北京航空航天大學 BEIHANG UNIVERSITY | 经济管理学院 SCHOOL OF ECONOMICS AND MANAGEMENT

Instance spaces for machine learning classification

ZhangYi

BUAA

Contents

1.Introduction and Methodological framework

- How do we objectively assess whether one classifier is more powerful than another?
- Common practice is to test a classifier on a well-studied collection of classification datasets, typically from the UCI repository (Wagstaff 2012).

Disadvantages

- The UCI repository is a very limited sample of problems,not a representative sample.
- Algorithms that work well on UCI datasets might not work well on new or less popular problem instance classes.

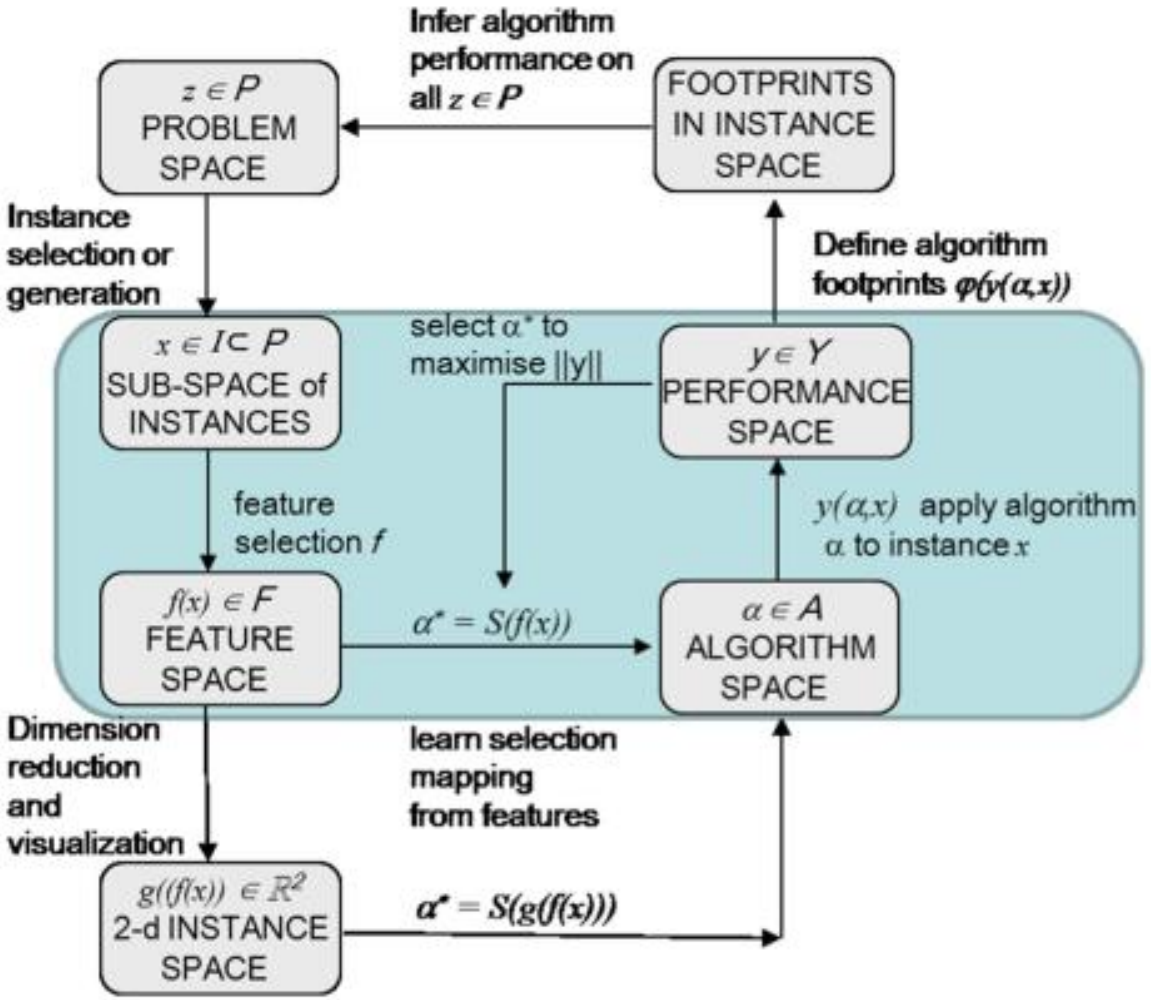
1.Introduction and Methodological framework

- Rice (1976) proposed a powerful framework. The framework relies on measurable features of the problem instances, correlated with instance difficulty, to predict which algorithm is likely to perform best. The idea uses machine learning methods to learn the model, developed into the well-studied field of meta-learning.

- SmithMiles and co-authors have developed a methodology over recent years through a series of papers that extend Rice's framework to provide greater insights into algorithm strengths and weaknesses. The performance of algorithms can be visualized.

1.Introduction and Methodological framework

Methodological framework



1.Introduction and Methodological framework

- For a given problem instance $x \in I$ and a given algorithm $\alpha \in A$, a feature vector $f(x) \in F$ and algorithm performance metric $y(\alpha, x) \in Y$ are measured. By repeating the process for all instances in I and all algorithms in A, the meta-data {I, F, A, Y}.

- In our extended framework, the meta-data is used to learn the mapping $g(f(x), y(\alpha, x))$, which projects the instance x from a high-dimensional feature space to a 2-dimensional space. A new approach to achieving an optimal 2-D projection has been proposed for this paper.

1.Introduction and Methodological framework

In summary, the proposed methodology requires:

1. Construction of the meta-data, including a set of candidate features (Sect. 3)
2. Selection of a subset of candidate features (Sect. 4)
3. Justification of selected features using performance prediction accuracy (Sect. 5)
4. Creation of a 2-D instance space for visualization of instances and their properties (Sect. 6)
5. Objective measurement of algorithmic power (Sect. 7)
6. Generation of new test instances to fill the instance space (Sect. 8)

2.Meta-data and Feature selection

meta-learning

- The learning task is to infer, from the training set, an function relating the attributes to the class labels. The inferred function is then used to predict the labels in the test set. The performance of the learning algorithm is measured by a metric comparing true and predicted labels.

- The focus in meta-learning is to study how measurable features of the problem instances (X^i, c^i) affect a given learning algorithm's performance metric. The four elements compose the meta-data {I , F , A, Y }. We will briefly describe these elements of the meta-data used in this study below.

2.Meta-data and Feature selection

Describe these elements of the meta-data

1. Problem instances I

① Consists of a collection of (X j , c j) pairs；

② Datasets： (UCI) repository (Lichman 2013) and the Knowledge Extraction Evolutionary Learning (KEEL) repository (Alcalá et al. 2010),a few datasets from the Data Complexity library (DCol—http://dcol.sourceforge.net/)

③ A total of 235 problem instances comprising 210 UCI instances, 19 KEEL instances and 6 DCol instances.

④ The selected 235 problem instances have up to 11,055 observations and up to 1558 attributes.

2.Meta-data and Feature selection

2. Algorithms A

The algorithms are Naive Bayes (NB), Linear Discriminant (LDA), Quadratic Discriminant (QDA), Classification and Regression Trees (CART), J48 decision tree (J48), k-Nearest Neighbor (KNN), Support Vector Machines with linear, polynomial and radial basis kernels (L-SVM, poly-SVM, and RB-SVM respectively), and random forests (RF).

3. Performance metric Y

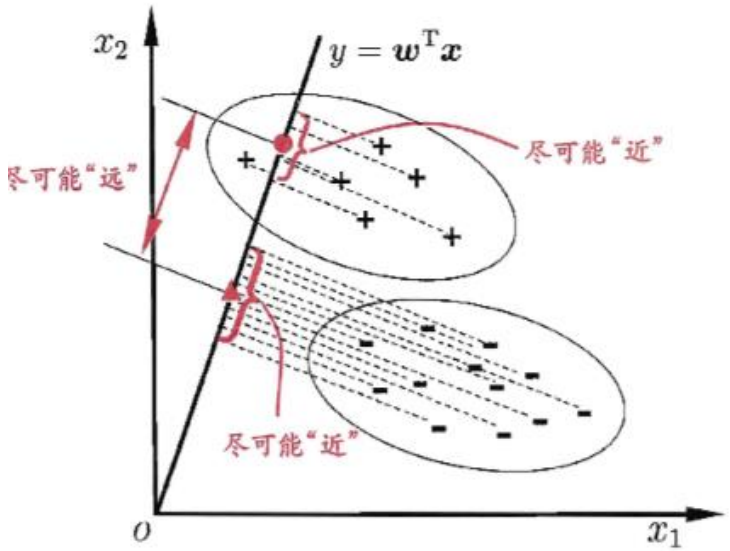
we consider measures of prediction accuracy/error which evaluate how well or poorly the labels are classified.

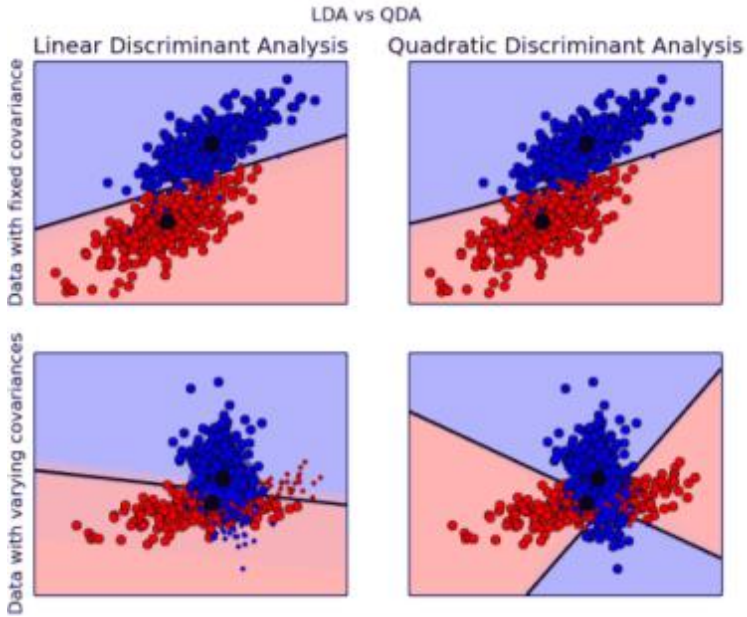
		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

Measure	Formula	
	Binary	Multi-class
Error Rate (ER)	$\frac{fp+fn}{n}$	$\sum_{k=1}^{K} ER_k \cdot w_k$
Precision	$\frac{tp}{tp+fp}$	$\sum_{k=1}^{K} Precision_k \cdot w_k$
Recall	$\frac{tp}{(tp+fn)}$	$\sum_{k=1}^{K} Recall_k \cdot w_k$
F1-measure (Fm)	$(\beta^2+1) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$	$\sum_{k=1}^{K} Fm_k \cdot w_k$

2.Meta-data and Feature selection

Linear Discriminant (LDA)&Quadratic Discriminant (QDA)





2.Meta-data and Feature selection

4. Features F

Classify the meta-features into eight different groups:

(i) simple

(ii) statistical

(iii) information theoretic

(iv)landmarking

(v) model-based

(vi) concept characterisation

(vii) complexity

(viii) itemset-based meta-features

In this study we have generated a set of 509 features derived from the eight types of features. Our goal is to represent the instances in a feature space. The process of selecting relevant features from this candidate set of 509 features will be discussed in the following section.

2.Meta-data and Feature selection

To identify relevant features, the procedure is as follows:

1.**Identify** broad characteristics that make a classification task harder (classification challenges);
2.**Alter** a problem instance to deliberately vary the hardness of the classification task (instance alteration);
3.**Calculate** all 509 features on both original and altered problems;
4.**Compare** features values of original and altered problems (statistical test);
5.**Identify** the set of relevant features as those most responsive to the challenges;
6.**Evaluate** the adequacy of the relevant features via performance prediction.

2.Meta-data and Feature selection

1. Based on previous investigations of classification algorithms ,we identify 12 challenging circumstances. They are:

– Non-normality within classes
– Unequal covariances within classes
– Redundant attributes
– Type of attributes
– Unbalanced classes
– Constant attribute within classes
– (Nearly) Linearly dependent attributes
– Non-linearly separable classes
– Missing values
– Data scaling
– Redundant instances
– Lack of information

2.Meta-data and Feature selection

2. Instance alteration Based on the above challenges, we alter a problem instance to change the difficulty of the classification task. For each challenge we obtain two problem instances that we want to compare: they are the original and altered datasets.

3. Each challenge is treated in a similar manner. Table 2 reports the identified challenges and highlights which ones are likely to be relevant for the investigated algorithms.

2.Meta-data and Feature selection

Table 2 Classification challenges specific to the investigated algorithms

Challenge	Challenged algorithm						
	NB	LDA	QDA	CART-J48	KNN	SVM	RF
Non-normality within classes		✓	✓				
Unequal covariance within classes		✓					
Redundant attributes					✓		
Type of attributes		✓	✓		✓		
Unbalanced classes	✓				✓		✓
Constant attribute within classes		✓	✓				
(Nearly) Linearly dependent attributes	✓	✓					
Non-linearly separable classes		✓					
Missing values		✓	✓		✓		
Data scaling					✓	✓	
Redundant instances					✓		
Lack of information	✓	✓	✓	✓	✓	✓	✓

Algorithms for which a specific challenge is expected to be relevant, based on their model assumptions, are highlighted with the symbol ✓

2.Meta-data and Feature selection

4. Statistical test original and altered datasets are compared in terms of their values of the 509 candidate features.

5. We identify features. The final set of relevant features is as follows:

1. $H(\mathbf{X})'_{\max}$—maximum normalized entropy of the attributes
2. H'_c—normalized entropy of class attribute
3. $\overline{M}_{CX}$—mean mutual information of attributes and class
4. DN_{ER}—error rate of the decision node
5. $SD(v)$—standard deviation of the weighted distance
6. $F3$—maximum feature efficiency
7. $F4$—collective feature efficiency
8. $L2$—training error of linear classifier
9. $N1$—fraction of points on the class boundary
10. $N4$—nonlinearity of nearest neighbor classifier

2.Meta-data and Feature selection

6. Assesse adequacy of the feature set via performance prediction. We adopt an approach based on model fitting and evaluation of model accuracy.

- We use a Support Vector Machine (SVM) model with Gaussian Radial Basis Function (RBF) kernel. The type of SVM used is -regression and C-classification .

- Table 5 reports values of SVM parameters and cross-validated error for both regression and classification studies.

2.Meta-data and Feature selection

Table 5 Parameters values and performance of the SVM models approximating the functional relationship between selected features and (i) algorithm performance (regression case), (ii) problem difficulty (classification case)

Algorithm	ϵ-regression					C-classification		
	γ	C	ϵ	cv-MSE	R^2	γ	C	cv-ER
NB	0.104	9	0.14	0.006	0.91	0.102	2	0.157
LDA	0.081	4	0.04	0.029	0.72	0.102	1	0.161
QDA	0.117	8	0.21	0.023	0.93	0.093	3	0.145
CART	0.095	2	0.17	0.004	0.91	0.105	1	0.099
J48	0.090	4	0.05	0.003	0.94	0.099	2	0.106
KNN	0.098	3	0.00	0.002	0.97	0.092	6	0.073
L-SVM	0.106	2	0.07	0.006	0.85	0.095	1	0.086
Poly-SVM	0.098	3	0.02	0.006	0.89	0.089	5	0.127
RBF-SVM	0.129	2	0.13	0.005	0.90	0.082	7	0.085
RF	0.099	3	0.37	0.027	0.63	0.092	1	0.132

- The small values of the cross-validated errors and the large R^2 values indicate that the selected features are adequate to accurately predict algorithm performance and problem difficulty.

3.Creating an instance space

The final aim is to expose strengths and weaknesses of classification algorithms and provide an explanation.

A critical step is the visualization of the instances, their features and algorithm performance in the instance space.

The instance space generation is an iterative process that might require multiple adjustments to identify an optimal subset of features and a suitable set of instances.

3.Creating an instance space

The steps that we have implemented to generate the instance space are:

1.Select a set of relevant features and evaluate their adequacy;

2.Generate an instance space and evaluate the adequacy of the instances;

3.If the instance space is inadequate, artificially generate new instances and return to Step 1.

3.Creating an instance space

How can we choose to project the instances from a high-dimensional feature vector to a 2-D instance space ?

- We used Principal Component Analysis to project graph features into a 2-dimensional space. However, the PCA model was somewhat unsatisfactory to predict performance, since PCA is only concerned with maximizing variance explained in the features with no regard for projecting to show trends in difficulty.

- We reformulate the dimensionality reduction problem to consider an optimal projection for our purpose below.

3.Creating an instance space

Given the feature data matrix $F = [f_1 f_2 \cdots f_n] \in R^{m \times n}$ and algorithm performance vector **y**$\in R^n$, where **m** is the number of features and **n** is the number of problem instances, we achieve an ideal projection of the instances if we can find $A_r \in R^{d \times m}$, $B_r \in R^{m \times d}$ and $C_r \in R^d$ which minimizes the approximation error. Thus, we have the following optimization problem:

$$\min \| F - B_r Z \|^2_F + \| y^T - c^T Z \|^2_F$$

$$s.t Z = A_r F$$

(D) $$A_r \in R^{d \times m}$$

$$B_r \in R^{m \times d}$$

$$C_r \in R^d$$

In Appendix A we prove the existence of a global optimum for D, and that such optimum has infinitely many solutions.

3.Creating an instance space

- The results from Appendix A also hold for a matrix of Y.

Let $Y \in R^{10\times235}$ be a matrix whose rows correspond to root-squared error rate of the ten algorithms in Table 5and its columns correspond to the 235 UCI/KEEL/DCol instances.

- The chosen transformation of instances from the 10-D feature space to the 2-D instance space is:

$$\mathbf{Z} = \begin{bmatrix} 0.070 & 0.180 \\ 0.094 & 0.618 \\ -0.277 & -0.052 \\ 0.114 & 0.192 \\ 0.045 & -0.100 \\ -0.128 & 0.151 \\ -0.045 & 0.077 \\ 0.184 & 0.017 \\ 0.449 & 0.223 \\ 0.132 & -0.112 \end{bmatrix}^{\top} \begin{bmatrix} H(\mathbf{X})'_{\max} \\ H'_c \\ \overline{M}_{CX} \\ DN_{ER} \\ SD(v) \\ F3 \\ F4 \\ L2 \\ N1 \\ N4 \end{bmatrix}$$

3.Creating an instance space

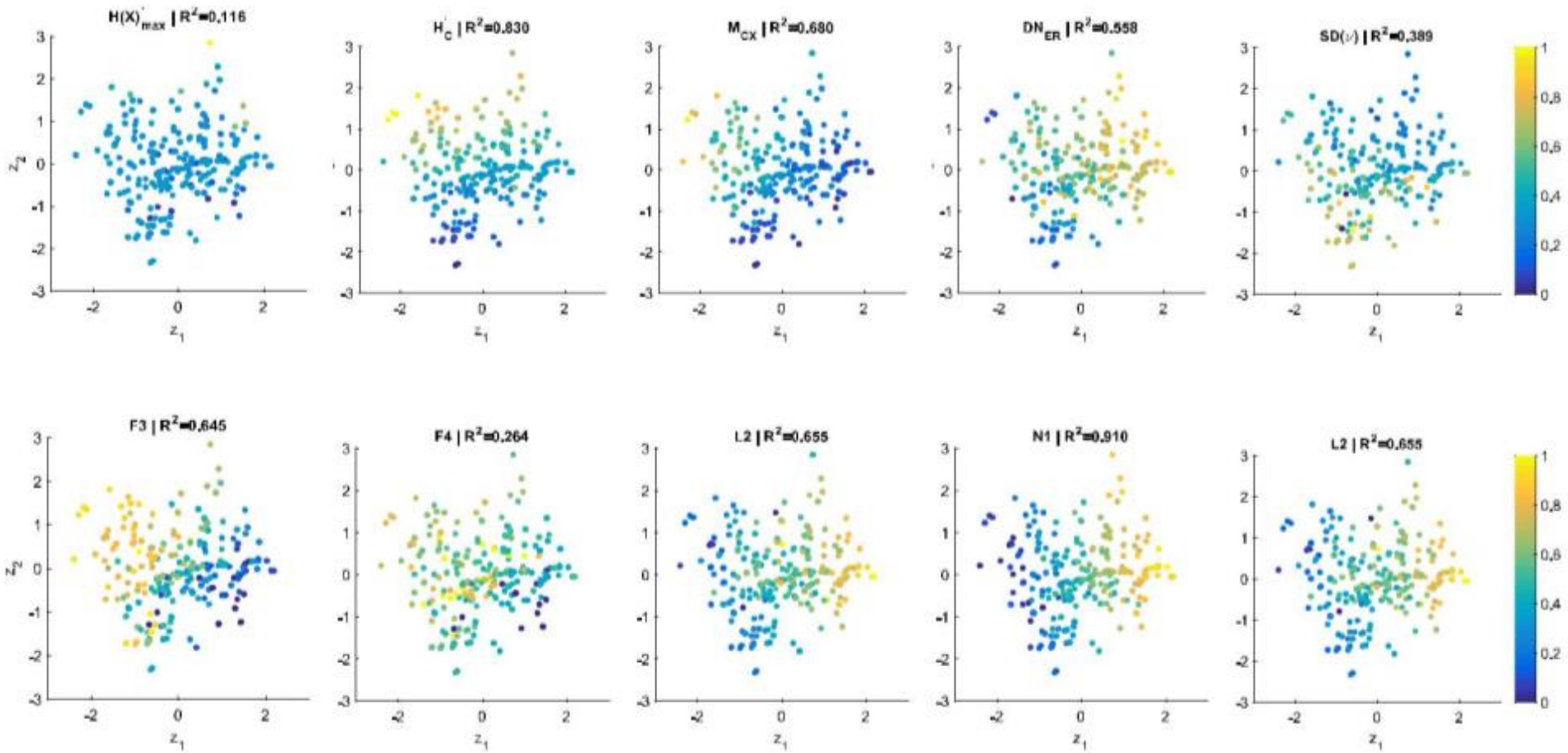


Fig. 2 Distribution of normalized features on the projected instance space

- Figure 2 enables us to visualize the instances described by their features selected in Sect. 4. The best fit is obtained for N1 R^2 = 0.910 ,and the worst fit for H (X)max R^2= 0.116 .

3.Creating an instance space

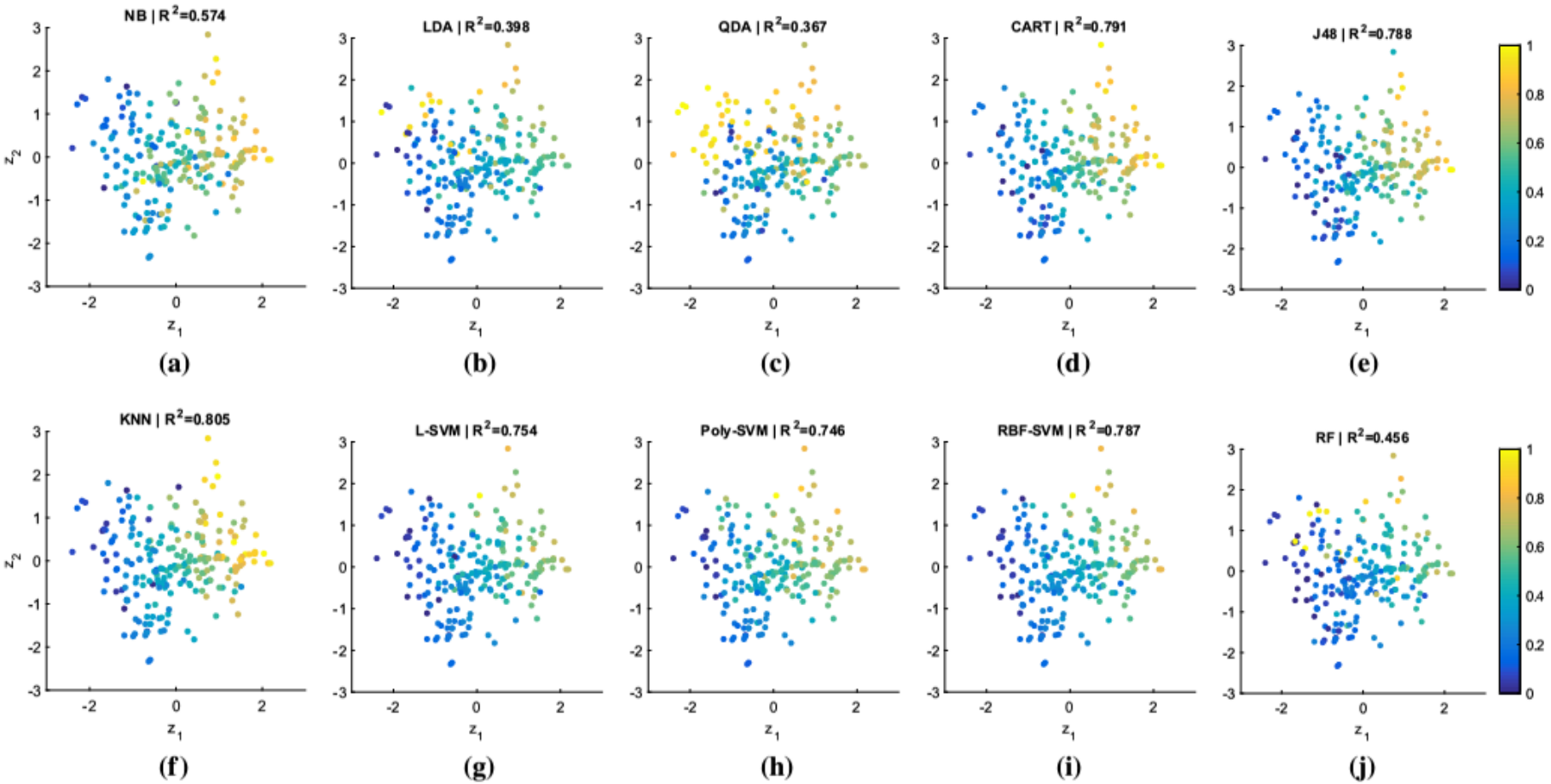


Fig.3 Normalized error rate of each classification algorithm on the projected instance space

- Fig. 3 shows the Error Rate (E R) of each algorithm in Sect. 3.2 distributed across the instance space. the best fit is obtained for KNN R^2= 0.805 , and the worst fit for QDA R^2= 0.367 .

3.Creating an instance space

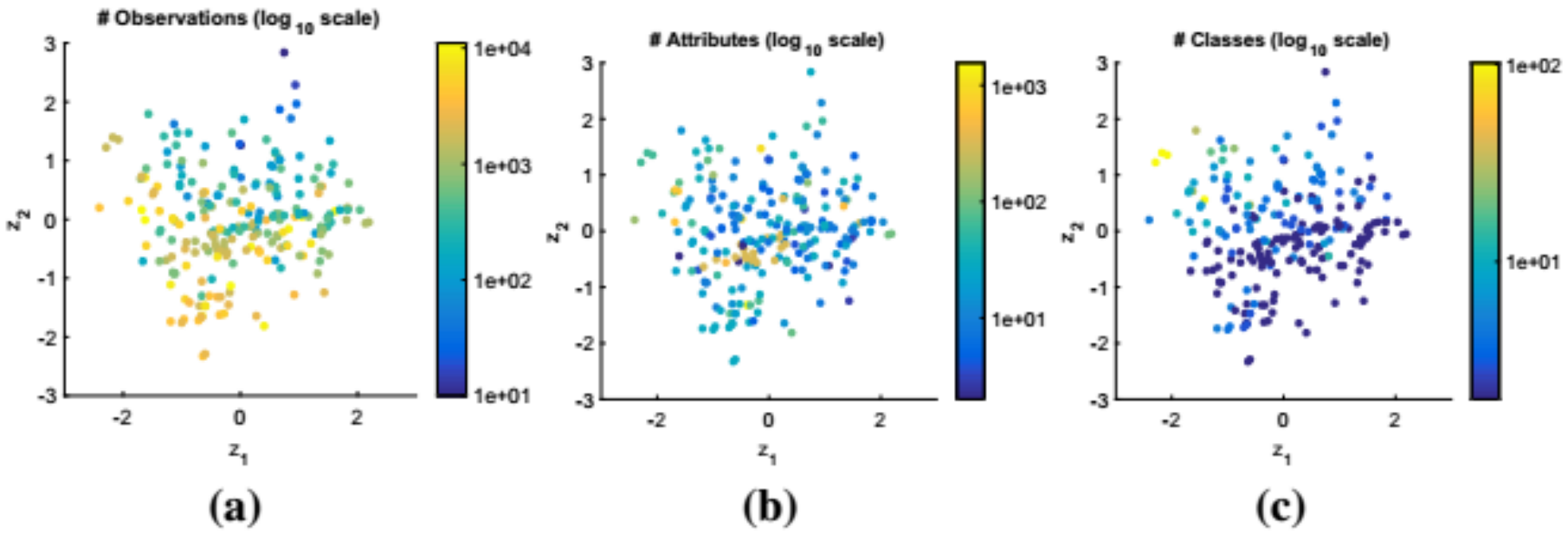


Fig. 4 Sizes of the instances in terms of the number of observations (p), attributes (q), and classes (K). All have been log10-scaled.

- Figure 4 illustrates the size of the instances by the number of (a) observations, (b) attributes, and (c) classes.
- For our selected 235 instances, the number of observations per instance increases from top to bottom, while the number of classes from right to left. There is no trend emerging from the number of attributes

4.Objective assessment of algorithmic power

Our method for objective assessment of algorithmic power is based on the accurate estimation and characterization of each algorithm's footprint.

$$h(ER) = \begin{cases} 0 \; if ER \leq 0.2 \\ 1 \; if ER > 0.2 \end{cases} \qquad (1)$$

- It takes on labels {0, 1} (corresponding to easy and hard instances respectively) and gives rise to a binary classification problem.
- An instance is also defined as β -hard with β = 0.5 if 50% of the algorithms have an error rate higher than 20%.

4.Objective assessment of algorithmic power

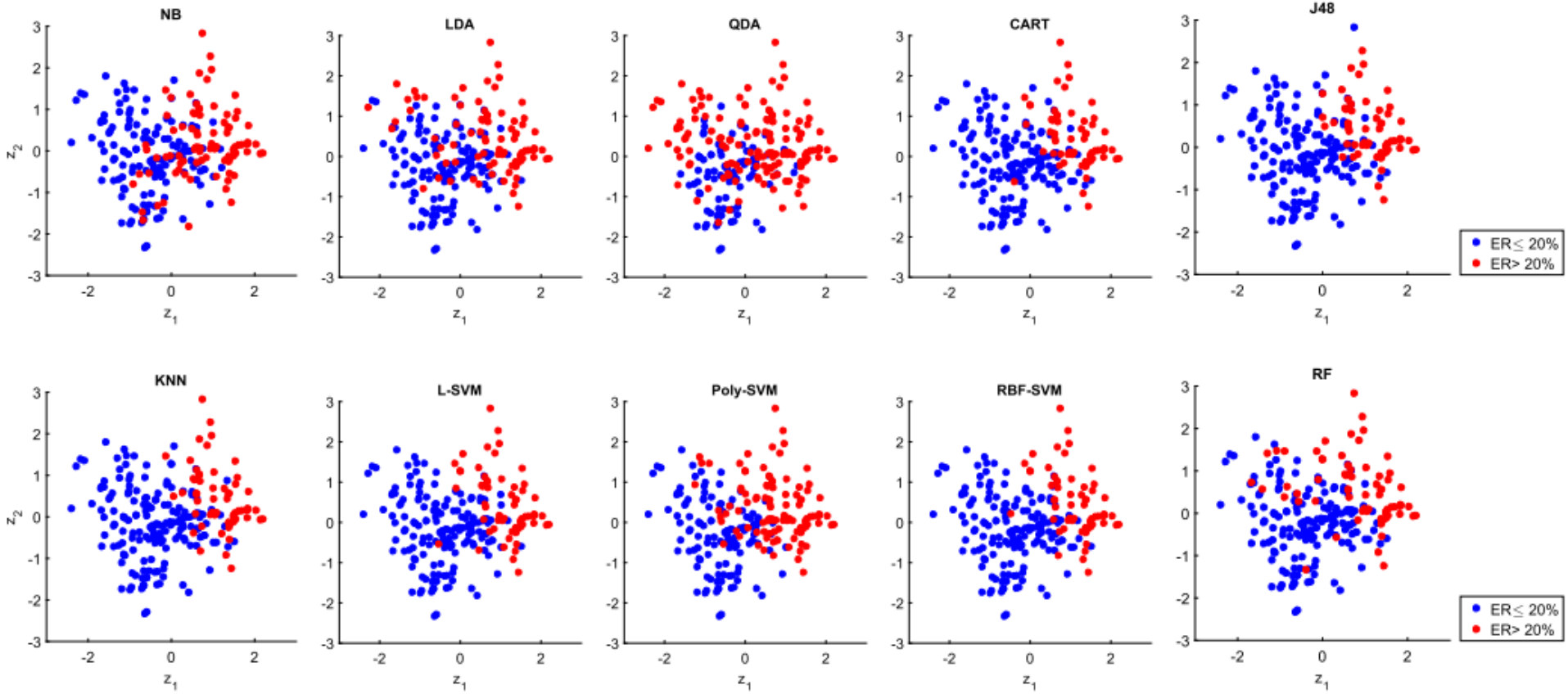


Fig. 5 Footprints of the algorithms in the instance space

- Fig. 5 shows QDA is the weakest algorithm i, while J48 could be considered the strongest. In fact, over half of the instance space is considered to have β -easy instances, while β -hard instances occupy only 20% approximately, for β = 0.5.

4.Objective assessment of algorithmic power

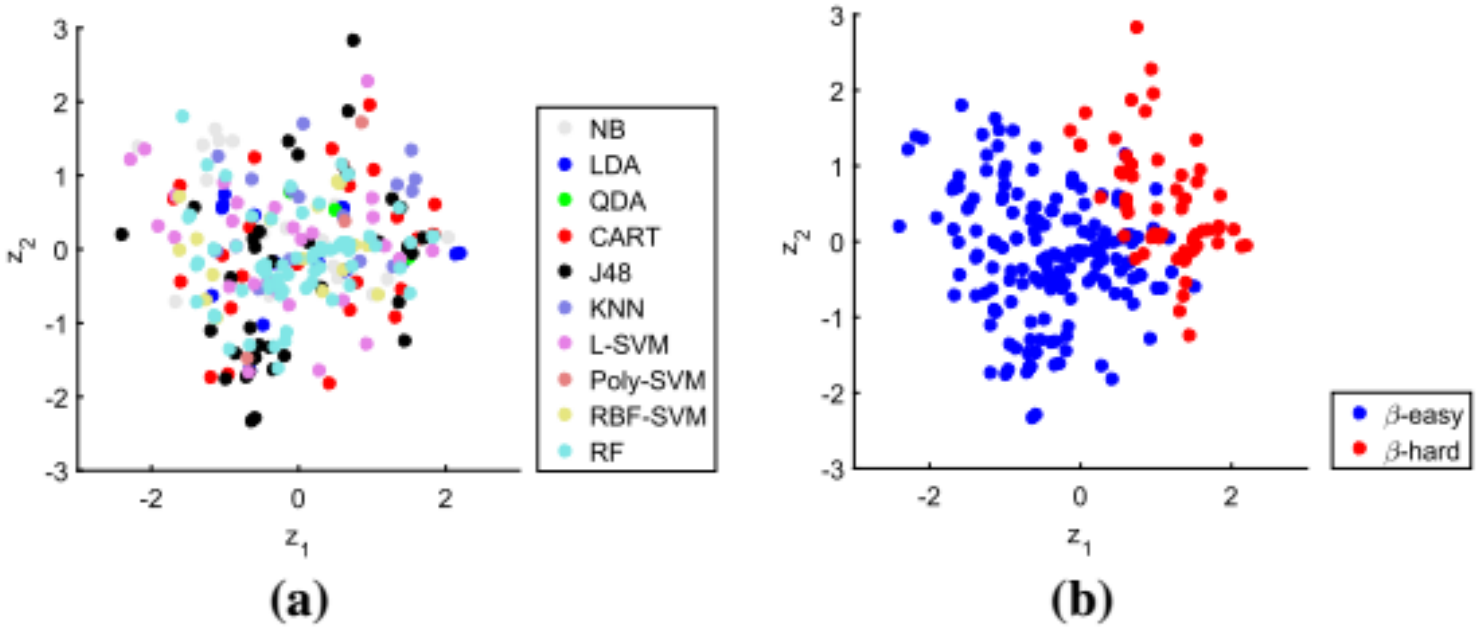


Fig. 6 Overall performance of the algorithm portfolio, with the best algorithm for each instance shown in (**a**), while **b** shows blue marks representing β-easy instances, and red marks representing β-hard instances

- Figure 6 illustrates in the instance space for each instance (a) their best algorithm and (b) their β-hardness.LDA, QDA, poly-SVM and RBF-SVM footprints cover 0% of the instance space. This means there is no dense concentration of instances for which these algorithms are the best.

4.Objective assessment of algorithmic power

Disadvantages

- Given the lack of diversity on the UCI/KEEL/DCol set, most algorithms fail in similar regions of the space, and we lack instances that reveal more subtle differences in performance.

- Large empty areas are visible.For example, a single instance is located at z = [0.744, 2.833], with the next closest located at z = [0.938, 2.281].

The instances in UCI/KEEL/DCol are not sufficiently diverse to reveal the kinds of insights we seek.

5.Generation of artificial problem instances

These limitations provide an opportunity to generate new instances that :

(i) may produce different performance from existing algorithms;

(ii) have features that will place them in empty areas within the space;

(iii) represent modern challenges in machine learning classification.

5.Generation of artificial problem instances

we first consider whether selecting other datasets beyond the UCI repository, KEEL and DCoL would have given a more diverse instance space

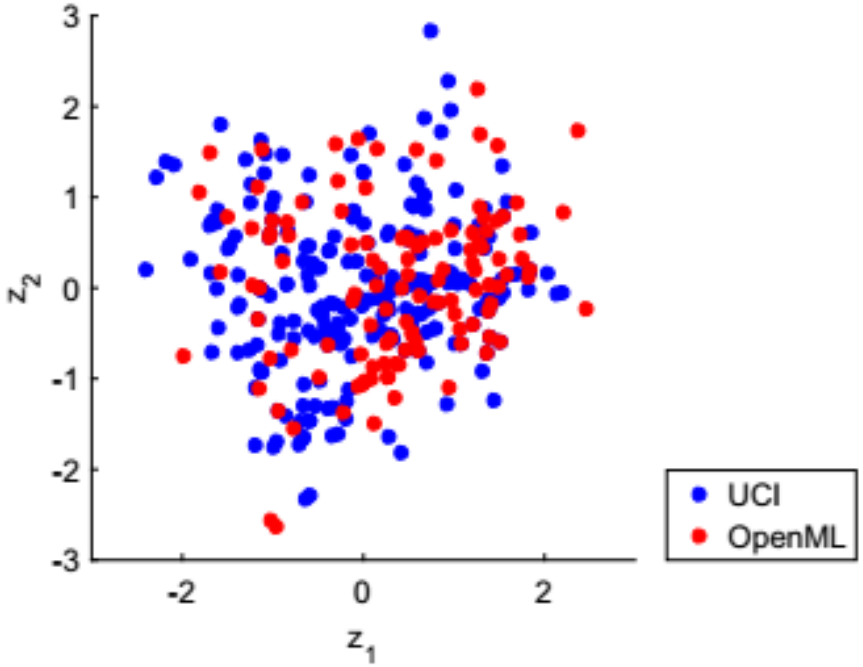


Figure 7 shows the results of projecting a set of OpenML datasets into our instance space. The blue marks represent the original UCI/KEEL/DCoL set, while the red marks are a selection of OpenML datasets of similar size to those in the UCI/KEEL/DCoL set.

5.Generation of artificial problem instances

Although relaxing the size restrictions used in this example may improve the diversity, complexity should be increased without resorting to expanding the dataset size.

To generate instances with a desired target vector of features, we enerate datasets by fitting Gaussian mixture models(GMM).

5.Generation of artificial problem instances

To test this proposed generation approach, we define two experiment types.

- The first one is aimed at validation, where we create datasets whose features mimic those of the well known Iris dataset. The purpose of the experiment is to test whether the generation mechanism can converge to a set of target features.

- For the second experiment, we aim to generate instances located elsewhere in the instance space. We use Iris as a template problem, but we try to evolve the dataset so that its features match a different target vector.

5.Generation of artificial problem instances

Table 7 Error rate of the test algorithms over the Iris-matching instances, with the target defined in the feature space (H) or its 2-D projection (L)

	e_t	NB	LDA	QDA	CART	J48	KNN	L-SVM	Poly-SVM	RBF-SVM	RandF
H	0.015	2.5	4.0	2.6	**8.6**	5.1	3.0	3.0	6.1	2.5	**1.6**
	0.017	4.1	**9.1**	3.9	5.8	6.8	2.8	5.3	6.3	3.1	**4.9**
	0.021	3.5	3.9	3.9	**9.8**	**8.8**	3.2	3.9	8.4	2.2	4.1
	0.029	4.0	6.5	**4.5**	6.3	6.1	**1.9**	6.5	12.0	3.3	**4.4**
	0.032	5.2	4.9	3.7	2.1	1.9	2.7	3.5	6.6	**3.7**	2.0
	0.033	5.2	4.7	4.2	**9.3**	6.3	3.6	5.2	8.7	3.1	3.6
	0.034	5.0	5.3	3.8	7.5	**7.9**	2.5	1.2	9.6	1.3	**5.3**
	0.047	5.6	7.8	3.0	**9.3**	**7.9**	**1.5**	**7.6**	12.0	**4.4**	**5.2**
	0.067	5.1	7.8	**5.6**	8.0	**7.8**	4.0	**6.9**	11.9	3.0	**4.7**
	0.139	6.2	**13.5**	4.2	**8.9**	5.5	4.8	**13.5**	**17.3**	**3.8**	4.2
L	0.000	3.1	**10.8**	0.5	4.8	4.3	3.2	6.3	10.2	**3.7**	2.2
	0.000	1.6	4.5	2.2	3.6	3.2	**1.8**	1.3	5.5	1.2	2.1
	0.000	1.2	3.0	1.2	0.9	0.9	**2.0**	0.7	3.9	1.5	0.9
	0.000	5.5	4.8	2.0	13.3	6.0	**1.5**	2.0	11.2	1.5	**5.0**
	0.030	3.9	5.6	3.7	6.7	6.6	**2.1**	3.7	11.6	**0.8**	**2.5**
	0.040	**6.9**	7.1	**6.7**	7.3	**7.2**	**0.7**	6.5	**15.8**	2.9	6.0
	0.100	**12.5**	**14.6**	3.7	**10.4**	**9.6**	**2.2**	**14.6**	**18.3**	2.7	**4.2**
	0.100	**11.9**	**18.2**	**5.3**	**12.3**	5.1	**0.9**	**10.7**	**19.7**	**3.5**	2.9
	0.160	**11.3**	**15.3**	**9.2**	**10.4**	**12.8**	4.4	**12.2**	**25.9**	**5.7**	**8.6**
	0.230	**12.1**	**8.3**	**6.7**	**10.2**	**10.6**	3.3	**9.5**	**16.2**	**5.3**	**9.2**
AVG	0.031	4.6	6.7	3.9	7.6	6.4	3.0	5.7	9.9	3.1	4.0
		(3.1)	(6.6)	(2.4)	(4.3)	(3.1)	(1.5)	(4.1)	(6.3)	(1.2)	(1.8)
Target		3.1	1.3	1.8	4.0	4.0	4.0	2.7	5.8	2.2	3.1
$\rho_{e,p}$		0.821	0.609	0.718	0.461	0.624	0.401	0.762	0.765	0.708	0.708

The results of the first experiment are presented in Table 7. Error rate of the test algorithms over the Iris-matching instances.

The table shows that as et increases, the difference in ER to Iris increases, as demonstrated by ρ with the exception of KNN.On average, the performance of the generated instances differs by 3.1% compared to Iris.

5.Generation of artificial problem instances

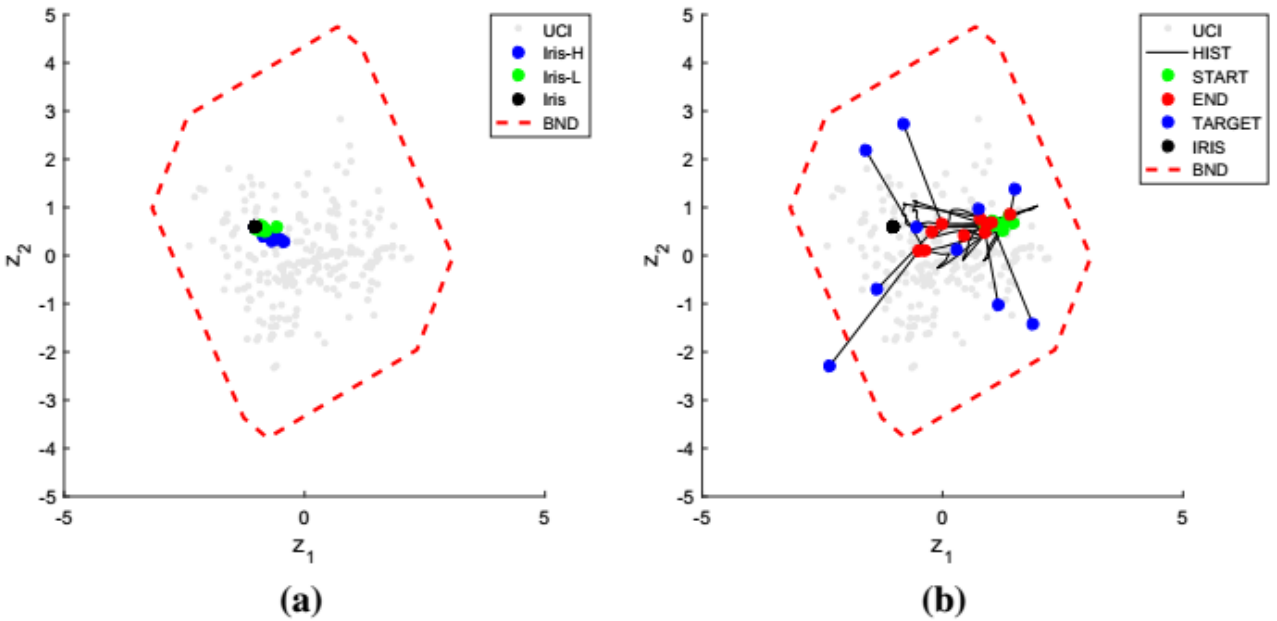


Fig. 8 Instance space showing Iris dataset (black) and attempts to generate new datasets that are **a** similar to Iris, **b** Iris-sized instances located elsewhere (red) based on target features (blue)

- The location of the Iris dataset and the newly generated Iris-like datasets in the instance space are shown in Fig. 8a and confirm that the new instances indeed have similar features to Iris.
- Our generation approach can produce instances and those new instances get similar performance from the algorithms.

5.Generation of artificial problem instances

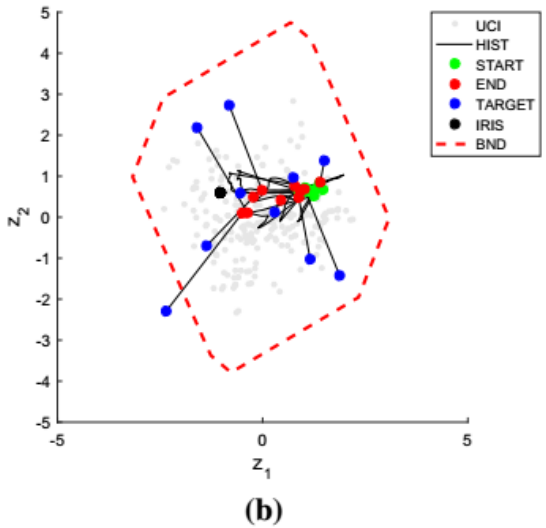


Figure 8b shows the results for the second experiment.

- We can generate datasets that are located away from Iris, and have different features despite having the same number of observations, attributes and classes.
- The potential of the method for generating new test instances has been demonstrated.

6.Conclusions

This paper addresses the issue of objective performance evaluation of machine learning classifiers, and examines the criticality of test instances to support conclusions.

- A comprehensive methodology has been developed to enable the quality of the UCI repository and other test instances and for new classification datasets to be generated with controllable features.
- We have proposed a new dimension reduction technique suited to our visualization objective.
- We proposed a method to generate new test instances, aiming to enrich the repository's diversity.

6.Conclusions

Future research

- Further research on this topic will develop theoretical upper and lower bounds on the features.

- We will also continue to examine the most efficient representation of a dataset to enable the instance space to be filled with new instances of arbitrary size.

- The OpenML repository provides an excellent collection of meta-data, additional features and algorithms.

- New features may need to be selected from the extended set of meta-features to explain the challenges of new instances.