Instance space analysis for unsupervised outlier detection

Sevvandi Kandanaarachchi[*], Mario A. Muñoz[†], Kate Smith-Miles[‡]

Abstract

The success of various methods for unsupervised outlier detection depends on how well their definition of an outlier matches the dataset properties. Given that each definition, and hence each method, has strengths and weaknesses, measuring those properties could help us to predict the most suitable method for the dataset at hand. In this paper, we construct and validate a set of meta-features that measure such properties. We then conduct the first instance space analysis for unsupervised outlier detection methods based on the meta-features. The analysis provides insights into the methods' strengths and weaknesses, and facilitates the recommendation of an appropriate method with good accuracy.

1 Introduction

The applications of outlier detection include taks as diverse as identifying fraudulent credit card transactions, and recognising terrorist plots in social media. The significance of these applications has spurred recent research efforts, leading to a steady growth in the number of detection methods [1], and with a variety of definitions of an outlier. As result, it is not uncommon for the methods to disagree on the number and location of the outliers. It therefore becomes challenging to identify the most suitable outlier detection method for the problem at hand. Indeed, it is unlikely that a single method outperforms all others in all applications [2, 3]. Therefore, on a new problem, our focus should be on understanding the suitability of various methods, by exploiting the experience gained through solving similar problems.

This paper extends the computational study of Campos et al. [4] by making the first exploration of algorithm selection for outlier detection methods. Using a comprehensive set of meta-features, i.e., measures of dataset properties, we are able to predict the suitability of a method for a given problem. Moreover, we perform the the first *instance space analysis* [5, 6, 7], which helps us gain further understanding of the comparative strengths and weaknesses of an outlier detection method, more nuanced than a summary table of statistical results can offer. As a resource for the research community, we make available our repository of more than 12000 datasets [8]. Moreover, we provide the R package `outselect` [9], containing the code to calculate all the meta-features, as well as the knowledge based used to train the instance space model. Hence, `outselect` facilitates the end-to-end evaluation for a new dataset.

2 Experimental setting

To demonstrate that an outlier detection method has regions of "good" performance within the instance space, we define an experimental setting including: (a) a portfolio of methods, (b) a collection of instances, and (c) a definition of good performance based on an evaluation metric. In the next sections, we describe the details of each component of our setting. We note that that the methodology would not change even if the experimental setting does.

2.1 Outlier detection methods We investigate 12 outlier detection methods studied by Campos et al. [4] using the ELKI software suite [10]. These are:

1. k nearest neighbours (KNN) [11]
2. KNN weight (KNNW) [12]
3. Outlier Detection using In-degree Number (ODIN) [13]
4. Local Outlier Factor (LOF) [14]
5. Simplified LOF [15]
6. Connectivity based Outlier Factor (COF) [16]
7. Influenced Outlierness (INFLO) [17]
8. Local Outlier Probabilities (LOOP) [18]
9. Local Density based Outlier Factor (LDOF) [19]
10. Local Density Factor (LDF) [20]
11. Kernel Density Estimation Outlier Score (KDEOS) [21]
12. Fast Angle Based Outlier Detection (FastABOD) [22]

All of these methods have as a free-parameter the neighbourhood size, k. Noting that no single value of k applies to all scenarios, we use a simple heuristic to select its value depending on the dataset, and use the same value across all methods. Arguably, k can be finely-tuned to improve the performance of each method. However, a detailed analysis of the impact of k is outside of the scope of this paper. The value is calculated as follows:

$$k(\text{dataset}) = \min\left(\lfloor N/20 \rfloor, 100\right) \tag{2.1}$$

where N is the number of observations in the dataset. The maximum of $k = 100$ limits the number of computations that can result from a large dataset. The motivation for choosing

[*]Faculty of Business and Economics, Monash University, Clayton VIC 3800, Australia. sevvandi.kandanaarachchi@monash.edu

[†]School of Mathematics and Statistics, The University of Melbourne, Parkville, VIC 3010, munoz.m@unimelb.edu.au

[‡]School of Mathematics and Statistics, The University of Melbourne, Parkville, VIC 3010, smith-miles@unimelb.edu.au

$\lfloor N/20 \rfloor$ is as follows: Choosing $k = N$ would result in a very large neighbourhood making some methods such as KNN and LOF not discriminate enough between outliers and non-outliers. On the other hand, a very small value of k such as $k = 1$ may miss small clusters of outliers, as within that small cluster the density is high and distances are low. As such we choose a small value of k, that is not too small.

2.2 Datasets Our benchmark set contains approximately 12000 datasets, most of which were generated by following the approach by [4, 23]. First, we take one of 170 datasets [24] which originally represented a classification problem sourced from the UCI Machine Learning Repository. Then, we generate a new dataset by down-sampling one of the classes. We repeat this procedure for all of the classes, labelling the down-sampled class as the outliers. In addition we also use datasets from [4, 23] which have outlier class down-sampled at rates 5% and 2%. Moreover, to guarantee that the new datasets are fit for outlier detection benchmarking, we carry out the following tasks:

Down-sampling For a dataset with K classes, each class in turn is designated the outlier-class and observations belonging to that class are down-sampled such that the outlier percentage $p \in \{2, 5\}$. The down-sampling is randomly carried out 10 times to produce 10 variants for each value of p.

Categorical variables From each down-sampled dataset, we create one dataset with categorical variables removed, and one with categorical variables converted to numerical values using the method of inverse data frequency [4].

Duplicate observations Since the distance between duplicates is zero, which in turn leads to infinite densities, duplicate observations are removed to avoid numerical instability caused by division by zero errors.

Missing values For each attribute in each dataset, the number of missing values are computed. Similar to [4], if an attribute has less than 10% of missing values, the observations containing the missing values are removed, otherwise, the attribute is removed.

2.3 Evaluation metric While outlier detection is considered an unsupervised learning task, assessing algorithm performance requires a *ground truth* given by the outlier labels [4]. This makes it possible to define an evaluation metric. Although no single universally accepted metric exists, the most popular ones in outlier detection are: (a) the Area Under the Receiver Operator Characteristic curve (AUC), and (b) the Precision-Recall curve. Other metrics include precision at n [25], average precision [26], and a combination of positive and negative predictive values. In our study we use AUC as the evaluation metric, and define *good* performance of a method on a dataset if $AUC > 0.8$. Alternative definitions can be considered within the methodology, however a detailed study on the impact of this choice is beyond the scope of this proof-of-concept methodological paper.

Table 1: Types of features calculated

Feature category	Number of features
Generic : Simple, Statistical and Information theoretical	25
Density based	77
Residual based	35
Graph based	41
Total	178

2.4 Meta-features Given that our outlier detection datasets have been generated by down-sampling classification datasets, we are able to borrow many of the features that have been used to summarize interesting properties of classification datasets, and then add some additional features that are unique to the outlier detection challenge. We have categorized the features in two main classes, with the first being generic features which measure various aspects of a dataset but are not particularly tailored towards outlier detection. While they are broadly relevant, they offer little insight into the outlier structure of a dataset.

The second class of features measure properties of the outlier structure of a dataset using density, residual and graph-based characteristics. These use the outlier labels in their feature computation. The reason for using class labels in feature computation is two-fold. Firstly, different outlier methods may disagree on outliers and their rankings. As such, we need to know how each method declares its outliers. Are they density related, distance related or graph related outliers? Secondly, the use of ground truth in labelling outliers gives the opportunity to learn when each method's definition is suitable. Learning these relationships from training data enables identificaton of the most suitable outlier method for a given problem. This is common in industry applications where training data contains labelled outliers, and the task is to find effective methods which extract these outliers for future unlabelled data [27, 28].

We compute a broad set of candidate features, and later select a subset for the instance space analysis. Due to space constraints, we provide a brief overview of these features here. More details of these features are given in our gihub repository [9]. Table 1 summarizes the features by category.

Simple features These are related to the basic structure of a dataset, e.g. the number of observations.

Statistical features These relate to statistical properties such as skewness and kurtosis of a dataset.

Information theoretic features measure the amount of information present in a dataset, e.g. entropy of a dataset.

Outlier features In order to make our feature set richer we include density-based, residual-based and graph-based features. We compute these features for different subsets of the dataset, namely outliers, non-outliers and proxy-outliers, with the last subset being data points that are either far away from "normal" data, reside in low density regions, or have different graph properties compared to the rest. We define proxy-outliers using distance, density, residual and graph based perspectives. If there is a significant overlap between proxy-outliers and actual outliers, then we expect certain outlier detection methods to perform well on such datasets. Proxy-outlier based features fall into the category of landmarking, which has been popular in meta-learning studies [29, 30, 31].

1. **Density based features** are computed either using the density based clustering algorithm DBSCAN [32] or kernel density estimates.
2. **Residual based features** are computed by fitting a series of linear models and obtaining residuals.
3. **Graph based features** are based on graph-theoretic measures such as vertex degree, shortest path and connected components. The dataset is converted to a directed graph using the software [33] to facilitate this feature computation.

3 Predicting outlier detection method performance from meta-features

Using a set of 178 candidate features, we predict suitable outlier detection methods for given datasets. For this purpose, we train 12 Random Forest classifiers - one for each outlier method - with the 178 features as inputs and a binary predictor for *good* performance as the output. Table 2 provides the average cross-validation accuracy over 10 folds for each outlier detection method. As most outlier methods are only *good* for a small number of datasets, a naive prediction that performance is not good for all datasets achieves the default accuracy. Since our Random Forest accuracy is greater than the default accuracy for all methods, the meta-features are clearly able to help distinguish which methods are well suited for some datasets, even when these are a minority.

Table 2: Default and prediction accuracy of a random forest classifier using all the available features. Good performance is described as $AUC > 0.8$.

Method	Default accuracy	Prediction accuracy
COF	81%	86%
FAST ABOD	75%	85%
INFLO	88%	91%
KDEOS	94%	95%
KNN	73%	86%
KNNW	71%	86%
LDF	82%	88%
LDOF	84%	89%
LOF	80%	86%
LOOP	82%	88%
ODIN	84%	88%
SIMLOF	80%	87%

4 Instance space construction and analysis

Having validated that the features contain sufficient information to be predictive of outlier detection method performance, we now use the features to construct an instance space to visualise the relationships between dataset (instance) features and strengths and weaknesses of methods.

Our approach to constructing the instance space is based on the most recent implementation of the methodology [24]. Critical to both algorithm performance prediction and instance space construction is the selection of features that are both distinctive (i.e. uncorrelated to other features) and predictive (i.e correlated with algorithm performance). Therefore, we follow a procedure to select a small subset of features that best meet these requirements, using a subset of 2053 instances for which there are three well performing algorithms or less. This choice is somewhat arbitrary, motivated by the fact that we are less interested to consider datasets where many algorithms perform well or poorly, given our aim is to understand unique strengths and weaknesses of outlier detection methods.

We pre-process the data such that it becomes amenable to machine learning and dimensionality projection methods. Given that some features are ratios, one feature can often produce excessively large or infinite values. Hence, we bound all the features between their median plus or minus five times their interquartile range. Any not-a-number value is converted to zero, while all infinite values are equal to the bounds. Next, we apply Box-Cox transformation to each feature to stabilise the variance and normalise the data. Finally, we apply a z-transform to standardise each feature.

Starting with a set of 178 features, we determine whether a feature has unique values for most instances. A feature provides little information if it produces the same value for most datasets. Hence, we discard those that have less than 30% unique values, leaving 135 features. Then, we check the correlation between the features and AUC. Sorting the absolute value of the correlation from highest to lowest, we select the top three features per algorithm. This results in 26 features, as some of them are correlated to more than one algorithm and we do not need replicates. Next, we identify groups of similar features, using a k-means cluster-

ing algorithm and a dissimilarity measure of $1 - |\rho|$, where ρ is the correlation between two features. As result, we obtain seven clusters. Retaining one feature from each cluster, there are 2352 possible feature combinations. To determine the best one for the dimensionality reduction, we project each candidate feature subset into a 2-d space using PCA with two components. Then, we fit a random forest model per algorithm to classify the instances in the trial instance space into groups of good and bad performance, as defined in Section 2.3. That is, we fit 12 models for each feature combination, and define the best combination as the one producing the lowest average out-of-the-bag error across all models. This process results in 7 features finally being selected.

We then use the Prediction Based Linear Dimensionality Reduction (PBLDR) method [24] to find a projection from 7-d to 2-d that creates the most linear trends of algorithm performance and feature values across the instance space, to assist visualisation of directions of hardness and feature correlations. Given PBLDR's stochasticity, we calculate 30 different projections and select the one with the highest topological preservation, defined as the correlation between high- and low-dimensional distances. The final projection matrix is defined by Equation 4.2 to represent each dataset as a 2-d vector $\mathbf{Z}$ depending on its 7-d feature vector.

(4.2)

$$\mathbf{Z} = \begin{bmatrix} -0.0862 & -0.2078 \\ 0.1737 & 0.1845 \\ -0.0460 & -0.2847 \\ -0.0938 & -0.2025 \\ 0.1202 & 0.0378 \\ 0.1854 & -0.0822 \\ 0.3543 & -0.1325 \end{bmatrix}^{\top} \begin{bmatrix} \text{Mean\_Entropy\_Attr} \\ \text{IQR\_TO\_SD\_95} \\ \text{OPO\_Res\_ResOut\_Median} \\ \text{OPO\_Res\_Out\_Mean} \\ \text{OPO\_Den\_Out\_95P} \\ \text{OPO\_GDeg\_PO\_Mean} \\ \text{OPO\_GDeg\_Out\_Mean} \end{bmatrix}$$

Figures 1 and 2 illustrate the resulting instance space, with colours represent the feature values in the former, and the algorithm performance in the latter. The scale has been adjusted to the $[0, 1]$ range. To interpret these graphs, we describe the details of each selected feature:

Mean_Entropy_Attr is the average entropy of the attributes of the dataset, and corresponds to a measurement of the information contained by the attribute. For example, if an attribute is constant for all observations, then the entropy is zero. Thus, high values of entropy indicate highly unpredictable fluctuations. If the outliers are causing high values of entropy, then it is likely that these points are different from the rest, resulting in multiple outlier methods performing well on these datasets, as seen in Figure 2.

IQR_TO_SD_95 is the 95$^{\text{th}}$ percentile of the IQR to standard deviation ratio of each attribute of a dataset. It captures the presence of outliers by means of comparing two measures of spread, of which one is affected by outliers and another is robust. As this feature is first computed per attribute before taking the 95$^{\text{th}}$ percentile, data points that stand out in a single dimension give a small IQR to SD ratio for that dimension. Thus, if the 95$^{\text{th}}$ percentile of the IQR to SD ratio is small, then that

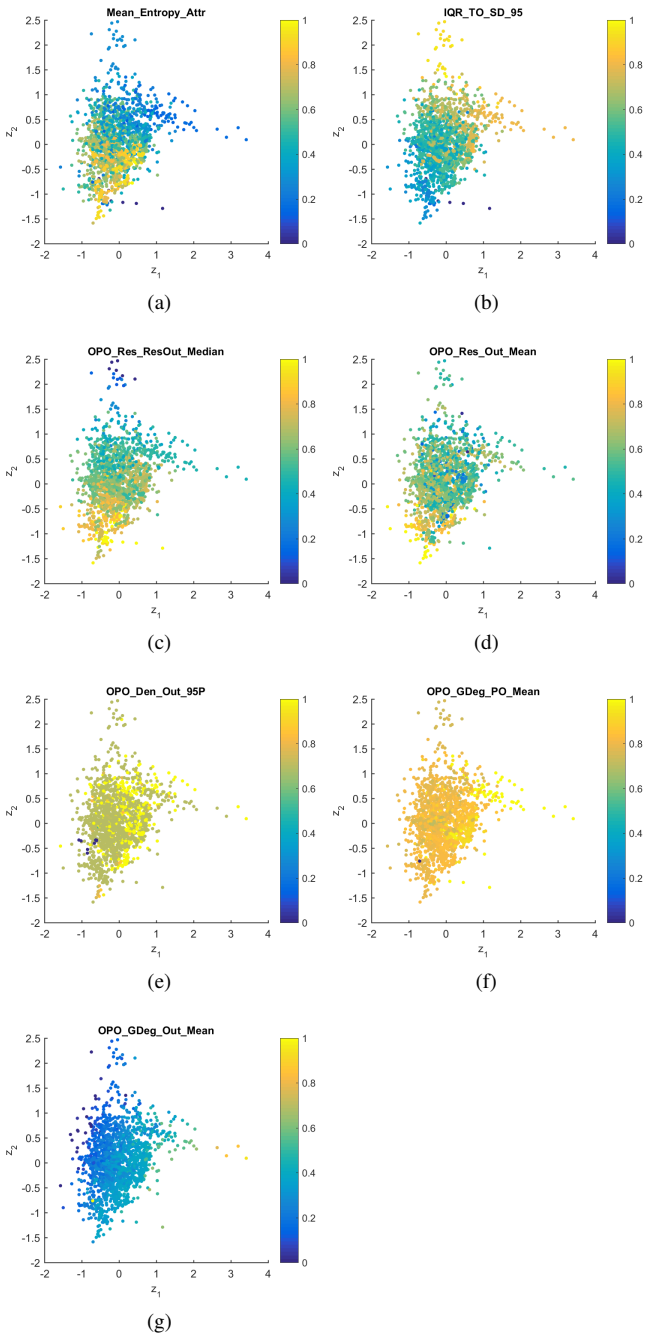


Figure 1: Distribution of normalised features in the projected instance space.

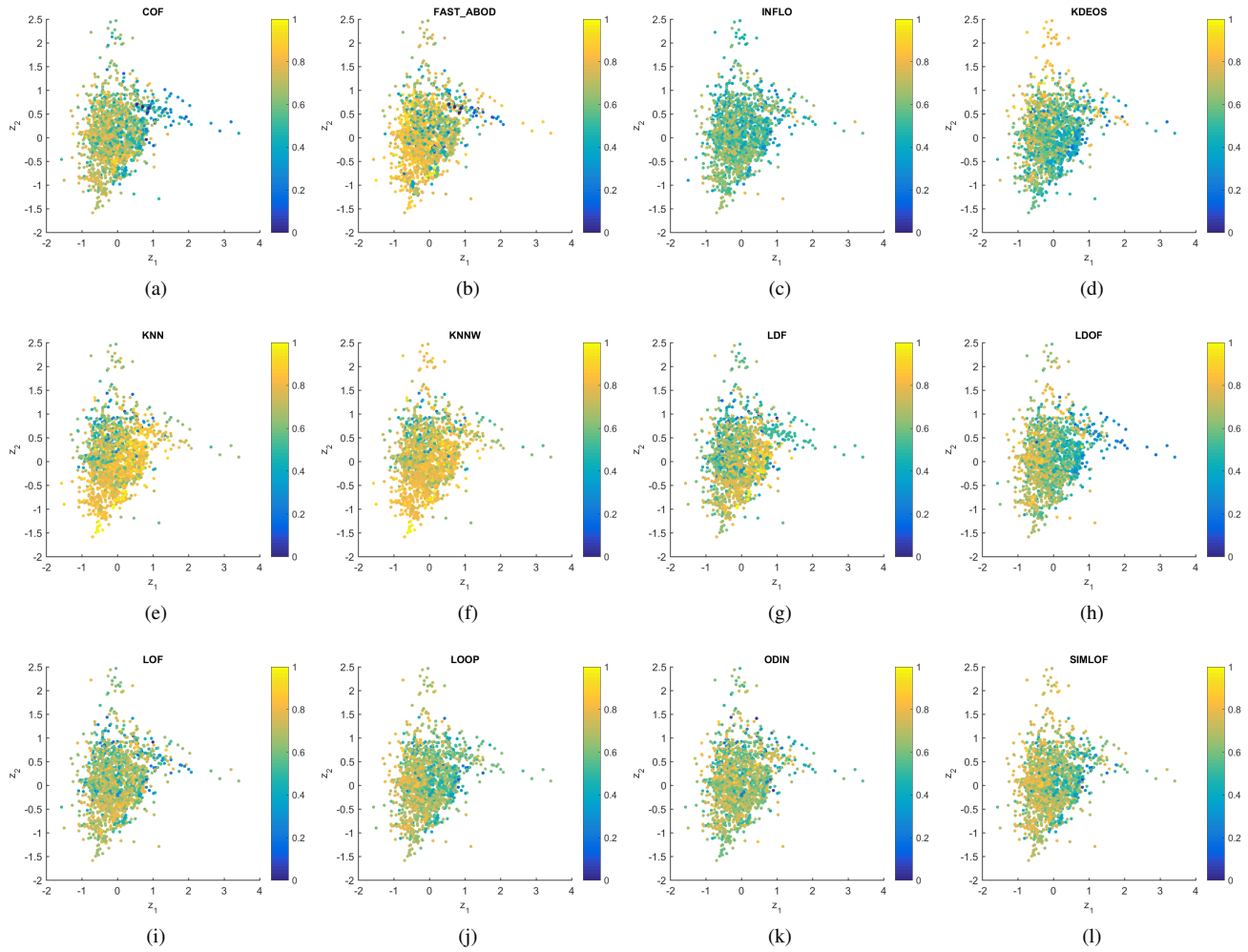


Figure 2: Distribution of AUC for each outlier detection method in the projected instance space.

implies there are data points that stand out in different dimensions. If these data points are indeed outliers, it should be easy for distance based methods to find them. We expect both KNN and KNNW to perform well on datasets with small values of this feature.

OPO_Res_ResOut_Median is the ratio between the median of residuals of proxy-outliers to the median of residuals of non-proxy-outliers. It is calculated by randomly choosing a subset of non-binary attributes from the dataset, s. Then, from each attribute a in s, we fit a linear model with a as the dependent variable and others as the independent variables. Next, for each model, we compute the residuals r. Define the residual based proxy-outliers ω as those with the top 3% absolute residual values. Finally, we compute $\kappa = \text{median}(r[\omega])/\text{median}(r[\bar{\omega}])$, where $\bar{\omega}$ denotes non-proxy-outliers. The average of κ across all models is the feature value.

OPO_Res_Out_Mean is the ratio between the mean of residuals of outliers to the mean of residuals of non outliers. It is calculated following a similar approach to that of OPO_Res_ResOut_Median with the exception that ω denotes actual outliers instead of proxy-outliers and the mean instead of the median is used. We expect KNN and KNNW to perform well on datasets with high values of OPO_Res_ResOut_Median and OPO_Res_Out_Mean.

OPO_Den_Out_95P is the ratio between the 95% of density estimates of non-outliers to 95% of density estimates of outliers. It is calculated by performing PCA on the dataset, omitting class information. Then, considering two consecutive components at a time and up to the 10^{th} component, we build nine 2-d spaces. For each one of these spaces, we compute kernel density estimates x.

Then, we compute $y_1 = x[\bar{\omega}]$ and $y_2 = x[\omega]$, where ω is the set of outliers and $\bar{\omega}$ denotes non-outliers. Let κ_1 be the 95$^{\text{th}}$ percentile of y_1 and κ_2 be the 95$^{\text{th}}$ percentile of y_2. Let $\kappa = \kappa_1/\kappa_2$, whose average across all 2-d is the feature value. If outliers reside in less dense regions, this gives rise to high values of this feature, enabling certain density based methods to perform well on these datasets. As such, we expect LDF to perform well on datasets with high values of this feature.

OPO_GDeg_PO_Mean is a ratio of mean degree of inner points to mean degree of non-inner points. It is calculated by generating a graph from the dataset using distances between points [33]. This graph would have v_1 and its nearest neighbour v_2 as vertices connected with an edge. Then, we compute the degree θ of each vertex v. Let ω be the set of inner points, which are defined as the vertices with the highest 3% degree values. Then the feature value is equal to mean$(\theta[\omega])$/mean$(\theta[\bar{\omega}])$ where ω denotes outliers and $\bar{\omega}$ denotes non-outliers.

OPO_GDeg_Out_Mean is similar to OPO_GDeg_PO_Mean with the exception of using outliers instead of inner points. These two features compute a ratio of vertex-degree between different groups: outliers and non-outliers, inner-points and non-inner points. As outliers are generally isolated, they have low vertex-degree values. Thus, these outliers make smaller angles with other data points, making it easier for angle based methods such as FAST ABOD to find them. Therefore we expect FAST ABOD to perform well on datasets with low values of OPO_GDeg_Out_Mean, even though we do not perform any angle based computation.

Combining all of this in Figures and 2, we see the performance of KDEOS tends to increase from the bottom to the top of the space. This is related to lower values of the ratio between the residuals from proxy-outliers to non-proxy-outliers (OPO_Res_ResOut_Median) and mean entropy of the attributes (Mean_Entropy_Attr). A low entropy suggests that an instance is predictable or less random and a smaller value of OPO_Res_ResOut_Median indicates that residuals of proxy-outliers are similar to those of non-proxy outliers. This signifies a unique strength of KDEOS as it performs well in an easy region in terms of entropy but in a difficult region in terms of residual ratios.

The performance of KNN and KNNW tend to increase from left to right of the space, while the opposite is true for FAST ABOD. Their performance is related to the values of the ratio between the interquartile range and the standard deviation of the features (IQR_TO_SD_95) taken at its 95%, and the mean graph degree of all vertices over the mean graph degree of outliers (OPO_GDeg_Out_Mean). Outliers which have high nearest neighbour distances generally have low graph degree, as their connections to other points are low in number, making OPO_GDeg_Out_Mean correlate with KNN and KNNW performances.

4.1 Footprint analysis of algorithm strengths and weaknesses We define a footprint as an area of the instance space where an algorithm is expected to perform well. To construct a footprint, we follow the approach first introduced in [34]: (a) we measure the length of the edges between all instances, and remove those edges with a length lower than a threshold, δ; (b) we calculate a Delaunay triangulation within the convex hull formed by the remaining edges; (c) we create a concave hull, by removing any triangle with edges larger than another threshold, Δ; (d) we calculate the density and purity of each triangle in the concave hull; and, (e) we discard any triangle that does not fulfil the density and purity thresholds. The values for parameters for the lower and upper distance thresholds, $\{\delta, \Delta\}$, are set to 1% and 25% of the maximum distance respectively. The density threshold, ρ, is set to 10, and the purity threshold, π, is set to 75%. We then remove any contradictions that could appear when two conclusions could be drawn from the same section of the instance space due to overlapping footprints, e.g., when comparing two algorithms. This is achieved by comparing the area lost by the overlapping footprints when the contradicting sections are removed. The algorithm that would loose the largest area in a contradiction gets to keep it, as long as it maintains the density and purity thresholds.

Table 3 presents the results from the analysis. The best algorithm is the one such that AUC is the highest for the given instance. The results are a percentage of the area (6.6703) and density (305.6825) of the convex hull that encloses all instances. The table demonstrate that some algorithms, such as INFLO, LOOP and ODIN have good performance in few, scattered instances; hence, we are unable to find an area that fulfils the density and purity requirements. On the other hand, FAST ABOD, KDEOS and KNN have the largest footprints. Combining with Figure 2 we can see that FAST ABOD tends to dominates the lower left corner, while KNN tends to dominate the lower right corner. KDEOS performs the best in the few instances at the top of the instance space.

4.2 Automated algorithm selection via instance space Visualising regions of strength for some outlier detection methods is a key advantage of the instance space. In addition, we can also use the instance space for automatic algorithm selection for new instances. By computing the instance space coordinates of a new instance using its meta-features, we can project it into the instance space and see in which algorithm's footprint it lies, and thus recommend outlier detection methods accordingly.

This process can be automated if we learn the best par-

Table 3: Footprint analysis of the algorithms. α_N is the area, d_N the density and ρ the purity. The footprint areas (and their density and purity) are shown where algorithm performance is good and best.

	$AUC > 0.8$			Best algorithm		
	a_N	d_N	ρ	a_N	d_N	ρ
COF	1.9%	253.0%	3.1%	0.5%	582.7%	7.7%
FAST ABOD	3.4%	144.7%	9.1%	14.2%	142.5%	20.4%
INFLO	4.0%	86.8%	7.1%	0.0%	0.0%	0.0%
KDEOS	15.5%	35.7%	8.0%	9.0%	23.0%	14.3%
KNN	2.3%	202.4%	3.1%	8.5%	169.3%	12.6%
KNNW	2.4%	178.5%	8.1%	1.6%	148.1%	18.4%
LDF	0.9%	729.4%	3.7%	2.9%	356.2%	14.9%
LDOF	5.1%	121.5%	7.9%	1.1%	69.4%	12.5%
LOF	0.8%	631.4%	6.1%	0.4%	303.7%	21.7%
LOOP	5.7%	94.6%	7.3%	0.0%	0.0%	0.0%
ODIN	1.7%	177.9%	5.0%	0.0%	0.0%	0.0%
SIMLOF	4.2%	117.8%	9.9%	0.1%	602.5%	12.5%

Table 4: Accuracy of SVM prediction of good performance based on instance space location of test sets.

Method	Prediction accuracy	Default accuracy
FAST ABOD	66%	62%
KDEOS	92%	92%
KNN	65%	54%
KNNW	58%	57%

titioning of the instance space to expose regions of strength for each method. We use support vector machines (SVM) for this partitioning. Of the 12 outlier methods we consider only FAST ABOD, KDEOS, KNN and KNNW, as these methods have larger footprints that span a contiguous region of the instance space. For these outlier methods, we use our definition of *good* performance as the output and the instance space coordinates as the input for the SVM. We train 4 SVMs, one for each of these selected outlier methods. The prediction accuracies using 5-fold cross validation, along with the percentage of instances in the majority class, are given in Table 4, showing better than default accuracy for all methods except KDEOS.

Figure 3 shows the dominant regions of algorithms FAST ABOD, KNN, KNNW and KDEOS. We can see that FAST ABOD is stronger on the left side of the instance space with a few sparse instances lighting up on the extreme right, whereas KDEOS is stronger on the top region of the instance space. KNN and KNNW have overlapping regions of strength with KNNW sharing some instances with KDEOS at the top. However, KDEOS is the stronger algorithm in the top region of the instance space.

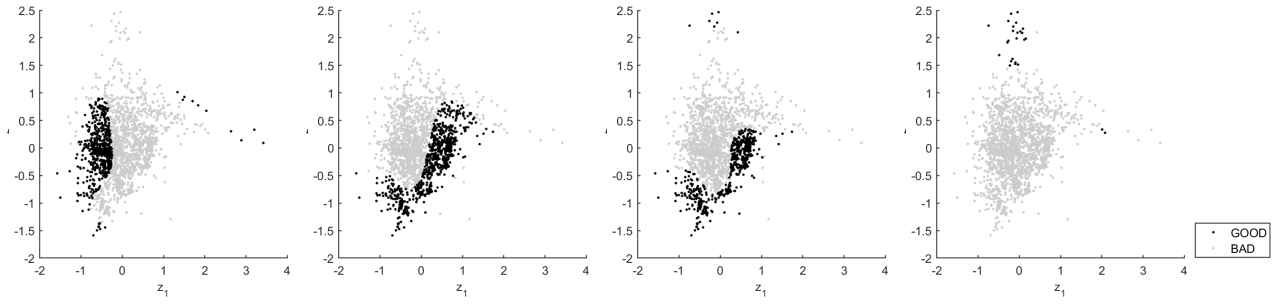
We combine these SVM predictions of regions of strength to obtain a final partitioning of the instance space. Regions of overlap mean that some instances have multiple preferred outlier methods. For such instances, we break ties using the prediction probability of the SVM and choose the method with the highest probability. The resulting partitioned instance space is shown in Figure 4, where FAST ABOD occupies the left, and KNN occupies the right of the instance space with KDEOS occupying the top region. The middle of the instance space is not occupied by any outlier method.

This analysis highlights an opportunity for new outlier detection methods to be developed for this part of the instance space with no genuine method superiority. Also, the statistical analysis conducted by Campos and co-authors ([4], Table 3) shows that KDEOS is inferior to most other methods. However, as seen from Figure 4 KDEOS has a niche in the instance space where it clearly outperforms other methods. This insight, which was missed by the standard statistical analysis, highlights another contribution of the instance space analysis.

5 Conclusion

We have investigated the algorithm selection problem for unsupervised outlier detection. Given measurable features of a dataset, we find the most suitable outlier method with reasonable accuracy. This is important because each method has its strengths and weaknesses and no single method outperforms all others for all instances. We have explored the strengths and weaknesses of outlier methods by analysing their footprints in the constructed instance space. Moreover, we have identified different regions of the instance space that reveal the relative strengths of different outlier detection methods. Our work clearly demonstrates for example that KDEOS, which gives poor performance on average, has a region of strength in the instance space where no other algorithm excels.

In addition to these contributions, we hope to have laid some important foundations for future research into new and



(a) (b) (c) (d)

Figure 3: Regions of strength for (a) FAST ABOD, (b) KNN, (c) KNNW and (d) KDEOS.

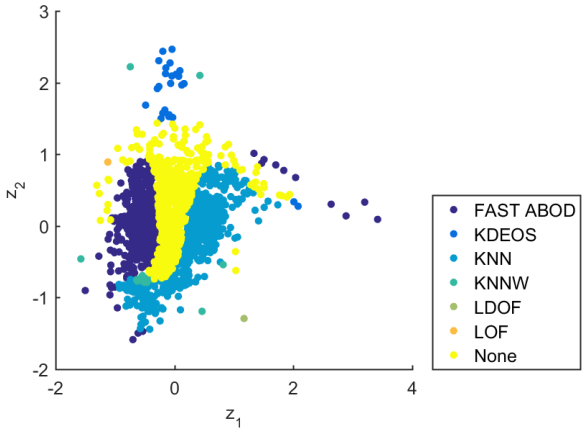


Figure 4: A partition of the instance space showing recommended outlier detection methods.

improved outlier detection methods, in the following ways: (a) rigorous evaluation of new methods using the comprehensive corpus of over 12000 datasets with diverse characteristics we have made available; (b) using the instance space, the strengths and weaknesses of new outlier methods can be identified, and their uniqueness compared to existing methods; (c) using the R package *outselect* the outlier methods suitable for a new dataset can be identified, and the new instance can be plotted on the instance space. Equally valuable, the instance space analysis can also reveal if a new outlier method is similar to existing outlier methods, or offers a unique contribution to the available repertoire of techniques.

As a word of caution, we note that our current instance space is computed using our collection of datasets, outlier methods and features. Thus, we do not make claim to have constructed the definitive instance space for all unsupervised outlier detection methods. Hence, the selected features for the instance space may change with the expansion of the corpus of datasets and outlier methods. Future research paths include the expansion of the instance space by generating new and diverse instances to fill the space and considering other classes of outlier detection methods, such as subspace, clustering and PCA approaches. In addition we have transformed non-numeric data to numeric in this study. Further avenues of research include incorporating datasets with non-numeric attributes in the instance space. To aid this expansion and future research, we make all of our data and implementation scripts available on our website.

Broadening the scope of this work, we have been adapting the instance space methodology to other problems besides outlier detection. For example, machine learning [24] and time series forecasting [35]. Part of this larger project is to build freely accessible web-tools that carry out the instance space analysis automatically, including testing of algorithms on new instances. Such tools will be available at `matilda.unimelb.edu.au` in the near future.

Acknowledgements

Funding was provided by the Australian Research Council through grants FL140100012 and LP160101885. This research was also supported by the Monash eResearch Centre and eSolutions-Research Support Services through the MonARCH HPC Cluster.

References

[1] 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.

[2] D. H. Wolpert, W. G. Macready *et al.*, "No free lunch theorems for search," Technical Report SFI-TR-95-02-010, Santa Fe Institute, Tech. Rep., 1995.

[3] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[4] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On

the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 891–927, 2016.

[5] K. Smith-Miles, D. Baatar, B. Wreford, and R. Lewis, "Towards objective measures of algorithm performance across instance space," *Comput. Oper. Res.*, vol. 45, pp. 12–24, 2014.

[6] K. Smith-Miles and S. Bowly, "Generating new test instances by evolving in instance space," *Computers & Operations Research*, vol. 63, pp. 102–113, 2015.

[7] K. Smith-Miles and T. T. Tan, "Measuring algorithm footprints in instance space," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, 2012, pp. 1–8.

[8] S. Kandanaarachchi, M. Munoz Acosta, K. Smith-Miles, and R. Hyndman, "Datasets for outlier detection," Feb 2019. [Online]. Available: https://monash.figshare.com/articles/Datasets_12338_zip/7705127/4

[9] S. Kandanaarachchi, *outselect: Algorithm selection for unsupervised outlier detection*, 2018, r package version 0.0.0.9000. [Online]. Available: https://github.com/sevvandi/outselect

[10] E. Achtert, H.-P. Kriegel, and A. Zimek, "Elki: a software system for evaluation of subspace clustering algorithms," in *International Conference on Scientific and Statistical Database Management*. Springer, 2008, pp. 580–585.

[11] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 427–438.

[12] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2002, pp. 15–27.

[13] V. Hautamaki, I. Karkkainen, and P. Franti, "Outlier detection using k-nearest neighbour graph," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3. IEEE, 2004, pp. 430–433.

[14] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM sigmod record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.

[15] E. Schubert, A. Zimek, and H.-P. Kriegel, "Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 190–237, 2014.

[16] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2002, pp. 535–548.

[17] W. Jin, A. K. Tung, J. Han, and W. Wang, "Ranking outliers using symmetric neighborhood relationship," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2006, pp. 577–593.

[18] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Loop: local outlier probabilities," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 1649–1652.

[19] K. Zhang, M. Hutter, and H. Jin, "A new local distance-based outlier detection approach for scattered real-world data," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2009, pp. 813–822.

[20] L. J. Latecki, A. Lazarevic, and D. Pokrajac, "Outlier detection with kernel density functions," in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer, 2007, pp. 61–75.

[21] E. Schubert, A. Zimek, and H.-P. Kriegel, "Generalized outlier detection with flexible kernel density estimates," in *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 2014, pp. 542–550.

[22] H.-P. Kriegel, A. Zimek *et al.*, "Angle-based outlier detection in high-dimensional data," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 444–452.

[23] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PloS one*, vol. 11, no. 4, p. e0152173, 2016.

[24] M. A. Muñoz, L. Villanova, D. Baatar, and K. Smith-Miles, "Instance spaces for machine learning classification," *Machine Learning*, vol. 107, no. 1, pp. 109–147, 2018.

[25] N. Craswell, "Precision at n," in *Encyclopedia of database systems*. Springer, 2009, pp. 2127–2128.

[26] E. Zhang and Y. Zhang, "Average precision," in *Encyclopedia of database systems*. Springer, 2009, pp. 192–193.

[27] P. Talagala, R. Hyndman, K. Smith-Miles, S. Kandanaarachchi, M. Muñoz *et al.*, "Anomaly detection in streaming nonstationary temporal data," *Journal of Compuational and Graphical Statistics*, 2019, accepted.

[28] C. Leigh, O. Alsibai, R. J. Hyndman, S. Kandanaarachchi, O. C. King, J. M. McGree, C. Neelamraju, J. Strauss, P. D. Talagala, R. D. Turner *et al.*, "A framework for automated anomaly detection in high frequency water-quality data from in situ sensors," *Science of The Total Environment*, 2019.

[29] B. Pfahringer, H. Bensusan, and C. G. Giraud-Carrier, "Meta-learning by landmarking various learning algorithms." in *ICML*, 2000, pp. 743–750.

[30] Y. Peng, P. A. Flach, C. Soares, and P. Brazdil, "Improved dataset characterisation for meta-learning," in *International Conference on Discovery Science*. Springer, 2002, pp. 141–152.

[31] K. A. Smith-Miles, "Cross-disciplinary perspectives on meta-learning for algorithm selection," *ACM Computing Surveys (CSUR)*, vol. 41, no. 1, p. 6, 2009.

[32] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[33] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal, Complex Systems*, vol. 1695, no. 5, pp. 1–9, 2006.

[34] M. Muñoz and K. Smith-Miles, "Performance analysis of continuous black-box optimization algorithms via footprints in instance space," *Evol. Comput.*, vol. 25, no. 4, pp. 529–554, 2017.

[35] Y. Kang, R. Hyndman, and K. Smith-Miles, "Visualising forecasting algorithm performance using time series instance spaces," *Int. J. Forecast*, vol. 33, no. 2, pp. 345–358, 2017.