KISS — A fast kNN-based Importance Score for Subspaces

Anna Beer
LMU Munich
Munich, Germany
beer@dbs.ifi.lmu.de

Ekaterina Allerborn
LMU Munich
Munich, Germany

Valentin Hartmann
EPFL
Lausanne, Switzerland
valentin.hartmann@epfl.ch

Thomas Seidl
LMU Munich
Munich, Germany
seidl@dbs.ifi.lmu.de

ABSTRACT

In high-dimensional datasets some dimensions or attributes can be more important than others. Whereas most algorithms neglect one or more dimensions for all points of a dataset or at least for all points of a certain cluster together, our method KISS (***k***NN-based **I**mportance **S**core of **S**ubspaces) detects the most important dimensions for each point individually. It is fully unsupervised and does not depend on distorted multidimensional distance measures. Instead, the k nearest neighbors (kNN) in one-dimensional projections of the data points are used to calculate the score for every dimension's importance. Experiments across a variety of settings show that those scores reflect well the structure of the data. KISS can be used for subspace clustering. What sets it apart from other methods for this task is its runtime, which is linear in the number of dimensions and $O(n \log(n))$ in the number of points, as opposed to quadratic or even exponential runtimes for previous algorithms.

1 INTRODUCTION

As sensors in fields like biology and chemistry become more and more sophisticated, websites collect more and more data about their users, and IoT and manufacturing devices get equipped with sensors that allow for predictive maintenance, the amount, granularity and dimensionality of data increases. In order to still be able to analyze the data in a meaningful way and in a reasonable time, one often needs to reduce at least one of the three; this paper will focus on the dimensionality. The more dimensions there are, the more of them are not important and distort further data mining tasks. The more dimensions, the longer it takes to process them all: the running time of many algorithms increases exponentially with the number of dimensions, especially of those designed for fewer dimensions. But not only that: many distance measures become more and more useless with an increasing number of dimensions [4]. Thus, instead of dragging along all dimensions of a point, many methods focus on working on only a small subset of the dimensions. The dimensions that a point is reduced to should, of course, be the ones that capture the most relevant information that this point contains. But to learn those important dimensions proves difficult: users do not want to waste time studying the data and its features thoroughly before applying a data mining algorithm. However, most algorithms still require user input that needs expert knowledge, or even require the user to label data by hand. In addition, the number of possibly important subspaces is exponential, making it a complex and time-consuming task to find the most important one. As datasets grow in size and contain data from different sources, one part of a dataset might differ a lot from a different part. Nonetheless, methods for dimensionality reduction typically try to find a common subspace for all data points, which can potentially be completely unsuited for heterogeneous data. Often, every single point has its own properties and thus the importance of a subspace may vary for each point, as shown in Fig. 1: for objects 1,4,7 in the first column the texture may be relevant, whereas it does not seem to be important for the other objects, since every other texture only occurs once. Also, for objects 1,2,3 in the first row and the quadrangles 4 and 5 in the second row the number of corners may be important. The color could be the best attribute to distinguish objects 1,5, and 9 in the diagonal from the others. Thus, for object 1 all three considered dimensions — number of corners, color, and texture — may be relevant, while there are other objects in the same dataset for which not all of those dimensions are important, e.g., for object 7 only the texture is relevant.

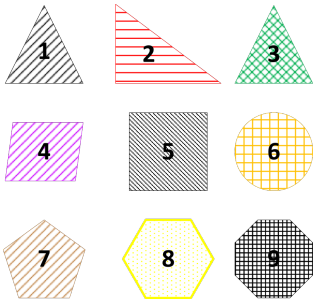

Figure 1: For different objects, different attributes or subspaces can be relevant: texture, number of corners, color, or a subset of those dimensions may be important.

A method to score the importance and expressiveness of each dimension for every point of a dataset individually without requiring any user input that scales to high dimensionalities would solve the problems mentioned above. In this paper, we develop KISS, a ***k***NN-based **I**mportance **S**core of **S**ubspaces, which fulfills all of these requirements. KISS can detect the most important subspace for a point fast and reliably in highly noisy data and data where only few dimensions are important per point.

One of the fundamental considerations that led to KISS is that those dimensions are most expressive for a point whose values lie in a cluster. We use the observation that if a point lies in a cluster in a certain subspace, the kNN of the projections of this point onto each dimension of this subspace intersect heavily. Since kNN in one-dimensional projections of the data can be computed fast, we

can efficiently calculate a score indicating the likelihood of the point lying in a cluster in the considered dimension. Usage of the kNN prevents relying on non-expressive distance measures, and there is no need for the users to know the data beforehand. KISS is deterministic, simple, fully unsupervised, and scalable w.r.t. the number of points as well as to the number of dimensions. It is easy to implement and reliably detects important dimensions for individual points fast. Our main contributions are as follows:

- We develop KISS, an importance scoring for every dimension for each individual point.
- KISS is fast w.r.t. both the number of points as well as the number of dimensions.
- KISS is fully unsupervised
- KISS does not rely on any multidimensional distance measure that gets useless for a high number of dimensions.

2 RELATED WORK

The problem of finding a global important subspace for all points has been addressed in previous work, which we introduce in Section 2.1. We restrict ourselves to algorithms that, like KISS (and contrary to, e.g., PCA or FOSSCLU), work in the standard basis of the vector space, as it simplifies getting insights into the data, which KISS was developed for.

2.1 Subspace Search

There exists some work on scoring of dimensions, where RIS, SURFING, and SCHISM are some of the most common algorithms.

RIS [6] produces a ranked list of all dimensions using a density-based quality criterion ("interestingness") that requires multiple parameters, which are set based on heuristic methods. The rating is only a relative comparison between different dimensions of the same dataset and is the same for all points.

SURFING [3] is a bottom-up approach that also returns the most "interesting" subspaces of a dataset. It is, like KISS, based on kNN, declaring subspaces as interesting in which "the k-nn-distances of the objects differ significantly from each other" [3]. The kNN distances are computed w.r.t. the subspaces, making their expressiveness dependent on the dimensionality of the subspaces. The algorithm has a runtime complexity of $O(mn^2)$, where n is the number of points and m is the number of different subspaces analyzed, which is 2^D in the worst case, making it much less scalable regarding both the number of points as well as the number of dimensions. Additionally, the minimum cluster size k has to be specified by the user.

SCHISM [8] extends the CLIQUE [2] principle and looks at the density of grid cells using an adaptive threshold function τ given by the user and applying the Chernoff-Hoeffding bound. It uses several preprocessing steps and requires three user given parameters u, τ, and ξ. Like RIS, and in contrast to KISS, it calculates a global score for "interesting" subspaces that is not adapted to individual points.

Although the dimension weightings at first glance seem to be suitable for comparing with KISS, such a comparison proves difficult: These dimension scoring methods do not return important subspaces for each point individually, or require at least two parameters set by the user, making it hard to objectively evaluate without overoptimism. But most notably, they are far more complex: The fastest of them, RIS, has runtime at least quadratic in the number of dimensions as well as the number of points. SCHISM is only linear in the number of points, but exponential in the number of dimensions. SURFING is quadratic in the number of points and exponential in the number of dimensions.

2.2 Subspace Clustering

We do not perform any clustering in this paper, but since we define "important dimensions" as dimensions in which a point lies in a cluster, there is a relation to the field of subspace clustering. Even though subspace clustering algorithms also deliver important dimensions in a way, their focus is different from KISS. Whereas those algorithms often need to perform a complete clustering of the dataset, we aim to get the relevant subspaces directly, individually for every point. We do not need to know the precise clusters to find important dimensions. Also, most of those algorithms rely on parameters that are not easy to set. We found that especially our first two goals, being fully unsupervised, and returning individual scores for different points, are to the best of our knowledge not achieved simultaneously by any other algorithm in this field. COSA and DISH are most related to our work, since they both consider subspaces for individual points:

COSA [5] finds important subspaces individually for each point using the kNN. A hierarchical clustering is applied based on a dimension weighting matrix and the relevant dimensions can be calculated based on the dimension weights of the respective cluster members. Despite the similarities to KISS, there are two major differences: First, users need to set a not quite intuitive parameter λ, which gives the "strength of incentive for clustering on more dimensions" [7]. Second, the kNN are calculated in the full-dimensional space, making COSA vulnerable to the loss of expressiveness of distance measures in high dimensions.

DiSH [1] is a density-based algorithm that finds cluster hierarchies and nested clusters. It has two parameters: a smoothing factor μ representing the minimum number of points in a cluster and ε for ε-range queries. Even though DiSH also uses only one-dimensional range queries and delivers subspace preference vectors for every point, the nesting of subspaces makes it impossible to determine the distinctly important subspaces. Also, the vectors are only calculated in an intermediate step, and depend on the parameter choices.

2.3 Possible Competitors

Finding suitable methods to compare KISS to is difficult: there are a number of subspace clustering algorithms, but they perform clustering, and not detection of the most important subspaces. For some algorithms one could extract the important subspaces of a point by looking at the subspace of the cluster the point was assigned to. This assignment, however, can only be obtained after an expensive clustering of the complete dataset. Some algorithms like, e.g., RIS or SURFING rank the subspaces in a similar way as KISS scores them, but they deliver one ranking for the complete dataset, not for each point individually. To the best of our knowledge, there is no algorithm yet that fulfills all of the requirements we impose. In particular, returning individual dimension ratings for each point and being completely unsupervised are very rare properties. Nevertheless, to at least have *some* point of reference, we exemplarily compare against CLIQUE, which is a grid-based bottom-up approach for subspace clustering. It requires two parameters, ξ and τ, which determine the number of intervals every dimension is partitioned into and the density threshold. We try out different parameter settings, showing that the results are very

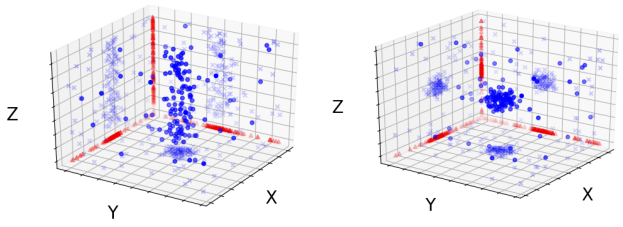

Figure 2: Projections of a 2-dimensional resp. 3-dimensional cluster. Blue crosses are projections onto the 2-dimensional subspace, red triangles projections onto one dimension.

sensitive to the parameter choices, whereas for KISS no parameters need to be tuned. In addition, we use the quality criterion of SURFING to obtain scores for every dimension and compare KISS to them. However, those are global scores for all points and not individual ones like those computed by KISS.

3 KISS

This section presents our newly developed dimension score KISS. We first describe the basic idea to use the k nearest neighbors in one-dimensional projections of the n data points to be able to compute KISS, which gives a scoring for the importance of every dimension for each individual point. Section 3.2 motivates our idea mathematically. In Section 3.3 we develop the exact formula of KISS, and present the complete KISS-based algorithm to obtain the most important subspace of a point. We analyze the complexity of our algorithm in Section 3.4.

3.1 Idea: Using One-dimensional kNN

If a point lies in a cluster in a d-dimensional subspace, most of the kNN of this point in the projection of the dataset onto those d dimensions will be members of that cluster, too. If we look at only one of those d dimensions (cf. the red triangles in Fig. 2), we still see the cluster structure: the cluster on the left lies in dimensions X and Y, and the red triangles on the according axes show a clear cluster structure. Projected onto those axes, most of the one-dimensional kNN of a point will lie in the same (original) d-dimensional cluster as the point itself.

Following this observation, for a given point p, we count for every other point q in how many dimensions it belongs to the one-dimensional kNN of p, giving us a Point Score $PS(p, q)$. A high Point Score means that q is likely to be contained in the same (higher-dimensional) cluster as p, meaning that the dimensions that they share carry more importance for p than the others. Summing up the Point Scores for each dimension individually gives us a measure for the importance of each dimension, where we account for outliers by incorporating the one-dimensional distances to the kNN of a point.

3.2 Mathematical Perspective

In the following we give some theoretical insights which support our idea. We denote cluster indices by superscripts and dimension indices by subscripts, and see clusters as collections of points drawn from a common probability distribution over $\mathbb{R}^D$.

Consider a cluster C^1 with center c^1 in dimensions $\{1, \dots, l\}$ (without loss of generality). We will consider the neighborhood of points in dimension 1. Let C^2 be a different cluster with center c^2 that overlaps with cluster C^1 in dimension 1, and assume that for all points $r^1 \in C^1$ and all $r^2 \in C^2$ we have $\Pr(|r_1^1 - c_1| \leq \varepsilon) \geq \delta$ and $\Pr(|r_1^2 - c_1| \leq \varepsilon) \geq \delta$, respectively, where $\varepsilon > 0$ and δ is a value close to 1. Furthermore, C^1 and C^2 should not overlap and be sufficiently far apart in the dimension that they share: $|c_1^1 - c_1^2| \leq 4\varepsilon + \varepsilon'$ for an arbitrarily small $\varepsilon' > 0$. This is, e.g., the case for all shared dimensions with high probability if c^1 and c^2 are uniform samples from a sufficiently large set in $\mathbb{R}^D$.
Let $p^1, \tilde{p}^1 \in C^1$, and $p^2 \in C^2$. In addition, let q be a point that does not lie in any cluster in dimension 1 and is drawn from a somewhat uniform distribution on a large enough interval. Precisely, we require it to fulfill $\Pr(|q - c_{1,2}| \leq 3\varepsilon) \leq \delta'$, where δ' is close to 0.
We now consider the distance of p^1 to the other three points in dimension 1.
For the point from the same cluster, we get

$$\Pr(|p_1^1 - \tilde{p}_1^1| \leq 2\varepsilon) \geq \Pr(|p_1^1 - c_1^1| + |c_1^1 - \tilde{p}_1^1| \leq 2\varepsilon)$$
$$\geq \Pr(|p_1^1 - c_1^1| \leq \varepsilon) \Pr(|c_1^1 - \tilde{p}_1^1| \leq \varepsilon) \geq \delta^2 \approx 1.$$

The point from the other cluster yields

$$\Pr(|p_1^1 - p_1^2| \leq 2\varepsilon) \leq \Pr(|p_1^1 - c_1^2| > \varepsilon \text{ or } |p_1^2 - c_1^1| > \varepsilon)$$
$$\leq \Pr(|p_1^1 - c_1^2| > \varepsilon) + \Pr(|p_1^2 - c_1^1| > \varepsilon) = 2(1 - \delta) \approx 0,$$

where for the first step we observed that at least one of p_1^1, p_1^2 needs to lie outside of the ε-interval around its cluster's center, and applied a simple union bound to obtain the second inequality. Finally, for the point that does not lie in any cluster in dimension 1, we have, using the same arguments as above,

$$\Pr(|p_1^1 - q_1| \leq 2\varepsilon) \leq \Pr(|p_1^1 - c_1^1| > \varepsilon \text{ or } |q_1 - c_1^1| \leq 3\varepsilon)$$
$$\leq \Pr(|p_1^1 - c_1^1| > \varepsilon) + \Pr(|q_1 - c_1^1| \leq 3\varepsilon) \leq (1 - \delta) + \delta' \approx 0.$$

Thus, if k is chosen smaller than the size of C^1, the kNN of p_1^1 will almost exclusively consist of points from C^1. Thus,

$$\mathbb{E}(|\{i \in \{1, \dots, l\} \mid \tilde{p}_i^1 \text{ is part of the } k\text{-NN of } p_i^1\}|) \approx l\frac{k}{|C^1|} > l\frac{k}{n},$$

where the last quantity corresponds to a uniform distribution of points.

3.3 The Full Algorithm

KISS, the score indicating the importance of a dimension d for a point p in a dataset DB, depends on the k nearest neighbors (kNN) of p in the one-dimensional projection onto d: $kNN_p(d)$. Note that their number can be larger than k in case of ties, since we chose the deterministic variant of kNN.

The more often a point occurs in the sets of one-dimensional nearest neighbors of p, the closer related it is to p, which we capture in the *Point Score* $PS(p, q)$, where $\mathbb{1}$ is the indicator function: $PS(p, q) = \sum_{d=1}^{D} \mathbb{1}\{q \in kNN_p(d)\}$.

Higher values for k lead to more accurate scores, as shown in Fig. 3. However, the runtime of our algorithm depends on k and we would like to keep it $O(n \log(n))$ in the number of points (see our complexity analysis in Section 3.4). For this reason, k is set to $\sqrt{n}$, which is also in line with previous literature [5].

The importance a dimension d has for a point p depends not only on the intersection of the kNN in this dimension with the kNN in the other dimensions, but also on the distance of those kNN. Otherwise, the important dimensions for outliers would be distorted. Thus, the farther away a point in the kNN is, the less influence it should have on the importance of the respective dimension, which is why we divide the Point Score of each

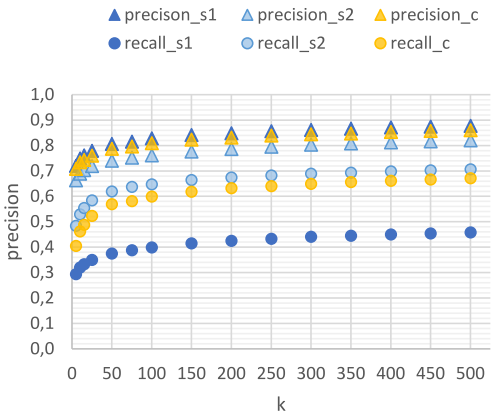

Figure 3: Results for different values of k. _s1, _s2, and _c denote different binarization methods, see Sec. 4.

$q \in kNN_p(d)$ by the distance between the corresponding projections of q and p. Additionally, the computed value is divided by the neighborhood size to account for ties among the nearest neighbors:

$$KISS'(p,d) = \frac{1}{|kNN_p(d)|} \sum_{q \in kNN_p(d)} \frac{1}{dist(p_d, q_d)} PS(p,q) \quad (1)$$

Finally, KISS is normalized for every point by dividing every value by the highest KISS occurring for the respective point:

$$KISS(p,d) = \frac{KISS'(p,d)}{\max_{e \in \{1,\ldots,D\}} KISS'(p,e)}. \quad (2)$$

This gives a value between 0 and 1 and allows for a meaningful comparison between different points of a dataset.

3.4 Complexity

The calculation of the kNN of all points in all dimensions needs $O(D * k * n + D * n * \log(n))$ steps, where D is the number of dimensions in the data set DB of size $|DB| = n$. Computing the kNN in one dimension can be performed efficiently by sorting the points w.r.t. this dimension and going to the left and right of the query point in the sorted list. Given the kNN of every point for every dimension, all Point Scores $PS(p,\cdot)$ w.r.t. a point p can be calculated in $O(D*k)$ by iterating through the k nearest neighbors of p in all D dimensions, keeping track of the scores via a hashmap where they get continuously updated. This has to be done for all n points, resulting in $O(n * D * k)$. For calculating the KISS for a point p and a dimension d, we need to sum up the Point Scores of all of p's kNN in d divided by their (one-dimensional) distance in this dimension, which can be done in $O(1)$. The summation can be performed in $O(k)$. We want to compute the KISS for all points and all dimensions, thus we get $O(n * D * k)$.

The KISS for all dimensions and all points can hence be computed in time $O(D*k*n+D*n*\log(n)+n*D*k+n*D*k) = O(D*n*(k+\log(n))$, which is linear in the dimension D and close to linear in the size of the dataset n. Runtime experiments confirmed this behavior, but were omitted due to space constraints.

4 EXPERIMENTAL EVALUATION

In Section 4.1 we introduce a technical tool that is needed to validate our method against a ground truth. In Section 4.2 we describe our experiments, and summarize the results in Section 4.3.

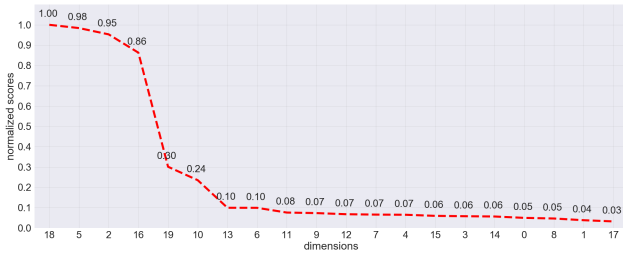

Figure 4: Typically distributed scores for different dimensions for a point p, sorted by descending normalized score.

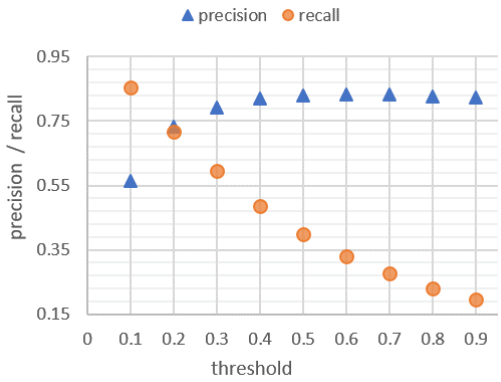

Figure 5: Precision and recall using simple binarization with different thresholds.

4.1 Binarization

To be able to validate our results and because for certain applications a division of the dimensions into important and unimportant ones can be needed, we suggest two possibilities to binarize the values obtained by KISS. Ordered by score value, a typical distribution of the scores for a point is shown in Fig. 4. If a point lies in a cluster, the KISS of the according dimensions clearly differs from the KISS of unimportant dimensions.

The naïve approach "simple binarization" of using a fixed threshold for the normalized score based on which we set the score to either 0 or 1, already delivers good results, as we show in Section 4.2. Fig. 5 shows recall and precision for our base case experiment and different values for the threshold, where 0.2 offers a good trade-off between the two. We performed the other experiments with the thresholds 0.5 and 0.2, denoted by _s1 and _s2, respectively.

Additionally, we developed a more sophisticated approach — "complex binarization" —, which comes with an only negligable increase in runtime, to improve our results even further. Here, we look for the most appropriate cut position in the ranked scores: e.g., for the KISS distribution depicted in Fig. 4 it could be dimension 12 since the scores for all dimensions to the right of it are significantly lower than the ones to the left of it. Our approach for detecting this cut position in the score ranking consists of first setting the importance of each dimension to 1 and then lowering it to 0 if its KISS lies below one of the three thresholds described below.

We set all parameters required for the complex binarization to the same reasonable values we give below for all experiments. Both strategies are introduced mainly to be able to validate our results against a binary ground truth. Note that the parameter values rely on the data being scaled to the D-dimensional unit

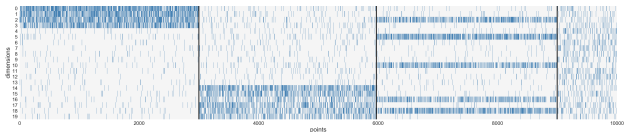

Figure 6: Transpose of the binary matrix for the base case dataset computed using KISS with complex binarization. The subspace boundaries are depicted as black lines.

hypercube. The three thresholds for the complex binarization are:

(1) *normalized threshold* t_n: If $KISS(p,d) < t_v = 0.1$, then $KISS(p,d)$ is set to 0.
(2) *unnormalized threshold* t_u: Because we normalize the KISS of each dimension by dividing by the largest KISS for this point, even a point that is just random noise has at least one dimension with score 1. However, the unnormalized value of dimensions with a high normalized KISS for this noise point will be significantly lower than the unnormalized values of dimensions with a high normlized KISS for a point that lies in a cluster. Thus, in addition to setting a threshold for the normalized KISS, we also set one for the unnormalized KISS' (cf. Equation 1): if $KISS'(p,d) < t_u$, the score for d is set to 0, where t_u equals the difference between mean and minimum of all unnormalized scores.
(3) *descent threshold* t_d: The descent threshold controls the decline between consecutive KISS values. If $\frac{KISS(p,e)-KISS(p,d)}{KISS(p,e)} > t_d = 0.7$, where $KISS(p,e)$ is the next largest KISS value of p, then all KISS values smaller than or equal to $KISS(p,d)$ become 0.

The result of the binarization can be expressed in a binary matrix as in Fig. 6.

Empirically, setting t_n significantly lower than 0.5 and t_d rather high allows for detecting more relevant dimensions and therefore detecting subspaces of higher dimensionality. t_u affects mostly how well outliers are detected. However, setting this parameter too high leads to very restricted binarized scores and can possibly decrease the detection rate of the important dimensions of the cluster points. In general, the parameters allow us to trade precision for recall. KISS is supposed to be used in settings where working with the original data without a significant reduction of the dimensionality is infeasible, either due to limited human capacity when manually analyzing the data or due to non-favorable dependence of a downstream task's performance on the number of dimensions. Hence we can live with a mediocre recall if in return the precision is high, allowing us to get rid of many dimensions, which is why we mainly focus on achieving a high precision.

In real-world settings where one needs a binary division of the dimensions, one typically has a (computational or storage) budget of dimensions one can deal with in the downstream task, and would binarize in a way so that exactly this many dimensions are labeled as important.

4.2 Experiments

We test with both the simple and complex binarization of KISS and denote the corresponding values with the abbreviations _s and _c, respectively. To have ground truth values for the importance of all dimensions, we generated data containing subspace clusters with possibly overlapping subspaces. The clusters are

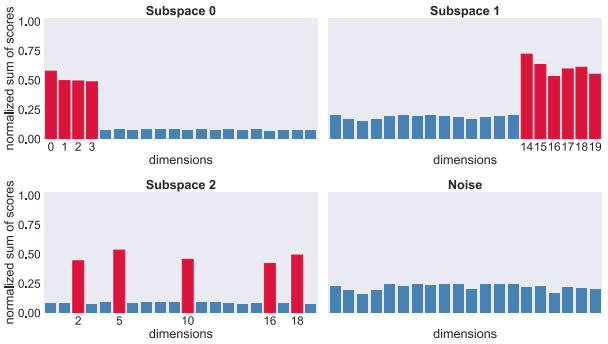

Figure 7: Average KISS for all points, partitioned according to ground-truth based important subspaces. Red bars show dimensions containing clusters.

Gaussians with mean randomly drawn from the uniform distribution on the D-dimensional unit hypercube. The values for the dimensions of a point that do not lie in a cluster are uniformly distributed in the hypercube [1].

Looking at the distribution of KISS per important (according to the ground truth) subspace in Fig. 7, we already see the correlation to the cluster subspaces: the average scores for the important dimensions (red bars) are visibly higher than those for unimportant dimensions (blue bars). Noise points that do not lie in any cluster are shown in the lower right diagram: the average KISS values do not differ much. The precision we achieve for both types of binarization are good, as can be seen in Fig. 8.

To the best of our knowledge there are no alternatives yet to KISS (see Section 2.3). However, with CLIQUE and SURFING we compare KISS to representative algorithms for subspace clustering and for subspace search. The comparison makes the disadvantages of having to set parameters as well as the benefit of individual scores in contrast to a global ranking clear.

Among others, Fig. 8 shows results obtained with CLIQUE for different parameter configurations. Even though CLIQUE was able to obtain high recall values, the precision was even for the best parameter settings much lower than for KISS (for both binarization methods). We also see that the results heavily depend on the choice of CLIQUE's parameters, with precision ranging from 35% to 77% and recall from 53% to 87%. Fig. 8 further includes the results obtained by binarizing the "quality" of each particular dimension as computed by SURFING for different values of k (in the same way as we binarize the KISS values). The classification performance of SURFING is very dependent on the parameter choice as well. With a good choice, it is able to achieve a high recall, but, as expected, the precision values are rather low, since the quality assignments are the same for all points, which does not match the ground truth.

Starting from this base case, we altered one parameter of the data in each of the following subsections to investigate KISS' behaviour w.r.t. this parameter.

4.2.1 Number of points. Changing the number of points n did not affect the precision, recall or accuracy of KISS significantly.

[1] The settings for our base case dataset are as follows: number of points $n = 10000$, dimensionality of point p: $dim(p) = 20$, percentage of noise $noise = 0.1$, set of dimensions in subspace S_i: $S_0 = \{0 \dots 3\}$, $S_1 = \{14 \dots 19\}$, $S_2 = \{2, 5, 10, 16, 18\}$, percentage of points w.r.t. n lying in subspace S_i: $|S_i| = [0.3, 0.3, 0.3]$, dimensionality of S_i: $dim(S_i) = [4, 6, 5]$, number of clusters in subspace S_i: $n_c(S_i) = [1, 2, 1]$, variance of cluster C_i: $var(C_i) = [1.5, 1.0, 1.3]$.

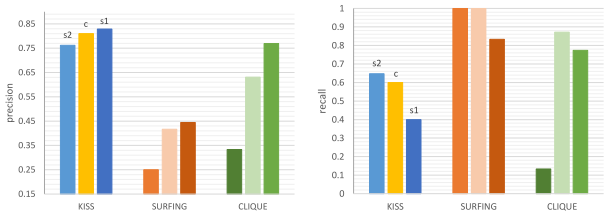

Figure 8: Results for KISS, SURFING, and CLIQUE for the base case dataset and different parameters ξ and τ resp. k. Same color indicates same parameters.

The three values deviated by at most 3% for $n \in \{5\,000, 10\,000, 25\,000, 50\,000, 75\,000, 100\,000\}$.

4.2.2 Number of dimensions. With growing number of dimensions (while keeping the ratio of dimensions lying in important subspaces the same) the recall decreases, but the precision, which we put more emphasis on, stays high.

4.2.3 Noise. Increasing the percentage of pure noise points leads to less points in each cluster, thus precision drops. Nevertheless, the decrease of quality is slow, and up to 50% of data can be pure noise points before precision falls below 75% (for the complex binarization).

4.2.4 Number of subspaces. We examine KISS for up to 10 different subspaces and obtain good results with precisions above 73% in all cases. Additionally, recall as well as accuracy diminish only slightly with increasing number of subspaces.

4.2.5 Subspace size ratio. We tested several size ratios between the three base case subspaces our dataset consists of. In our base case, every subspace contains one third of the non-noise data points. We set the share of instances in the first subspace to values $\{0.4, 0.5, 0.6, 0.7, 0.8\}$, while dividing the remaining points equally among the other two subspaces (and additionally keeping the 10% noise of the base case). The quality of the scores hardly changed: we received precision values for the simple binarization between 83% and 85%, and between 84% and 86% for the complex binarization. Recall values ranged between 39% and 41%, and 47% and 49%, respectively, showing that the size ratio of the subspaces does not constitute a problem for KISS. Thus, even subspaces containing only very few points of the complete dataset can be found as easily as bigger subspaces.

4.2.6 Number of clusters per subspace. With an increasing number of clusters, precision as well as recall decrease, since there are fewer points per cluster that could help identify a point in the cluster.

4.2.7 Density of clusters. We tested several density settings for the clusters. When all subspaces contain similarly dense clusters, the quality decreases with lower density (i.e., higher standard deviation). If each subspace contains differently dense clusters, the results are rather determined by the average density than by the lowest or highest occurring density. Thus, a large difference in cluster density does not influence the results negatively.

[1]When adding more subspaces to the base case dataset, we use the same settings as for the original subspaces: the points are evenly distributed among the subspaces, and the cluster settings of the clusters lying in subspaces S_{0+3i}, S_{1+3i}, S_{2+3i} correspond to the settings of the clusters lying in subspaces S_0, S_1, S_2.

4.3 Summary of Results

Our experiments show that KISS achieves a high precision and reasonable recall across a wide range of settings. With a high number of subspaces or clusters the performance starts to degrade, but KISS is robust to noise and can deal with high numbers of points as well as clusters of different density. We would like to point out that the experiments only show a small part of KISS' capabilities, since the original KISS is a continuous value, which we just binarized here, and likely not even optimally.

5 CONCLUSION AND FUTURE WORK

We developed KISS, a scoring that assigns an importance value to each dimension of each point of a dataset. It is scalable, does not suffer from the curse of dimensionality, since it replaces multi-dimensional distance measures by one-dimensional ones, and does not require significant user involvement to set parameters. Its runtime is linear in the dimensionality and close to linear in the number of points, setting it apart from similar methods.

KISS has numerous applications, both as a tool to get an insight into datasets as well as a foundation for data mining applications, in particular to accelerate downstream tasks or to make them more robust to noise. We are currently working on some of the most immediate extensions: (1) performing clustering using especially the most relevant dimensions for each point; and (2), using KISS for outlier and noise detection, following the observation that points that have a low KISS in every dimension are typically in none of those in a cluster. We encourage the usage of KISS for preprocessing data and gaining knowledge in an early stage of a data anlysis process, since it is simple, fast, delivers good results and does not require parameter tuning.

ACKNOWLEDGMENTS

This work has been partially funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibilities for its content.

REFERENCES

[1] Elke Achtert, Christian Böhm, Hans-Peter Kriegel, Peer Kröger, Ina Müller-Gorman, and Arthur Zimek. 2007. Detection and visualization of subspace cluster hierarchies. In *International Conference on Database Systems for Advanced Applications*. Springer, 152–163.

[2] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. 1998. *Automatic subspace clustering of high dimensional data for data mining applications*. Vol. 27. ACM.

[3] Christian Baumgartner, Claudia Plant, K Railing, H-P Kriegel, and Peer Kroger. 2004. Subspace selection for clustering high-dimensional data. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*. IEEE, 11–18.

[4] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. 1999. When is "nearest neighbor" meaningful?. In *International conference on database theory*. Springer, 217–235.

[5] Jerome H Friedman and Jacqueline J Meulman. 2004. Clustering objects on subsets of attributes (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 66, 4 (2004), 815–849.

[6] Karin Kailing, Hans-Peter Kriegel, Peer Kroeger, and Stefanie Wanka. 2003. Ranking interesting subspaces for clustering high dimensional data. In *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 241–252.

[7] Lance Parsons, Ehtesham Haque, and Huan Liu. 2004. Subspace clustering for high dimensional data: a review. *Acm Sigkdd Explorations Newsletter* 6, 1 (2004), 90–105.

[8] Karlton Sequeira and Mohammed Zaki. 2004. SCHISM: A new approach for interesting subspace mining. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*. IEEE, 186–193.