Fast Display of Massive, High-Dimensional Data

Alexander Gray

Georgia Institute of Technology

www.fast-lab.org

The FASTlab

Fundamental **A**lgorithmic and **S**tatistical **T**ools Laboratory

www.fast-lab.org

1. Arkadas Ozakin: **Research scientist**, Math, Physics; PhD Physics
2. Nikolaos Vasiloglou: **Visiting scholar**, EE; PhD EE
3. Abhimanyu Aditya: **Affiliate**, CS; MS CS

4. Dong Ryeol Lee: **PhD student**, CS + Math
5. Ryan Riegel: **PhD student**, CS + Math
6. Sooraj Bhat: **PhD student**, CS
7. Wei Guan: **PhD student**, CS
8. Nishant Mehta: **PhD student**, CS
9. Parikshit Ram: **PhD student**, CS + Math
10. William March: **PhD student**, Math + CS
11. Hua Ouyang: **PhD student**, CS
12. Ravi Sastry: **PhD student**, CS
13. Long Tran: **PhD student**, CS
14. Ryan Curtin: **PhD student**, EE
15. Ailar Javadi: **PhD student**, EE

+ 5-10 **MS students**

Goal: Seeing the Unseeable

Given a collection of data objects, can we show their relationships in a 2-d plot?

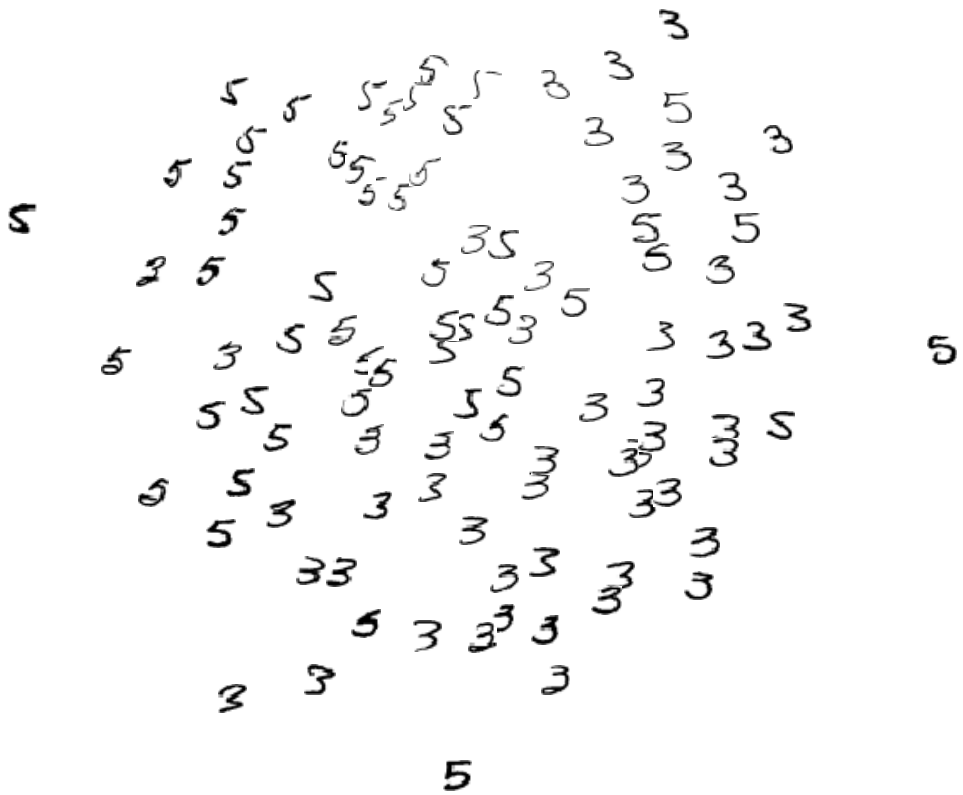


Figure: Recovered locations of USPS handwritten digits "3" and "5" given by RankMap. The original data are in $16 \times 16 = 256$ dimensions.

Basic Approach

1. Choose a notion of distance that defines the relationships between the objects (defines a graph or kernel matrix, and choose **something you want to preserve** about those relationships
2. Perform a **computation** (graph construction + *convex* optimization) that defines the relationship-preserving mapping to 2-d points

Nonlinear Dimension Reduction: Key Bottlenecks

- **Isomap:** all-pairs shortest paths, all-nearest-neighbors, SVD
- **Locally linear embedding:** all-nearest-neighbors, SVD
- **Maximum variance unfolding:** all-nearest-neighbors, SDP
- **Rankmap (Ouyang and Gray 2008, ICML):** (all-nearest-neighbors), SDP/QP
- **Isometric non-neg matrix fac (Vasiloglou, Gray, and Anderson 2009, SDM):** all-nearest-neighbors, SDP
- **Isometric separation maps (Vasiloglou, Gray, and Anderson 2009, MLSP):** all-nearest-neighbors, SDP
- **Density-preserving maps (Ozakin and Gray, in prep):** kernel summation, SDP

What about "supervised dimension reduction"?

- **Sparse support vector machines:** LP/QP/DC/MINLP, kernel summation

Sidebar: Forget about Preserving Distances?

A theorem that goes back to Gauss and Riemann implies that it is impossible to preserve the intrinsic distances between points in an intrinsically curved d-dimensional space by representing the points in d-dimensional Euclidean space. A familiar instance of this theorem is the fact that it is impossible to preserve all the distances between the points on the surface of the Earth by representing them on a flat map. Although seemingly (or "extrinsically") curved, surfaces such as the Swiss roll are intrinsically flat; however, a sphere is intrinsically curved. Various manifold learning methods, when faced with data on such a curved space, distort the data upon performing the dimensional reduction in various ways. In other words: **Are distances the right things to preserve at all?**

How About Preserving Densities?

Ozakin, Vasiloglou and Gray, in prep: You can't always preserve distances, but you *can* always preserve densities:

Theorem

Let $(M, \mathbf{g}_M)$ and $(N, \mathbf{g}_N)$ be two closed, connected, d-dimensional Riemannian manifolds, diffeomorphic to each other, with the same total Riemannian volume. Let X be a random variable on M, i.e., a measurable map $X : \Omega \to M$ from a probability space $(\Omega, \mathcal{F}, P)$ to M. Assume that $X_*(P)$, the pushforward measure of P by X, is absolutely continuous with respect to the Riemannian volume measure μ_M on M, with a continuous density f on M. Then there exists a diffeomorphism $\phi : M \to N$ such that the pushforward measure $P_N := \phi_*(X_*(P))$ is absolutely continuous with respect to the Riemannian volume measure μ_N on N, and the density of P_N is given by $f \circ \phi^{-1}$.

Density-Preserving Maps: SDP

Density-Preserving Maps (Ozakin, Vasiloglou, and Gray, in prep):

$$
\begin{aligned}
&\max_{K} \operatorname{trace}(K) \\
&\quad \text{such that:} \\
&\qquad \hat{f}_i = \frac{N_e}{h_i^d} \sum_{j \in I_i} \epsilon_{ij} \\
&\qquad \epsilon_{ij} = (1 - d_{ij}^2 / h_i^2) \\
&\qquad d_{ij}^2 = K_{ii} + K_{jj} - K_{ij} - K_{ji} \\
&\qquad K \succeq 0 \\
&\qquad \epsilon_{ij} \geq 0 \\
&\sum_{i,j=1}^{n} K_{ij} = 0
\end{aligned}
$$

Estimating High-Dimensional Densities

Submanifold Kernel Density Estimation (Ozakin and Gray 2009, NIPS) can perform high-dimensional nonparametric density estimation, if the data are on a manifold M.

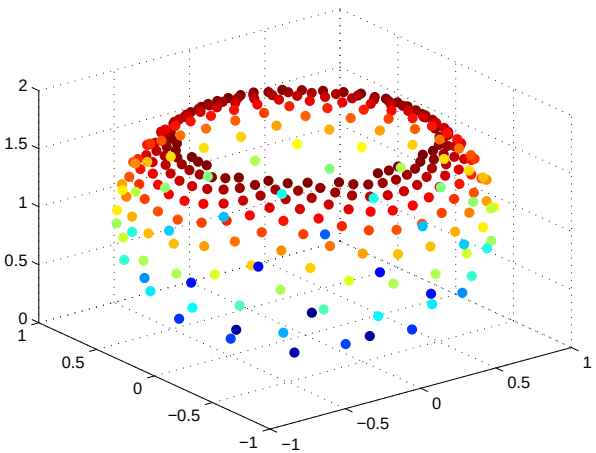
Theorem

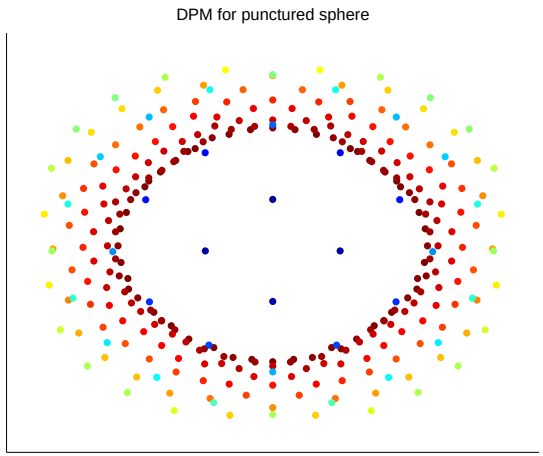
Let $f : M \to [0,\infty)$ be a probability density function defined on M (so that the related probability measure is fV), and $K : [0,\infty) \to [0,\infty)$ be a continous function that satisfies vanishes outside $[0,1)$, is differentiable with a bounded derivative in $[0,1)$, and satisfies, $\int_{\|\mathbf{z}\|\leq 1} K(\|\mathbf{z}\|)d^n\mathbf{z} = 1$. Assume f is differentiable to second order in a neighborhood of $p \in M$, and for a sample $q_1, \ldots, q_m$ of size m drawn from the density f, define an estimator $\hat{f}_m(p)$ of $f(p)$ as, $\hat{f}_m(p) = \frac{1}{m}\sum_{j=1}^{m} \frac{1}{h_m^n} K\left(\frac{u_p(q_j)}{h_m}\right)$ where $h_m > 0$. If h_m satisfies $\lim_{m\to\infty} h_m = 0$ and $\lim_{m\to\infty} m h_m^n = \infty$, then, there exists non-negative numbers m_*, C_b, and C_V such that for all $m > m_*$ we have,

$$\mathrm{MSE}\left[\hat{f}_m(p)\right] = \mathrm{E}\left[\left(\hat{f}_m(p) - f(p)\right)^2\right] < C_b h_m^4 + \frac{C_V}{m h_m^n}. \qquad (1)$$

If h_m is chosen to be proportional to $m^{-1/(n+4)}$, this gives,
$\mathrm{E}\left[(f_m(p) - f(p))^2\right] = O\left(\frac{1}{m^{4/(n+4)}}\right)$ *as* $m \to \infty$.

Density-Preserving Maps: Example





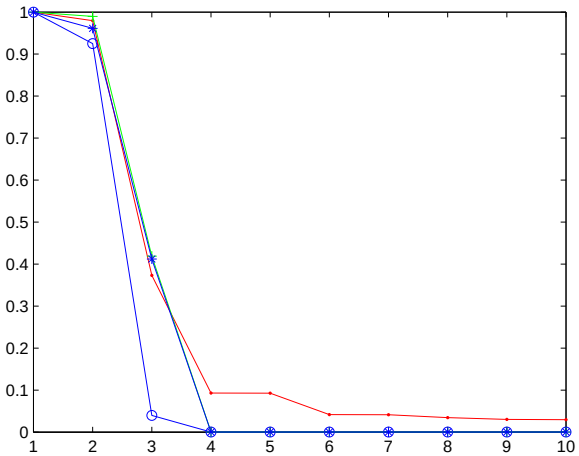


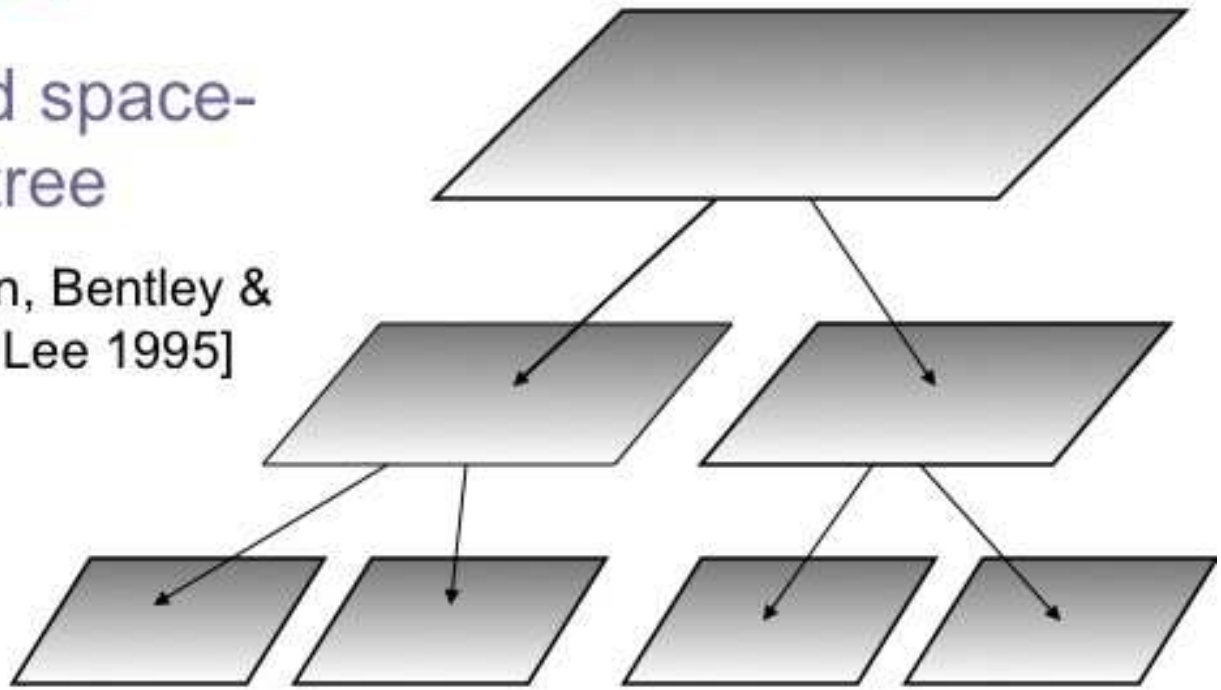
Figure: a) A punctured sphere data set. b) The data reduced by density preserving maps. c) The eigenvalue spectra of the inner product matrices learned by PCA (green, '+'), Isomap (red, '.'), MVU (blue, '*'), and DPM (blue, 'o').

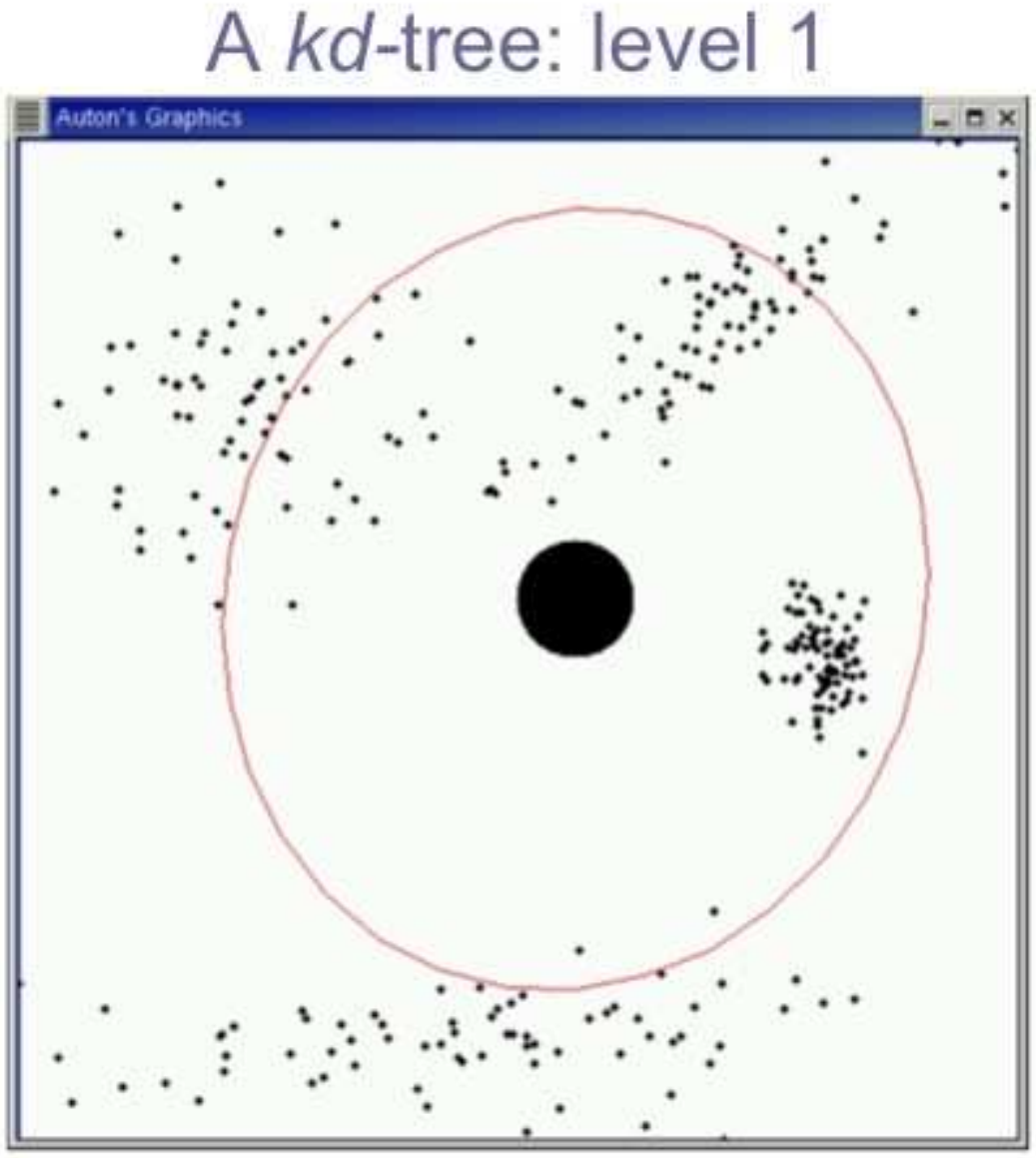
Idea: Break up the problem....

***kd*-trees:**

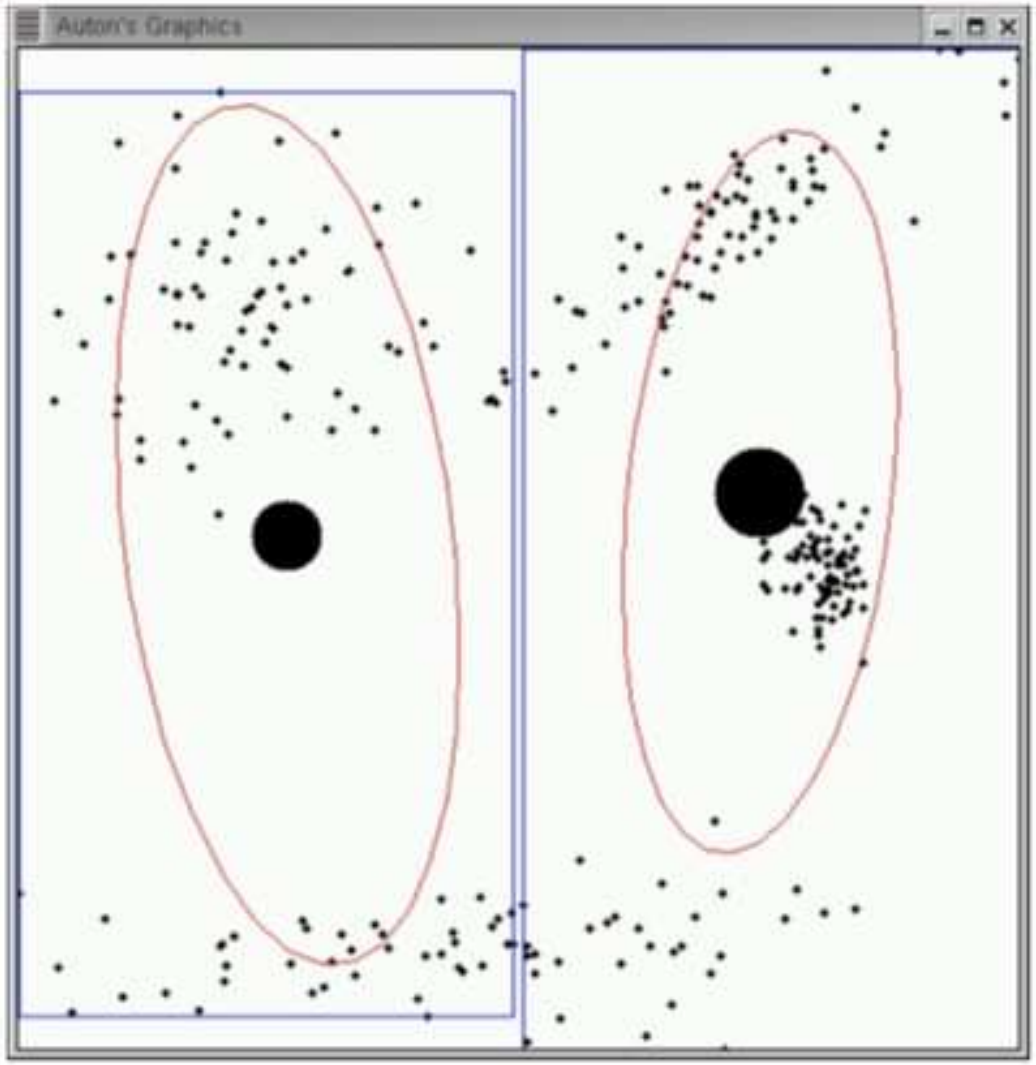
most widely-used space-partitioning tree

[Bentley 1975], [Friedman, Bentley & Finkel 1977],[Moore & Lee 1995]

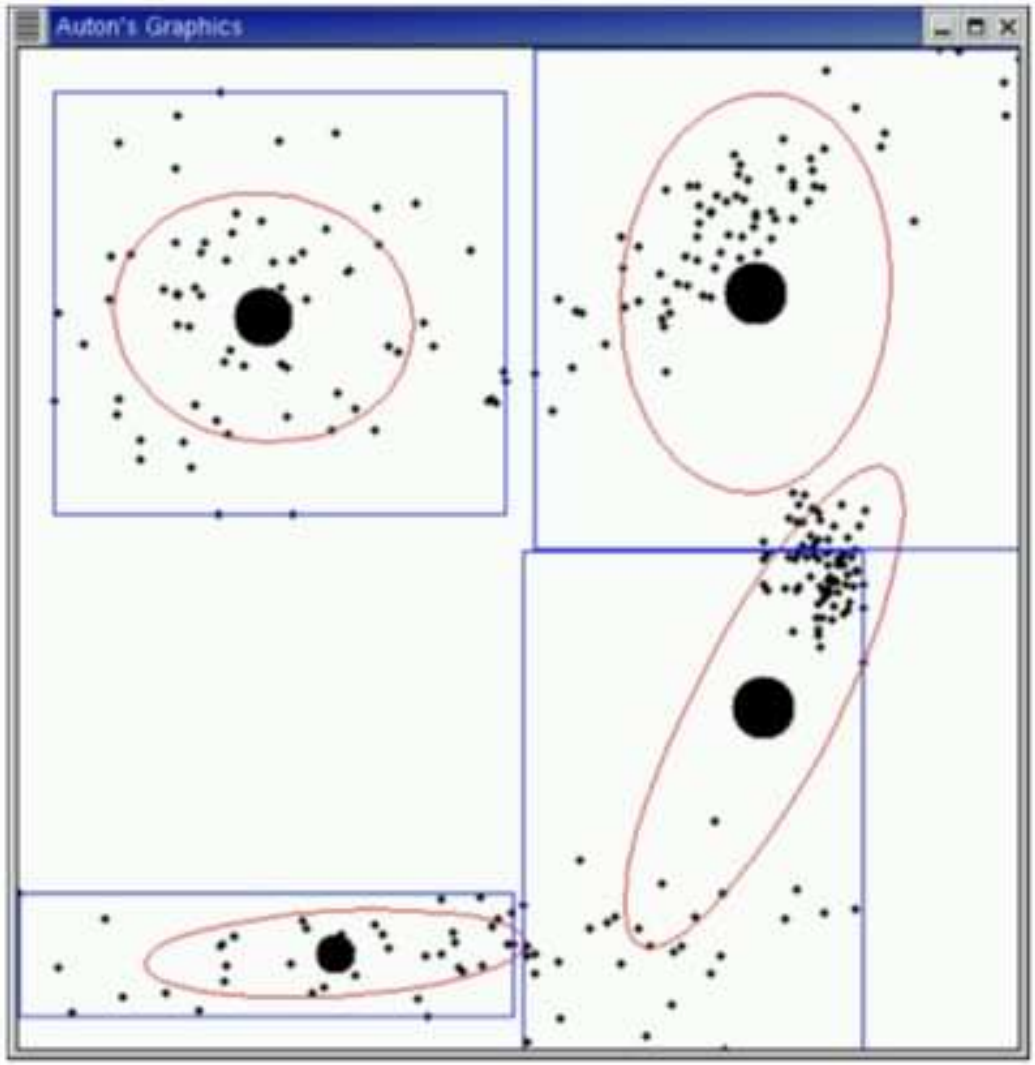



A kd-tree: level 1
Auton's Graphics

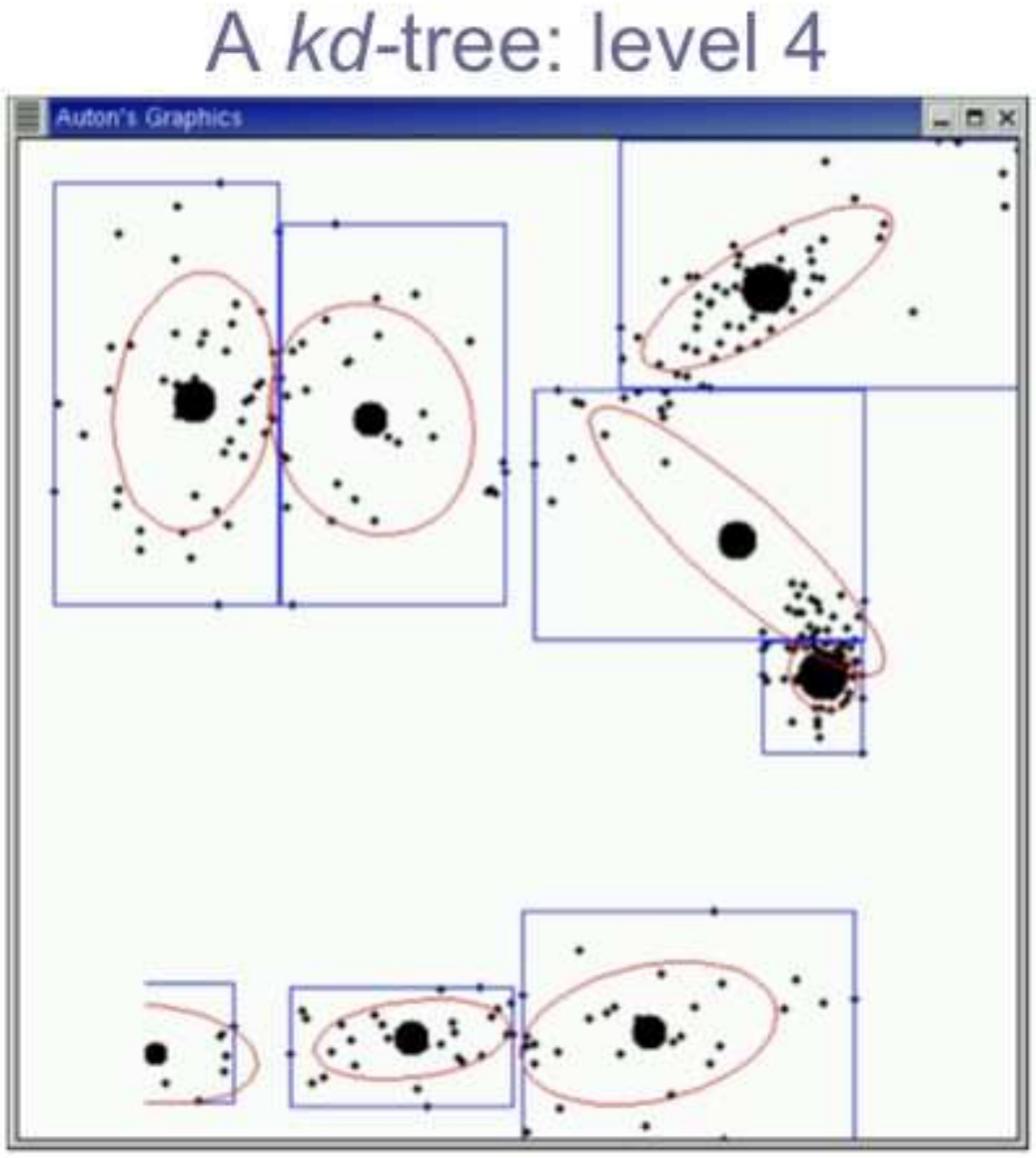
A *kd*-tree: level 2



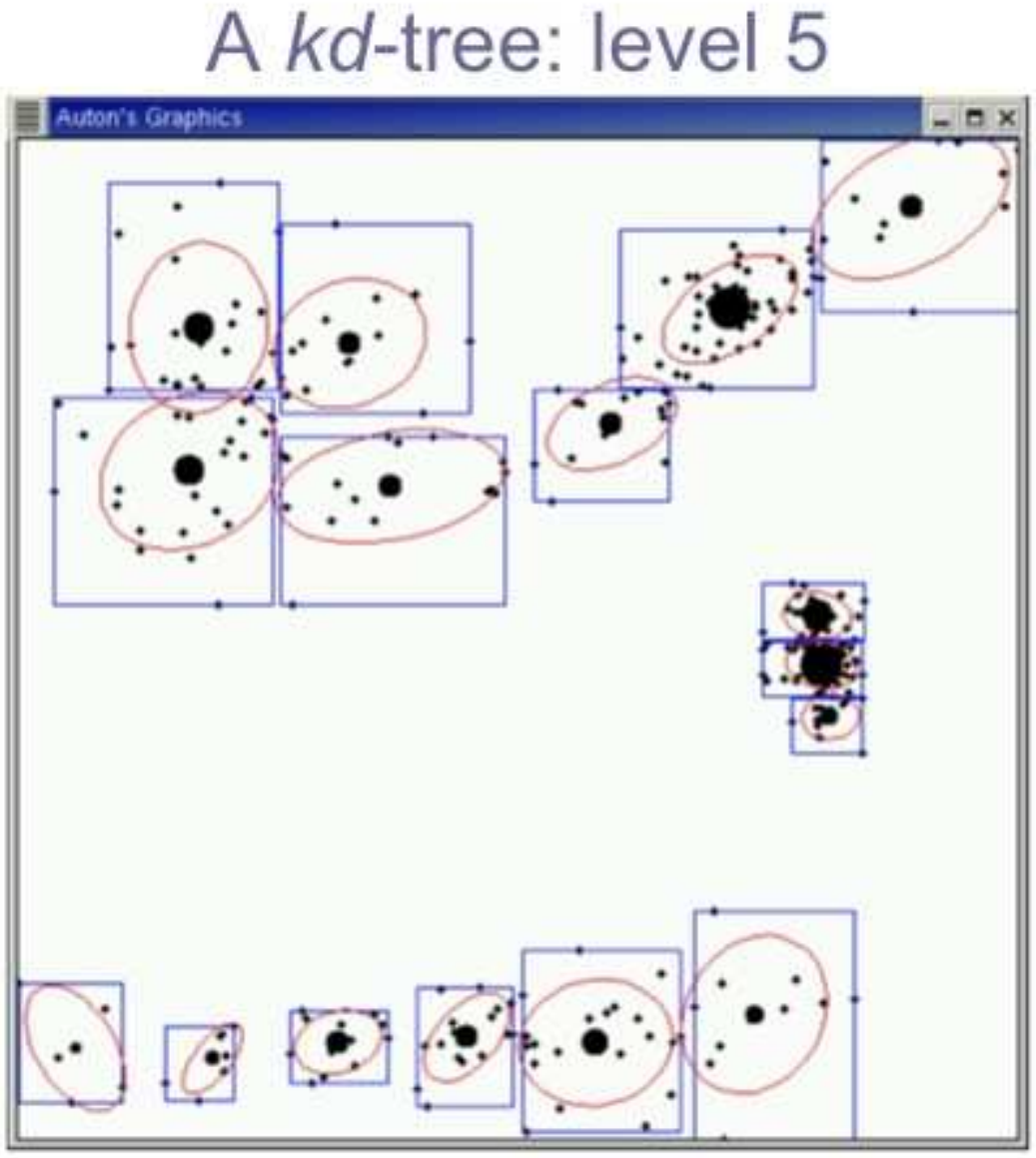
A *kd*-tree: level 3



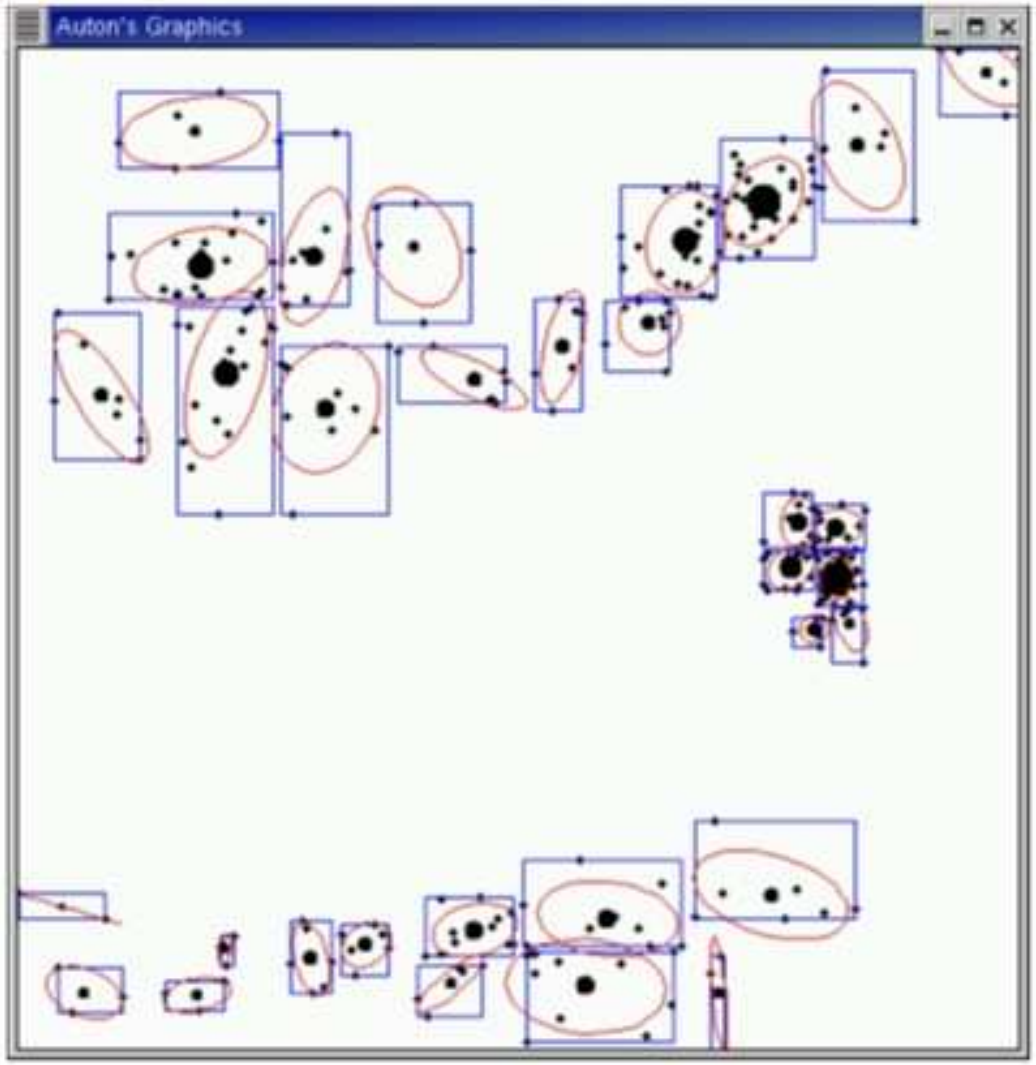
kD-trees



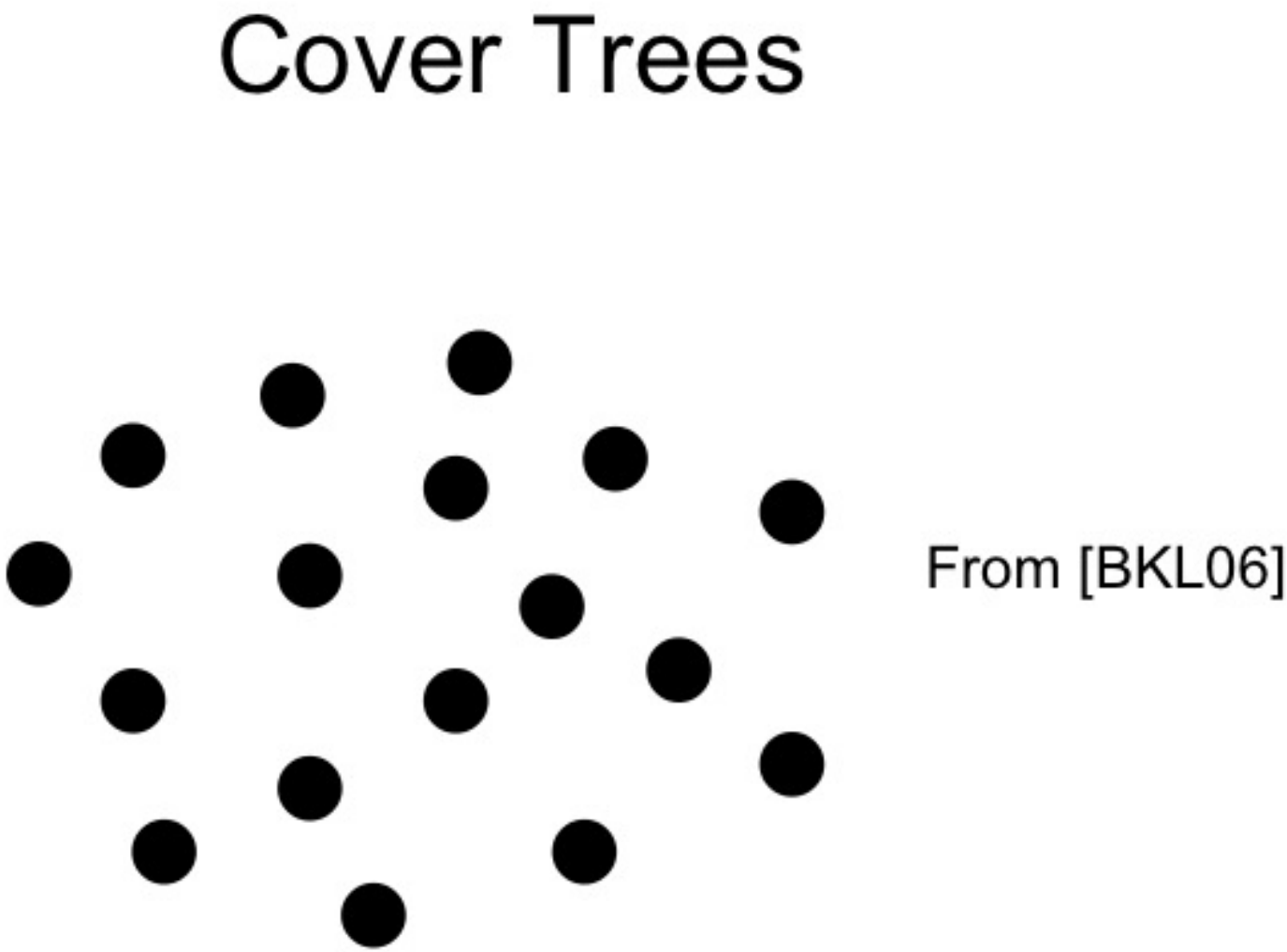
kD-trees



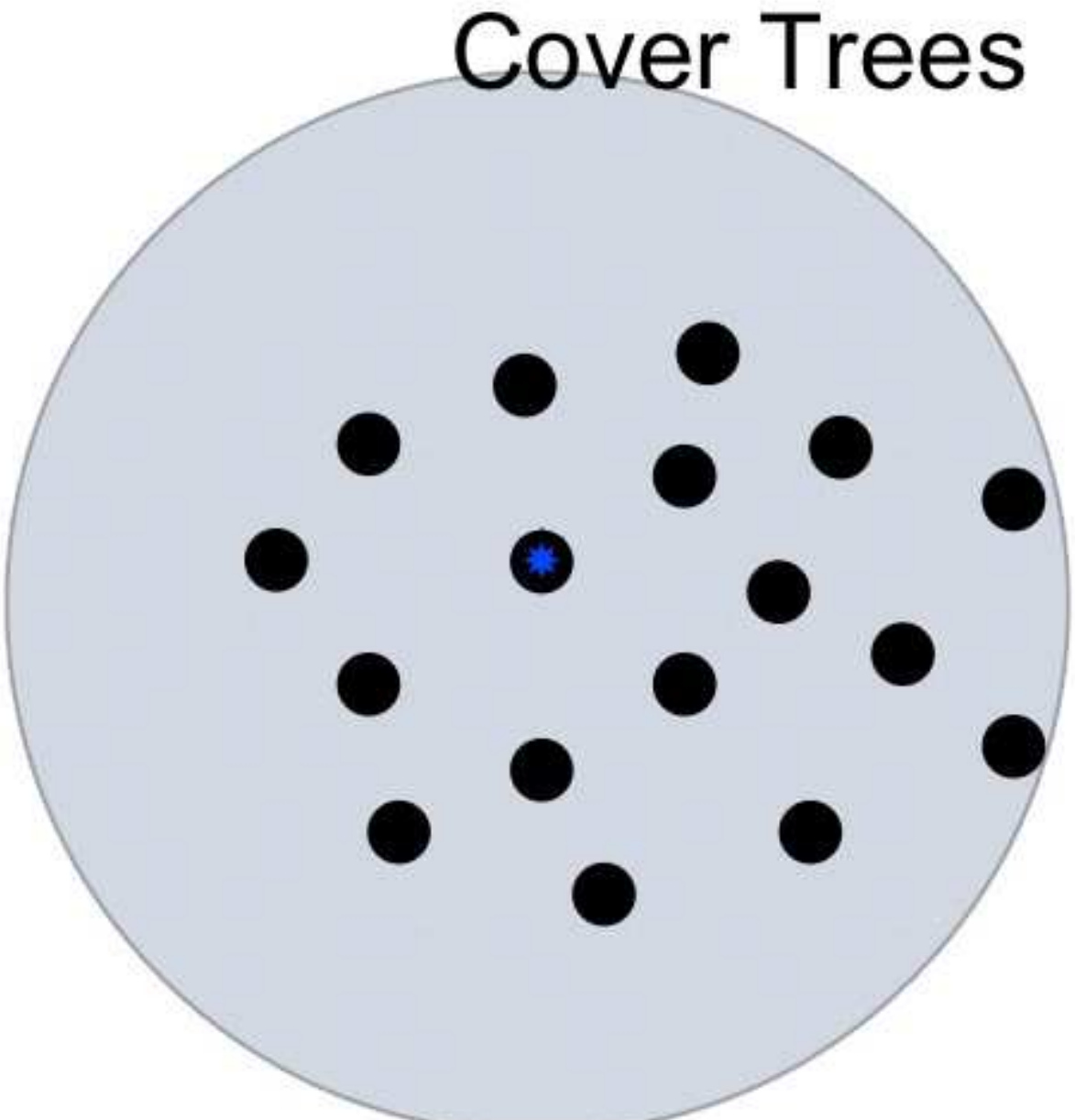
A *kd*-tree: level 6



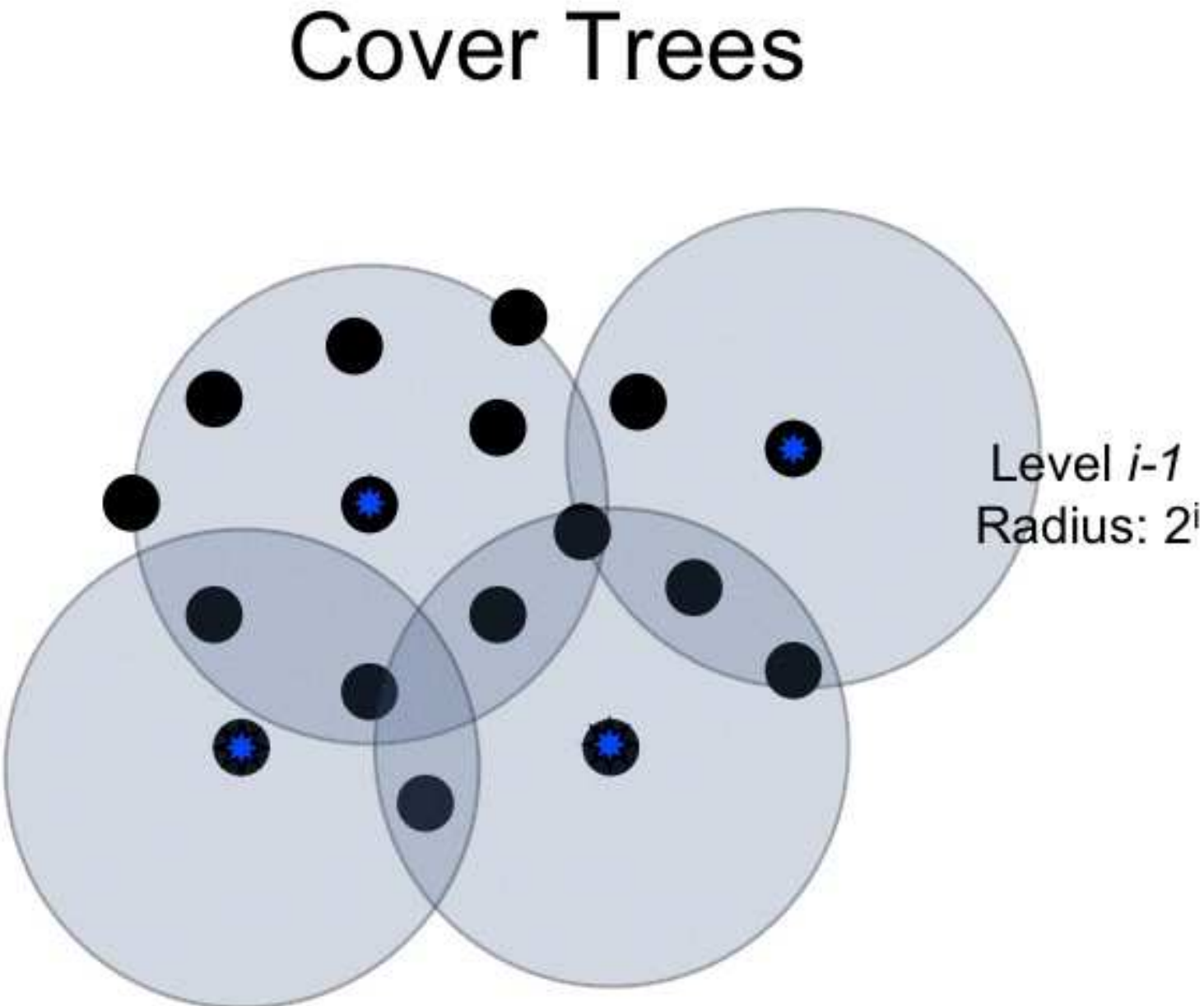
Cover trees



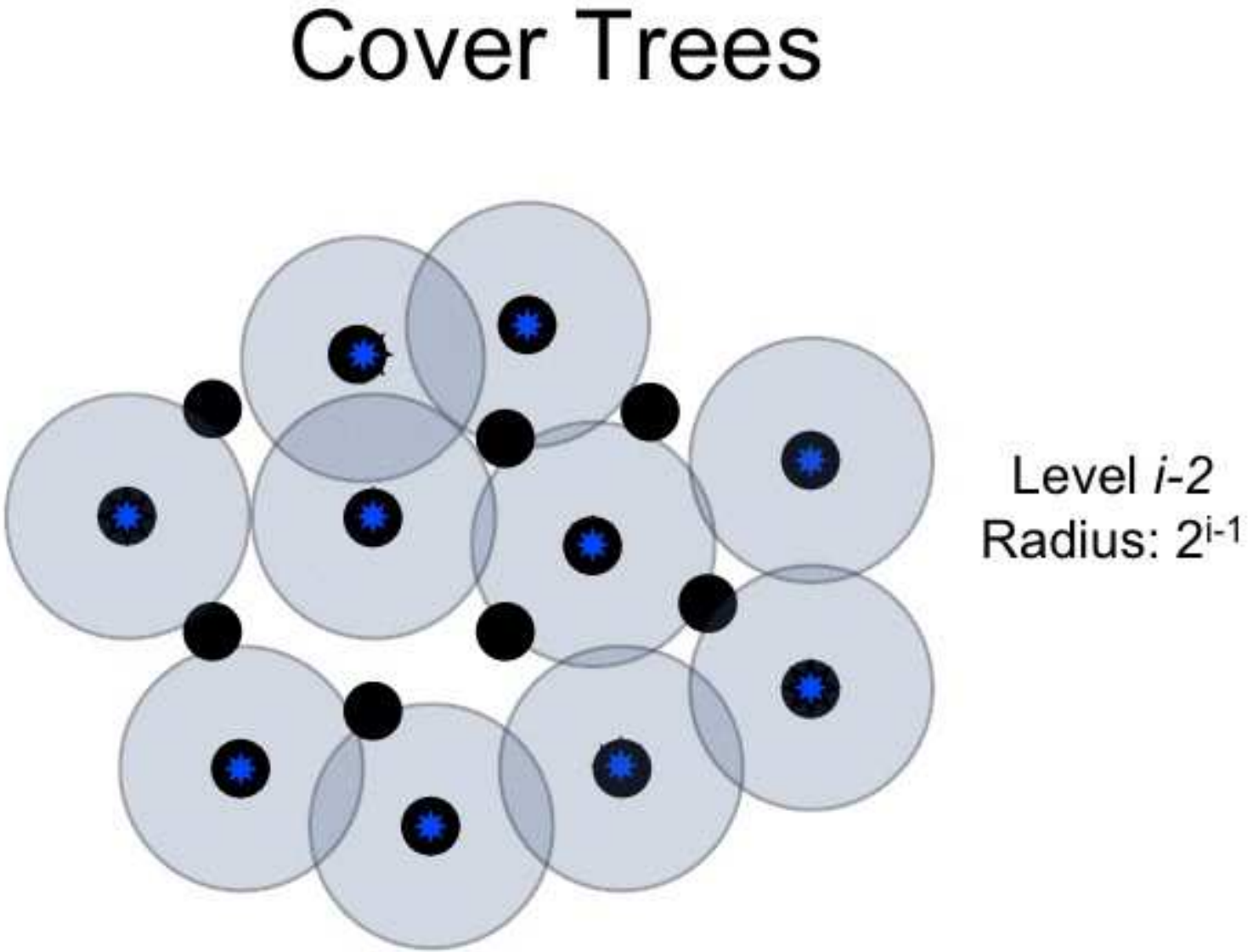
Cover trees

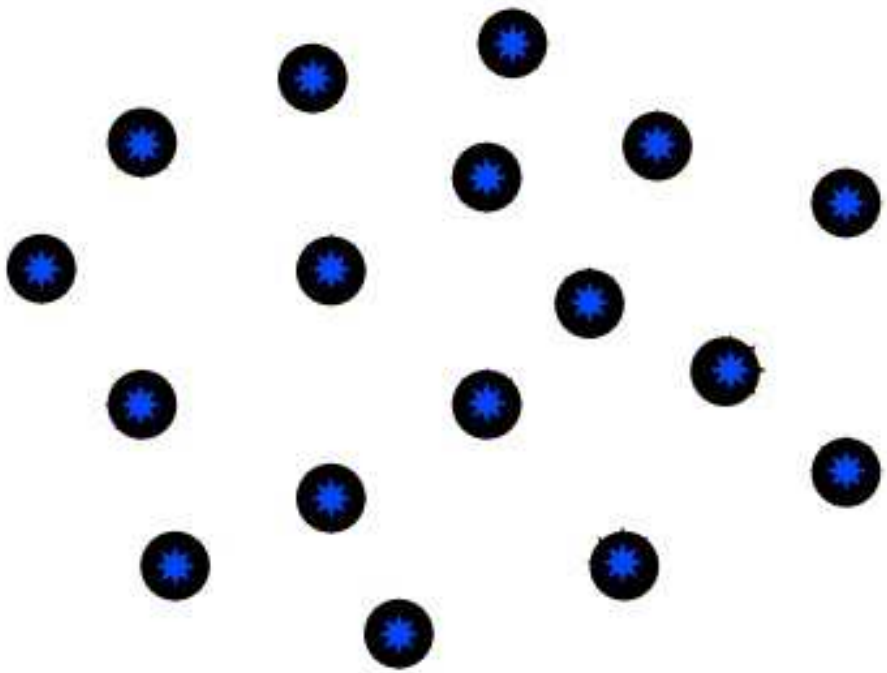


Cover trees



Cover trees




Cover Trees
Space: O(N) nodes
Construction: O(N log N)

Fast All-Nearest-Neighbors

Common graph: Use k-nearest-neighbors. Problem: This is $O(N^2)$ naively. Often graph construction is the most expensive step of manifold learning.

Fast solution: 1) Use space-partitioning trees, such as kd-trees or cover-trees. This is the fastest approach for exact single-query searches. 2) Traverse two simultaneously for the greatest speed in *all*-nearest-neighbor searches via **dual-tree algorithms (Gray and Moore 2000, NIPS)**.

Analysis: Shown to be $O(N)$, or **linear-time**, in general bichromatic case, on cover-trees **(Ram, Lee, March, and Gray 2009, NIPS)**.

Fast Nearest-Neighbor in High Dimensions

In general the original data are high-dimensional. Problem: A curse of dimensionality (Hammersley 1950) says that distances approach the same numerical value as dimension goes up. This makes tree algorithms ineffective in very high dimensions.

A recent trend has been to approximate nearest-neighbor by returning a point within $(1 + \epsilon)$ of the true nearest-neighbor distance with high probability, e.g. LSH (Andoni and Indyk, 2006). Problem: In high dimensions, all points could satisfy this criterion, making the results junk.

More accurate solution: **rank-approximate** nearest-neighbor **(Ram, Lee, Ouyang, and Gray 2009, NIPS)**, which is in general faster and more accurate than LSH.

Fast Nearest-Neighbor in High Dimensions

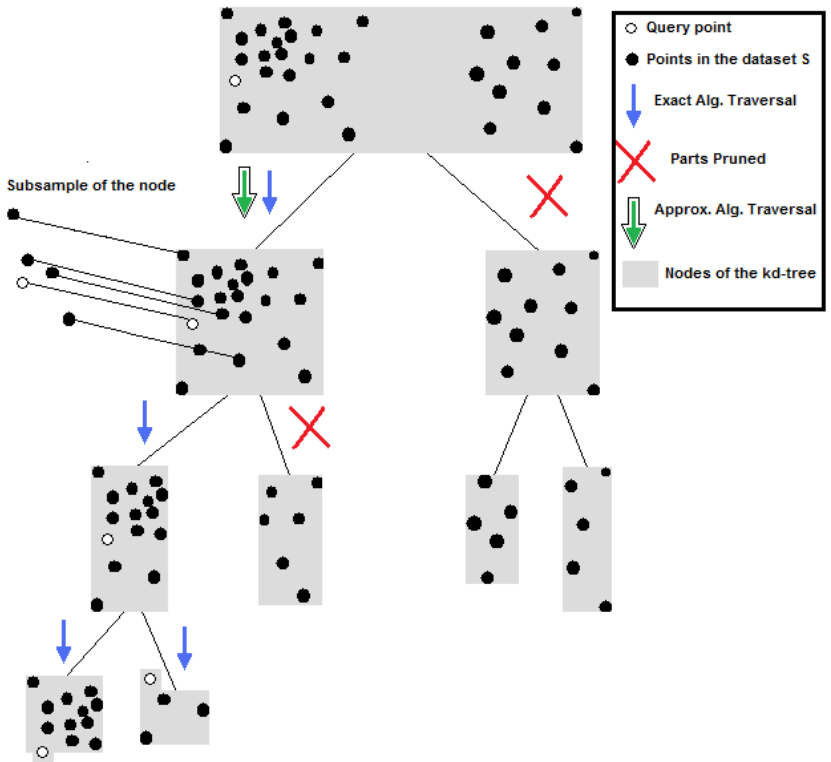


Figure: The traversal paths of the exact and the rank-approximate algorithm in a *kd*-tree.

Fast Nearest-Neighbor in High Dimensions

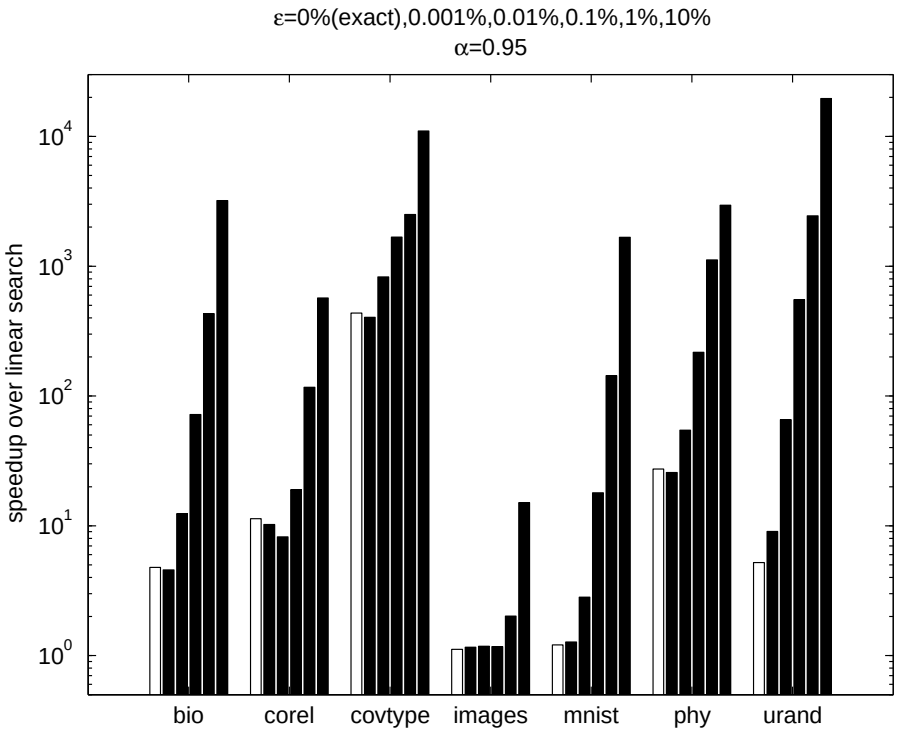


Figure: Speedups (logscale on the Y-axis) over the linear search algorithm while finding the NN in the exact case or $(1 + \varepsilon N)$-RANN in the approximate case with $\varepsilon = 0.001\%, 0.01\%, 0.1\%, 1.0\%, 10.0\%$ and a fixed success probability $\alpha = 0.95$ for every point in the dataset. The first(white) bar in each dataset in the X-axis is the speedup of exact dual tree NN algorithm, and the subsequent(dark) bars are the speedups of the approximate algorithm with increasing approximation.

Fast Nearest-Neighbor in High Dimensions

Generally more accurate than LSH, automatically.

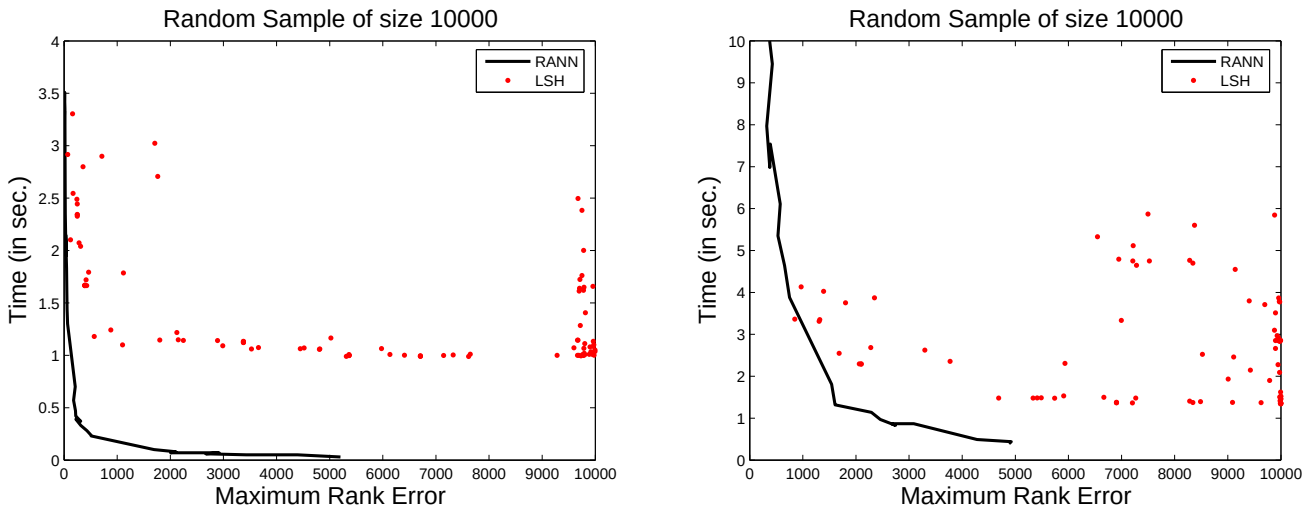


Figure: Query times on the X-axis and the Maximum Rank Error on the Y-axis. Left: Layout histogram data. Right: MNIST data.

Fast Kernel Summation

For DPM, kernel density estimation (KDE) is needed as the first step: $\forall x, \widehat{f}(x) = \frac{1}{N} \sum_i^N K_h(\|x - x_i\|)$. Problem: This is $O(N^2)$ naively.

The fastest recent methods for this problem are physics-inspired fast multipole-like methods **Lee and Gray 2006, UAI)**. Problem: These approaches require computing a number of coefficients that explodes as dimension goes up, and thus are good only in low dimensions.

Better solution for high dimensions: **Monte Carlo multipole** methods **(Lee and Gray 2008, NIPS)**, which has been shown to be effective in up to 800 dimensions, at the cost of its accuracy guarantee only holding with high probability.

Fast KDE Learning

To learn the optimal bandwidth for KDE, it must effectively be run many times, one for each bandwidth in cross-validation. Problem: This multiplies the cost significantly.

Faster solution: **multi-tree Monte Carlo** methods **(Holmes, Gray and Isbell 2008, UAI)** make this scalar sum fast, with accuracy holding with high probability.

Fast Approximate Singular Value Decomposition

For most manifold learning methods, the final computation is a singular value decomposition (SVD). Problem: This is $O(N^3)$ for an $N \times N$ matrix.

Recent approaches have applied Monte Carlo ideas to linear algebra (cf. NIPS 2009 tutorial). Problem: These methods are driven by theoretical sample complexity bounds, which are not data-dependent, and assume the rank is known.

Faster solution: A Monte Carlo approach based on **cosine trees** **(Holmes, Gray and Isbell 2008, NIPS)** samples more efficiently, and stops as soon as the original matrix is well-approximated, making it faster as well as automatic.

Fast Semidefinite Programming for Manifold Learning

The most recent manifold learning methods result in semidefinite programs (SDPs). Problem: These can be $O(N^3)$ or worse.

In **(Vasiloglou, Gray, and Anderson 2008, MLSP)** it shown how the Burer-Monteiro method, a relaxation to a non-convex problem which preserves the same optimum, in conjunction with L-BFGS, can make MVU-like methods scalable to nearly a million points.

Fast Support Vector Machines

Various SVM formulations lead to quite different optimization, and hard, problems. Fastest methods to date include:

- For L_2 **SVMs with squared hinge loss**: our **stochastic Frank-Wolfe** method for this QP (online nonlinear SVM training) **(Ouyang and Gray 2010, SDM; ASA Computational Statistics Student Paper Prize)**
- For $L_{0<q<1}$ **SVMs**: our **DC programming** method **(Guan and Gray 2010, under review)**
- For L_0 **SVMs**: our **perspective cuts** method for this MINLP **(Guan and Gray 2010, under review)**

Software

- **MLPACK (C++)**
  - First scalable comprehensive ML library
- **MLPACK-db (C#, C++)**
  - Fast data analysis in relational databases (SQL Server; on-disk)
- **Commercial (email me if interested)**
  - Very-large-scale data (parallel, streaming)