P2P network [36]

Table 1: Some of the network datasets we studied.

Sponsored ("paid") Search

Text-based ads driven by user query

recipe indian food - Yahoo! Search Results - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://search.yahoo.com/search?p=recipe+indian+food&fr=yfp-t-501&toggle=1&cop=mss&ei=UTF-8

Yahoo! My Yahoo! Mail Welcome, Guest [Sign In] Advertiser Sign In Help

Web | Images | Video | Local | Shopping | more

YAHOO! SEARCH recipe indian food Search

Answers

Search Results 1 - 10 of about 7,260,000 for recipe indian food - 0.19 sec. (About this page)

SPONSOR RESULTS

- **Recipe Indian Food**
  www.MonsterMarketplace.com - Browse and compare great deals on **recipe indian food**.
- **Indian Food**
  sanfrancisco.citysearch.com - Find great **Indian** restaurants in your area today. Search here.

1. **indian food recipe**
   indian food recipe ... Title: **Indian Food Recipe**. Yield: 4 Servings. Ingredients. 1 bunch ... to the echo by: Jonathan Kandell **Indian Food** Recipes Put ...
   recipes.chef2chef.net/**recipe**-archive/43/231458.shtml - 13k - Cached - More from this site
2. **Recipe** Gal: **Indian Foods**
   Indian Recipes from **Recipe** Gal's Archives ... All **Food** Posters. Travel Posters. **Indian** Recipes. **Indian** Breads **Indian** Chicken Recipes ...
   www.**recipe**gal.com/**indian** - 10k - Cached - More from this site
3. **Indian** Recipes, **Indian Food Recipe**, South **Indian** Recipes, **Indian** ...
   indian recipes, **indian food recipe**, south **indian** Recipes, **indian** cooking Recipes, ... **Indian** Recipes, **Indian Food Recipe**, South **Indian** Recipes, **Indian** Cooking **Recipe**, ...
   www.india4world.com/**indian-recipe** - 17k - Cached - More from this site
4. Paav Bhaaji - **Recipe** for Paav Bhaaji - Pao Bhaji

SPONSOR RESULTS

Indian Food
Buy **indian food** at SHOP.COM. Search our free shipping offers.
www.SHOP.com

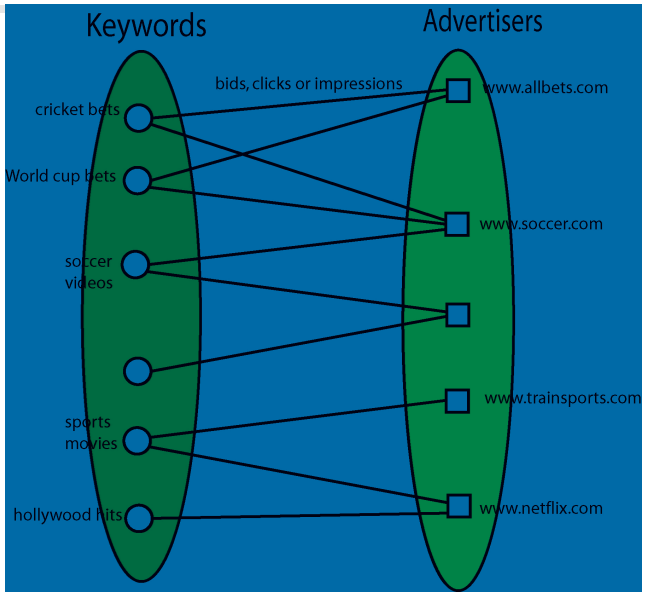
Recipe India **Food**
Find and Compare prices on **recipe** india **food** at Smarter.com.
www.smarter.com

Chinese **Food Recipe** Books on Cataloglink
Find chinese **food recipe** books on CatalogLink.
www.CatalogLink.com

$19.97 Over 500 Chinese **Recipes** Cookbook
100% Satisfaction Guaranteed, 543-Page Chinese Cookbook Only $19.97.

Done

Sponsored Search Problems



Keyword-advertiser graph:

- provide new ads
- maximize CTR, RPS, advertiser ROI

Motivating cluster-related problems:

- Marketplace depth broadening:

  find new advertisers for a particular query/submarket

- Query recommender system:

  suggest to advertisers new queries that have high probability of clicks

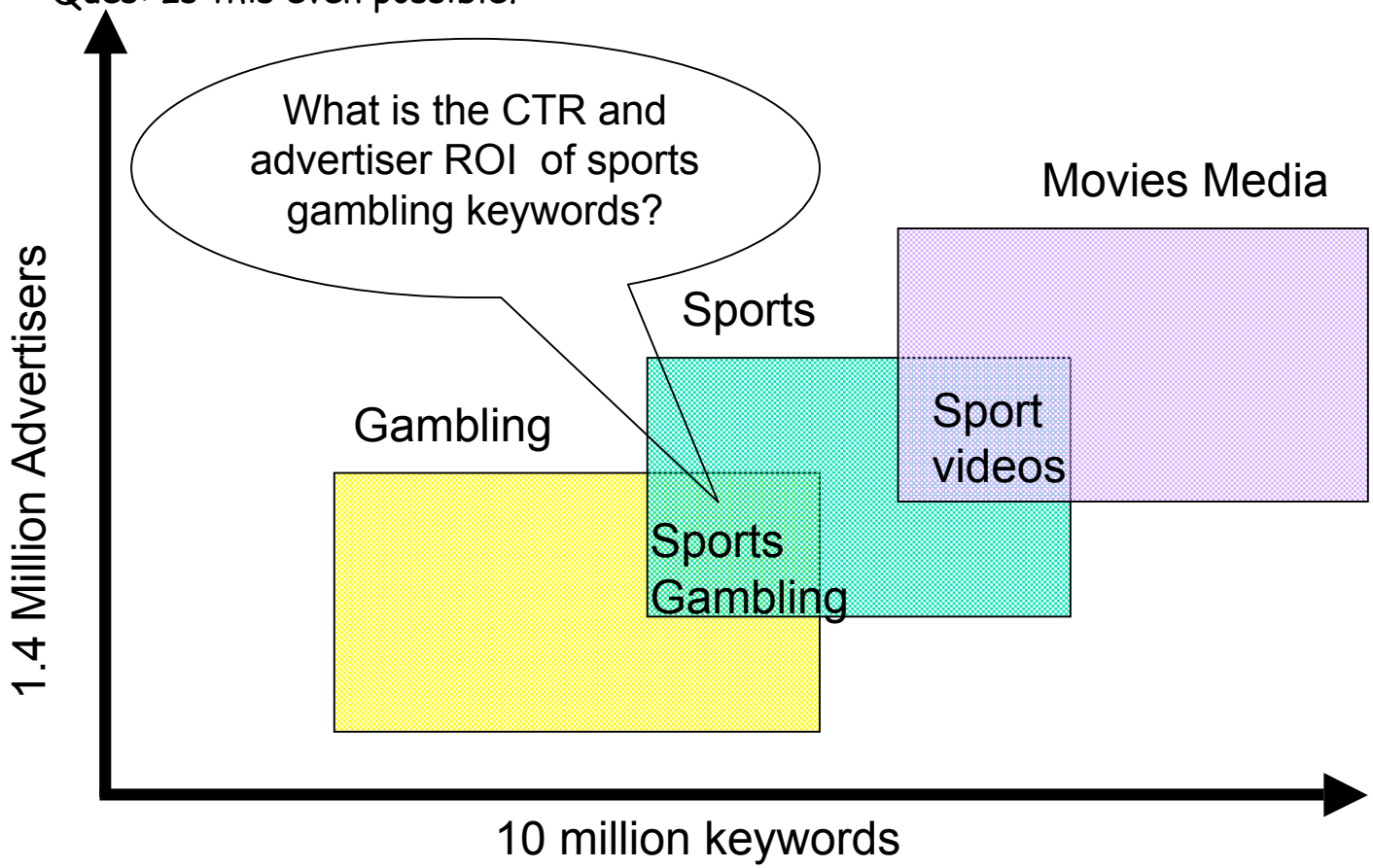
- Contextual query broadening:

  broaden the user's query using other context information

Micro-markets in sponsored search

Goal: Find *isolated* markets/clusters (in an advertiser-bidded phrase bipartite graph) with *sufficient money/clicks* with *sufficient coherence*.

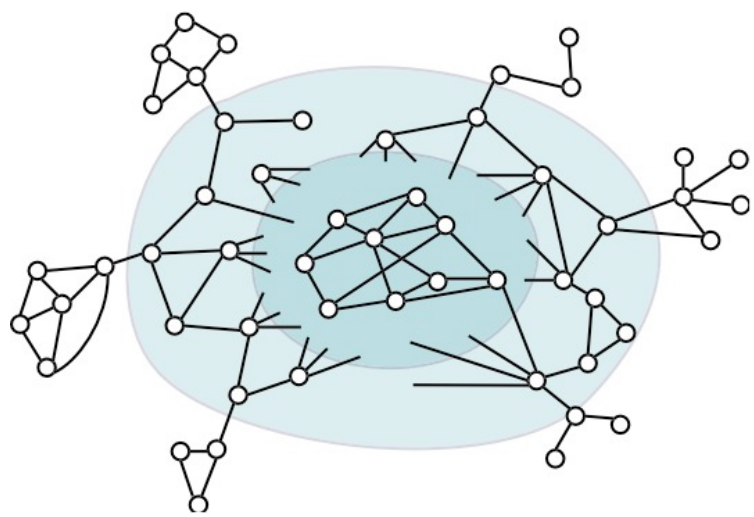
Ques: Is this even possible?



How people think about networks

"Interaction graph" ***model*** of networks:

- Nodes represent "entities"
- Edges represent "interaction" between pairs of entities

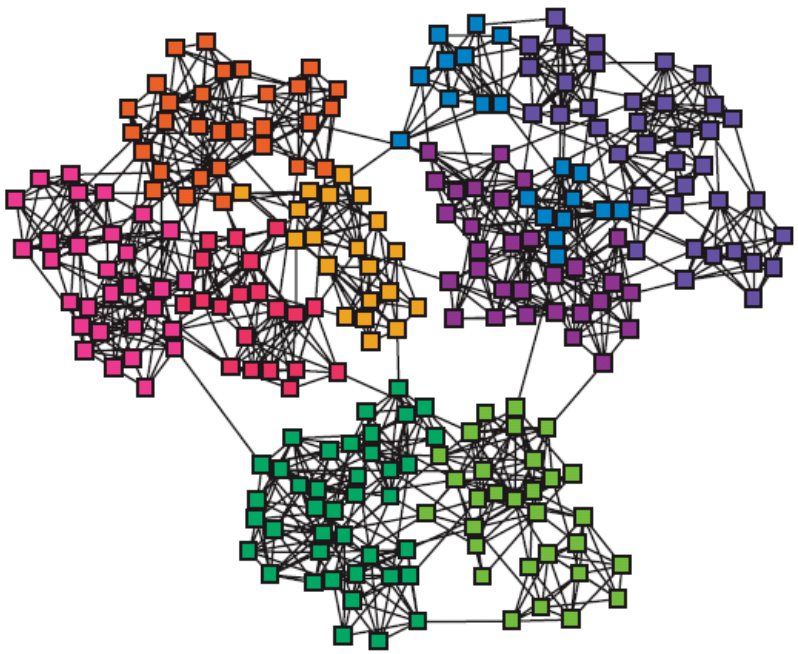


Graphs are combinatorial, not obviously-geometric

- Strength: powerful framework for analyzing *algorithmic complexity*
- Drawback: geometry used for learning and *statistical inference*

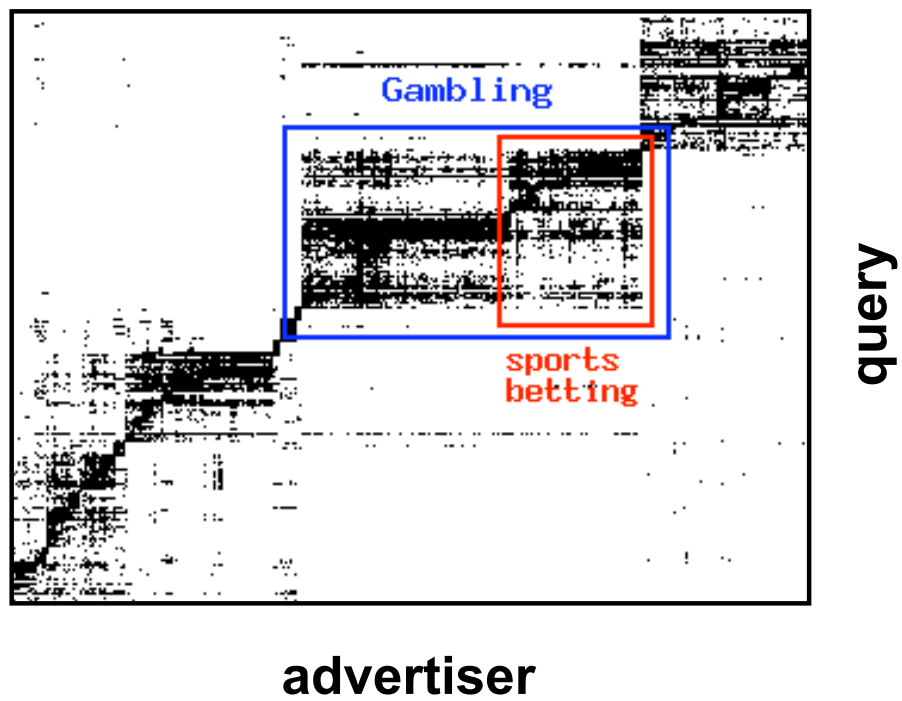
How people think about networks

A schematic illustration …



… of hierarchical clusters?

Some evidence for micro-markets in sponsored search?



Questions of interest ...

What are degree distributions, clustering coefficients, diameters, etc.?

Heavy-tailed, small-world, expander, geometry+rewiring, local-global decompositions, ...

Are there natural clusters, communities, partitions, etc.?

Concept-based clusters, link-based clusters, density-based clusters, ...

(e.g., *isolated* micro-markets with *sufficient money/clicks* with *sufficient coherence*)

How do networks grow, evolve, respond to perturbations, etc.?

Preferential attachment, copying, HOT, shrinking diameters, ...

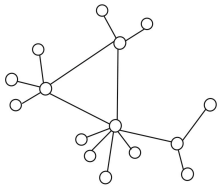
How do dynamic processes - search, diffusion, etc. - behave on networks?

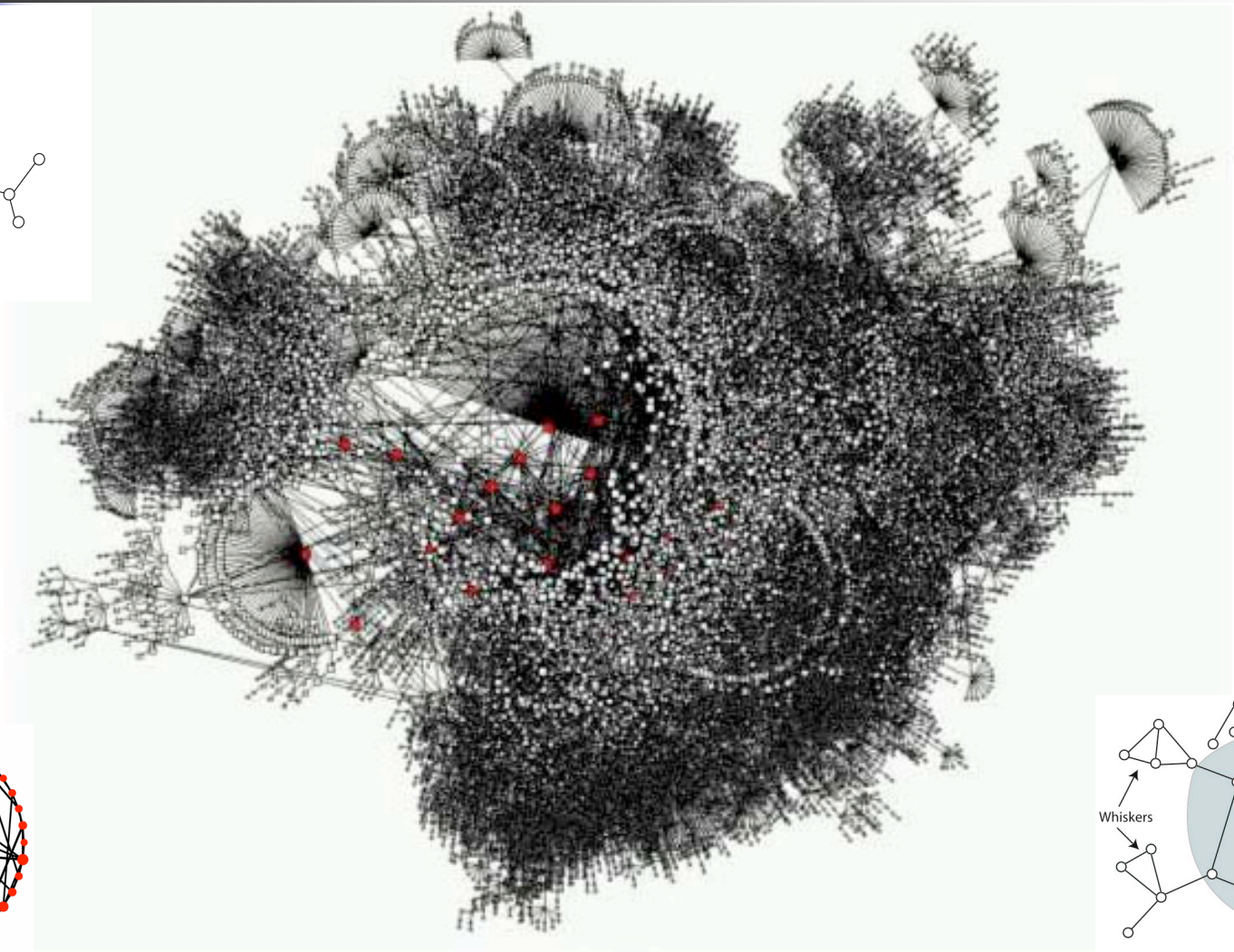
Decentralized search, undirected diffusion, cascading epidemics, ...

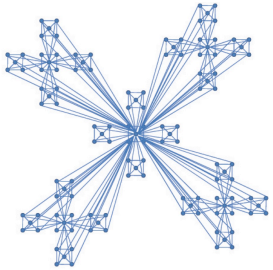
How best to do learning, e.g., classification, regression, ranking, etc.?

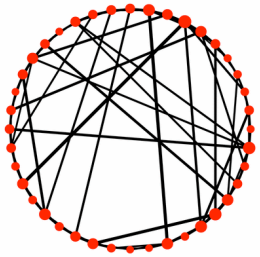
Information retrieval, machine learning, ...

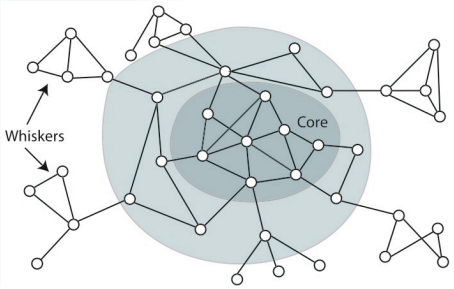
What do these networks "look" like?



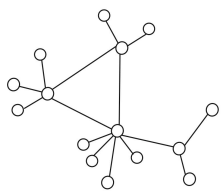








Popular approaches to large network data

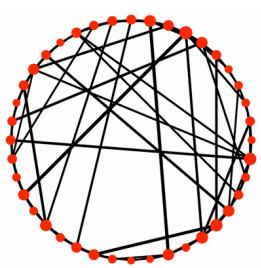


Heavy-tails and power laws (at *large size-scales*):

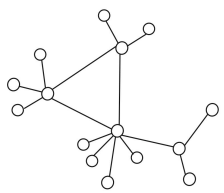
- extreme heterogeneity in local environments, e.g., as captured by degree distribution, and relatively unstructured otherwise
- basis for preferential attachment models, optimization-based models, power-law random graphs, etc.

Local clustering/structure (at *small size-scales*):

- local environments of nodes have structure, e.g., captures with clustering coefficient, that is meaningfully "geometric"
- basis for small world models that start with global "geometry" and add random edges to get small diameter and preserve local "geometry"



Popular approaches to data more generally

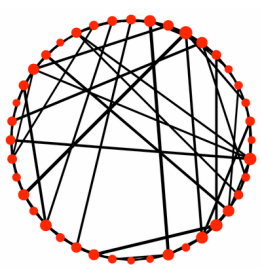


Use **geometric data analysis** tools:

- Low-rank methods - very popular and flexible
- Manifold methods - use other distances, e.g., diffusions or nearest neighbors, to find "curved" low-dimensional spaces

These geometric data analysis tools:

- View data as a point cloud in R^n, i.e., *each of the m data points is a vector in R^n*
- Based on SVD*, a basic vector space *structural* result
- Geometry gives a lot -- scalability, robustness, capacity control, basis for inference, etc.



*perhaps implicitly in an infinite-dimensional non-linearly transformed feature space (as with manifold and other Reproducing Kernel methods)

Can these approaches be combined?

These approaches are *very* different:

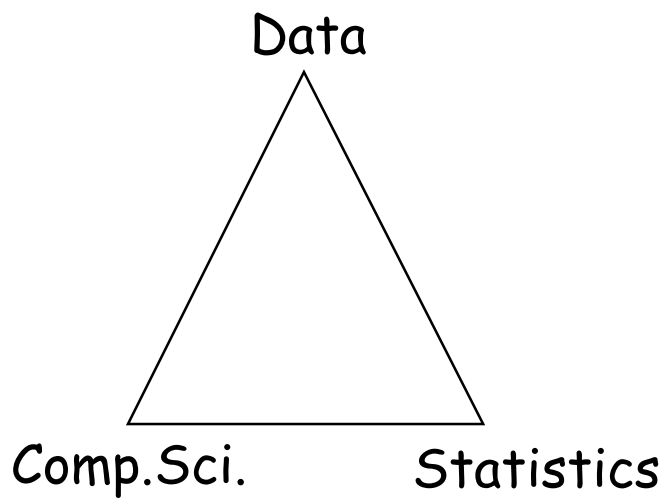
- network is a *single data point*---*not a collection of feature vectors* drawn from a distribution, and not really a matrix
- can't easily let m or n (number of data points or features) go to infinity---so nearly every such theorem fails to apply

Can associate matrix with a graph and vice versa, but:

- often do more damage than good
- questions asked tend to be *very* different
- graphs are really *combinatorial* things*

*But graph geodesic distance is a metric, and metric embeddings give fast algorithms!

Modeling data as matrices and graphs



In computer science:

- data are typically discrete, e.g., graphs
- focus is on fast algorithms for the given data set

In statistics*:

- data are typically continuous, e.g. vectors
- focus is on inferring something about the world

*very broadly-defined!

Algorithmic vs. Statistical Perspectives

Lambert (2000)

Computer Scientists

• Data: are a record of everything that happened.

• Goal: process the data to find interesting patterns and associations.

• Methodology: Develop approximation algorithms under different models of data access since the goal is typically computationally hard.

Statisticians

• Data: are a particular random instantiation of an underlying process describing unobserved patterns in the world.

• Goal: is to extract information about the world from noisy data.

• Methodology: Make inferences (perhaps about unseen events) by positing a model that describes the random variability of the data around the deterministic model.

Perspectives are NOT incompatible

- Statistical/probabilistic ideas are central to recent work on developing improved randomized algorithms for matrix problems.

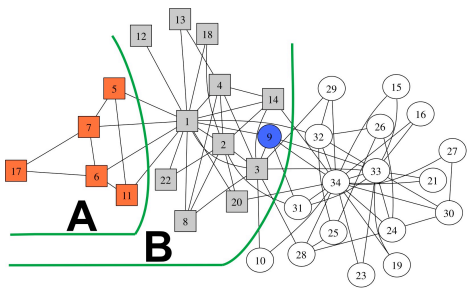
- Intractable optimization problems on graphs/networks yield to approximation when assumptions made about network participants.

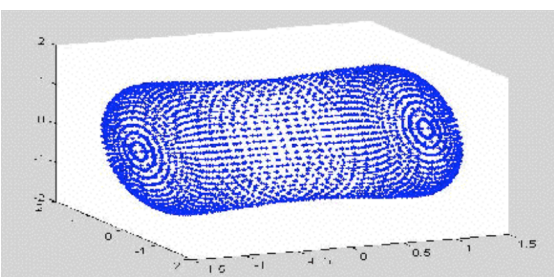
- In boosting, the computation parameter (i.e., the number of iterations) also serves as a regularization parameter.

- Approximations algorithms can implicitly regularize large graph problems (which can lead to *geometric network analysis tools*!).

What do the data "look like" (if you *squint* at them)?

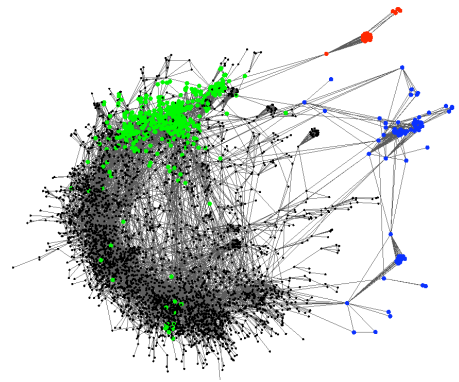
A "hot dog"?

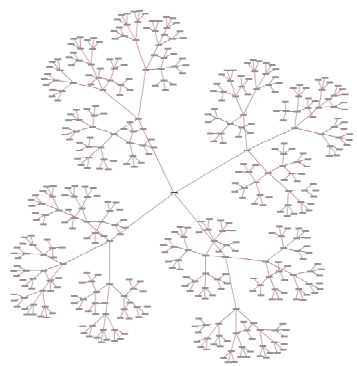




(or pancake that embeds well in low dimensions)

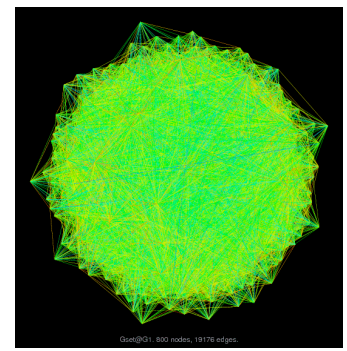
A "tree"?

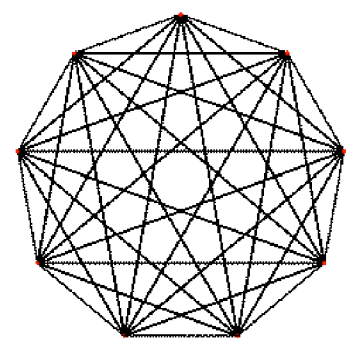




(or tree-like hyperbolic structure)

A "point"?





(or clique-like or expander-like structure)

Goal of the tutorial

Cover algorithmic and statistical work on identifying and exploiting "geometric" structure in large "networks"

- Address underlying theory, bridging the theory-practice gap, empirical observations, and future directions

Themes to keep in mind:

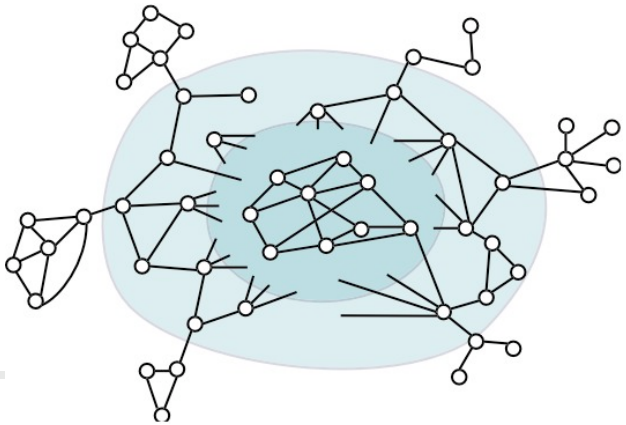
- Even infinite-dimensional Euclidean structure is too limiting

  (in adversarial environments, you never "flesh out" the low-dimensional space)

- Scalability and robustness are central

  (tools that do well on small data often do worse on large data)

Overview



Popular algorithmic tools with a geometric flavor

• PCA, SVD; interpretations, kernel-based extensions; algorithmic and statistical issues; and limitations

Graph algorithms and their geometric underpinnings

• Spectral, flow, multi-resolution algorithms; their implicit geometric basis; global and scalable local methods; expander-like, tree-like, and hyperbolic structure

Novel insights on structure in large informatics graphs

• Successes and failures of existing models; empirical results, including "experimental" methodologies for probing network structure, taking into account algorithmic and statistical issues; implications and future directions

Overview (more detail, 1 of 4)

Popular algorithmic tools with a geometric flavor

- PCA and SVD, including computational/algorithmic and statistical/geometric issues
- Domain-specific interpretation of spectral concepts, e.g., localization, homophily, centrality
- Kernel-based extensions currently popular in machine learning
- Difficulties and limitations of popular tools

Overview (more detail, 2 of 4)

Graph algorithms and their geometric underpinnings

- Spectral, flow, multi-resolution algorithms for graph partitioning, including theoretical basis and implementation issues
- Geometric and statistical perspectives, including "worst case" examples for each and behavior on "typical" classes of graphs
- Recent "local" methods and "cut improvement" methods; methods that "interpolate" between spectral and flow
- Tools for identifying "tree-like" or "hyperbolic" structure, and intuitions associated with this structure

Overview (more detail, 3 of 4)

Novel insights on structure in large informatics graphs

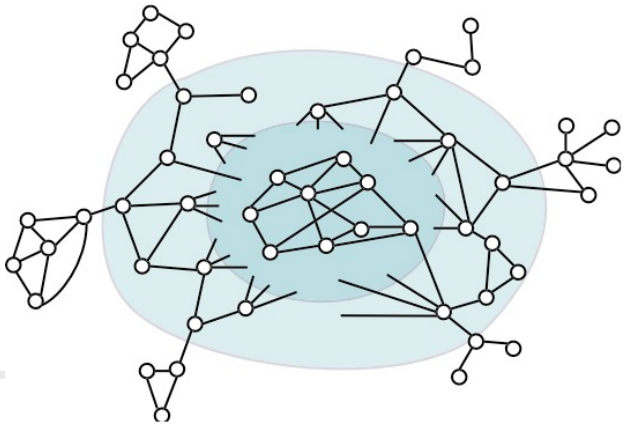
- Small-world and heavy-tailed models to capture local clustering and/or large-scale heterogeneity
- Issues of "pre-existing" versus "generated" geometry
- Empirical successes and failings of popular models, including densification, diameters, clustering, and community structure
- "Experimental" methodologies for "probing" network structure

Overview (more detail, 4 of 4)

Novel insights, (cont.)

- Empirical results on "local" geometric structure, "global" metric structure, and the coupling between these
- Implicit regularization by worst-case approximation algorithms
- Implications for clustering, routing, information diffusion, visualization, and the design of machine learning tools
- Implications for dynamics evolution *of* graphs, dynamics *on* graphs, and machine learning and data analysis on networks

Overview



Popular algorithmic tools with a geometric flavor

- PCA, SVD; interpretations, kernel-based extensions; algorithmic and statistical issues; and limitations

Graph algorithms and their geometric underpinnings

- Spectral, flow, multi-resolution algorithms; their implicit geometric basis; global and scalable local methods; expander-like, tree-like, and hyperbolic structure

Novel insights on structure in large informatics graphs

- Successes and failures of existing models; empirical results, including "experimental" methodologies for probing network structure, taking into account algorithmic and statistical issues; implications and future directions

The Singular Value Decomposition (SVD)

<u>The formal definition</u>:

Given any m x n matrix *A*, one can decompose it as:

$$\underset{m \times n}{\begin{pmatrix} \\ A \\ \\ \end{pmatrix}} = \underset{m \times \rho}{\begin{pmatrix} \\ U \\ \\ \end{pmatrix}} \cdot \underset{\rho \times \rho}{\begin{pmatrix} \\ \Sigma \\ \\ \end{pmatrix}} \cdot \underset{\rho \times n}{\begin{pmatrix} \\ V \\ \\ \end{pmatrix}}^T$$

ρ: rank of *A*

U (V): orthogonal matrix containing the left (right) singular vectors of *A*.

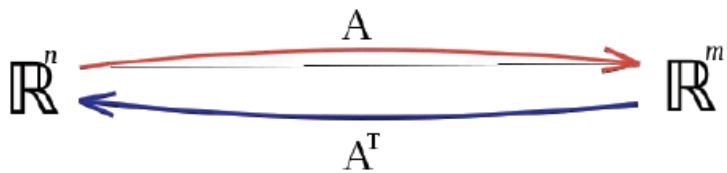
Σ: diagonal matrix containing $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_\rho$, the singular values of *A*.

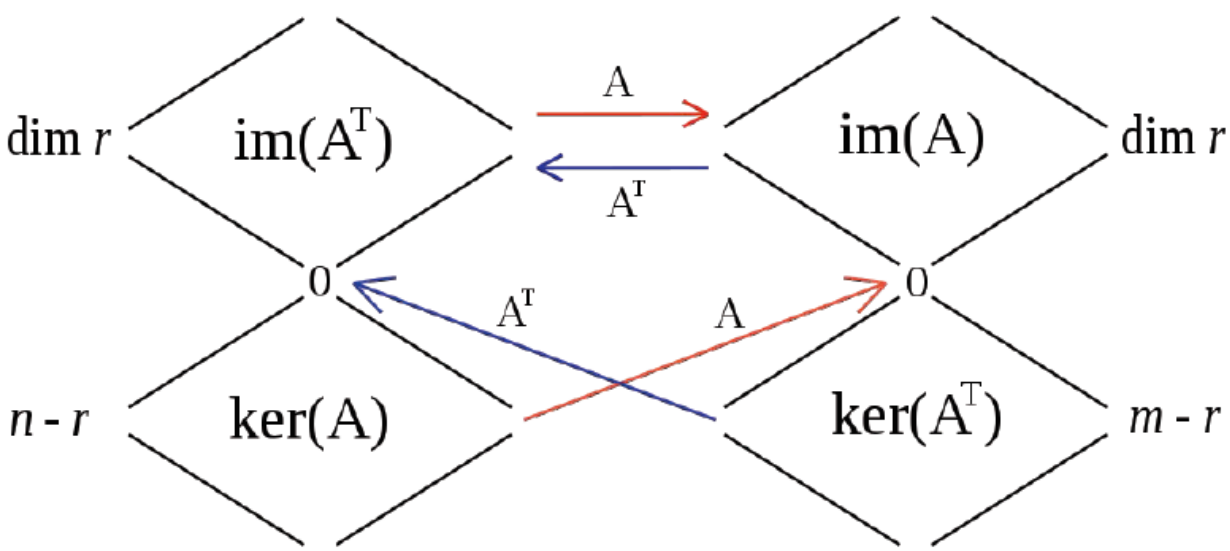
SVD is the "the Rolls-Royce and the Swiss Army Knife of Numerical Linear Algebra."*

*Dianne O'Leary, MMDS 2006

SVD: A fundamental *structural* result

SVD: a fundamental *structural* result of vector spaces (with both algorithmic and statistical consequences)





U: orthogonal basis for the column space

V: orthogonal basis for the row space

Σ: gives *orthogonalized* "stretch" factors*

*i.e., in the basis of U and V, *A* is diagonal.

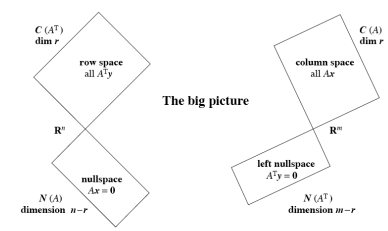


Figure 3.5: The dimensions of the Four Fundamental Subspaces (for R and for A).

Rank-k approximations (A_k)

Truncate the SVD at the top-k terms:

Keep the "most important" k-dim subspace.

$$\underset{m \times n}{\begin{pmatrix} \\ A_k \\ \\ \end{pmatrix}} = \underset{m \times k}{\begin{pmatrix} \\ U_k \\ \\ \end{pmatrix}} \cdot \underset{k \times k}{\begin{pmatrix} \\ \Sigma_k \\ \\ \end{pmatrix}} \cdot \underset{k \times n}{\begin{pmatrix} \\ V_k^T \\ \\ \end{pmatrix}}$$

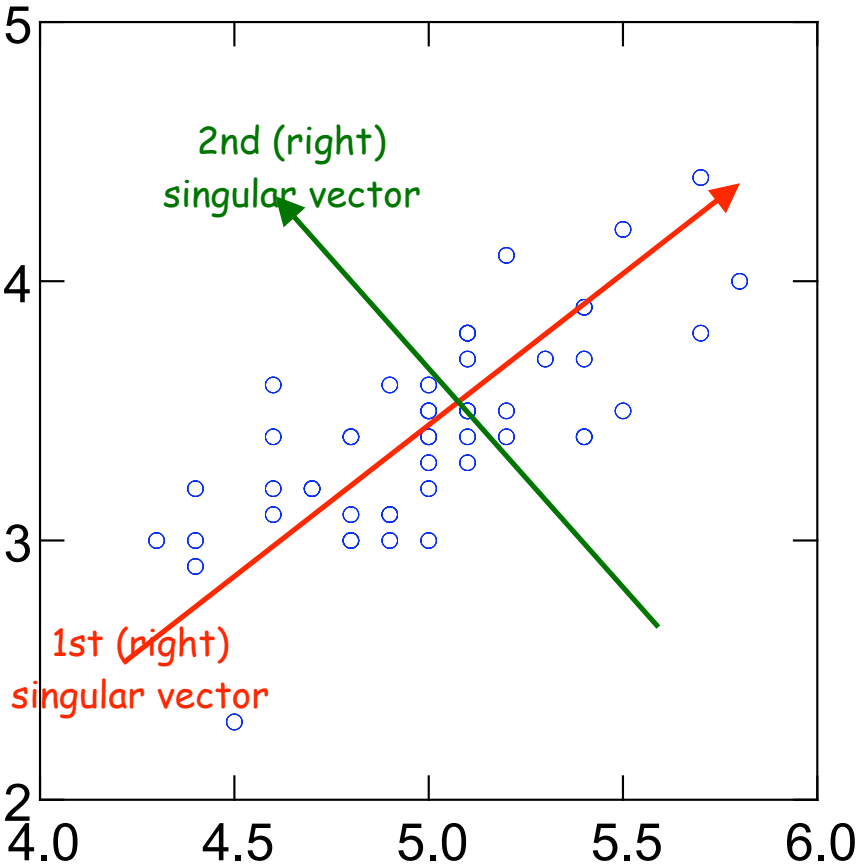
U_k (V_k): orthogonal matrix containing the top k left (right) singular vectors of A.

Σ_k: diagonal matrix containing the top k singular values of A.

<u>Important</u>: Keeping top k singular vectors provides "best" rank-k approximation to A (w.r.t. Frobenius norm, spectral norm, etc.):

$$A_k = \text{argmin}\{ \|A-X\|_{2,F} : \text{rank}(X) \leq k \}.$$

Singular vectors, intuition



Let the blue circles represent *m* data points in a 2-D Euclidean space.

Then, the SVD of the *m-by-2* matrix of the data will return …

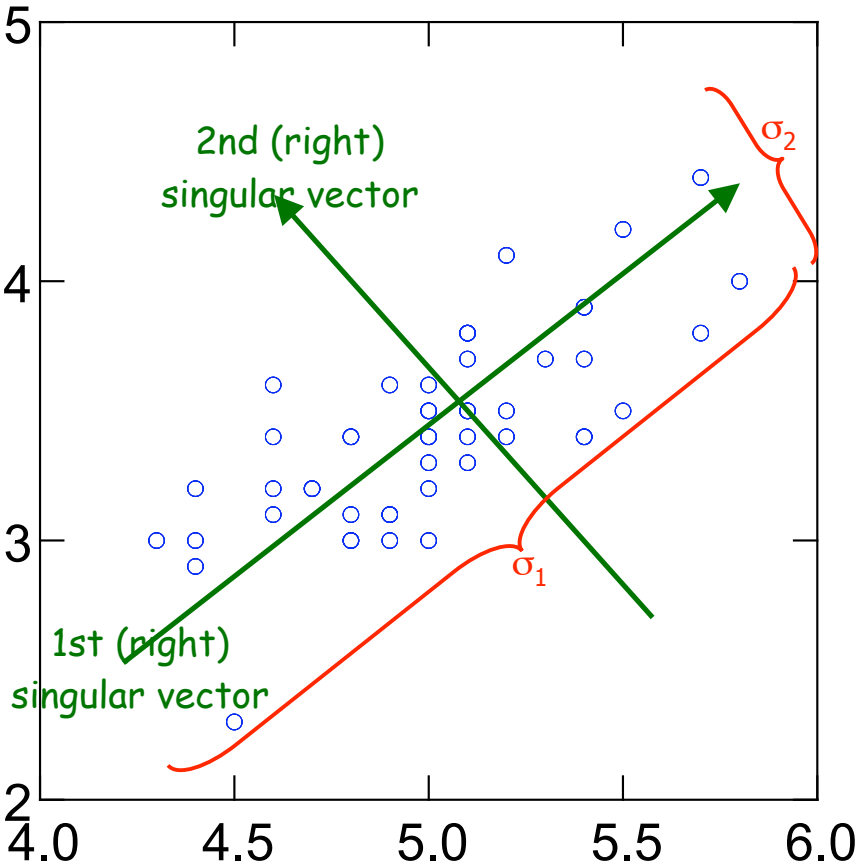
<u>1st (right) singular vector:</u>

direction of maximal variance,

<u>2nd (right) singular vector:</u>

direction of maximal variance, after removing the projection of the data along the first singular vector.

Singular values, intuition



σ_1: measures how much of the data variance is explained by the first singular vector.

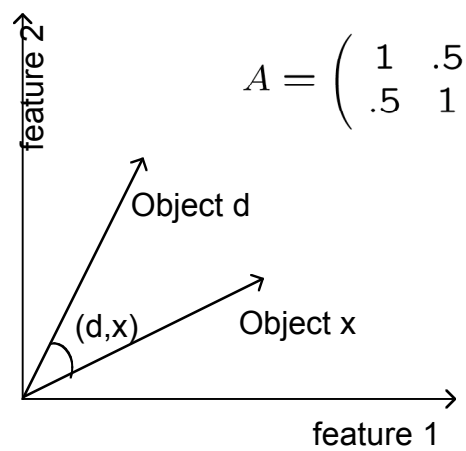
σ_2: measures how much of the data variance is explained by the second singular vector.

A first use of the SVD in data analysis

Common to *model the data* as points in a vector space -- this gives a matrix, with *m* rows (one for each object) and *n* columns (one for each feature).

Matrix rows: points (vectors) in a Euclidean space, e.g., given 2 objects (*x* & *d*), each described with respect to two features, we get a 2-by-2 matrix.

$$A = \begin{pmatrix} 1 & .5 \\ .5 & 1 \end{pmatrix}$$



Common assumption: Two objects are "close" if angle between their corresponding vectors is "small."

Common hope: k « m,n directions are important -- e.g., A_k captures most of the "information" and/or is "discriminative" for classification, etc tasks.

LSI: A_k for document-term "matrices"

(Berry, Dumais, and O'Brien '92)

Latent Semantic Indexing (LSI)

Replace A by A_k; apply clustering/classification algorithms on A_k.

n terms (words)

m documents

$$A$$

A_{ij} = frequency of j-th term in i-th document

Pros

- Less storage for small k.

$O(km+kn)$ vs. $O(mn)$

- Improved performance.

Documents are represented in a "concept" space.

Cons

- A_k destroys sparsity.
- Interpretation is difficult.
- Choosing a good k is tough.

- Sometimes people interpret document corpus in terms of k topics when use this.
- Better to think of this as just selecting one model from a parameterized class of models!

LSI/SVD and heavy-tailed data

Theorem: (Mihail and Papadimitriou, 2002)

The largest eigenvalues of the adjacency matrix of a graph with power-law distributed degrees are also power-law distributed.

- I.e., heterogeneity (e.g., heavy-tails over degrees) plus noise (e.g., random graph) implies heavy tail over eigenvalues.
- Idea: 10 components may give 10% of mass/information, but to get 20%, you need 100, and to get 30% you need 1000, etc; i.e., no scale at which you get most of the information
- No "latent" semantics without preprocessing.

Singular-stuff and eigen-stuff

If A is any $m \times n$ matrix:

$A = U \Sigma V^T$ (the SVD - general eigen-systems can be non-robust and hard to work with)

A is diagonal in orthogonal U and V basis; and Σ nonnegative

If A is any $m \times m$ square matrix:

$A = U \Lambda U^T$ (the eigen-decomposition - of course, A also has an SVD)

A is diagonal in orthogonal U basis; but Λ is not nonnegative

If A is any $m \times m$ SPSD (i.e., correlation) matrix:

$A = U \Sigma U^T$ (SVD = eigen-decomposition)

A is diagonal in orthogonal U basis; and Σ nonnegative

In data analysis, structural properties of SVD are used most often via square (e.g., adjacency) or SPSD (e.g., kernel or Laplacian) matrices

Algorithmic Issues with the SVD

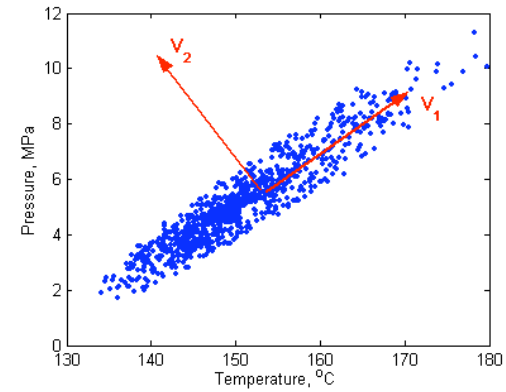
A big area with a lot of subtleties:

- "Exact" computation of the *full SVD** takes $O(\min\{mn^2, m^2n\})$ time.
- The top k left/right singular vectors/values can be computed faster using *iterative* Lanczos/Arnoldi methods.
- Specialized numerical methods for very large sparse matrices.
- A lot of work in TCS, NLA, etc on randomized algorithms and ε-approximation algorithms (for $\varepsilon \approx 0.1$ or $\varepsilon \approx 10^{-16}$).

*Given the full SVD, you can do "everything." But *you "never" need the full SVD*. Just compute what you need!

PCA and MDS

Principal Components Analysis (PCA)



- Given $\{X_i\}_{i=1,\ldots,n}$ with $X_i \in R^D$,

  Find k-dimensional subspace P and embedding $Y_i = PX_i$

  s.t. Variance(Y) is maximized or Error(Y) is minimized

- Do SVD on covariance matrix $C = XX^T$

Multidimensional Scaling (MDS)

- Given $\{X_i\}_{i=1,\ldots,n}$ with $X_i \in R^D$,

  Find k-dimensional subspace P and embedding $Y_i = PX_i$

  s.t. $\mathrm{Dist}(Y_i - Y_j) \approx \mathrm{Dist}(X_i - X_j)$, i.e., dot products (or distances) preserved

- Do SVD on Gram matrix $G = X^T X$

SVD is the structural basis behind PCA, MDS, Factor Analysis, etc.

Statistical Aspects of the SVD

Can always compute best rank-k SVD approximation

- in "nice" Gaussian settings, corresponding statistical interpretation
- more generally, model selection in a place with nice geometry

Least-squares regression and PCA

- optimal (in terms of mean squared error) linear compression scheme for compressing and reconstructing any high-dimensional vectors
- if the data were generated from Guassian distributions, then it is the "right thing to do"
- several related ways to formalize these ideas

Geometric Aspects of the SVD

Can always compute best rank-k SVD approximation

- in "nice" Gaussian settings, corresponding statistical interpretation
- more generally, model selection in a place with nice geometry

Least-squares regression and PCA

- embed the data in a line or low-dimensional hyperplane
- reconstruct clusters when data consist of "separated" Gaussians
- geometry permits Nystrom-based and other out-of-sample schemes and "robustness" due to constraints imposed by low-dimensional space
- several related ways to formalize these ideas

These are a *very* strong properties

Contrast these properties with tensors*

- Computing the rank of a tensor (*qua* tensor) is intractable, and best rank k approximation may not even exist
- Many other strong hardness results (Lim 2006)
- Researchers "fall back" on matrices along each mode

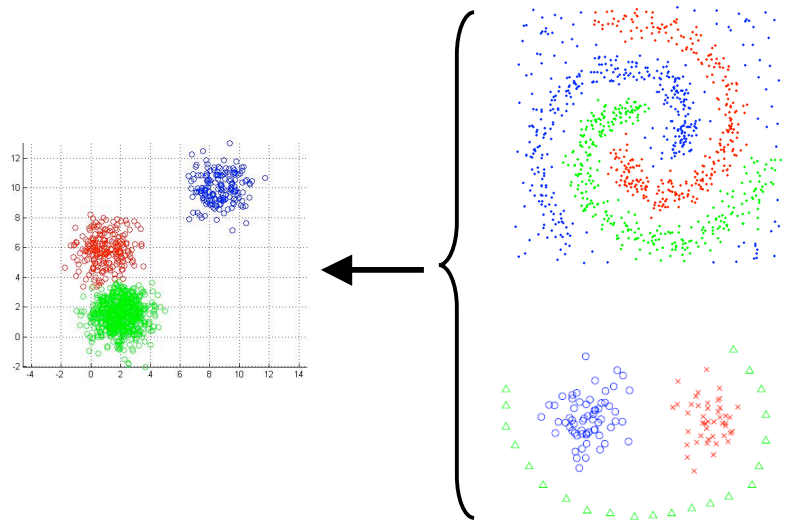
That matrices are so nice is the exception, not the rule, among algebraic structures---vector spaces are *very* structured places, with associated benefits and limitations.

*Tensors are another algebraic structure used to model data: Think of them as A_{ijk}, i.e., matrices with an additional subscript, where multiplication is linear along each "direction"

Kernel Methods

Many algorithms access data only through elements of Correlation or Gram matrix.

$$< x_i, x_j > \rightarrow K(x_i, x_j)$$



- Can use another SPSD matrix and to encode nearness information.
- Many learning bounds generalize
- E.g., $K(x_i,x_j) = f(||x_i-x_j||)$, Gaussian r.b.f., polynomial kernels, etc - good but limited
- Data-dependent kernels - operationally define a kernel on graph constructed from point cloud data; typically viewed as implicitly defining a *manifold*

Kernels and linear methods

Kernel methods are basically linear methods in some other feature space that is non-linearly related to the original representation of the data:

- Good news: still linear (classify with hyperplanes, have capacity control since hyperplanes are structured objects, etc.)
- Bad news: still linear (so still boiling down to SVD); determining features is an art; very hard to deal with very non-linear metrics

Kernel methods basically give you a lot more statistical (or descriptive) flexibility without too much additional computational cost.

Data-dependent kernels, cont.

ISOMAP:

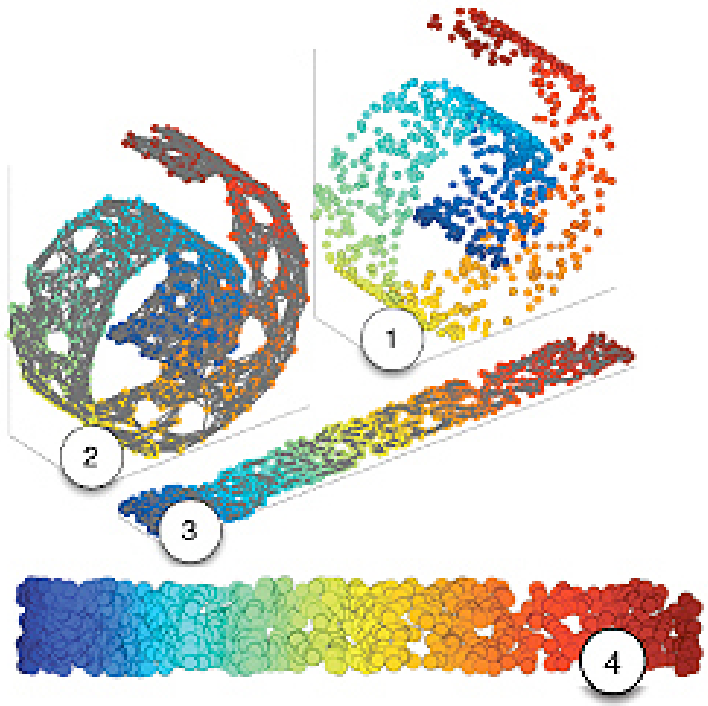
- Compute geodesics on adjacency graph
- MetricMDS gives k eigenvectors for embedding

LLE:

- Compute edge weights from local least-squares approximation
- Compute global embedding vectors as bottom k+1 eigenvectors of a matrix

Laplacian eigenmaps:

- Assign edge weights $W_{ij} = \exp(-\beta||x_i-x_j||_2^2)$
- Compute embedding vectors as bottom k+1 eigenvectors of Laplacian



Kernels and Manifolds and Diffusions

Laplacian Eigenmaps:

- Defined on graphs, but close connections to "analysis on manifolds"

Laplacian in R^d: $$\Delta f = -\sum_i \frac{\partial^2 f}{\partial^2 x_i}$$

Manifold Laplacian

- measure change along tangent space of manifold

Connections with diffusions (and Markov chains):

$$\frac{\partial \Psi(t)}{\partial t} = -L\Psi(t)$$

$$K_{(t)} = \exp(-Lt) \ (\sim \text{Green's function})$$

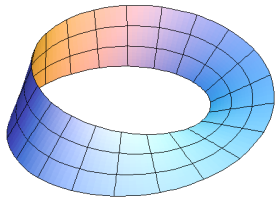
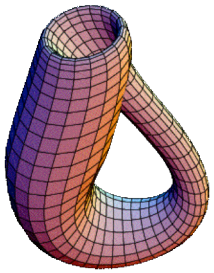
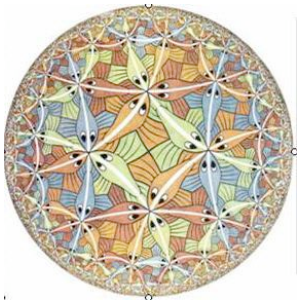
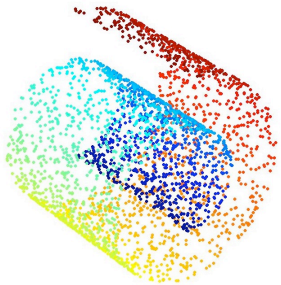
$$\Psi(t) = K_{(t)}\Psi(0)$$

$$K_{(t)} \sim \frac{1}{2}L^{+} \ (\text{under "nice" sssumptions})$$

What is a manifold?

A topological manifold is a topological space which locally looks Euclidean in a certain (weak) sense

A Riemannian manifold is a *differentiable* manifold in which the tangent space is R^n. (Tangent space has *inner product* that varies smoothly and that gives lengths, angles, areas, gradients, etc.)

Barring "pathological" curvature or density behavior, i.e., permitting a ***huge*** *amount of descriptive flexibility, think of a ML manifolds as a "curved" low-dimensional space.*

Kernels and learning a manifold

Practice and Theory:

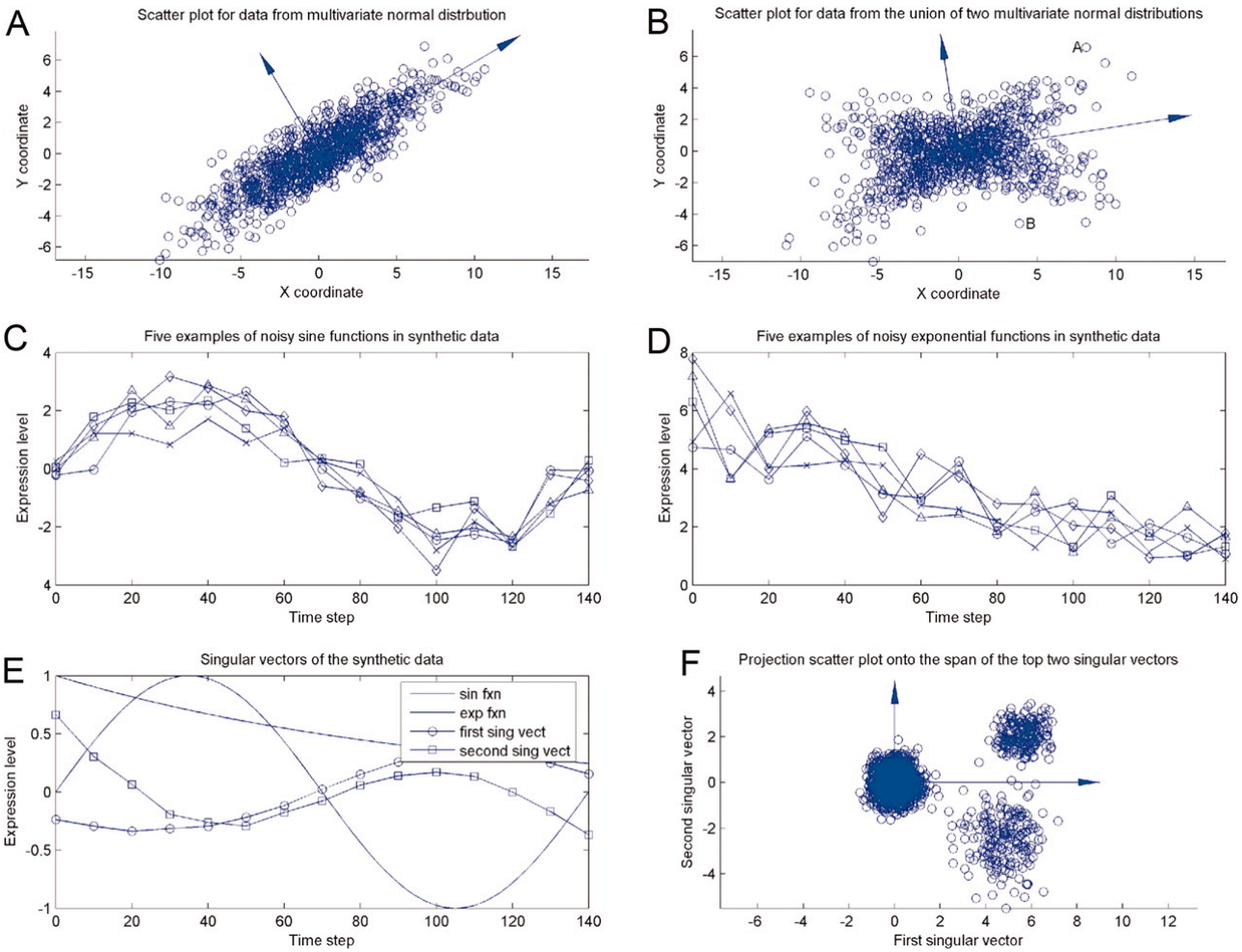
- Choose kernel, and see if eigen-methods give good visualization, clustering, etc.
- Thm: If the hypothesized manifold and sampling density are "nice," then L_{graph} will converge to $L_{manifold}$.

Manifold learning is *not* of classification, clustering, regression; but of the hypothesized manifold

- Empirically (or theoretically) useful when two large clusters
- Basically, "exploratory" data modeling, using one class of models

Interpreting the SVD - be *very* careful

Mahoney and Drineas (PNAS, 2009)

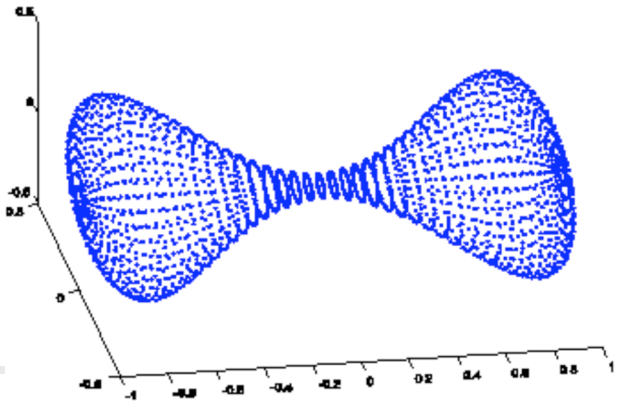


Reification

- assigning a "physical reality" to large singular directions
- invalid in general

Just because "If the data are 'nice' then SVD is appropriate" does NOT imply converse.

Interpretation: Centrality



Centrality (of a vertex) - measures relative importance of a vertices in a graph

- degree centrality - number of links incident upon a node
- betweenness centrality - high for vertices that occur on many shortest paths
- closeness centrality - mean geodesic distance between a vertex and other reachable nodes
- eigenvector centrality - connections to high-degree nodes are more important, and so on iteratively (a "spectral ranking" measure)

Motivation and behavior on nice graphs is clear -- but what do they actually compute on non-nice graphs?

Eigen-methods in ML and data analysis

Eigen-tools appear (*explicitly* or *implicitly**) in many data analysis and machine learning tools:

- Latent semantic indexing
- Manifold-based ML methods
- Diffusion-based methods
- k-means clustering
- Spectral partitioning and spectral ranking

*What are the limitations imposed when these methods are implicitly used? Can we get around those limitations with complementary methods?

k-means clustering

(Drineas, Frieze, Kannan, Vempala, and Vinay '99; Boutsidis, Mahoney, and Drineas '09)

k-means clustering

A standard objective function that measures cluster quality.

(Often denotes an iterative algorithm that attempts to optimize the *k*-means objective function.)

k-means objective

Input: set of m points in R^n, positive integer k

Output: a partition of the m points to k clusters

Partition the m points to k clusters in order to minimize the sum of the squared Euclidean distances from each point to its cluster centroid.