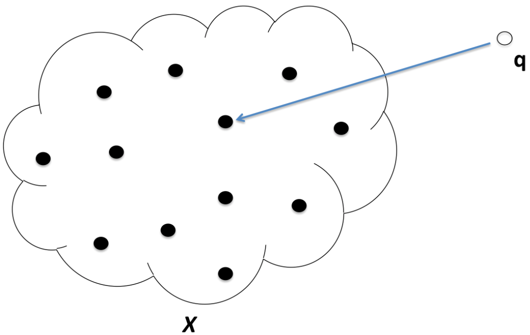
# Cover Trees for Nearest Neighbor

Alina Beygelzimer, Sham Kakade, John Langford

*(Presented by: Parameswaran Raman, params@ucsc.edu)*

April 11, 2016

# Nearest Neighbor Problem




Given a set $S$ of $n$-points in some metric space $(X, d)$, pre-process $S$ s.t given a query point $q \in X$, one can efficiently find a point $p \in S$ which *minimizes* $d(p, q)$.

# Comparison of Methods and Challenges

**Brute Force**

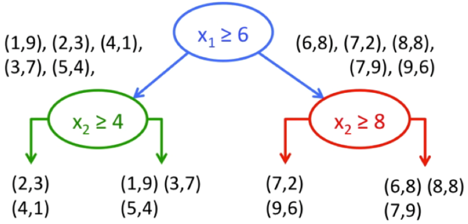
- requires no pre-processing
- query time is $O(n)$, space is $O(n)$

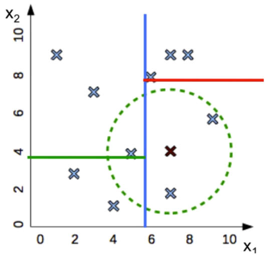
# Comparison of Methods and Challenges

## K-d Tree [FBL77]

**Data points in 2D: ($x_1$, $x_2$)**

(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)

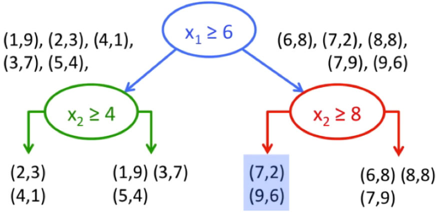


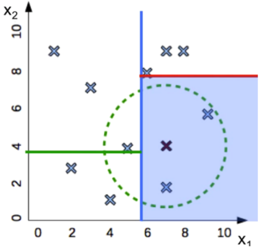



# Comparison of Methods and Challenges

## K-d Tree [FBL77]

**Query point: (7, 4)**

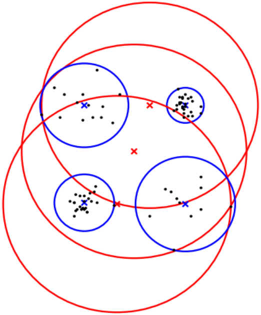



- works only for low dimensions ($\approx$10)
- inexact NN

# Comparison of Methods and Challenges

**Ball Tree / Metric Tree** [Omo87][Uhl91]



- works on moderately high dimensions (better than K-d trees)
- more generally applicable than K-d trees

# Comparison of Methods and Challenges

**Methods that assume structure (intrinsic dimensionality)** [KR02][KL04b]

- Karger and Ruth - [KR02] defines *expansion constant* - a measure of intrinsic dimensionality
- Navigating Nets by [KL04b] uses a similar measure

- Drawback: Although query times are good, space requirements are exponential in $d$ (# of dimensions)

# What are the highlights of the paper?

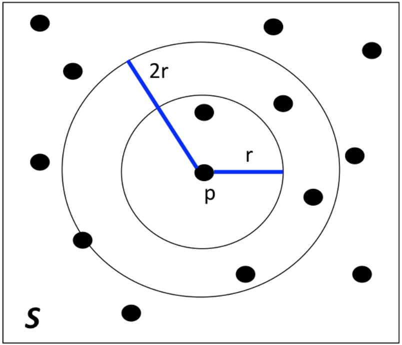
| | Brute Force | K-d Tree | Ball Tree | **Cover Tree** |
|---|---|---|---|---|
| Const Time | O(n) | O(nlogn) | O($n^2$) | O($c^6$ nlogn) |
| Const Space | O(n) | O(n) | O(n) | O(n) |
| Insert/Remove | O(n) | O(log n) | O(n) | O($c^6$ logn) |
| Query Time | O(n) | O(log n) | O(n) | O($c^{12}$ logn) |

- Linear space requirement independent of metric structure and dimensionality
- Query time as good as or better than other methods
- Theoretical guarantees
- Efficient implementations available

# Assumption about the structure: Expansion Constant

$B(p, r) = \{q \in S : d(q, p) < r\}$ = points within distance $r$ of $p$.

$\exists c$: $\forall p \in S, r > 0$: $c|B(p, r)| \geq |B(p, 2r)|$

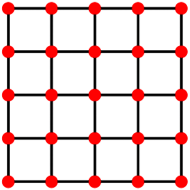



$c$ = Expansion Constant of the set of points $S$

# Assumption about the structure: Expansion Constant

If $S$ (set of points in the metric space $X$) is arranged uniformly on some surface of dimension $d$, then:
$c \approx 2^d$

- Eg: grid of equally spaced points in 2-D will have $c = 4$

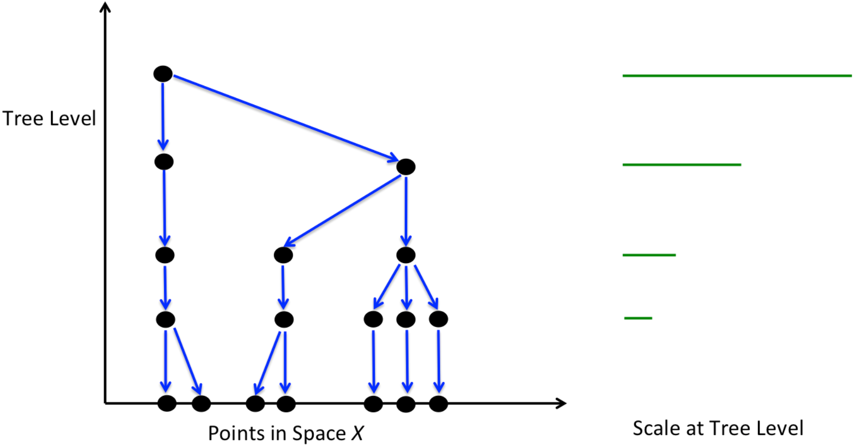


- Eg: grid of equally spaced points in 3-D will have $c = 8$
- Other interesting cases

# What is a Cover Tree?

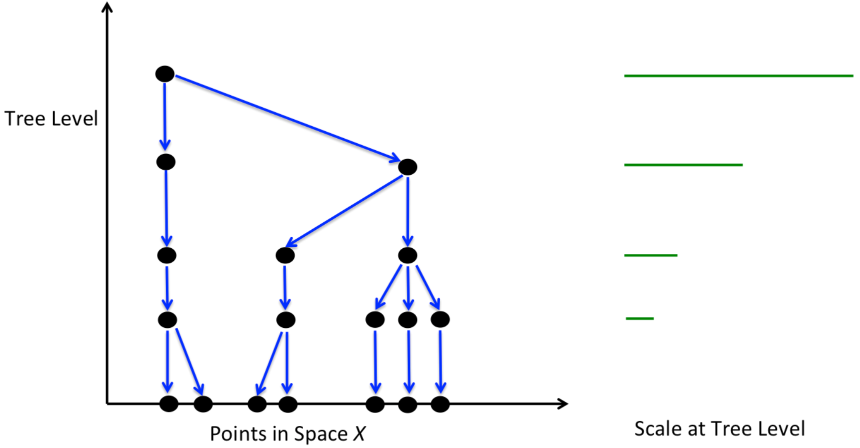
- Leveled tree satisfying invariants
- One point per node
- Lemma: for all nodes, number of children $\leq c^4$
- Lemma: maximum depth $= O(c^2 \log n)$

# What is a Cover Tree?



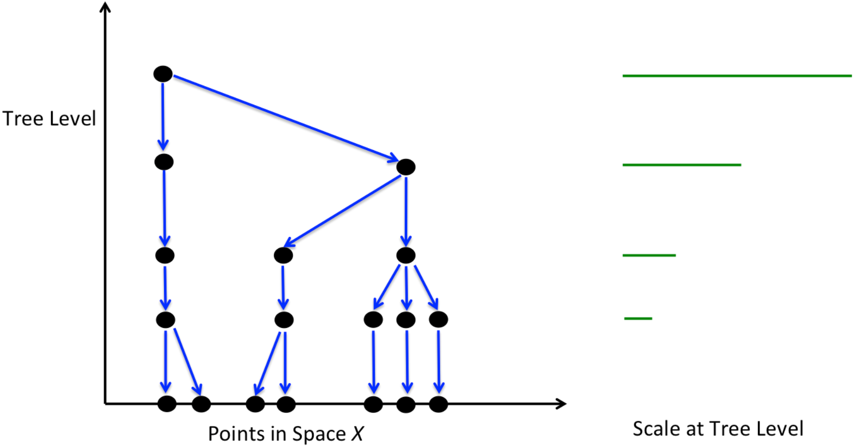

- (Nesting invariant): $C_i \subseteq C_{i-1}$ ($C_i$ = nodes at level $i$)

# What is a Cover Tree?



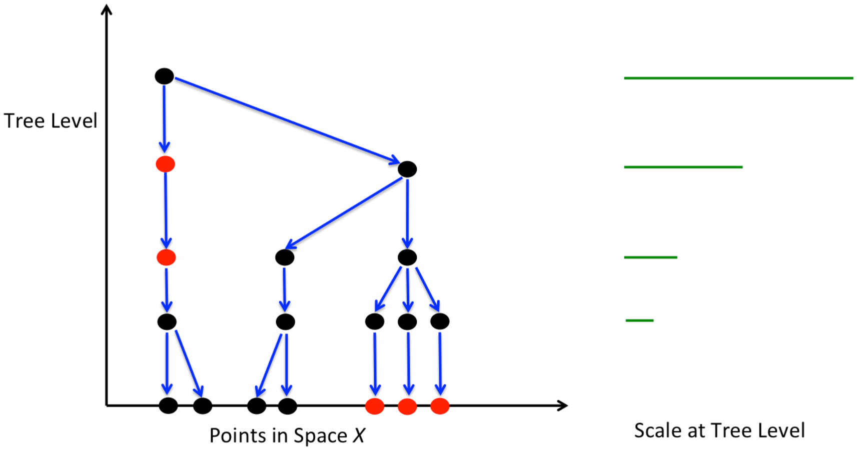

- (Covering Tree invariant): $\forall p \in C_{i-1}, \exists q \in C_i$: d(p,q) $\leq 2^i$ ($p$ is a child of $q$)

# What is a Cover Tree?



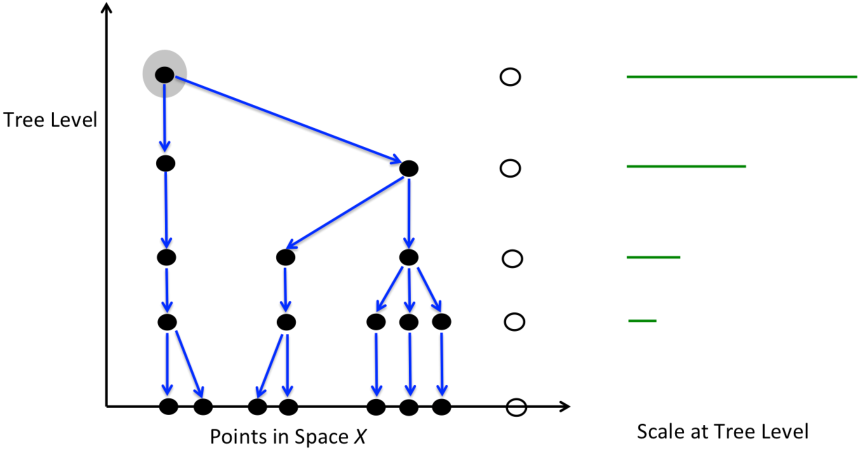

- (Separation invariant): If $p, q \in C_i$, then $d(p, q) \geq 2^i$

# What is a Cover Tree?



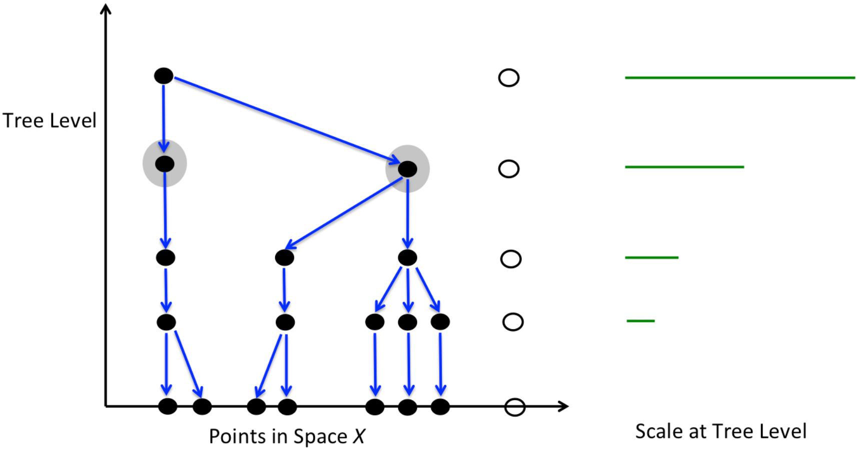

- Explicit Representation = Implicit Representation with duplicates removed

# How to use the Cover Tree for NN Query?



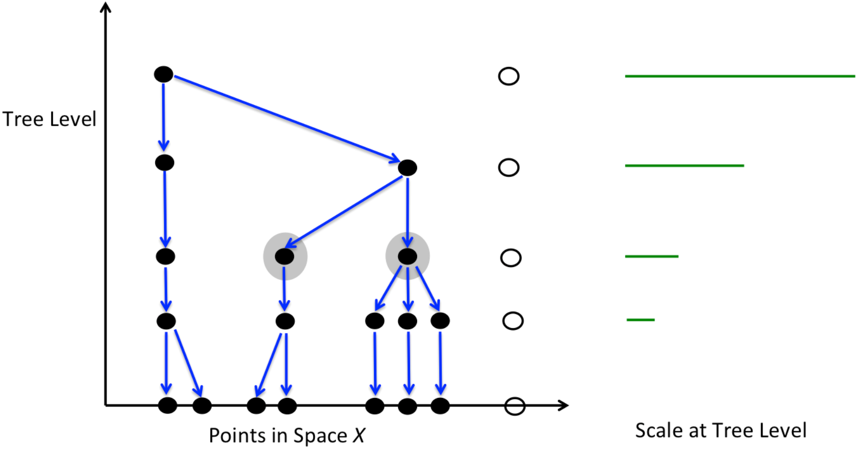

- Start with upper bound at root
- Descend tree, maintain a "cover set"

# How to use the Cover Tree for NN Query?



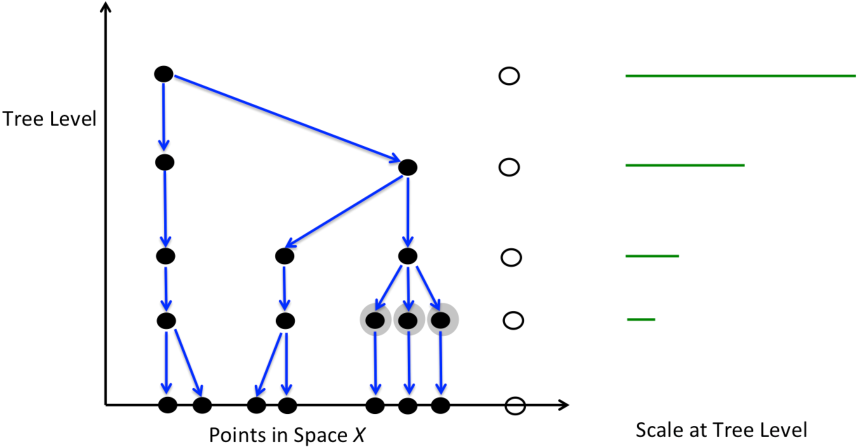

- Start with upper bound at root
- Descend tree, maintain a "cover set"

# How to use the Cover Tree for NN Query?



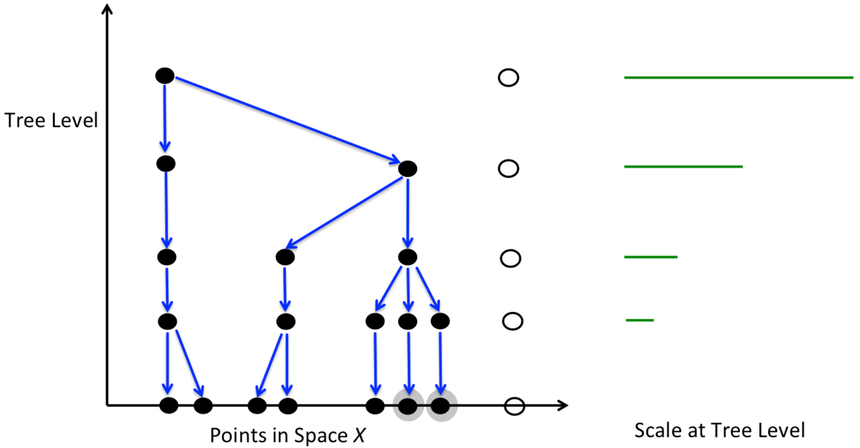

- Start with upper bound at root
- Descend tree, maintain a "cover set"

# How to use the Cover Tree for NN Query?




- Start with upper bound at root
- Descend tree, maintain a "cover set"

# How to use the Cover Tree for NN Query?

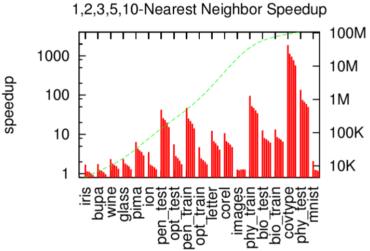



- Start with upper bound at root
- Descend tree, maintain a "cover set"

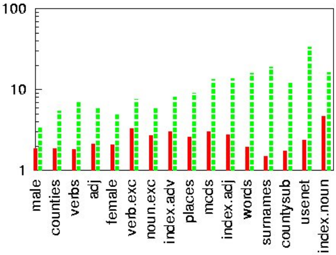
# How to make a Cover Tree?

- Single Point Insertion - $O(c^6 \log n)$
- Batch/Lazy Construction

# Empirical Study



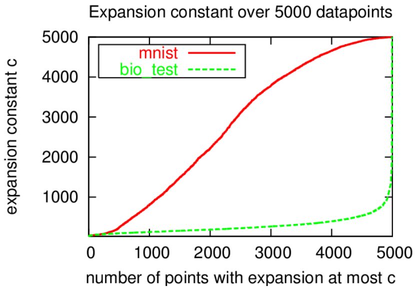

Speedup over Brute Force

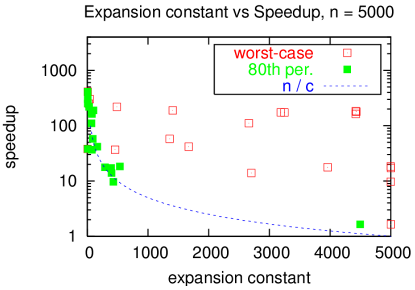



Speedup over sb(S) data structure [Ken Clarkson]

- Note: mnist is the largest dataset (n=60,000, d=784)

# Effects of expansion constant



# Summary

Cover Trees are:

- space-wise efficient than all other discussed methods
- time-wise on par with other discussed methods
- theoretically more elegant than K-d Trees / Metric Trees
- empirically not fully convincing yet - experiments need to be done on large-scale datasets comparing against Ball Trees and Navigation Nets

- Code: http://hunch.net/~jl/projects/cover_tree/cover_tree.html

# References


[1] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104. ACM, 2006.

[2] Kenneth L Clarkson. Nearest neighbor queries in metric spaces. *Discrete & Computational Geometry*, 22(1):63–93, 1999.

[3] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[4] David R Karger and Matthias Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 741–750. ACM, 2002.

[5] Robert Krauthgamer and James R Lee. Navigating nets: simple algorithms for proximity search. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 798–807. Society for Industrial and Applied Mathematics, 2004.

[6] Stephen M Omohundro. *Efficient algorithms with neural network behavior*. Department of Computer Science, University of Illinois at Urbana-Champaign, 1987.

[7] Jeffrey K Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information processing letters*, 40(4):175–179, 1991.