Fast NN Data Structures

$D = \{(x_i, y_i)\}_{i=1}^{n}$

$x_i \in \mathbb{R}^d$, y_i

$x' \longrightarrow D$

Training: $O(d)$

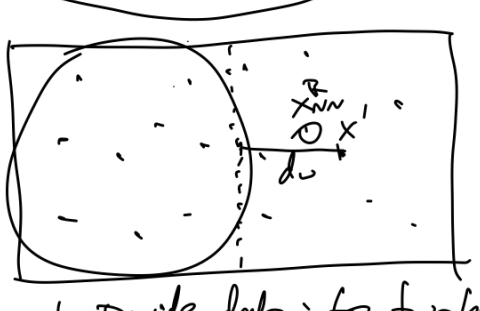
Testing: $O(dn)$

$\text{dist}(x', D)$

$\boxed{\text{Goal}}$: To make KNN faster during testing!!!!

Idea: Use clever data structures!

KD-Trees



a) Identify which half the test point x' lies in (R)

b) Find x_{NN}^R in R to x'

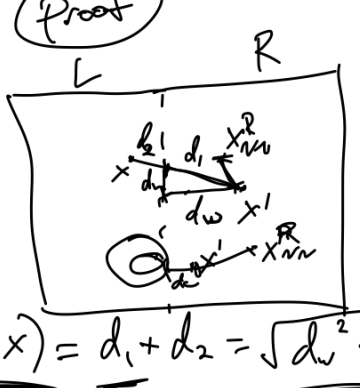
c) Compute d_w

$d_w > d(x', x_{NN}^R)$

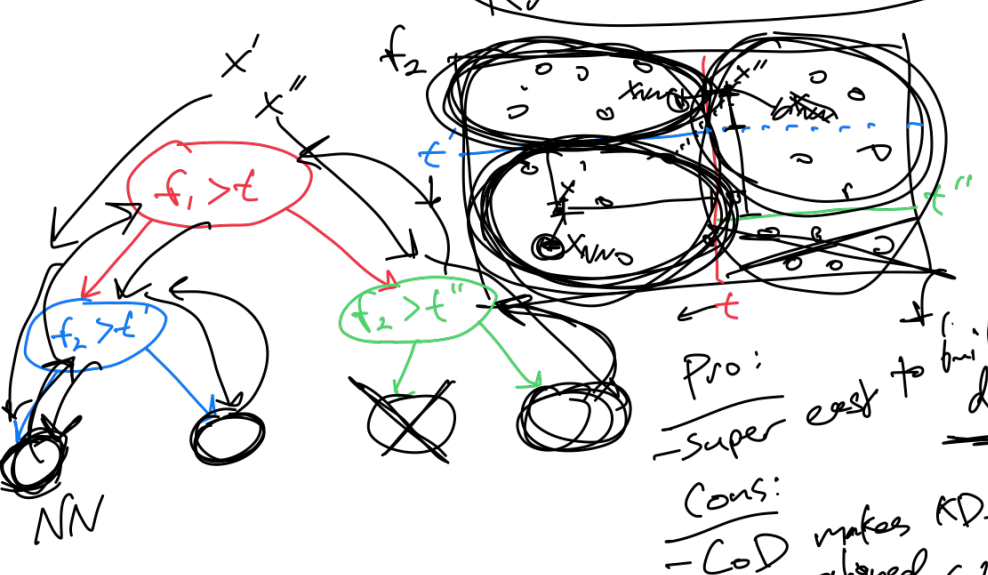
1. Divide data into two halves along one feature
2. For each training point, keep track of which half it's in

Claim: If $d_w > d(x', x_{NN}^R)$ then x_{NN}^R is closer than any other point x in L; thus is the true NN of x'

Proof



KD-Tree Data Structure



Pro:
- super easy to build

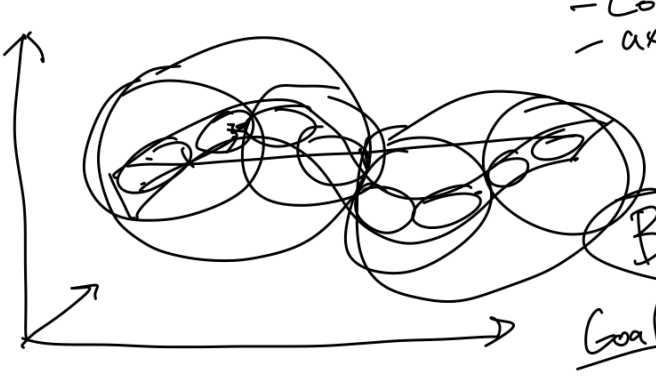
Cons:
- CoD makes KD-Trees ineffective in high dimensions
- axis-aligned splits

$d(x', x) = d_1 + d_2 = \sqrt{d_w^2 + d_\perp^2} + d_2$

$\geq \sqrt{d_w^2} + d_2$

$= d_w + d_2$

$\geq d_w > d(x', x_{NN}^R)$

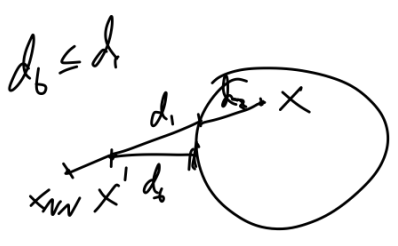


Approx:
- Search m leafs $\longrightarrow$ return closest neighbor

Ball Trees

Goal: Find direction w/ a lot of spread in data

Idea:



$d(x', x) = d_1 + d_2$

$\geq d_b + d_2$

$\geq d_b > d(x', x_{NN})$

Ball Tree Building Algorithm

Input: $S = \{(x_i)\}_{i=1}^{n}$

k (i.e. # of neighbors)

```
function BallTree(S, k)
    if |S| < k then stop end if
    pick x_0 ∈ S uniformly at random
    pick x_1 = argmax_{x∈S} d(x_0, x)
    pick x_2 = argmax_{x∈S} d(x_1, x)
    ∀i = 1, ..., |S|  z_i = (x_1 - x_2)^T x_i
    m = median(z_1, ..., z_|S|)
    S_L = {x_i ∈ S | z_i < m}
    S_R = {x_i ∈ S | z_i ≥ m}
    return tree: - center c = mean(S)
                 - radius r = max_{x∈S} d(x, c)
                 - children
                     · BallTree(S_L, k)
                     · BallTree(S_R, k)
end
```