My Journey to Data Mining

Hans-Peter Kriegel
Ludwig-Maximilians-Universität München
München, Germany
In the beginning…

• spatial databases – spatial data mining

height profile:
Maunga Whau Volcano (Mt. Eden), Auckland, New Zealand
Density-based Clustering: Intuition

- probability density function of the data
- threshold at high probability density level
- cluster of low probability density disappears to noise
Density-based Clustering: Intuition

- low probability density level
- 2 clusters are merged to 1
Density-based Clustering: Intuition

- medium (good) probability density level
- 3 clusters are well separated
DBSCAN: Density-Based Spatial Clustering of Applications with Noise

[Ester, Kriegel, Sander, Xu KDD 1996]

- Core points have at least $\text{minPts}$ points in their $\varepsilon$-neighborhood
- Density connectivity is defined based on core points
- Clusters are transitive hulls of density-connected points

$\text{minPts} = 5$
DBSCAN

• DBSCAN received the 2014 SIGKDD Test of Time Award

• DBSCAN Revisited: Mis-claim, Un-Fixability, and Approximation [Gan & Tao SIGMOD 2015]
  – Mis-claim according to Gan & Tao:
    
    \[ \text{DBSCAN terminates in } O(n \log n) \text{ time.} \]
    
    \[ \text{DBSCAN actually runs in } O(n^2) \text{ worst-case time.} \]
  
  – Our KDD 1996 paper claims:
    
    \[ \text{DBSCAN has an “average” run time complexity of } O(n \log n) \text{ for } \]
    
    \[ \text{range queries with a “small” radius (compared to the data space size) when using an appropriate index structure (e.g. } R^*\text{-tree)} \]
  
  – The criticism should have been directed at the “average” performance of spatial index structures such as \( R^* \)-trees and not at an algorithm that uses such index structures
• Contributions of the SIGMOD 2015 paper (apply only to Euclidean distance)

1. Reduction from the USEC (Unit-Spherical Emptiness Checking) problem to the Euclidean DBSCAN problem
   → lower bound of $\Omega \left( n^{4/3} \right)$ for the time complexity of every algorithm solving the Euclidean DBSCAN problem in $d \geq 3$

2. Proposal of an approximate grid-based DBSCAN algorithm for Euclidean distance running in $O(n)$ expected time
• **DBSCAN Revisited, Revisited:** Why and how you should (still) use DBSCAN [E. Schubert, Sander, Ester, Kriegel, Xu, to appear in ACM TODS, 2017]

- Experiments in the SIGMOD 2015 paper not of practical value
- Parameter $\varepsilon$ for the range queries was chosen much too large $\Rightarrow$ the approximate algorithm puts all objects into 1 cluster
- Extensive experiments show that for adequate choice of $\varepsilon$, the original DBSCAN algorithm with an R*-tree index outperforms the SIGMOD’15 approximate algorithm
• **DBSCAN Revisited, Revisited: Why and how you should (still) use DBSCAN** [E. Schubert, Sander, Ester, Kriegel, Xu, to appear in ACM TODS, 2017]

  – Lessons learnt from SIGMOD 2015 and ACM TODS 2017:
    • Lower bound of $\Omega \left( n^{4/3} \right)$ for the time complexity of any algorithm solving the Euclidean DBSCAN problem (SIGMOD 2015)
    • Original DBSCAN algorithm is still the method of choice (ACM TODS 2017)
Variants of Density-based Clustering

- **OPTICS**: Ordering Points To Identify the Clustering Structure
  [Ankerst, Breunig, Kriegel, Sander SIGMOD 1999]

- ordering of the database representing its density-based clustering structure

- suitable for data of different local densities and for hierarchical clusters
clusters point objects as well as spatially extended objects according to spatial and non-spatial attributes and more…

• GDBSCAN: Generalized DBSCAN
  [Sander, Ester, Kriegel, Xu DMKD Journal 1998]
recent survey on density-based clustering:
Subspace Clustering in High-dimensional data spaces

- **SUBCLU: Density-Connected SUBspace CLUstering for High-Dimensional Data** [Kailing, Kriegel, Kröger SDM 2004]

discovers dense clusters in axis-parallel subspaces of the high-dimensional data space

\[ p \text{ and } q \text{ density-connected in } \{A,B\}, \{A\} \text{ and } \{B\} \]

\[ p \text{ and } q \text{ not density-connected in } \{B\} \text{ and } \{A,B\} \]
Outlier Detection

- **LOF (Local Outlier Factor)**: Density-based, local outlier detection [Breunig, Kriegel, Ng, Sander SIGMOD 2000]
- quantifies how outlying an object is in its local neighborhood
High-dimensional Outlier Detection

• **ABOD**: Angle-Based Outlier Degree
  [Kriegel, M. Schubert, Zimek SIGKDD 2008]

- variance of the angles of the potential "outlier" to pairs of points
- angles are more stable than distances in high-dimensional spaces
Subspace Outlier Detection

- **SOD: Subspace Outlier Degree**
  [Kriegel, Kröger, E. Schubert, Zimek PAKDD 2009]
- detects outliers in subspaces of the high-dimensional data space
Approximations for Outlier Detection

- Fast and Scalable Outlier Detection with Approximate Nearest Neighbor Ensembles [E. Schubert, Zimek, Kriegel DASFAA 2015]
  - avoids pairwise comparison of objects to compute nearest neighbors
  - computes nearest neighbors in near-linear time using an ensemble of space-filling curves
Trend Detection

• SigniTrend: Scalable Detection of Emerging Topics in Textual Streams by Hashed Significance Thresholds
  [E. Schubert, Weiler, Kriegel SIGKDD 2014]
  – introduces a new significance measure using outlier detection
  – tracks all keyword pairs using hash tables in a heavy-hitter type algorithm
  – aggregates the detected co-trends into larger topics using clustering
Runtime Evaluation

• The (Black) Art of Runtime Evaluation: Are we comparing (data mining) algorithms or implementations?
  [Kriegel, E. Schubert, Zimek KAIS Journal, 1-38, 2016]
  – extensive study of runtime behavior of several algorithms (single-link, DBSCAN, k-means, LOF)
  – implementation details often dominate algorithmic merits
  – the same algorithm can exhibit runtime differences of two orders of magnitude and more in different implementations
Runtime Evaluation

- For more realistic comparisons, all algorithms should be implemented
  - in the same framework, in the same version
  - at the same level of generality, modularization, and optimization
  - using the same backing features (DB layer, index structures)
  and all algorithms should be suitably parameterized.

- We should
  - compare the behavior of algorithms in scalability experiments, not in single absolute runtime values,
  - demonstrate at which point (data set size, dimensionality, parameter values) the asymptotic behavior kicks in.
• **Subspace clustering, clustering high-dimensional data** [Kriegel, Kröger, Zimek]
  - Tutorials at ICDM, KDD, VLDB, PAKDD
  - Survey ACM TKDD 2009

• **Outlier detection**
  - Tutorials at PAKDD, KDD, SDM [Kriegel, Kröger, Zimek]

• **Outlier detection in high-dimensional data** [Zimek, E. Schubert, Kriegel]:
  - Tutorials at ICDM, PAKDD
  - Survey Statistical Analysis and Data Mining 2012
Implementations

- all these algorithms (and many more) are available in the ELKI framework: http://elki.dbs.ifi.lmu.de/
- ELKI is a java framework, integrating fast data management (e.g., indexing) and many data mining algorithms in a flexible way

release 0.6:
Thank You!