

Efficient Exact k-NN and Non-parametric Classification

Ting Liu, Andrew Moore, and Alexander Gray

NIPS 2003

Presented by Jan Vounq :: May 10, 2005

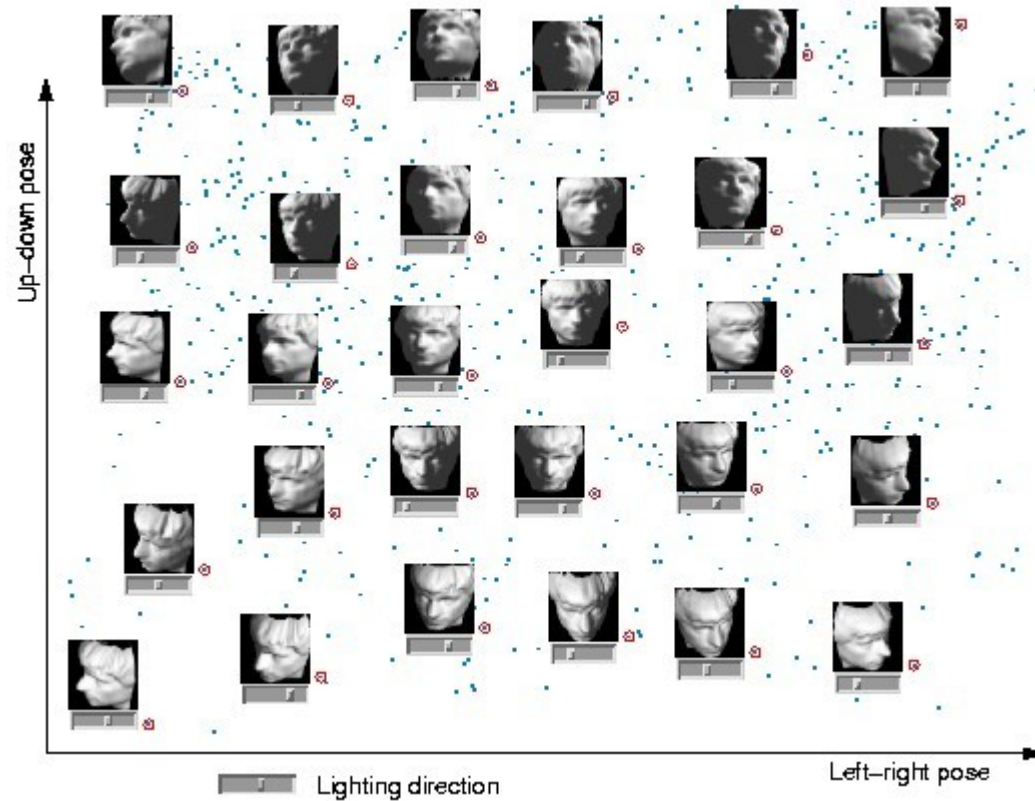
UCSD CSE 254

k-Parts of Talk

- Working in High Dimensions
- Exact k-NN
 - Ball Trees: “KNS1”
 - Two Ball Trees: “KNS2”
 - Fewer strings attached: “KNS3”
- Approximate k-NN
- Experiments

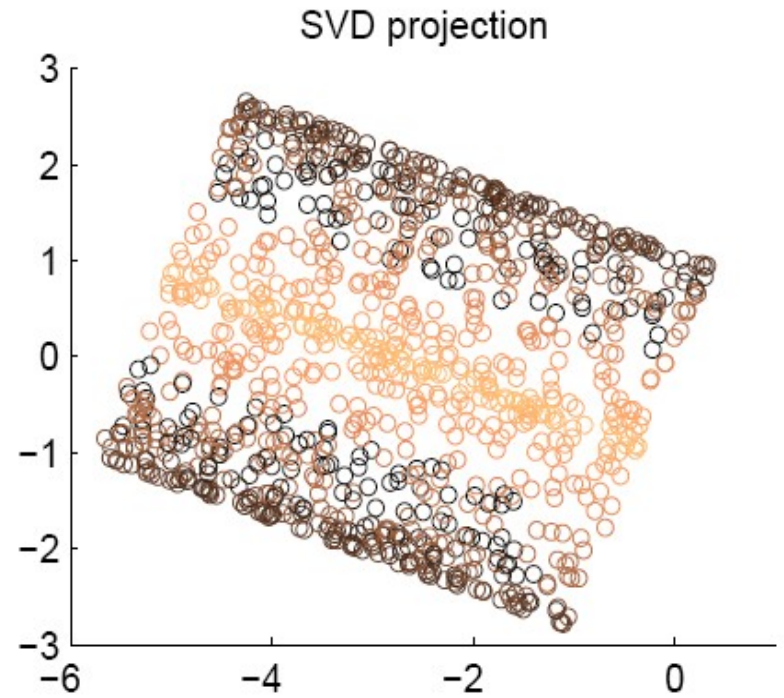
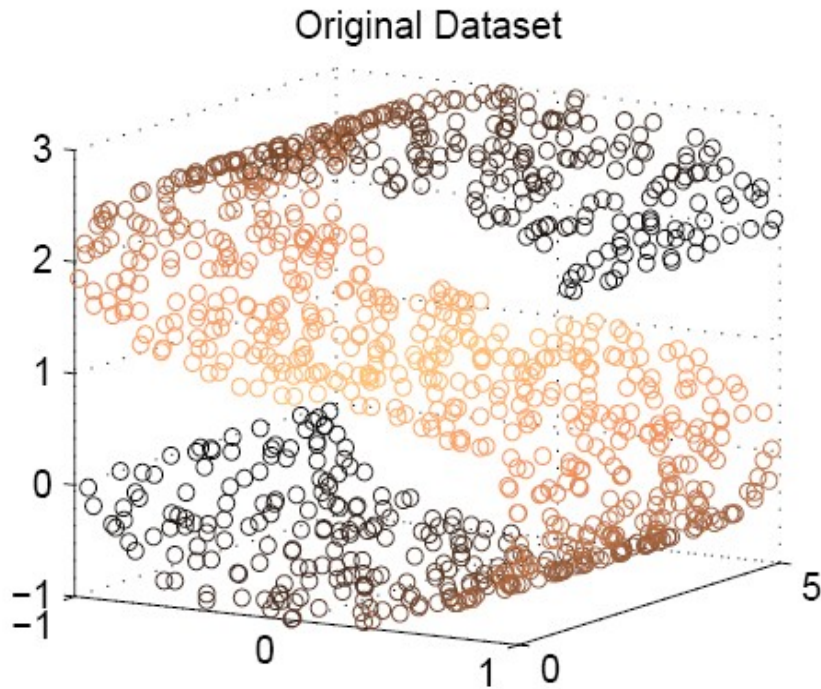
High Dimensional Data

- Text
- Images
- Genes
- Financial data



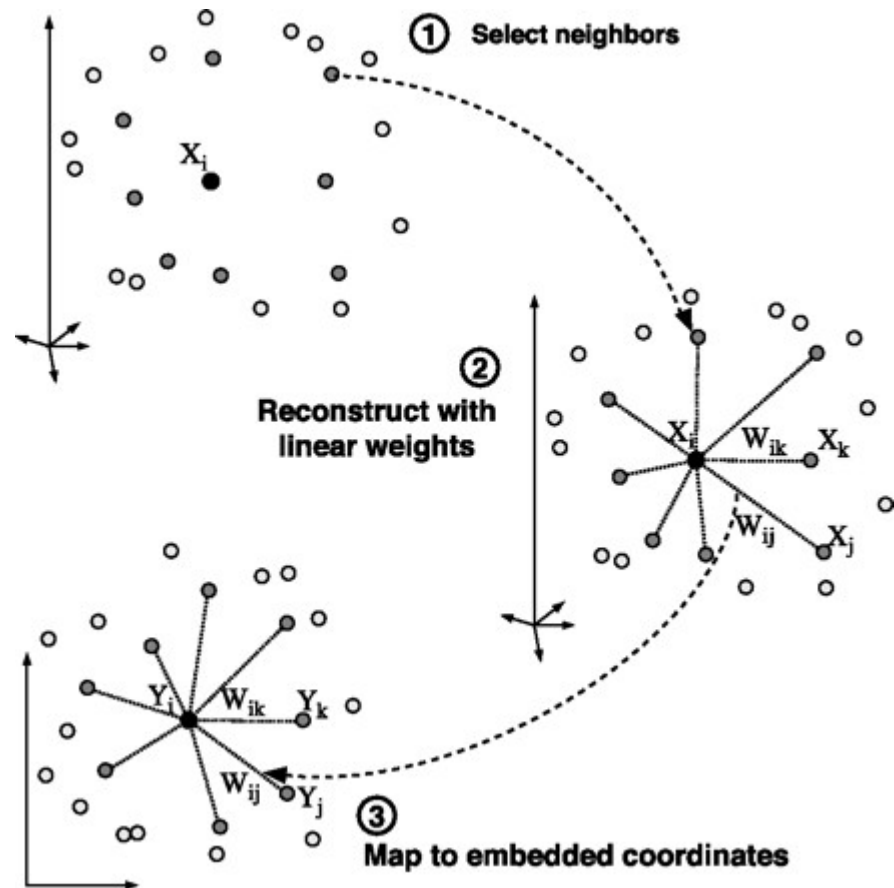
Dimension Reduction?

- Could collapse data points that are different



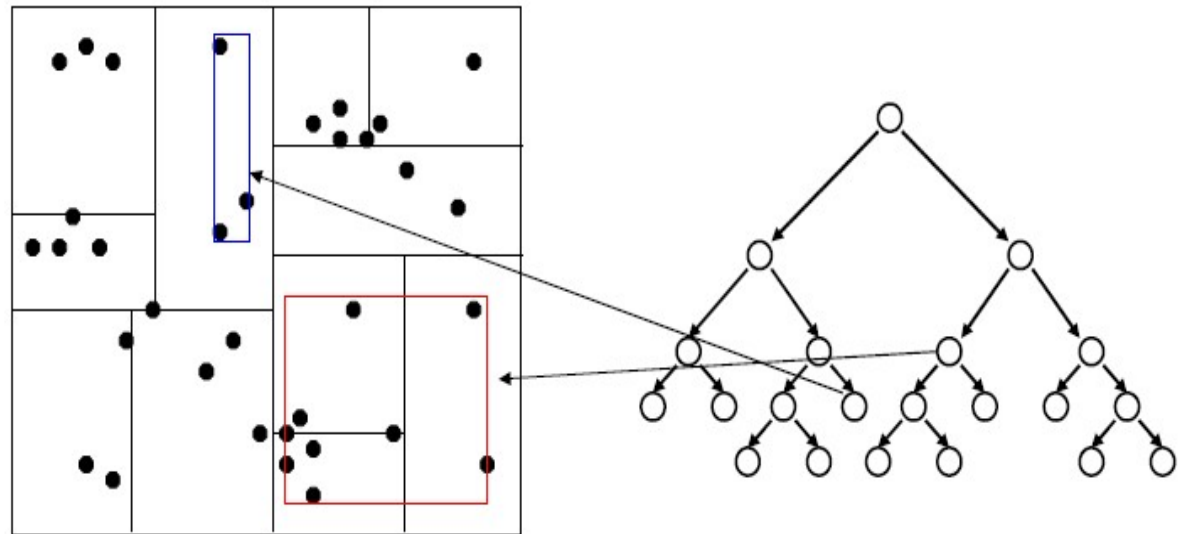
Dimension Reduction? (Cont'd)

- The earlier example was a linear method. How about non-linear methods?
- ISOMAP and LLE rely on k-NN anyway



Acceleration Structures

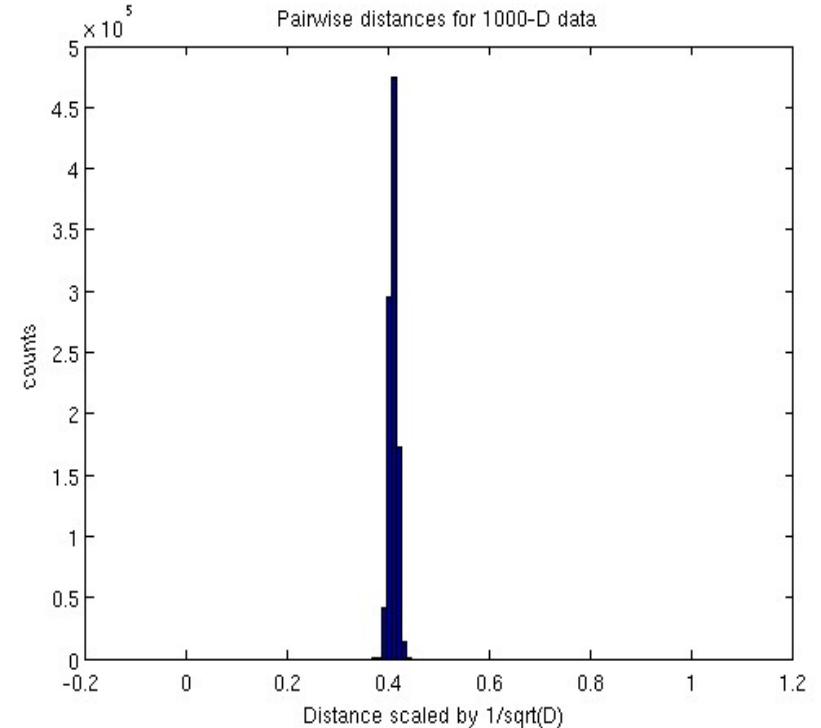
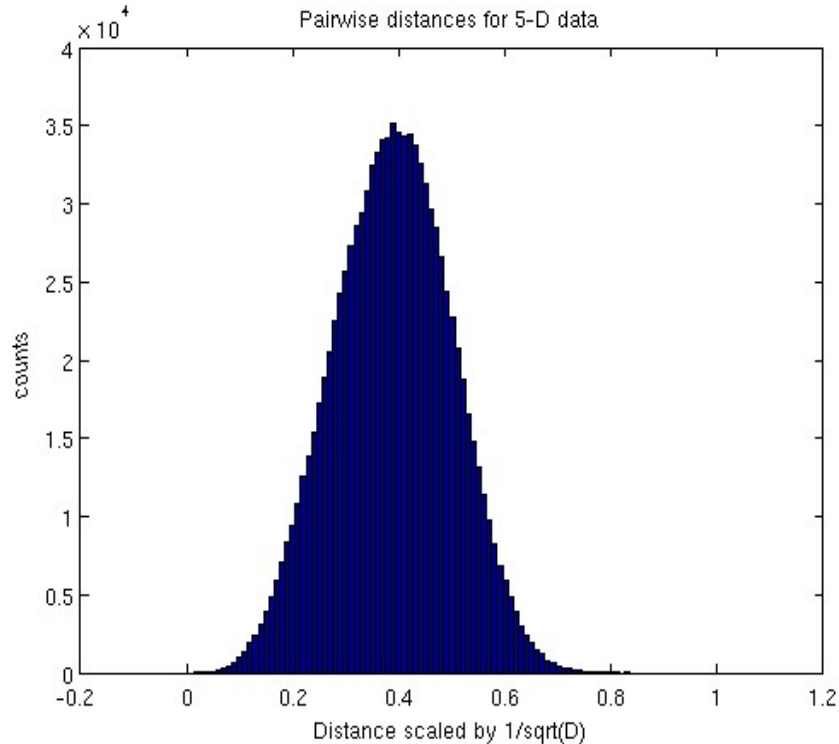
- Use efficient data structures to deal with multiple k-NN queries
 - kd-trees
 - R-trees
 - Metric trees
 - Ball trees



For more info, see: <http://www-2.cs.cmu.edu/~agray/icml.html>

“Acceleration” Structures

- These can give lower performance than brute-force methods on high dimensional datasets.



k-Parts of Talk

- Working in High Dimensions
- **Exact k-NN**
 - Ball Trees: “KNS1”
 - Two Ball Trees: “KNS2”
 - Fewer strings attached: “KNS3”
- Approximate k-NN
- Experiments

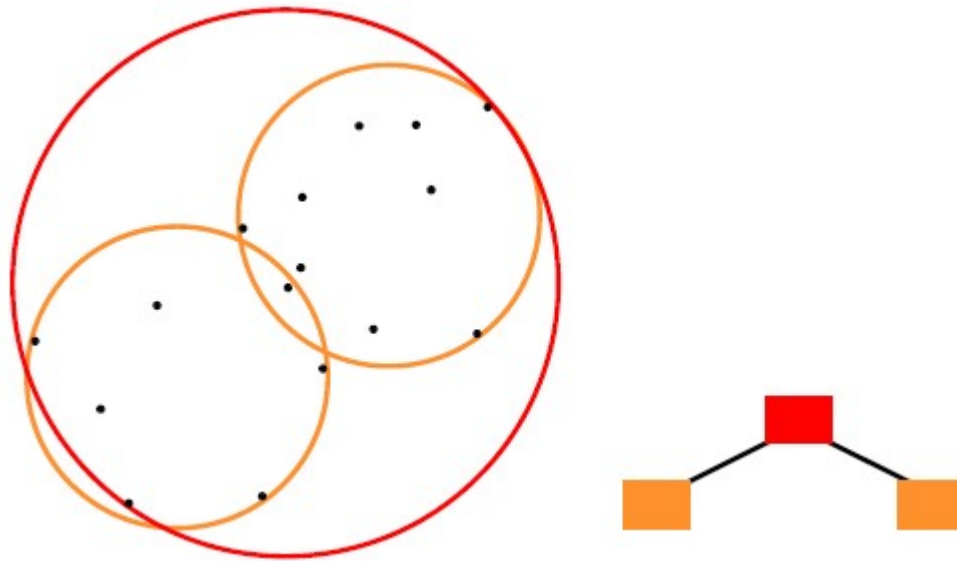
Key Idea: Ask the Right Question

Exact k-NN has some flexibility. What do we really want to answer?

- 1) *“What are the k nearest neighbors of \mathbf{q} ?”*
 - 2) *“How many of the k nearest neighbors of \mathbf{q} are from the positive class?”*
 - 3) *“Are at least f of the k nearest neighbors of \mathbf{q} from the positive class?”*
- Use Ball Trees in various ways

What Is a Ball Tree?

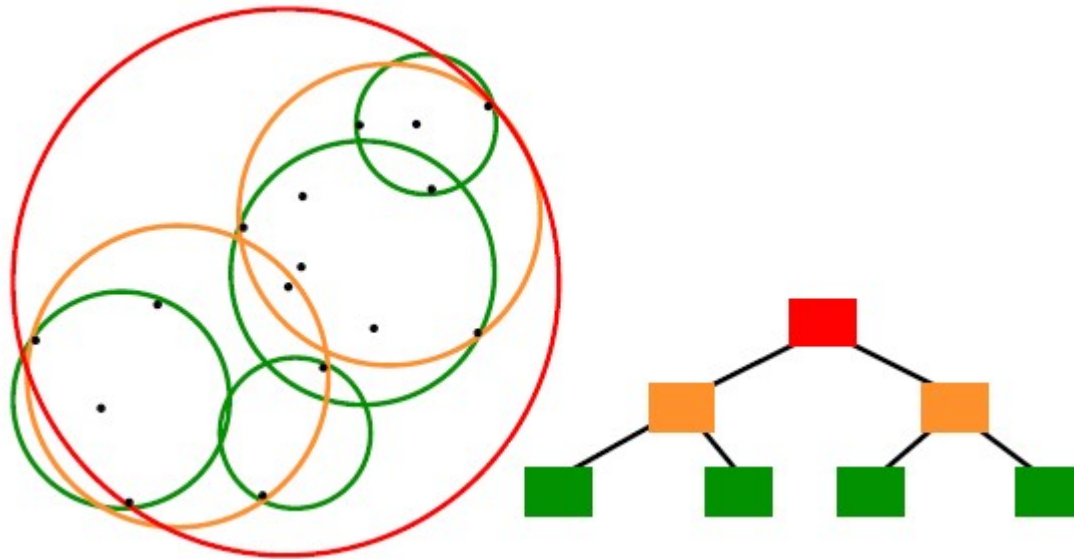
- It is a spatial binary tree (2 kids per node)
- Each node is a sphere that contains points.



Ball Tree – Child Relationships

- Child i is responsible for point \mathbf{x} if \mathbf{x} is closer to the center of child i

$$\mathbf{x} \in \text{Points}(\text{Node.child1}) \Rightarrow |\mathbf{x} - \text{Node.child1.C}| \leq |\mathbf{x} - \text{Node.child2.C}|$$
$$\mathbf{x} \in \text{Points}(\text{Node.child2}) \Rightarrow |\mathbf{x} - \text{Node.child2.C}| \leq |\mathbf{x} - \text{Node.child1.C}|$$

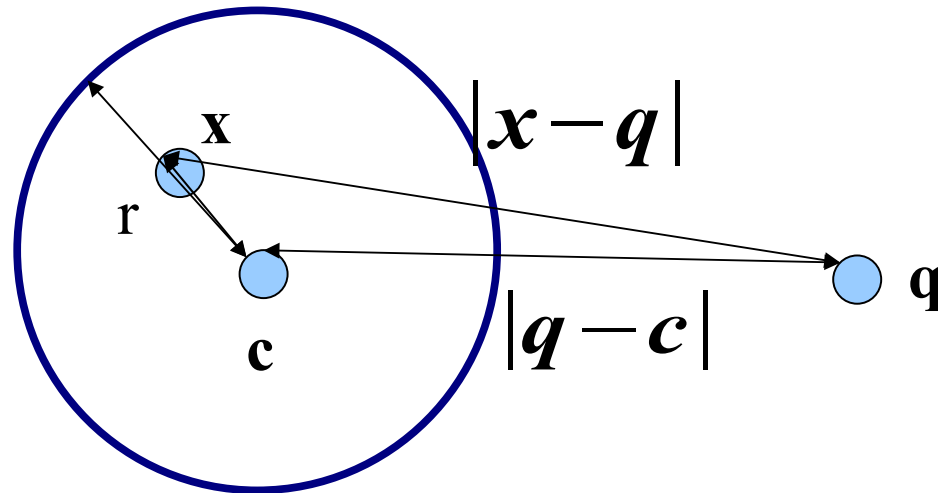


Ball Tree – Distance Guarantees

Given query point q , if $x \in Points(Node)$

$$|q - c| - r \leq |x - q| \leq |q - c| + r$$

$$|q - c| \leq |x - q| + |x - c| \quad |x - q| \leq |q - c| + |x - c|$$



KNS1: Using Ball Trees to find k-NN

Answers question “What are the k-NN of \mathbf{q} ?”

Define the recursive procedure

$$PS^{out} = \text{BallKNN}(q, PS^{in}, \text{Node})$$

BallKNN : Point \times PointSet \times Node \rightarrow PointSet

1) Invariants:

Let $V =$ visited points, then

- $PS^{in} =$ k-NN of query point q in V
- $PS^{out} =$ k-NN of q in $V \cup \text{Points}(\text{Node})$.

2) Initial call is $\text{BallKNN}(q, \{ \}, \text{Root})$

Tools for Pruning During Search

- Let D_k be the distance between q and the current guess for the k^{th} nearest neighbor.
- Let D^{Node} be the minimum possible distance between any point in $Node$ and q . Using the earlier result:

$$D^{\text{Node}} = \begin{cases} \max(|\mathbf{q} - \text{Node.C}| - \text{Node.R}, D^{\text{Node.parent}}) & \text{if } \text{Node} \neq \text{Root} \\ \max(|\mathbf{q} - \text{Node.C}| - \text{Node.R}, 0) & \text{if } \text{Node} = \text{Root} \end{cases}$$

BallKNN Algorithm at leaf nodes

BallKNN (\mathbf{q} , PS^{in} , Node)

if($D^{node} \geq D_k$) *return* PS^{in}

else if(Node is a **LEAF**)

$PS^{out} := PS^{in}$

for each \mathbf{x} *in* Points(Node)

if($|\mathbf{x} - \mathbf{q}| < D_k$)

add \mathbf{x} *to* PS^{out}

if($|PS^{out}| = k+1$)

remove furthest neighbor from PS^{out}

update D_k

else // (node is not a leaf)...

BallKNN Algorithm (non-leaf nodes)

else // (node is not a leaf)

node1 := child of Node closest to \mathbf{q}

node2 := child of Node further from \mathbf{q}

$PS^{temp} := \text{BallKNN}(\mathbf{q}, PS^{in}, \text{node1})$

$PS^{out} := \text{BallKNN}(\mathbf{q}, PS^{temp}, \text{node2})$

k-Parts of Talk

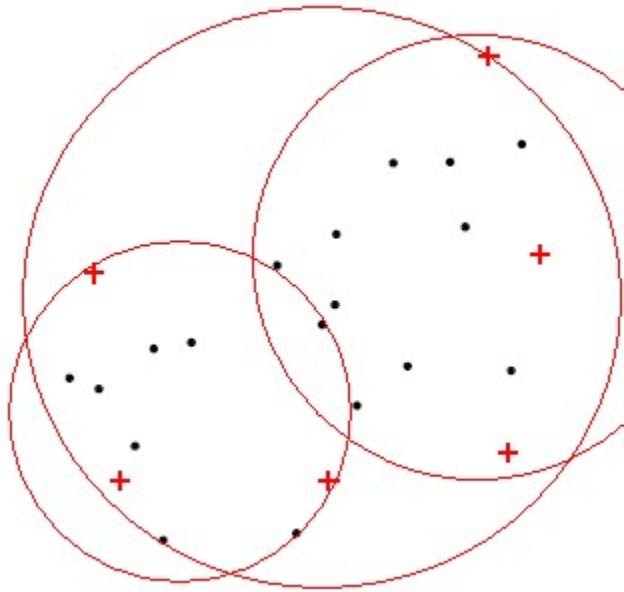
- Working in High Dimensions
- **Exact k-NN**
 - Ball Trees: “KNS1”
 - Two Ball Trees: “KNS2”
 - Fewer strings attached: “KNS3”
- Approximate k-NN
- Experiments

New Question, New Algorithm

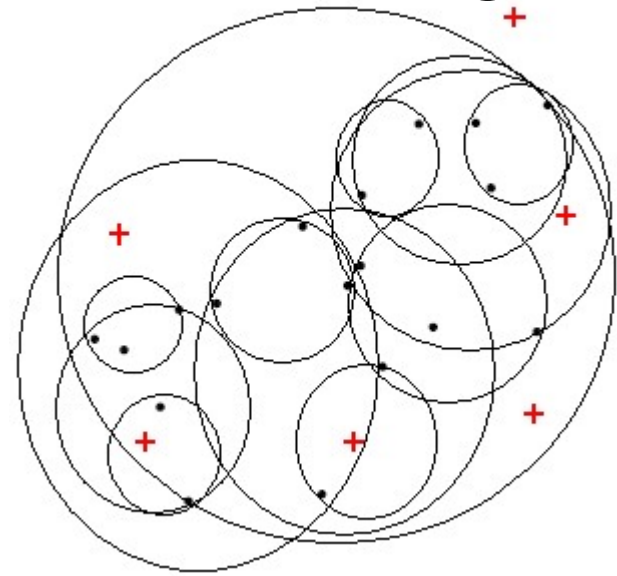
- Assume a classification setting
- Answer the question: *"How many of the k nearest neighbors of \mathbf{q} are from the positive class?"*
- There's room for optimization when classes are skewed!
- Skewness can arise from
 - Anomaly detection
 - Multi-class converted to two class problems

KNS2: Two Classes, Two Ball Trees

1) Find exact k rare NN
with full BallKNN



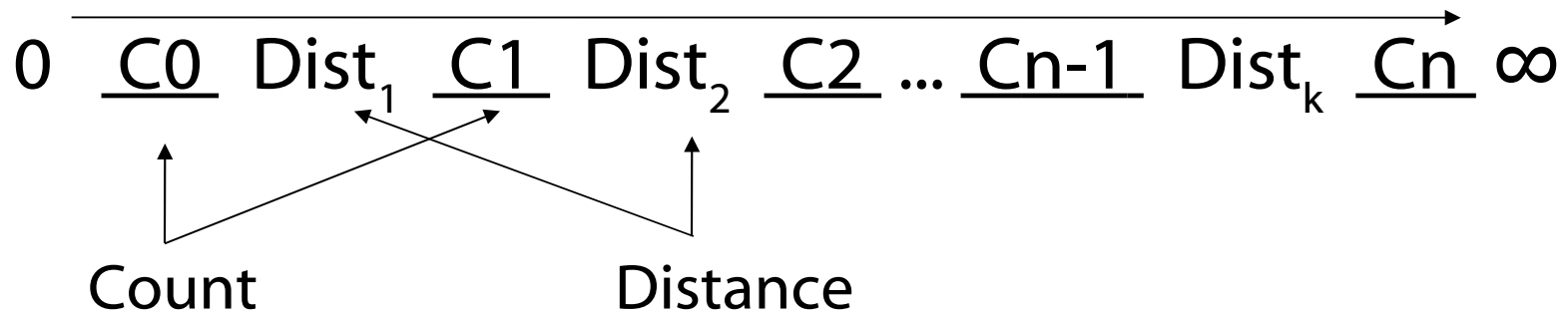
2) Count nearer
common neighbors
(more pruning)



Tools for Counting in Step 2

- Let $Dist_1 \dots Dist_k$, be the distances to k nearest rare neighbors
- Let $V =$ visited points in negative ball
- Let $C_i = \#$ of common neighbors \mathbf{x} s.t.

$$Dist_i \leq d(\mathbf{q}, \mathbf{x}) \leq Dist_{i+1}$$



How Many C_i ?

- Let $n+1$ be the number of C_i ($C_0 \dots C_n$)
 - $n \leq k$
- The algorithm starts by assuming that all common neighbors \mathbf{x} are farther than rare neighbors ($d(\mathbf{q}, \mathbf{x}) \geq Dist_k$).
 - $C_0 = C_1 = \dots = C_{n-1} = 0; \quad C_n = |C|$
- We are only interested in tracking k of the nearest neighbors, so n should decrease as more searched.

$$\sum_{i=0}^{n-1} C_i < k$$

Counting in Step 2

Another recursive procedure:

$$(n^{out}, C^{out}) = ComCount(q, n^{in}, C^{in}, Node, Dists)$$

$$ComCount : Point \times \mathbb{N} \times \mathbb{N} \times \mathbb{R}^k \rightarrow \mathbb{N} \times \mathbb{N}$$

Invariants:

- (n^{in}, C^{in}) summarize common counts for V
- (n^{out}, C^{out}) summarize common counts for $V \cup Points(Node)$.

Initial call is $ComCount(q, k, C^0, ComTree.Root, Dists)$

ComCount Algorithm (init, prune)

ComCount ($\mathbf{q}, n, C, \text{Node}, \text{Dists}$)

$n^{\text{out}} := n; C^{\text{out}} := C$

Let $T = \sum_{i=0}^{n-1} C_i$

(num. Com. points closer than n^{th} pos. point)

if($D^{\text{node}} \geq \text{Dist}_n$) return ($n^{\text{out}}, C^{\text{out}}$)

else if...

ComCount Algorithm (leaf)

*else if(Node is a **LEAF**)*

for each \mathbf{x} in Points(Node)

use binary search to find j in $[0, n^{\text{out}}]$, s.t.

$$\text{Dist}_j \leq |\mathbf{x} - \mathbf{q}| < \text{Dist}_{j+1}$$

$$C_j^{\text{out}} := C_j^{\text{out}} + 1 ; T := T + 1$$

if ($T > k$) decrement n^{out} until $T = \sum_{i=0}^{n^{\text{out}}-1} C_i^{\text{out}} < k$

$$\text{Dists}_{n^{\text{out}}+1} := \infty$$

if($n^{\text{out}} = 0$) return $(0, C^{\text{out}})$

return $(n^{\text{out}}, C^{\text{out}})$

else // (node is not a leaf) ...

ComCount Algorithm (non-leaf)

else // (node is not a leaf)

node1 := child of Node closest to \mathbf{q}

node2 := child of Node further from \mathbf{q}

$(n^{temp}, C^{temp}) := \text{ComCount}(\mathbf{q}, n, C, \text{node1}, \text{Dists})$

if($n^{temp} = 0$)

return $(0, C^{out})$ //not C^{temp} ? Doesn't matter

$(n^{out}, C^{out}) := \text{ComCount}(\mathbf{q}, n^{temp}, C^{temp}, \text{node2}, \text{Dists})$

When Is This Faster Than KNS1?

- Stops when k nearer com. instances found.
 - Not necessarily the k nearest!
 - Happens frequently with skewed datasets, and the test instances fall in the common class
- As n gets smaller, compare node against a closer rare point as opposed to only using the k^{th} point

KNS3 Briefly

- Compared to KNS2, it can be efficient without skewed classes
- Answers the question "*are at least f nearest neighbors positive?*"
 - e.g. $f = \lceil k/2 \rceil$
- See authors' journal paper (7 pages just for KNS3)!

k-Parts of Talk

- Working in High Dimensions
- Exact k-NN
 - Ball Trees: “KNS1”
 - Two Ball Trees: “KNS2”
 - Fewer strings attached: “KNS3”
- Approximate k-NN
- Experiments

Approximate k-NN

- Attempts to find neighbors within $(1 + \epsilon)$ of the true k nearest.
- Apparently, nearest neighbors are found early with these data-structures, but most of the time is spent proving that the solutions are indeed the nearest.

Approximate NN: One Method

- Build a (metric) tree called a Spill Tree.
- Similar to Ball Trees, but a point can belong to both children of a node if it's near the border.
- Only search child closest to the query point
 - If the query point is not near a decision boundary, this is ok.
 - If the query point is near a decision boundary, possible neighbors will be in both subtrees anyway!

k-Parts of Talk

- Working in High Dimensions
- Exact k-NN
 - Ball Trees: “KNS1”
 - Two Ball Trees: “KNS2”
 - Fewer strings attached: “KNS3”
- Approximate k-NN
- Experiments

Experiments (Exact k-NN)

- Run k-NN classifiers
 - Divide datasets a la 10-fold cross validation
- Goal is measure speedup
 - # distance calculations
 - Wall clock time
- Compare brute-force, KNS1, KNS2, and KNS3
 - Vary dataset
 - Vary dimension within a dataset (using PCA)
 - Vary k ($k = 9$ or $k = 101$)

Tasks and Datasets

- Life Science: two sparse datasets called “ds1” and “ds2”.
 - Similar to the National Cancer Institute Open Compound Database.
 - O=C(NC1=CC(=CC=C1)C(=O)NC2=CC=C(C=C2)C3=NCCN3)C4=CC=C(C=C4)C5=NCCN5
 - etc...
 - Varied dimension by PCA (10, 100, and original)

Tasks and Datasets (cont'd)

- Link Detection
 - Given a list of collaborators, did person P also take part in the work? Yes or no?
 - Citeseer: Predict whether J_Lee was a collaborator for each work based on other authors. Processed with PCA to 100-D.
 - IMDB: Same, but with Mel Blanc and movies

Tasks and Datasets (cont'd)

- UCI/KDD
 - Needed to transform/clean data
 - Convert multi-class to two-class datasets
 - e.g. A and not-A in the letter classification task
 - Convert categorical attributes with 1-of-n encoding
 - e.g., NBA Team = {76ers, Blazers, Bobcats, ...} replaced with a 30-bit number. Only one bit can be set at a time.

Datasets (Dimensions, Skew)

Dataset	Num. records	Num. Dimensions	Num. pos.	Dataset	Num. records	Num. Dimensions	Num. pos.
ds1	26733	6348	804	ds1.10pca	26733	10	804
ds1.100pca	26733	100	804	ds2.100anchor	88358	100	211
ds2	88358	1100000	211	J_Lee.100pca	181395	100	299
Letter	20000	16	790	Blanc__Mel	186414	10	824
Movie	38943	62	7620	Kdd99(10%)	494021	176	97278
Ipums	70187	60	119				

- Skew (greatest to least):
 - J_Lee, Ipums, ds2, Blanc, ds1, Letter, Movie, KDD
 - rare/total: 0.0016 to 0.1969

Results

		NAIVE		KNS1		KNS2		KNS3	
		dists	time (secs)	dists speedup	time speedup	dists speedup	time speedup	dists speedup	time speedup
ds1	k=9	6.4×10^8	4830	1.6	1.0	4.7	3.1	12.8	5.8
	k=101			1.0	0.7	1.6	1.1	10	4.2
ds1.10pca	k=9	6.4×10^8	420	11.8	11.0	33.6	21.4	71	20
	k=101			4.6	3.4	6.5	4.0	40	6.1
ds1.100pca	k=9	6.4×10^8	2190	1.7	1.8	7.6	7.4	23.7	29.6
	k=101			0.97	1.0	1.6	1.6	16.4	6.8
ds2	k=9	8.5×10^9	105500	0.64	0.24	14.0	2.8	25.6	3.0
	k=101			0.61	0.24	2.4	0.83	28.7	3.3
ds2.100-	k=9	7.0×10^9	24210	15.8	14.3	185.3	144	580	311
	k=101			10.9	14.3	23.0	19.4	612	248
J_Lee.100-	k=9	3.6×10^{10}	142000	2.6	2.4	28.4	27.2	15.6	12.6
	k=101			2.2	1.9	12.6	11.6	37.4	27.2
Blanc_Melk	k=9	3.8×10^{10}	44300	3.0	3.0	47.5	60.8	51.9	60.7
	k=101			2.9	3.1	7.1	33	203	134.0
Letter	k=9	3.6×10^8	290	8.5	7.1	42.9	26.4	94.2	25.5
	k=101			3.5	2.6	9.0	5.7	45.9	9.4
Movie	k=9	1.4×10^9	3100	16.1	13.8	29.8	24.8	50.5	22.4
	k=101			9.1	7.7	10.5	8.1	33.3	11.6
Ipums	k=9	4.4×10^9	9520	195	136	665	501	1003	515
	k=101			69.1	50.4	144.6	121	5264	544
Kddcup99 (10%)	k=9	2.7×10^{11}	1670000	4.2	4.2	574	702	4	4.1
	k=101			4.2	4.2	187.7	226.2	3.9	3.9

Remarks on Results

- KNS1 worse than brute-force on 1,000,000-D dataset (ds2)
- KNS2's largest time speedup is on KDD99 dataset, which had the lowest skew (but most samples)
- More speed up for $k = 9$ than $k = 101$
 - But, same time for brute-f at $k = 9$ and $k = 101$
- “dists speedup” vs time speedup

Questions? Comments?
