Generating custom classification datasets by targeting the instance space

Mario A. Muñoz
School of Mathematical Sciences, Monash University
Clayton, Victoria 3800, Australia
mario.munoz@monash.edu

Kate Smith-Miles
School of Mathematical Sciences, Monash University
Clayton, Victoria 3800, Australia
kate.smith-miles@monash.edu

ABSTRACT

While machine learning has evolved at a fast pace in the last decades, the testing procedure of new methods may be not keeping pace. It often relies on well-studied collections of classification datasets such as the UCI repository. However, a meta-analysis through features has showed that most datasets from UCI are not sufficiently challenging to expose unique weaknesses of algorithms. In this paper we present a method to generate datasets with continuous, binary and categorical attributes, through the fitting of a Gaussian Mixture Model and a set of generalized Bernoulli distributions. By targeting empty areas of the instance space, this method has the potential to generate datasets with more diverse feature values.

CCS CONCEPTS

•**Computing methodologies** → **Model verification and validation;** *Supervised learning by classification;* •**Information systems** → *Clustering and classification;*

ACM Reference format:
Mario A. Muñoz and Kate Smith-Miles. 2017. Generating custom classification datasets by targeting the instance space. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 7 pages.
DOI: http://dx.doi.org/10.1145/3067695.3082532

1 INTRODUCTION

Machine learning has evolved at a fast pace in the last decades, thanks to cheaper computing power and steady advances in algorithm development. However, the common testing procedure of new methods may be not keeping pace, as it often relies on well studied collections such as the UCI repository [22]. Without a doubt, this repository has had enormous impact on research by ensuring comparability of results. However, concerns have emerged on the representativeness of the datasets in the repository and the possibility of over-fitting the algorithms. In recent work [14], we demonstrated the limitations of this set for refined algorithm evaluation. In particular, a meta-analysis through features showed that most datasets from UCI are very similar and not sufficiently challenging to expose unique weaknesses of algorithms; hence, the performance from fundamentally different algorithms, such as Support Vector Machines (SVM) and Random Forests (RF), is also similar. Therefore, there is an urgent need for new datasets that: (i) may produce different performance from existing algorithms, such that their strengths and weaknesses can be better understood; (ii) have features that will place them away from the existing repository, or help push the boundaries of the currently studied datasets; and (iii) represent modern challenges in machine learning classification.

Perhaps the most common way to artificially generate test datasets is to select and sample an arbitrary probability distribution. However, this approach lacks control as there is no guarantee that the resulting dataset will have specific features. In [12], an alternative method is proposed, in which a "seed" dataset is adjusted by evolving each observation. However, this approach resulted in very little change in the features of the dataset. Furthermore, as the number of observations increases, the evolution process becomes quickly intractable. An alternative is provided in [20], where new datasets are obtained by switching an independent attribute with the class vector. Assuming q categorical attributes, it is possible to obtain q new derived datasets. However, this approach is susceptible to missing target values, skewed class distributions, or difficulties when the new class is completely uncorrelated to the independent variables.

In [14], we presented a proof-of-concept for a method to generate classification datasets based on the fitting and sampling of a Gaussian Mixture Model (GMM). To obtain a dataset with a target feature vector, we minimized the error between the dataset features and the target, assuming a constant random seed. While this method had a number of advantages, it produced datasets whose attributes can only be Gaussian distributed real values, eliminating the possibility of more complex attribute types. Therefore, in this paper we present an extension of this method that includes the possibility to generate binary and categorical attributes, through the fitting of generalized Bernoulli distributions (GBDs). This new functionality allowed us to generate datasets with a more diverse feature values.

The remainder of this paper is organized as follows. In Section 2, we present details of the process employed to generate a two dimensional *instance space* where the relative difficulty of the UCI datasets and algorithm performances across the space can be visualized. The instance space also allow us to identify areas where the UCI is not concentrated; hence, they are ideal candidates for the generation of new datasets. In Section 3, we describe the technical details of the new generator. In Section 4, we present a two-stage validation procedure and its results: first, we attempt to mimic a number of existing datasets from UCI, then, we attempt to fit datasets targeting specific new areas of the instance space. We conclude the paper

GECCO '17 Companion, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3067695.3082532

by presenting the open problems and some suggestions for further research in Section 5.

2 GENERATING AN INSTANCE SPACE

The instance space is a visual representation of a set of problems, which was conceived as tool for the analysis of the strengths and weaknesses of individual algorithms [17–19]. Fundamentally, it is an extension of Rice's algorithm selection framework [16], whose aim is to predict which algorithm is likely to perform best in a given problem. The extended framework, which is illustrated in Figure 1, relies on measurable features of the problem instances that correlate with difficulty to make its predictions.

The first step to construct the instance space is to select a set of benchmark problems, for which their features and algorithm performance are collected. In the particular case of machine learning, the work in meta-learning has provided a significant collection of statistical features for classification problems [1, 3, 6, 10]. Once the meta-data has been collected, a custom dimensionality reduction algorithm is employed to project the information in two dimensions, such that both features and algorithm performance vary smoothly and predictably across the space [14]. This exposes trends in features and algorithm performance, and helps to partition the instance space as pockets of strength and weaknesses, which can be used to understand which features are being exploited or are causing difficulties [17]. Objective measures are used to summarize each algorithm's relative power across the broadest instance space [17]. Moreover, the location of the existing benchmarks in the instance space reveals much about their diversity and challenge. Finally, a methodology is used to evolve new test instances by targeting points in the empty areas of the instance space [18].

In our recent work [14], we constructed an instance space using as benchmark instances 210 datasets from the University of California Irvine (UCI) [11], 19 datasets from the Knowledge Extraction Evolutionary Learning (KEEL) repositories [2], and six datasets from the Data Complexity library[1]: in total 235 datasets. Using meta-analysis, we identified a set of uncorrelated features that are linearly related to algorithm performance, and measured characteristics of the dataset that are known or expected to make a classification task harder. For example, non-normality within classes, and redundant or (nearly) linearly dependent attributes. These features are:

- **Maximum normalized entropy of the attributes,** $H(\mathbf{X})'_{\max}$, quantifies the highest amount of information contained in the data assuming independent attributes [13].
- **Normalized entropy of class attribute,** H'_c, it is a measure of problem imbalance, which is maximal when all the classes are equally probable.
- **Mean mutual information of attributes and class,** $\overline{M}_{CX}$, is a measure of the shared information between attribute X_i and class C [13].
- **Error rate of the decision node,** DN_{ER}, provides with an indication of linear separability in the data [4]. A decision node is a tree consisting only of its root node. The splitting attribute and value in the root are selected so as to maximize the information gain ratio [5].

[1] DCol - http://dcol.sourceforge.net/

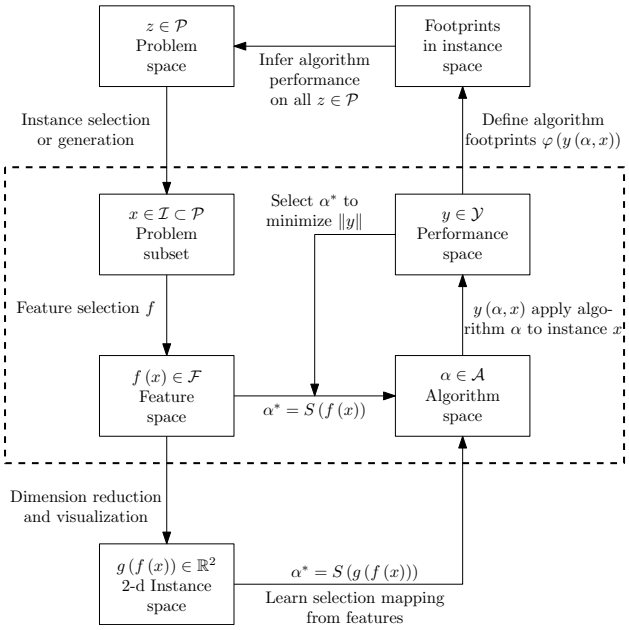

Figure 1: Methodological framework, extending Rice's Algorithm Selection Problem shown within the dashed box

- **Standard deviation of the weighted distance,** $SD(\nu)$, is a measure of the sparsity of the instances in a problem [21]. Small values of $SD(\nu)$ indicate that instances are homogeneously distributed in the feature space, whereas large values indicate non-homogeneous instances, e.g., present clusters.
- **Maximum feature efficiency,** $F3$, measures the of linear separability achievable when using the most discriminative attribute.
- **Collective feature efficiency,** $F4$, extends the concept in $F3$ to account for the discriminative ability of multiple attributes in the dataset [15].
- **Training error of linear classifier,** $(L2)$, is used to assess linear separability of the classes in the training set [8].
- **Fraction of points on the class boundary,** $N1$, estimates the length of the class boundary and provides with a complexity measure of the boundary between classes [15]. Small values of $N1$ indicate that there are only a few points along the boundary, whereas large $N1$ values indicate that the majority of points lay along the boundary.
- **Nonlinearity of the one-nearest neighbor classifier,** $N4$, estimates the nonlinearity of the class boundary using the one-nearest neighbor classifier and the concept of linear interpolation [8, 9].

Next, we analyzed ten popular supervised learners representing a comprehensive range of learning mechanisms: 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). Using the error rate,

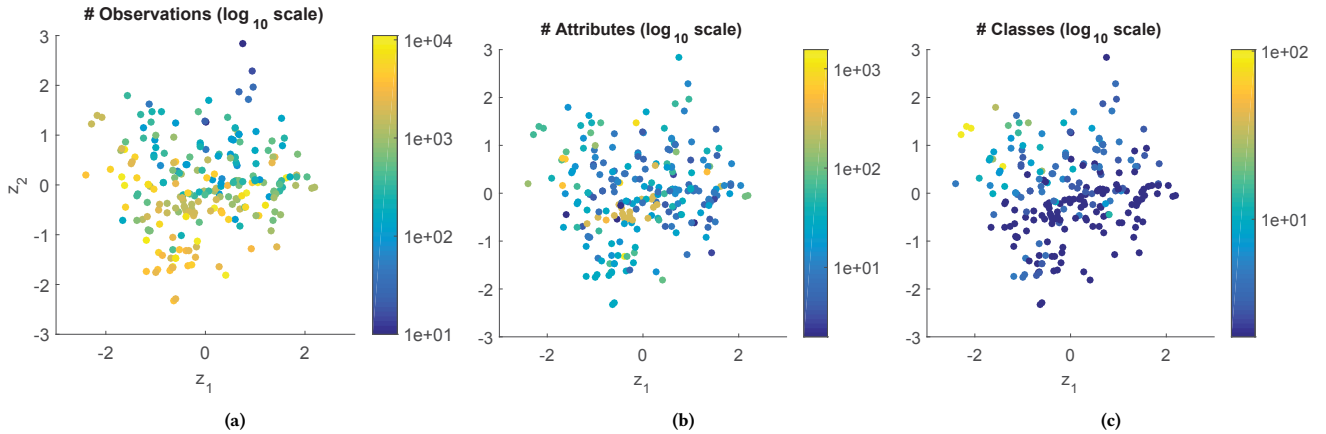

Figure 2: Sizes of the instances in terms of the number of (a) observations, (b) attributes, and (c) classes. All have been $\log_{10}$-scaled.

ER, as performance measure, we calculated the projection that defines the instance space using the following custom dimensionality reduction method [14]. Let $\mathbf{F} = [\mathbf{f}_1\ \mathbf{f}_2\ \ldots\ \mathbf{f}_n] \in \mathbb{R}^{m\times n}$ be the feature data matrix $\mathbf{Y} = [\mathbf{y}_1\ \mathbf{y}_2\ \ldots\ \mathbf{y}_n] \in \mathbb{R}^{\alpha\times n}$ be an algorithm performance matrix , where m is the number of features, n is the number of datasets, and α is the number of algorithms. Then, we achieve an ideal projection of the instances into d dimensions if we can find $\mathbf{A}_r \in \mathbb{R}^{d\times m}$, $\mathbf{B}_r \in \mathbb{R}^{m\times d}$ and $\mathbf{C}_r \in \mathbb{R}^{\alpha\times d}$ that solves the following optimization problem:

$$
(\mathcal{D}) \quad \begin{aligned} \min \quad & \|\mathbf{F} - \mathbf{B}_r\mathbf{Z}\|_F^2 + \|\mathbf{Y} - \mathbf{C}_r\mathbf{Z}\|_F^2 \\ \text{s.t.} \quad & \mathbf{Z} = \mathbf{A}_r\mathbf{F} \\ & \mathbf{A}_r \in \mathbb{R}^{d\times m} \\ & \mathbf{B}_r \in \mathbb{R}^{m\times d} \\ & \mathbf{C}_r \in \mathbb{R}^{\alpha\times d} \end{aligned} \tag{1}
$$

Let $\mathbf{F} \in \mathbb{R}^{10\times 235}$ be a matrix whose rows correspond to the features described above, and its columns correspond to the 235 datasets. Each feature was transformed as follows: $F4$ was scaled to $[-0.99999, 0.99999]$ and $\tanh^{-1}$-transformed, $\{H_C', \overline{M}_{CX}, DN_{er}, SD(\nu), F3, L2, N1\}$ were root-squared. Let $\mathbf{Y} \in \mathbb{R}^{10\times 235}$ be a matrix whose rows correspond to root-squared error rate of the algorithms listed above. Both features and error rates were normalized to $\mathcal{N}(0, 1)$.

We solve numerically $(\mathcal{D})$ using BIPOP-CMA-ES, a stochastic, iterative, variable metric method with demonstrated effectiveness in middle sized optimization problems [7]. To use this method, we represent $\{\mathbf{A}_r, \mathbf{B}_r, \mathbf{C}_r\}$ as a column vector by concatenating the matrix columns. We run 30 times BIPOP-CMA-ES starting from random positions, using the default parameters and a maximum of 10^5 evaluations of $(\mathcal{D})$. Given that all runs converged to the same error, we selected the best solution using a measure of topological preservation: the Pearson Correlation between the distances in the feature space, $\|\mathbf{f}_i - \mathbf{f}_j\|$, and the distances in the instance space, $\|\mathbf{z}_i - \mathbf{z}_j\|$ [23]. The chosen projection from the 10-dimensional feature space to the 2-dimensional instance space is:

$$
\mathbf{Z} = \begin{bmatrix} 0.070 & 0.180 \\ 0.094 & 0.618 \\ -0.277 & -0.052 \\ 0.114 & 0.192 \\ 0.045 & -0.100 \\ -0.128 & 0.151 \\ -0.045 & 0.077 \\ 0.184 & 0.017 \\ 0.449 & 0.223 \\ 0.132 & -0.112 \end{bmatrix}^\top \begin{bmatrix} H(\mathbf{X})'_{\max} \\ H_c' \\ \overline{M}_{CX} \\ DN_{ER} \\ SD(\nu) \\ F3 \\ F4 \\ L2 \\ N1 \\ N4 \end{bmatrix} \tag{2}
$$

Figure 2 illustrates the instance space, where each point represents a dataset. We used color gradients to illustrate the relationship between the axes and the number of (a) observations, (b) attributes, and (c) classes. The figure shows that, for our selected 235 datasets, the number of observations increases from top to bottom, while the number of classes from right to left. There is no trend emerging from the number of attributes; hence, it does not influence the performance of the algorithms as much as the number of observations or classes [14]. The links between the problem size will become influential as we attempt to generate new datasets in the next section. Using the coefficient of determination, R^2, to measure the fit of the instance space with the performance of each algorithm, we found that the best fit is achieved for KNN $(R^2 = 0.805)$, and the worse for QDA $(R^2 = 0.367)$. However, with a median R^2 of 0.650, the instance space describes a linear trend between most features and algorithms. In fact, {NB,CART,J48,KNN,L-SVM,Poly-SVM,RMB-SVM} share a trend, finding easier the instances on the bottom left side of the space, whereas {LDA,QDA,RF} tend to find easier those in the bottom center of the space. This means that most of the instances with a high number of observations and classes are relatively easier for most algorithms [14]. This may imply that the larger datasets under examination are not as complex as previously though, which is another limitation of the benchmark set.

3 GENERATING DATASETS BY FITTING DISTRIBUTIONS

The instance space presented in Section 2 is used to identify potential targets for the generation of new test instances [18]. In our previous work [14], we approached this task by tuning and sampling a Gaussian Mixture Model (GMM), using a fixed seed to guarantee some level of repeatability. Then, the feature vector from the sample, $\mathbf{f}_S$, was compared to the desired target vector of features, $\mathbf{f}_T$, using the Mean Squared Error (MSE). To tune the GMM, we also used BIPOP-CMA-ES. The results in [14] were very encouraging and this approach had a number of advantages: (i) it is scalable by increasing the number of attributes, observations and classes; (ii) it allows some flexibility; (iii) it enables control over the covariance between attributes; (iv) it produces immediately a model of the data distribution, which is a solution to the classification and clustering problem; and (v) the optimization problem is unconstrained. However, this approach produces datasets whose attributes are Gaussian distributed real values, eliminating the possibility of more complex attribute types. To address this limitation, we now include a generalized Bernoulli distribution (GBD) to model binary and categorical attributes to enable more realistic classification datasets to be generated.

As in [14], let us define the GMM as having κ components on g attributes; therefore, the probability of an observation $\mathbf{x}$ being sampled from the GMM is given by:

$$\mathrm{pr}\,(\mathbf{x}) = \sum_{k=1}^{\kappa} \phi_k \mathcal{N}\left(\boldsymbol{\mu_k}, \Sigma_k\right)$$

where $\left\{\phi_k \in \mathbb{R}, \boldsymbol{\mu}_k \in \mathbb{R}^g, \Sigma_k \in \mathbb{R}^{g\times g}\right\}$ are the weight, mean vector, and covariance matrix of a g-variate normal distribution respectively. Since Σ_k must be a positive semi-definite matrix, we can assume the existence of its Cholesky decomposition, i.e., an upper triangular matrix $\mathbf{A}_k$ such that $\Sigma_k = \mathbf{A}_k^\top \mathbf{A}_k$. Assume that some attribute pairs are linearly independent; hence, they will have covariance zero. Therefore, we define a constant upper triangular boolean matrix $\mathbf{B}_k \in [0,1]^{g\times g}$, which indexes the non-zero elements of $\mathbf{A}_k$, $\mathbf{a}_k$.

Now, let us define a GBD per each binary and categorical attribute with $\Lambda \geq 2$ labels. Therefore, the probability of an observation x being sampled from the distribution is given by:

$$\mathrm{pr}\,(x|\boldsymbol{\gamma}) = \prod_{\lambda=1}^{\Lambda} \gamma_\lambda^{\delta_{x\lambda}}$$

where $\boldsymbol{\gamma}$ is a vector of Λ elements that represents the probability of observing each label λ and $\sum_{\lambda=1}^{\Lambda} \gamma_\lambda = 1$, and $\delta_{x\lambda}$ is the Kronecker delta. We control the number of labels per attribute by setting the value of Λ, which is stored in a vector $\boldsymbol{\Lambda}$ of size q. We used BIPOP-CMA-ES to tune both distributions simultaneously. To use this method, we must represent all the distribution parameters, $\left\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_\kappa, \mathbf{a}_1, \ldots \mathbf{a}_\kappa, \boldsymbol{\phi}, \boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_q\right\}$, as a vector $\boldsymbol{\theta}$. A dataset is completely defined by setting the number of observations, p.

Algorithms 1 and 2 present the key procedures in the generation approach. Algorithm 1 stores a random seed, initializes the parameters, creates the indexing matrices, $\mathbf{B}_k$, defines the number of labels per categorical variable, and generates the class vector, $\mathbf{y}$. On

Input: The number of observations, p, continuous attributes, g, gaussian components, κ, categorical attributes, p, classes, K, the minimum and maximum number of labels $\{\Lambda\text{min}, \Lambda_{\max}\}$, and the lower and upper bounds for the means and covariances $\{a, b\}$.
Output: The vector parameter, $\boldsymbol{\theta}$, the indexing matrices $\{\mathbf{B}_1, \ldots, \mathbf{B}_\kappa\}$, the vector of labels numbers, $\boldsymbol{\Lambda}$, the class attribute vector, $\mathbf{y}$, and the random seed, s.

$s \leftarrow$ GetCurrentRandomSeed(); // Save the current random seed
// Initialize the parameters for the GMM
for $k \leftarrow 1$ **to** κ **do**
  // Generate an array of real randoms between $[a, b]$.
  $\boldsymbol{\mu}_k \leftarrow$ UniformRealRandom$(1, g, a, b)$;
  // Generate an array of binary randoms
  $\mathbf{B}_k \leftarrow$ BinaryRandom(g, g);
  $\mathbf{B}_k \leftarrow$ UpperTriangularMatrix$(\mathbf{B}_k)$;
  $\mathbf{A}_k \leftarrow$ UniformRealRandom(g, g, a, b);
  // Calculate the Hadamard product to index $\mathbf{A}_k$
  $\mathbf{A}_k \leftarrow \mathbf{A}_k \circ \mathbf{B}_k$;
  // Find the non-zero elements of $\mathbf{A}_k$
  $\mathbf{a}_k \leftarrow \{a : a \in \mathbf{A}_k, a \neq 0\}$;
  $\phi_k \leftarrow$ UniformRealRandom$(1, 1, 0, 1)$
end
// Initialize the parameters for the GBDs
for $i \leftarrow 1$ **to** p **do**
  // Set the number of labels using a random integer
  $\Lambda_i \leftarrow$ UniformIntegerRandom$(1, 1, \Lambda_{\min}, \Lambda_{\max})$;
  $\boldsymbol{\gamma} \leftarrow$ UniformRealRandom$(1, \Lambda_i, 0, 1)$;
end
// Set all parameters as vector
$\boldsymbol{\theta} \leftarrow \left[\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_\kappa, \mathbf{a}_1, \ldots \mathbf{a}_\kappa, \boldsymbol{\phi}, \boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_q\right]^\top$;
// Generate $\mathbf{y}$ as a vector of p random integers between $[1, K]$
$\mathbf{y} \leftarrow$ UniformIntegerRandom$(p, 1, 1, K)$;

Algorithm 1: Procedure to initialize the parameters of the generator, given the specified dataset size.

Input: The vector parameter, $\boldsymbol{\theta}$, the indexing matrices $\{\mathbf{B}_1, \ldots, \mathbf{B}_\kappa\}$, the vector of labels numbers, $\boldsymbol{\Lambda}$, the class attribute vector, $\mathbf{y}$, and the random seed, s.
Output: The generated dataset, X, and its features, $\mathbf{f}_S$.

// Extract the data from the vector of parameters
$\left[\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_\kappa, \mathbf{a}_1, \ldots \mathbf{a}_\kappa, \boldsymbol{\phi}, \boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_q\right] \leftarrow$ ExtractParameters$(\boldsymbol{\theta})$;
// Reconstruct the covariance matrices
for $k \leftarrow 1$ **to** κ **do**
  $\mathbf{A}_k \leftarrow \mathbf{a}_k(\mathbf{B}_k)$; // Reconstruct $\mathbf{A}_k$ by indexing $\mathbf{a}_k$
  $\Sigma_k \leftarrow \mathbf{A}_k^\top \mathbf{A}_k$; // Calculate Σ_k using Cholesky decomposition
end
$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} \oslash$ SumVector$(\boldsymbol{\phi})$; // Scale $\boldsymbol{\phi}$ between $[0, 1]$
// Store the current random seed and then use the inputed random seed
$s_{\text{aux}} \leftarrow$ GetCurrentRandomSeed();
SetRandomSeed(s);
// Sample the Gaussian Mixture Model
$\mathbf{X}_g \leftarrow$ SampleGaussianMixtureModel$(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_\kappa, \Sigma_1, \ldots \Sigma_\kappa, \boldsymbol{\phi}, p, g)$;
// Sample the generalized Bernoulli distributions
$\mathbf{X}_c \leftarrow []$; // Set an empty matrix $\mathbf{X}_c$
for $i \leftarrow 1$ **to** p **do**
  $\boldsymbol{\gamma}_i \leftarrow \boldsymbol{\gamma}_i \oslash$ SumVector$(\boldsymbol{\gamma}_i)$; // Scale $\boldsymbol{\gamma}_i$ between $[0, 1]$
  $\mathbf{x}_i \leftarrow$ SampleGenBernoulli$(\boldsymbol{\gamma}_i, \Lambda_i, p)$;
  $\mathbf{X}_c \leftarrow [\mathbf{X}_c\ \mathbf{x}_i]$; // Concatenate the sampled vectors
end
SetRandomSeed(s_{aux}); // Restore the random seed
// Concatenate the attribute matrices and class vector
$\mathbf{X} \leftarrow \left[\mathbf{X}_g\ \mathbf{X}_c\ \mathbf{y}\right]$;
$\mathbf{f}_S \leftarrow$ CalculateFeatures$(\mathbf{X})$; // Calculate the features

Algorithm 2: Procedure to generate a dataset, $\mathbf{X}$, using the vector of parameters, $\boldsymbol{\theta}$, and calculate the features. The $\oslash$ operator represents element-wise division of a vector.

the other hand, Algorithm 2 generates the dataset and calculates the features described in Section 2. While this approach maintains the advantages discussed above after extending to binary and categorical attributes, the following issues remain: (a) the fitting problem is known to have local optima; and (b) it can be can be computationally expensive for very large datasets, or inaccurate for very small ones.

4 VALIDATION

For testing the generation approach, we carry out two different experiments. On the first one, the aim is to also test the suitability of the instance space as representation. To do so, we create datasets whose features mimic those of two UCI datasets: Iris and Heart Switzerland with missing values removed (Heart). Iris is a well known dataset with $\{p = 150, g = 4, K = 3\}$, whereas Heart is a smaller dataset with $\{p = 43, g = 4, q = 7, K = 5\}$ located at the upper right area of the space, an mostly empty area in which the tested algorithms performed poorly. Therefore, new datasets are required to clarify the algorithm performance in this area. Figure 3 illustrates the location of Iris and Heart compared to the UCI set. Unless stated otherwise, we set $\kappa = 3K$, $\Lambda_{\min} = 2$ and $\Lambda_{\max} = 6$. For the Iris set, we run the following trials:

High-Continuous Targets are set in the 10-dimensional feature space, with $\{p = 150, g = 4, K = 3\}$.

Low Targets are set in the 2-dimensional instance space with

Continuous $\{p = 150, g = 4, K = 3\}$.

Categorical $\{p = 150, q = 4, K = 3\}$.

Mix 3/3 $\{p = 150, g = 4, q = 3, K = 3\}$.

The purpose of the **High-Continuous** trial is to demonstrate our ability to approximate target the actual feature vector, whereas the purpose of the **Low** trials is to demonstrate or ability to approximate the target with diverse attribute types in the instance space. Given that Iris has only continuous attributes, the performance the algorithms is expected to be different for the categorical and mix trials.

For the Heart set, we run the following trials with targets set in the 2-dimensional instance space:

Low-Continuous $\{p = 43, g = 11, K = 5\}$.

Low-Categorical $\{p = 43, q = 11, K = 5\}$.

Low-Mixture 4/7 $\{p = 43, g = 4, q = 7, K = 5\}$.

All trials are repeated ten times with a soft bound of 10^5 function evaluations, i.e., the number of times a dataset is generated and tested, and a stopping criteria of $MSE < 10^{-3}$. The values of $\boldsymbol{\theta}$ are initialized by sampling from a uniform distribution between $[-10, 10]$ for the covariances, and between $[0, 1]$ for the label probabilities. The results are presented in Table 1, where each column represents the error rate, *ER*, of each algorithm. MSE is the mean squared error from the target features, $\mathbf{f}_T$, to the dataset features, $\mathbf{f}_S$. The correlation between the *ER* of the target and the average *ER* of the generated datasets is presented in ρ.

We can observe that fitting Iris in the high dimensional feature space produces the datasets with the highest correlation ($\rho = 0.640$), with the largest difference in *ER* being for LDA with a 3.7%, and the lowest being for KNN with 0.1%, and the average being 1.0%. This indicates that the datasets on average produce similar behavior from the selected algorithms. This can be attributed to the fact that

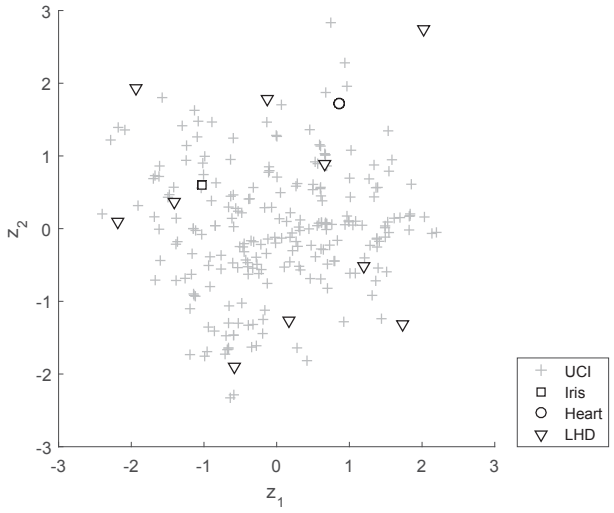

Figure 3: Location of generation targets for both experiments in comparison to the UCI sets.

the instance space loses some information by projecting the data into 2-dimensions. Evidence of this can also be seen in the slightly lower correlation ($\rho = 0.569$), and larger average difference of 1.5%. However, when the attributes are of a different type, the results can have strong variations. For example, LDA and QDA produce higher *ER* for the Categorical and Mixture trials, with QDA obtaining *ER*s above 60.0%. This can be explained by the low predictive ability of the instance space for these two algorithms. This result also indicates that categorical variables strongly influence the performance of LDA and QDA. On the other hand, the results for Heart show correlations above 0.6. Moreover, all the algorithms struggle with these datasets, with an average *ER* of 37.5%. Using categorical variables improves the ability of the method to reduce MSE.

In the second experiment, we aim to generate instances located elsewhere in the instance space. To do so, we set the target feature vectors using a latin hyper-cube design (LHD) sample in the instance space, with bounds determined by the largest and smallest values for $\mathbf{Z}$. We fix the dataset size to $\{p = 150, g = 4, K = 3\}$ or $\{p = 43, g = 4, q = 7, K = 5\}$, which are the same size as Iris and Heart datasets. This restriction would limit our ability to achieve MSE=0, due to the relationships observed in Figure 2 between $\{p, g, q, K\}$ and the instance location. However, this experiment will give us an indication of the location bounds of the Iris- and Heart- sized datasets in the space and their complexity. As before, we set the value of $\kappa = 3K$, and initialize the values of the covariances by sampling a uniform distribution between $[-10, 10]$. All trials are repeated ten times with a soft bound of 10^5 function evaluations.

Figure 4 demonstrates that there may be a boundary outside of which Iris- and Heart-sized datasets can not be generated, explaining the higher average MSE of 1.3841 and 3.3966 respectively. This is in part due to the correlation (0.533) between the number of classes, K, and the normalized entropy of class attribute, $H_c^{'}$; and the correlation (0.656) between the number of observations, p, and the standard deviation of the weighted distance, $SD(\nu)$. Therefore,

Table 1: Results from the first experiment type, where each row represents the average of ten trials, and the columns represent the error rate, *ER*, of each algorithm. MSE is the mean squared error from the target features, $\mathbf{f}_T$, to the dataset features, $\mathbf{f}_S$. The correlation between the *ER* of the target and the average *ER* of the generated datasets is presented in ρ.

		MSE	ρ	*ER*									
				NB	LDA	QDA	CART	J48	KNN	L-SVM	poly-SVM	RBF-SVM	RF
Iris	High - Continuous	0.1668	0.640	2.7%	5.0%	1.6%	4.9%	4.2%	4.1%	3.5%	7.5%	3.7%	2.6%
	Low - Continuous	0.0006	0.569	1.2%	3.0%	0.6%	2.8%	2.5%	1.2%	1.4%	4.4%	1.5%	1.5%
	Low - Categorical	0.0006	-0.612	0.4%	42.4%	66.2%	0.1%	0.2%	1.5%	0.1%	2.5%	1.6%	0.0%
	Low - 3/3 Mixture	0.0007	-0.490	2.8%	17.9%	66.0%	1.5%	1.8%	1.9%	0.9%	3.9%	3.0%	0.8%
	Target			3.1%	1.3%	1.8%	4.0%	4.0%	4.0%	2.7%	5.8%	2.2%	3.1%
Heart	Low - Continuous	0.0006	0.624	30.8%	32.1%	78.2%	36.2%	33.0%	29.0%	31.0%	35.2%	32.1%	31.9%
	Low - Categorical	0.0004	0.585	26.7%	46.8%	78.1%	29.1%	26.5%	32.9%	27.4%	31.9%	31.8%	28.7%
	Low - 4/7 Mixture	0.0004	0.504	29.6%	68.4%	77.7%	31.0%	31.9%	30.6%	30.4%	28.3%	36.5%	31.1%
	Target			48.5%	47.0%	71.7%	42.6%	40.3%	39.8%	41.1%	38.4%	39.7%	71.7%

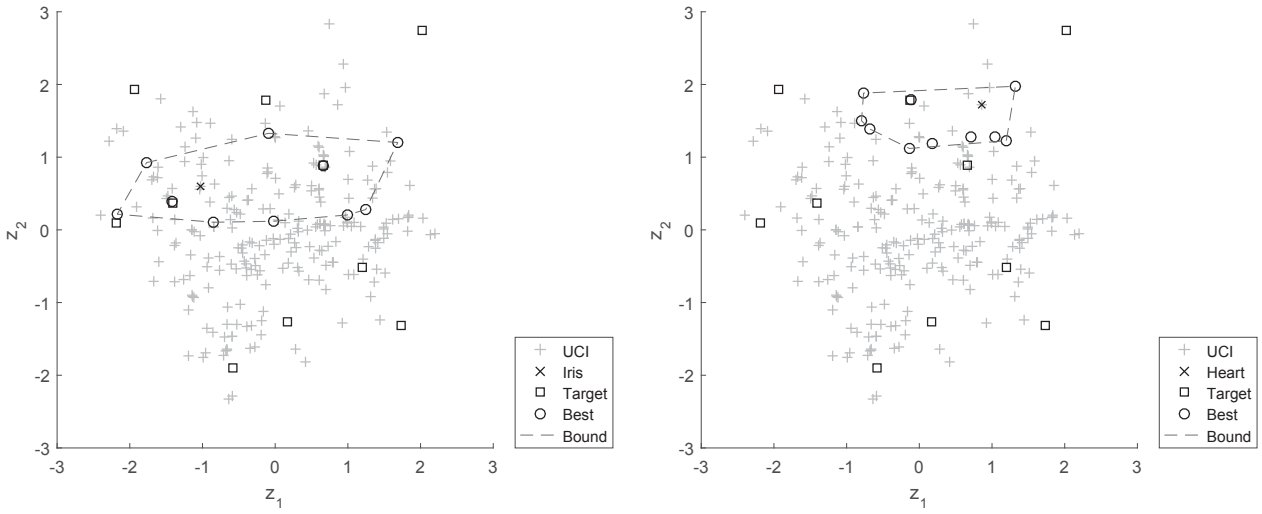

Figure 4: Location of UCI sets, the template problems, the LHD targets and the resulting datasets. Due to the correlation between some features and the dataset size, there may be a bound outside of which similarly sized datasets can not be generated.

unless $\{p, g, q, K\}$ are modified, it would not be possible to cross these boundaries. This result also implies that in order for our generator to have the best chance of reaching the empty spaces of the instance space, we need to set the dataset size to be equal to that of the closest dataset.

5 CONCLUSIONS

In this paper, we have presented a method to generate datasets with continuous, binary and categorical attributes, through the fitting of a Gaussian Mixture Model and a set of generalized Bernoulli distributions. By targeting empty areas of the instance space, this method could generate datasets with a more diverse feature value. However, there are a some of limitations that must be addressed before the system is complete. For example, (a) the fitting problem is known to have local optima; and (b) it can be can be computationally expensive for very large datasets, or inaccurate for very small ones. These issues may be solved by finding a more efficient representation of a dataset. Of course, once we have generated a large number of new datasets with different features and located in unique parts of the instance space, we must verify that they enhance our ability to understand unique strengths and weaknesses of algorithms. It may also be necessary to reevaluate the features that describe the instance space, once all the data is collected.

On the other hand, there are theoretical and computational issues that limit our ability to extensively explore and fill the gaps in the instance space at this time. For example, the precise theoretical bounds of the instance space are unknown. Further research on the upper and lower bounds on the features and their dependencies

would allow us to define the targets more accurately than what was presented in this paper.

ACKNOWLEDGMENTS

This work is funded by the Australian Research Council through the Australian Laureate Fellowship FL140100012. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research. We also thank Dr. Toan Nguyen from Monash University's eResearch team, who implemented parallelized versions of the meta-features routines, resulting in large reductions in the overall computation time.

REFERENCES

[1] D.W. Aha. 1992. Generalizing from case studies: A case study. In *ICML '92*. 1–10.
[2] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. 2010. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Mult. Valued Log. S.* 17, 2-3 (2010), 255–287.
[3] S. Ali and K.A. Smith. 2006. On learning algorithm selection for classification. *Appl. Soft Comput.* 6 (2006), 119–138. DOI: http://dx.doi.org/10.1016/j.asoc.2004.12.002
[4] H. Bensusan and C. Giraud-Carrier. 2000. Casa batló is in passeig de gràcia or how landmark performances can describe tasks. In *Proceedings of the ECML-00 workshop on meta-learning: Building automatic advice strategies for model selection and method combination*. 29–46.
[5] H. Bensusan and C. Giraud-Carrier. 2000. Discovering task neighbourhoods through landmark learning performances. In *Principles of Data Mining and Knowledge Discovery*. Springer, 325–330.
[6] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. 2008. *Metalearning: Applications to data mining*. Springer.
[7] N. Hansen. 2009. Benchmarking a bi-population CMA-ES on the BBOB-2009 Function Testbed. In *GECCO '09*. ACM, 2389–2396. DOI: http://dx.doi.org/10.1145/1570256.1570333
[8] T.K. Ho and M. Basu. 2002. Complexity measures of supervised classification problems. *IEEE Trans. Pattern Anal. Mach. Intell.* 24, 3 (2002), 289–300.
[9] A. Hoekstra and R.P.W. Duin. 1996. On the nonlinearity of pattern classifiers. In *ICPR 13*, Vol. 4. IEEE, 271–275.
[10] J.W Lee and C. Giraud-Carrier. 2013. Automatic selection of classification learning algorithms for data mining practitioners. *Intell. Data Anal.* 17, 4 (2013), 665–678.
[11] M. Lichman. 2013. UCI Machine Learning Repository. (2013). http://archive.ics.uci.edu/ml
[12] N. Macia and E. Bernadó-Mansilla. 2014. Towards UCI+: a mindful repository design. *Inform. Sciences* 261 (2014), 237–262.
[13] D. Michie, D.J. Spiegelhalter, and C.C. Taylor (Eds.). 1994. *Machine learning, neural and statistical classification*. Ellis Horwood.
[14] M.A. Muñoz, L. Villanova, D. Baatar, and K. Smith-Miles. 2017. Instance Spaces for Machine Learning Classification. *Mach. Learn.* (2017).
[15] A. Orriols-Puig, N. Macia, and T.K. Ho. 2010. *Documentation for the data complexity library in C++*.
[16] J.R. Rice. 1976. The Algorithm Selection Problem. In *Advances in Computers*. Vol. 15. Elsevier, 65–118. DOI: http://dx.doi.org/10.1016/S0065-2458(08)60520-3
[17] K. Smith-Miles, D. Baatar, B. Wreford, and R. Lewis. 2014. Towards objective measures of algorithm performance across instance space. *Comput. Oper. Res.* 45 (2014), 12–24. DOI: http://dx.doi.org/10.1016/j.cor.2013.11.015
[18] K. Smith-Miles and S. Bowly. 2015. Generating New Test Instances by Evolving in Instance Space. *Comput. Oper. Res.* 63 (2015), 102–113. DOI: http://dx.doi.org/10.1016/j.cor.2015.04.022
[19] K. Smith-Miles and L. Lopes. 2012. Measuring instance difficulty for combinatorial optimization problems. *Comput. Oper. Res.* 39, 5 (2012), 875–889. DOI: http://dx.doi.org/10.1016/j.cor.2011.07.006
[20] C. Soares. 2009. UCI++: Improved Support for Algorithm Selection Using Datasetoids. In *PAKDD '09*. 499–506. DOI: http://dx.doi.org/10.1007/978-3-642-01307-2_46
[21] R. Vilalta. 1999. Understanding accuracy performance through concept characterization and algorithm analysis. In *Proceedings of the ICML-99 Workshop on Recent Advances in Meta-Learning and Future Work*. 3–9.
[22] K.L. Wagstaff. 2012. Machine Learning that Matters. In *ICML '12*. 529–536.
[23] S. Yarrow, K.A. Razak, A.R. Seitz, and P. Seriès. 2014. Detecting and Quantifying Topography in Neural Maps. *PLoS ONE* 9, 2 (02 2014), 1–14. DOI: http://dx.doi.org/10.1371/journal.pone.0087178