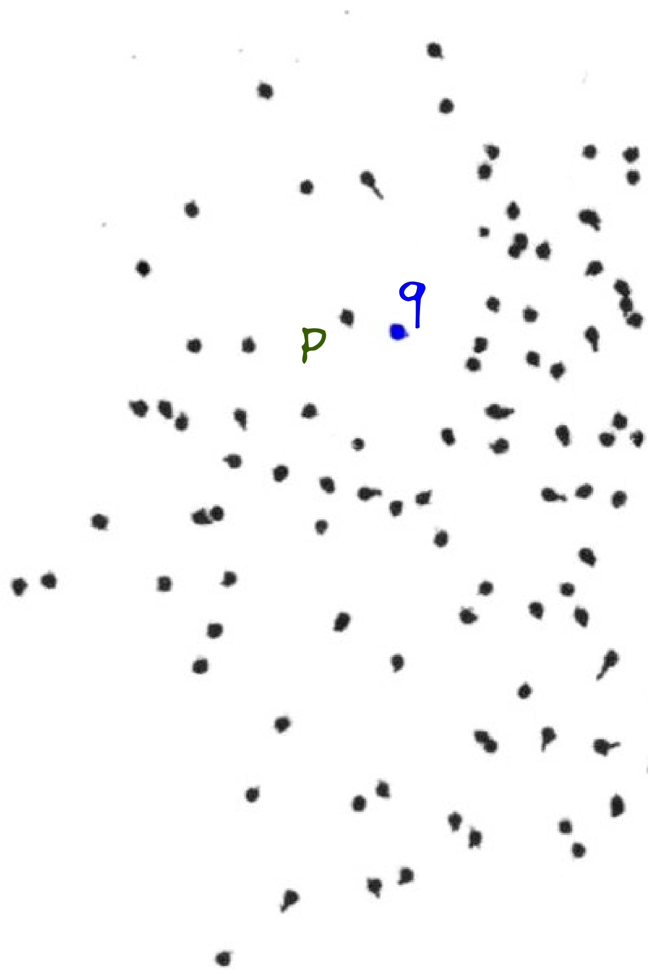
COMS-4771

# Nearest Neighbor Methods in Learning

April 3, 2008

# Nearest Neighbor Search

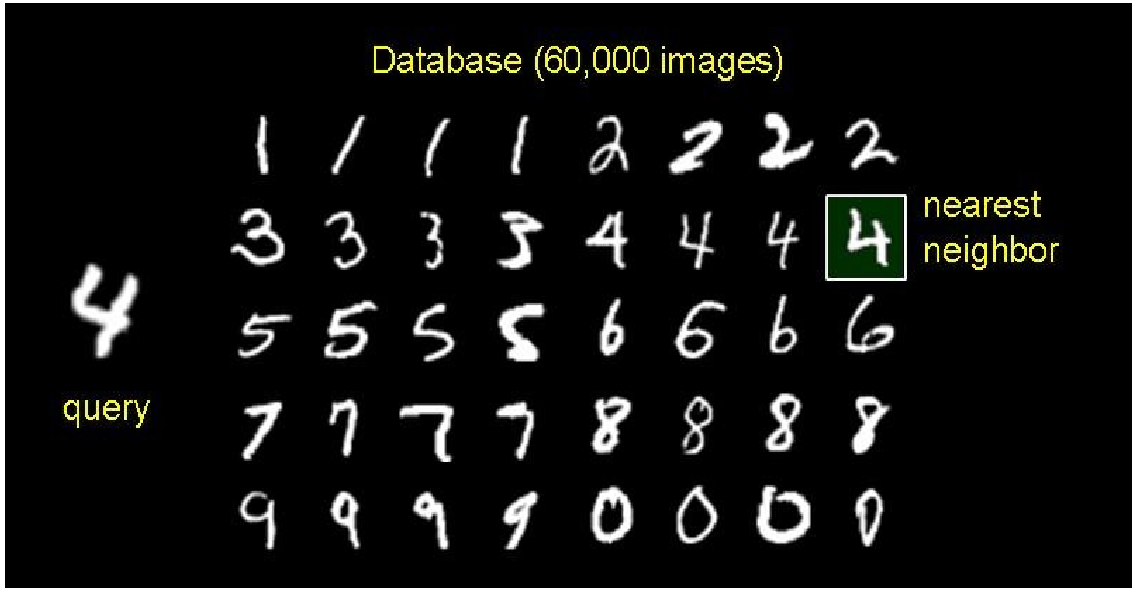
- Metric space (M,d): a set of points M,
  a distance function d over M
- Pre-process a given subset S of M, |S| = n
- Given a query point q in M, quickly return a point p in S minimizing d(q,p).
- Linear scan: n queries (no preprocessing), space O(n)



A basic computational primitive used in solving a large class of proximity problems: minimum spanning tree, diameter, closest pair, spread, facility location, reverse nearest neighbor, range queries (intersection of S with a query object), etc

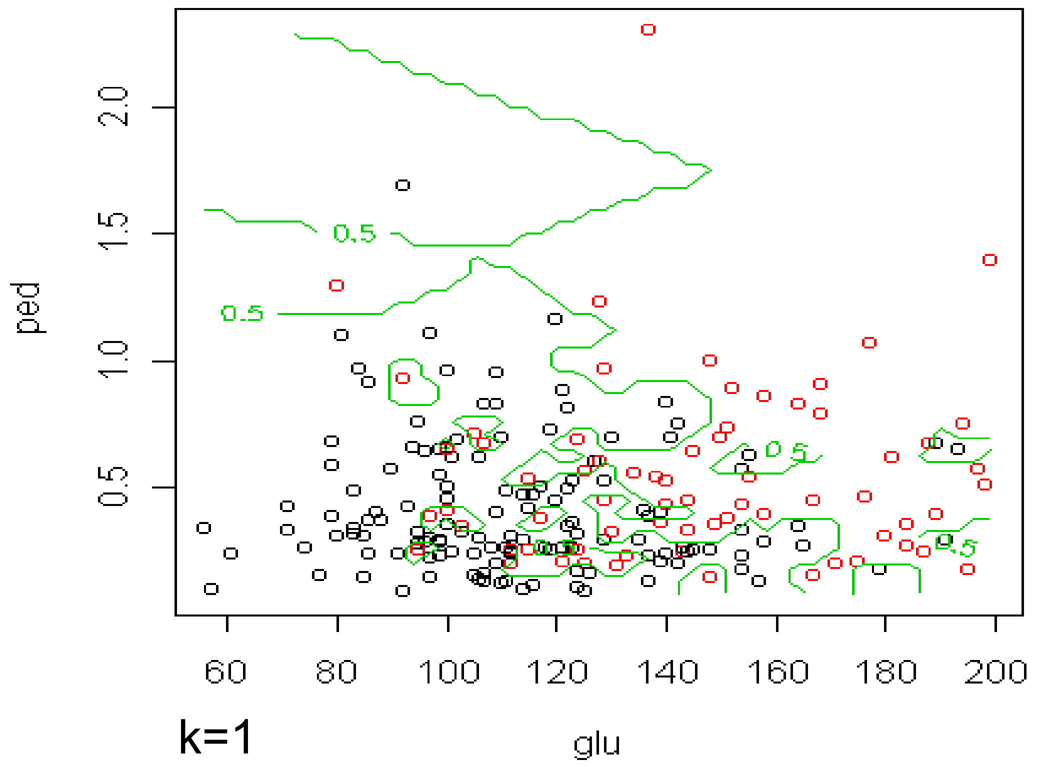
# Applications in Learning

- k-nearest neighbor classification and regression

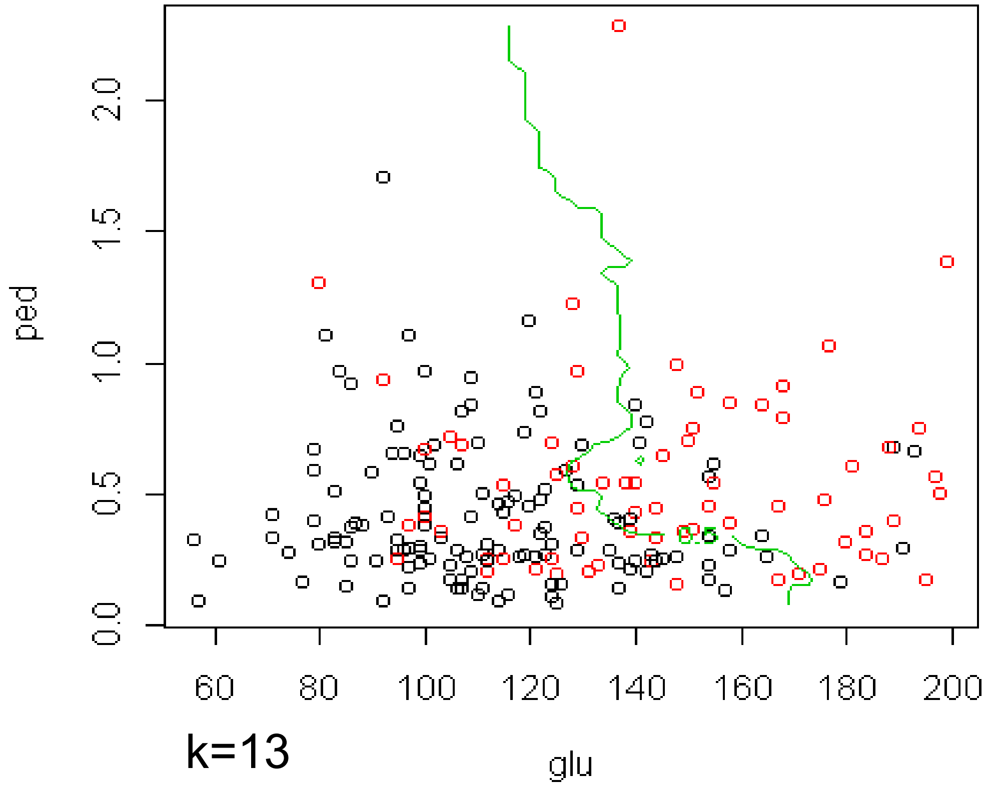


- Classification: An object is assigned to the class most common among its k nearest neighbors (majority in binary classification)
- Regression: An object is assigned the average of the values of its k nearest neighbors (can be weighted by the distance).
- Lazy learning

## Diabetes classification



k=1

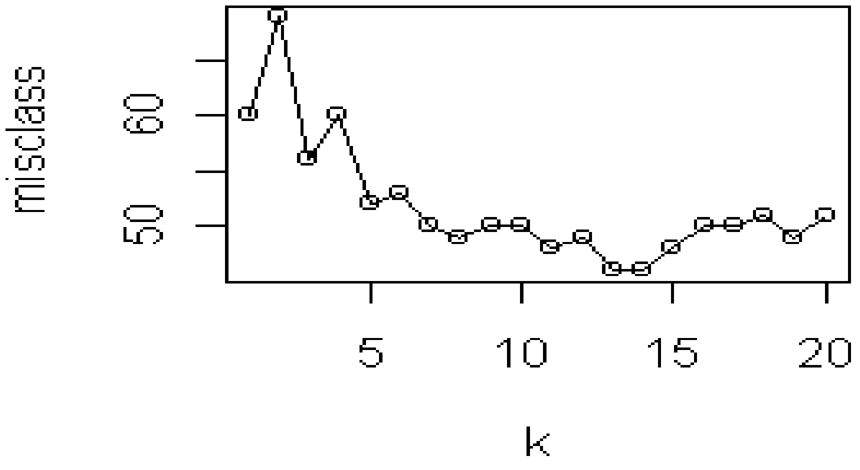


k=13

The classes overlap, so we get lots of isolated regions where the predicted class changes. To get a smoother boundary, use a larger k.

### How to choose the best k?

Use cross-validation



## K-Nearest Neighbor Algorithm

- We need a way to measure distances between the query and database objects. The algorithm is very sensitive to the local structure of the data
  - Examples of metrics: Euclidean distances, edit distance for strings, Hausdorff matching for edge images, the Kullback-Leibler distance and the Earth Mover's Distance for probability distributions
  - When features have different scales, it's a good idea to standardize them first so that they have standard deviation 1 (by dividing each feature value by its standard deviation on the training set).
  - Distance metric learning: Learn the metric to optimize classification accuracy!
- Problem: efficiency
  - Nearest neighbor search data structures (this lecture)
  - Prune away some of the examples. The algorithm is largely unaffected by removing examples. Only the boundary examples are important.

## Properties of the K-Nearest Neighbor Algorithm

- Provides no concise model of the dataset
- Very robust to noise in the dataset (unlike many other methods, which can change substantially from the alteration of a single training example.)
- Very sensitive to irrelevant features, and doesn't perform well when some features are more important than others (the opposite of logistic regression and classification trees. This problem can be fixed by designing your own distance measure or doing feature selection before running the algorithm.)
- Nearest-neighbor is especially useful for domains where a distance measure between examples is straightforward to define but a model relating the features to the response is not.

# Nonlinear Dimensionality Reduction

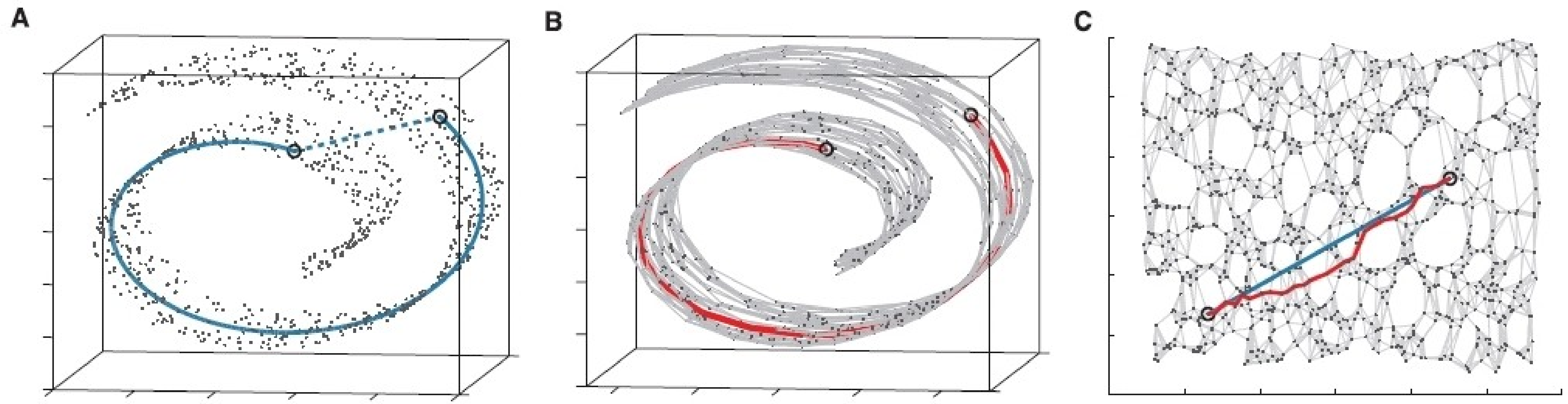


**Fig. 3.** The "Swiss roll" data set, illustrating how Isomap exploits geodesic paths for nonlinear dimensionality reduction. (**A**) For two arbitrary points (circled) on a nonlinear manifold, their Euclidean distance in the high-dimensional input space (length of dashed line) may not accurately reflect their intrinsic similarity, as measured by geodesic distance along the low-dimensional manifold (length of solid curve). (**B**) The neighborhood graph *G* constructed in step one of Isomap (with *K* = 7 and *N* = 1000 data points) allows an approximation (red segments) to the true geodesic path to be computed efficiently in step two, as the shortest path in *G*. (**C**) The two-dimensional embedding recovered by Isomap in step three, which best preserves the shortest path distances in the neighborhood graph (overlaid). Straight lines in the embedding (blue) now represent simpler and cleaner approximations to the true geodesic paths than do the corresponding graph paths (red).

Isomap, locally linear embeddings, maximum variance unfolding

Isomap: 1) Construct neighborhood graph, 2) compute shortest paths, 3) compute d-dimensional embedding (using eigen-information)

## Other Applications

- Clustering (e.g, k-means)
- Information retrieval, data mining
  - Similarity search, content-based retrieval, near-duplicate detection, recommendation systems, problem diagnosis, etc.
- Compression, vector quantization

- Probably more applications than any other geometric problem

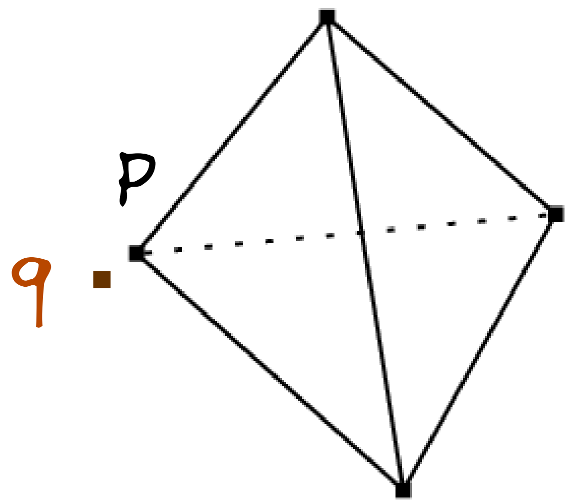
## General case is hard, even for approximate queries

return p with $d(p,q) \le (1+\varepsilon)d(q,S)$

Example: A near-uniform S with

$d(x,y) \approx 1$ for all x, y in S.

Query complexity: $\Omega(n)$



But many data sets don't have large uniform subsets.

How do we quantify this?

## Special case: Euclidean spaces ($l_2$)

- Low-dimensional (<20-30 depending on n):
  - kd-trees (Bentley, 1975, >1300 citations); not the most efficient in theory, but works well in practice
- Exact algorithms (curse of dimensionality):
  - time or space exponential in the dimension (seems unavoidable); $O(d^{O(1)} \log n)$ query time, but $n^{O(d)}$ space.
- Approximate (randomized) search:
  - both space and time polynomial in the dimension:
    - Dimensionality reduction (Johnson-Lindenstrauss-type projections: n points in Euclidean space can be projected down to $O(\varepsilon^{-2} \log n)$ dimensions with distortion at most $\varepsilon$); space is an issue $n^{O(\varepsilon^{-2})}$ (Kushilevitz-Ostrovsky-Rabani 1998, etc)
    - Locality Sensitive Hashing (Indyk-Motwani 1998, etc)

## Un-normed metrics

- Not all metrics of interest are Euclidean/normed.
  - Problem-specific distances (between time series, sequences, images, etc), on-demand measurements, functions learned via optimization
- Most methods developed for the Euclidean space don't apply to non-vector spaces
- Low-distortion embeddings: Embed your metric into a normed metric and solve the problem there:
  - Embedding introduces additional distortion and complexity
  - Works (well) only for very specific metrics (may not work for yours): l-infinity (max), Hausdorff metric (max-min between sets), Earth-Mover (min matching between sets), string/block edit distance

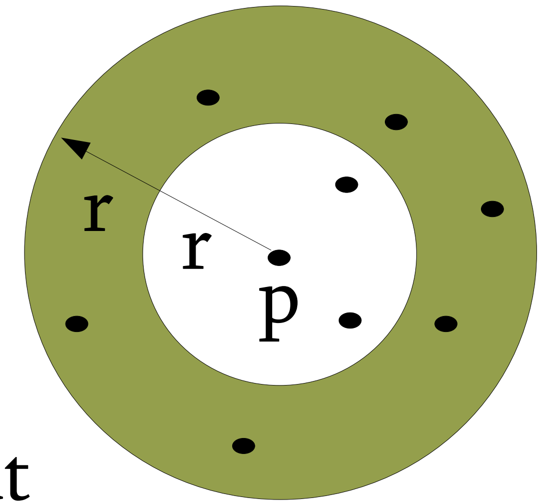
## Metrics with low intrinsic dimension

- The “real” dimension of the data is typically low
- We want complexity to depend on the intrinsic dimension rather than on the explicit dimension
- The distance function is used only as a black box (no manipulation of coordinates). Pruning relies only on metric inequalities.
- How to quantify the real dimension?
- How to exploit it?

# Notions of Intrinsic Dimension: Bounded Growth

Ball of radius r around p:

$$B_r(p) = \{q \in S : d(q,p) < r\}$$



Expansion constant = smallest c such that

$$\text{For all } p \in S, r \geq 0, \quad |B_{2r}(p)| \leq c|B_r(p)|$$

Expansion dimension = log c

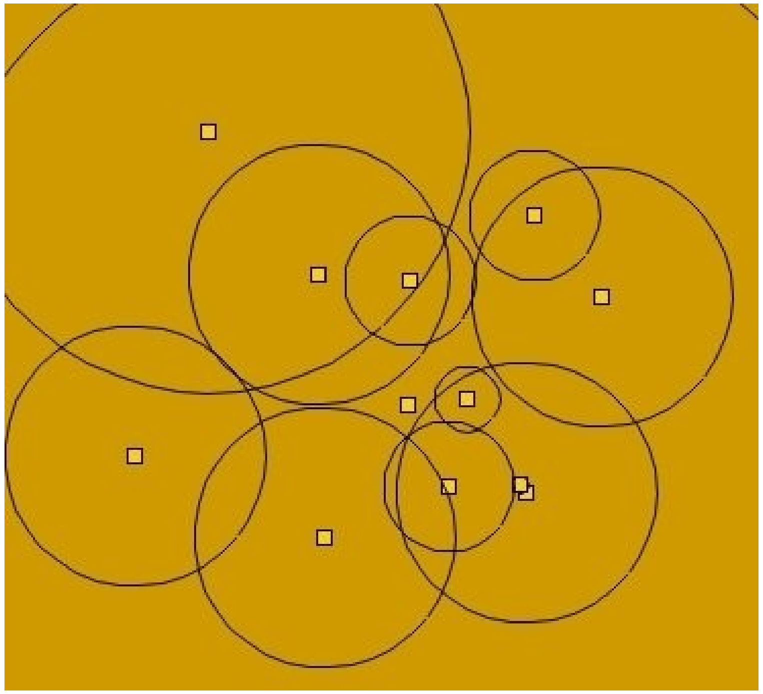
If S is arranged uniformly on some surface of dimension d, c ≈ $2^d$

For the uniform metric, c = O(n)

(aka Federer measure)

# Cover Trees

- A simple, deterministic data structure for exact and approximate NNS
- Linear space, independent of any notion of dimensionality
- Query time: $\min\{O(c^{O(1)}\ln n), O(n)\}$
- Dynamic (insert/remove): $\min\{O(c^{O(1)}\ln n), O(n)\}$
- The algorithm does not need to know c
- Works well in practice
- Supports range queries, k-NNS; batch and lazy construction, batch queries



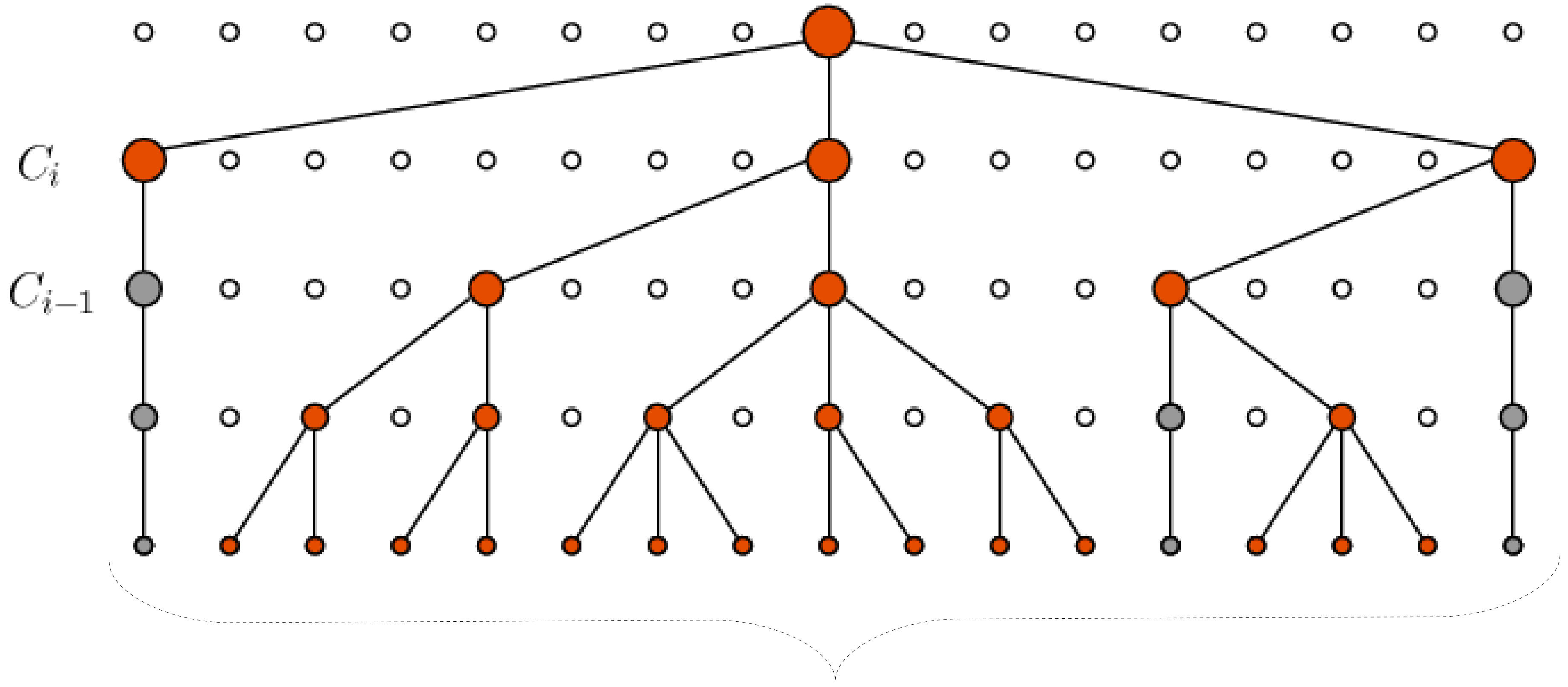
Joint work with

Sham Kakade
(TTI-Chicago)

John Langford
(Yahoo! Research)

# What is a Cover Tree? (slide 1 of 3)

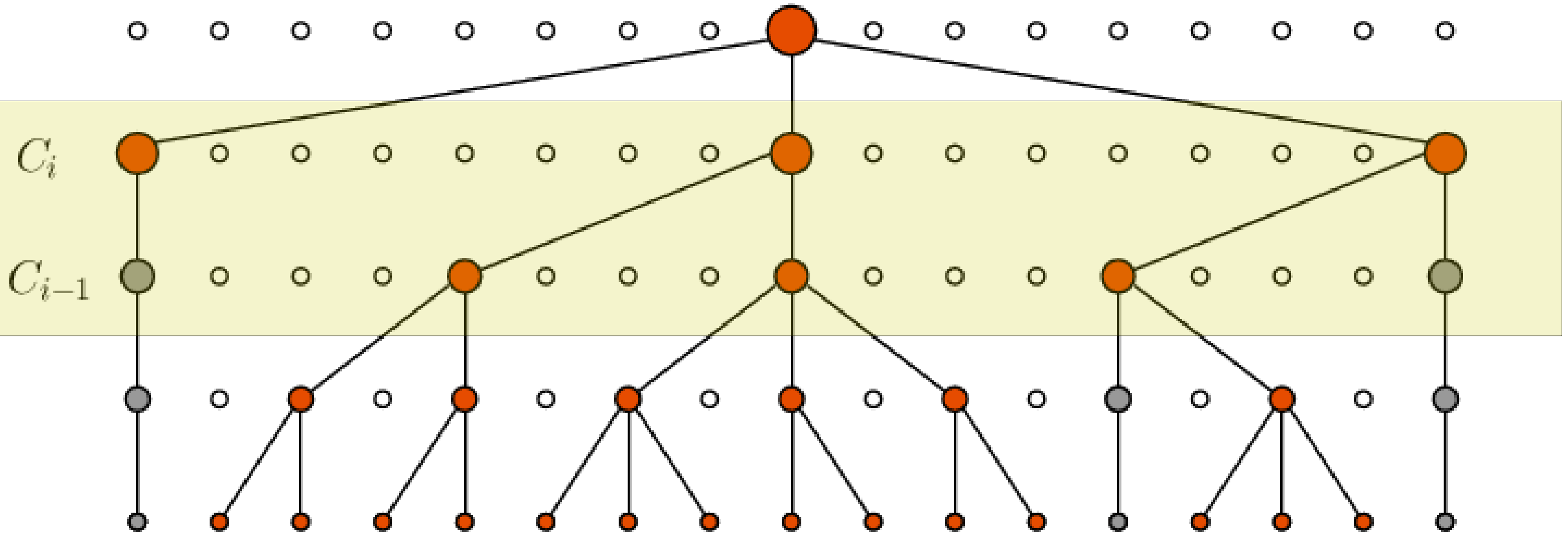
**Nesting invariant : For all** $i,\ C_i \subset C_{i-1}$



points in S

## Covering:

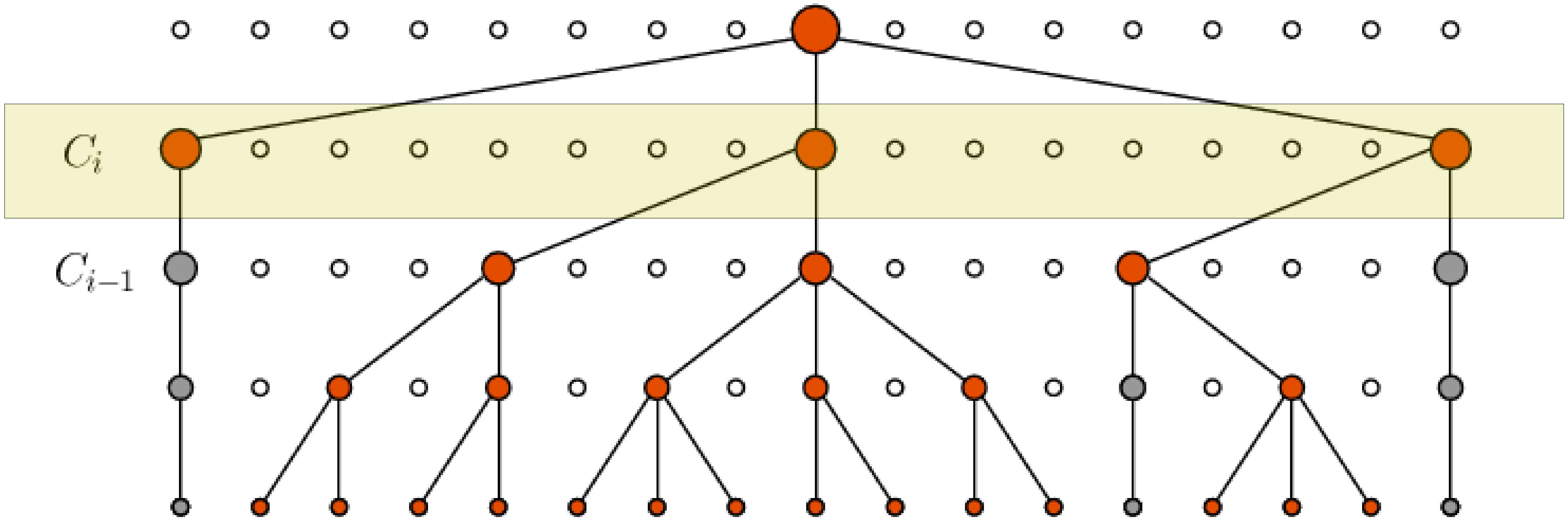
**Every node in $C_{i-1}$ is within $2^i$ from some node in $C_i$ ; one such node is chosen as parent**



Navigating nets [Krauthgamer-Lee'04]: a node is connected to all possible parents; cover trees preserve the runtime properties while throwing away most links.

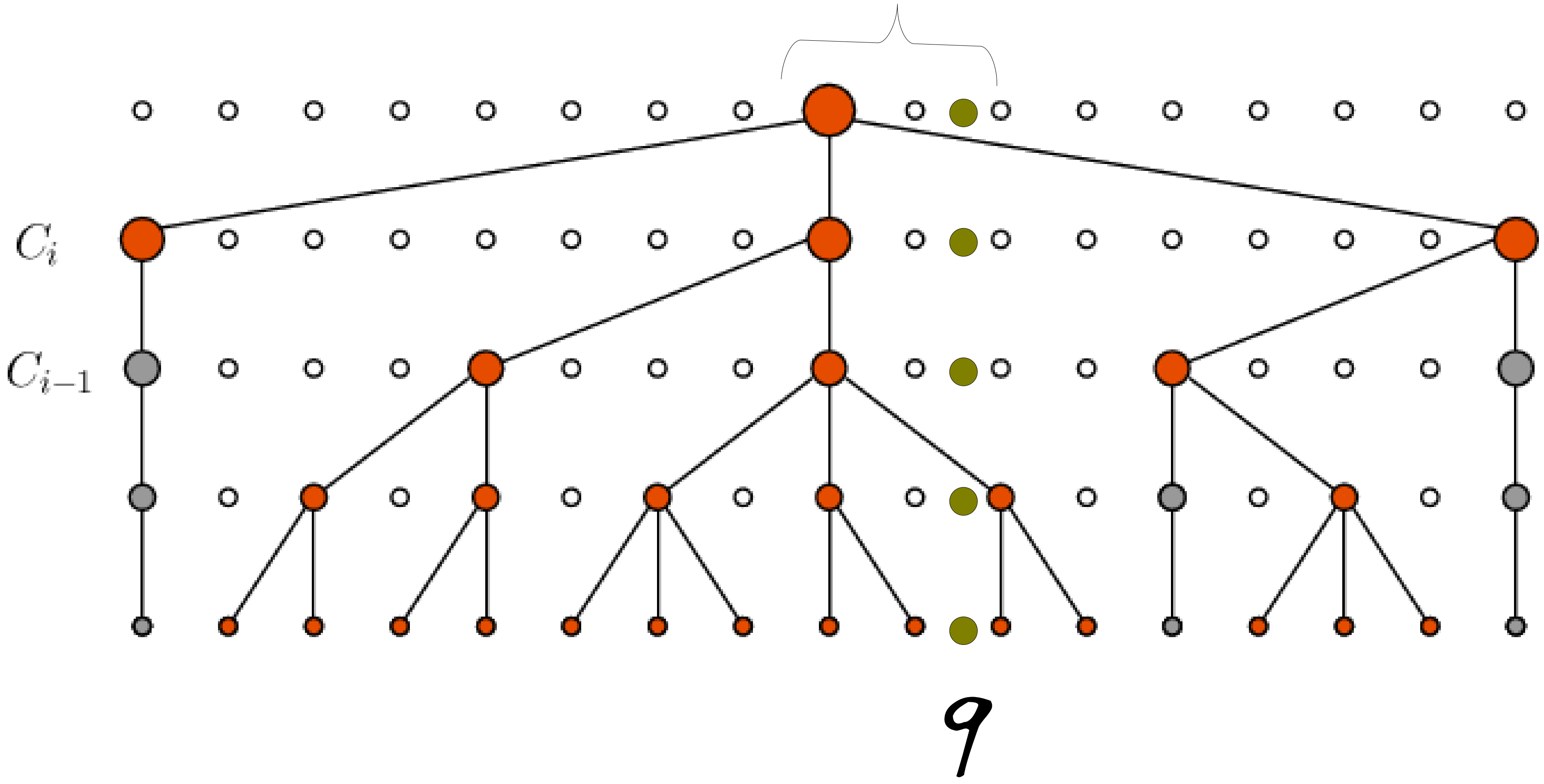
## Separation:

All nodes in $C_i$ are at least $2^i$ apart



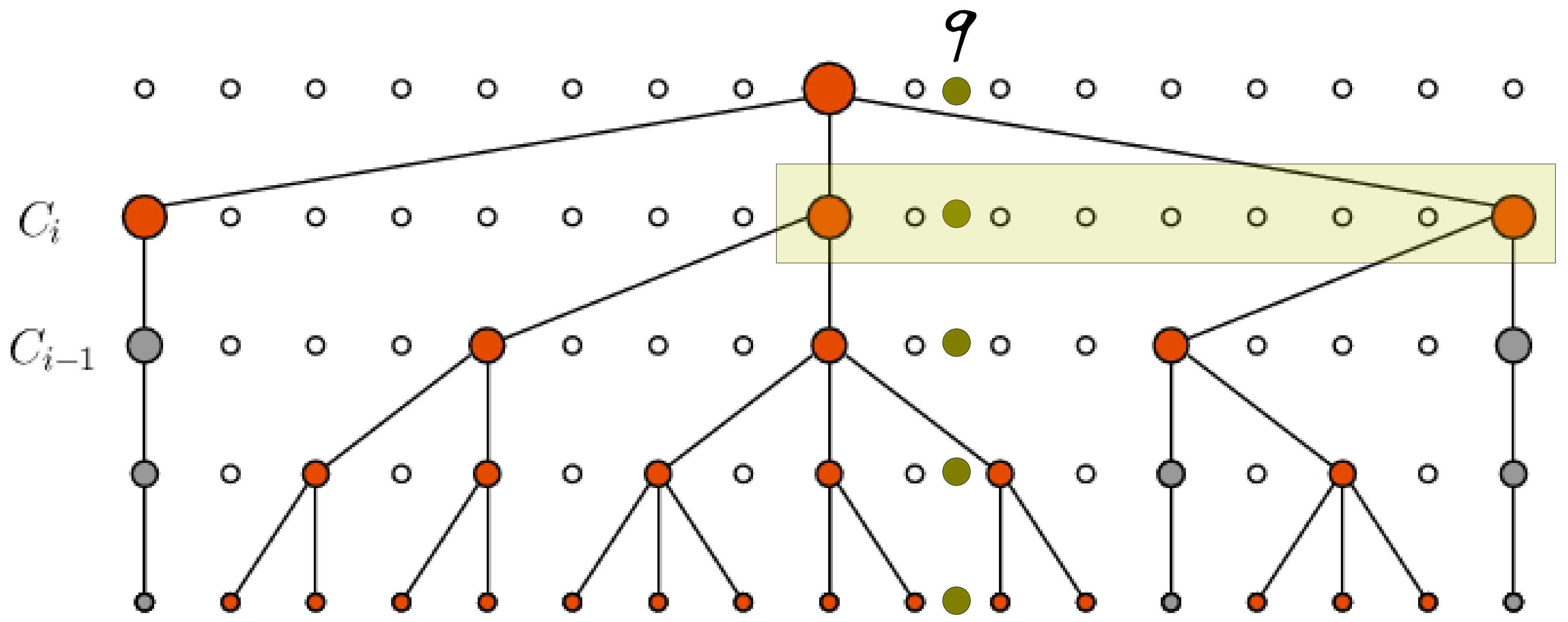
# Search: query point q

Start with an upper bound = distance to root node



## Search: query point $q$

Descend maintaining a **cover set** = set of nodes that may contain the nearest neighbor of $q$

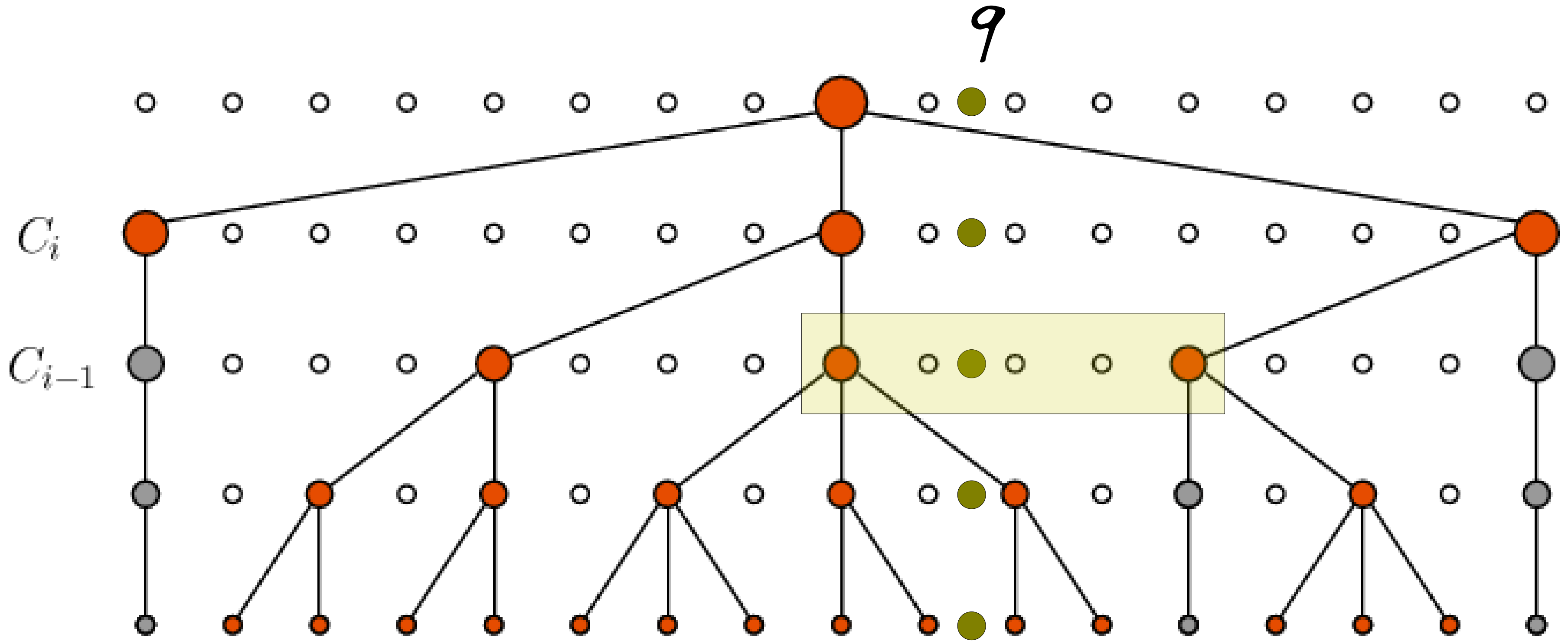


**Pruning:** A node in level $i$ is at most away from any of its descendants

$$\sum_{j=i}^{-\infty} 2^j \leq 2^{i+1}$$

# Search: query point $q$

Descend maintaining a **cover set** $Q_i$ = set of nodes that may contain the nearest neighbor of $q$

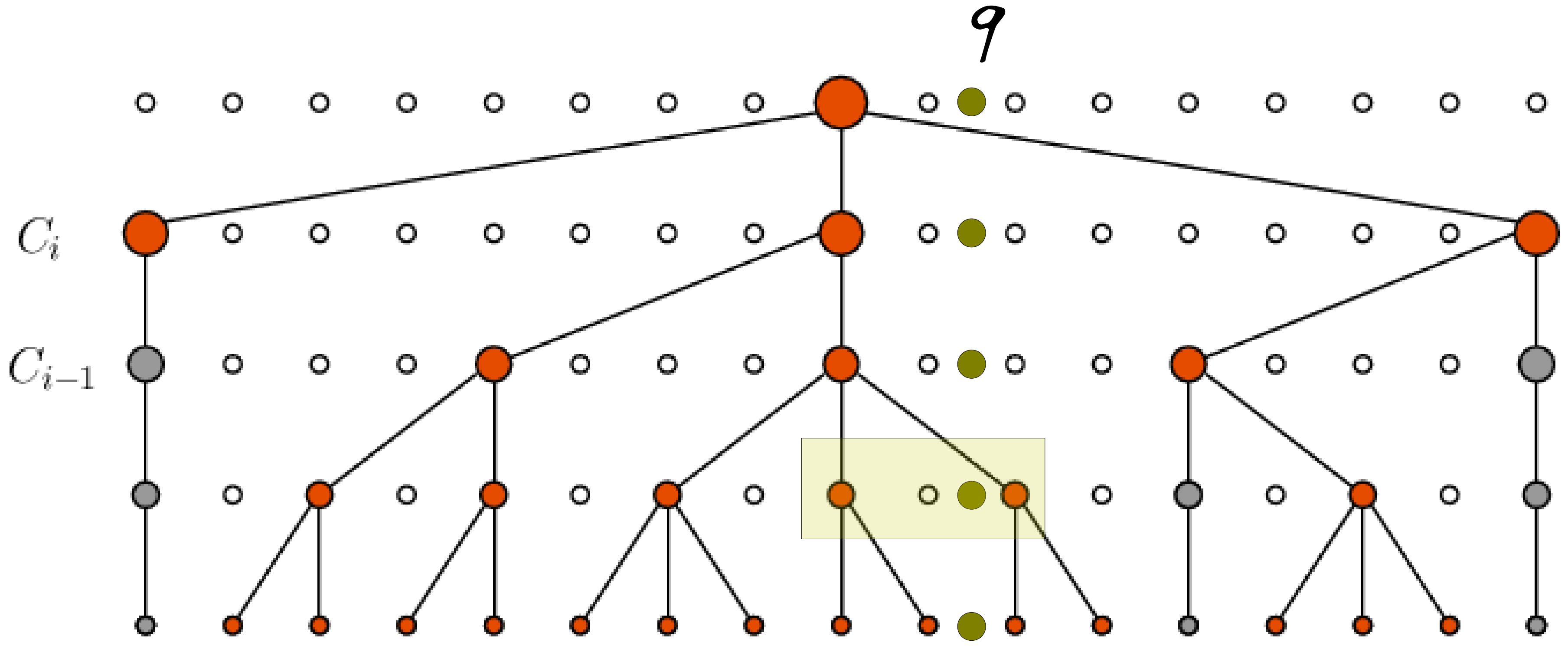


$$Q_{i-1} = \{p \in \textsc{Children}(Q_i) :$$

$$d(q,p) \leq d(q, \text{Children}(Q_i)) + 2^i\}$$

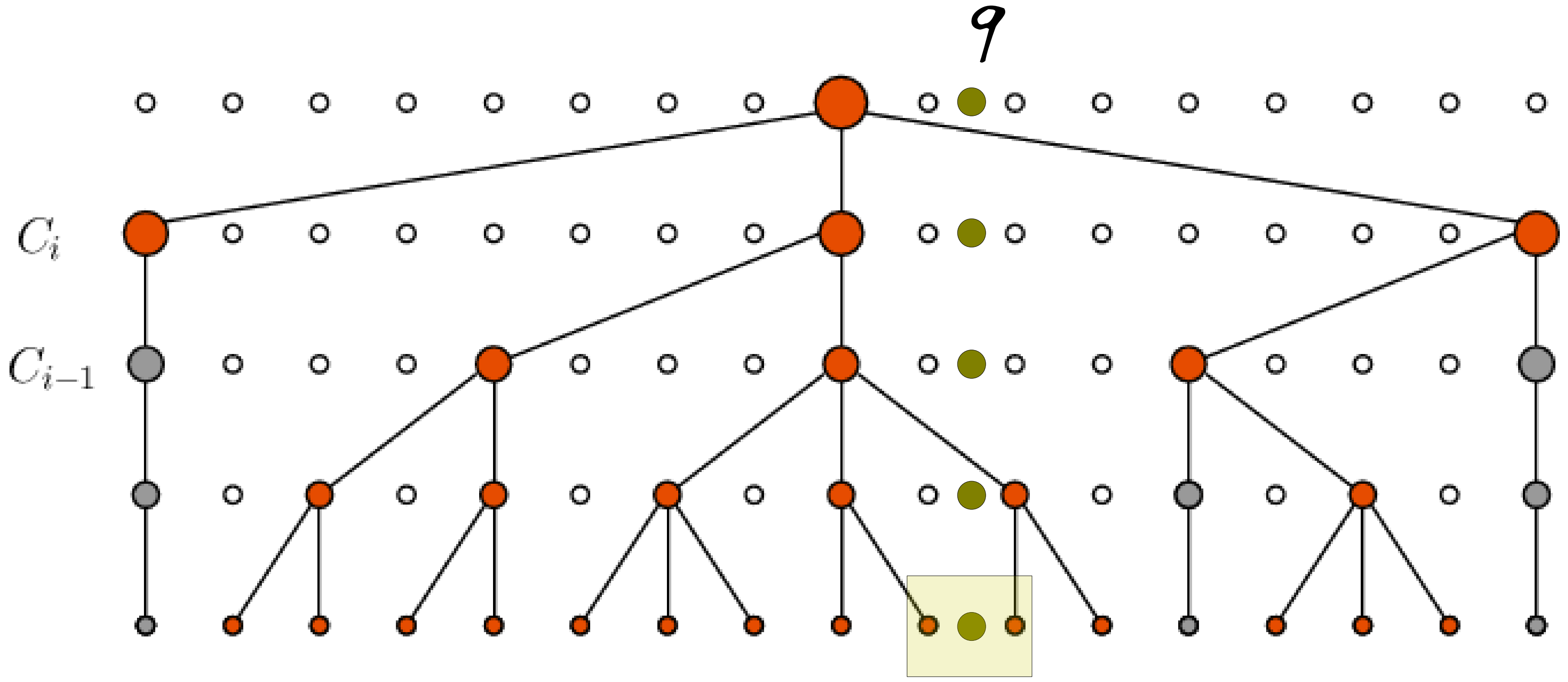
# Search: query point $q$

Descend maintaining a **cover set** = set of nodes that may contain the nearest neighbor of $q$



# Search: query point $q$

**Descend maintaining a cover set = set of nodes that may contain the nearest neighbor of $q$**



## Approximate Search

Given a query q, return a point p in S with

$$d(q,p) \leq (1+\epsilon)d(q,S)$$

Modification of the algorithm: Stop as soon as

$$2^{i+1}(1+1/\epsilon) \leq d(q,Q_i) \qquad (*)$$

Proof: Suppose we terminated at level $i$. Then

$$d(q,Q_i) \leq d(q,S) + 2^{i+1} \qquad (**)$$

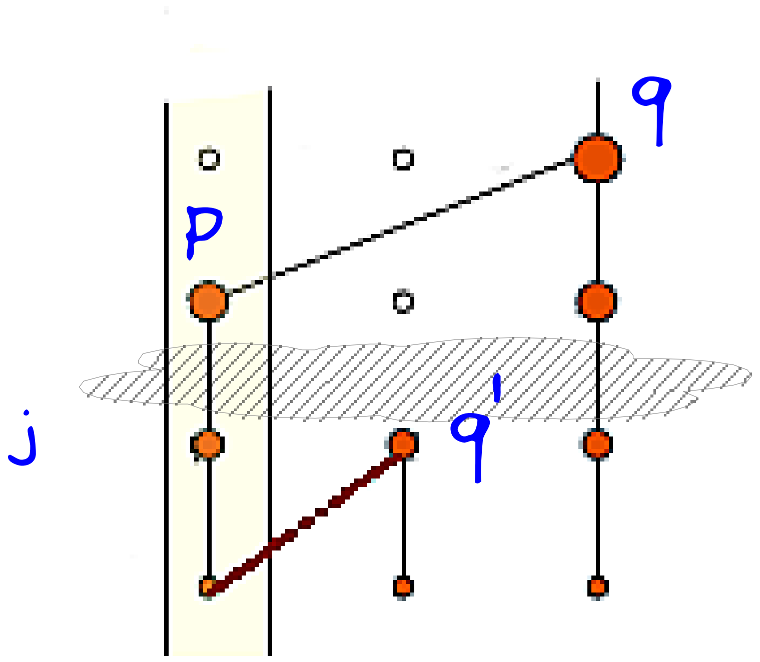
Combining with (*) gives

$$2^{i+1} \leq \epsilon d(q,S)$$

Combining with (**) proves the claim.

# Space Bound

**Claim:** Every point has at most one parent other than itself.



**Proof:** Suppose $p$ has two parents, $q$ and $q'$. They cannot be parents of $p$ at the same level (by construction). Let $q'$ be the parent at the lower level $j$. But $p$ is also at level $j$ (nesting invariant), thus $d(p,q') > 2^j$, a contradiction.

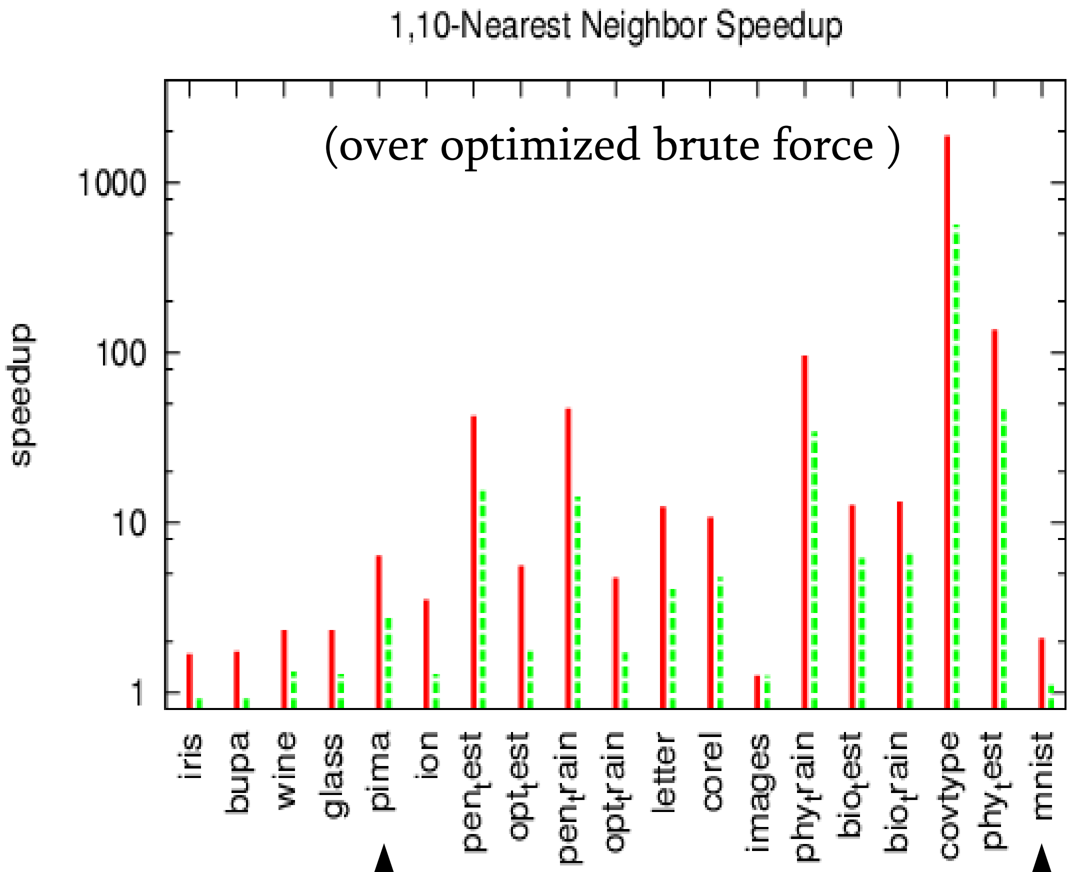
There are no chains, so there are at most O(n) links.

## Query Time Analysis: Basic Lemmas

**Width bound:** Any node p has at most $c^4$ children.

**Depth bound:** The maximum depth of any point is $O(c^2 \log n)$

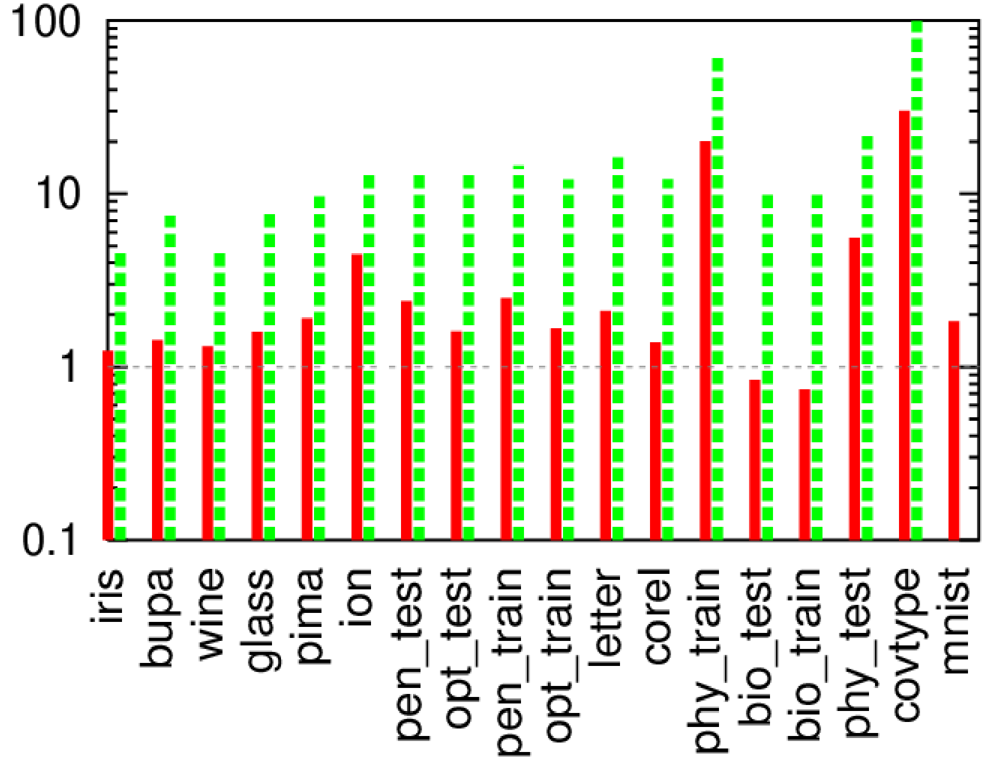
**Query time:** If $S \cup \{p\}$ has expansion constant $c$, the NN of $p$ can be found in time $O(c^{12} \log n)$



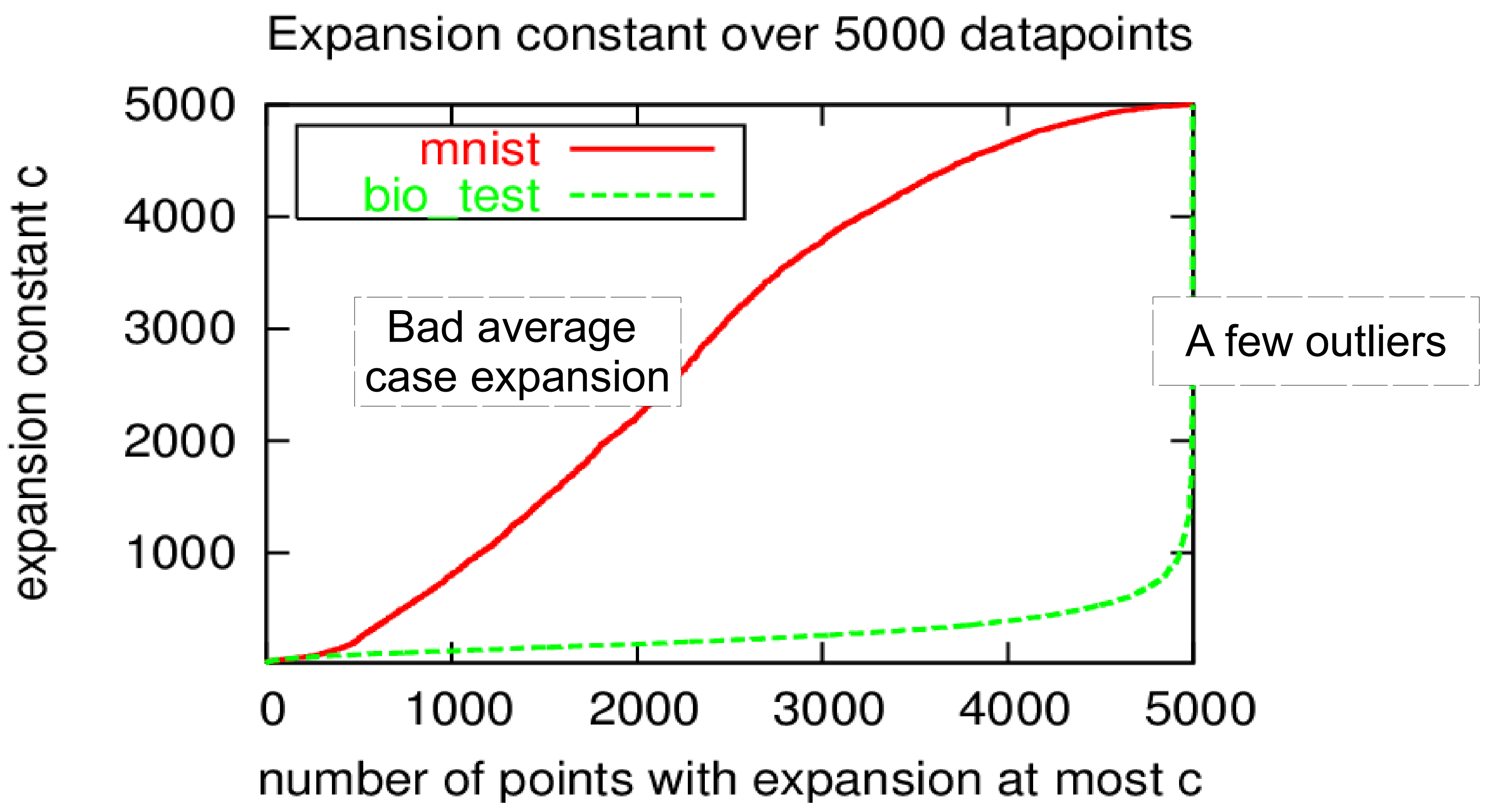
Diabetes dataset

Handwritten digits

Comparison with sb(S).
Euclidean norm, 1,2-nearest neighbor



## Does c capture the complexity of NNS?



## Metric repair: My distance function is not a metric

Symmetry: $d(p,q) = d(q,p)$

- $d'(p,q) = [d(p,q)+d(q,p)] / 2$

Triangle inequality: $d(p,q) \leq d(p,r) + d(r,q)$

- Any relaxation works: $d(p,q) \leq C(d(p,r) + d(r,q))$ for some $C>1$; deal with the outliers if it holds for most triples.
- Use $d'(p,q) = d(p,q)^w$ for some sufficiently small w. If the smallest w is 0 (or $C \sim \infty$), uniform metric (bad).
- Use the shortest path (graph) metric: $d'(p,q) = \inf \sum_i d(x_i,x_{i+1})$, where the inf is over all sequences $p = x_1, x_2, \ldots, x_N = q$.

## Final comments

- Other algorithms/packages:
  - Locality-Sensitive Hashing (LSH), Indyk and Andoni (MIT)
  - ANN (Arya and Mount), Clarkson's sb(S)
  - Metric and ball trees
- Excellent surveys by
  - Piotr Indyk (MIT)
  - Ken Clarkson (IBM Almaden)