Learning Partition Trees for Nearest Neighbor Search

VecDB @ VLDB 2026

Sanjeev Khanna[1] · Ashwin Padaki[2] · Erik Waingarten[2]

[1]New York University [2]University of Pennsylvania

1 Learn from your queries

Build an index for the queries you actually get.

- Benchmarks measure **recall vs QPS** on fixed query sets
- This assumes online performance ≈ offline performance

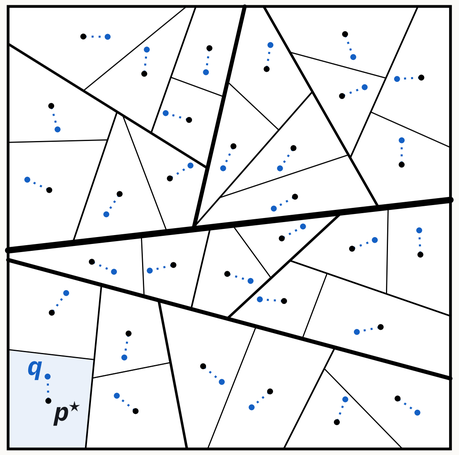
Given *samples* from the query distribution, can we compete with the *optimal* ANN index within a certain class?

GIVEN

- A dataset $P \subset \mathbb{R}^d$
- Training pairs $(\boldsymbol{q}, \boldsymbol{p}^\star)$, where $\boldsymbol{p}^\star = \mathrm{NN}(\boldsymbol{q})$

GOAL

- Find a **balanced halfspace tree** of minimal *search cost* $= \mathop{\mathbb{E}}_{(\boldsymbol{q},\boldsymbol{p}^\star)} \left[|\mathrm{LCA}_T(\boldsymbol{q}, \boldsymbol{p}^\star)|\right]$



A perfect tree (thickness ↔ tree depth). Search cost is 1.

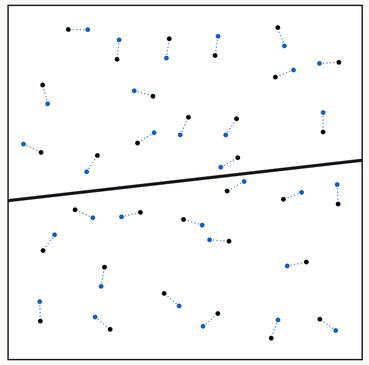
2 Good halfspaces are hard to find

Finding a low-error balanced halfspace is NP-hard.

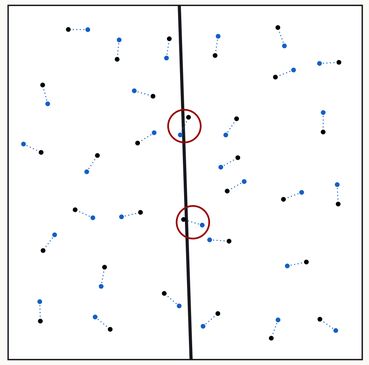
BALANCED HALFSPACE CUT PROBLEM

Find a halfspace that

- keeps $\geq \beta$ of the dataset on each side
- minimizes the fraction of pairs that are **cut**



No pairs cut: error 0



Two pairs cut: error 2/32

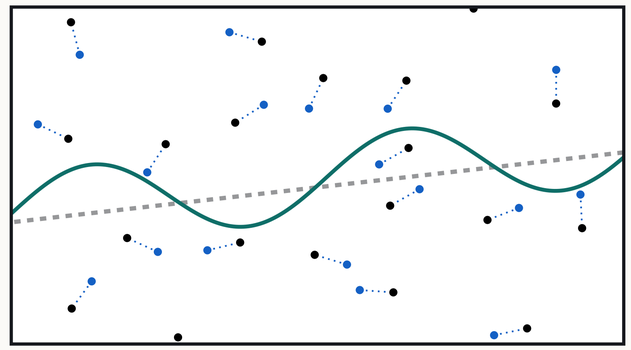
THEOREM 1

It is **NP-hard** to distinguish between:

- there exists a halfspace with 1/2 balance and 0 error
- every halfspace of $\geq 1/3$ balance has $\geq 1\%$ error

3 The polynomial fix

Output a polynomial cut that matches the best halfspace.



THEOREM 2

If the best halfspace has error α, PolyCut outputs a polynomial cut of degree $\widetilde{O}(1/\varepsilon^2)$ and error $O(\sqrt{\alpha + \varepsilon})$.

We assume that $\boldsymbol{q}, \boldsymbol{p}^\star$ are Gaussian-like.

WHY A POLYNOMIAL?

- Halfspace error is challenging because $H(\boldsymbol{q}) \in \{-1, 1\}$
- Instead, we find a polynomial A such that $A(\boldsymbol{q}) \approx H(\boldsymbol{q})$ (we use that $\boldsymbol{q}$ is Gaussian-like)
- We minimize a tractable proxy: $\mathbb{E}\left[(A(\boldsymbol{q}) - A(\boldsymbol{p}^\star))^2\right]$

4 The punchline: beating brute force

We efficiently compute a balanced halfspace tree that solves exact NNS faster than brute force.

MAIN THEOREM

GIVEN

- a dataset of n points in $\mathbb{R}^d$
- samples access to queries

WE COMPUTE IN POLY TIME

an $O(nd)$ space index that:

- answers query $\boldsymbol{q}$ in $o(nd)$ time
- returns $\mathrm{NN}(\boldsymbol{q})$ with prob. 0.99

ASSUMING

- $(\boldsymbol{q}, \boldsymbol{p}^\star)$ admits a perfect 1/3-balanced halfspace tree
- $\boldsymbol{q}, \boldsymbol{p}^\star$ are Gaussian-like

OPEN PROBLEMS

- Improve cut runtime: $d^{\mathrm{poly}(1/\varepsilon)} \to \mathrm{poly}(d, 1/\varepsilon)$?
- Learn the best **graph-based** index
- Design a practical data structure using these ideas