# *Telescopic Vector (TV) Trees*

*Access to point data in very large dimensional spaces should* be highly e cient.

*A document d may be viewed as a vector d~ of length k, where* the singular valued matrix, after decomposition, is of size (k k).

*Thus, each document may be thought of as a point in a k-* dimensional space.

*A document database may be thought of as a collection of such points, indexed appropriately.*

*When a user u presents a query Q, s/he is in e ect specifying,* a vector vec(Q) of length k. We must nd the p documents in the database that are maximally relevant to Q.

*This boils down to attempting to nd the k-nearest neighbors* present in the document database, of the query Q.

*The TV-tree is a data structure that borrows from R-trees in* this e ort.

*The TV-tree attempts to dynamically and exibly decide* how to branch, based on the data that is being examined. If lots of vectors all agree on certain attributes (e.g. if lots of documents all have many common terms), then we must or- ganize our index by branching on those terms (i.e. elds of the vectors) that distinguish between these vectors/documents.

# *Organization of a TV-Tree*

*NumChild: This is the maximal number of children that any* node in the TV-tree is allowed to have.

*: is a number, greater than 0 and less than k, called the number of active dimensions.*

*TV(k; NumChild; ) denotes TV-tree used to store k-dimensional data with NumChild as the maximal number of children, and*

*as the number of active dimensions.*

*Each node in a TV-tree represents a region. For this purpose,* each node N in a TV-tree contains three elds:

*1. N:Center: This represents a point in k-dimensional space.*

*2. N:Radius: This is a real number greater than 0.*

*3. N:ActiveDims: This is a list of at most dimensions.* Each of these dimensions is a number between 1 and k.

*Thus, N:ActiveDims is a subset of f1;::: ; kg of cardi-*

*nality or less,*

# *Region associated with a node N*

 *Suppose x and y are points in k-dimensional space, and ActiveDims is some set of active dimensions. The active distance between x and y, denoted act dist(x; y) is given by:*



*act dist(x; y) =*



*X x2 y2:*

*ti2ActiveDims*

*i*

*i*

*Here, xi and yi denote the value of the i'th dimension of x*

*and y, respectively.*

*EX: k = 200 and = 5 and the set ActiveDims = f1; 2; 3; 4; 5g.*

*Suppose:*

*x = (10; 5; 11; 13; 7; x6; x7;::: ; x200):*

*y = (2; 4; 14; 8; 6; y6; y7;::: ; y200):*

*Then the active distance between x and y is given by:*

*act dist(x; y) = r(10 2)2 + (5 4)2 + (11 14)2 + (13 8)2 + (7*

*p*



*= 100*

*= 10:*

*Node N represents the region containing all points x such* that the active distance (w.r.t. the active dimensions in N:ActiveDims) between x and N:Center is less than or equal to N:Radius.

*EX: If we had a node N with its center at*

*N:Center = (10; 5; 11; 13; 7; 0; 0; 0; 0;: :: ; 0)*

*and N:ActiveDims = f1; 2; 3; 4; 5g, then this node repre- sents the region consisting of all points x such that:*



*r(x1 10)2 + (x2 5)2 + (x 11)2 + (x4 13)2 + (x5 7)2 N:Ra*

*We use the notation Region(N ) to denote the region repre-* sented by a node N in a TV-tree.

*N also contains an array, Child of NumChild pointers to other nodes of the same type.*

# *Proprties of TV-Trees*

*All data is stored at the leaf nodes;*

*Each node in a TV-tree (except for the root and the leaves)* must be at least half full, i.e. at least half the Child pointers must be non-NIL.

*If N is a node, and N1;::: ; Nr are all its children, then*

*r*

*Region(N ) =*

*i=1*

*[*

*Region(Ni):*

# *Insertion into TV-trees*

*There are thre key steps.*

*1. Branch Selection: The rst operation is called branch* selection. When we insert a new vector into the TV-tree, and we are at node N (with children Ni, for 1 i NumChild), we need to determine which of these children to insert the key into.

*2. Splitting: The second approach is what to do, when we are* at a leaf node that is full and cannot accommodate the vector v we are inserting. This causes a split in that node.

*3. Telescoping: Suppose a node N is split into subnodes* N1; N2. In this case, it may well turn out that the vectors in Region(N1) all agree on not just the active dimensions of the parent N , but a few more as well. The addition of these extra dimensions is called telescoping. Telescoping may also involve the removal of some active dimensions, as we shall see later.

# *Example*

*5-dimensional space.*

*TV(5; 3; 2).*

*Space is a sphere centered at (0; 0; 0; 0; 0) with radius 50.*

*Initially, tree is empty.*

*Insert (5; 3; 20; 1; 5). This is handled straightforwardly by the* creation of a root node with

*1. Root:Center = (0; 0; 0; 0; 0);*

*2. Root:Radius = 50.*

*3. In this case, the root is also a leaf, with a pointer to the* information relevant to the point v1 = (5; 3; 20; 1; 5).

*4. Suppose Root:ActiveDims = f2; 3g.*

*See (a).*

*Insert v2 = (0; 0; 18; 42; 4).*

*Insert v3 = (0; 0; 19; 39; 6). At this stage, the root is \full".*

*Insert v4 = (9; 10; 2; 0; 16).*

*1. We must split the root.*

*2. Take the four vectors involved and \group" them together* into two groups, say. v1; v4 and v2; v3. See gure (d).

*3. Insert v5 = (18; 5; 27; 9; 9). Branch selection needed to* determine how to branch. See Figure 2(a).

*4. Insert v6 = (0; 0; 29; 0; 3). Again, we must perform branch* selection, and this time, we may choose to branch right, as shown in Figure 2(b).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (0,0,0,0,0) | | | 50 | 2,3 |
|  | |  | |  |
|  |  |
|  |  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (0,0,0,0,0) | | | 50 | 2,3 |
|  | |  | |  |
|  |  |  |  |
|  |  | |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (0,0,0,0,0) | | | 50 | 2,3 | |
|  | |  | |  | |
|  |  |  |  |  |  |
|  |  | |  | |  |

(5,3,20,1,5)

(5,3,20,1,5) (0,0,18,42,4)

(5,3,20,1,5) (0,0,18,42,4) (0,0,19,39,6)

v1 v1 v2 v1 v2 v3

(a) (b) (c)

LEGEND

S2

S1

2,3

50

(0,0,0,0,0)

|  |  |  |  |
| --- | --- | --- | --- |
| Center | | Ra | d ActiveD |
| child1 | child2 | | child3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (20,9,30,5,5) | | | 20 | 1,4 |
|  | |  | |  |
|  |  |  |  |
|  |  | |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (1,1,30,4,4) | | | 13 | 2 |
|  | |  | |  |
|  |  |  |  |
|  |  | |  | |

v1 v4 v2 v3

(d)

S2

S1

2,3

50

(0,0,0,0,0)

S2

S1

2,3

50

(0,0,0,0,0)

v1 v4 v5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (20,9,30,5,5) | | | 20 | 1,4 | |
|  | |  | |  | |
|  |  |  |  |  |  |
|  |  | |  | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (1,1,30,4,4) | | | 13 | 2 |
|  | |  | |  |
|  |  |  |  |
|  |  | |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (20,9,30,5,5) | | | 20 | 1,4 | |
|  | |  | |  | |
|  |  |  |  |  |  |
|  |  | |  | |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (1,1,30,4,4) | | | 13 | 2 | |
|  | |  | |  | |
|  |  |  |  |  |  |
|  |  | |  | |  |

v2 v3

v1 v4 v5

v2 v3 v6

# *Branch Selection*

*Consider node N with 1 j NumChild children, denoted N1;::: ; NNumChild.*

*expj(v) denotes the amount we must expand Nj:Radius so that v's active distance from Nj:Center falls within this ra- dius.*

*First, select all j's such that expj(v) is minimized.*

*EX: if we have nodes N1;::: ; N5 with exp values 10; 40; 19; 10; 32*

*respectively, the two candidates selected for possible insertion* are N1 and N4. If a tie occurs, as in the above case, pick the node such that the distance from the center of that node to v is minimized.

# *Splitting*

*When we attempt to insert a vector v into a leaf node N that* is already full, then we need to split the node.

*Wemust create subnodes N1; N2, and each vector in node N* must fall into one of the regions represented by these two subnodes.

*Split vectors in leaf N into two groups (G1; H1).*

*We may be able to enclose all vectors in G1 within a region* with center c1 and radius r1 and all vectors in H1 within a region with center c2 and radius r2.

*Many such splits are possible in general.*

*Split (G1; H1) is ner than split (G2; H2) i the sums of the*

*radii, (r1 + c0 ) is smaller than the sum of the radii (r2 + r0 ).*

*1 2*

*Still not enough to uniquely identify a \ nest" split.*

*If (G1; H1) and (G2; H2) are splits such that neither is ner than the other and no other split is ner than each of them, then we say that (G1; H1) is more conservative than (G2; H2) i*

*r + r 0*

*1*

*1*

*act dist(c1*

*; c0 ) r*

*+ r 0*

*act dist(c2*

*; c0 ):*

 *Split (G; H) is the selected split i :*

*1*

*2*

*2*

*2*

*1. there is no split (G0; H0) that is ner than (G; H) and*

*2. there is no split (G0; H0) that satis es condition (1) (i.e.* there exists no split (G ; H ) ner than (G0; H0)) and that is more conservative than (G; H).

# *Telescoping*

*Suppose N is the node into which we are to insert a vector* v.

*Insertion of v may cause two types of changes to N : it may* cause N to be split into two subnodes N1; N2, or it may

*\modify" the set of active dimensions of node N (e.g. if*

*vector v does not agree, on the active dimensions, with other* vectors stored at node N ).

*When node N gets split into two sub-nodes N1; N2, the set of* vectors at either node N1 or node N2 (but not both) must be a subset of the set of vectors at node N before the insertion.

*Suppose N1 has this property.*

*The vectors in N1 may agree not only on the active dimen-* sions of N , but on some other dimensions as well. In this case, we can expand the set of active dimensions of node N ,

*by adding these new dimensions. See gure below.*

N N

v1 v2

vr

d1, ... , ds

r

c

N2

v1 v2

vw

vw+1

vr v

d1, ... , ds

r2

c2

d1, ... , ds ,ds+1

r1

c1

d1, ... , ds

r

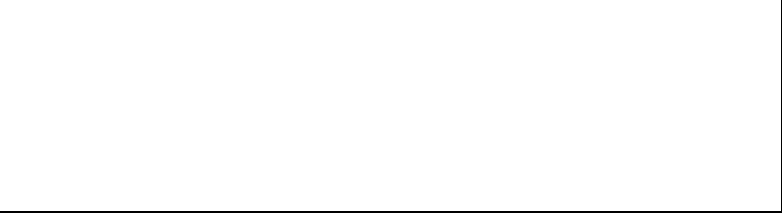
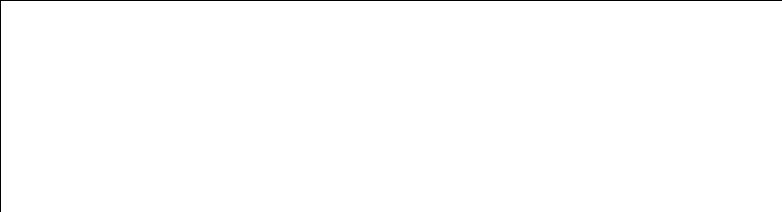
c

N1

*The other case when telescoping occurs is when a vector is* added to a node, N , but no split occurs. If N originally

*contained the vectors v1;::: ; vr and v is the vector being added, even though vectors v1;::: ; vr originally agreed on the active dimensions d1;::: ; ds of node N , they only agree now on a subset (for example, d2;::: ; ds) and hence, the set of active dimensions of node N must be contracted to re ect this fact.*

# *Searching in TV-Trees*



*Algorithm 2 Search(T,v)*

*if Leaf (T ) then Return (T:Center = v); Halt g else*

*f if v 2 Region(T ) then*

*NumChild*

*g*

*end*

*Return Wi=1 Search(T:Child[i]; v)*

*min(N; v) =*

*0 if v 2 Region(N )*

*act dist(v; N:Center) N:Radius otherwise*

*max(N; v) = act dist(v; N:Center)+ N:Radius:*

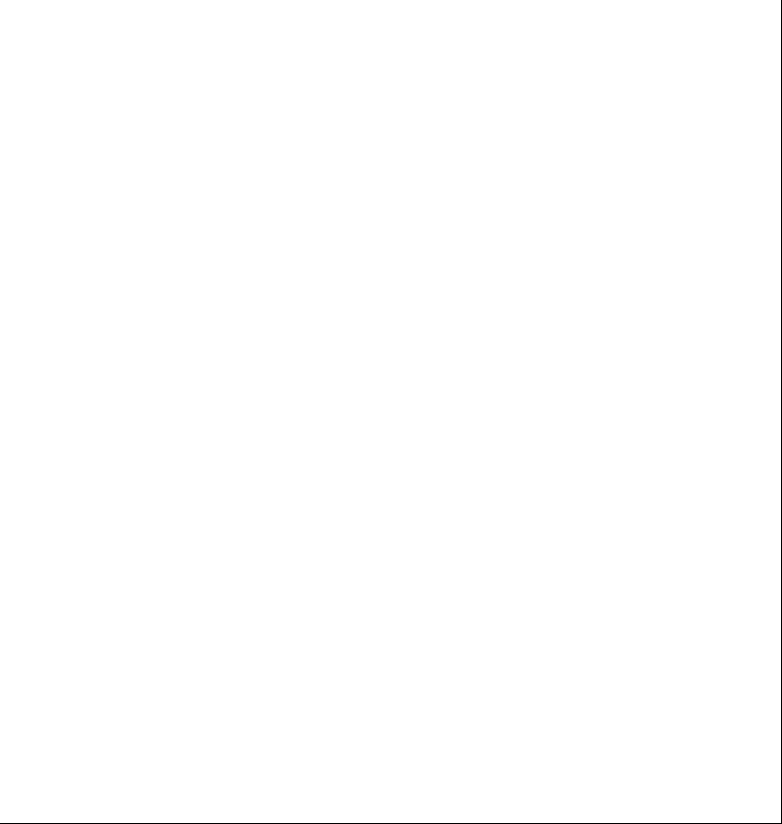
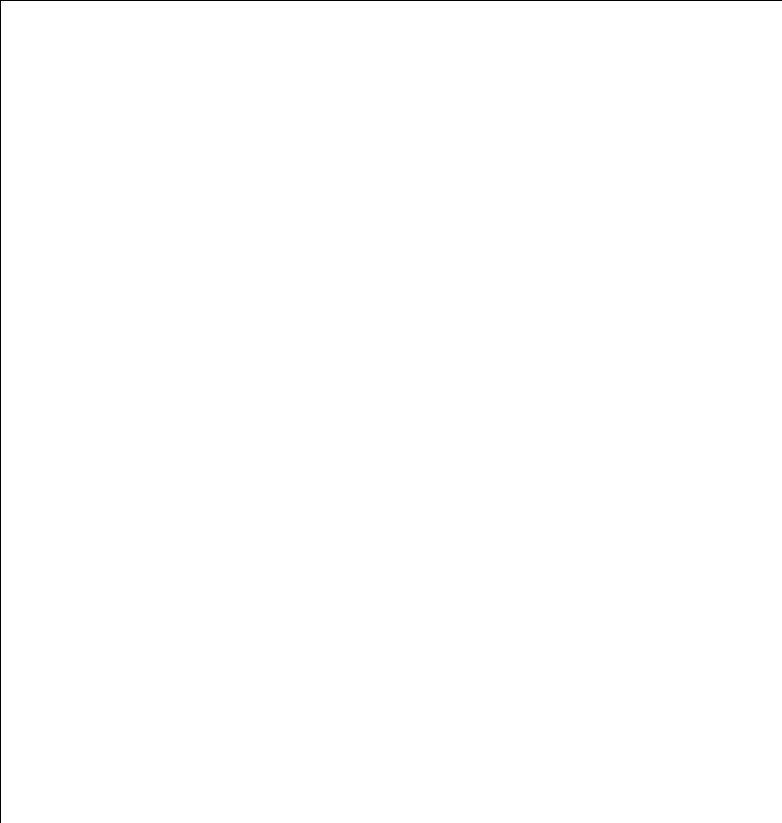
*Maintain an array SOL of length p,i.e. with indices running*

*1 through p.*

*The algorithm NNSearch uses a routine called Insert that takes as input, a vector vec, and an array SOL maintained in non-descending order of active distance from vec.*

*Insert returns as output, the array SOL with vec inserted* in it, at the right place, and with the p'th element of SOL eliminated.

# *(Contd.)*



*Algorithm 3 NNSearch(T,v,p)*

*for i = 1 to p do SOL[i]= 1; NNSearch1(T,v,p);*

*end (? end of program NNSearch ?)*

*procedure NNSearch1(T,v,p);*

*if Leaf (T )& act dist(T:val; v) < SOL[p] then*

*Insert T:val into SOL; else*

*f*

*if Leaf (T ) then r = 0;*

*else f Let N1;::: ; Nr be the children of T ;*

*Order the Ni's in ascending order w.r.t. min(Ni; v); Let N (1);::: ; N (r) be the resulting order;*

*g;*

*done = false; i = 1;*

*while ((i r) ^ :done) do f*

*NNSearch(N (i); v; p);*

*if SOL[p] < min(N (i+1); v) then done = true;*

*i = i + 1;*

*g; (? end of while ?) g (? end of else ?)*

*Return SOL;*

*end proc (? end of subroutine NNSearch1 ?)*