AMSI VACATION RESEARCH SCHOLARSHIPS 2019–20

EXPLORE THE MATHEMATICAL SCIENCES THIS SUMMER

Generating New Instances For Stress Testing Classification Algorithms

Kulunu Dharmakeerthi

Supervised by Professor Kate Smith-Miles

University of Melbourne

Vacation Research Scholarships are funded jointly by the Department of Education and Training and the Australian Mathematical Sciences Institute.

1 Abstract

As machine learning algorithms become increasingly widespread, it is essential that rigorous stress-testing frameworks become foundational to algorithm development and application. A diverse collection of data is necessary for in-depth analysis of algorithm performance . However, recent work has shown that current repositories do not meet these requirements. In particular, classification datasets fail to separate ML classifiers by performance. Popular algorithms perform similarly well or similarly poorly on commonly used test instances, thus, a test instance space lacks the discriminatory ability necessary for thorough algorithm testing. This paper discusses a method to create discriminating classification datasets that would increase the diversity and breadth of a test instance space. Genetic algorithms are employed to tackle the task of data generation which is structured as a multi-objective optimisation problem.

2 Introduction

Recent decades have seen large advancements in the development and application of machine learning (ML) algorithms. Objective performance evaluation of ML techniques is now of paramount importance in justifying their real-world use as we start to increasingly rely on them for AI-based automated systems. Worryingly, potential inadequacies in the current accepted methodology for testing classification algorithms have been been reported for over a decade now (Salzberg (1997); Munoz et al. (2018))

Recent work has shown that current repositories fail to adequately separate classifiers by performance, often times multiple algorithms perform similarly well. Standard practice recommends classifiers are tested on a well-studied collection of classification datasets, i.e. from the UCI repository, KEEL repository. Opposers to such practice raise concerns about over-tuning algorithm development to a set of test-instances that may not be representative of the larger population of classification problems; 'the UCI repository is a very limited sample of problems, many of which are quite easy for a classifier' (Salzberg (1997)). Popularity of the UCI repository, and consequently, over-reliance on its test instances is potentially problematic as new algorithms may be developed with bias towards the known properties of the UCI datasets (Munoz et al. (2018)). For unbiased evaluation of classifiers, and improved understanding, a

more diverse and expansive test instance space is required. When such a space is absent, it must be created.

In this paper, we focus particularly on generating datasets that are capable of discriminating algorithms by performance. In essence, we are interested in statistical and information theoretical qualities of datasets that may suggest suitability of particular classification algorithms. With this in mind, a methodology is developed to generate idealised artificial test data instances designed to be suited to specific algorithms.

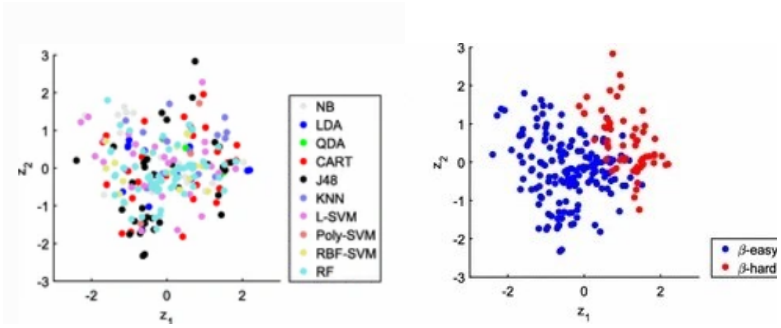
As a preliminary step binary classification tasks from the UCI repository are modified through iterative class re-labelling, attempting to drive the difference in classification accuracy between a given pair of algorithms higher. A multi-objective fitness function searching for maximum difference in algorithm accuracy is fed into a genetic algorithm. Specifically, NSGA-II (Deb et al. (2002)) and MOEA/D (Zhang & Li (2007)) are used.

The natural next step is to generate datasets from scratch. This is structured as a two-step optimisation process. In the first step, a data structure is generated via searches for optimal Gaussian Mixture Model (GMM) parameters. The dataset produced through sampling from the GMM parameter vector is then superimposed with a class label vector, again sought out via GA optimisation.

3 Preliminaries

3.1 Instance Space

Recent work done by Smith-Miles and Munoz establish the motivating backdrop for the current investigation. The 2017 paper, "Instance Spaces For Machine Learning Classification" (Munoz et al. (2018)) investigates creating a 2-dimensional projected space that embeds classification datasets as points in an instance space. The utility of the space is that it allows visualisation of collections of datasets and visual separation indicates diverging statistical and information theoretical properties within collections. Crucially, the instance space is constructed to reveal hard and easy instances, and enable strengths and weaknesses of individual classifiers to be identified.



(a) (b)

Figure 1: Overall performance of the algorithm portfolio, with the best algorithm for each instance shown in (a), while b shows blue marks representing -easy instances, and red marks representing -hard instances - Munoz et al. 2017

Applied to the well established UCI repository, it was found that current test instances are inadequate in discriminating algorithms by performance. Often, algorithms perform similarly well or similarly poorly on existing test instances.

The initial purpose of this paper is to demonstrate a viable method for adjusting existing classification datasets so they are more discriminating. Binary classes are re-weighted and relabelled so a particular algorithm is favoured in classification tasks. The Haberman dataset (UCI) is chosen as a proof of concept.

3.2 NSGA-II

NSGA-II (Non-dominated Sorting Genetic Algorithm) (Deb et al. (2002)) will drive the optimisation task. A genetic algorithm (GA) is a metaheuristic inspired by the processes of natural selection that seeks optimal values for an objective function. In a GA, a population of candidate solutions is adjusted over many generations (iterations), to evolve toward better solutions; solutions with higher fitness (better solutions to objective function). Like in natural selection, 'fitter' individuals are carried over to the next generation. In addition, mutation and crossover operators reestablish population diversity in future generations. NSGA-II is a multi- objective GA chosen for its low computational complexity and elitist approach.

3.3 Machine Learning Algorithms

We consider a portfolio of 7 popular supervised learning algorithms. The algorithms are Naive Bayes (NB), Linear Discriminant (LDA), Quadratic Discriminant (QDA), Classification and Regression Trees (CART), k-Nearest Neighbor (KNN) and Support Vector Machines with linear and radial basis kernels (L-SVM, and RB-SVM respectively). NB and CART are expected to give uncorrelated errors while providing a good diversity of classification mechanisms (Lee & Giraud-Carrier (2013)); LDA and QDA are expected to further extend the diversity of the algorithm portfolio, whereas KNN and SVM are considered because of their popularity.

3.4 Gaussian Mixture Modelling

In statistics, a mixture model is a probabilistic model for representing the presence of sub-populations within an overall population, without requiring that an observed data set should identify the sub-population to which an individual observation belongs

Multivariate Gaussian Mixture models are considered in this paper. However, rather than fitting a mixture model to our data, we sample from a mixture model to generate a desirable data set. See Methodology section for explanation.

4 Haberman Methodology

We structure the task of generating discriminating datasets as a multi-objective optimisation problem. The objectives in this case concern the classification accuracies of chosen algorithms. Optimising these objectives drives the difference in performance of chosen algorithms higher. The initial moments of this paper will discuss a manipulation of the Haberman dataset (UCI) (Dua & Graff (2017)) as a small scale proof of concept of the proposed methodology.

4.1 Haberman Dataset

The initial purpose of this research is to demonstrate a viable method for adjusting existing classification datasets so they are more discriminating. Binary classes are re-weighted and relabelled so a particular algorithm is favoured in classification tasks. The Haberman dataset

concerns cases from studies conducted on the survival of patients who had undergone surgery for breast cancer. It has 3 attributes, 1 class attribute (survival) and 306 observations.

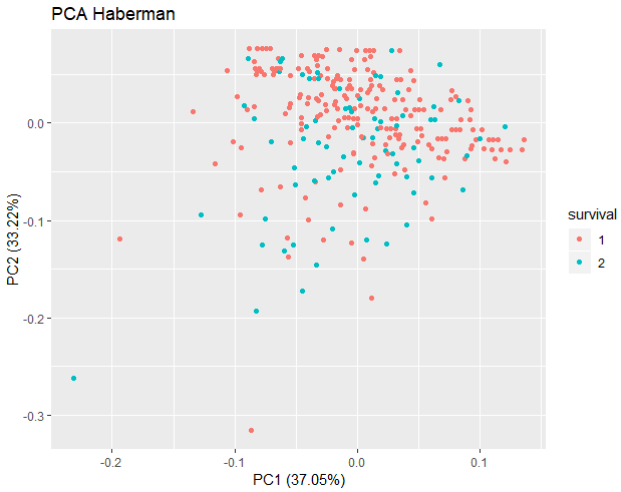


Figure 2: Principal Component Projection of Haberman Dataset

The Haberman dataset is one of 300+ UCI repository datasets considered in the 2017 paper by Munoz et al. It is chosen for this initial problem due to its small size; hence low computational complexity for the optimisation task, and as it displays the symptoms of the UCI repository discussed earlier. Indeed, looking at ML algorithm performance on the Haberman dataset, we see our suite of algorithms performing similarly.

Naive Bayes	LDA	QDA	Classification Tree	KNN	SVM Linear	SVM Radial
0.7488508	0.7508211	0.7410461	0.7397164	0.7037183	0.7211404	0.7322043

Figure 3: Classification accuracy after 10-fold cross-validation

4.2 Optimisation

Pairwise competition between two different ML algorithms drives the optimisation of the Haberman dataset. For a chosen pair, the aim is to maximise accuracy of one algorithm while minimising the accuracy of the other by relabelling the binary 'Survival' class via NSGA-II. The objective being minimised, i.e. the multi-objective fitness function, is given below:

$$F(x) = (f_1(x) = ER_{algorithm} - 1,\ f_2(x) = 1 - ER_{other})$$
$$\text{subject to } \mathrm{x} \in [0,1]^N$$

Here x refers to the class label vector of the Haberman dataset with N being the number of observations. As this a a binary class dataset, the search space for x is restricted to $[0,1]^N$. $ER_{algorithm}$ refers to the error rate of our desired algorithm, ER_{other} indicates the error rate of the discriminated algorithm

Due to the implementation of NSGA-II in R (MCO package, Mersmann 2014) being a real-valued solver, a bit string approach (1's and 0's representing the two classes) was not possible. Fitness function implementation in R, therefore, had to be structured to overcome this incapability. Given a class vector x, each value x_i, was assigned to class '1' if $0 \leq x_i \leq 0.5$ and class '2' if $0.5 < x_i \leq 1$. The 'fitness' of a candidate solution is computed after 10-fold cross validation.

NSGA-II was initialised with a population size of 100 and was run for 100 generations. The final population after GA can be expressed as an objectives plot as below:

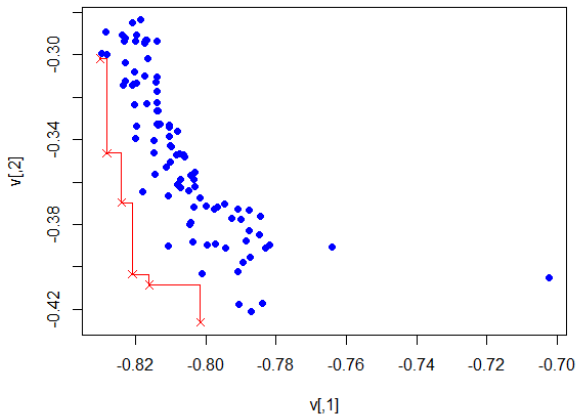


Figure 4: NSGA-II Objectives Plot

The objectives plot displays the fitness values of the final population after running GA. From this final population of candidate solutions, the ideal member is chosen as follows:

- Solution set reduced to Pareto Optimal solutions (lying on red line)
- Remove candidates with f_2 value significantly less than 0.5
- Select member closest in Euclidean distance to $(-1, 0.5)$

The ideal class labelling of the Haberman dataset for the chosen pairwise competition is then determined. Next steps include visualisation of the new dataset via PCA and in an "instance space".

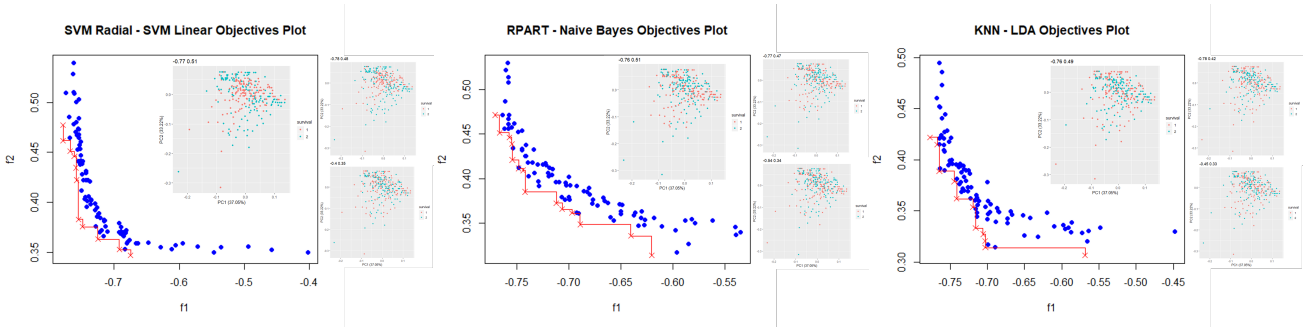
4.3 Haberman Results

For each pair of ML algorithms, the GA optimisation methodology outlined above is implemented. The table below indicates the difference in classification accuracy, $-(f_1 + f_2)$, for a given algorithm pair after running GA and choosing an improved class labelling

	Naive Bayes	**LDA**	**QDA**	**Classification Tree**	**KNN**	**SVM Linear**	**SVM Radial**
Naive Bayes	0.00000000	0.32612926	0.33039674	0.37556150	0.30285618	0.30408718	0.28198253
LDA	0.28780752	0.00000000	0.29754658	0.33467232	0.34982411	0.09768625	0.35835349
QDA	0.03335558	0.27102995	0.00000000	0.24896274	0.25102823	0.28101575	0.29868534
Classification Tree	0.10373331	0.12476757	0.16394165	0.00000000	0.18181799	0.07952772	0.13083009
KNN	0.23231855	0.26872312	0.19541157	0.22260753	0.00000000	0.19900955	0.20191949
SVM Linear	0.23065029	0.05735076	0.24525839	0.35336022	0.28433560	0.00000000	0.29723674
SVM Radial	0.07508737	0.11511170	0.13659542	0.39528342	0.25584376	0.09526882	0.00000000

Figure 5: Results from pairwise competition *For each cell in the table, the column name (in bold) indicates the algorithm whose performance was maximised, and row name the algorithm whose performance was minimised*

As evident, certain pairs of ML algorithms are difficult to separate in terms of cross-validated accuracy through class relabelling. Indeed, algorithms inherently similar in their classification strategies can be expected to perform comparably on new data. Thus optimisation is less successful. As an example consider SVM Linear and LDA. Both seek to find a linear boundary between two classes. Although the methodology each employ is vastly different it is not adverse to think simply altering the class labelling of the Haberman dataset is insufficient to separate the two in performance. This is reflected in the pairwise competition table above.
Moving forward, the above set was reduced to 3 pairs to explore further and confirm viability of the GA method. Three members from the final GA population; the ideal solution, the solution with maximum f_1 and the solution with minimum f_1., were visualised using PCA of a relabelled Haberman dataset.



(a) SVM Radial vs. SVM Linear (b) Classification Tree vs. Naive Bayes (c) KNN vs. LDA

Figure 6: Objectives Plot accompanied by PCA: main PCA plot depicts ideal class labelling

Next steps should involve projecting these altered datasets on to the "instance space" (Munoz et al. (2018)) to visualise changes in statistical and information theoretical properties. Further, by considering and applying this approach to the 300+ UCI binary classification datasets, it may be possible to create a complete picture that captures trends in changes as we optimise a given dataset for a particular algorithm. A robust set of altered data, and information of altered qualities would provide indications of strengths and weaknesses of specific algorithms beyond that which is known already. However, the computational time and complexity of such an undertaking was not manageable in the 6 weeks provided for the AMSI research program. Instead we consider a new method for generating test datasets from scratch.

5 Generating Test Data via Gaussian Mixture Modelling

5.1 Methodology

Gaussian mixture models are incorporated into the GA optimisation framework to generate test datasets from scratch. Rather than relying on an existing dataset and relabelling, a multi-step optimisation approach is considered where both an underlying dataset and a class label overlay are found.

Consider sampling a 500 observation dataset from a 5 mixture 5-variate Gaussian Mixture Model defined by θ:

$$p(\theta) = \sum_{i=1}^{5} \phi_i N(\mu_i, \Sigma_i) \qquad \theta = (\mu_{1,1}, ..., \mu_{5,5}, \sigma_{1,1}, ..., \sigma_{5,5}, \phi_1, ..., \phi_5)$$
$$\mu_{i,j} \in [0, 1000] \quad \sigma_{i,j} \in [0, 1000] \quad \phi_i \in [0, 1]$$

Here, ϕ_i refers to the probability an observation is made from the i_{th} multivariate normal governed by μ_i and Σ_i.

Sampling from such a distribution allows the generation of a 5 variable dataset, onto which an additional class attribute can be appended.

5.1.1 GA Optimisation Process

Let $\hat{\theta}$ denote the current "best" GMM parameter vector.

Let $\hat{\gamma}$ denote the current "best" class label vector.

Here, "best" refers to most optimal; using these estimates the difference in algorithm performance is at its highest (at the current iteration step).

We first initialise $\hat{\theta}$ and $\hat{\gamma}$ randomly.

Generate Data: Providing a class label vector as input, GA searches for a dataset that minimises the multi-objective fitness function. The dataset is found by **determining the optimal $\hat{\theta}$ GMM parameter vector for the input $\hat{\gamma}$** . Of course, sampling from a specified mixture is random, thus datasets are stored by seeding.

Generate Class Labels: Providing the optimal $\hat{\theta}$ from the previous step as input, GA **searches for $\hat{\gamma}$ to minimise** multi-objective fitness function.

This two-step process is repeated till fitness value improvement diminishes

5.1.2 Memory Strategy

There is a discontinuity between iterations of this two-step process that must be addressed. If, for example, the GMM sub-step has no knowledge of improvements made in previous GMM steps, it attempts to optimise a parameter vector from the 'ground up'. That is, it does not build on, or extrapolate from previous populations and is, in essence, a disjoint and independent optimisation task. In this way, a clear direction for optimisation is lost.

To bypass the problem, best members from a final GA population are stored , and reintroduced as members of the initial population during the next associated sub-step ('remembered and recalled'). To better elucidate the overall methodological framework, a diagram is provided:

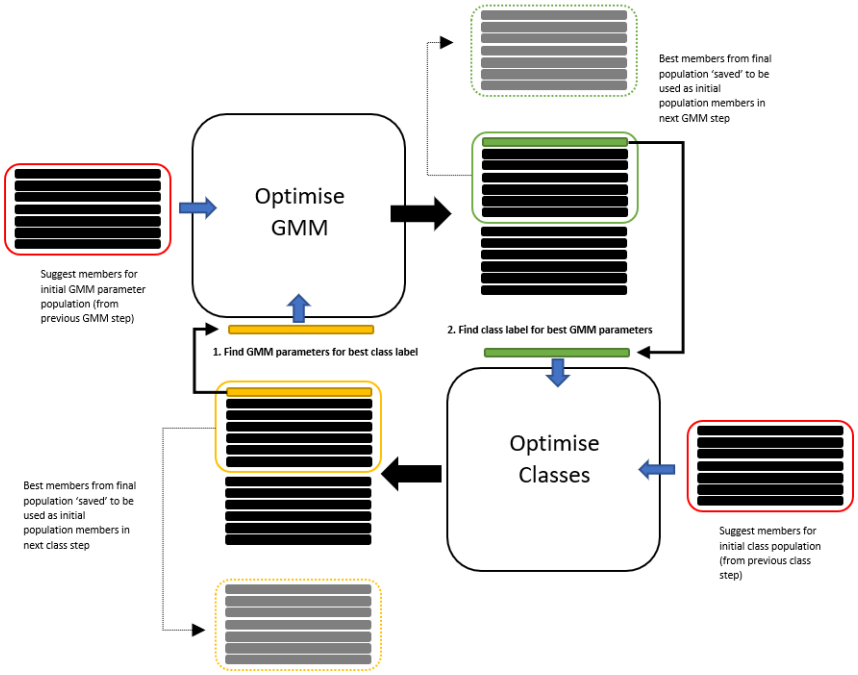


Figure 7: Multi-step Optimisation

5.2 Results

For the same pairs considered in the Haberman problem, the new GMM optimisation framework is applied. Generated datasets induced a difference in classification accuracy of 25% - 27% for all three pairs. It is likely that more discriminating datasets could be generated, however, GA was stopped at the 25% threshold as, by this point, comparable results to the initial Haberman exercise were achieved and the success of this approach was confirmed. Refer to 'Improvements' section for discussion on this matter.

For the pair, SVM radial vs. SVM Linear (performance of SVM Radial maximised), the generated dataset can be visualised as follows:

Dataset is rooted in the following ("best") GMM parameters:

$$\mu = \begin{bmatrix} 437.46 & 328.53 & 439.56 & 468.43 & 433.75 \\ 550.84 & 456.60 & 531.44 & 500.41 & 517.15 \\ 543.56 & 426.59 & 458.14 & 460.11 & 482.04 \\ 612.53 & 284.54 & 539.15 & 536.47 & 516.17 \\ 552.21 & 509.64 & 542.06 & 536.09 & 542.03 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 49.87 & 46.04 & 49.09 & 44.81 & 38.13 \\ 71.98 & 48.60 & 51.75 & 43.99 & 46.54 \\ 48.15 & 48.55 & 50.44 & 60.24 & 61.61 \\ 42.11 & 45.70 & 25.61 & 46.95 & 59.06 \\ 51.02 & 36.51 & 45.92 & 50.62 & 59.67 \end{bmatrix}$$

$$\lambda = \begin{bmatrix} 0.20 & 0.20 & 0.20 & 0.20 & 0.20 \end{bmatrix}$$

Visualising dataset:

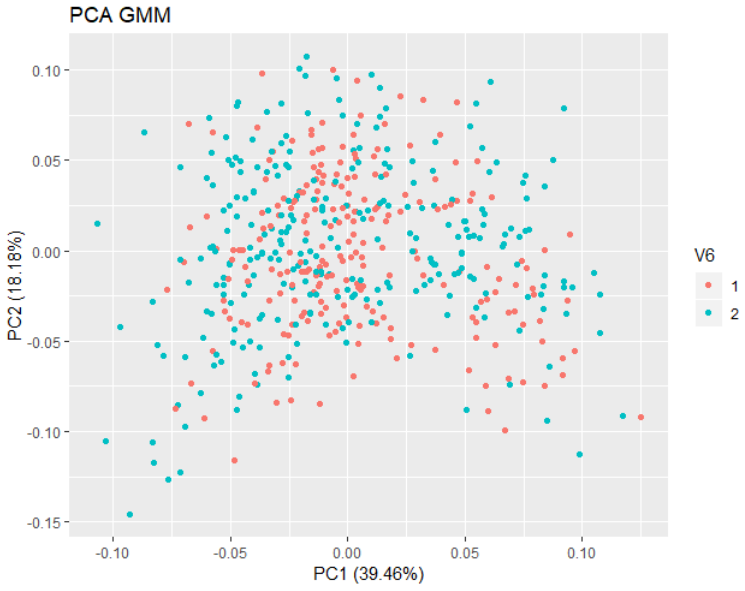


Figure 8: PCA of generated dataset

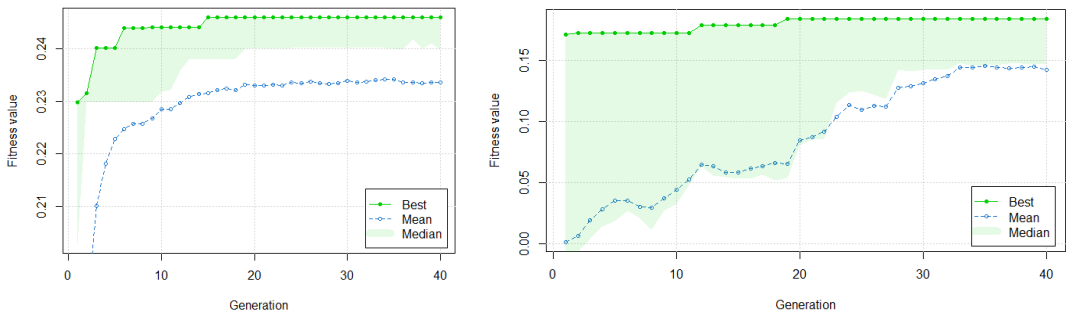
SVM Radial Accuracy : 76% | SVM Linear Accuracy : 50.1%

Difficulty of separating classes linearly is apparent, however nice radial class clusters are difficult to discern from the above visual. Could be because principal components are not very informative (39.46% and 18.19% variance) and high dimensional patterns not apparent in projection.

5.3 Improvements

5.3.1 Number of iterations between step switching

To begin each optimisation sub-step with a healthy initial population, we require that the final population from the previous associated sub-step be healthy. A healthy population, in this context, is one in which the average fitness value of the population is close to the best fitness value of the population. In this way, when a subset of the final population is chosen to be reenter the optimisation process as initial members in subsequent sub-steps, this subset is already, on average, of improved fitness, and optimisation in the new step can begin more smoothly.
However,in the current setup, each sub-step (GMM step and Class step) runs for 40 generations before switching to the next sub-step. This does not guarantee a "healthy" final population at the conclusion of a sub-step.



(a) Healthy optimisation

(b) Unhealthy optimisation

Figure 9: Changes in fitness value over 40 generations

Introducing variable iteration length dependent on mean and best fitness value levels during the GA process is a possible solution. Rather than a fixed 40 generation limit, GA could instead

be stopped once mean fitness values of a population are sufficiently close to the best fitness value.

5.3.2 GA Parameters

When optimising class labels during the "Class step", GA must optimise a bit string of length 500. Optimal population size, and number of generations, for high-dimensional parameter search is an active area of research, and this investigation, by no means, used ideal values for initialising NSGA-II. Further, it is likely that there are genetic algorithms better suited for the high-dimensional, mixed integer task at hand (Diaz-Gomez & Hougen (2007); Chen et al. (2015)). Finally, to handle the computational complexity of the task, cluster computing should be utilised.

6 Conclusion

A viable method for generating discriminating test instances has been demonstrated. Possible improvements to the methodological framework have been identified. Crucially, to gain new insights into algorithm strengths and weaknesses a larger space of these new test instances must be generated and analysed. The report discusses a successful approach by which to do so, however due to time constraints, does not establish a large cohort of optimised test instances. Creating and projecting such a set on to an instance space will allow algorithm performance to be understood at greater depth than what is currently afforded through existing repositories. For example, where in an instance space do datasets optimised for certain classification strategies tend to be? What does that imply about the properties of such datasets, and the behaviour of other classifiers in this region? Building a more complete instance space in this way can allow for refined analysis of the strengths and weaknesses of new and emerging ML algorithms. In this way, we create a valuable tool for stress-testing ML algorithms.

References

Chen, S., Montgomery, J. & Bolufé-Röhler, A. (2015), 'Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution', *Applied Intelligence* **42**.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002), 'A fast and elitist multiobjective genetic algorithm: Nsga-ii', *IEEE Transactions on Evolutionary Computation* **6**(2), 182–197.

Diaz-Gomez, P. & Hougen, D. (2007), Initial population for genetic algorithms: A metric approach., pp. 43–49.

Dua, D. & Graff, C. (2017), 'UCI machine learning repository'.
URL: *http://archive.ics.uci.edu/ml*

Lee, J. W. & Giraud-Carrier, C. (2013), 'Automatic selection of classification learning algorithms for data mining practitioners', *Intell. Data Anal.* **17**(4), 665–678.

Munoz, M. A., Villanova, L., Baatar, D. & Smith-Miles, K. (2018), 'Instance spaces for machine learning classification', *Machine Learning* **107**, 109–147.

Salzberg, S. (1997), 'On comparing classifiers: Pitfalls to avoid and a recommended approach', *Data Mining and Knowledge Discovery* **1**(3), 317–328.

Zhang, Q. & Li, H. (2007), 'Moea/d: A multiobjective evolutionary algorithm based on decomposition', *IEEE Transactions on Evolutionary Computation* **11**(6), 712–731.