Fast and Scalable Outlier Detection with Approximate Nearest Neighbor Ensembles

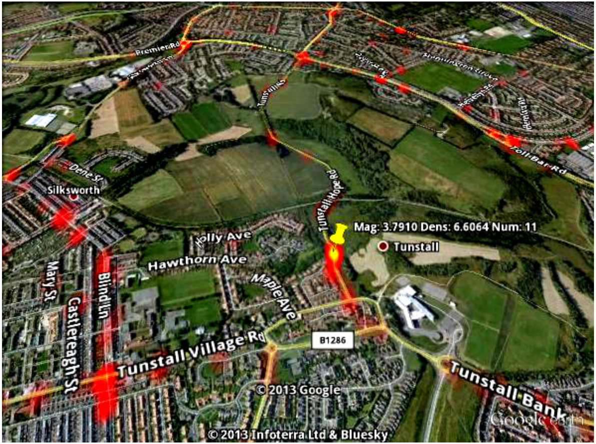
Erich Schubert, Arthur Zimek, Hans-Peter Kriegel

Lehr- und Forschungseinheit Datenbanksysteme
Institut für Informatik
Ludwig-Maximilians-Universität München

Hanoi, 2015-04-22

Outlier Detection – Use Cases

Outliers – Car crash hotspots (using KDEOS): [SZK14a]




Sunderland Echo

12/12/13 | 10°C to 13°C Light rain | Like us | Follow us | Place your Ad | Subscribe

LOCAL | NEWS LIVE | CRIME | BUSINESS | EDUCATION | HEALTH | POLITICS | NORTH EAST | NATIONAL

ALL NEWS | SUNDERLAND | HOUGHTON & HETTON | WASHINGTON | SEAHAM & EAST DURHAM | DURHAM CITY

Accident black spot road to be reopened

Published on the 26 October 2004 16:33

A "DEATHTRAP" road, sealed off after a string of accidents will finally reopen this week after a safety probe.

Five people have been taken to hospital and six cars written off on the same spot along Tunstall Hope Road, after it was resurfaced.

The road, between Tunstall and Ryhope, was closed more than a week ago after an investigation was launched into the crashes.

But one victim today said only "time would tell" whether the improvements would work.

Jacqueline Pattison, 47, had a lucky escape when her Fiat Punto skidded off Tunstall Hope Road, flipped on its side and smashed into a tree in September.

Print this

Sponsored by

New crash on 'danger road'

Updated on the 28 April 2005 18:15

Published 27/04/2005 10:56

TWO drivers have been injured in the latest smashes on a road branded one of the most dangerous in Sunderland.

Tunstall Hope Road was closed for more than two hours after a collision between a Ford Escort van and Toyota Yaris.

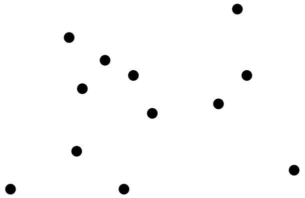
The accident comes after five people were taken to hospital and six cars written off after the road was resurfaced last year.

Print this

Using Open Data (7 years, 1.2 million accidents) from the UK.

Outlier Detection: kNN-Outlier

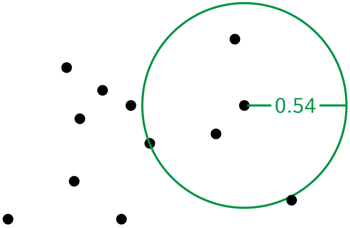
kNN outlier [RRS00]: $score(o) := k\text{-}dist(o)$ (here: $k = 3$)



Many outlier detections based on kNN and LOF [Bre+00].
Examples: [AP02; Jin+06; Kri+09; SZK14b]

Outlier Detection: kNN-Outlier

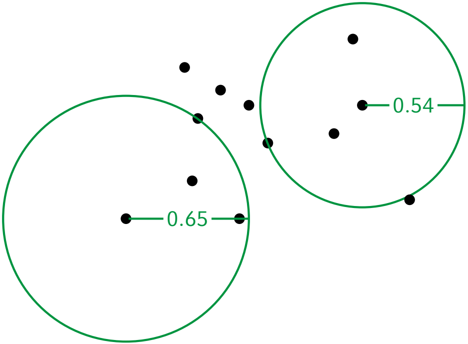
kNN outlier [RRS00]: $score(o) := k\text{-}dist(o)$ (here: $k = 3$)



Many outlier detections based on kNN and LOF [Bre+00].
Examples: [AP02; Jin+06; Kri+09; SZK14b]

Outlier Detection: kNN-Outlier

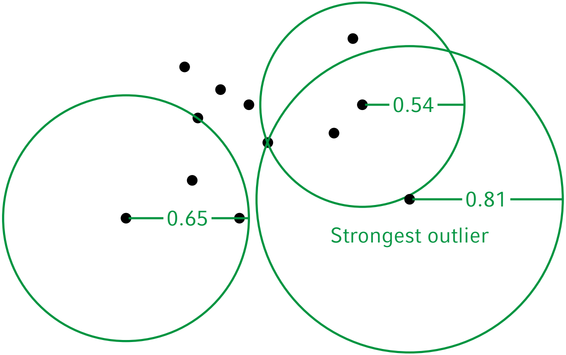
kNN outlier [RRS00]: $score(o) := k\text{-}dist(o)$ (here: $k = 3$)



Many outlier detections based on kNN and LOF [Bre+00].
Examples: [AP02; Jin+06; Kri+09; SZK14b]

Outlier Detection: kNN-Outlier

kNN outlier [RRS00]: $score(o) := k\text{-}dist(o)$ (here: $k = 3$)



Many outlier detections based on kNN and LOF [Bre+00].
Examples: [AP02; Jin+06; Kri+09; SZK14b]

Outlier Detection: Local Outlier Factor [Bre+00]

$$\mathrm{LOF}(o) := \underbrace{\frac{1}{|k\mathrm{NN}(o)|} \sum_{p \in k\mathrm{NN}(o)}}_{\text{average}} \underbrace{\frac{\mathrm{lrd}(p)}{\mathrm{lrd}(o)}}_{\text{relative density}}$$

where $\mathrm{lrd}(o)$ is the local reachability density:

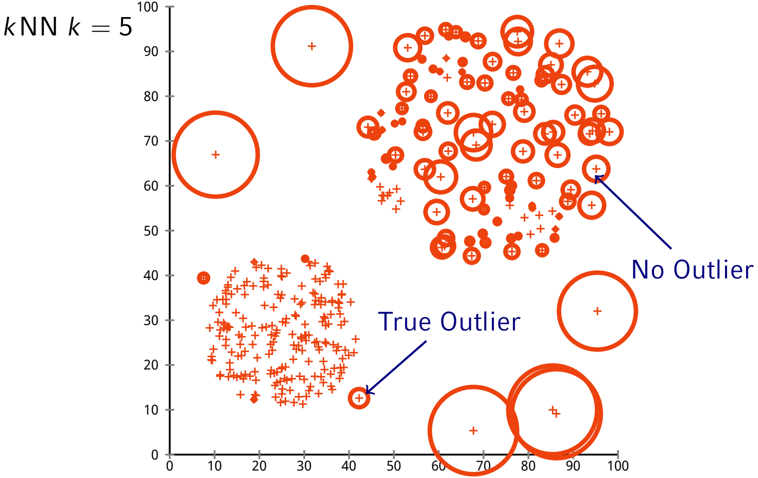
$$\mathrm{lrd}(o) := \underbrace{1 \Big/}_{\text{inverse}} \underbrace{\frac{1}{|k\mathrm{NN}(o)|} \sum_{p \in k\mathrm{NN}(o)}}_{\text{average}} \underbrace{\text{reach-dist}(o \leftarrow p)}_{\text{reachability distance}}$$

and the (asymmetric) reachability of o *from* p is:

$$\text{reach-dist}(o \leftarrow p) := \max\{\underbrace{\mathrm{dist}(o,p)}_{\text{true distance}}, \underbrace{k\text{-dist}(p)}_{\text{core size of neighbor}}\}$$

Outlier Detection: Local Outlier Factor [Bre+00]

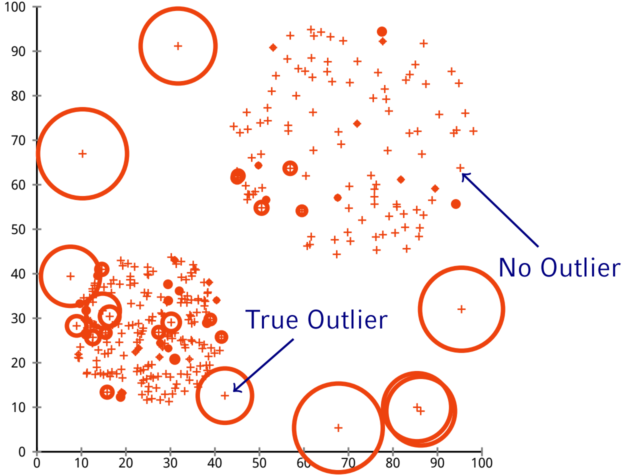
kNN has difficulties with *different densities*



Outlier Detection: Local Outlier Factor [Bre+00]

LOF is designed to cope with different densities

LOF $k = 5$



Outlier Detection

Many outlier detection methods are based on the k nearest neighbors.

Unfortunately, computing the kNN for large data is expensive:
Pairwise distance computation is $\mathcal{O}(n^2)$ – prohibitive for big data.

Outlier Detection

Many outlier detection methods are based on the k nearest neighbors.

Unfortunately, computing the kNN for large data is expensive:
Pairwise distance computation is $\mathcal{O}(n^2)$ – prohibitive for big data.

- R*-Tree [Bec+90] good up to $\approx$ 30 dimensions (best: $\leq$ 10), but not easy to *distribute* to a cluster.
- PINN [dCH10; dCH12]: random projections + kd-tree.
- LSH [IM98] may find less than k neighbors for outliers.

Outlier Detection

Many outlier detection methods are based on the k nearest neighbors.

Unfortunately, computing the kNN for large data is expensive:
Pairwise distance computation is $\mathcal{O}(n^2)$ – prohibitive for big data.

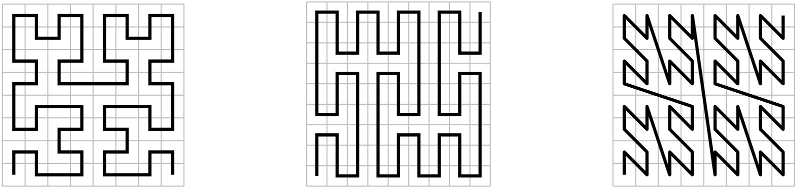
- R*-Tree [Bec+90] good up to ≈ 30 dimensions (best: ≤ 10), but not easy to *distribute* to a cluster.
- PINN [dCH10; dCH12]: random projections + kd-tree.
- LSH [IM98] may find less than k neighbors for outliers.

Wanted: an approximative approach to find the k nearest neighbors:

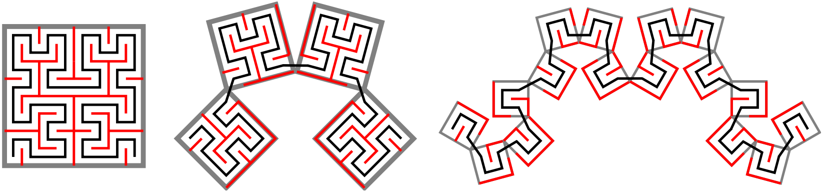
- High probability of finding the correct neighbors
- Errors should not hurt much
- Distributable to a cluster
- Supports high-dimensional data

Ingredients: Space-Filling Curves

Space-filling curves project multiple dimensions to one.
(Hilbert curve [Hil91], Peano curve [Pea90], and Z-curve [Mor66])

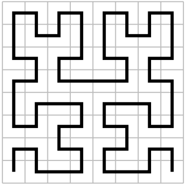


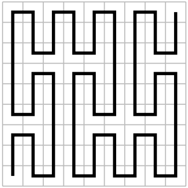
Neighbors remain neighbors on the curve with high probability.
Each curve has "cuts" where neighborhoods are not well preserved.

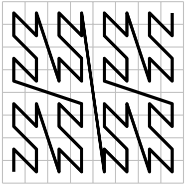


Ingredients: Space-Filling Curves

Space-filling curves project multiple dimensions to one.
(Hilbert curve [Hil91], Peano curve [Pea90], and Z-curve [Mor66])





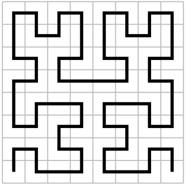


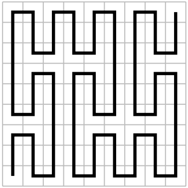
Neighbors remain neighbors on the curve with high probability.

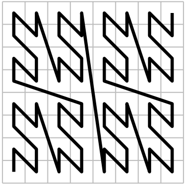
Distributed sorting large data is well understood.

Ingredients: Space-Filling Curves

Space-filling curves project multiple dimensions to one.
(Hilbert curve [Hil91], Peano curve [Pea90], and Z-curve [Mor66])







Neighbors remain neighbors on the curve with high probability.

However, they do not work well with too many dimensions either, because they split one dimension at a time.

We need more ingredients to improve the results.

Ingredients: Random projections (c.f. [dCH10])

Random projections can reduce the dimensionality, and preserve distances well (e.g. database-friendly [Ach01], p-stable [Dat+04]).

In contrast to other dimensionality reduction (PCA, MDS), these project one vector at a time and thus can be *distributed* easily.

Ingredients: Random projections (c.f. [dCH10])

Random projections can reduce the dimensionality, and preserve distances well (e.g. database-friendly [Ach01], p-stable [Dat+04]).

In contrast to other dimensionality reduction (PCA, MDS), these project one vector at a time and thus can be *distributed* easily.

Often, multiple projections are used and combined in an ensemble.

Objective: Design an ensemble based on random projections and space-filling curves, to find the k nearest neighbors.

- Distributable to a cluster with $\mathcal{O}(n)$ communication
- Different curves and projections avoid correlated errors

Ensemble for k-Nearest Neighbors

1. Generate m space-filling curves (with high diversity):
   - Different curve families (Peano, Hilbert, Z-Curve)
   - Random projections or random subspaces
   - Different shift offsets
2. Project the data to each space-filling curve
3. Sort the data for each space-filling curve
4. Use a sliding window of width $w \times k$ to generate candidates
5. Merge the neighbor candidates for each point
6. Compute the real distances, and keep the k nearest neighbors
7. If needed, also emit reverse k nearest neighbors

All steps can well be implemented on a cluster.
Except for sort and sliding window as "map" and "reduce".

Ensemble for *k*-Nearest Neighbors

2. Project the data to each space-filling curve

```
distributed on every node do
    // Blockwise I/O for efficiency
    foreach block do
        foreach curve do
            // Map to the SFC
            project data to curve
            // ...but delay the shuffle step
            store projected data locally
            // Sample data for sorting
            send sample to coordination node
        end
    end
endon
// Complete sort using sample distribution
```

Ensemble for k-Nearest Neighbors

1. Generate m space-filling curves (with high diversity):
   - Different curve families (Peano, Hilbert, Z-Curve)
   - Random projections or random subspaces
   - Different shift offsets
2. Project the data to each space-filling curve
3. Sort the data for each space-filling curve
4. Use a sliding window of width $w \times k$ to generate candidates
5. Merge the neighbor candidates for each point
6. Compute the real distances, and keep the k nearest neighbors
7. If needed, also emit reverse k nearest neighbors

All steps can well be implemented on a cluster.
Except for sort and sliding window as “map” and “reduce”.

Ensemble for k-Nearest Neighbors

4. Use a sliding window of width $w \times k$ to generate candidates

distributed on *every node* **do**
    `// Blockwise processing of sorted data`
    foreach *curve* **do**
        foreach *projected and sorted block* **do**
            `// "Map" each block to (object, neighbors)`
            foreach *object (using sliding windows of width $w \times k$)* **do**
                emit (object, neighbors in window)
            end
        end
    end
endon
shuffle to (object, neighbor list)

Ensemble for k-Nearest Neighbors

1. Generate m space-filling curves (with high diversity):
   - Different curve families (Peano, Hilbert, Z-Curve)
   - Random projections or random subspaces
   - Different shift offsets
2. Project the data to each space-filling curve
3. Sort the data for each space-filling curve
4. Use a sliding window of width $w \times k$ to generate candidates
5. Merge the neighbor candidates for each point
6. Compute the real distances, and keep the k nearest neighbors
7. If needed, also emit reverse k nearest neighbors

All steps can well be implemented on a cluster.
Except for sort and sliding window as "map" and "reduce".

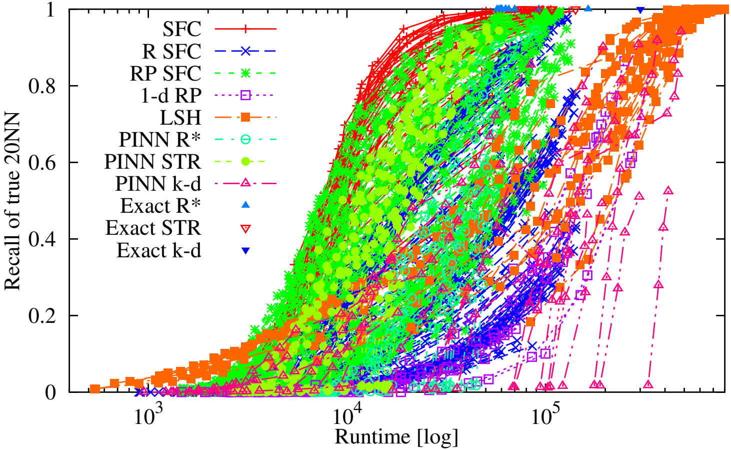
Ensemble for k-Nearest Neighbors

5. Merge the neighbor candidates for each point
6. Compute the real distances, and keep the k nearest neighbors
7. If needed, also emit reverse k nearest neighbors

```
distributed on every node do
    foreach (object, neighbor list) do
        // Reduce to true kNN
        Remove duplicates from neighbor list
        Compute distances
        emit (object, neighbors, ∅)          // Keep forward neighbors
        // If RkNN needed, also map to inverse list:
        foreach neighbor do
            emit (neighbor, ∅, [object])    // Build reverse neighbors
        end
    end
endon
shuffle to (object, kNN, RkNN)
```

Experiments

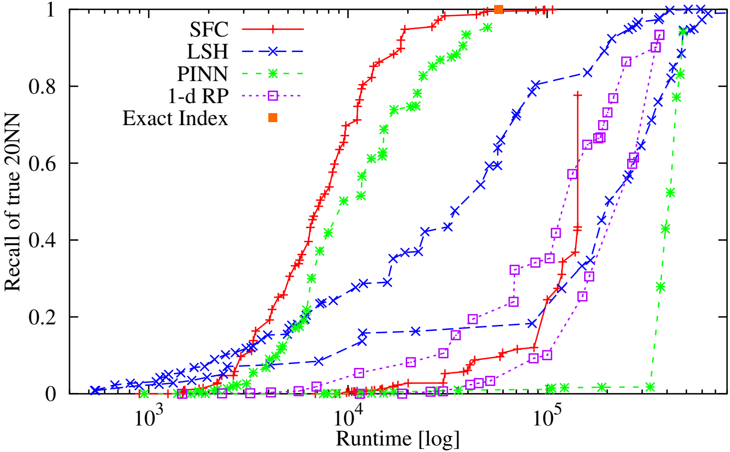
ALOI image database, 64 dimensions, recall of true kNN



Complete evaluation results.

Experiments

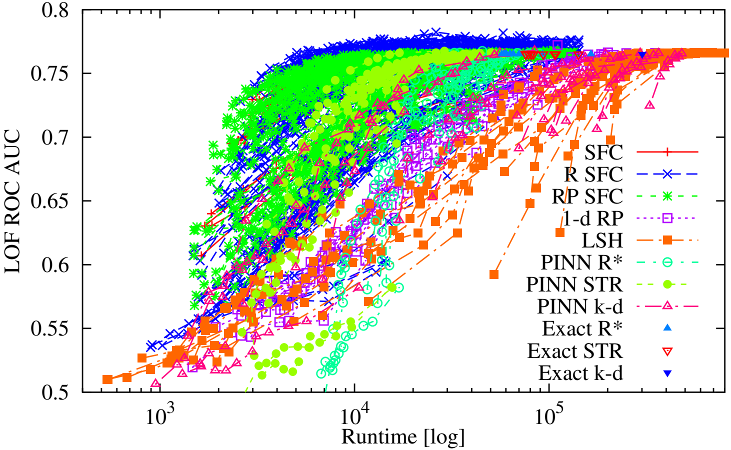
ALOI image database, 64 dimensions, recall of true kNN



Skyline results (results not dominated by other results)

Experiments

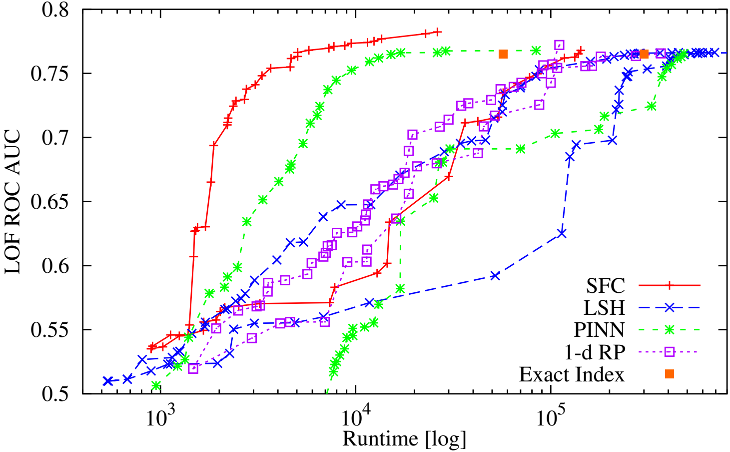
ALOI image database, 64 dimensions, LOF [Bre+00] quality



Complete evaluation results.

Experiments

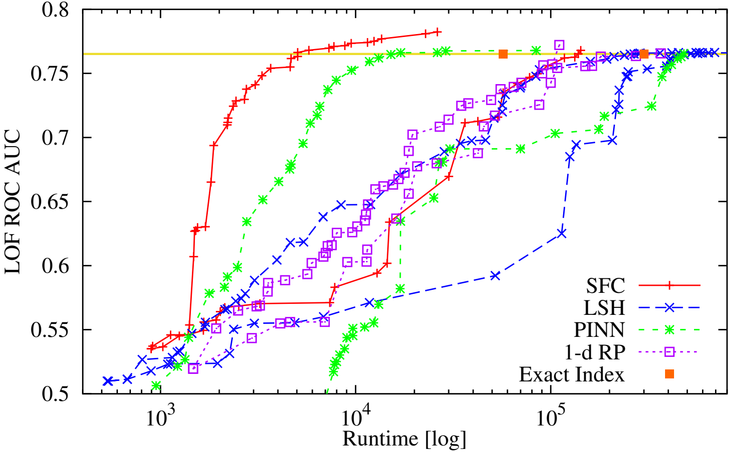
ALOI image database, 64 dimensions, LOF [Bre+00] quality



Skyline results (results not dominated by other results)

Experiments

ALOI image database, 64 dimensions, LOF [Bre+00] quality



Results via approximation can be *better* than exact results.

Better than exact?

This observation contradicts our intuition.

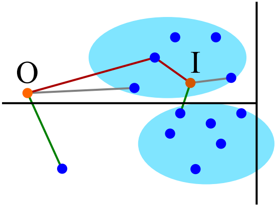
This is *not* an error.

- Random Forests [Bre01] ignore parts of the data and parts of the attributes – but work better than "exact" decision trees!
- Many other ensemble techniques, including:
  Feature bagging for outlier detection [LK05]
  Subsampling for outlier detection [Zim+13]
  Data perturbation for outlier detection [ZCS14]
- Our ensemble operates on a lower level (kNN),
  and improves *scalability* to big data.

Better than exact?

This observation contradicts our intuition.

Explanation:



- For inliers, missing a true kNN makes next to no difference. (It does not matter which highly similar points we choose.)
- For outliers, the true kNN may contain other outliers. If we miss them, and compare to cluster points instead, this makes the outlier more pronounced.

Interesting: errors do not have to be a problem.

A key observation:

Data often is not exact / complete.

Do we then need exact results?

Of course, we want exact results e.g. in accounting
– but on dirty data with *outliers*?

How to choose an indexing strategy:

The best method depends on your data.

- On low-dimensional data, R*-trees [Bec+90] are hard to beat.
- For sparse data, compressed inverted lists are excellent.
- PINN [dCH10] has nice theoretical guarantees, but quickly becomes expensive because of that.
- If you know the query radius ε, LSH [IM98] works well
- For k-nearest-neighbors on dense high-dimensional data, our new method [SZK15] works very well.

Note: space-filling-curves are desinged for Minkowski-norms.
LSH can support a few other distances, and the R*-tree too [SZK13].

Thank you!

Questions & Discussion

Outline

References I

[Ach01] D. Achlioptas. "Database-friendly Random Projections". In: *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Santa Barbara, CA*. 2001.

[AP02] F. Angiulli and C. Pizzuti. "Fast Outlier Detection in High Dimensional Spaces". In: *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD), Helsinki, Finland*. 2002, pp. 15–26. DOI: 10.1007/3-540-45681-3_2.

[Bec+90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles". In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Atlantic City, NJ*. 1990, pp. 322–331. DOI: 10.1145/93597.98741.

[Bre+00] M. M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander. "LOF: Identifying Density-based Local Outliers". In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX*. 2000, pp. 93–104. DOI: 10.1145/342009.335388.

[Bre01] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

References II

[Dat+04] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. “Locality-sensitive hashing scheme based on p-stable distributions”. In: *Proceedings of the 20th ACM Symposium on Computational Geometry (ACM SoCG), Brooklyn, NY*. 2004, pp. 253–262.

[dCH10] T. de Vries, S. Chawla, and M. E. Houle. “Finding Local Anomalies in Very High Dimensional Space”. In: *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM), Sydney, Australia*. 2010, pp. 128–137. DOI: 10.1109/ICDM.2010.151.

[dCH12] T. de Vries, S. Chawla, and M. E. Houle. “Density-preserving projections for large-scale local anomaly detection”. In: *Knowledge and Information Systems (KAIS)* 32.1 (2012), pp. 25–52. DOI: 10.1007/s10115-011-0430-4.

[Hil91] D. Hilbert. “Ueber die stetige Abbildung einer Linie auf ein Flächenstück”. In: *Mathematische Annalen* 38.3 (1891), pp. 459–460.

[IM98] P. Indyk and R. Motwani. “Approximate nearest neighbors: towards removing the curse of dimensionality”. In: *Proceedings of the 30th annual ACM symposium on Theory of computing (STOC), Dallas, TX*. 1998, pp. 604–613. DOI: 10.1145/276698.276876.

References III

[Jin+06] W. Jin, A. K. H. Tung, J. Han, and W. Wang. "Ranking Outliers Using Symmetric Neighborhood Relationship". In: *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Singapore*. 2006, pp. 577–593. DOI: 10.1007/11731139_68.

[Kri+09] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "LoOP: Local Outlier Probabilities". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM), Hong Kong, China*. 2009, pp. 1649–1652. DOI: 10.1145/1645953.1646195.

[LK05] A. Lazarevic and V. Kumar. "Feature Bagging for Outlier Detection". In: *Proceedings of the 11th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Chicago, IL*. 2005, pp. 157–166. DOI: 10.1145/1081870.1081891.

[Mor66] G. M. Morton. *A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing*. Tech. rep. International Business Machines Co., 1966.

[Pea90] G. Peano. "Sur une courbe, qui remplit toute une aire plane". In: *Mathematische Annalen* 36.1 (1890), pp. 157–160. DOI: 10.1007/BF01199438.

References IV

[RRS00] S. Ramaswamy, R. Rastogi, and K. Shim. "Efficient algorithms for mining outliers from large data sets". In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX*. 2000, pp. 427–438. DOI: 10.1145/342009.335437.

[SZK13] E. Schubert, A. Zimek, and H.-P. Kriegel. "Geodetic Distance Queries on R-Trees for Indexing Geographic Data". In: *Proceedings of the 13th International Symposium on Spatial and Temporal Databases (SSTD), Munich, Germany*. 2013, pp. 146–164. DOI: 10.1007/978-3-642-40235-7_9.

[SZK14a] E. Schubert, A. Zimek, and H.-P. Kriegel. "Generalized Outlier Detection with Flexible Kernel Density Estimates". In: *Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA*. 2014, pp. 542–550. DOI: 10.1137/1.9781611973440.63.

[SZK14b] E. Schubert, A. Zimek, and H.-P. Kriegel. "Local Outlier Detection Reconsidered: a Generalized View on Locality with Applications to Spatial, Video, and Network Outlier Detection". In: *Data Mining and Knowledge Discovery* 28.1 (2014), pp. 190–237. DOI: 10.1007/s10618-012-0300-z.

[SZK15] E. Schubert, A. Zimek, and H.-P. Kriegel. "Fast and Scalable Outlier Detection with Approximate Nearest Neighbor Ensembles". In: *Proceedings of the 20th International Conference on Database Systems for Advanced Applications (DASFAA), Hanoi, Vietnam*. 2015.

References V

[ZCS14] A. Zimek, R. J. G. B. Campello, and J. Sander. "Data Perturbation for Outlier Detection Ensembles". In: *Proceedings of the 26th International Conference on Scientific and Statistical Database Management (SSDBM), Aalborg, Denmark*. 2014, 13:1–12. DOI: 10.1145/2618243.2618257.

[Zim+13] A. Zimek, M. Gaudet, R. J. G. B. Campello, and J. Sander. "Subsampling for Efficient and Effective Unsupervised Outlier Detection Ensembles". In: *Proceedings of the 19th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Chicago, IL*. 2013, pp. 428–436. DOI: 10.1145/2487575.2487676.