5 REAL-WORLD PROBLEMS VS. ARTIFICIAL DATA SETS

Summary. This chapter studies the complexity of real-world and synthetic problems and highlights some gaps in the complexity space, which will ultimately motivate the generation of new data sets. This comes from the need of having diversity in terms of characteristics to either perform a specialised testing or scan the learners' response across a wider range.

5.1 INTRODUCTION

After realising that data structure and its complexity influence the performance of machine learning techniques in many different ways, it seems interesting to first focus on disentangling the benefits/limitations of real-world problems, and then, focus on the techniques to analyse their behaviour according to the set of complexity estimates. Over the last two decades the competitiveness of classification techniques[1], typically developed for a general purpose, has been claimed over a small and repetitive set of problems. Although a common test bed is useful and necessary to make fair comparisons among algorithms, it can lead to incomplete conclusions about the quality of the learning algorithms if we do not have control over its characteristics, such as similarity between the data sets, as seen in Chapter 2. The study of how real-world problems distribute in data complexity dimensions is aimed at providing a remedy, so that classifiers' behaviours can be understood in the context of the problems' characteristics.

The purpose of this chapter is to critically examine with our complexity yardstick the repositories used by the machine learning community for empirical analysis which contain both (1) real-world problems and (2) synthetic data sets. By visualising the data complexity, we obtain a projection with which we are able to study the current coverage and verify the existence of gaps.

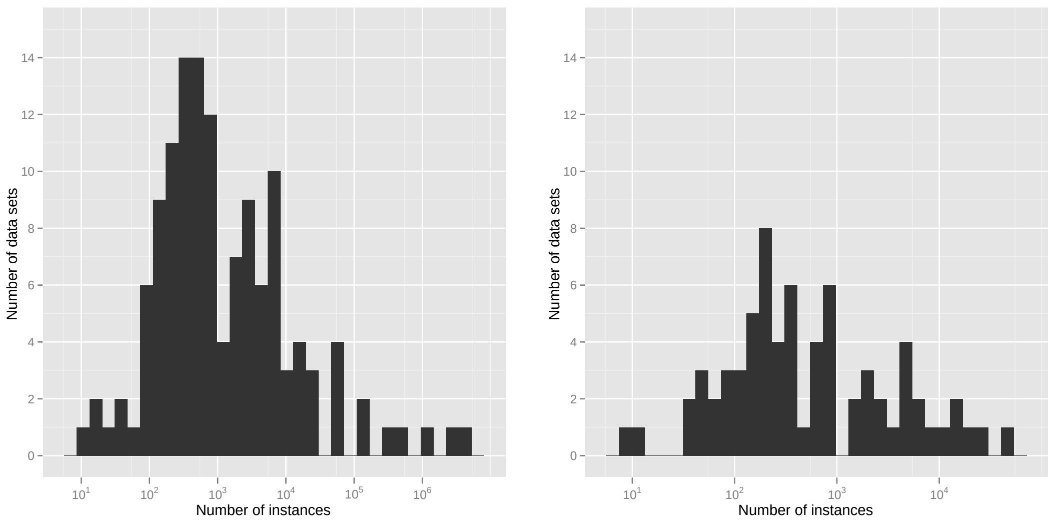
In the following, we first give an inside look at real-world problems. We then analyse the complexity of data sets from the UCI repository, of some manipulated real-world problems from the field of bioinformatics, and of a representative synthetic problem.

5.2 INSIDE LOOK AT REAL-WORLD PROBLEMS

Any data set is a representation of some problem and it is thereby not arbitrary. Data are collected for specific purposes; they are collected to solve specific problems. Nonetheless, the experimental community has relied on real-world or quasi-real-world problems—mostly for classification—to test algorithms and dilute the main concern in the field: to relate the theoretical research to the real world, i.e. to applications. This section briefly discusses the reality of these problems and the enormous amount of non-deterministic actions that affect the definition of a problem.

The scientific community misses many critical aspects of real-world problems, which find their origin in the data collection [Duda et al., 2000]. When does the practitioner have an adequately large and representative set of examples? We often ignore the number of examples we need, the ones which are typical, the attributes we have to measure, etc. Data collection is a crucial phase which should be accompanied by a thoroughly statistical analysis of the real data distribution, and the identification of the noise and tolerance of the physical measures performed. In addition, enlarging or sampling the data can transform the sample set into a very different problem.

1 Our discussions will be restricted to *supervised* classification techniques, also referred to as classifiers, learning algorithms, or learners.



(a) 134 classification problems (b) 70 classification problems

Figure 5.1: Distribution of (a) the entire collection from the UCI and (b) the sample under study.

Indeed, scalability, imbalance, and geometry condition the complexity of the data and the success of the classifier.

The aforementioned issues are left behind since practitioners are not part of this step, they just manipulate given matrices of numbers. Real-world problems found in industry are private—collated from observations and raw statistics derived from areas such as scientific testing and customer data in, often, a large-scale way—, so data from public repositories or released by competitions—often incomplete, obsolete, or unrealistic—are employed.

5.3 REAL-WORLD PROBLEMS FROM THE UCI REPOSITORY

The UCI machine learning repository [Frank and Asuncion, 2010] is the most well-known collection of real-world problems in machine learning; being cited over 1000 times, it is one of the top 100 most cited *papers*[2] in all of computer science. This section closely scrutinises of several problems from the UCI repository. We first describe the data sets and then provide the corresponding complexity analyses.

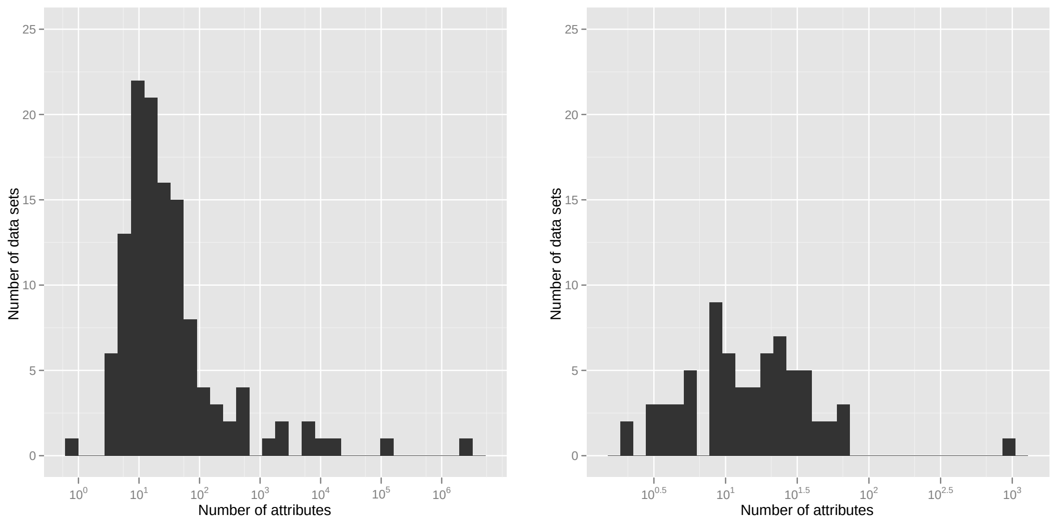
5.3.1 *Description*

The UCI repository is a collection of databases, domain theories, and data generators, which is commonly utilised to conduct empirical analyses of machine learning algorithms. It was created as an File Transfer Protocol (FTP) archive in 1987 by David Aha and fellow graduate students at the University of California, Irvine and has been, hitherto, widely used as a primary source of machine learning data sets. Nevertheless, how much do we know about these data sets?

Two years ago, we gathered 70 classification problems from the UCI repository and transformed them into two-class data sets. Nowadays, the collection has been extended to 134 problems, so our study is based on half the sample. Although we believe that our conclusions can be generalised over the entire collection, since in terms of the number of instances and number of attributes the distribution is similar (see Figs. 5.1 and 5.2), it is necessary to extend the analysis.

Table A.1 in Appendix A provides a description of their extrinsic characteristics. Fig. 5.3 represents the variation of the external descriptors.

2 http://archive.ics.uci.edu/ml/about.html



(a) 134 classification problems (b) 70 classification problems

Figure 5.2: Distribution of (a) the entire collection from the UCI and (b) the sample under study.

We observe that the characteristics range within a very short interval with few outliers. This is typical of a repository composed of small data sets. This matches with Holte [1993] who snubbed the UCI repository for only being a limited sample. In his experimentation, he showed that most of the data sets were correctly classified, reaching 100% of accuracy, with simple rules. So, this sample, despite being composed of real-world problems, does not reflect the conditions that arise in practice. Nowadays, the UCI repository has grown, but its diversity is still low in terms of complexity, as demonstrated in the next section.

5.3.2 *Intrinsic complexity*

Table A.2 shows the output of the DCoL over the aforementioned problems. Fig. 5.4 depicts the distribution of the intrinsic characteristics. Based on the fact that most of the complexity measures lay within the range $[0,1]$, we see that the diversity is still low. Regarding difficulty, we confirm that the sample seems, in general, quite easy since the low values of F2 and moderate high values of F1v and F4 ($F2_\mu = 1.68e-01$, $F1v_\mu = 14.05$, and $F4_\mu = 0.51$) indicate that there are discriminative attributes in the problems. In addition, the low values of N1, N2, and N3 ($N1_\mu = 0.24$, $N2_\mu = 0.56$, and $N3_\mu = 0.16$) that show the complexity of their class boundary is low. If we have a look at the maximum values, except for L1 and N2, outliers only reach medium complexities with respect to boundary difficulty, e.g.

· $L1_{max} = 1.97$,

· $L2_{max} = 0.42$,

· $N1_{max} = 0.68$,

· $N2_{max} = 1$, and

· $N3_{max} = 0.68$.

In Table A.2, red numbers highlight both minimum and maximum values for each complexity measure. Although these local thresholds are distributed across the entire sample, we can clearly see that some problems hold more than one extreme value for different complexity measures at the same time.[3]

3 We realise, then, that these data sets (i.e. *aba*, *bal*, *benford*, *flg*, *fourclass*, *hay*, *hrs*, *irs*, *krk*, *aba*, *lng*, *msh*, *nrs*, *pbc*, *pos*, *shs*, *aba*, *spec*, *tis*, *tnc*, *trn*, *vot*, *wav21*, *whi*, *wpbc*, *yea*, and *zoo*) may be candidates as seeds to generate synthetics data sets, since they appear to be very adequate to push the complexity borders.

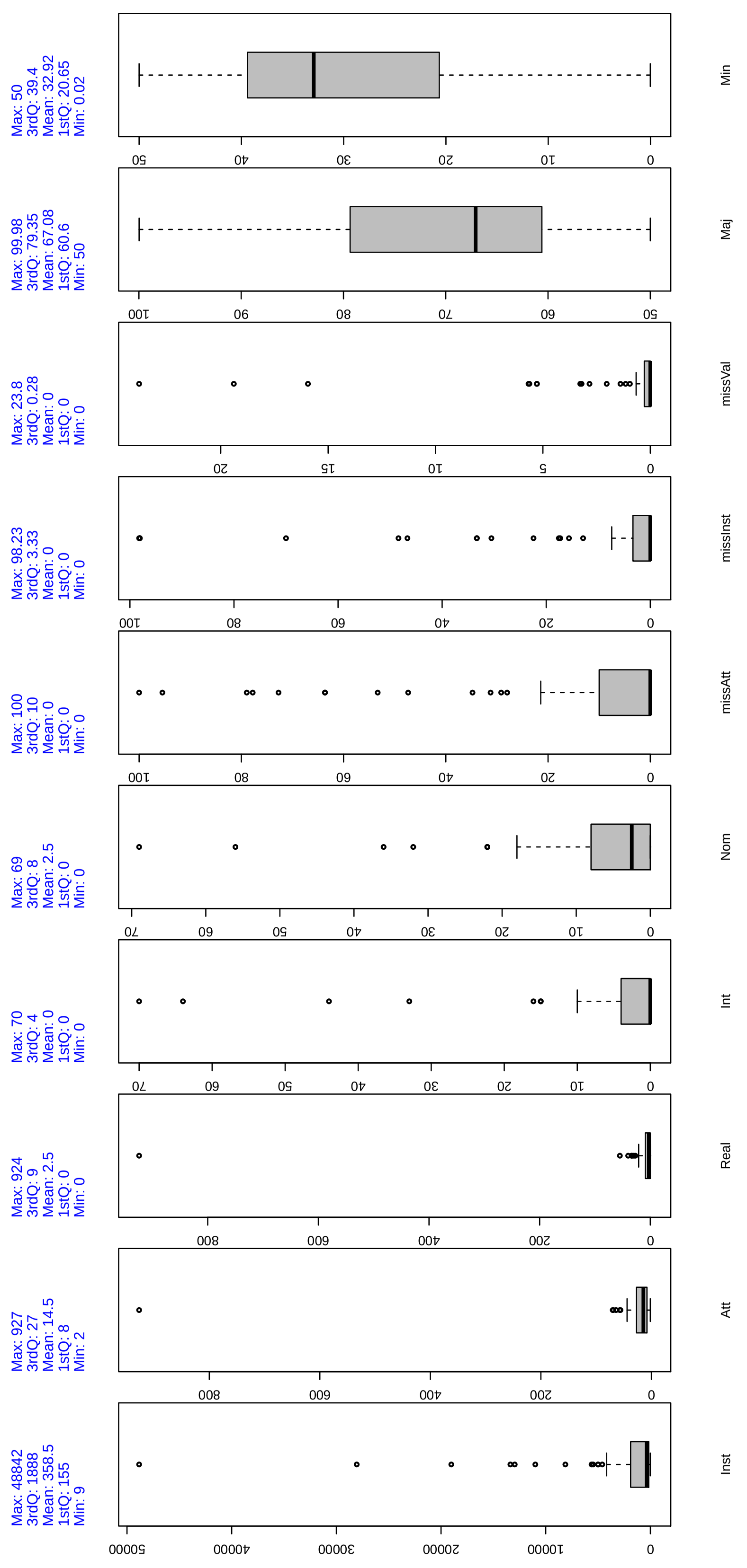


Figure 5.3: Distribution of the extrinsic characteristics of the UCI repository data sets.

On the other hand, Fig. 5.5 shows the correlation between complexity measures; red stars represent the significance level. In a previous study, Ho et al. [2006] detected some correlations which should be analysed so as to find the smallest set of descriptors. Correlations involved the following pairs: F2-L2, F2-L3, F2-N2, L2-N1, L2-N2, L3-N1, L3-N2, N1-N2, and N1-N3. Nevertheless, these correlations are not evident here, at least with this sample. We just notice the strongest correlations (> 0.70) between N1 and N3 with a coefficient of 0.93, F3 and F4 with 0.80, and L2 and N1 with 0.70. This may be interpreted as the sample is not representative enough to reveal such relationships.

Certainly, in order to evaluate the behaviour and performance of classifiers—and even the complexity measures—, the test bed is designed to have different characteristics in mind, such as diversity in the dimensionality—number of instances and number of attributes—, number of classes, class imbalance, etc. This diversity can be found in real-world problems, but is definitely not common enough in the UCI repository. Real-world data sets usually contain noise and inconsistencies, characteristics that are useful to evaluate the robustness of a learning system. In this regard, the UCI repository offers some problems with missing values even if it leaves room for improvement. If the objective of testing learners' limitations is, however, to evaluate the scalability of the algorithms, a suitable set of problems would be required. In such a case, synthetic data are an interesting alternative since they can be adjusted on a per-dimensions basis. Another strategy is to alter the real-world data sets by expanding their description with irrelevant attributes. In both cases though, some works consider that there is always some degree of bias introduced by the synthesising procedure and propose an adjustable real-world family of benchmarks suitable for testing scalability.

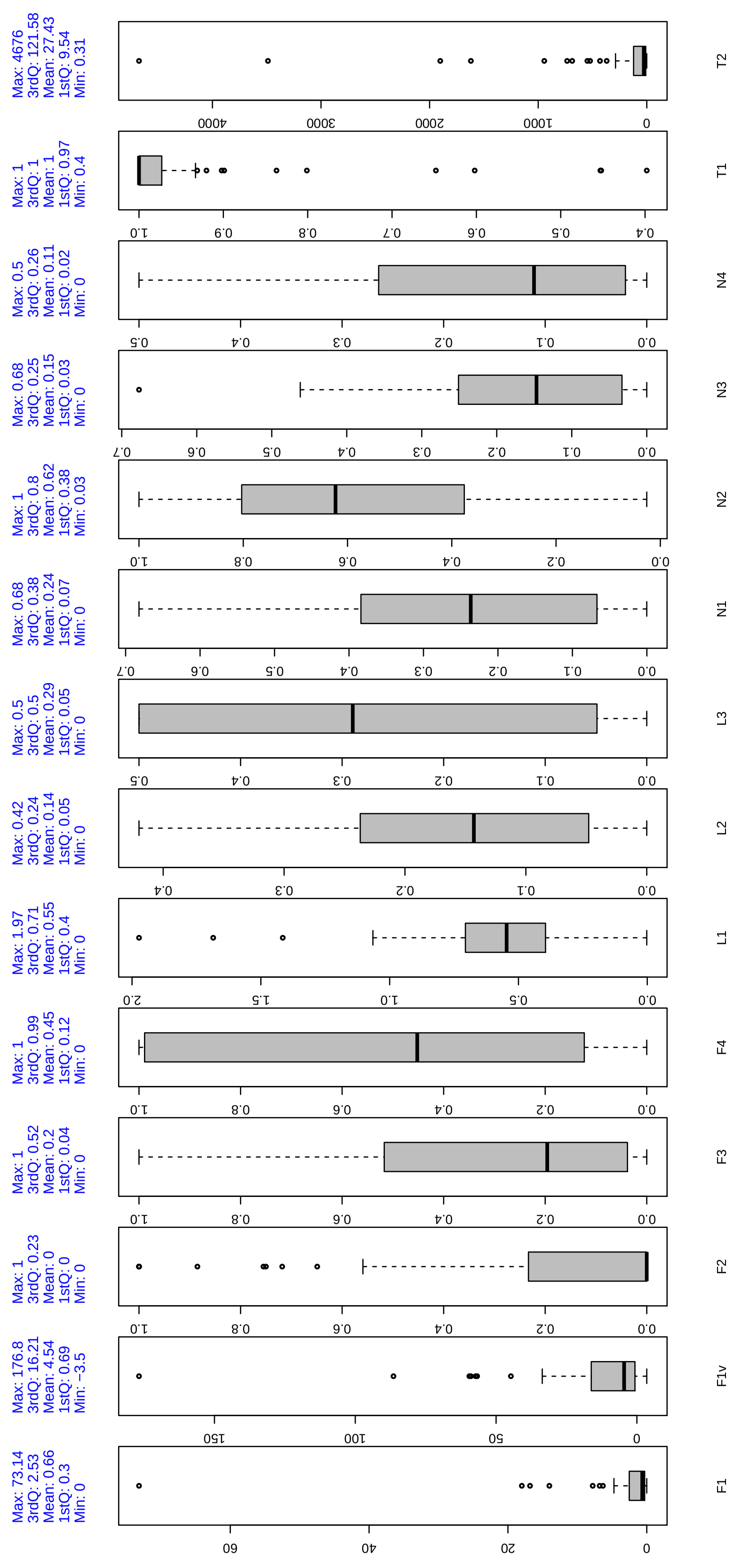


Figure 5.4: Distribution of the intrinsic characteristics of the UCI repository data sets. Note that F1 ranges in the interval [0,73.14]. In [García et al., 2009], the study was based on a shorter interval [0.035,2.670]. This may be construed as either partial results or the need of working with a common implementation.

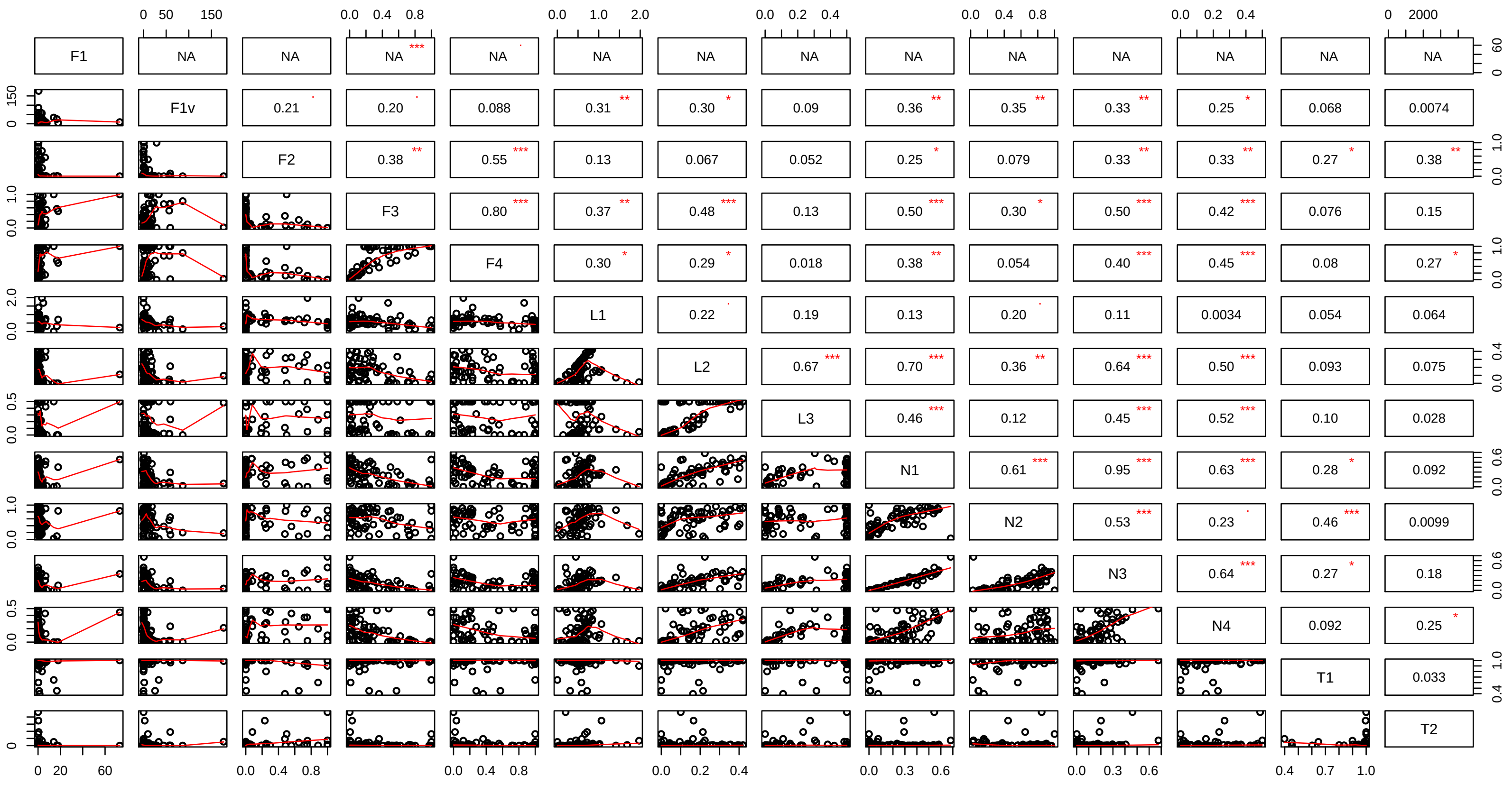


Figure 5.5: Correlation of the complexity measures.

5.4 REAL-WORLD PROBLEMS FROM BIOINFORMATICS

The analysis of bioinformatics problems is challenging due to the high dimensionality of the data sets. This section presents the Infobiotics Protein Structure Prediction (PSP) benchmarks repository [Stout et al., 2008; Bacardit and Krasnogor, 2008] which offers an arbitrarily adjustable collection of real-world problems. In the following, we describe the collection of data and present a detailed analysis of their complexity.

5.4.1 *Description*

PSP aims to predict the 3D structure of a protein based on its primary sequence, a chain of amino-acids represented by a string using a twenty-letter alphabet. Although PSP is an optimisation problem, each amino-acid can be characterised by several structural features which can be predicted as classification/regression problems. Thus, the Infobiotics PSP benchmarks repository provides 140 versions of the same prediction problem divided into 120 classification problems and 20 regressions problems—which constitutes 4.3GB of data in a compressed format. The features of the problem are predicted from the local context—a window of amino acids—of the target in the chain. For instance, a feature called coordination number for residue i is predicted using the information of itself and its two nearest neighbours in the chain sequence, i.e. $i-1$ and $i+1$. By generating different versions of the problem using different window sizes, they obtain a set of data sets with an increasing number of attributes and number of instances. Moreover, these structural features are defined as continuous variables which can be discretised at will and treated as classification problems with different number of classes. Two discretisation strategies were applied: (1) Uniform-Frequency (UF) discretisation, data sets with well-balanced class distribution and (2) Uniform-Length (UL) discretisation, data sets with uneven class distribution. Finally, there are two types of basic input information that have been used to create the data sets: (1) the Amino-Acids (AA) of the primary sequence directly and, where one amino acid is defined by one nominal variable which can take twenty possible values and (2) the Position-Specific Scoring Matrices (PSSM) representation derived from the primary sequence. That is, a statistical profile of the primary sequence and its evolution. Each amino acid is defined by twenty continuous variables.

In short, PSP provides us with a large variety of machine learning data sets, obtained from the problem of trying to predict the same protein structural feature with different formulations of inputs and outputs. The data sets ranges from 1 to 380 attributes of different kinds, i.e. discrete and continuous attributes, and their number of classes vary from three to five. In the repository, the data sets are partitioned into training/test sets for ten-fold cross-validation. However, we are interested in analysing the entire data set, and as such we treat the data as a whole, rather than considering it as separate training/testing sets.

Table 5.1 summarises the variations of the constructed repository.

5.4.2 *Intrinsic complexity*

Taking into account the dimensionality of the data sets and the computational cost of calculating the complexity measures using DCoL, we applied for computational resources to the *Centre de Supercomputació de Catalunya* (CESCA), which granted us with 20,000 hours of computation[4]. The equipment used was a SGI Altix UV 1000 composed of 1,344 nodes distributed in 224 Xeon (R) processors, 6.14 TB of memory, and 112 TB of disk space. It is worth noting that (1) these are shared resources and (2) DCoL does not implement parallel computing yet, so we do not utilise distributed processing. However, the high computational cost in terms of the time of the complexity measures motivates future work in redesigning the DCoL to harness the power of multi-processor computation in order to shorten the time required to obtain results. In summary, although we do not perform parallel computation, we need to run in parallel. Table 5.2 shows

4 http://www.cesca.es/novetats/2010/octubre.html#proves

Table 5.1: Description of the extrinsic characteristics of the PSP repository.

Data set	#Classes	#Instances	#Attributes
aa-uf/ul-w0	2, 3, 5	257,560	1
aa-uf/ul-w1	2, 3, 5	257,560	3
aa-uf/ul-w2	2, 3, 5	257,560	5
aa-uf/ul-w3	2, 3, 5	257,560	7
aa-uf/ul-w4	2, 3, 5	257,560	9
aa-uf/ul-w5	2, 3, 5	257,560	11
aa-uf/ul-w6	2, 3, 5	257,560	13
aa-uf/ul-w7	2, 3, 5	257,560	15
aa-uf/ul-w8	2, 3, 5	257,560	17
aa-uf/ul-w9	2, 3, 5	257,560	19
pssm-uf/ul-w0	2, 3, 5	257,560	20
pssm-uf/ul-w1	2, 3, 5	257,560	60
ppsm-uf/ul-w2	2, 3, 5	257,560	100
ppsm-uf/ul-w3	2, 3, 5	257,560	140
ppsm-uf/ul-w4	2, 3, 5	257,560	180
ppsm-uf/ul-w5	2, 3, 5	257,560	220
ppsm-uf/ul-w6	2, 3, 5	257,560	260
ppsm-uf/ul-w7	2, 3, 5	257,560	300
ppsm-uf/ul-w8	2, 3, 5	257,560	340
ppsm-uf/ul-w9	2, 3, 5	257,560	380

Table 5.2: Resources consumed by DCoL, run over data sets from the PSP repository.

Data set	CPU time (s)-(days)		Max memory (MB)	Max swap (MB)
pssm-uf-w0/w2/w1/w3	885,551.88		440	502
ppsm-uf-w4	492,535.91	- 5.7	558	620
ppsm-uf-w5	607,679.00	- 7.0	676	738
ppsm-uf-w6	702,376.31	- 8.1	793	856
ppsm-uf-w7	806,708.00	- 9.3	911	974
ppsm-uf-w8	919,463.00	- 10.6	1,029	1,092
ppsm-uf-w9	1,021,782.00	11.8	1,147	1,209
pssm-ul-w0/w2/w1/w3	870,544.00		440	502
ppsm-ul-w4	478,240.69	- 5.5	558	620
ppsm-ul-w5	583,413.19	- 6.7	676	738
ppsm-ul-w6	694,044.75	- 8.0	793	856
ppsm-ul-w7	799,213.50	- 9.3	911	974
ppsm-ul-w8	891,098.12	- 10.3	1,029	1,092
ppsm-ul-w9	994,176.62	- 11.5	1,147	1,209

the resource usage of some experiments presented in this section, especially of the experiments performed over the two-class problems. Both the CPU time and memory—maximum memory plus maximum swap memory—linearly increase with the number of attributes. The calculation for a single data set can take up to twelve days and require a maximum of 2.35 GB of memory. These numbers definitely justify the need of optimising the complexity measure algorithms and resorting to distributed computing, as proposed in Chapter 3, in order to gain efficiency. Furthermore, the high values of maximum swap indicate that there is a intensive memory access that should be fixed in the next iteration of the DCoL.

Table A.3 gives the results of the complexity computation. If we compare these values with the values obtained for the UCI repository (see Fig. 5.6), we observe that although real-world problems reach further minimum and maximum complexities and present larger variation, the complexity of the bio-synthesised problems is higher—there are more samples spread along the upper scale. Fig. 5.6 plots the difference between the UCI repository and the PSP repository represented in yellow and green respectively. For L2, L3, N1, N3, and N4, there are still some regions uncovered; in particular the following intervals $I_{L2} = [0.42, 0.599] \cup [0.976, 1]$, $I_{L3} = [0.511, 1]$, $I_{N1} = [0.682, 1]$, $I_{N3} = [0.47, 0.869] \cup [0.99, 1]$, and $I_{N4} = [0.5, 1]$ which define a moderate-high complexity in terms of separability.

We graphically represent the complexity measures with respect to the number of attributes since this PSP collection is designed to study scalability (see Fig. 5.7). Note the logarithmic scale of the x-axis. At a glance, both versions of AA—balanced and unbalanced—have the same curved shape and the two versions of PSSM also behave similarly. Thus, class balance (UF or UL) has little effect on the complexity computation; or, at least, complexity measures are not so sensitive to it in this case, where the imbalance ratio is 30-70%. Only L1, L2, L3, N1, and N3—class separability descriptors—estimate balanced data sets with higher complexity than the unbalanced one. Concerning AA problems, these measures point out possible lower/upper bounds in terms of linear separability. For AA-UL, L1, L2, and L3 collapse and get stuck at 0.59, 0.29, and 0.5 respectively, while AA-UF fluctuates slightly. This is comprehensively due to the fact that the SVM training is biased to the majority class. In particular, L1 has a difference of 0.37 between its minimum and maximum value. The correlation with the imbalance ratio should be studied more comprehensively to find some rules that allow us to determine or quantify the influence of such imbalance on the learners' error.

In the complexity analysis, measures such as T1 whose value is constant and Fs with a mixture of constants and low variability are not beneficial to spot behavioural differences among the data sets. Nevertheless, the value of these descriptors tells us several particularities of the problems. For instance, $F2 = 1$ indicates that AA problems present a noticeable overlapping among attributes and cannot discriminate between classes. The latter concept is reinforced by the zero values of F3 and F4. This tendency is reversed as many attributes are inserted into the data sets. Values of F2 for PSSM-UF problems drop from 0.877 to 0.197 and for PSSM-UL from 0.944 to 0.098. Unfortunately, although the overlapping is diluted for the high number of dimensions, values close to zero for F3 and F4 indicate that the discriminative power of attributes is still weak.

Another significant piece of information is given by N1, which describes the number of points near the class boundary. This estimate is affected by the dimensionality of the problem and exemplifies the dilemma about how many attributes have to be included in the definition of a problem. We see that for AA problems, by adding more attributes to the data set, the complexity of the class boundary increases. Indeed, few attributes usually give way to simple boundaries. Thus, it is important, then, to complement that estimate, and differentiate between simplicity and lack of representativeness. From twenty attributes upwards, i.e. the PSSM problems, the more attributes the data sets have, the less complicated the class boundary becomes. This means that the added attributes contribute to better define the learnable concept. However, these problems present values greater than 0.4, understood as a complex boundary. This pattern is also visible in N3, which seems highly correlated to N1 as shown in the previous section.

The upshot is that the AA problems have more complex class boundaries than PSSM problems and deriving the primary sequence provides a better description of the problem. In fact, from the complexity point of view, there is no much difference between the data sets generated with a uniform frequency and uniform length. Although adding more than 200 attributes does not seem to provide more information[5], such a description suffers from high dimensionality.

To overcome scalability limitations and deal with such large data sets, Bacardit and Krasnogor [2006] designed and implemented the Bioinformatics-oriented Hierarchical Evolutionary Learning (BioHEL). This new technique presents promising results on these kinds of problems. However, in a naive experimentation, detailed in the next section, we observe that BioHEL has a poor response when faced with data sets generated with the Checkerboard problem.

5 To confirm the stability of the complexity for greater than 200 attributes, one should ideally analyse at least one more order of magnitude, where possible.

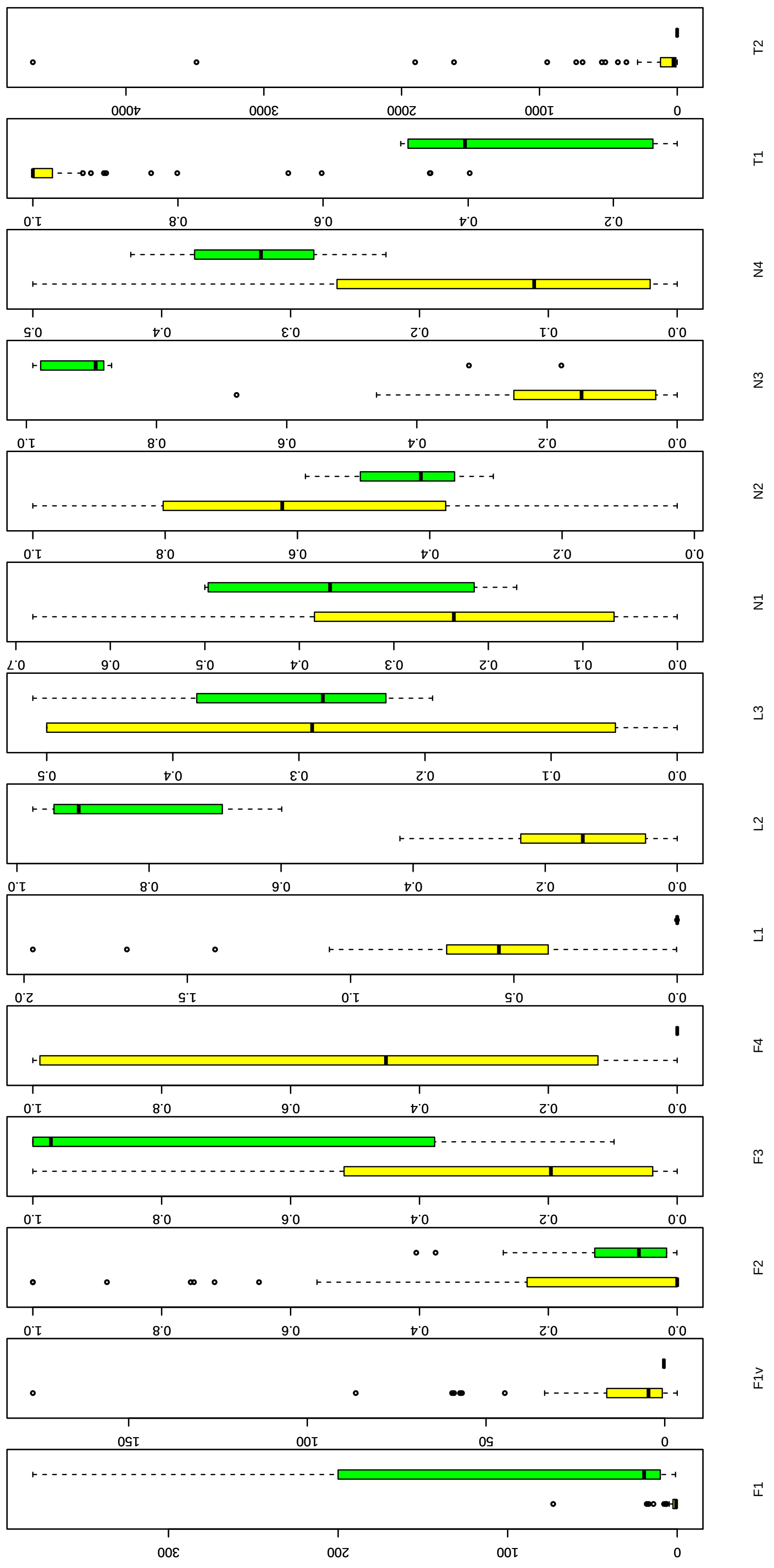


Figure 5.6: Comparison between the UCI repository (yellow boxplot) and the PSP repository (green boxplot).

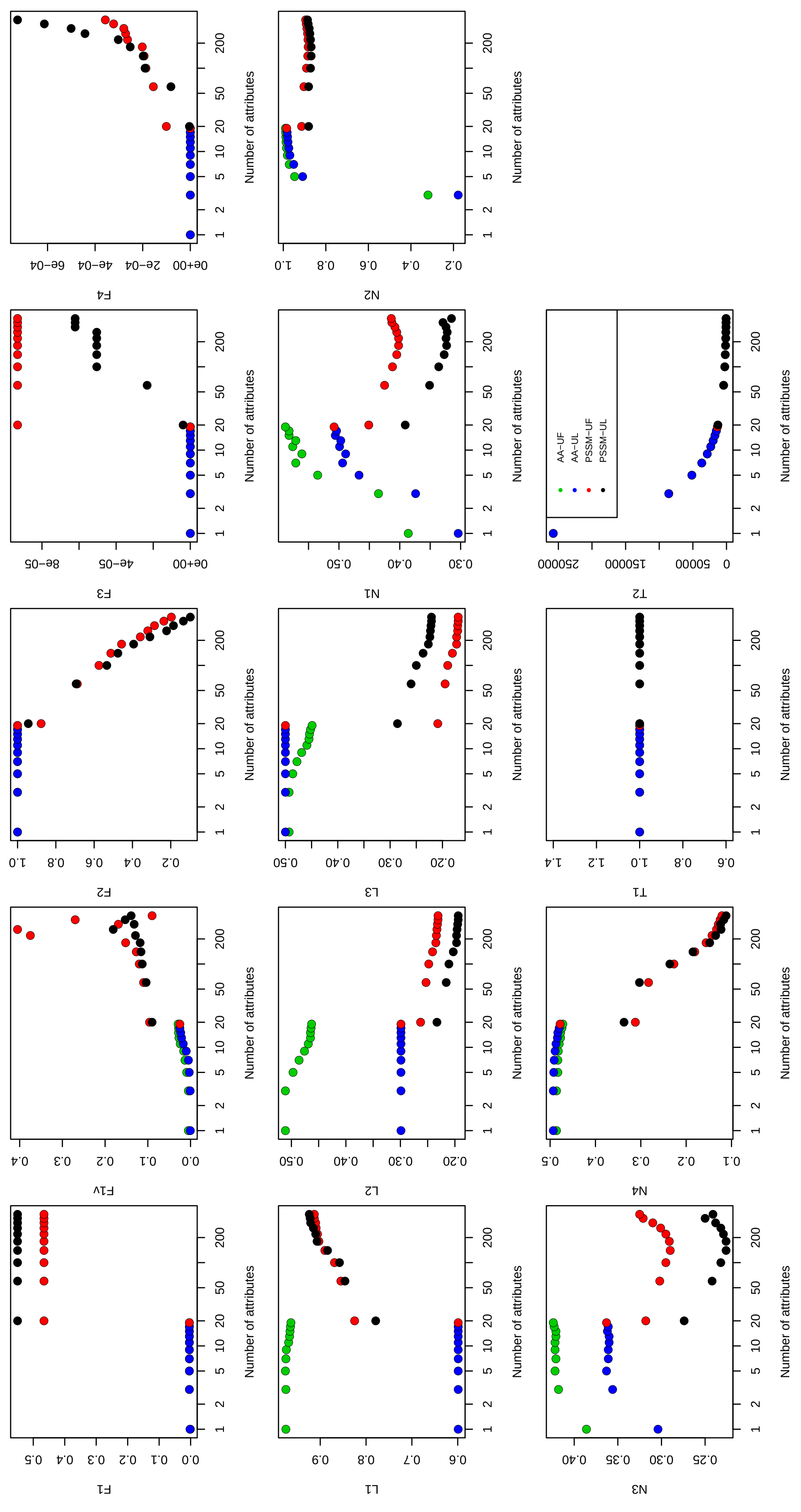


Figure 5.7: Graphical representation of the complexity measure with respect to the inclusion of attributes.

Table 5.3: BioHEL and GAssist baseline performances: percentage of correctly classified instances

Data set	BioHEL	GAssist
Checkerboard	0.498866	0.974782
Spiral	0.921920	0.933520
Wave Boundary	0.701240	0.789280
Yin Yang	0.884894	0.898436
Pima	0.719324	0.737936

5.5 CHECKERBOARD

Synthetic problems are very welcome to probe particular behaviours in specific environments. In fact, some have become benchmarks in certain knowledge areas (e.g. learning Boolean functions such as the bit multiplexer). Other than a minority, synthetic problems are homemade testing data sets that remain undisseminated—just for personal, private testing—and with a very small scope—generally used for only one project—, whereas they could be broadcasted to a wider audience in order to better the scientific community. This section explores the classical synthetic problem Checkerboard and its potential to study learners' limitations as well as complexity measures. First, we describe the generation process of this new collection focused on the Checkerboard problem and we then utilise it to study learners' performance and complexity.

5.5.1 *Description*

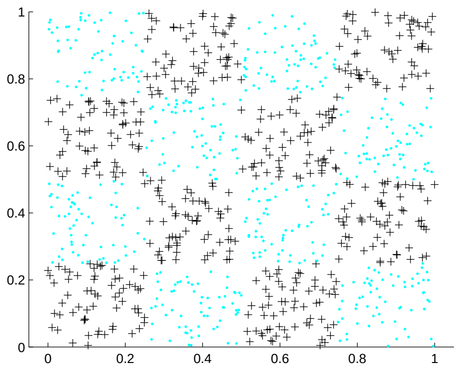
The checkerboard layout, under a guise of simplicity, causes—in its class balance version—trouble for many supervised learners. Yet its two dimensions concede the possibility of easily visualising the shape and opting for the right parametrisation of the learner. We decide to mask the layout by adding irrelevant attributes and quiz several learners.

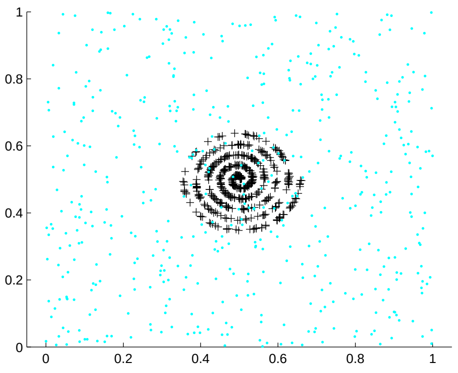
As a first approach, we add only eighteen irrelevant attributes following a Gaussian distribution, nine of them with a mean of 0.6 and the other nine with a mean of 0.4. The process is repeated for four other seed problems representing: a spiral layout, a wave-shaped boundary, the Yin-Yang, and a real-world problem which belongs to the UCI repository (Pima Indians Diabetes). Thus, the four synthetic problems are data sets with a known concept (see Fig. 5.8).

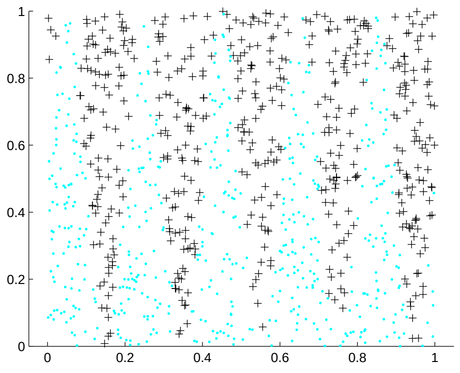
1. *Checkerboard.* A classical non-linear problem with heavily interleaved classes following a checkerboard layout.
2. *Spiral.* A problem with a non-linear class boundary following a spiral layout.
3. *Wave Boundary.* A linearly separable problem defined by a sinusoidal function.
4. *Yin Yang.* A linearly problem with small disjuncts.

Table 5.3 shows the baseline performance of BioHEL and GAssist Bacardit [2007]—two learning techniques used on bioinformatics data sets—for the aforementioned five seed problems. Without any previous parameter adjustment, BioHEL gives a rough accuracy of 0.49. Such a low value draws attention to the Checkerboard layout.

After this first approach, we proceed to generate a new collection of problems focused on the Checkerboard by varying the number of irrelevant attributes and controlling its distributions in order to produce more conclusive results. Table 5.4 details the characteristics manipulated and their values. All possible combinations—twenty times each—form a collection of 51,840 problems.


(a) Checkerboard


(b) Spiral


(c) Wave Boundary

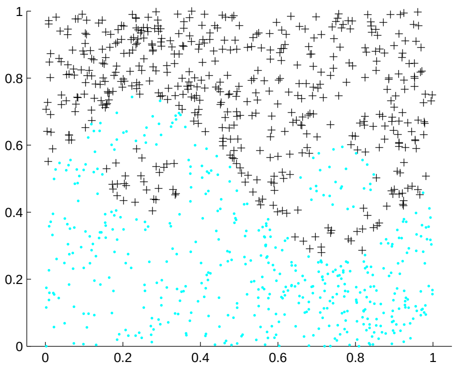

(d) Yin Yang

Figure 5.8: Seed data sets with known structure.

Table 5.4: Description of the extrinsic characteristics of the Checkerboard problem.

Characteristic	**Values**
Number of instances	1,000 – 5,000 – 10,000
Number of attributes (Relevant+Irrelevant)	2+2 – 2+8 – 2+14 – 2+18
Number of squares	16 (4 × 4) – 36 (6 × 6) – 64 (8 × 8)
Distribution of irrelevant attributes	Gaussian – uniform
Standard deviation	From 0.1 to 0.8 in steps of 0.1
Mean	0.25 – 0.50 – 0.75

5.5.2 *Intrinsic complexity*

In the following, we provide some complexity analysis of the simple version of the Checkerboard and one with irrelevant attributes added.

Regarding the 2-dimensional version of this problem, we vary the standard deviation and the number of clusters (see Fig. 5.9). We notice that the complexity of the class boundary is not reliant on the increment of squares but on the standard deviation used to generate the instances. In Fig. 5.9a, the x-axis represents the number of clusters (squares of the checkerboard) and the y-axis represents the fraction of points on the class boundary (N1). Moreover, we add a third dimension that shows the evolution of the ratio of the standard deviation. Note that for any configuration concerning the number of clusters, the greater the standard deviation, the more *complex* the length of the class boundary. This means that the complexity measure is related to the overlapping of classes and is useful to estimate it. This is made clearer in Fig. 5.9b where we plot the estimate of complexity according to the ratio of the standard deviation.

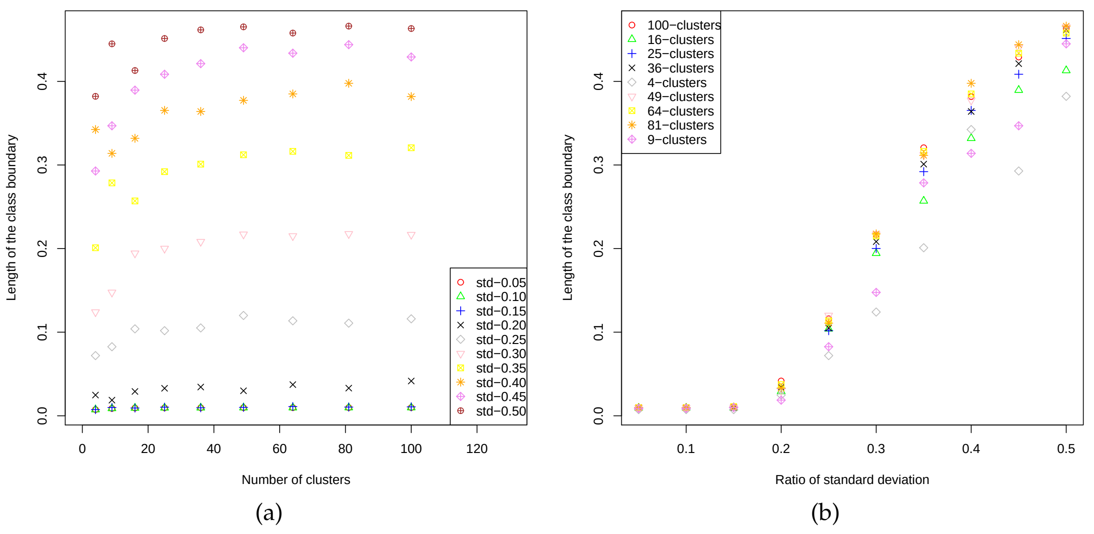


Figure 5.9: Standard deviation according to the length of the class boundary

Regarding the version of the problem with irrelevant attributes, we experiment with feature selection methods, learners, and complexity measures.

Feature selection. First of all, we try four feature selection methods—*InfoGainAttribute*, *OneR-AttributeEval*, *PrincipalComponent*, and *ReliefFAttribute* from Weka [Witten and Frank, 2005]—to check whether these specialised tools are able to recognise the padding. Note that the additional attributes are not noise—we consider noise as those instances that are mislabelled—, just unrelated information that hides the learning concept. When run with their default configuration, all the methods keep the whole set of features to proceed to the classification. The irrelevant information encoded by Gaussian and uniform distribution is transparent to them. However, *OneRAttributeEval* is, in a few cases, able to identify—good ranking—which two dimensions encoded the learning concept. *ReliefFAttribute* is also able to identify this, but only when the irrelevant attributes are generated with a Gaussian distribution.

Learners. By running different classifiers, i.e. C4.5, IBk ($k = 3$), NB, PART, RT, and SMO, over the whole collection, we can see how none of them learn the concept either. Having a look at the output files, decision tree generation methods do not manage to build a tree, just a single leaf which achieves 50% accuracy. Only IB3 and RT stand out from all the other classifiers with a modest 56.48% and a maximum of 70.18% classification accuracy, respectively (see Fig. 5.10).

Complexity measures. Table A.4 gathers the complexity of the Checkerboard collection. If we apply the meta-learning idea presented in the previous chapter, we ascertain that L1 is able to identify when the irrelevant attributes have been generated with Uniform or Gaussian distributions (see Fig. 5.11). The cutoff point is at 0.66; above this number the irrelevant attributes are pure noise—random—and below, follow a Gaussian distribution. The Gaussian distribution randomly generated are grouped together with the Gaussian generated with specific standard deviations.

If we run a clustering technique instead—SimpleKMeans parameterised with the default values of Weka—, we obtain the output reported in Fig. 5.12. This experiment is meant to detect the two clusters—data sets with irrelevant attributes generated (1) with a Gaussian distribution or (2) with a uniform distribution—based on the complexity measures. The values of the F measures, relating to the discriminative power of the attributes, seem to be the more important one with large differences between one cluster and the other:

- $\{F1_{Gaussian} = 0.0017;\ F1_{Random} = 0.0096\}$
- $\{F1v_{Gaussian} = 0.0108;\ F1v_{Random} = 0.0706\}$

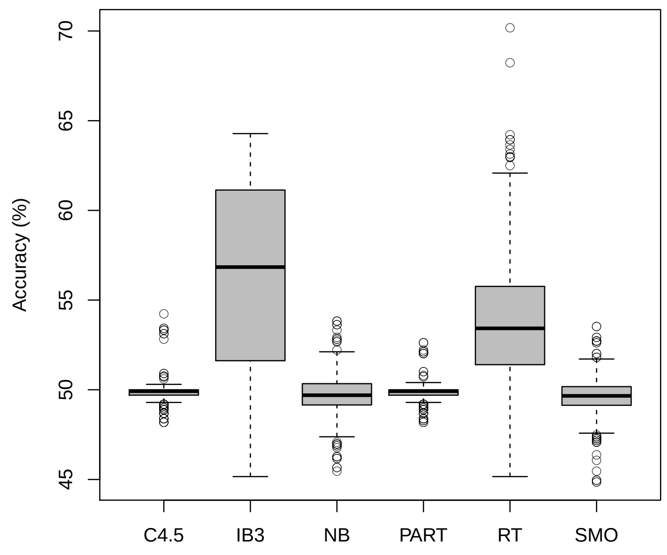


Figure 5.10: Percentage of correctly classified instances of the Checkerboard collection by C4.5, IB3, NB, PART, RT, and SMO.

L1 $\leqslant$ 0.6606: Gaussian (18.0/3.0)
L1 $>$ 0.6606: Random (3.0)

Figure 5.11: Tree built by C4.5 over the Checkerboard of 16 squares.

- $\{F3_{Gaussian} = 0.0014; F3_{Random} = 0.0092\}$
- $\{F4_{Gaussian} = 0.0121; F4_{Random} = 0.0865\}$

As soon as we study the problems with more irrelevant attributes the tree changes; F2 and N2 become the decisive complexity measures (see Fig. 5.13). Output 0 refers to the data sets generated with a Gaussian distribution, and output 1 refers to the data sets generated with a uniform distribution. Actually, we only need to consider F2 which is the single measure that splits the instances into two groups.

For the classical layout, the complexity measures describe the problem as an interleaved problem—values of F2 high—but with well-defined boundaries—values of N1 and N2 low. When we add attributes with a random distribution, the complexity measures maintain the interleaving trait whereas the difficulty of the boundary increases. For irrelevant attributes following Gaussian distributions, the data analysis highlights that values of F2 are lower and then there is some kind of noise that can be distinguished from the original problem.

The number of attributes plus the number of instances plus the complexity measures N1 and N2 maybe help to define the interval of the rules for classifier like BioHEL.

These results reflect that the meta-learner is conditioned by the definition of the problem and we will need representative problems and reference learners to extract a stable set of guidelines to build a recommender system.

5.6 TOWARDS A NEW BREED OF PROBLEMS

In many applications researchers pursue perfection of classification techniques since there are obvious benefits to obtain the maximum accuracy, e.g. in performing medical diagnosis. However, there have been relatively few systematic studies on whether perfect classification is possible in a specific problem. Most attention has been devoted to fine tuning the techniques instead. Problems that provide a good coverage of the data complexity space would be necessary to perform this kind of study.

```
kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 16.49132195903528
Missing values globally replaced with mean/mode

Cluster centroids:
                              Cluster#
Attribute           Full Data          0          1
                         (21)       (14)        (7)
==================================================
numberOfInstances   5333.3333       7500       1000
F1                     0.0043     0.0017     0.0096
F1v                    0.0307     0.0108     0.0706
F2                     0.2644     0.2931      0.207
F3                      0.004     0.0014     0.0092
F4                     0.0369     0.0121     0.0865
L1                     0.6554     0.6528     0.6604
L2                     0.9983     0.9974     1.0002
L3                     0.4937     0.4906     0.4999
N1                     0.3065      0.342     0.2354
N2                     0.9802     0.9837     0.9731
N3                     0.4756     0.4795     0.4679
N4                     0.4713     0.4758     0.4622
T1                          1          1          1
T2                      265.2      373.5       48.6
class                Gaussian   Gaussian   Gaussian


Clustered Instances

0      14 ( 67%)
1       7 ( 33%)
```

Figure 5.12: Weka output for the command weka.clusterers.EM.

F2 $\leqslant$ 0.908
| N2 $\leqslant$ 0.958: 1 (2.0)
| N2 $>$ 0.958: 0 (25.0)
F2 $>$ 0.908: 1 (46.0)

Figure 5.13: Tree built by C4.5.

By comparing the UCI and the PSP repositories, we pinpoint gaps in the intervals of the complexity measures. By analysing the Checkerboard problem, we realise how challenging synthetic problems are. These two observations lead us to generate a new breed of problems by combining both real-world problems and artificial ones—problems that fill the gaps in the measurement space and permit the testing of the limitations of learners.

Contribution.

1. Characterisation of the intrinsic complexity of the UCI and PSP repository.
2. Generation of a data set collection based on the Checkerboard problem.

6 GENERATION OF ARTIFICIAL DATA SETS

Summary. This chapter proposes an approach to generate artificial data sets. The quest for artificial problems with bounded complexity pursues a better coverage of the complexity space which should offer a more complete framework to study (1) complexity estimates and (2) learners' behaviour. Our proposal uses an evolutionary computation search mechanism guided by the complexity measures; an alternative mechanism to enhance the most popular repository in machine learning—the UCI. A second approach that varies the properties of specific layouts is outlined.

6.1 INTRODUCTION

Empirical results have shown that certain learning techniques are more suitable than others to solve certain types of problems. Nevertheless, currently available real-world problems and popular synthetic data sets do not cover the whole complexity space as seen in Chapter 5 and, therefore, do not allow us to either (1) properly identify the domain of competence of learners or (2) thoroughly test learners' behaviour at the edge of their domain of competence. Thus, the necessity for developing a more suitable testing scenario arises—a controlled scenario with specific data of known complexity. With this in mind, data complexity analysis has shown promise in characterising the difficulty of classification problems through a set of complexity descriptors which, used in the generation of Artificial Data Sets (ADS), could supply the required framework with the sufficient knowledge to refine and design learners as well as the methodology for assessment and comparison.

The purpose of this chapter is to formulate the complexity measurement space and present new approaches to generate ADS to fill the holes of such space.

In the following, we revisit previous studies as a starting point for the analysis of the universe of problems, and the further proposal of its projection on a complexity measurement space. We first consider the elements involved in the characterisation of the problem complexity and define the properties of the measurement space. After juggling with different options, we suggest the use of instance selection based on an evolutionary multi-objective optimisation technique to generate data sets that meet specific characteristics established by the complexity descriptors proposed by Ho and Basu [2002]. Through a set of experiments, we show the advantages and disadvantages of the proposed method. In addition, we outline a second approach based on the perturbation of specific layouts.

6.2 PROBLEMS UNIVERSE AND ITS PROJECTION ON A COMPLEXITY MEASUREMENT SPACE

The study of the universe of problems is closer to a lost cause due to the infinity of both the space and its dimensions. Thus, we need to bound this space in order to analyse the nature of data and provide a framework to test learners' performance. This section defines a measurement space based on data complexity inspired by previous works. In the following, we revise what is done and we announce what is coming.

6.2.1 *What is done*

Some works have tried to characterise problems' complexity by using different complexity descriptors [Ho et al., 2006]. These measures give an apparent complexity estimation—apparent since the measure is computed over a sample of the problem—by taking into account different

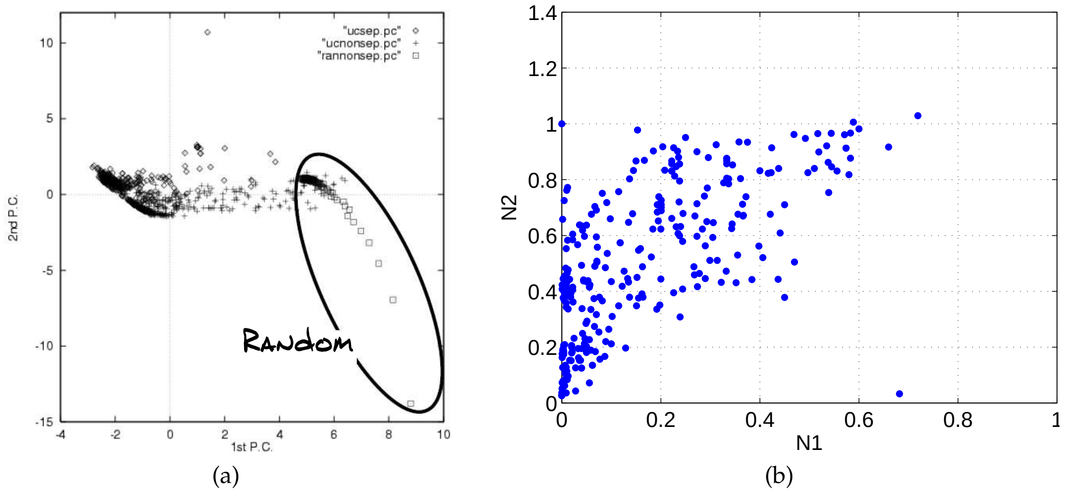


Figure 6.1: Projection of (a) real-world and artificial problems onto the two principal components of the complexity measures and (b) real-world problems onto two complexity measures, N1 and N2.

geometrical aspects such as the discriminative power of features, the class separability, and the manifold characteristics. Basu and Ho [2006] proposed a measurement space built upon the two principal components of the measures on which a set of problems—real-world problems from the UCI repository and problems synthetically generated following a random distribution—were projected.

Fig. 6.1 represents the aforementioned measurement space, where they found out a continuum which differentiates between problems with a learnable structure, i.e. real-world problems, and random problems. Nevertheless, some blind spots appear in this characterisation, what fosters the refinement of such a space.

Fig. 6.1b also reveals sharp discontinuities. This space was built upon only two complexity measures: the fraction of points on the class boundary (N1) and the ratio of the average intra/inter class nearest neighbours distance (N2) (for further details about these complexity measures, see Chapter 3). The case study involved an enhanced test set containing 264 problems, which were obtained from 54 problems from the UCI repository. Multiple class data sets were split into binary class problems by discriminating each class against the rest. Despite the broad experimentation, the resultant measurement space presents many empty areas, i.e. regions that are not covered by any data set. For example, in the test bed there is no data set whose N1 value is greater than 0.75. These areas may be due to (1) the test bed, which may be composed of problems that do not spread across all complexities or (2) the fact that no real-world problem has these characteristics. Both are plausible explanations and, at the same time, both support the necessity of setting up common benchmarks. In the first case, we cannot comb all repositories looking for real-world problems that have the characteristics we need. In the second case, we cannot assume that all gathered real-world problems form a representative sample of the real-world problem space. Besides, we ignore whether in the near future a new problem with a structure never seen before will appear.

6.2.2 *What is coming*

We aim at achieving a frame of mind similar to the one proposed by Ho et al. [2006], but without focusing on differentiating between learnable and random structures. Instead, we direct our efforts toward the synthetic generation of any kind data sets to cover the blind spots of the complexity space. This should help us to gain more insight into the universe of problems and answer the following questions:

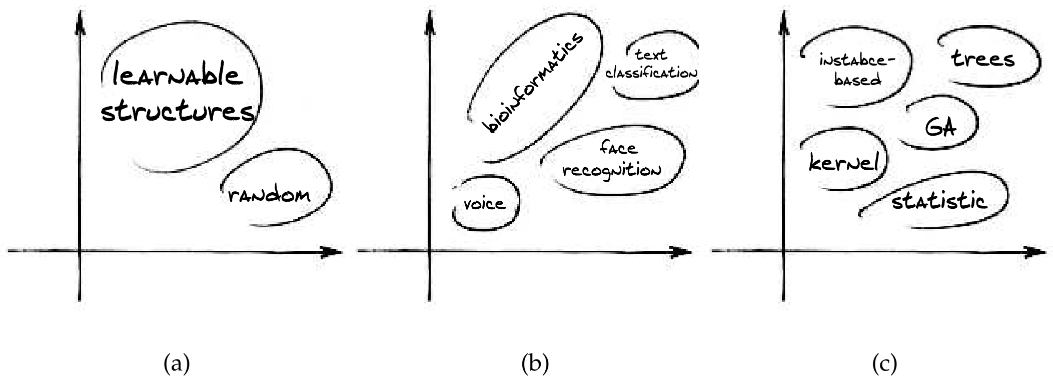


Figure 6.2: Different measurement spaces: (a) continuum of learnable structures, (b) taxonomy of problems, and (c) recommender system.

1. Do the empty regions mean that this kind of problem does not occur in nature?
2. Is the proposed set of complexity measures enough to capture the problem complexity?
3. Can we shrink the set of measures to characterise the problem complexity while maintaining the same accuracy estimating the difficulty?

With this aim, we consider that any measurement space should meet the following properties: (1) completeness, (2) resolution, and (3) representativeness.

Completeness. The complexity spectrum has to be covered.

Resolution. The dimensions used to build the space should provide sufficient granularity to reveal differences among problems and even group together problems from the same knowledge areas.

Representativeness. Real-world problems—or any other kind of problem—have to be located in the space.

With a *complete* problem space in mind, we envisage three different uses of the measurement space: (1) determine whether a problem contains learnable structures or not (see Fig. 6.2a), (2) identify the typology of the problem through its complexity (see Fig. 6.2b), and (3) recommend the suitable learner to solve a specific problem (see Fig. 6.2c).

Furthermore, if we are able to cluster the problems according to their complexity, we may work on finding the transformations to modify the complexity of the problem and mapping them onto the domain of competence of the desired learner.

6.3 WHAT KIND OF, AND HOW TO GENERATE ADS

One of the great advantages of studying data complexity is that it should help us to make the bridge from real-world problems to synthetic problems. This section draws and discusses ADS generation. We present a general picture of this incipient topic by answering what kind of characteristics ADS should have and how to provide them.

6.3.1 *What kind?*

Being aware of the need for artificial data sets, we have to define what kind of data set should be generated. To this end, we focus on classification and take into consideration (1) underlying knowledge and (2) complexity factors.

Underlying knowledge present in real-world problems is significant in data analysis since it contains the learnable concept. Thus, we should force data sets to resemble real-world problems and attain such real structures. It means that data not only should to follow uniform or Gaussian distributions but also can include physical processes. Moreover, for classification problems, class labelling could conform with clustering rules.

Complexity factors are related to the aforementioned aspects that are measured by the complexity measures, such as the discriminative power of attributes, class separability, and geometry.

Firstly, we have to generate well-defined problems with a known underlying concept and whose definition is complete and without ambiguity. After defining which complexity characteristics describe data, constraints have to be introduced by varying their degree of difficulty. This implies relating difficulty factors to the type of performance we want to assess, such as robustness, scalability, and predictive accuracy. For instance, noise, missing values, or ambiguity, as well as irrelevant or redundant attributes, are suitable characteristics to test the learner robustness. Learner scalability can be tested by varying the number of attributes and the number of instances. Determining the number of classes of the problem adds another layer of difficulty, since some of the complexity factors have to be interpreted differently.

When creating data sets, these two main considerations refer to the following internal phases:

1. Generation of the distribution of points.
2. Perturbation of the data set previously obtained to attain specific complexity constraints in the wider sense.

Indeed, for real-world problems, the first step is given.

Altogether, this leads to set (1) extrinsic and (2) intrinsic characteristics. Both may be divided into two sub-categories.

Extrinsic characteristic should define the external properties directly measurable—such as number of instances, number of attributes, and type of attributes—and structural anomalies—such as labelling noise, missing values, and class imbalances. In addition, there are two extra characteristics only for synthetic problems to configure the choice of additional attributes. They specify the relevance of the attributes and the distribution followed—for instance, Gaussian or uniform.

Intrinsic characteristics should distinguish between class-dependent and non-class-dependent measures. That is, the class-dependent measures, such as those that estimate the value of the class boundary, need to compute information taking into account, for instance, the number of classes or the distances between them. The non-class-dependent measures do not base the computation on the classes in the problem, e.g. T1, which is a ratio between the number of instances and the number of attributes.

Table 6.1 shows our proposal of meta-parameters, corresponding to a complete description of a given problem. Generators of artificial data sets have to permit tuning of all of these characteristics to test learner efficiency in particular cases and comprehend learner behaviour when dealing with specific constraints.

Table 6.2 shows an example of a typical experimentation framework, where the accuracies of several classifiers are compared on a set of problems extracted from the UCI. Each problem is characterized only by extinsic characteritics, the number of attributes and the number of instances. These two dimensions are frequently misunderstood as indicators of the problem complexity. Often one wrongly assumes that the higher the dimensionality, the higher the complexity.

Figures 6.3a and 6.3b depict the relation between the accuracy of several classifiers—an induction tree (C4.5) [Quinlan, 1995], an instance based learning (IBk, $k = 3$) [Aha et al., 1991], a rule learner (PART) [Witten and Frank, 2005], and SMO [Platt, 1999]—and the number of instances and number of attributes of the data sets respectively. From these plots, we cannot observe any pattern nor any sort of correlation between the classifiers' accuracy and these data set characteristics.

Certainly, these dimensions provide an estimation of the data volume and can have some relationship with data sparsity. However, there are other complexities hidden in the data sets

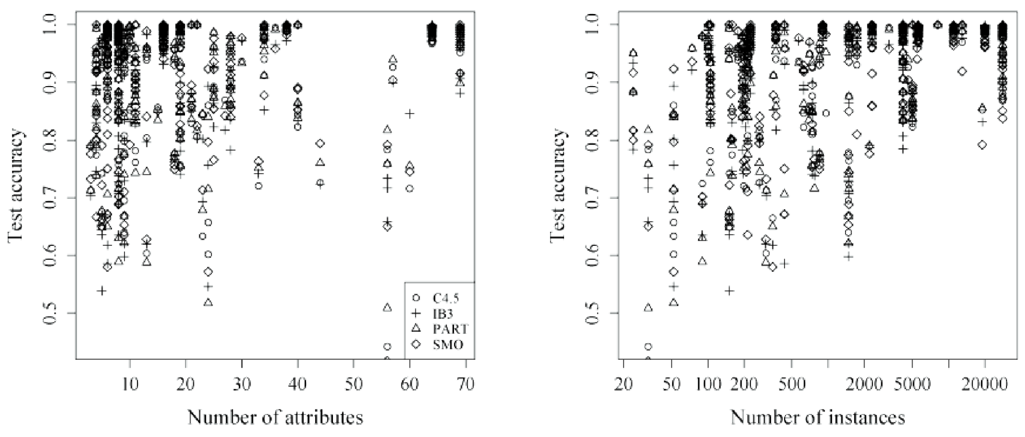
Table 6.1: Taxonomy of meta-parameters grouped into two main categories: (1) extrinsic characteristics θ_{1-8} and (2) intrinsic characteristics θ_M.

Label	Meta-parameters	Additional information
θ_1	Number of instances	Also called examples or points
θ_2	Number of attributes	Also called features, variables, or dimensions
θ_3	Labelling noise	Erroneous labelling or outliers
θ_4	Missing values	Unknown values
θ_5	Class imbalance	Presence of a majority and minority class
θ_6	Type of attributes	Continuous or nominal
θ_7	Relevance of attributes	
θ_8	Data distribution	
θ_M	Complexity measures	Measures that depend on class labelling Measures that do not depend on class labelling

Table 6.2: Classical experimental framework. #Attr is the number of attributes, and #Inst is the number of instances.

Data set	#Attr	#Inst	C4.5	IB3	PART	SMO.
Abalone	8	4177	0.9998	0.9998	0.9998	0.9998
Balance Scale	4	625	0.8467	0.8866	0.8625	0.9297
Breast Cancer Wisconsin	30	569	0.9367	0.9719	0.9332	0.9771
Chess (King-Rook vs. King)	6	28056	0.9791	0.9010	0.9975	0.9003
Glass Identification	9	214	0.8071	0.8355	0.8407	0.7104
Heart Disease	13	303	0.8037	0.7963	0.7444	0.8407
Hepatitis	19	155	0.8008	0.7996	0.8020	0.8804
Ionosphere	34	351	0.9120	0.8518	0.9118	0.8776
Iris	4	150	0.9933	1	0.9933	1
Lenses	4	24	0.8167	0.9333	0.8833	0.8000
Letter Recognition	16	20000	0.9960	0.9994	0.9965	0.9916
Lung Cancer	56	32	0.7583	0.7333	0.7583	0.6500
Optical Recognition	64	5620	0.9939	0.9996	0.9957	0.9977
Pima Indians Diabetes	8	768	0.7566	0.7382	0.7161	0.7669
Statlog (Image Segmentation)	19	2310	0.9939	0.9952	0.9931	0.9965
Statlog (Vehicle Silhouettes)	18	846	0.7695	0.7731	0.7766	0.7494
Thyroid Disease	5	215	0.9344	0.9485	0.9299	0.7907
Waveform Database Generator	21	5000	0.8290	0.8496	0.8360	0.8588
Wine	13	178	0.9604	0.9833	0.9604	0.9889
Yeast	8	1484	0.7223	0.7069	0.7156	0.6880

that may be more influential. Hence, the importance of previously defining all the type of complexities and studying the influence of each meta-parameter on the performance of learners with respect to the other meta-parameters.



(a) (b)

Figure 6.3: Accuracy of classifiers with respect to (a) number of attributes and (b) number of instances in logarithmic scale

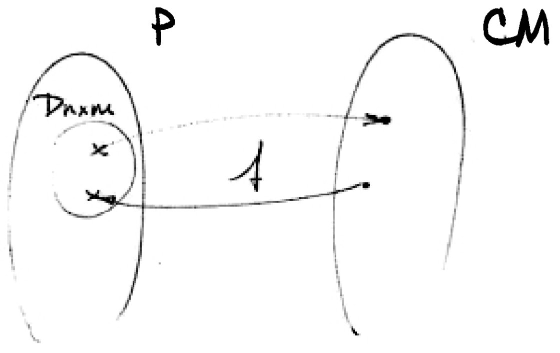


Figure 6.4: Vector spaces for the linear transformation.

6.3.2 *How?*

Formally, we assume there are two vector spaces (see Fig. 6.4): (1) the problem space $\mathcal{P}$ which contains all possible data sets—represented by matrices of dimension $n \times m$ where n is the number of instances and m is the number of attributes that describe the problem—and (2) the complexity space $\mathcal{CM}$, along this work defined as a fourteen-dimensional space, where each dimension represents a complexity measure [Ho and Basu, 2002; Orriols-Puig et al., 2010]—$\mathcal{CM} =< F1v, F1, F2, F3, F4, L1, L2, L3, N1, N2, N3, N4, T1, T2 >$. This assumption allows us to (1) represent the problem algebraically without losing information, (2) use known spaces without formally proving their vectorial nature (operations, neutral elements, etc.) and (3) focus on the definition of the transformation instead of the algebraic definition of the spaces.

To generate ADS with a specific complexity, we should design an analytical method that enables us to move from one space to the other in both directions. Simplicity of formulae urges us to adopt some linear algebra tools to achieve such a purpose.

Definition 2. *Given two vector spaces $\mathcal{P}$ and $\mathcal{CM}$ over the same field $\mathbb{K}$ and a mapping transformation $f : \mathcal{P} \to \mathcal{CM}$, such transformation is linear if it satisfies that*

$$\forall \mathcal{D}_1, \mathcal{D}_2 \in \mathcal{P}, \forall \lambda, \mu \in \mathbb{K}, f(\lambda \mathcal{D}_1 + \mu \mathcal{D}_2) = \lambda f(\mathcal{D}_1) + \mu f(\mathcal{D}_2). \quad (6.1)$$

From Eq. 6.1, we deduce the following properties for the linear transformation $f : \mathcal{P} \to \mathcal{CM}$.

1. $f(\mathbf{0}_{\mathcal{P}}) = \mathbf{0}_{\mathcal{CM}}$
2. $\forall \mathcal{D} \in \mathcal{P}, f(-\mathcal{D}) = -f(\mathcal{D})$
3. $\forall \mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n \in \mathcal{P}, \forall \lambda_1, \lambda_2, ..., \lambda_n \in \mathbb{K}, f(\sum_{i=0}^{n} \lambda_i \mathcal{D}_i) = \sum_{i=0}^{n} f(\lambda_i \mathcal{D}_i)$

Unfortunately, the computation of the complexity measures does not meet the first condition. The neutral element of $\mathcal{P}$ does not correspond to the neutral element of $\mathcal{CM}$, which is a necessary condition—e.g. given a data set $\mathcal{D}$ described by thirty instances and three attributes whose values are zero, the complexity vector in its fourteenth dimension, i.e. the complexity measure T2, is ten.

In order to satisfy such a property, we should (1) modify the computation of the complexity measures to obtain linear operations or (2) define a set of independent measures with no correlation among them. Both strategies mean redefining and/or removing some complexity measures from the descriptive set. However, even if we achieve the properties of a linear function, the next step would be to build an orthonormal basis. The size of the basis increases rapidly as a function of the dimensionality of the data sets—the number of attributes and number of instances—, which fixes the number and size of the matrices. Therefore, if we have a data set $\mathcal{D}_{n \times m}$, where m is the number of attributes and n is the number of instances, the basis will be composed of $n \times m$ matrices of size m columns and n rows. This entails an infeasibly high computational cost and indicates that any kind of transformation is strongly conditioned by the dimensionality of the elements of the problem space. As a result, we settle on what can be done with evolutionary computation, although the mapping suggested previously will be less exact.

To generate data sets which meet different complexity levels for different difficulty factors and whose structure resemble real-world problems, we proceed as follows.

Considering the complexity factors, we address the problem as an optimisation problem in which each objective to minimise or maximise becomes a complexity measure. In this regard, Evolutionary Multi-objective Optimisation (EMO) algorithms [Coello et al., 2006] are a natural methodology to conduct optimisation problems with several objectives.

Concerning the underlying structure of the problems, we achieve realistic concepts by generating the initial data set according to fixed distributions. Correlations among features can be set by users taking into account statistics extracted from real data [Jeske et al., 2005]. Existing samples of real-world problems combined with learning techniques, such as instance selection and feature selection, are alternatives to dynamically managing distributions. Regarding the class labelling, some instance classes can be fixed or grouped following real-world problems' labelling.

6.4 GODS: GENERATOR OF DATA SETS

A challenge here is to develop a framework that searches for data sets with specific complexity. This section presents two approaches, and starts with problems where the class concepts are described by the data collected in the experiments or from synthetic data following tailored distributions. Then, perturbations are introduced to the sample problems to extend their coverage in the complexity space. In the following, we present the formulation of the optimisation problem and detail the process organisation of the evolutionary multi-objective search. We also introduce an alternative way to generate ADS based on specific layouts.

6.4.1 *Formulation of multi-objective optimisation problems*

We are looking for a collection of ADS with different complexity levels for several estimates provided by the complexity measures.

In [Macià et al., 2008] we propose an approach that deals with the optimisation of the problem complexity by evolving the assignment of class labels to the instances of real-world problems.

Optimisation 1. We consider a set of n labelled instances $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n\}$ drawn from a real-world problem or from a synthetic problem generated with reference to a physical process. We search the labelling of the instances to satisfy the desired complexity (see Fig. 6.5a). Note the fixed size of the new data set, with the same size as the original problems.

Yet, this arbitrary class labelling results in data sets whose structure is unrealistic since the underlying concepts are broken. We propose a sampling selection from real-world problems where the concept is described by the samples collected in the experiments or from synthetic data sets following specific, known distributions [Macià et al., 2010b]. Thus, the problem is still expressed as a multi-objective optimisation problem but now based on instance selection where data sets have to comply with both the minimisation or maximisation of the values of the specified complexity measures, and this time internal constraints.

Optimisation 2. We consider a set of n labelled instances $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n\}$ drawn from a real-world problem or from a synthetic problem generated with reference to a physical process. We search the sub-set of instances—variable size—(see Fig. 6.5b) that minimises or maximises a set of complexity measures.

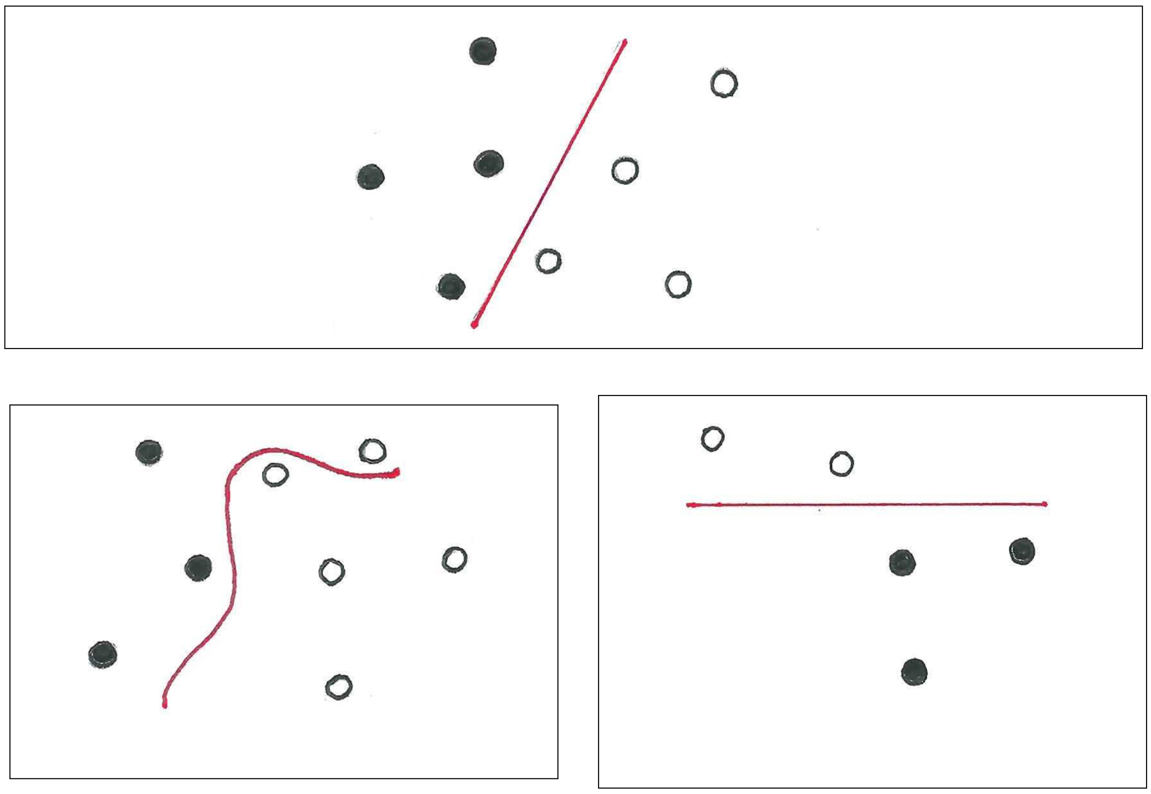
Fig. 6.5 formulates the two Multi-objective Optimisation Problem (MOP). For instance, if we wish to obtain a data set with as many points laying on the class boundary as possible, we should maximise the complexity measure N1.

Constraints. The feasibility and correctness of the resulting data sets also depends on the extrinsic complexities, such as the number of instances, the number of attributes, or a measure of balance between the classes. These external characteristics have been taken into account by including them into the approach as constraints which are responsible for (1) maintaining a minimum number of instances, (2) conserving a specific class balance, (3) ensuring that the calculation of the complexity measures is feasible, and (4) avoiding the duplicity of instances.

1. *Minimum number of instances.* Individuals must have a minimum number of instances, specified by the user, to avoid sparse data sets with poor representativeness. This value corresponds to a percentage of the total of instances that describes the original data set.
2. *Class imbalance.* Individuals have to maintain a balance among classes which depends on the ratio of the original data set and the range specified by the user.
3. *Complexity measures values.* Estimates calculation has to be feasible, i.e. if we cannot compute the value of a measure—for instance, the singular value decomposition algorithm cannot converge—, then (1) the data set should be removed or (2) a fitness penalty system should be implemented.
4. *Duplicity.* Data sets cannot contain duplicate instances. This requires a pre-processing step that removes duplicates from the original data set. Although one can think that the repetition of instances reinforces the importance of this particular concept, we insist that the proposed generation is based on the geometric characteristics and just one sample suffices to represent the concept. In addition, we note that some measures, such as N1, are extremely influenced by duplicity, biasing the estimation.

6.4.2 *EMO Implementation*

We use an evolutionary multi-objective strategy, more specifically the Nondominated Sorting Genetic Algorithm II (NSGA-II) [Deb et al., 2002] which has demonstrated its ability to efficiently and scalably solve complex problems. The EMO approach follows the classical procedure of a simple GA but includes two sorting concepts: (1) the fast non-dominated sorting and (2) the crowding distance assignment. These mechanisms enable the system to optimise different objectives in a single simulation run. The fast non-dominated sorting procedure organises the population into different fronts according to how well they satisfy the multiple objectives. The crowding distance assignment estimates the density of the solutions surrounding a particular solution in the population to measure the diversity of the population and maintain it.



[Macià et al., 2008]

Find the vector $\mathbf{t} = [y_1, ..., y_n]^T$, where n is the number of instances and y_i is the label of instance i,
which optimises

$$f(\mathbf{x},\mathbf{t}) = [F1v(\mathbf{x},\mathbf{t}), F1(\mathbf{x}.\mathbf{t}), F2(\mathbf{x},\mathbf{t}), F3(\mathbf{x},\mathbf{t}), F4(\mathbf{x},\mathbf{t}), L1(\mathbf{x},\mathbf{t}), L2(\mathbf{x},\mathbf{t}), L3(\mathbf{x},\mathbf{t}), N1(\mathbf{x},\mathbf{t}), N2(\mathbf{x},\mathbf{t}), N3(\mathbf{x},\mathbf{t}), N4(\mathbf{x},\mathbf{t}), T1(\mathbf{x},\mathbf{t}), T2(\mathbf{x},\mathbf{t})]$$

(a)

[Macià et al., 2010b]

Find the vector $\mathbf{t} = [\mathbf{z}_1, ..., \mathbf{z}_k]^T, 2 \leqslant k \leqslant n$, where n is the total number of instances and $\mathbf{z}_i$ is instance i, ($\mathbf{t}$ is a sub-set of $[\mathbf{z}_1, ..., \mathbf{z}_n]^T$)

which optimises

$$f(\mathbf{t}) = [F1v(\mathbf{t}), F1(\mathbf{t}), F2(\mathbf{t}), F3(\mathbf{t}), F4(\mathbf{t}), L1(\mathbf{t}), L2(\mathbf{t}), L3(\mathbf{t}), N1(\mathbf{t}), N2(\mathbf{t}), N3(\mathbf{t}), N4(\mathbf{t}), T1(\mathbf{t}), T2(\mathbf{t})]$$

and subject to the following constraints

$k \geqslant k_{min}$, where k_{min} is the minimum number of instances specified by the user,

Class imbalance (specified by the user), and

No duplicates.

(b)

Figure 6.5: Two approaches of MOP: (a) labelling perturbation and (b) instance selection.

Algorithm 1 NSGA-II

```
Input: $P_0$ and $Q_0$ initial populations, $N$ (Population size), $t_{max}$ (Maximum number of generations)
$t = 0$
while $(t < t_{max})$ do
  Create $R_t = P_t \bigcup Q_t$
  Perform a non-dominated sorting of $R_t$ and identify different fronts: $FR_i$, $i = 1, 2, ..., n$ by using the ranking algorithm
  $P_{t+1} = P_t$
  $i = 1$
  while $(|P_{t+1}| + |FR_i| < N)$ do
    $P_{t+1} = P_{t+1} \bigcup F_i$
    $i = i + 1$
  end while
  Include in $P_{t+1}$ the most widely spread $(N - |P_{t+1}|)$ solutions by using the crowding distance
  Create $Q_{t+1}$ from $P_{t+1}$ by using crowded tournament selection, crossover, and mutation operators
  $t = t + 1$
end while
Output: Front $FR_1$ from set $P_{t_{max}} \bigcup Q_{t_{max}}$
```

These two concepts are used in the selection procedure by applying the crowded-comparison operator. That is to say, an individual is defined by two parameters: (1) the non-domination rank and (2) the crowding distance. Then, individuals of higher rank and those that lay in regions of the solution space with a low density of solutions are preferred over individuals with lower rank or that lay in crowded regions of the feature space. This process drives the population to the Pareto-optimal line (see the pseudo code in Algorithm 1).

The system requires defining (1) the meta-information fed into algorithm, (2) the knowledge representation, i.e. the genetic representation of the solution of the problem, (3) the fitness function to evaluate each candidate solution, and (4) the process organisation accompanied by the definition of the genetic operators employed. These are described as follows.

6.4.2.1 *Meta-information to describe the data structure*

In this case, the meta-information corresponds to the extrinsic characteristics. The input of the system is a data set, whatever its distribution is—real-world distribution or synthesised distribution—, described by n learning instances, where each instance is defined by m continuous and/or nominal attributes. This data set will be altered through an instance selection process until it reaches the desired complexity.

6.4.2.2 *Knowledge representation*

The NSGA-II system evolves a population composed of N individuals, where each individual is a candidate solution, i.e. an artificial data set characterised by a selection of instances from the input data set. The representation of the individual is a vector of size $k \leqslant n$, where k is the number of instances selected by the current individual and n is the number of instances in the original data set. The array contains the indexes of the selected instances, e.g. if an individual contains {2,5,14}, it means that instances 2, 5, and 14 of the original data set are selected. The EMO searches the best selection of instances that satisfies the required complexity, i.e. that minimises or maximises the set of specified complexity measures. Then, the final data set is obtained by moving from the genotypic representation to the phenotypic representation, i.e. by grouping the instances coded in the individual into a file and adding the corresponding header to provide a data set in Weka format.

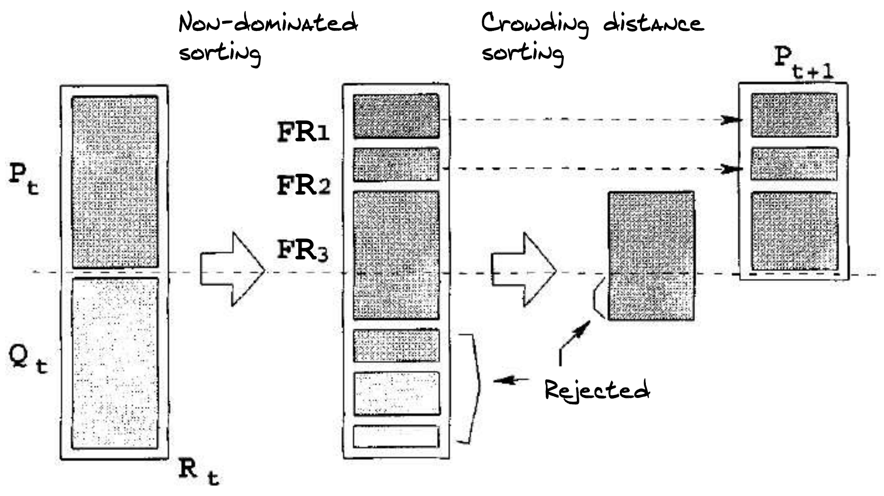


Figure 6.6: Fast non-dominated sorting and crowding distance assignment

6.4.2.3 *Fitness*

The fitness function evaluates each candidate solution; each objective refers to a complexity measure. Their computation is performed by the DCoL which implements the complexity measures originally proposed by Ho and Basu [2002]. To calculate the fitness of the individual, we simply translate the genotype into the phenotype according to the information encoded in the individual and compute the value of the complexity measures using the DCoL.

6.4.2.4 *Organisation of the search process*

In our implementation, we use an EMO technique with a population of fixed size N that works as follows. At the beginning, the individuals of the initial population P_0 are randomly initialised and evaluated. Then, we create a second random population Q_0 whose individuals are also evaluated and will be the offspring population in the next iterations. Then, the GA bases the search on the interaction of two specific and three primary genetic operators that are iteratively applied: *fast non-dominated sorting*, *crowding distance assignment*, *selection*, *crossover*, and *mutation*.

First, the populations P_t and Q_t are combined and the resulting population R_t is sorted according to the non-dominated procedure. Note that this approach ensures elitism since all the members of the parent population P_t are included in the combined population. Then, starting from the first front FR_1, all the solutions of each front i are introduced into the new population P_{t+1} provided that there is enough room to allocate all the solutions of the given front. Otherwise, the solutions with the highest crowding distance of front FR_i are introduced into P_{t+1} until the population is filled; thence, no more solutions of higher fronts are added to the population. Fig. 6.6 schematises the procedure.

Thereafter, selection is applied to choose the parent population of the next generation. Then, pairs of these parents are selected without replacement, and they undergo crossover and mutation with probabilities χ and μ respectively. If neither crossover nor mutation are applied, the offspring are exact copies of the parents. Then, all the individuals of the population are evaluated. This process is repeated until the stopping criterion is reached, which is the limit on the maximum number of generations.

6.4.2.5 *Genetic operators*

In order to generate feasible individuals and, consequently, realistic data sets, we have to address some constraints and slightly modify the functioning of the genetic operators to include them. Hence, the following are taken into consideration in (1) selection, (2) crossover, and (3) mutation.

Selection. Tournament selection is applied to choose the individuals to mate. Tournaments of s randomly chosen parents are held, and the best parent, according to the crowded-comparison operator, is selected for recombination. Regarding hermaphrodite recombination, if the two tournaments result in the same candidate, then we have to repeat the tournament until a different candidate is found.

Crossover. Two-point crossover is the most popular strategy to recombine two individuals. It consists in randomly choosing two cut-points along the individuals and shuffling the information of both parents. The range of the cut-points corresponds to `[0,min(size individual1, size individual2)]` to maintain the consistency of the offspring.

No duplicate instances in the individual are allowed, so they will be removed. In fact, by construction, recombination can produce only two repetitions of the same instance. After removing all of the duplicated instances, it is necessary to check whether the individual contains the minimum number of instances. Therefore, once individuals are recombined, if the offspring does not comply with the minimum size, the following policy is applied: (1) determine the number of instances to be added at random, (2) select the instances at random maintaining the class balance constraint, and (3) insert the instances to the individual at the end of the structure. Instead of replacing the removed instances with new ones from their respective classes, we prioritised the procedure described above due to its simplicity and low computational cost.

Mutation. Addition or deletion of instances brings some diversity into the population. First, we randomly assign a Boolean value to each instance of the initial data set. Then, for each instance, if the value is $true$, we look for this instance in the individual to mutate. If the instance exists, we remove it. Otherwise, we add it at the end of the structure. As with the crossover operator, the minimum number of instances has to be conserved. We therefore apply the same policy, as described above.

It is worth noting that with such decisions an instance can be in different positions of the individual (i.e. occupying different alleles), which provides a higher flexibility that favours the identification and propagation of building blocks.

To sum up, this evolutionary technique allows us to stretch different dimensions of complexity at the same time. However, correlations among variables (i.e. measures) or a high number of dependent variables affect the optimisation performance. This suggests that we should perform the generation in steps by combining the complexity measures three by three.

6.5 EXPERIMENTATION

Thus far, we have provided the details of the multi-objective method for ADS generation. This sections moves on to the experimentation to show that the system is able to generate data sets with difficulty bounded by several complexity measures. We describe the experimental methodology and then present some results concerning the generation of data sets with constraint characteristics.

6.5.1 *Experimental methodology*

The purpose of the experimentation is (1) to empirically show that our approach is indeed able to evolve a Pareto set of problems that approximate the desired difficulties and (2) to analyse whether these artificial problems fill regions of the complexity measurement space that were not covered by the collection of 264 real-world data sets shown in Sect. 6.2.1. For this purpose, we run the EMO system fed with three different problems from the UCI Repository: *pim*, *tao*, and *wbcd*. We remark that we use real-world problems to ensure that the structure of artificial data set resemble that of the real-world problems. The data sets are binary classification problems described by continuous- and real-valued attributes. The system was initialised with a population of 200 individuals and was run for 30 generations. The probabilities of crossover and mutation were 0.80 and $1/n$ respectively, where n is the number of instances of the original data set (i.e. $n_{pim} = 768$, $n_{tao} = 1888$, $n_{wbcd} = 699$).

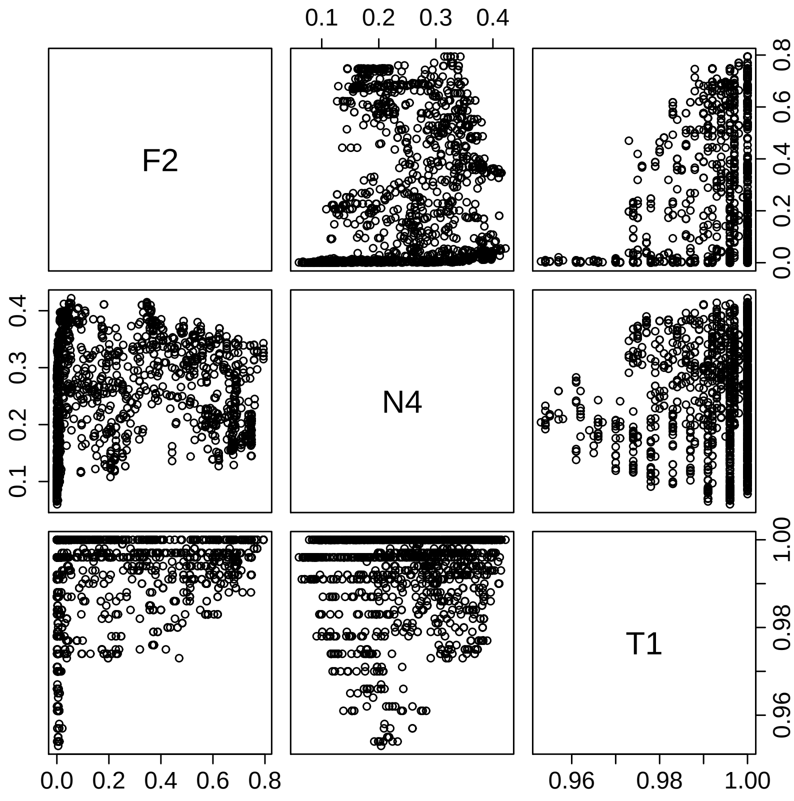


Figure 6.7: Projection of the data set obtained from the experiment 1 on a complexity space built upon F2, N4, and T1.

6.5.2 *Results*

In the following, we gather the results obtained from the experimentation, which is divided into two phases: (1) a case study and (2) the coverage of the measurement space.

Experiment 1 - Case Study. After wide experimentation based on tuning the combination of complexity measures to be optimised, we report an illustrative case study that shows the interesting usage of the resulting data sets and the correct functioning of the system. We evolved three dimensions of complexity that estimate different difficulty aspects: (1) the overlap of the per-class bounding boxes (F2), (2) the non-linearity of the one-nearest neighbour classifier (N4), and (3) the fraction of maximum covering spheres (T1). We ran the system with the aim of optimising eight configurations according to the objective optimisation {min,max} (see Table 6.3). The outcome of this experiment, for instance, should offer a complete analysis scenario that allows us to study how sparsity affects the learner behaviour, since F2 and N4 are measures that identify the existence of regions where instances of classes are highly overlapped and T1 gives an estimation of the scatter within the class [Ho, 2001].

Fig. 6.7 projects all of the data sets obtained from the different runs on the complexity space. The x-axis of the first column refers to F2, the second column refers to N4, and the third column refers to T1. The y-axis, in turn, represents F2 for the first row, N4 for the second row, and T1 for the third row. We can observe that the data sets are distributed across the space, although some resolutions have not been achieved (e.g. N4 cannot go further than 0.4220 and T1 variability is extremely limited in the interval [0.9530-1.000]).

Figures 6.8a and 6.8b plot, in three dimensions, the Pareto fronts obtained from the "degeneration" of the *pim* data set for two combinations: (1) minimisation of all three objectives and (2) minimisation of F2 and T1, and maximisation of N4. The difference between the experiments lays in the optimisation of the complexity estimate N4 which first has to be minimised (see Fig. 6.8a) and then maximised (see Fig. 6.8b). We see how the complexity for each experiment

Table 6.3: Information about the objective selected for optimisation, number of individuals located in the Pareto set, the minimum and maximum values and the average crowding distance (ACD) for each complexity estimate.

Objectives			Pareto-Optimal	F2			N4			T1		
F2	N4	T1		Min	Max	ACD	Min	Max	ACD	Min	Max	ACD
min	min	min	35	0.0000	0.0100	0.0559	0.0600	0.2240	0.0565	0.9530	0.9960	0.0581
min	min	max	4	0.0000	0.0040	0.3333	0.0650	0.0780	0.4103	0.9910	1.0000	0.3333
min	max	min	72	0.0010	0.1090	0.0267	0.2680	0.4030	0.0274	0.9730	1.0000	0.0276
min	max	max	33	0.0010	0.0550	0.0584	0.2870	0.4220	0.0606	0.9930	1.0000	0.0491
max	min	min	104	0.0920	0.7600	0.0191	0.1160	0.2470	0.0189	0.9740	1.0000	0.0194
max	min	max	11	0.2060	0.7480	0.1970	0.1080	0.1900	0.1683	0.9920	1.0000	0.2000
max	max	min	98	0.1690	0.7260	0.0200	0.2790	0.3960	0.0194	0.9730	1.0000	0.0195
max	max	max	24	0.3460	0.7940	0.0762	0.3430	0.4150	0.0833	0.9880	1.0000	0.0725

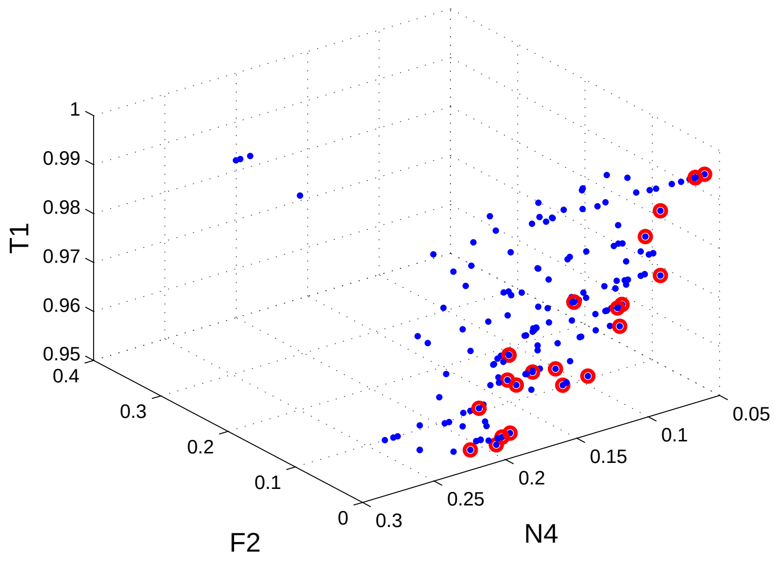
moves from [0.06-0.22] to [0.26-0.40] satisfying the specified optimisation requirements. Table 6.3 complements the graphs summarising the number of individuals in the Pareto set, the minimum and maximum values reached for each complexity dimension and the average crowding distance of this set of solutions. For the calculation of the average crowding distance, the objective values were normalised. On average, the Pareto set consists of the 20% of the individuals of the population. Note that, although all individuals are valid data sets, those located on the Pareto set are the closest to the desired complexity. In some cases, the Pareto set contains few solutions (e.g. four or eleven) which does not mean a malfunctioning of the system, but a need for either more iterations to shift the solutions explored to the Pareto line or diversity in the population.

Experiment 2 - Coverage of the measurement space. We evolve two objectives: (1) the fraction of points on the class boundary (N1) and (2) the ratio of the average of intra/inter nearest neighbour distance (N2). We run the system with four different configurations according to the optimisation objective. This experiment involves two complexity measures of class separability, N1 and N2. Both measures, highly correlated, are used to define a preliminary complexity space on which some learner domains of competence are sketched [Bernadó-Mansilla et al., 2006]. Nevertheless, the dearth of real-world problems available called the measurement space into question and laid fragility at the conclusions drawn from it. With our approach, we expect to generate new data sets that complete the measurement space defined by these two complexity descriptors and also contribute to the study of the complexity estimates.

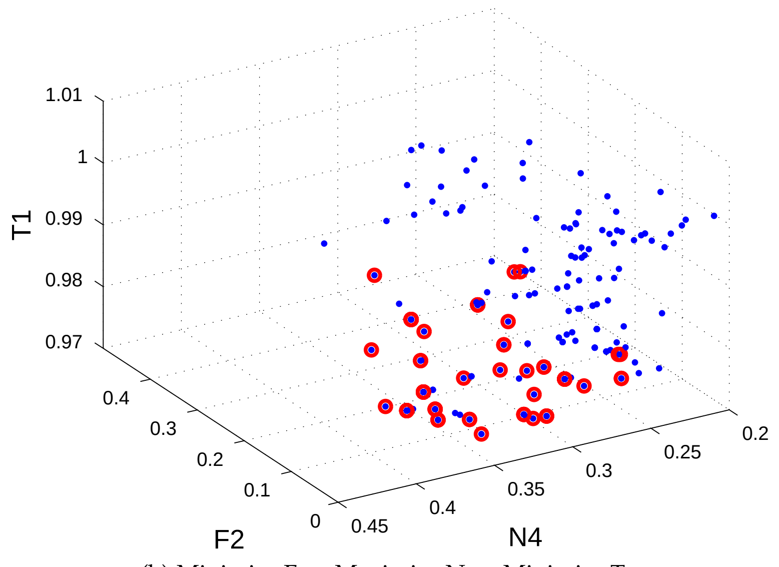
Figures 6.9 and 6.10 plot the results of eight experiments in which we take two problems and performed the four optimisation combinations (min-min, min-max, max-min, max-max). Again, we observe how the approach is able, from a given problem, to generate a collection of artificial problems spread out on the measurement space. However, it is important to note that the approach will push the limits of the complexity until a certain extent which depends on the structure of the problem. For example, the *pim* problem allows the system to vary the complexity of N1 from 0.0090 to 0.9189 and N1 from 0.0680 to 1.2220, whereas the complexity of the data sets built from the *tao* problem oscillates from 0.0020 to 0.2210 for N1 and from 0.0270 to 0.2560 for N2.

In addition, these synthetic data sets cover regions of the problem space that are not covered by the collection of 264 real-world problems employed in the case study, achieving the creation of data sets diverse enough to fall into some of the blind spots detected with real-world problems (see Fig. 6.11).

To sum up, the experiments provided throughout this section have empirically shown the feasibility of generating targeted-complexity problems. By doing so, we answer our first question *Q1. Do the empty regions mean that this kind of problem does not occur in nature?* Our experiments suggest that the answer is no, because at least some holes have been covered thanks to the selection of samples of real-world problems. In the next chapter, we uses this approach to generate problems in order to propose a more complete test set.



(a) Minimise F2 - Minimise N4 - Minimise T1



(b) Minimise F2 – Maximise N4 – Minimise T1

Figure 6.8: Experiment 1. Generation of ADS following a three-targeted complexity defined by F2, N4, and T1 from *pim* data set. The circled points belong to the Pareto set.

6.6 an alternative approach

Another approach to generate ADS is to use specific layouts and apply some variation while preserving their underlying structure. This section exposes a couple of directions to proceed. We present preliminary ideas regarding the checkerboard problem and a physical process generation. It is fair, though, to admit that the input data set continues to limit the approach since there is an associated upper bound and lower bound complexity inherent to the problem; more perturbations have to be applied to the data.

6.6.1 *Checkerboard*

We can think of the checkerboard problem as being easy, but just because the layout is obvious does not mean that is trivial for an automated learner to solve. The problem here is the inherent complexity and the mismatch with the learning paradigms. For this reason, we decide to

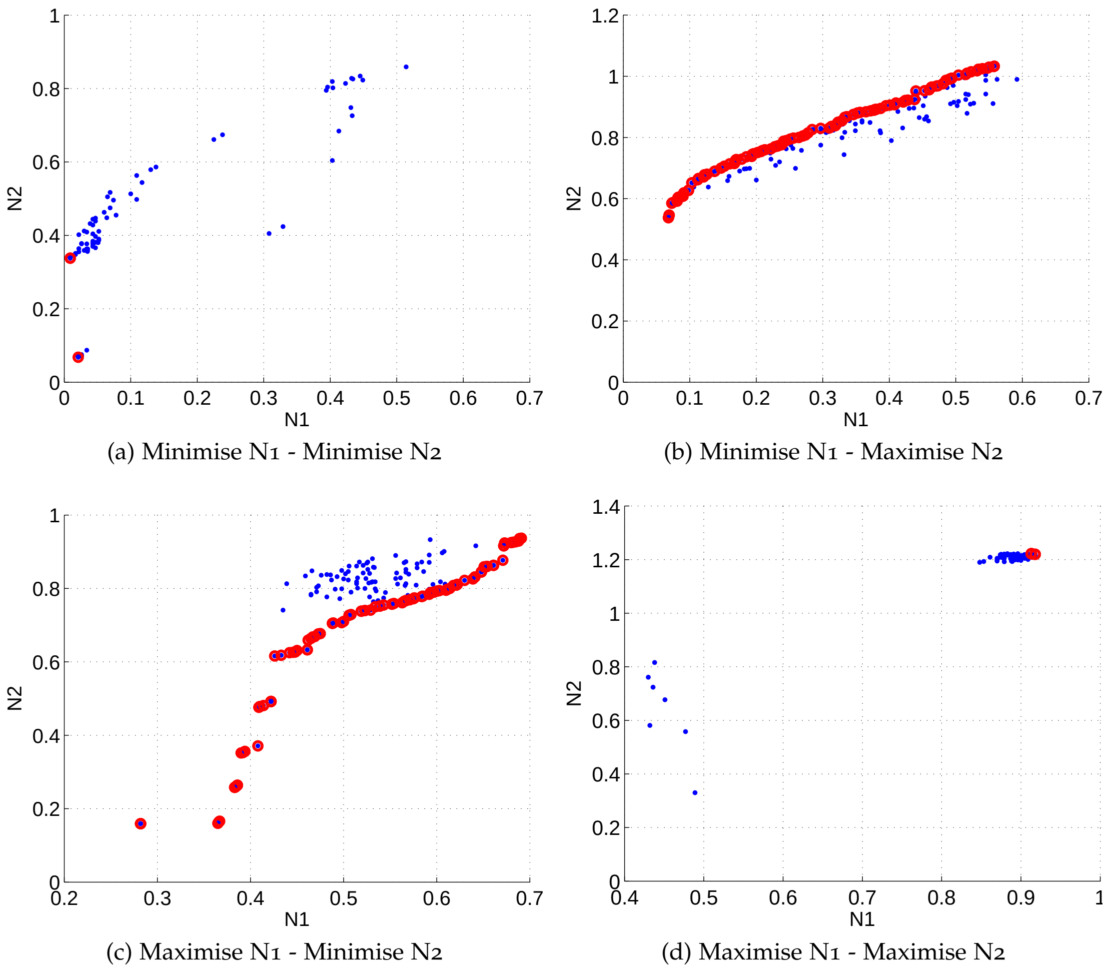


Figure 6.9: Experiment 2. Generation following a two-targeted complexity defined by N1 and N2 from the *pim* problem. The circled points indicate the data sets that belong to the Pareto set.

generate different versions of the problem by varying the number of squares, the mean, and the variance (see Fig. 6.12). First, we generate c mother instances whose values are ranged in [0,1]. These instances correspond to the centre of each cluster which are equidistant. Next, we generate for each cluster, k daughter instances using a Gaussian distribution. The mean of the point k_{ij} is the coordinate of the mother m_{ij} where i refers to the cluster and j the attribute. The standard deviation is varied to analyse how the spread and overlapping affect the classifier behaviour. We select the standard deviation by dividing the distance between mothers into twenty partitions. Then, for each configuration problem, we build ten data sets with ten different standard deviations from $d/20$ to $d/2$.

The main difference between this approach—which can be applied to other known concepts such as the spiral, a wave boundary, or a linear boundary—and the EMO remains in the underlying concept. The layout is preserved and well-defined for all the problems. Furthermore, these problems can be used as input data sets—seeds—for the EMO. The guided selection of instances is useful to study the effect of sparsity.

6.6.2 *Physical process*

We also suggest to create a new breed of data sets based on the use of different distributions, in particular the Neyman-Scott process [Neyman and Scott, 1958], and organise instances into

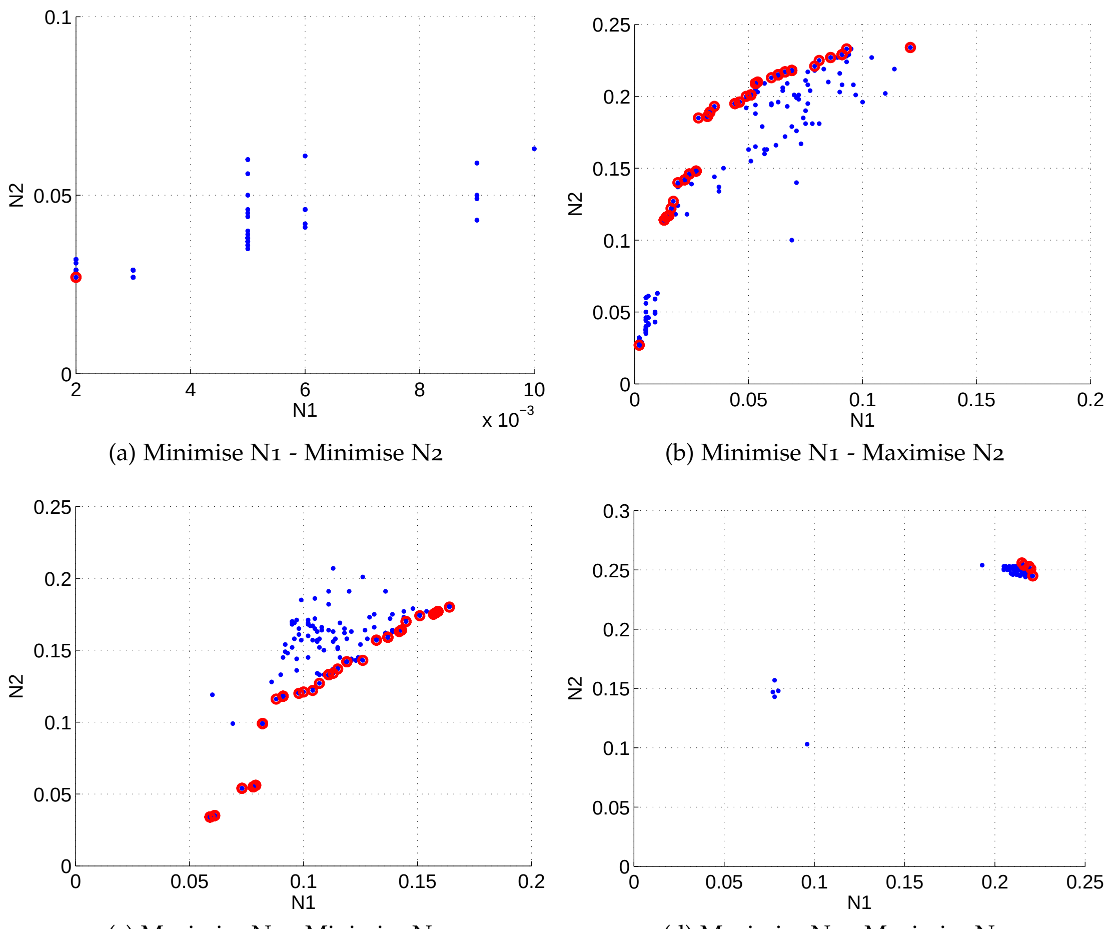


Figure 6.10: Experiment 2. Generation following a two-targeted complexity defined by N1 and N2 from the *tao* problem. The circled points indicate the data sets that belong to the Pareto set.

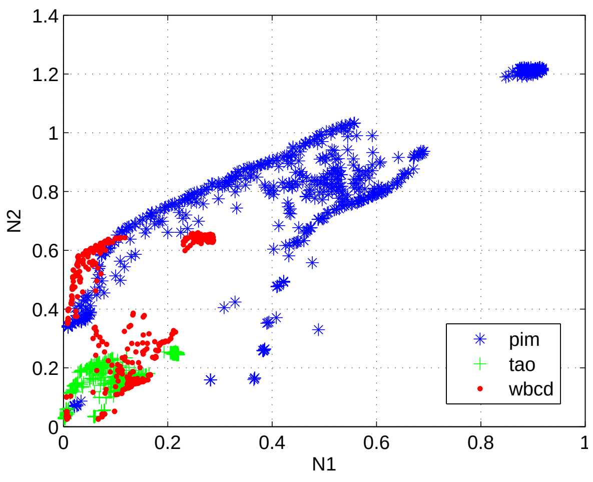
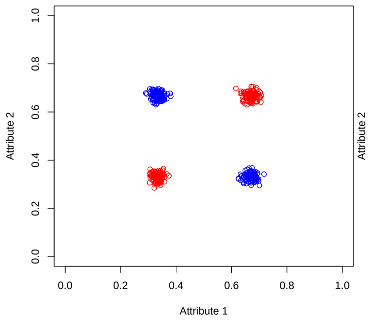
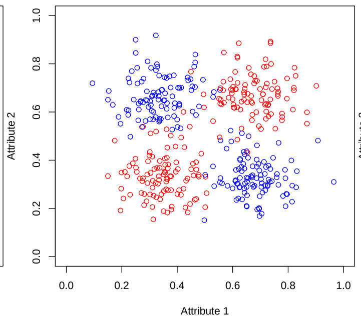


Figure 6.11: Projection of the data set obtained from the experiment 2 on a complexity space built upon N1 and N2.

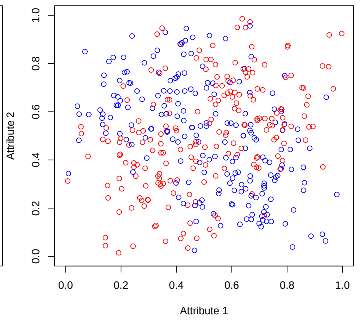
clusters. First, we generate c mother instances by means of a Poisson distribution—implicit physics process—with $\lambda = 0.5$ and normalise the values. These instances correspond to the centre of each cluster. Then, for each cluster, we generate k daughter instances using a Gaussian



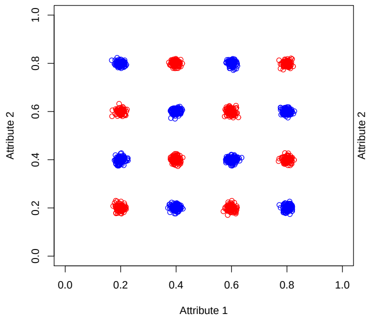
(a) std=0.01000



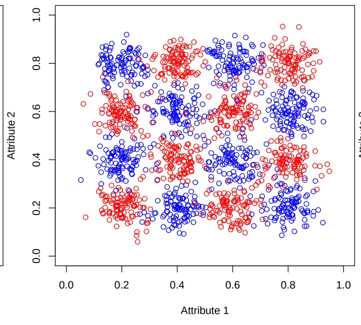
(b) std=0.05000



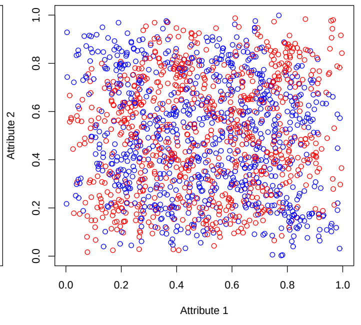
(c) std=0.10000



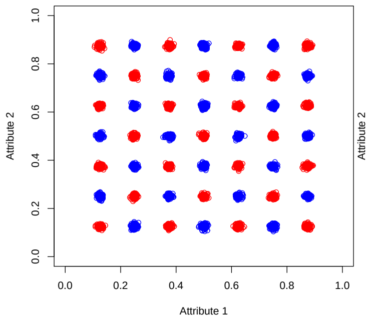
(d) std=0.00294



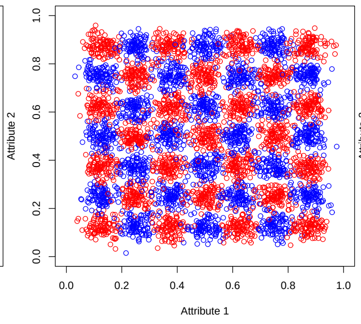
(e) std=0.01471



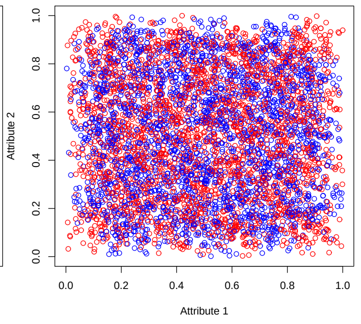
(f) std=0.02941



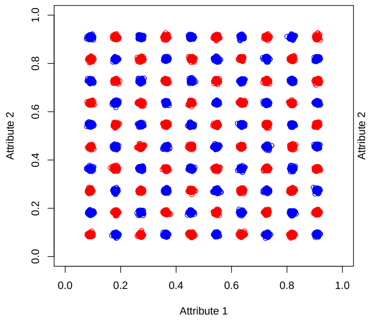
(g) std=0.00100



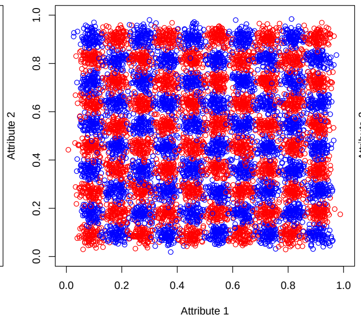
(h) std=0.00500



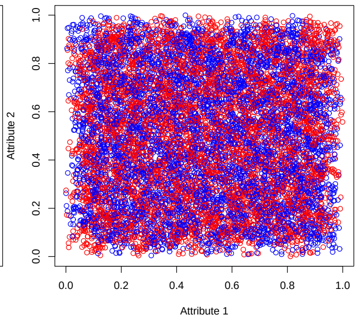
(i) std=0.01000



(j) std=0.00050



(k) std=0.00248



(l) std=0.00495

Figure 6.12: Checkerboard distribution for 4, 16, 49, and 100 clusters.

distribution. We use the terms mother and daughters since the latter will have the same class label as the mother (see Fig. 6.13). The mean of the point k_{ij} is the coordinate of the mother m_{ij} where i refers to the cluster and j to the attribute (see Fig. 6.14).

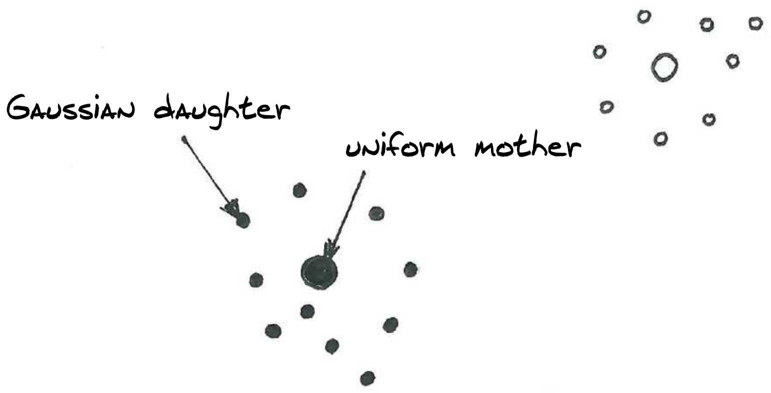


Figure 6.13: Mothers seed generated following a uniform distribution and daughters generated with a Gaussian distribution.

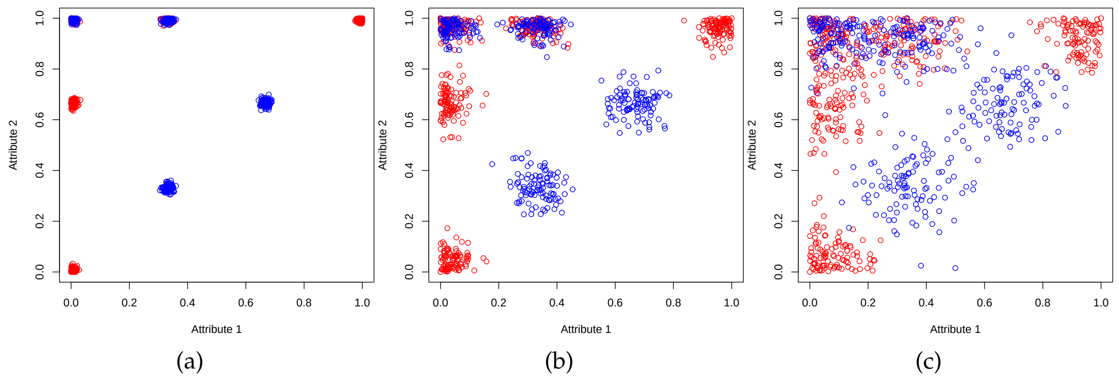


Figure 6.14: Poisson distribution for 9 clusters.

6.7 DISCUSSION AND FURTHER WORK

This EMO approach has served to confirm that problems with a certain complexity that can not be found in public repositories can be generated and may exist in nature. This work has also served to point out four aspects that we would like to address in further work: (1) practicability, (2) scalability to many objectives and facet analysis, (3) complexity measures, and (4) analysis of the learner's domains of competence. In the following, each one of these aspects is elaborated.
Practicability. In computational terms, the proposed solution for the generation of ADS is costly, since we have to add the cost of the complexity measures computation to the well-known cost of EMO algorithms. Changing the internal representation of the individual into a bit string representation could reduce memory allocation and the computational cost of genetic operators. In future work, we would like to carefully revise the design of genetic operators and determine the advantages of the implementation of some constraints with respect to the balance between convergence and time.
Scalability and facet analysis. The search ability of multi-objective optimisation algorithms declines with the increase of the number of objectives, usually more than three [Ishibuchi et al., 2008]. Taking into account that we have to manage a set of fourteen complexity measures, the non-dominated sorting system could be modified to approach many-objective optimisations as proposed in [Hughes, 2005]. However, despite the enhancement, facet analysis—i.e. an incremental analysis of each complexity measure and its interaction with the rest—is a promising way to

go in the quest for a complete complexity space. This could also enable a better understanding of each individual complexity measure.

Complexity measures. The set of complexity estimates is a preliminary proposal and its consolidation is without doubt the first request for classification benchmark problems. To this end, the EMO approach may be helpful to the study of the correlation of complexity descriptors. On the other hand, some of the complexity measures use Euclidean distance and ignore the shape of the decision boundaries. Although it seems reasonable to measure complexity by using the units managed by learners, such as hyper-rectangles or hyper-spheres, we should bear in mind that real-world problems do not follow such perfect distributions, and this stresses the decisive role of the knowledge representation in the design of learners and the linkage to the data complexity.

Analysis of the learner's domain of competence. Once we are able to generate data sets with targeted-complexity, we have to tackle another arduous task: how to use them to probe specific properties of techniques. The idea is to select reference learners and run them over a massive collection of data sets, some of them picked of the Pareto set and some others of the rest of the search. The analysis of the results would focus on defining the domains of competence of learners, and provide some rules that explain the ins and outs of the gap between data characteristics and learner properties.

6.8 HOW MANY GENERATORS MORE?

In the literature, the motivation to design synthetic data sets has been, and still is, the cost of collecting real samples, data confidentiality, and the pursuit of controlled scenarios. Rachkovskij and Kussul [1998] proposed DataGen which permits varying the number of input attributes, the number of instances, the number of output classes, the complexity and realisations of class regions focusing on local peculiarities of the class boundary—e.g. bends or convex and concave segments—, the distributions of the samples, and the noise level. Hoag and Thompson [2007] determined the necessity of generating *industrial-sized* problems and designed the Parallel Synthetic Data Generator (PSDG). This option uses parallel computation to speed the generation of the samples and is more oriented to build commercial data, more goal-directed to scalability than complexity control. Some other generators can be found online such as DatGen[1] [Melli, 1999], datgen[2], and datagen[3]. Our interest for generating artificial problems comes from the incompleteness and lack of resolution in the test bed. We propose an approach to evolve both extrinsic and intrinsic complexities. However, among this range of possibilities, the community sticks to testing over real-world problems, disregarding the limited amount of data, the unknown characteristics, or their toy-like appearance. Again, this exposes a lack of soundness in the experimental methodology, especially in the testing process which should consist of two stages: (1) a functionality test on synthetic problems, i.e. to gauge the learner's ability under a controlled scenario and (2) a validation test on real-world problems, i.e. to assess the performance on the problems which the learner was designed for. Hence, although data generators intend to create realistic problems, synthetic and real-world problems are both mandatory elements of the experimental methodology.

Contribution.

1. Taxonomy of the complexity characterisation of data sets.
2. Proposal and implementation of an evolutionary approach to generate artificial data sets with a targeted complexity.

1 http://www.gabormelli.com/datgen/overview.html
2 http://www.datasetgenerator.com/
3 http://www.burningart.com/meico/inventions/datagen/index.html