A Theory-Based Evaluation of Nearest Neighbor Models Put Into Practice

**Hendrik Fichtenberger and Dennis Rohde**

**PART OF DATA PROCESSING PIPELINE**

- given: point-set in euclidean space
- build
- nearest neighbor model
- output

**Query Complexity**

- Testing k-nearest Neighborhood:
  - $\Theta(n^d \log n)$ queries sufficient in bounded average-degree graphs
  - $\Omega(n^{(d/2)} + k)$ queries required in general graphs

**Property Testing Algorithm**

- accepts every k-nearest neighborhood graph with high probability
- rejects every graph that is $\epsilon$-far from being a k-nearest neighborhood graph with high probability
  - at least an $\epsilon$-fraction of edges are faulty
- can freely decide otherwise

**Our Algorithm**

- sample $\Theta(n \epsilon^{-d})$ vertices uniformly at random
- throw away vertices with high degree
- sample $\Theta(n \epsilon^{-d})$ vertices uniformly at random
- for every vertex in first sample check if any vertex from second sample lies nearer than any neighbor

**Example Graphs**

- brute force 3-nn graph
- graph from Annoy(Spotify) model
- 60 vertices are incident with faulty edges

**Experiments**

- recall of algorithm by $\epsilon$-distance of model
- all models were built ten times for each parameterization, then tested once
- constants of $O$-notation

**Our Code**

- The algorithm: [github.com/derohde/knn_test](https://github.com/derohde/knn_test)
- Extension of ann-benchmarks: [github.com/hfichtenberger/ann-benchmarks](https://github.com/hfichtenberger/ann-benchmarks)

**Models**

- AAALgo: [github.com/aaalgo/kgraph](https://github.com/aaalgo/kgraph)
- NMSLIB: [github.com/nmslib/nmslib](https://github.com/nmslib/nmslib)
- Annoy: [github.com/spotify/annoy](https://github.com/spotify/annoy)

**Extension of ann-benchmarks**

- [github.com/hichtenberger/ann-benchmarks](https://github.com/hichtenberger/ann-benchmarks)

**What is the quality of the output?**

1. implicit conversion of k-nn model to geometric graph
2. test if given graph is k-nearest neighbor graph
- directed edges
- regular (out-degree = k)
- edges point to k-nearest neighbors of a vertex

**no need to compute full graph**

**more details on arxiv**