Fast approximate hubness reduction for large high-dimensional data

Roman Feldbauer*¶, Maximilian Leodolter†, Claudia Plant‡§, Arthur Flexer*
* Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria
† Austrian Institute of Technology, Vienna, Austria
‡ Faculty of Computer Science, University of Vienna, Vienna, Austria
§ ds:UniVie, University of Vienna, Vienna, Austria
¶ Department of Microbiology and Ecosystem Science, University of Vienna, Vienna, Austria
roman.feldbauer@ofai.at, maximilian.leodolter@ait.ac.at, claudia.plant@univie.ac.at, arthur.flexer@ofai.at

***Abstract*—High-dimensional data mining is challenging due to the "curse of dimensionality". Hubness reduction counters one particular aspect of the dimensionality curse, but suffers from quadratic algorithmic complexity. We present approximate hubness reduction methods with linear complexity in time and space, thus enabling hubness reduction for large data for the first time. Furthermore, we introduce a new hubness measure especially suited for large data, which is, in addition, readily interpretable. Experiments on synthetic and real-world data show that the approximations come at virtually no cost in accuracy in comparison with full hubness reduction. Finally, we demonstrate improved transport mode detection in massive mobility data collected with mobile devices as concrete research application. All methods are made publicly available in a free open source software package.**

***Index Terms*—curse of dimensionality, high-dimensional data mining, hubness, linear complexity, interpretability, smartphones, transport mode detection**

I. Introduction

Data mining in high-dimensional spaces is often challenging due to a number of phenomena commonly referred to as "curse of dimensionality" [1]. Concentration of distances is one of the more well-known aspects [2]: Distances between all pairs of data objects become increasingly similar with increasing dimensionality, thereby losing their discriminative power. Many popular distance measures are known to concentrate, including general ℓ^p and fractional norms, cosine similarities, dynamic time warping-based scores, and others [2]–[6].

Hubness is a related, albeit less known phenomenon [7]. In high but finite dimensional regimes, some objects have on average smaller distances to all others. These central objects have a high probability of becoming *hubs*, that is, objects that are unwontedly often among the nearest neighbors of many other objects in the data set. Simultaneously, *antihub* objects are extruded from neighborhood lists. As a consequence, neighborhood relations in high-dimensional spaces are often asymmetric: Any hub object $\boldsymbol{x}$ is nearest neighbor to many other objects, which, however cannot all be the nearest neighbor of $\boldsymbol{x}$. Hubs are known to propagate their information, for example class labels, (too) widely in distance space, while information carried by antihubs is essentially lost. As a consequence, many learning algorithms based on distances may be impaired by hubness. The literature contains many examples of detrimental effects of hubness to classification, clustering, visualization, outlier detection, and other data mining tasks [7]–[11].

Several hubness reduction methods have been proposed to mitigate detrimental effects of hubness. Three methods have been identified as state-of-the-art in terms of reducing hubness and improving learning performance in a recent review [12]. Mutual proximity (MP, [13]) and local scaling (LS, [14]) reduce hubness by repairing asymmetric neighborhood relations. DisSimLocal (DSL, [15]) flattens density gradients, which have been proposed as an alternative source of hubness [16]. All three methods transform *primary distances* to *secondary distances*. MP and LS may be applied to any dissimilarities, while DSL is restricted to (squared) Euclidean distances.

High computational complexity is a major restriction of secondary distance-based algorithms. Assuming hubness reduction methods require square distance matrices as input, the complexity of computing and storing distances between all pairs of objects can trivially be identified as $\Omega(n^2)$ in terms of both time and space, where n is the number of objects in the data set. Both quadratic time and space complexity are prohibitive for large data sets, essentially restricting hubness reduction to small data sets until now. In order to support hubness reduction in linear time and space complexity, we propose (i) a new hubness measure which enables reliable estimation of hubness even from small samples, and (ii) approximation strategies for established hubness reduction methods, both of which are required for successful hubness analysis and reduction of massive data. Extensive experiments demonstrate that both approaches support fast and powerful hubness reduction in large and high-dimensional data.

Hubness measures and reduction methods are briefly reviewed in Section II. Our main contributions are presented in Section III. The proposed methodology is evaluated in Section IV. Successful approximate hubness reduction is demonstrated for a real-world application in Section V, showing improved transport mode detection based on massive mobile phone sensor data. We discuss our findings in Section VI and conclude in Section VII, where we also point to future work.

This research is supported by the Austrian Science Fund (FWF): P27082 and P27703.

II. Related work

A. Hubness measures

Several hubness measures have been proposed so far, most of which are based on the notion of k-occurrence. Let $D \subseteq \mathbb{R}^m$ be a non-empty data set containing n objects in m-dimensional space. The k-occurrence $O^k(x)$ of an object $\boldsymbol{x} \in D$ counts, how often $\boldsymbol{x}$ occurs among the k-nearest neighbors of all other objects $D \setminus x$ indexed in the database. In low dimensional spaces, k-occurrence distributions typically are consistent with binomial distributions [7]. For example, assuming uniform i.i.d. data, this is in line with the intuition that all objects should be among the nearest neighbors of other objects approximately equally often. With increasing dimensionality, k-occurrence distributions are, however, increasingly skewed to the right [7]. This skewness is often used as a measure for hubness:

$$S^k = \mathbb{E}[(O^k - \mu_{O^k})^3] \;/\; \sigma^3_{O^k}, \tag{1}$$

where μ and σ indicate mean and standard deviation, respectively. The mean of a k-occurrence distribution is exactly k.

Computing the k-occurrence requires $\Omega(n)$ time for calculating all distances and partitioning. The measure defined above incorporates the k-occurrences of all objects, and thus requires $\Omega(n^2)$ time. The complexity can be trivially reduced by sampling, and calculating estimates of the above. The sample size may be varied *ad libitum* to approximate the true value. However, this is problematic in high dimensions, which we demonstrate in Section III-A, where we also present a new hubness measure overcoming these issues.

B. Hubness reduction methods

A recent survey recommends three methods for hubness reduction [12]. Let us briefly recall the formulations of full hubness reduction methods (quadratic complexity), before introducing approximate variants (linear complexity).

Local scaling (LS, [14]) transforms primary distances $d_{x,y}$ between objects $\boldsymbol{x}$ and $\boldsymbol{y}$ with local neighborhood information:

$$\mathrm{LS}^k(d_{x,y}) = 1 - \exp\left(-d^2_{x,y} \;/\; \left(r^k_x r^k_y\right)\right). \tag{2}$$

The scaling factors r^k_x and r^k_y refer to the distance of objects $\boldsymbol{x}$ and $\boldsymbol{y}$ to their k-th nearest neighbors, respectively.

Mutual proximity (MP, [13]) considers distances between all pairs of objects in a probabilistic way. Primary distances $d_{x,i\in\{1,\ldots,n\}\setminus x}$ between an object $\boldsymbol{x}$ and all other objects are modeled with some distribution. Let X be the resulting random variable (analogously for random variable Y and object y), and P the joint probability density function, then

$$\mathrm{MP}(d_{x,y}) = P(X > d_{x,y} \cap Y > d_{y,x}), \tag{3}$$

that is, the joint probability of $\boldsymbol{x}$ and $\boldsymbol{y}$ being nearest neighbors to each other. The complement $1 - \mathrm{MP}$ yields secondary distances. Here, we model primary distances with independent normal distributions $X \sim \mathcal{N}(\hat{\mu}_x, \hat{\sigma}^2_x)$, and $Y \sim \mathcal{N}(\hat{\mu}_y, \hat{\sigma}^2_y)$, so that secondary distances are calculated as

$$\mathrm{MPG}(d_{x,y}) = SF(d_{x,y}, \hat{\mu}_x, \hat{\sigma}^2_x) \cdot SF(d_{y,x}, \hat{\mu}_y, \hat{\sigma}^2_y), \tag{4}$$

with the survival function $SF(d, \mu, \sigma^2) = 1 - CDF(d, \mu, \sigma^2)$.

Two dissimilarity measures were introduced in [15] to reduce hubness in (squared) Euclidean spaces. The local variant DSL rescales distances using local centroids $c^k(\cdot)$:

$$\mathrm{DSL}^k(x,y) = \|x - y\|^2_2 - \|x - c^k(x)\|^2_2 - \|y - c^k(y)\|^2_2 \tag{5}$$

The local centroids are estimated as $c^k(x) = \frac{1}{k}\sum_{x' \in \mathrm{kNN}(x)} x'$, where $\mathrm{kNN}(x)$ is the set of k-nearest neighbors of $\boldsymbol{x}$.

C. Complexity of hubness reduction

Two factors dominate the complexity of hubness reduction. First, calculation of the primary distance between a pair of objects is typically in $\mathcal{O}(m)$, that is, linear in dimensionality. This holds for Euclidean and general ℓ^p distances. For dynamic time warping-based distances, near-optimal alignments can also be obtained in $\mathcal{O}(m)$, while exact alignments require $\mathcal{O}(m^2)$ [17].

Without loss of generality, assume the database is split into index $I \subset D$ and queries $Q \subset D$, with $n_{\text{indexed}} := |I|$ and $n_{\text{query}} := |Q|$. The set sizes are further determined by a split ratio $p \in]0,1[$, so that $n_{\text{indexed}} = p \cdot n$, $n_{\text{query}} = (1-p)n$, and $I \cup Q = D$, similarly e.g. to an evaluation scenario, where data is split into training and test sets. Calculating distances between each query object and all indexed objects thus requires $\Omega(n_{\text{query}} n_{\text{indexed}} m) = \Omega(n^2 m)$ time.

The second dominating factor is exact neighbor search. Hubness reduction methods often require knowledge of local neighborhoods. For example, consider scaling factors r^k in (2), or local centroids c^k in (5). Probing an object's neighborhood requires partitioning its distance list at the k-th nearest neighbor. $\Theta(n_{\text{indexed}}) = \Theta(n)$ time is required to find k nearest neighbors, for example using the introselect algorithm. Neighbor search thus amounts to $\Theta(n^2)$ for the whole data set. [1]

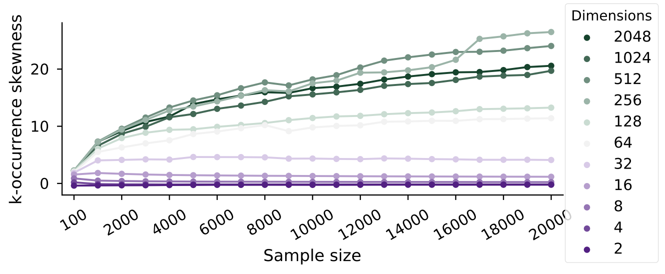
The quadratic complexity of hubness reduction methods is prohibitive for large data sets. To this end, we present approximate hubness reduction with linear complexity with respect to the data set size n in Sections III-B and III-C. [2]

III. Approximate hubness reduction methods

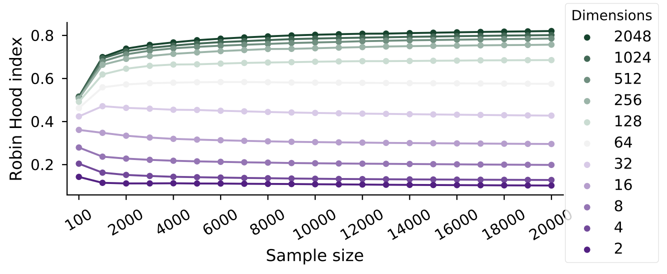
A prototypical hubness analysis pipeline was described in [12], and must answer two questions: (i) To what extent is the data set at hand affected by hubness? (ii) Can hubness reduction improve learning performance? The pipeline thus comprises two essential steps: hubness estimation, and hubness reduction. To perform hubness analysis in linear time and space, neither step must exceed linear complexity. To this end we present approximation strategies for hubness estimation and hubness reduction in linear time and space. In combination with a new hubness measure, this enables hubness analysis in large data sets.

[1]It is unnecessary to sort complete distance lists, which would require $\Theta(n \log n)$ time for each object.

[2]We assume dimension m to be fixed for each data set. Lower dimensional embeddings could be used additionally to the methodology suggested below to reduce complexity also w.r.t. dimension m. They are, however, out of scope for this work. Therefore, constant $\mathcal{O}(m)$ will be omitted henceforth.



(a) Commonly used hubness measure: skewness



(b) Proposed hubness measure: Robin Hood index

Fig. 1. Hubness measure dependency on sample size (x-axis), and dimensionality (lines). Each line in the plots constitutes one experiment of a MADELON-like data set (cf. Section IV-C) with 20 000 vectors in m-dimensional space, with $m \in \{2^1, 2^2, \ldots, 2^{11}\}$. Hubness is measured in random subsamples with sizes between 100 and 19 000 (bootstrapped ten times, averaged). (a) Skewness of k-occurrence histogram, (b) Robin Hood index of k-occurrence histogram.

A. Approximate hubness estimation

The sampling strategy for estimating hubness in linear time as described in Section II-A is problematic in high dimensions. We demonstrate issues of this strategy in synthetic data sets. For a motivating example, we consider clustered Gaussian data (MADELON-like, as described in Section IV-C). Fig. 1a shows the effect of sampling on k-occurrence skewness (k-occurrence parameter $k = 10$ is used throughout the paper). Values increase with increasing dimensionality as expected. In low dimensions (up to 16 or 32), skewness in small samples approximates skewness in large samples well. In high dimensions (64 and higher), however, convergence to the true value is slow. This is undesirable for fast hubness estimation, which should ideally be accurate already at small sample sizes.

To counter this, we propose a new hubness measure borrowed from socio-economics. The Robin Hood index (also known as Ricci-Schutz/Piètra/Hoover index) is traditionally used to measure income inequality [18]. We use it for measuring k-occurrence inequality:

$$\mathcal{H}^k = \frac{1}{2} \frac{\sum_{x \in D} |O^k(x) - \mu_{O^k}|}{\left(\sum_{x \in D} O^k(x)\right) - k} = \frac{\sum_{x \in D} |O^k(x) - k|}{2k(n-1)} \quad (6)$$

Fig. 1b shows the behavior of the new hubness measure on the same synthetic data used before. Again, we observe a trend of increasing values with increasing dimensionality, as expected and as desired. There is, however, only little dependence on sample sizes. That is, values converge quickly to the 'true' values (as observed in the full data set). Similar behavior of both hubness measures was observed in i.i.d. uniform and normal data as well (not shown).

In addition, the Robin Hood index is more easily accessible for interpretation than the skewness measure. Values fall in the fixed interval $[0, 1]$, where $\mathcal{H}^k = 0$ indicates that all objects are equally often among the nearest neighbors of others ($\forall x \in D : O^k(x) = k$), whereas $\mathcal{H}^k = 1$ means that a single hub is nearest neighbor to all objects ($\exists x \in D : O^k(x) = n - 1$). In other words, the Robin Hood index answers the question: What share of 'nearest neighbor slots' must be redistributed to achieve k-occurrence equality among all objects?

B. Instance selection for approximate hubness reduction

Reducing the number of objects indexed in the database is a straight-forward approach to achieve linear time and space complexity. Selecting a fixed number $s \ll n_{\text{indexed}}$ *representative* objects yields $\mathcal{O}(n_{\text{query}} s)$ query time and memory usage. Varying s trades off time and space requirements versus accuracy. We consider the following instance selection algorithms:

- *Random sampling* is fast and easy to use. It may be used in unsupervised fashion, or combined with class labels for stratification. Random sampling is rather trivial, and will serve as baseline here.
- *k-means++* is k-means clustering with 'careful seeding' [19]. Seed cluster centers are selected by an initialization strategy so that clustering converges in fewer iterations. We use this strategy for unsupervised instance selection, expecting to obtain better representatives than by random sampling.
- Other popular instance selection methods, such as *DROP3*, *ICF*, and related methods, use supervised algorithms [20]. We restrict this work to unsupervised techniques, however.

Algorithm 1 describes approximate hubness reduction with generic instance selection and the example of local scaling. Assume selectInstances() yields instances according to any one of the above mentioned methods. Within this algorithmic framework hubness reduction methods are easily exchangeable. For details, see Section III-C.

Instance selection essentially drops many objects from the database. While this reduces storage footprint to $\mathcal{O}(sm)$, information loss occurs, which may deteriorate learning performance. To overcome this issue, we present a second approximation strategy for hubness reduction with linear complexity.

C. Approximate nearest neighbors for hubness reduction

The complexity of hubness reduction may also be reduced by relaxing the requirement of exact neighbor search (cf. Section II-C). Approximate nearest neighbor (ANN) search retrieves true nearest neighbors with high recall, but allows for some errors. This relaxation makes sublinear query time per

Algorithm 1 Approximate local scaling with instance selection

```
Input:
    $k$                    ▷ nearest neighbor parameter
    $s$                    ▷ Number of selected instances
    $I, Q$                 ▷ Indexed objects, query objects, respectively
    $r_I, r_Q$             ▷ Empty vectors of sizes $n_{\text{indexed}}$, $n_{\text{query}}$, resp.
Output:
    $distLS$               ▷ Sparse matrix of shape $n_{\text{query}} \times n_{\text{indexed}}$, $\mathcal{O}(n_{\text{query}} s)$
 1: procedure FIT
 2:     $selectedSample \leftarrow$ SELECTINSTANCES$(I, s)$
 3:     for all $y \in I$ do                              ▷ $\mathcal{O}(n_{\text{indexed}} m)$
 4:         $r_I(y) \leftarrow$ K-TH-NN$(y, selectedSample)$
 5: procedure TRANSFORM
 6:     for all $x \in Q$ do                              ▷ $\mathcal{O}(n_{\text{query}} sm)$
 7:         $r_Q(x) \leftarrow$ K-TH-NN$(x, selectedSample)$
 8:         for all $y \in candidates$ do                 ▷ $\mathcal{O}(sm)$
 9:             $dist \leftarrow$ PRIMARY DISTANCE$(x, y)$
10:             $distLS(x, y) \leftarrow$ LS$(dist)$
        return $distLS$
```

Algorithm 2 Approximate local scaling with ANN

```
Input:
    $k$                    ▷ nearest neighbor parameter
    $s$                    ▷ Candidate neighborhood size
    $I, Q$                 ▷ Indexed objects, query objects, respectively
    $r_I, r_Q$             ▷ Empty vectors of sizes $n_{\text{indexed}}$, $n_{\text{query}}$, resp.
Output:
    $distLS$               ▷ Sparse matrix of shape $n_{\text{query}} \times n_{\text{indexed}}$, $\mathcal{O}(n_{\text{query}} s)$
 1: procedure FIT
 2:     $annIndex \leftarrow$ FITANNINDEX$(I)$
 3:     for all $y \in I$ do                              ▷ $\mathcal{O}(n_{\text{indexed}} m)$
 4:         $r_I(y) \leftarrow$ K-TH-ANN$(y, annIndex)$
 5: procedure TRANSFORM
 6:     for all $x \in Q$ do                              ▷ $\mathcal{O}(n_{\text{query}} sm)$
 7:         $candidates \leftarrow$ RANGEANN$(s, x, annIndex)$
 8:         $r_Q(x) \leftarrow$ K-TH-ANN$(x, annIndex)$
 9:         for all $y \in candidates$ do                 ▷ $\mathcal{O}(sm)$
10:             $dist \leftarrow$ PRIMARY DISTANCE$(x, y)$
11:             $distLS(x, y) \leftarrow$ LS$(dist)$
        return $distLS$
```

object possible. A plethora of ANN methods has been developed, most of which are based on tree algorithms [21], locality-sensitive hashing [22], product quantization [23], or proximity graphs [24]. ANN techniques have previously been suggested for hubness estimation [25], and hubness reduction [12]. To the best of our knowledge, this has not yet been evaluated or implemented in related software packages. We strive to fill this gap with the research presented here.

Combining ANN with hubness reduction yields a filter-and-refine methodology. For each object some $s \ll n_{\text{indexed}}$ approximate nearest neighbors are filtered from the database in sublinear time with respect to n_{indexed}. In the refinement step, hubness reduction is performed on these candidate neighbors, effectively reordering the approximate neighborhood. Assuming hubness reduction is then in $\mathcal{O}(s)$ per object rather than in $\mathcal{O}(n_{\text{indexed}})$, linear time complexity is obtained for the complete set of query objects.

Algorithm 2 exemplarily describes approximate local scaling with ANN techniques. Comments on the right-hand side (▷) indicate time spent in each loop, including nested loops, and subroutines. The fit() procedure initially sets up the ANN index. Assume fitANNIndex() creates this index in linear time, and k-th-ANN(y) returns the distance between $\boldsymbol{y}$ and its k-th approximate nearest neighbor in sublinear time[3]. LS scaling factors for all indexed objects are stored in vector r_I.

Query objects are handled in the transform() procedure. In the filtering step, rangeANN() returns the s approximate nearest neighbors ('candidates') for each query object $\boldsymbol{x}$. Primary distances are calculated between $\boldsymbol{x}$ and all candidates. Subsequently, these distances are transformed to secondary distances by LS in the refinement step. The order of candidate neighbors may have changed due to hubness reduction. Secondary distances are stored in a sparse matrix with shape $n_{\text{query}} \times n_{\text{indexed}}$ and $n_{\text{query}} s$ nonzero elements, which determines the memory requirements of $\mathcal{O}(n_{\text{query}} s)$.

Approximate mutual proximity (Algorithm 3) bears close resemblance to the algorithm above. For each object, a normal distribution is estimated from its distances to t indexed objects ($t = 30$ was suggested in [13]). MPG incorporates these Gaussians as described in (4).

Algorithm 3 Approximate mutual proximity (Gaussian)

```
Input:
    $s$                    ▷ Candidate neighborhood size
    $t$                    ▷ Sample size for estimating $\mu$, $\sigma$
    $I, Q$                 ▷ Indexed objects, query objects, resp.
    $\mu_I, \mu_Q$         ▷ Empty vectors of sizes $n_{\text{indexed}}$, $n_{\text{query}}$, resp.
    $\sigma_I, \sigma_Q$   ▷ Empty vectors of sizes $n_{\text{indexed}}$, $n_{\text{query}}$, resp.
Output:
    $distMPG$              ▷ Sparse matrix of shape $n_{\text{query}} \times n_{\text{indexed}}$, $\mathcal{O}(n_{\text{query}} s)$
 1: procedure FIT
 2:     $annIndex \leftarrow$ FITANNINDEX$(I)$
 3:     $samples \leftarrow$ RANDOM SAMPLE$(t, I)$
 4:     for all $y \in I$ do                              ▷ $\mathcal{O}(n_{\text{indexed}} m)$
 5:         $distances \leftarrow$ PRIMARY DISTANCE$(y, samples)$
 6:         $\mu_I(y) \leftarrow$ MEAN$(distances)$
 7:         $\sigma_I(y) \leftarrow$ STANDARDDEVIATION$(distances)$
 8: procedure TRANSFORM
 9:     for all $x \in Q$ do                              ▷ $\mathcal{O}(n_{\text{query}} sm)$
10:         $distances \leftarrow$ PRIMARY DISTANCE$(x, samples)$
11:         $\mu_Q(x) \leftarrow$ MEAN$(distances)$
12:         $\sigma_Q(x) \leftarrow$ STANDARDDEVIATION$(distances)$
13:         $candidates \leftarrow$ RANGEANN$(s, x, annIndex)$
14:         for all $y \in candidates$ do                 ▷ $\mathcal{O}(sm)$
15:             $dist \leftarrow$ PRIMARY DISTANCE$(x, y)$
16:             $distMPG(x, y) \leftarrow$ MPG$(dist)$
        return $distMPG$
```

DisSimLocal estimates local centroids for each object. A local centroid is the mean vector of an object's k nearest neighbors (see (5) and the paragraph below). Approximate DSL (Algorithm 4) estimates the centroids from approximate nearest neighbors.

All described methods are implemented in the publicly available OFAI Hub-Toolbox[4] [26].

[3] This holds for LSH, HNSW, among other ANN techniques. Index creation scales loglinearly in case of HNSW. For details please refer to related publications and documentation.

[4]Source code available at https://github.com/OFAI/hub-toolbox-python3. Python package available at https://pypi.org/project/hub-toolbox/. The Hub-Toolbox is free open source software licensed under GPLv3.

Algorithm 4 Approximate DisSimLocal

```
Input:
    $k$                                   ▷ nearest neighbor parameter
    $s$                                   ▷ Candidate neighborhood size
    $I, Q$                                ▷ Indexed objects, query objects, resp.
    $c_I$                                 ▷ Empty matrix of size $n_{\text{indexed}} \times m$.
    $c_Q$                                 ▷ Empty matrix of size $n_{\text{query}} \times m$.
Output:
    $distDSL$     ▷ Sparse matrix of shape $n_{\text{query}} \times n_{\text{indexed}}$, $\mathcal{O}(n_{\text{query}} s)$
 1: procedure FIT
 2:     $annIndex \leftarrow$ FITANNINDEX($I$)
 3:     for all $y \in I$ do                          ▷ $\mathcal{O}(n_{\text{indexed}} m)$
 4:         $neighbors \leftarrow$ RANGEANN($k, y, annIndex$)
 5:         $c_I(y) \leftarrow$ LOCALCENTROID($y, neighbors$)
 6: procedure TRANSFORM
 7:     for all $x \in Q$ do                          ▷ $\mathcal{O}(n_{\text{query}} sm)$
 8:         $neighbors \leftarrow$ RANGEANN($k, x, annIndex$)
 9:         $c_Q(x) \leftarrow$ LOCALCENTROID($x, neighbors$)
10:         $candidates \leftarrow$ RANGEANN($s, x, annIndex$)
11:         for all $y \in candidates$ do             ▷ $\mathcal{O}(sm)$
12:             $dist \leftarrow$ PRIMARY DISTANCE($x, y$)
13:             $distDSL(x, y) \leftarrow$ DSL($x, y, c_I, c_Q$)
        return $distDSL$
```

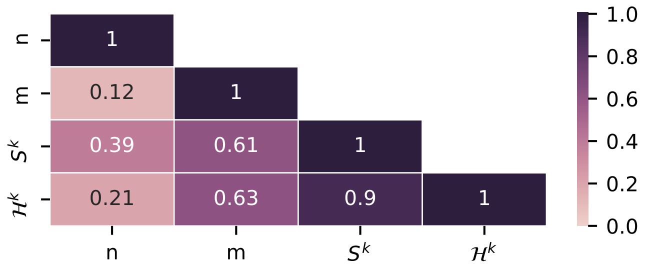


Fig. 2. Evaluation of Robin Hood ($\mathcal{H}^k$) measure. Values indicate Spearman correlation between attributes (n data set size, m dimension, S^k skewness).

IV. Evaluation

We evaluate the proposed hubness measure and approximate hubness reduction methods with multiple experiments on both synthetic and real-world data.

A. Robin Hood index

The findings on synthetic data presented in Fig. 1 suggest the Robin Hood index to be a suitable measure for hubness. We further evaluate the Robin Hood index on 50 real-world data sets, which have previously been used for analyzing aspects of hubness measures in [12]. Skewness $S^{k=10}$ and Robin Hood index $\mathcal{H}^{k=10}$ are calculated for each data set without any sampling. The proposed hubness measure shows very strong rank correlation (Spearman $r = 0.90$) with the well-established measure (Fig. 2). We observe equal correlations of both measures with data dimensionality, but weaker dependency of the Robin Hood index on data set size, compared to the skewness measure. Both results are in line with the results from Fig. 1. We thus deem the Robin Hood index a suitable measure for hubness. Since it allows to estimate hubness from small sample sizes, hubness can now be estimated accurately in linear time.

B. Approximate hubness reduction - preface

The methodology presented in Section III-C is agnostic to the choice of ANN method. We select two methods from different ANN families as showcases for approximate hubness reduction:

- *Locality-sensitive hashing* (LSH) is an established and commonly used ANN method [22]. LSH is based on hash functions, which are constructed so that close objects have high probability of collisions. Objects with colliding hashes are mapped to the same approximate neighborhood.
- *Navigable small-world graphs* (NSW) are a more recent, graph-based approach to ANN. Indexed objects form the vertices of a graph, whose edges are built so that the network contains an approximate Delaunay subgraph [24]. Insertions and queries are performed by greedy search. The HNSW variant features a controllable hierarchy [27], and was particularly successful in a recent ANN benchmark [28].

We use the FALCONN [29] and nmslib [30] libraries for LSH- and HNSW-based experiments, respectively.

Previous work showed hubness reduction methods to be robust with respect to the neighborhood parameter k. We use fixed LS/DSL parameter $k = 5$ for all experiments in this work. MP is parameter-free.

We select a restrictive sample size $s = 100$ for all experiments. This is relaxed to $s = 1000$ only for the two largest data sets (UCI character fonts, Table IIg; and mobility data, Table III).

C. Results with synthetic data

We evaluate approximate hubness reduction in synthetic data sets of sizes $n \in \{1000, 2000, \dots, 20\,000\}$, constructed similarly to MADELON [31]. For each data set, n objects in ten clusters are generated at the vertices of an 800-dimensional hypercube, and 200 uninformative noise features are added, yielding a 1000-dimensional embedding space. Five class labels are assigned to the data, with each class comprising two clusters. We perform five-fold stratified cross-validation, and consider three evaluation metrics: (i) runtime (Fig. 3), (ii) hubness reduction, measured with the Robin Hood index (Fig. 4a), and (iii) accuracy in ten-nearest neighbor classification (Fig. 4b), where random classification yields 20 % accuracy.

Full hubness reduction scales quadratically with the number of objects. This is visualized in the top row of Fig. 3 for model creation (fit procedure) in the left column, and (secondary) distance calculations (transform procedure) for query objects in the right column. The red line in Fig. 3ab represents baseline k-NN with Euclidean distances. Each of the following rows illustrates the time complexity of one approximation method. For example, Fig. 3c shows time requirements for instance selection with random sampling, which is, by far, the fastest method, because it does not consider properties of the indexed data. In contrast, k-means++ processes the indexed data to

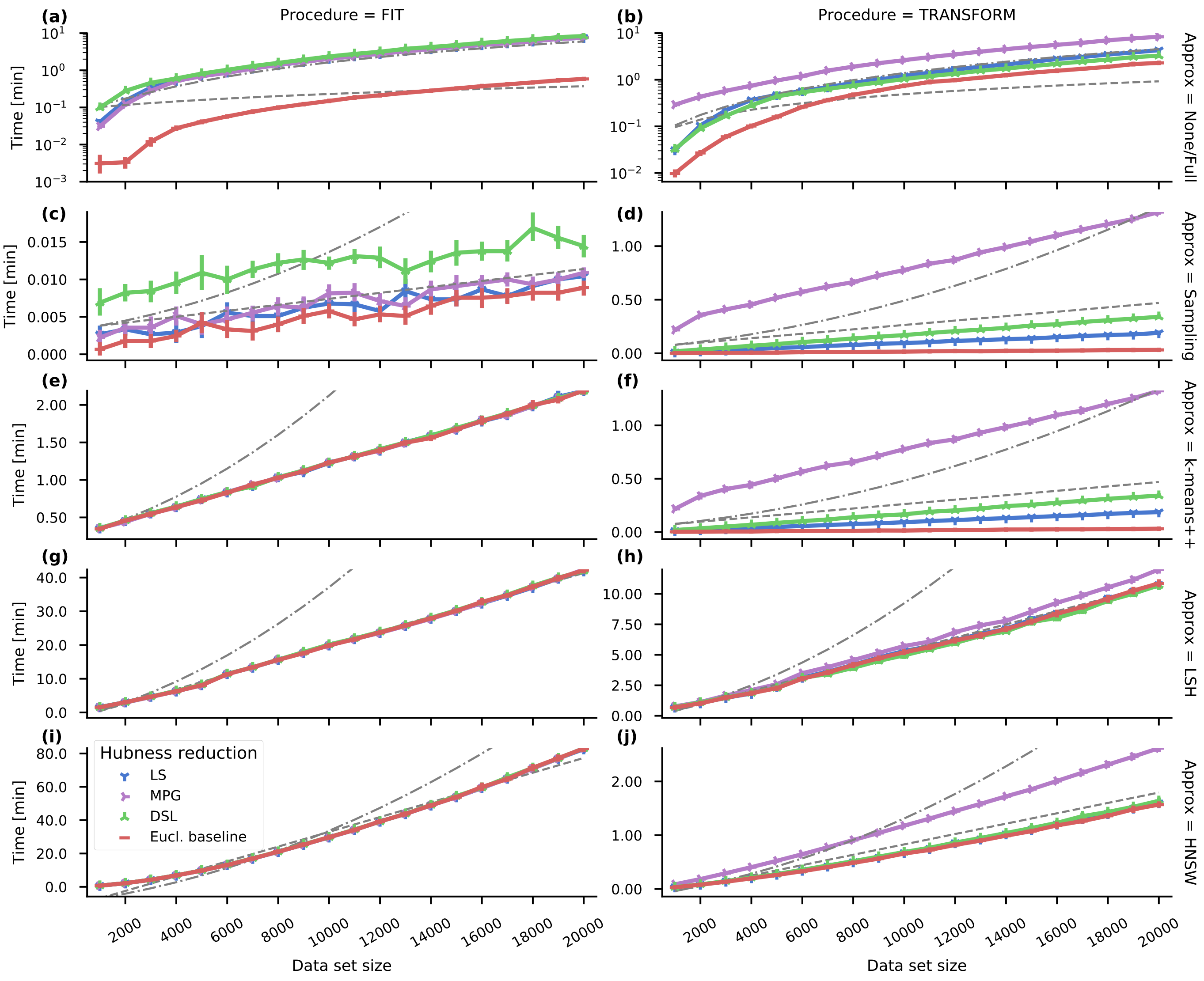


Fig. 3. Linear time complexity of approximate hubness reduction. Runtime for 80 % / 20 % train / test-split with synthetic data. Colors code for hubness reduction methods (blue: local scaling, violet: mutual proximity, green: DisSimLocal, red: Euclidean distances without hubness reduction). Each row corresponds to one approximation method, as indicated on the right-hand side. Left column: model creation time, right column: (secondary) distance compute time. For example, the blue line in subplot h shows time spent in procedure TRANSFORM of Algorithm 2. It is mostly overlayed by the red line, indicating that primary distance calculation dominates overall runtime. Values are averages of ten replicates. Error bars indicate standard deviation (generally low and clearly visible only in subplot c). Gray lines show linear (dashed) and quadratic (dash-dotted) functions for comparison. Note the log-scaled y-axes in a and b.

select suitable instances. It thus spends more time in the fit procedure, while the transform time is identical for both instance selection methods.

Runtime of approximate hubness reduction with ANN methods is dominated by index structure generation, as indicated by the Euclidean baseline in red, which overlays the other curves. Index generation is linear in case of LSH (Fig. 3g), and linearithmic in case of HNSW (Fig. 3i; consider the slightly increased growth compared to the fitted linear function displayed as dashed gray line).

Query time is only slightly longer with hubness reduction than without, especially in case of LS and DSL (Fig. 3hj). In other words, hubness reduction comes essentially ‘for free’ compared to solely performing ANN search.

Fig. 4 shows hubness and classification performance in synthetic data with 20 000 objects (corresponding to the rightmost values in Fig. 3). Fig. 4a shows that ANN-based methods reduce hubness in terms of the Robin Hood index to comparable levels as full hubness reduction. Instance selection, however, nearly maximizes the Robin Hood index. This is caused by measuring hubness with respect to the original database. Selecting hundred instances from twenty thousands objects essentially creates 19 900 antihubs and thus leads to extremely imbalanced k-occurrences.

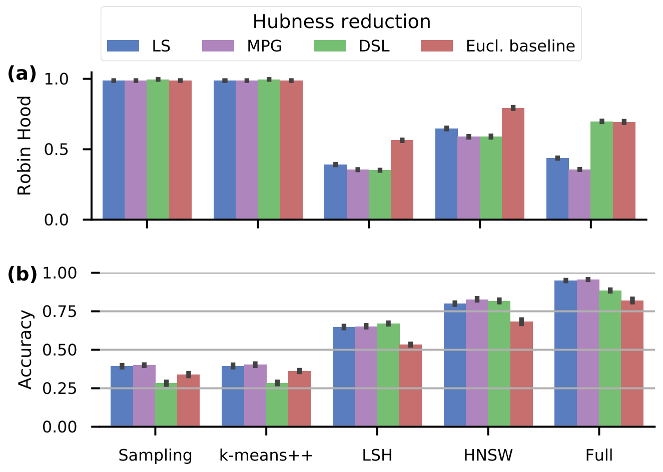


Fig. 4. Evaluation results of medium-sized synthetic data.

TABLE I
APPROXIMATE HUBNESS REDUCTION THROUGHPUT IN 1M DATA.

	Random sampling	k-means++	LSH	HNSW
LS	103 412	538	10	31
MPG	113 636	566	11	30
DSL	99 304	577	10	31
Eucl.	446 428	3240	25	59

(a) Throughput in procedure FIT (objects/s)

	Random sampling	k-means++	LSH	HNSW
LS	33 344	30 211	35	1857
MPG	8501	8730	39	1509
DSL	22 036	19 527	35	1419
Eucl.	255 754	264 550	86	2759

(b) Throughput in procedure TRANSFORM (objects/s)

Fig. 4b shows that hubness reduction methods improve classification in all cases of approximation or no approximation compared to the Euclidean baselines depicted as red bars (except for DSL with instance selection). HNSW with hubness reduction is on par with full nearest neighbor classification using Euclidean distances, despite the very small sample size of 100, i.e. only $0.625\,\%$ of the training set size.

The feasibility of hubness reduction in large data sets is demonstrated on a synthetic data set of one million objects, for which full hubness reduction would require nearly $4\,\mathrm{TiB}$ main memory. Table I shows the throughput of approximate hubness reduction algorithms at runtime on a Intel Xeon Silver 4114 CPU @ $2.20\,\mathrm{GHz}$ in a single-threaded environment. Hubness and classification performance are in line with the results in Fig. 4 (not shown). Hubness reduction calculates secondary distances from Euclidean distances. Therefore, it requires additional compute time compared to the Euclidean baseline. The extra time expenditure is, however, moderate for LSH and HNSW, in which cases hubness reduction requires roughly twice the baseline time. Instance selection methods are orders of magnitude faster, but suffer from severely impaired accuracy.

D. Results with public machine learning data sets

Having established successful approximate hubness reduction in synthetic data, we evaluate our methods on seven real-world data sets affected by hubness from public machine learning repositories also [32]–[34].

For all data sets, the Robin Hood index is improved by full hubness reduction methods, as expected (Table II, left-hand side). ANN without hubness reduction yields the same level of hubness as the complete Euclidean distance matrices, with LSH reducing hubness slightly in some cases. Approximate hubness reduction yields highly comparable Robin Hood values compared to full hubness reduction in all cases, except for HNSW, which does not perform well for UCI farm-ads.

For each data set, the best result in terms of classification accuracy is obtained with one of the hubness reduction methods (Table II, right-hand side). The results of approximate methods closely follow those of full methods, that is, both full and approximate hubness reduction improve accuracy over the Euclidean baseline. Indeed, LSH-based hubness reduction achieved slightly higher accuracy than full hubness reduction in five data sets. Note, that in case of UCI character fonts, full hubness reduction is not feasible with reasonable compute resources. We were, however, able to compute the full Euclidean baseline. Approximate hubness reduction improves over this baseline also in this large data set.

Both random sampling and k-means++ yield high Robin Hood values, comparably to the ones described in Section IV-C. Classification accuracy is on average $15\,\%$-points below ANN-based approximate hubness reduction. Since instance selection is clearly inferior to ANN for hubness reduction here, we do not report the results in Table II for brevity.

V. APPLICATION: TRANSPORT MODE DETECTION

We test our methods on accelerometer time series collected by ninety participants equipped with smartphones and a special smartphone application that records sensor data during traveling in an urban environment. The users labeled the recorded trips with the chosen transport mode. Table III reports the amount of collected data per mode and the corresponding labels for classification.

For this work, we preprocess the data by splitting each recorded trip into segments of 30 seconds and aggregate the time series to a dimension of 600. Further, to make the recorded signal independent of the orientation of the device we use the 2-norm of the three-dimensional accelerometer vector as in [35]. Dynamic time warping (DTW) alignments are computed between time-series vectors with the IncDTW package for R [36]. DTW-based similarities are used to classify trip segments as either active mobility, transport on the road, or on the rail. The fourth class indicates the user having reached his or her destination (Table III). The complete DTW similarity matrix is used for experiments with full hubness

TABLE II
EVALUATION RESULTS OF REAL-WORLD DATA SETS.

	Hubness (Robin Hood)			Classification accuracy		
	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.282	0.282	0.282	0.932	0.932	0.933
MPG	0.282	0.283	0.294	0.933	0.935	0.930
DSL	0.303	0.323	0.354	**0.938**	0.906	0.927
Eucl	0.329	0.329	0.329	0.904	0.906	0.902

(a) OpenML Semeion (1593 samples, 256 features, 10 classes)

	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.294	0.309	0.293	0.831	0.836	0.845
MPG	0.290	0.292	0.279	0.846	0.850	**0.852**
DSL	0.413	0.417	0.675	0.834	0.830	0.794
Eucl	0.388	0.433	0.414	0.773	0.779	0.780

(b) LibSVM DNA (2000 samples, 180 features, 3 classes)

	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.308	0.307	0.297	**0.746**	0.744	0.745
MPG	0.323	0.331	0.357	0.715	0.715	0.693
DSL	0.315	0.556	0.332	0.735	0.500	0.740
Eucl	0.419	0.419	0.419	0.669	0.670	0.669

(c) Corel1000 (1000 samples, 192 features, 10 classes)

	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.345	0.358	0.327	0.888	0.889	0.883
MPG	0.357	0.376	0.358	0.860	0.861	0.887
DSL	0.345	0.364	0.389	0.895	0.889	**0.896**
Eucl	0.485	0.511	0.508	0.875	0.856	0.864

(d) UCI CNAE-9 (1080 samples, 856 features, 9 classes)

	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.329	0.353	0.338	**0.783**	0.776	0.775
MPG	0.320	0.334	0.320	0.777	0.768	0.769
DSL	0.338	0.369	0.515	0.742	0.761	0.782
Eucl	0.470	0.526	0.517	0.707	0.695	0.699

(e) LibSVM splice (1000 samples, 60 features, 2 classes)

	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.430	0.681	0.383	0.801	0.683	0.832
MPG	0.505	0.670	0.462	0.814	0.747	0.848
DSL	0.447	0.671	0.595	**0.852**	0.769	0.838
Eucl	0.567	0.676	0.616	0.780	0.699	0.757

(f) UCI farm-ads (4143 samples, 54877 features, 2 classes)

	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.303	0.308	n/a	0.675	0.675	n/a
MPG	0.304	0.335	n/a	0.678	0.665	n/a
DSL	0.322	0.335	n/a	**0.687**	0.680	n/a
Eucl	0.355	0.358	0.359	0.656	0.655	0.633

(g) UCI character-fonts (832670 samples, 410 features, 153 classes)

reduction. In case of LSH and HNSW, the candidate neighbors are first retrieved using the methods' intrinsic distance measures. These distances are subsequently replaced by DTW similarities, which are used for the baseline in Fig. 5 ('LSH', 'HNSW'). Nearest neighbor classification performance is estimated in leave-one-user-out cross-validation. Classification accuracy macro-averaged over the folds is reported as an estimate, how well the system performs for a new user. No attempt was made to optimize hyperparameters, since the objective here is to demonstrate improvements due to hubness reduction, rather than maximizing performance of the complete classifier pipeline. Therefore, we use fixed $k = 10$ neighbors for classification to allow for direct comparisons between pipelines with and without hubness reduction. We postpone hyperparameter optimization to future work, which will focus on maximizing transport mode detection performance.

TABLE III
ACCELEROMETER DATA FOR TRANSPORT MODE DETECTION

Class	Mode	Time (h)	# 30 s trip segments
ACTIVE	bicycle	52	6273
	waiting	9	1108
	walking	245	29 443
ROAD	bus	49	5828
	car	108	12 937
	motorcycle	8	932
RAIL	metro	46	5481
	train	74	8848
	tram	33	3997
DESTINATION	destination	61	7263
		626	82 110

Hubness reduction is performed on DTW-based similarities with LS and MPG. We omit DSL due to its tailoring to Euclidean spaces. The database size of 82 110 allows us to compute a full DTW similarity matrix, and perform full hubness reduction for this evaluation scenario.[5] We observe marked improvements in terms of Robin Hood index and classification accuracy due to hubness reduction with LS (Fig. 5, 'Full'). In contrast, full MPG is detrimental to both hubness and accuracy.

Hubness appears to be heavily increased by the approximate methods. For instance selection, this is expected, as described in Section IV-C. Rather surprisingly, ANN methods show similar behavior given this data set. Analysis of k-occurrences revealed the imbalance again to be caused by a very high number of antihubs.

The approximations show reduced baseline classification performance compared to the full scenario. Applying hubness reduction on top, however, strongly improves accuracy. For both HNSW and random sampling, applying LS more than doubles accuracy. Strikingly, random sampling with LS yields the best classification performance among the approximate methods, and is on par with the full scenario. HNSW and k-means++ show competitive results also. The approximations profit from MPG as well, showing accuracy slightly below LS.

VI. DISCUSSION

Hubness analysis of massive data requires both hubness estimation and hubness reduction with linear complexity in time and space w.r.t. the data set size. The new Robin Hood index enables accurate hubness estimation from small samples. Linear algorithmic complexity of hubness reduction methods is achieved by introducing approximations based on instance

[5] 1750 h DTW compute time and 80 GiB peak memory during hubness reduction clearly show the infeasibility of this approach in productive systems.

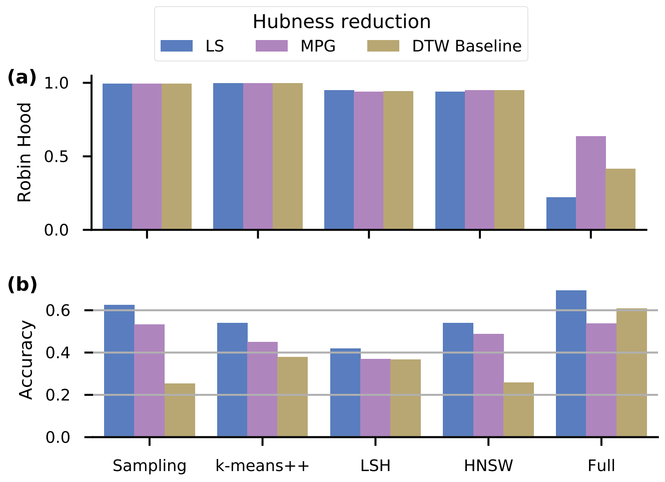


Fig. 5. Transport mode detection performance. Column ‘Full’ refers to using the complete DTW similarity matrix.

selection or ANN methods. We find that this comes at almost no expense in terms of classification accuracy in case of the ANN-based approach: Both LSH and HNSW showed overall comparably good performance. Instance selection performs worse than ANN in most cases. In contrast to our expectations, we observe no advantages of k-means++ instance selection over random sampling. The extra time required for k-means++ may be better used for larger random samples. Our experiments on synthetic data show only moderate additional time required for hubness reduction compared to approximate neighbor search alone. The ANN libraries used in this work were among the fastest in a recent benchmark [28]. We thus deem our methods to be fast enough for use in productive systems.

Experiments on data sets from machine learning repositories show, that successful approximate hubness reduction is not limited to synthetic data sets, which may exhibit special properties, such as Gaussian distributions etc. The approximate methods performed on par with the full methods in all evaluated data sets, hinting at their potential to completely replacing expensive full hubness reduction.

We demonstrate, how a real-world application is enhanced by approximate hubness reduction. Transport mode detection based on mobile phone sensor data is improved, despite supposedly unsuccessful reduction of hubness *per se*. Let us discuss the high Robin Hood indices of approximate hubness reduction in mobility data in detail: LSH and HNSW use metric distances for approximate neighbor search. These are generally not well suited for time-series data. Theoretical hubness research found, that hubs are usually close to the data centroid. For normed accelerometer data, the time-series centroid is a vector of approximately $9.81\,\mathrm{m/s^2}$ at each point in time, i.e. zero movement plus earth gravity. Indeed we find, that mobility hub objects in Euclidean space are mostly trip segments of very little user movement. This includes waiting at red traffic lights or traffic jams, which may occur in nearly any vehicle, or during walking. Segments with little movement may thus belong to essentially any of the presented transport modes. It is easy to imagine how this severely affects classification. Metric ANN seems to be influenced by this behavior as well, retrieving many low movement segments, but only few high movement segments as candidate neighbors. We hypothesize that hubness reduction is able to reorder the candidate neighborhood in a way that penalizes segments of very low movement. Unless the candidate neighborhood size is too restrictive, the high number of antihubs appears to be noncritical, and hubness reduction thus improves classification accuracy.

VII. Conclusion and Outlook

We presented approximate hubness reduction methods with linear complexity, which enable hubness reduction in large data sets. The approximations come at almost no cost in terms of accuracy compared to full hubness reduction as demonstrated for synthetic data, and even surpassed the full methods in several real-world data sets. We also showed how transport mode detection is improved by both full and approximate hubness reduction. In addition, the new Robin Hood index enables fast hubness estimation in massive data sets, and is more easily interpretable than a traditional measure. Having eliminated the bottleneck of high complexity, we hope hubness reduction will aid many researchers and practitioners with their specific tasks for large data sets. To this end, we make all our hubness reduction methods publicly available in a free open source software package.

In future work on general hubness, we will investigate additional approximate kNN-graph construction methods [37], [38] for hubness estimation, and additional ANN methods, such as tree-based methods or product quantization, for approximate hubness reduction. In terms of concrete applications, we will perform in-depth analysis of hubness in mobility data. We will test our hypotheses of how hubness reduction improves classification. In addition, we will examine, whether different ANN techniques can yield better candidate neighbors and avoid high numbers of antihubs. A focus will be on non-metric techniques, such as distance-based hashing [39] and others.

Acknowledgment

We thank Jan Schlüter for fruitful discussions, and Thomas Rattei for computational resources.

References

[1] R. E. Bellman, *Adaptive control processes: A guided tour*. Princeton university press, 1961.

[2] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is “nearest neighbor” meaningful?” In *Database Theory – ICDT'99*, C. Beeri and P. Buneman, Eds., Springer Berlin Heidelberg, 1999, pp. 217–235.

[3] A. Zimek, E. Schubert, and H.-P. Kriegel, “A survey on unsupervised outlier detection in high dimensional numerical data,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 5, no. 5, pp. 363–387, 2012. DOI: 10.1002/sam.11161.

[4] D. Francois, V. Wertz, and M. Verleysen, “The concentration of fractional distances,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 7, pp. 873–886, 2007.

[5] A. Nanopoulos, M. Radovanovic, and M. Ivanovic, “How does high dimensionality affect collaborative filtering?” In *Proceedings of the 2009 ACM Conference on Recommender Systems, New York, NY, USA*, L. D. Bergman, A. Tuzhilin, R. D. Burke, A. Felfernig, and L. Schmidt-Thieme, Eds., ACM, 2009, pp. 293–296.

[6] D. Yu, X. Yu, and A. Wu, “Making the nearest neighbor meaningful for time series classification,” in *4th International Congress on Image and Signal Processing*, vol. 5, Oct. 2011, pp. 2481–2485.

[7] M. Radovanović, A. Nanopoulos, and M. Ivanović, “Hubs in space: Popular nearest neighbors in high-dimensional data,” *Journal of Machine Learning Research*, vol. 11, pp. 2487–2531, 2010.

[8] D. Schnitzer and A. Flexer, “The unbalancing effect of hubs on k-medoids clustering in high-dimensional spaces,” in *International Joint Conference on Neural Networks (IJCNN)*, Jul. 2015, pp. 1–8.

[9] A. Flexer, “Improving visualization of high-dimensional music similarity spaces.,” in *Proceedings of the 16th International Society for Music Information Retrieval (ISMIR) Conference*, 2015, pp. 547–553.

[10] M. Radovanović, A. Nanopoulos, and M. Ivanović, “Reverse nearest neighbors in unsupervised distance-based outlier detection,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1369–1382, 2015.

[11] A. Flexer, “An empirical analysis of hubness in unsupervised distance-based outlier detection,” in *16th International Conference on Data Mining Workshops (ICDMW)*, IEEE, 2016, pp. 716–723.

[12] R. Feldbauer and A. Flexer, “A comprehensive empirical comparison of hubness reduction in high-dimensional spaces,” *Knowledge and Information Systems*, May 2018. [Online]. Available: https://doi.org/10.1007/s10115-018-1205-y.

[13] D. Schnitzer, A. Flexer, M. Schedl, and G. Widmer, “Local and global scaling reduce hubs in space,” *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2871–2902, 2012.

[14] L. Zelnik-Manor and P. Perona, “Self-tuning spectral clustering,” *Advances in Neural Information Processing Systems*, vol. 17, L. Saul, Y. Weiss, and L. Bottou, Eds., pp. 1601–1608, 2005.

[15] K. Hara, I. Suzuki, K. Kobayashi, K. Fukumizu, and M. Radovanović, “Flattening the density gradient for eliminating spatial centrality to reduce hubness,” in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2016, pp. 1659–1665.

[16] T. Low, C. Borgelt, S. Stober, and A. Nürnberger, “The hubness phenomenon: Fact or artifact?” In *Towards Advanced Data Analysis by Combining Soft Computing and Statistics*, C. Borgelt, M. Á. Gil, J. M. Sousa, and M. Verleysen, Eds., Springer Berlin Heidelberg, 2013, pp. 267–278.

[17] S. Salvador and P. Chan, “Toward accurate dynamic time warping in linear time and space,” *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, Oct. 2007.

[18] E. M. Hoover, “The measurement of industrial localization,” *The Review of Economics and Statistics*, vol. 18, no. 4, pp. 162–171, 1936. [Online]. Available: http://www.jstor.org/stable/1927875.

[19] D. Arthur and S. Vassilvitskii, “K-means++: The advantages of careful seeding,” in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[20] S. García, J. Luengo, and F. Herrera, “Tutorial on practical tips of the most influential data preprocessing algorithms in data mining,” *Knowledge-Based Systems*, vol. 98, pp. 1–29, 2016.

[21] M. Muja and D. G. Lowe, “Scalable nearest neighbor algorithms for high dimensional data,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.

[22] P. Indyk and R. Motwani, “Approximate nearest neighbors: Towards removing the curse of dimensionality,” in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, ACM, 1998, pp. 604–613.

[23] H. Jégou, M. Douze, and C. Schmid, “Product quantization for nearest neighbor search,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, 2011.

[24] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov, “Approximate nearest neighbor algorithm based on navigable small world graphs,” *Information Systems*, vol. 45, pp. 61–68, 2014.

[25] N. Tomašev, M. Radovanović, D. Mladenić, and M. Ivanović, “The role of hubness in clustering high-dimensional data,” *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 3, pp. 739–751, 2014.

[26] R. Feldbauer and A. Flexer, “Centering versus scaling for hubness reduction,” in *Artificial Neural Networks and Machine Learning – ICANN 2016*, A. E. Villa, P. Masulli, and A. J. Pons Rivero, Eds., ser. Lecture Notes in Computer Science, vol. 9886, Cham: Springer International Publishing, 2016, pp. 175–183.

[27] Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs,” *ArXiv e-prints*, Mar. 2016. arXiv: 1603.09320 [cs.DS].

[28] M. Aumüller, E. Bernhardsson, and A. Faithfull, “ANN-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms,” in *International Conference on Similarity Search and Applications*, Springer, 2017, pp. 34–49.

[29] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt, “Practical and optimal LSH for angular distance,” in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 1225–1233.

[30] L. Boytsov and B. Naidan, “Engineering efficient and effective non-metric space library,” in *Proceedings of the 6th International Conference on Similarity Search and Applications SISAP, A Coruña, Spain*, 2013, pp. 280–293.

[31] I. Guyon, S. R. Gunn, A. Ben-Hur, and G. Dror, “Result analysis of the NIPS 2003 feature selection challenge,” in *Advances in Neural Information Processing Systems 17 (Vancouver, Canada]*, 2004, pp. 545–552.

[32] M. Lichman, *UCI machine learning repository*, 2013. [Online]. Available: http://archive.ics.uci.edu/ml.

[33] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, 27:1–27:27, May 2011.

[34] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, “Openml: Networked science in machine learning,” *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013.

[35] M. A. Shafique and E. Hato, “Travel mode detection with varying smartphone data collection frequencies,” *Sensors*, vol. 16, no. 5, p. 716, 2016.

[36] M. Leodolter, *IncDTW: Incremental calculation of dynamic time warping*, R package version 1.0.3, 2018. [Online]. Available: https://CRAN.R-project.org/package=IncDTW.

[37] J. Chen, H.-r. Fang, and Y. Saad, “Fast Approximate kNN Graph Construction for High Dimensional Data via Recursive Lanczos Bisection,” *Journal of Machine Learning Research*, vol. 10, pp. 1989–2012, 2009.

[38] W. Dong, C. Moses, and K. Li, “Efficient k-nearest neighbor graph construction for generic similarity measures,” in *Proceedings of the 20th international conference on World wide web*, ACM, 2011, pp. 577–586.

[39] V. Athitsos, M. Potamias, P. Papapetrou, and G. Kollios, “Nearest neighbor retrieval using distance-based hashing,” in *24th IEEE International Conference on Data Engineering*, Apr. 2008, pp. 327–336.